PERSONALIZED NUTRITION AND OBESITY INTERVENTION SYSTEM

BY

FELYXIA LIM SIN NEE

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION SYSTEMS (HONOURS)
BUSINESS INFORMATION SYSTEMS
Faculty of Information and Communication Technology
(Kampar Campus)

JUNE 2025

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisors, Norazira Binti A Jalil who has given me this bright opportunity to engage in a mobile development project. It is my first step to establish a career in Mobile development field. A million thanks to you.

To a very special person in my life, Chee Fei Teng, for her patience, unconditional support, and love, and for standing by my side during hard times. Finally, I must say thanks to my parents and my family for their love, support, and continuous encouragement throughout the course.

COPYRIGHT STATEMENT

This Final Year Project proposal is submitted in partial fulfillment of the requirements for the degree of Bachelor of Information Systems (Honours) Business Information Systems at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project proposal represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project proposal may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ABSTRACT

Obesity is a condition characterized by an excess accumulation of body fat which is a major health problem prevalent in millions of people across the world and there is a consistent rise in prevalence in both adults as well as youths. In this project we propose the creation of a Personalized Nutrition and Obesity Intervention System which is investigated by providing the user with personalized diet options based on personal profiles in Mobile Application. This app is made with a primary focus on personalized nutrition, in contrast to many other apps that consider nutrition as a secondary feature. Some of its features include a menu that can be adjusted depending on the end user, features that allow users to monitor their health and features that help to promote engagement and continued use of the application. With the help of an integrated smart nutrition calculator in the mobile application, users may input personal information like age, weight, gender, activity level, and health goals to receive personalized recommendations for calorie intake. The dietary planner ensures that consumers may modify their meal plans as their health needs change by providing flexible customization based on dietary choices, allergies, and real-time feedback. There is also a diet and health option on the software where the user can record the food they take and other health aspects such as weight and Body mass index (BMI). Charts and graphs are used to show users daily activity and goals progress. Like in most mobile applications, motivational features such as daily check-ins, challenges, and point-based reward systems present in the app to enhance the users' experience. The basic features of such applications are returning points to the activity in hope of receiving vouchers or discounts to enhance retention. In addition, multilingual support ensures that the users with different languages and culture can easily access the application thus allowing it to be used by a global population of users. Java was the main programming language used in Android Studio. To ensure its reliability and scalability other tools such as Git version control and XAMPP control panel were used for backend services. The project addresses several of the fundamental limitations of current mobile health solutions, such as the difficulty of sustaining long-term good eating habits and the lack of customization in obesity intervention tools. Through the aid of the app, the target is to fill a gap in the market by offering specialized nutrition which will empower and encourage user to achieve their desired wellness and health in nutrients.

Area of Study (Maximum 2): Mobile Application Development, Personalized Nutrition System

Keywords (Minimum 5 and Maximum 10): Personalized Nutrition, Obesity Intervention, Dietary Customization, User Engagement, Mobile Application

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

LIST OF ABBREVIATIONS

XAMPP	Cross-Platform (X), Apache (A), MariaDB(M), PHP (P), and Perl (P)
BMI	Body Mass Index
BMR	Basal Metabolic Rate
UAT	User Acceptance Testing
MVP	Minimum Viable Product
ML	Machine Learning
AL	Artificial Intelligence
SDLC	System Development Life Cycle
API	Application Programming Interface
URL	Uniform Resource Locator
HTTP	Hypertext Transfer Protocol
QR Code	Quick Response code

CHAPTER 1 Introduction

In this chapter, the project presents the background and motivation of the research on a personalized nutrition and obesity intervention mobile application system. This project also discusses the contributions to the field and outline the problem statement.

1.1 Project Background

Obesity is a chronic complex disease whereby the excessive fat deposits in human body casing negative impact to human body health. A person who having this obesity disease, will have the possibility in getting the three high disease which is high blood pressure, high cholesterol and high blood sugar which is diabetes [1]. Abnormal hormones from obesity create decreased fertility in both men and women because of their effects on sperm production and ovulation [2]. Being overweight has many adverse effects on daily life activities including movement and sleep quality.

A Body mass index (BMI) analyses body weight status by comparing weight and height numbers. BMI tables determine weight status according to a person's age group and gender. A person with 25 BMI or higher falls into the overweight category and someone with 30 BMI or higher determines obesity according to the standards [1].

Obesity has become a major for concern in global health. As of 2022, some researchers have found out that one in eight people globally was having the disease obesity. They also discovered that 2.5 billion adults aged 18 and older were classified as overweight and 890 million of these adults were classified as obese.

The statistics reveal that 43 percent of adults face overweight conditions with obesity affecting 16 percent of these individuals. The young members of our population join adults in this weight problem. Researchers found that 390 million youngsters aged 5 to 19 are overweight and 160 million of these youth fulfil obesity criteria [1]. This data

shows clearly that new effective action is needed to control obesity and treat the health dangers it brings to people.

The approach of personalized nutrition presents an effective method to manage obesity by developing specific dietary advice that meets personal requirements and medical characteristics of individual patients. Contrasting standard food recommendations personalized nutrition makes use of DNA information alongside life choices and bacteria makeup and personal reaction data to construct optimal dietary solutions [3].

Research shows individualized dietary advice promotes people to follow balanced eating patterns better and helps them lose weight and minimize obesity complications due to personal nutritional variations and dietary preferences [3]. Personalized nutrition system can provide real-time or guidance to making it easier for user to adopt sustainable lifestyle change with the advanced technologies such as machine learning and data analytics.

New mobile technology becomes an important factor which strengthens the power of personalized nutrition. Users can conveniently access custom diet assistance through mobile applications at any place and at any time. Users who employ the applications will gain the ability to select nutritious meals based on evidence while receiving individual feedback for sustained healthy eating habits.

This report presents an approach for building a mobile application featuring individualized nutrition and obesity intervention methods. The application utilizes machine learning together with data analytics to deliver customized nutritional guidance as an approach to supporting sustainable weight management for improving health outcomes of users who fight obesity.

1.2 Problem Statement and Motivation

1.2.1 Problem Statement

Obesity continues to be a major worldwide health crisis while population rates of obesity increase within multiple demographic groups. Current health technology has made significant progress though interventions today do not manage individual requirements efficiently.

This segment identifies the fundamental challenges behind the project starting with restricted customization options and inadequate standalone applications and rising obesity statistics and unsustainable healthy eating behaviors while also specifying the initiative to create an individualized technology-based solution.

1. Limits in Customizing

In the research it found out that the current approaches to obesity intervention often rely on generic dietary guidelines that do not have the features customizing people's requirement, it is making a limit in a person choosing their dietary. For example, if a person has an allergy to some specific food, the generic dietary guidelines will be limited to the person in choosing their diet because the guidelines are not variable.

Most of the existing mobile applications only provide standard nutritional information without having the features in customization. So that the application will be unable to keep the users interested and unable to adherence the user in dietary plans.

2. Lack of Comprehensive and User-Friendly Standalone Applications for Personalized Nutrition

Besides, the research found out that there are very few approaches having the comprehensive tools that integrate personalized nutrition advice with a user-friendly interface. This shows the gap in the market having the opportunity to develop a mobile application that can personalize nutritional guidance and aids users in achieving and maintaining a healthy weight.

For example, most of the applications that have been created related to personalizing nutrition become a sub feature of fitness applications. Which most of the user download

the application mainly is for fitness guideline purpose not the personalize nutrition. Since these are sub features the developer will not focus more on this feature and the personalized nutrition feature will become less interactive and non-user friendly on the interface.

3. Increasing Obesity Rates

The high number of obesities has been rising globally over the last few decades, impacting not only adults but also kids and teenagers. On year 2022, showing the data that 2.5 billion adults aged 18 and older were classified as overweight, within this amount there are 890 million of them living with obesity [1].

Besides, the rise in obesity is not just limited to adults but also impacts on the younger population. In the research data it shows that 390 million children and teenagers aged 5 to 19 being overweight and within the population 160 million classified as obese [1].

A primary cause of this trend is the lack of personalized intervention approaches that can adjust to the different needs of various groups of people. Most of the mobile applications and obesity prevention programs currently in use are based on general dietary recommendations that do not take into consideration individual variations in genetics, lifestyle, and medical conditions [4].

For example, the application only requires the user to input their height and weight to calculate the user BMI and set a general diet base on the BMI given. This approach might not be effective for someone who has a genetic background to obesity or who lives in areas where access to healthy foods is limited.

Extremely difficult would be the coping with the problems that people of different ages face in the absence of offers, including children, adolescents, and adults of diverse cultures. Because of this, it's possible that these groups won't get the advice they need to properly avoid or control obesity, which will keep the worldwide obesity rate rising [4].

4. Difficulty Sustaining Healthy Eating Patterns

Another big challenge that a lot of people experience is the ability to sustain the good practices that are needed for eating habits for the long term. For instance, people are usually very keen and determined when it comes to adopting diets for change. However, it is often hard to maintain such changes continually [5]. This becomes a problem when dietary advice doesn't consider the requirements, interest, and lifestyles of the people in question.

The fact that many of the existing methods ignore the behavioral components of food modification is one of the main problems. To sustain these habits, a lot of people need more than absolute prohibitions concerning the food they consume. They also require constant encouragement, support and being made to realize that they are being held accountable for something. As was mentioned earlier, in the absence of individualized support, people may easily revert to their previous ways of consuming food which will lead to weight gain and inconsistent health effects [5].

Personalized nutritional advice is currently provided by mobile applications, but it is sometimes a secondary function within fitness-focused apps, which is another issue. As a result, the dietary advice that was given may not have been as comprehensive or exciting as the users need. Specifically, the complexity of usage of these features and if the users perceive them as basic, then people will likely engage in unhealthy food choices. Therefore, people are not motivated to engage with the app frequently, which impacts their usage frequency.

1.2.2 Motivation

These interrelated issues involving limited customization options together with insufficient standalone tools and rising obesity rates as well as unsustainable behavior practices demonstrate why an innovative strategy becomes necessary. The strategy exists to improve the success rate of obesity interventions through personalized dietary guidance that fits individual user needs. The proposed system addresses specific patient requirements such as allergic reactions and genetic predispositions to enhance

healthcare results and provide action-oriented recommendations beyond generic advice.

Higher market demand for health and wellness products stimulates this work. To deliver an independent mobile solution our project establishes itself by addressing market deficiencies while meeting user needs for applications that integrate naturally into their routines. The system achieves user investment in dietary plans through features which engage users as well as a simple interface to promote lasting good health practices. The main objective of our work strives to teach people the right methods for obesity control which will enhance both their lifestyle quality and public health programs fighting obesity at a global.

1.3 Research Objectives

1. To investigate how user-specific preferences, health metrics, and engagement features influence the effectiveness of personalized nutrition interventions for obesity management.

The purpose of the study is to determine the effectiveness of individual preference, health-related metrics, and engagement characteristics in predicting the success of personalised nutrition programs in obesity management. The study will examine the connection between user inputs in terms of allergies, target BMI, tracked weight and BMI data. Evaluation of motivation aspects, including the efficacy of day-to-day sign-in rewards and motivation hints depending on health progress, are also under consideration in the assessment. The compliance with dietary guidelines relies on the characteristics of such a program as plan customization and interactive devices, which increase the user adherence to their obesity management objectives.

The study performs a study of the key factors of concern to uncover key success factors that will streamline the individual development of nutrition approach in the management of obesity. The study involves quantifying the data of the users by assembling feedback of the User Acceptance Testing (UAT) with the system logs, backed by secondary data of the previous studies of the personalized nutrition and the

voucher reward. The aim aims to solve problematic concerns of current interventions on obesity such as limited customization and challenges in maintaining healthy behaviour. The aspects of engagement described in the literature review like the one in the apps like HealthifyMe reveal how interactive elements improve adherence, and the expected result will bring evidence-based results to improve the design of personalized nutrition and obesity intervention system to add to the existing user-centric obesity management options.

2. To develop a personalized nutrition and obesity intervention system that delivers tailored dietary plans for users with obesity or those requiring nutritional intervention, promoting sustainable weight management through a user-centric mobile application.

This objective focus on developing a mobile application which provides personalized nutritional support with tailored dietary plans designed for obesity patients along with restricted-dieters and individuals with health conditions. User-specific data consisting of health goals and dietary preferences together with allergies and age information as well as weight and height measurements and physical activity levels enables the system to produce sustainable weight management dietary plans.

Real-time dietary planning functions together with a smart nutrition calculator and a customizable dietary planner serve the user-friendly design of the app alongside tracking features for sustained user adherence to the program. The goal strives to complete a market deficiency for specific nutrition approaches by fixing standard nutritional guidelines along with insufficient self-contained dietary applications.

MainActivity.java serves as main logic along with backend services designed to run on XAMPP control panel through the development process that uses Android Studio and Java programming where developers will apply user feedback and iterative sprints throughout.

A motivational element within the system will feature advisory gameplay elements and reward structures following gamification concepts in literature. A complete mobile application emerges that helps users achieve health objectives with custom dietary solutions through accessible interfaces which leads to better obesity results and strengthened public health management.

3. To evaluate the usability of the mobile application on user adherence to personalized nutrition plans and its effectiveness in improving health outcomes related to obesity management.

The objective aims to demonstrate a performance evaluation of the mobile application's design along with its ability to help users maintain personalized nutrition plans and achieve obesity-related health benefits. The application interface design together with navigation and daily check-ins and multilingual functionalities will be assessed through user feedback during User Acceptance Testing (UAT).

The application tracks health metrics including weight and BMI and additional obesity-related indicators through its built-in diet and health tracking capability for effectiveness evaluation. The goal focuses on maintaining nutritious diets while addressing existing problems from inadequate user-centric nutritional support systems. Development of this section relies on research from the literature review which demonstrates critical roles of real-time feedback and user engagement in applications resembling "HealthifyMe".

Through this research the researcher will gain complete knowledge about the application while assessing its strengths alongside its weaknesses in helping patients achieve measurable health benefits such as weight loss or BMI reduction. Research findings will verify how the app contributes to obesity intervention programs along with identifying recommendations for future improvements which may inspire other digital health solution designers.

1.4 Project Scope and Direction

1.4.1 Project Scope

The mobile application project delivers individualized nutritional support to people facing obesity and those needing dietary assistance such as restriction dietary and weight goal seekers.

Core Functionalities

1. Dietary Planner and Customization

Based on the input and preferences of the user, the nutritional planner within the app will create customized dietary plans. Customers could make changes to their chosen meals and plans based on the customers' nutritional needs, preferences, allergies, or health goals. This feature ensures that the user gets to enjoy the meals and at the same time the meals are of equal benefit to the user's body.

Further, the features for diet customization depending on reactions of the user and the achieved results will be also integrated into the dietary planner. As a result of this, the users can easily change the recommendations with the new ones that will suit their requirements and preferences as they make their health goals.

2. Diet and Health Tracker

A daily diet record and a health monitoring system, which will allow the client to record his or her daily diet, concentration on weight, BMI, and other symptoms, will be a part of the application. The tracker's charts and graphs will allow the users to get a better perspective of their progress as well as assist them in improving motivation and comprehension of their health improvement process.

To provide valuable information and feedback for the users of the tracker, some additional tools set for goal achievements and controls such as muscles mass gaining or weight losing will also be part of the features included in the tracker. Also, this

feature is required in maintaining user engagement and to help people stay on track with their dietary and health choices.

3. User Engagement Features

Some of the activities will include challenges and daily signs in features that will ensure people keep opening the app. Such actions may help users in accumulating point which is utilised for vouchers or savings. The aim of this strategy is to make the user more enjoyable by rewarding something and in the process encourage users to keep on using the application.

For additional help in users' nutrition plan compliance, notification reminders and messages will also be included into the application. Alerts and status reports are going to be personalised and will help the users remain on track with their healthy habits. Meal notification reminders will be sent to the user to remind the user the take their meals on time and to avoid user having bad eating habits.

Technologies

The app will be developed using Java in Android Studio, with XAMPP control panel for backend services, Cohere AI to analysed user data and generate personalized dietary plans and Pexels API to generate the image base on the keywords of dietary plan. This technological framework ensures the reliability, flexibility, scalability and advanced in dietary analysis on the mobile application.

Target Audience

The system focuses on providing services to adults who have obesity along with people who need nutritional care and individuals with allergies or those with health requirements for weight control.

Deliverables

The initiative will result in a working mobile application together with analysis on factors affecting personal nutrition alongside usability ratings and health benefits

measurements for the application. The produced output serves public health by making obesity intervention tools available to users.

Exclusions

The project excludes features that integrate with fitness trackers and do not support genetic or microbiome data analysis or exercise tracking functionalities. The application runs exclusively on Android systems while excluding web and desktop versions.

1.4.2 Project Direction

The research objectives lead the project toward establishing a personalized nutrition system for investigating and development before evaluation to enhance obesity management outcomes.

1. Investigation Phase

The examination of behavioural factors that influence individual nutrition programs forms the focus of Objective 1. The user data collection during surveys and interviews in the data-gathering phase should be paired with existing study analysis. User insights will help the app's dietary planner deliver improved services to users while enhancing their overall adherence.

2. Development Phase

Creating the mobile application depends upon Objective 2 while using an iterative process which includes phases of two to four weeks. The design of the mobile application will prioritize user feedback for creating features that include real-time dietary planning and engagement aids. The analytical capabilities of Cohere AI generate specific dietary plans from user inputs thus closing the market void for such nutrition-related mobile applications. Efficient collaboration and updates will be achieved through using Git and GitHub for version control.

3. Evaluation Phase

User Acceptance Testing (UAT) of the app will determine its effectiveness and usability through testing the interface design and engagement features and tracking weight-related health metrics obtained from the app's tracker. During this stage the system will demonstrate its capability to promote user adherence and generate health enhancements by resolving the issue of maintaining long-term healthy dietary choices.

4. Innovation and Impact

The project presents an independent nutrition-focused app that differs from fitness apps and utilizes Cohere AI for customized recommendations. This tool helps users develop better dietary knowledge which leads to reduced public health risks for obesity. The successful evaluation results can serve as an inspiration for future digital health innovations because the project adds value to the development of digital healthcare solutions.

1.5 Contributions

1.5.1 Academic Contribution

With the help of this project, a personalised nutrition algorithm that is specific to each user's profile will be introduced. In other words, the planned messages will include age, gender, BMI, food preference and allergies and medical conditions of the user will be used to present meal planning and nutrition recommendations.

This algorithm will adjust to the needs of the person, compared with standard methods that provide general guidance, guaranteeing that the recommendations are applicable and useful [3]. Higher user engagement and diet plan adherence are planned as the outcome of such levels of personalisation, which will eventually improve health outcomes. The given algorithm is going to be built with the aid of data and feedback

from users which means that it is going to improve in the course of time, making it more accurate and efficient.

1.5.2 Technical Contribution

A mobile app based on Cohere AI provides technical innovation to analysed user data for producing dietary plans that match individual needs. The autonomous application separates itself from other apps by giving nutrition top priority while featuring individual dietary planning and a flexible meal planning system and health monitoring functions.

Projecting a high-quality mobile app with user experience-focused perspective is one of the major innovations of the project. The application will possess an easily navigable user interface which will enable the users to interact with the program. To maintain engagement of the customers certain activities such as individual feedback, real time updates on the progress made and an integrated meal selection tool will be incorporated seamlessly.

Ease of use will be given the most importance in the design so that even users with little technical knowledge may take advantage of the program. The program tries to keep users interested over time by emphasising the user experience, which is important for maintaining good eating habits and reaching weight control goals.

1.5.3 Market Gap Fulfilment Contribution

With the creation of a standalone app that concentrates entirely on personalised nutrition, this project closes a significant market gap. This app will focus on giving the user detailed, engaging nutritional recommendations, not like other systems that view nutrition as an insignificant feature of huge fitness apps [4].

The app will provide a more targeted and interesting experience, offering specifically to consumers looking for individualised nutrition recommendations. In addition to setting itself apart from competitors, the project's resolution of this unfulfilled

requirement offers it a chance to grab a small market of concerned about their health people who place a high value on nutrition as part of their aim for wellness.

1.5.4 Contribution to Public Health

By offering easily available personalised obesity intervention options, the project has the potential to have a major influence on public health. Obesity trends are increasing across the world and it therefore important to discover ways that the public can embrace readily [1]. This is the reason this mobile application aims to meet the need of easily accessible, friendly and evidenced-based professional advice on personalised nutrition.

The program is helpful in preventing obesity related diseases and improving the general wellbeing of the community by availing an interface through which users can make proper diet choices. Besides, this application's success can influence other digital health programs, and how doctors and developers address obesity and associated health issues.

1.6 Report Organization

The "Personalized Nutrition and Obesity Intervention System" mobile application development along with its evaluation emerges from this Final Year Project 2 (FYP2) report. A summary of the report organization follows which demonstrates the seven-chapter structure and reference list and appendix segment that support investigation of behavioural factors and system development and effect examination.

Chapter 1: Starts by discussing project history and define problem areas, set objectives, and explore organizational sections to explain why users need a customized nutrition application.

Chapter 2: Reviews technologies used and existing systems like Foodvisor, Cronometer, and HealthifyMe to establish needed solutions and validate the project foundation.

Chapter 3: Describes the methodology, requirements, architecture, and timelines for FYP1 and FYP2.

Chapter 4: Presents system design with flowcharts, use cases, and ERD.

Chapter 5: Details implementation, setups, operations with screenshots, issues, and remarks.

Chapter 6: Evaluates the system through testing, results, challenges, objective assessment, and concluding remarks.

Chapter 7: Provides conclusion and recommendations for future work.

References in the last section use the IEEE style to properly cite academic sources.

Appendix supplementary materials with code samples and project poster.

CHAPTER 2 Literature Review

2.1 Previous Works on Personalize Nutrition Algorithm

Personalized nutrition is an emerging approach of providing dietary advice that is individualised based on certain parameters. Previous research was focused on combining several elements like nutrition, lifestyle, microbiota, and genetics. This field of algorithms aims at enhancing the odds of good nutrition by assessing an individuals' body's unique reaction to types of foods.

2.1.1 Implementing Machine Learning and AI in Personalized Nutrition

The latest trends, including machine learning feature, in recent developments analyze large data sets and improve the nutritional recommendations and customization. Research indicates that personalised nutrition, as compared to standard dietary recommendations, can result in superior health outcomes [6].

It is therefore has had significant impact in the field of personal nutrition with the help of machine learning (ML) and artificial intelligence (AI) in analysing large amount of data for personalising diet. By applying these technologies, it is possible to integrate data of genetic origin, biometric data, and the data regarding the dieting habits to create personalized nutrition plans that may include objectives such as, for instance, increase of nutrient uptake or the management of the body weight [6].

This trend is most effectively shown by AI-driven apps like Foodvisor, which utilise complex algorithms to assess food intake and deliver users personalised nutritional advice [7]. Foodvisor is a multilingual nutrition app accessible in French, English, Spanish, and German on Google Play and the App Store [8].

It was developed in France. Showing in Figure 2.1 with the features like meal and calorie tracking, nutrient adjustment, hydration tracking, and weigh-ins, it helps users achieve many kinds of health goals, including weight loss, muscle growth, and maintaining body weight.

Figure 2.1 Calories tracking, calories calculate and activity tracking.

Besides Foodvisor also having another feature, in Figure 2.1 shows the image recognition technology using the AI and machine learning to enable users to snap pictures of their meals and instantly obtain nutritional information [7]. The AI models of the app can help users record their diets and indicate different foods along with Calories & Nutrient values. It would be difficult for people to stick to their diet and health objectives without this real-time analysis's quick feedback along with customised advice [8].

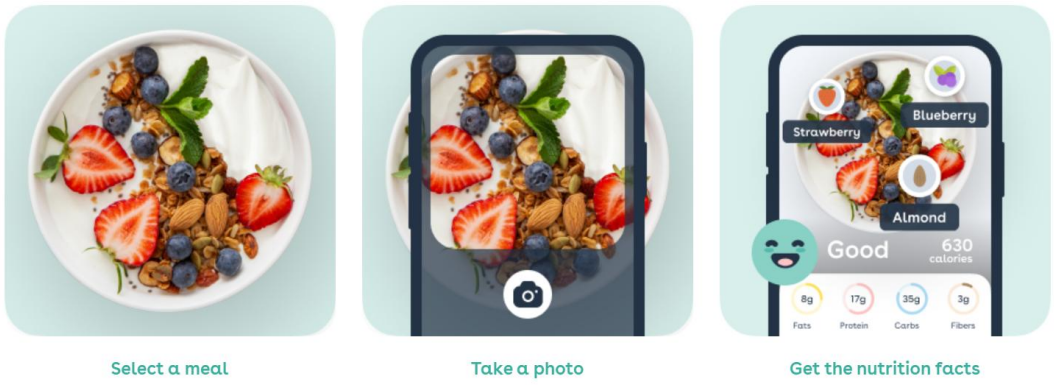


Figure 2.2 Snap Pictures to get the nutrition information.

2.1.2 Behavioral Aspects of Diet and Weight Management

Lifestyle factors impact diet and weight control because they influence consumers' ability to adhere to dietary guidelines and execute healthy lifestyle choices [5]. Applying these behavioural features, mobile applications like as Foodvisor gather personal data in-depth, such as height, weight, lifestyle choices, food preferences, and emotional components that have shown in Figure 2.2. The app's algorithm can produce highly customised guide based on each user's specific health goals and challenges given to this wide range of information.

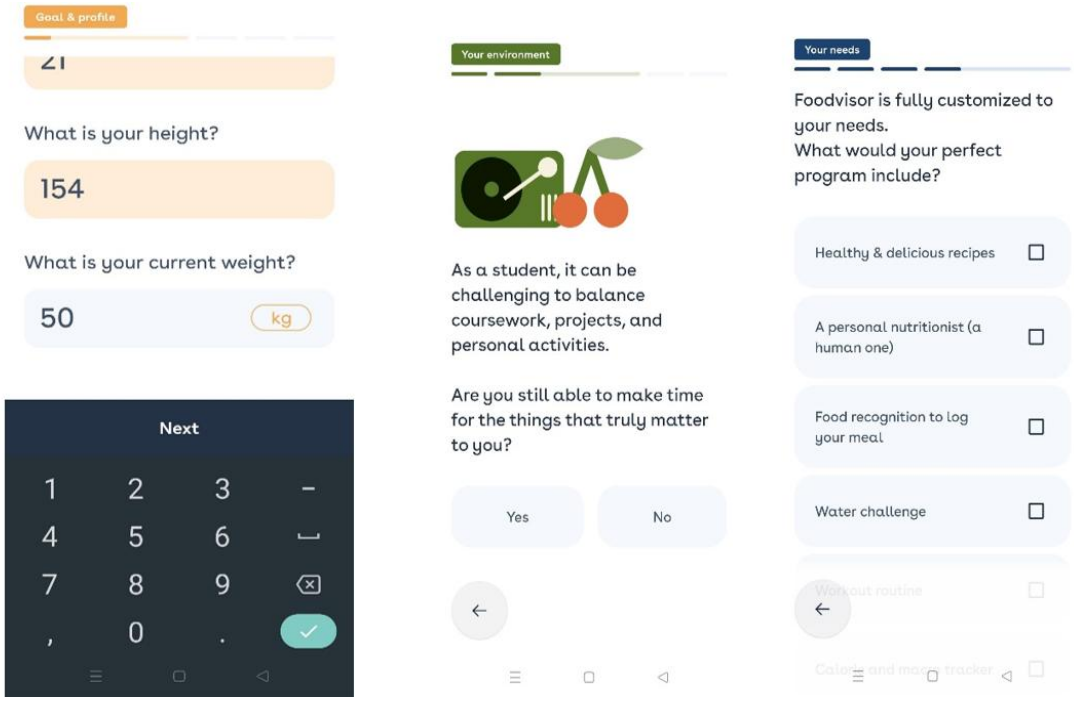


Figure 2.3 Foodvisor application getting the information of user's goal, environment, and needs.

Foodvisor is unique in that it requires consideration of a wide range of health-related criteria, such as past dieting experiences, exercise routines, and emotional moods, as along with standard diet plans [8]. The app provides a more individualised and successful dietary management experience by helping users match their food intake with their unique health goals with a focus on macro and calorie tracking. This rich approach to the concept of nutrition is complemented by the app's utilising artificial intelligence and machine learning: the technology provides users with real-time feedback and suggestions to maintain a diet that fits their goals [8].

Similar to this, Cronometer goes well beyond many other applications' important focus on calories and macronutrients by tracking up to 84 nutrients, providing a thorough and extensive approach to personalised nutrition. Since its launch in 2005, Cronometer has developed into an effective resource for anyone wishing to carefully analyse and enhance their eating habits [9].

In Figure 2.3, Cronometer with the ability to work with multiple dietary features and connectivity with activity monitors, it's especially useful for people with specialised

nutritional requirements, including athletes or people managing long-term medical illnesses. With the help of the app's comprehensive feature set, which includes the ability to track fasting and create personalised reports, users may better understand their diet and health and make decisions that will benefit their general well-being [10].

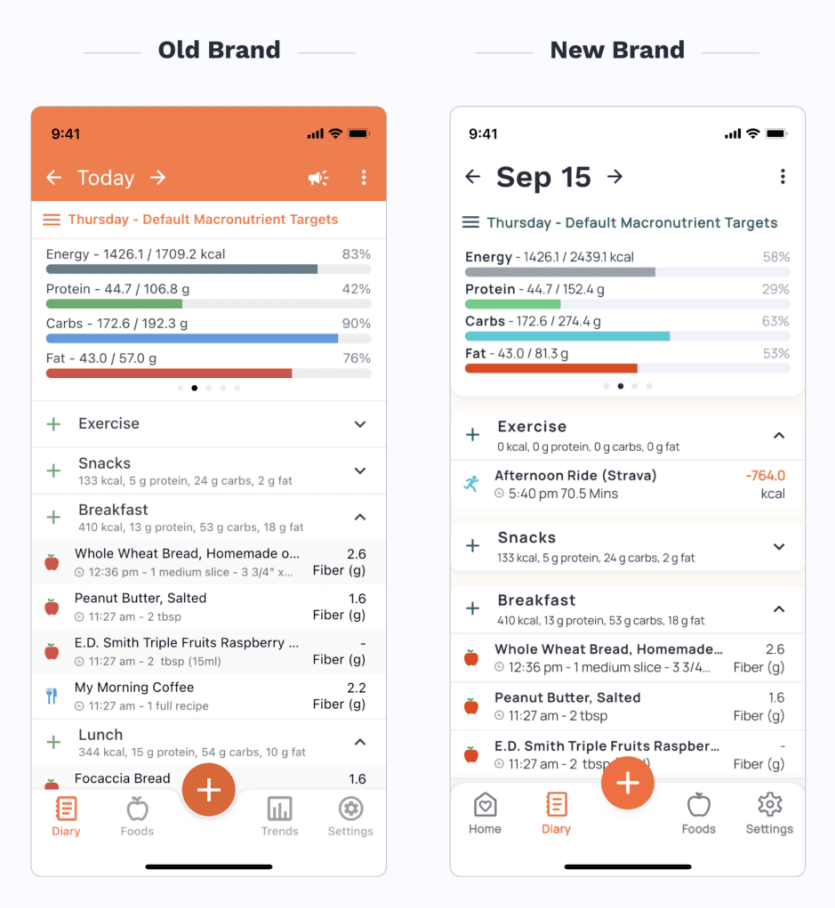


Figure 2.4 Daily dietary activity report from Cronometer application.

2.1.3 Wearable Technology in Enhancing Personalized Nutrition

By providing real-time data on sleep, physical activity, and other health variables, wearable technology has transformed personalised nutrition [11]. Devices such as fitness trackers and smartwatches are designed to capture a user's activities comprehensively enough and it becomes possible to estimate individual's health need. Wearables monitor heart rate, sleep habits, and physical activity levels continually to assist customise dietary suggestions to a user's objectives and lifestyle [11].

The use of wearable technology in personalised nutrition is best demonstrated by HealthifyMe. In Figure 2.4 The software gathers information about glucose level and physical activity by connecting to a variety of wearables [12]. Adaptive goal settings and personalised meal plans are then produced using this data. For instance, HealthifyMe may change the calorie plan and add more protein or any nutrient that will help a user have the best recovery and performance if there is an indication that the user is more active.

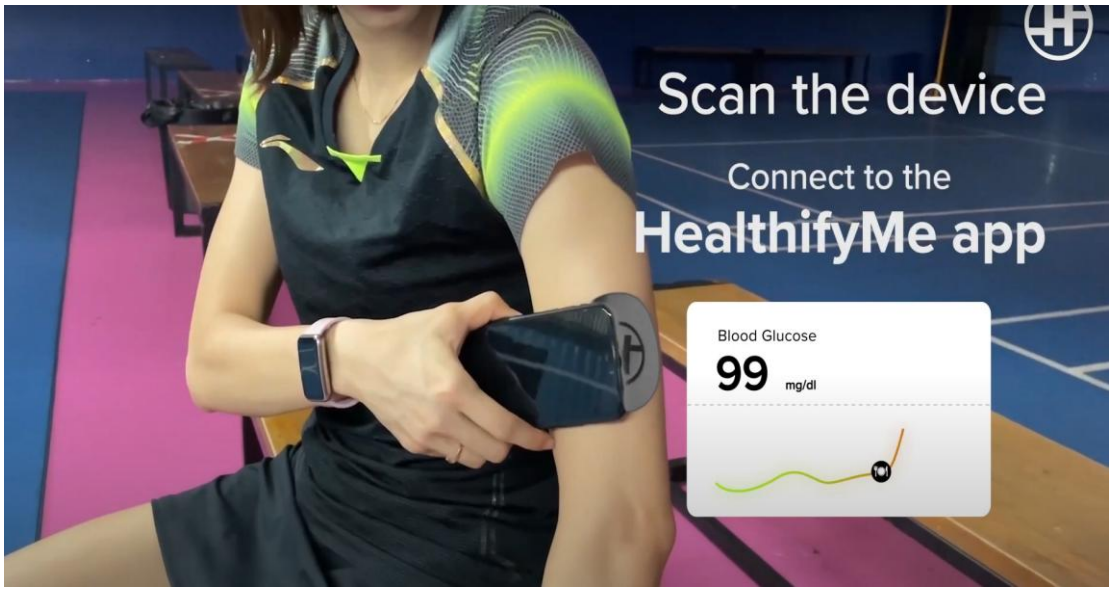


Figure 2.5 Wearable technology from HealhifyMe application.

All things considered, the combination of wearable technology and nutrition apps such as HealthifyMe improves the effectiveness of customised nutrition plans. These apps can offer more focused recommendations based on the consumers' specific health data in real-time, which inspires better health [13]. In Figure 2.5, with the real-time health data, a coach will give a quick feedback on time to users, the ability to chat with coach will let users more understand their progress and what should do for the next step. Further integration will be made between wearables and nutrition applications as time passes; better strategies will surface allowing the effective monitoring of health indices.

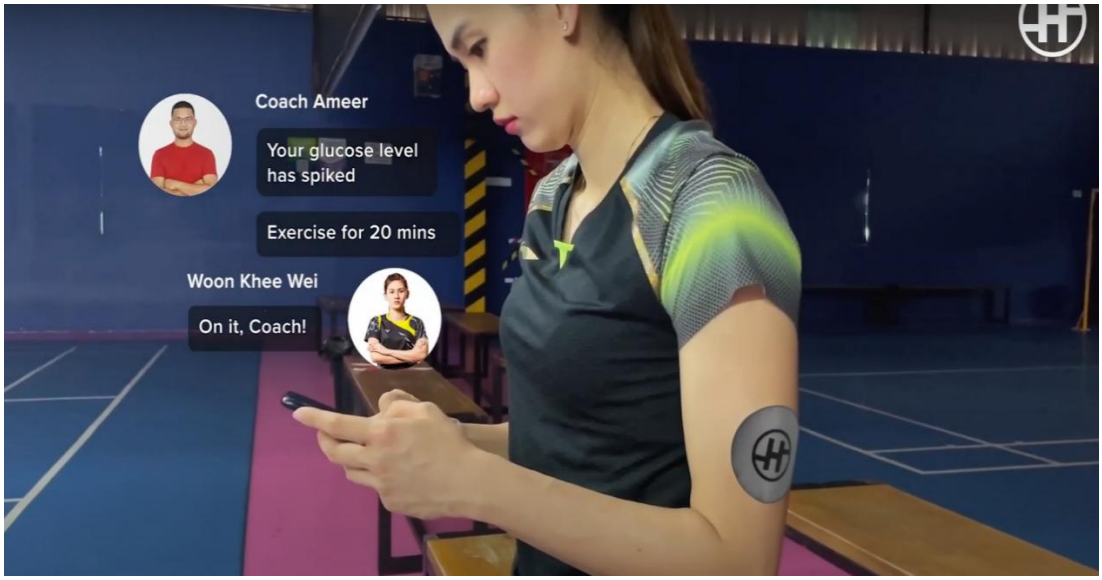


Figure 2.6 Real-time feedback from coaching.

2.2 Limitation of Previous Studies

2.2.1 Restricted Access to Comprehensive Features for the Application

The restricted access to comprehensive features, which are frequently hidden behind paywalls or subscription models, is a major issue of many personalised nutrition applications [4]. There are fewer developers choose to spend time on developing these apps because personalised nutrition technology is still in its relatively narrow market. Hence, the public may have less choice, and their evolution and support may require more subscription-based economic models.

Thus, the most advanced features which are necessary for comprehensive and personalised dietary advice are frequently exclusive to subscribers who can afford to pay for them. For instance, in Figure 2.6, the subscription offered by Foodvisor which costs around MYR 129.99 every three months which is equivalent to MYR 43.33 per month offering such features as personalised nutrition consultancy, and detailed meal plan .In contrast, the app's free version only offers basic features such like provide only basic calorie and macro tracking, hydration logging, and limited meal logging capabilities [8].

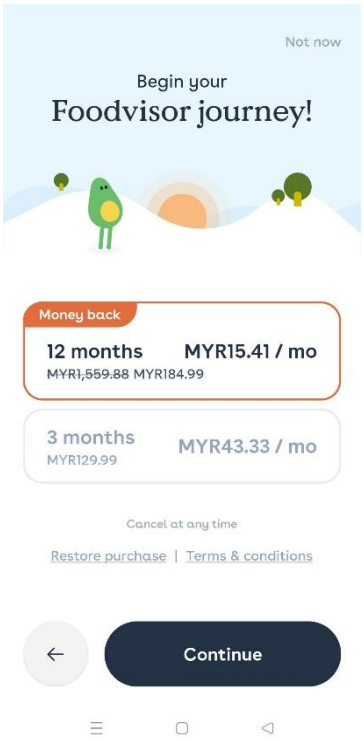


Figure 2.7 Subscription price for Foodvisor application

To the same extent, in Figure 2.7, Cronometer has a free trial version that has many restrictions on the nutrient tracking ability. However, features such as custom reports and more detailed nutrients logs and fast-mission tracking are part of the gold package which goes for approximately MYR 47.99 per month in MYR 264.99 per year [10].

The paid tier of services also poses a problem for those who cannot afford the premium services to get the best from the application. A similar business strategy is followed in HealthifyMe, it cost MYR 46.30 to MYR 69.45 per month or, the Malaysian ringgit equivalent of MYR 276.43 to MYR 461.25 annually for the individual nutrition programs and one-to-one sessions depending on different plans customers select [12].

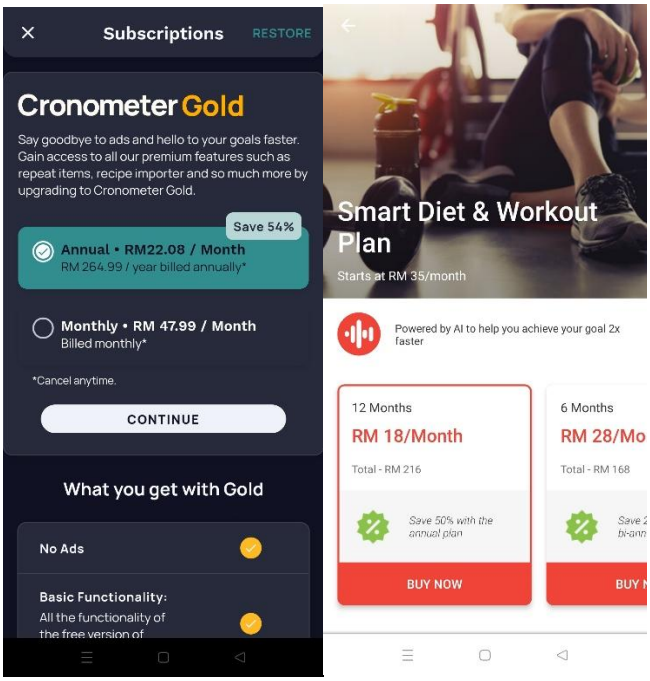


Figure 2.8 Premium subscription price for Cronometer and HealthifyMe application

People may also reduce their stake and attention in these apps because of the rising cost of using these applications to their full potential. If the costs for the membership are too much for them, the users can quit using the app before they can achieve maximum usage of the app's function and their dietary goals. Besides, the cost factor restricts the potential of the app for users who cannot buy these extra services, contribute to the decrease in the adoption of personalised nutrition applications.

This extra layer of difficulty just increases the challenge of obtaining significant user groups with other needs, such as athletes or people with chronic ailments including obesity or diabetes. It's significant that personalised nutritional products are already less common. Some cost issues remain and if solved together with increase of access to necessary features, personalized nutrition tools may be more effective overall and so retain users.

2.2.2 Limitations of AI and Machine Learning in Personalized Nutrition

Specific apps such as HealthifyMe, Foodvisor, and Cronometer have incorporated artificial intelligent and machine learning to a certain degree and there are many

disadvantages. A key issue that we identify in managing these platforms is the continual maintenance of data quality. To offer customized solutions, several of these apps utilize the entered data by the user in the form of food diaries, daily physical activity, and even biomarkers.

Nonetheless, with users inconstant input entry, inconstant user involvement, and low-quality input data the AI imperative dietary suggestions may be less specified. For instance, users may forget to input certain meals or even wrongly input the meals they have consumed and therefore the AI algorithms used in those apps may provide recommendations that do not tally with the nutritional needs of the user [8].

A successful deployment of AI in these personalised nutrition applications is limited tremendously by privacy issues. To personalise the results, Foodvisor, Cronometer and HealthifyMe request users to input their personal health data. This includes factors such as weight, exercises, genetics possibly and even dieting habits apart from the meals taken. This data is very private and raises several privacy concerns especially with regards to its storage and processing in the event of a security breach [6].

The ability of the app to give similar extremely precise and individualized suggestions could be hampered by the limitations which people put in terms of figures they are willing to reveal with developers or other third parties. Still, for these platforms to gain and retain users' trust, top-notch data protection measures are a prerequisite [6].

Moreover, the complex data sources such as microbiome and genetic data cannot be fully integrated by the AI algorithms now in use in these applications. Even though all the three apps which is Foodvisor, Cronometer, and HealthifyMe offer analysis based on the data collected, they still lack in offering concise and accurate nutritional advice.

Perhaps the scope of the data to be integrated into these platforms may expand in the future given the improvement in the AI algorithms hence giving customized and accurate results. But to achieve this integration, considerable progress must be made in the still-developing fields of data processing and algorithmic capabilities [6].

In conclusion, while having great potential for the future, AI based PN including Foodvisor, Cronometer, HealthifyMe cannot grow now since they are overstepping consumer's privacy boundaries and have questionable data accuracy. Thus, applying such technologies, it is possible to increase the effectiveness and adaptivity of diet methods to the profiles of the users.

2.2.3 Real-Time Tracking Progress

This issue poses a severe challenge to the development of personalised nutrition applications due to the realities of real-time data monitoring. While wearable technology has allowed an app like HealthifyMe to monitor the user's health parameters constantly, this approach is still constrained.

Besides membership fees, there is a revelatory necessity for extra charges for the acquisition of wearable gadgets which creates financial barriers or may limit the user from employing the service to their full potential. The deployment of real-time tracking may be affected by this additional cost, especially among users who stand to gain the most from it and those who manage long-term illnesses or seek out specific health gains [11].

Furthermore, even while wearable technology has the capacity to capture data in real time, its efficacy depends on regular use and specific synchronisation. Sometimes in the monitoring process, there can be a few shortfalls because users feel that setting up the process is complicated or they experience issues with data consolidation [11]. Therefore, the effectiveness of personalised dietary recommendations may be limited by potential weaknesses of data collected.

The lack of real-time data tracking is another issue for apps of the kind and necessity of which Foodvisor and Cronometer demonstrate when patients need to input their dietary and medical information. This may be challenging and error-prone manual approach may affect the accuracy of recommendations given by AI algorithms in these apps. That is why the need for the applied high technologies, which enable automation and optimisation of the data collection process, stems from the use of user-entered data.

Conclusively, wearable technology and AI integration has a lot of possibilities in personalized nutrition due to its real-time data monitoring, but it lacks associated with higher costs and possible inaccuracy of data. For personalised nutrition apps to be of more benefit and for more people to be able to afford and use them these factors must be enhanced.

2.3 Proposed Solutions

2.3.1 Broadening Access to Comprehensive Features

Having a more inclusive freemium model is the next step to eliminating the challenges of a lack of access to feature-rich personalised nutrition programs. This model might offer of a wider important feature at no cost in the present model such as the advanced nutrient tracking and simple meal planning hence making the program to appeal to many people without medical conditions.

In addition, instead of subscribing to a whole package of services that can be found in a premium package, a micropayment structure can be initiated whereby the user pays for only the extra services that he or she wants [14]. As a result, it enhances the application's functionality and makes it more accessible to the public thus reducing on costs which user must incur.

Furthermore, it may be possible for the utility of implementing programs on incentives to be useful in improving user loyalty and participation. Through the frequency of using the app or referring friends, the users can get access to the premium option. Beside the practical consequence of dismantling the financial barrier, this approach also increases the level of users' participation and generates the feeling of togetherness.

Personalised nutrition applications can be more effective in a wider group of users if its availability will be increase and the methods of payment will be more varied. This will make the users happy and sticky [14].

2.3.2 Enhancing with AI tool for Personalized Nutrition

To enhance the data reliability by adding more precise data validation mechanisms and informing end-users about a correct data input approach, personalised nutrition applications can enhance the AI-driven recommendations. Through these measures user's input data is ensured to be correct thus the usefulness of diet AI recommendations produced is increased.

Privacy issues should also be tackled using strong data encryption and anonymously techniques, thus ensuring that the application safely handles sensitive information from the users [15]. Incorporating options like a "privacy mode", with user's options of which data is processed and retained could also help affirm and encourage sharing of information need for processing and offering individualized recommendations.

Second, the personalized dietary recommendations can be made more accurate by signal update a person's dietary pattern database with inputs from other reliable and updated sources such as genetics and microbiome. Although it may be the case that present day's technology has its confines, simple basic genetic information could pave for extended nutrition guidance as the capacity of AI expands.

Integrations with specialized businesses or making some of these features paid for those who want to use this application may enhance the app's effectiveness even further. Personalised nutrition applications have the potential to improve user experience and outcomes by offering more dependable and customised health solutions by emphasising data quality and privacy [16].

2.3.3 Cost-Effective and User-Friendly Real-Time Tracking

By partnering with developers of reasonably priced wearable devices, we can lower the cost barrier that comes with real-time tracking through wearable technology and increase user accessibility to these tools [17]. Forcing the customers to buy additional products they don't necessarily need will not only not help improve the app's functionality, but it will also be inspiring more people to make purchases by either offering more appealing discounts or creating attractive packages.

Moreover, enhancing the easiness in the setting up or synchronizing these devices in the app will reduce the level of frustration of the users in using the real-time tracking feature for better utilization of personalization and data collection [17].

Another couple of ways that could make a user experience better when automating the data entry operations, such as using barcodes or voice logs, are getting rid of the possibility of making mistakes while entering a lot of data and, at the same time, speeding up the process of data gathering.

Further, recorded data might be used to provide users immediate, meaningful information based on real time feedback loops [17]. It keeps users interested in this and makes them adhere to their health goals. Real-time monitoring can be made cheaper and therefore more attractive by expanding the potential user engagement of personalised nutrition apps.

2.3.4 Enhancing User Engagement and Retention

The problem of low user engagement can be solved through personalised nutrition apps that may have specially developed sections, where users can monitor their progress and be encouraged to proceed with their positive activities. Those benefits which motivate the users to remain loyal to the scripts, could look like badges, discounts for the paid options, and customized congrats messages.

Additionally, the concept of making accountability as well as creating an identification through the users whereby utilizing the forums, challenge groups or social connections, makes the users continue being engaged and active towards their health personalities [18].

It is also important to reduce the user's demand fatigue, and this can also be eliminated by constantly updating the contents of the app with new challenges, recipes and instructional materials. Users of the application can therefore be retained for long-term, and they must be always satisfied by ensuring that the application offers what is relevant

in the market or at some time, important to users. They enhance the general retention and success rates since the tactics do not only enhance the quality of the user experience, but they also change the probability of the user's likelihood to continue using the app to achieve their wellness and health objectives [18].

2.4 Summary

Table 2.1 Comparison features of existing application

Feature	Foodvisor	Cronometer	HealthifyMe
Basic Features	Calorie and macro tracking, hydration logging, limited meal logging	Calorie, macro, and up to 84 micronutrient tracking	Calorie, macro tracking, and activity tracking
AI-Powered Meal Recognition	Use image recognition technology to analyze meals and provide nutritional information	Not available	Not available
Wearable Integration	No direct integration	Integrates with devices like Garmin and Fitbit	Integrates with multiple wearables, e.g., Fitbit, Google Fit, smart scales, continuous glucose monitors (CGMs)
Personalized Diet Plans	Available through premium subscription	Available through premium (Gold) subscription	Personalized meal plans based on data from wearables
Nutrient Tracking	Tracks calories, macronutrients, and basic micronutrients (limited in free version)	Tracks up to 84 micronutrients	Tracks macronutrients and integrates with wearables for customized suggestions based on user activity levels
Premium Subscription Cost	MYR 43.33 per month (premium features like personalized coaching and	MYR 47.99 per month (advanced features like custom reports, fasting tracking, advanced	MYR 46.30–69.45 per month (personalized programs, one-on-one coaching,

	advanced meal planning)	micronutrient tracking)	depending on the plan)
Custom Reports	Provides basic meal summaries (premium required for detailed reports)	Detailed nutrient and biometrics report available with (Gold) subscription	Provides activity and glucose level-based feedback, real-time coaching and personalized reports
Real-Time Feedback	Real-time nutritional insights based on AI meal recognition	Not available	Real-time feedback via coaches and data from wearable devices
User Engagement Tools	Gamification through badges, challenges; real-time feedback; community interaction	Basic progress tracking and reporting, no community features	Chat with health coaches, receive rewards and progress feedback
Educational Content	Limited education content available for free users, more in-depth with premium	Extensive food database with nutrient information	Detailed nutrition and health education, workouts, recipes, and dietary tips based on user data

The comparison table have highlighted that the Foodvisor in AI-powered meal recognition, providing instant feedback by analysing meal photos. Then Cronometer pays more focus to nutrient tracking and analysis including micronutrients which are not available in other applications. Lastly, HealthifyMe incorporates wearable technologies as well as in-app real-time coaching whereby the program develops a plan of action based on the data input from users' daily activity and biometric.

CHAPTER 3 Proposed Method/Approach

The development process of the Personalized Nutrition and Obesity Intervention System mobile application receives detailed explanation in this chapter. The approach integrates methodologies to develop requirements alongside design elements and architectural frameworks as well as scheduling standards to fulfil user needs regarding customizability and interaction and reachability.

3.1 Development Methodology

The mobile application development adopts the System Development Life Cycle (SDLC) structure with Agile Scrum methodology for supporting user-led iterative development needs while enabling flexible collaboration.

3.1.1 What is Agile Scrum

Agile Scrum functions as an iterative and incremental software development life cycle framework which puts collaborative engagement at its core together with flexible delivery and produces functional software in rapid fashion. Each product development cycle runs for 2–4 weeks as part of the short sprint process which yields usable product segments. Various essential phases form the core of Agile Scrum methodology when handling a project.

The SDLC framework of Agile Scrum consists of stages which begin with Product Backlog Creation followed by Sprint Planning and Sprint Backlog Creation then enters the Work on Sprint portion and culminates with Testing and Product Demonstration and reaches its conclusion with Retrospective and Next Sprint Planning.

User feedback enhances the design of the dietary planner and health tracker features via Agile Scrum implementation since the project follows an iterative process for feature development. Figure 3.1 shows the Agile Scrum SDLC Diagram presents the iterative

cycle which demonstrates that sprints create advances through planning steps, execution, review and retrospective stages.

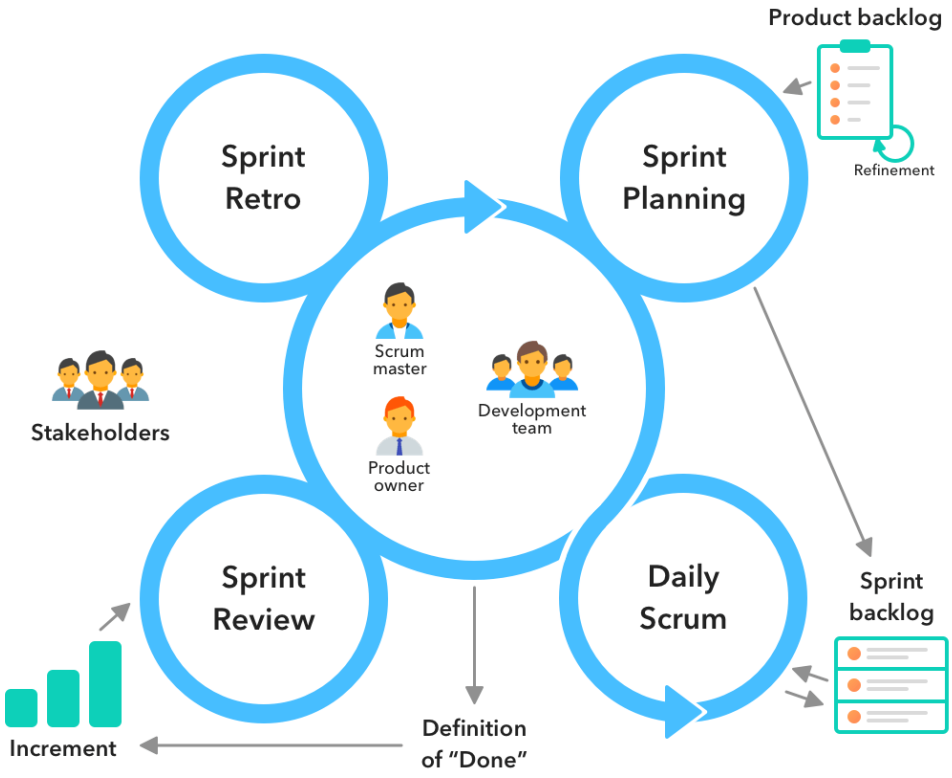


Figure 3.1 Agile Scrum Flow Diagram.

3.1.2 Why Agile Scrum

Agile Scrum represents the best choice for implementing the Personalized Nutrition an Obesity Intervention System because it fits with the project's dynamic requirements that focus on user needs. During sprint iterations the system delivers little-by-little updates that include designing user interfaces with prototyping alongside establishing backend structures and implementing Cohere AI platforms and designing dietary planner forms and health tracker charts to satisfy Objective 2 application development needs.

User feedback occurs during sprint reviews and User Acceptance Testing (UAT) activities which allow users to provide comments about prototype improvements for

navigation simplicity and user customization according to Objective 3 requirements. The flexible nature of the Product Backlog enables project development by accepting new requirements which include multilingual support alongside fresh engagement features that support the project's innovative nature.

Agile Scrum maintains excellent collaboration standards by implementing GitHub standups which helps project components like mobile UI and backend together with AI components merge into modular increments to deliver technical quality. The structured sprint approach of Agile Scrum provides better performance than rigid development methodologies because it integrates prototyping for UI validation which results in superior development of mobile applications that require quick user-driven development.

3.1.3 Comparison with other SDLC Methodology

The selection procedure for Agile Scrum includes comparing it against Waterfall and Prototyping through analysis of criteria which align with project demands to build systems and address unclear user needs and unfamiliar technology while accounting for complexity and compressing timelines.

Table 3.1 Comparison of SDLC Methodology

Criteria	Agile Scrum	Waterfall	V-Model
Ability to Develop System	Excellent	Good	Good
With Unclear User Requirement	Excellent	Poor	Poor
With Unfamiliar Technology	Good	Poor	Poor
Complexity	Good	Good	Excellent
With Short Time Schedule	Excellent	Poor	Poor

Ability to Develop Systems: Agile Scrum shows exceptional capability in developing systems by organizing iterative sprints to create functional prototypes that consist of backend APIs and UI prototypes as well as Cohere AI integration and Pexels API image which supports achievement of Objective 2. The stable and well-defined project requirements make Waterfall and V-Model suitable methods, yet these frameworks challenge dynamic user-centric development that needs continuous platform changes.

With Unclear User Requirement: The Agile Scrum process shines at handling unclear user needs because sprint evaluations and sprint development lead to user need clarification. The fixed requirements of Waterfall and V-Model create challenges for managing unclear and changing user needs.

With Unfamiliar Technology: The iterative approach together with testing and prototyping in Agile Scrum allows teams to utilize unknown technologies such as Cohere AI efficiently and Pexels API image. Waterfall and V-Model suffer from restricted ability to perform experiments since they need well-defined technology specifics.

Complexity: Implementation of the V-Model on complex systems remains excellent because it strengthens each development stage through thorough testing procedures. Waterfall and Agile Scrum operate well for managing system complexity although Agile Scrum modules and Waterfall phases work better than V-Model testing protocol.

With Short Time Schedule: The 28-week timeline makes Short Time Schedule possible because Agile Scrum divides work into sprints that bring rapid delivery results. Waterfall's structured phase sequence along with V-Model's exhaustive testing procedures slow down development which makes them problematic for brief deadlines.

3.1.4 Agile Scrum Stages and Project Activities

The Personalized Nutrition and Obesity Intervention System was developed in the Agile Scrum style and included three FYPs (Weeks 15-28), each week 1-14 of the project was divided into FYP 1 (Weeks 1-14). Every stage had 2-week sprints to provide incremental functionality, which gave it flexibility and responsiveness to user feedback.

The Agile Scrum phases, such as Product Backlog Creation, Sprint Planning, Sprint Execution, Sprint Review, and Sprint Retrospective, were used to develop the main functionality, such as user authentication, BMI and BMR calculator, custom diet planning with the suggest, shuffle, and custom features, health tracker, voucher earning system, and options such as text size and profile management. Each of the stages is described in the following, representing what was done during the 28-week period and integrating the innovations of FYP 2, including the voucher system and integration of Cohere AI.

Product Backlog Creation

The Product Backlog was created at the beginning of FYP 1 with the help of user stories, which were gathered and prioritized to determine the requirements of the Personalized Nutrition and Obesity Intervention System. The following user stories identified fundamental features, including enabling users to add dietary preferences, track weight and BMI using charts, and interact with inspirational features, including voucher redemption. The examples of user stories are the following: As a user, I would like to define my eating limitations to get personalized meal plans, and As a user, I would like to see my weight progress through graphs.

Since the 28-week limit is 14 weeks per stage, non-core features were pushed to the backburner to have the Minimum Viable Product (MVP) consisting of diet planning, tracking, and user engagement tools. Product Backlog FYP 2 Product Backlog was revised according to the feedback of FYP 1 testing, with new requirements, i.e. the

voucher reward system and improved AI-based diet shuffling using Cohere AI. The new Product Backlog entailed the prioritization of specific work items such as the development of the voucher redemption module and the integration of AI-driven diet plans shuffling, which guarantees the alignment of the work to the user-driven goals and customization requirements throughout the 28-week period.

Sprint Planning

Every 2-week sprint started with the choice of the user stories in the Product Backlog to develop a detailed Sprint Backlog with particular tasks. In FYP 1 (Weeks 1-14), sprints were devoted to the elements of the basic components, including developing UI mockups of the dietary planner, developing the XAMPP MySQL back-end with PHP APIs, and user authentication. As an example, Sprint 2 (Weeks 3-4) included such tasks as developing UI prototype of dietary planner and creating a user profile table in MySQL, whereas Sprint 3 (Weeks 5-6) was devoted to the implementation of the logic of a BMI calculator.

During FYP 2 (Weeks 15-28), the sprint planning switched to advanced features and refinements, such as work on such items as Integrate Cohere AI API for diet suggestions (Sprint 8, Weeks 15-16), Develop voucher redemption UI and backend (Sprint 9, Weeks 17-18), and Improve text size adjustment functionality (Sprint 11, Weeks 21-22). A Sprint Backlog was created in every single sprint that defined modular work to make certain that there was consistent progression in the 28-week program, balancing core functionality under FYP 1 with improvements such as rewards and AI-inspire personalization under FYP 2.

Sprint Execution

During the sprint implementation, the development team worked on the scheduled increment within two weeks, where daily stand-ups were used to organize activities and resolve problems. In FYP 1, the focus was made on developing Android Studio Java prototypes of the UI, set up of the XAMPP MySQL database, and preliminary API links

with user data management. Indicatively, Sprints 1-4 (Weeks 1-8) supplied with login or registration screens, the BMI calculator and simple diet planning interfaces.

In FYP 2, the execution was aimed at the incorporation of selected advanced functionalities, including Cohere AI to create individualized meal plans in ViewEditPlanActivity.java, the implementation of the voucher system in VoucherActivity.java and the improvement of the tracking with the use of graphical outputs in WeightHistoryActivity.java. Everyday stand-up was used to tackle such problems as API response time optimization of Cohere AI and user data and rewards data scalability. Android Studio testing and physical device testing were done to verify compatibility and APIs were verified using Postman. The increments that were created were UI prototypes, database tables, AI-generated meal plans, and voucher redemption functionality, which would form a strong MVP by the conclusion of Week 28.

Sprint Review

The project supervisor present at the end of each sprint reviewed the completed increment to give feedback on the functionality and usability. Sprint review In FYP 1, UI prototypes and backend components were presented. Suggestions like simplify input form navigation, improve chart readability were added to the Product Backlog that could be worked on later in the sprint.

The reviews in FYP 2 were dedicated to the new features, including the voucher redemption interface and AI-based diet plans. An example is Sprint 10 (Weeks 19-20), which introduced the voucher system, with feedback such as adding more voucher expiration information to the system resulting in the refinements. These reviews helped to ensure that we cover the objective of usability and customization as per the objective of providing the user-centric MVP within the 28-week schedule. The feedback in the Product Backlog was prioritized systematically to lead to the iterative improvements of both FYP 1 and FYP 2.

Sprint Retrospective

The sprint retrospective was a chance to check what the processes of every sprint are, and how it can be improved in order to increase the efficiency during the timeframe of 28 weeks. In FYP 1, the retrospectives were working towards estimation of tasks to optimize them according to the 2-weekly sprints and enhance collaboration during the development of UI. As an example, Sprint 3 (Weeks 5 -6) detected time wastage in setting up the database, which resulted in improved resource allocation in future sprints.

In FYP 2, the perspectives sought to resolve such issues as API bottlenecks with Cohere AI and optimized testing processes to facilitate User Acceptance Testing (UAT). Technical quality was improved by introducing process changes like the use of automated testing tools as API validation, and improved branching strategy in Git. These improvements made the development process to be highly agile and efficient that would help deliver a functioning MVP at the end of Week 28, with such features as personalized diet planning, weight tracking, and user engagement tools being entirely achieved.

3.2 System Requirement

3.2.1 Hardware

The hardware involved in this project is computer and android mobile device. For the creation of the mobile application that will be used in the personalized nutrition program, both development and testing devices were used to enhance versatility. Big project files may be handled and Android studio may run fine with the setup physical power and space.

Other smartphones and tablets were also used for testing with different screen size, operating system including Android 10 and Android 11. The variety made that the program worked and was suitable with a wide range of real-world situations. To improve development and accelerate workflow, extra devices including external monitors and keyboards were also used.

Table 3.2 Specifications of laptops

Description	Specifications
Model	Victus by HP Gaming Laptop 15-fa2xxx
Processor	13th Gen Intel Core i5-13420H
Operating System	Windows 11
Graphic	NVIDIA GeForce GT RTX 4050 Laptop GPU
Memory	16GB RAM
Storage	477GB

3.2.2 Software

- Android Studio Meerkat (2024.3.1): IDE for Java coding.
- XAMPP Control Panel (v3.3.0): MySQL, PHP APIs.
- Cohere AI: Dietary recommendations.
- Pexels API: Generate Dietary Image
- Git (2.49.0), GitHub Desktop: Version control.
- Android SDK: UI, networking, testing libraries.

Android Studio, which is an android official integrated development environment for developing android applications, was used to develop the software base of the project. The application was coded, debugged, and tested in an easy-to-use environment using Android Studio Meerkat | 2024.3.1. version and Java were the key programming languages employed, which meant that the engineers were able to build the application's features.

These other tools and frameworks were included in the Android SDK to support different functionalities including XAMPP control panel for back-end services and Cohere AI for analytics. The Pexels API tools help in providing the dietary image to improve the system interface. Moreover, for managing version control the Git tool was being employed and the repository to store the project was GitHub, ensuring the code to be original and not copied. Espresso was employed for the execution of unit tests and UI tests respectively while these frameworks measure the reliability and effectiveness of the program.

3.3 Project Timeline

3.3.1 Timeline – FYP 1

Activity	Period													
	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14
Sprint 1 Investigation														
Conduct investigation into obesity intervention challenges and existing nutrition apps														
Define project scope and objectives														
Gather initial user stories														
Create Product Backlog with prioritized requirements														
Sprint 2 Literature Review														
Perform literature review on personalized nutrition systems and obesity management.														
Analyze technologies														
Refine user stories based on literature findings.														
Set up development environment														
Sprint 3 Proposed Approach														
Design UI mockups for login, registration, and dietary planner														
Define database schema for user profiles														
Propose system architecture														
Document proposed approach for supervisor review.														
Sprint 4 Proposed Approach														
Develop user authentication module														
Create MySQL tables for user profiles.														
Implement initial PHP APIs for user data														
Validate approach with supervisor feedback.														
Sprint 5 Proposed Approach / Development (Initial)														
Develop BMI/BMR calculator using Harris-Benedict formula														
Create UI prototype for diet planning														
Conduct unit testing for authentication and BMI modules.														
Update Product Backlog based on prototype feedback.														
Sprint 6 Development (Initial)														
Implement weight tracking with basic graphs														
Perform usability testing on prototypes with a small group.														
Refine UI based on feedback														
Sprint 7 Development (Initial) / Evaluation (Preliminary)														
Conduct initial User Acceptance Testing (UAT) on authentication and BMI modules.														
Document FYP 1 findings and challenges														
Update Product Backlog for FYP 2, prioritizing advanced features.														
Prepare FYP 1 report and presentation.														

Figure 3.2 Gantt Chart of FYP 1 (Week 1-Week14)

3.3.2 Timeline – FYP 2

Activity	Period													
	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14
Sprint 8 Development														
Integrate Cohere AI API for personalized diet suggestions														
Develop PHP APIs for diet plan storage/retrieval														
Test AI API integration for response accuracy and speed.														
Create database tables for diet plans														
Sprint 9 Development														
Implement voucher reward system														
Create MySQL tables for rewards														
Develop PHP APIs for voucher management														
Conduct unit testing for voucher functionality														
Sprint 10 Development														
Enhance diet planning with Custom mode and macro visualizations.														
Implement activity tracking with advanced progress graphs														
Optimize database queries for performance														
Sprint 11 Implementation														
Deploy enhanced UI for diet planning and voucher redemption.														
Implement text size customization for accessibility														
Test system on physical Android devices for compatibility.														
Address UI feedback														
Sprint 12 Implementation / Evaluation														
Conduct comprehensive User Acceptance Testing (UAT) with beta testers.														
Collect feedback on usability														
Implement fixes for identified issues														
Validate system performance across devices														
Sprint 13 Evaluation														
Perform final bug fixes and performance optimizations														
Conduct end-to-end testing to ensure database and API integrity.														
Validate all features														
Document testing results and user feedback														
Sprint 14 Evaluation														
Finalize documentation, including user manual and technical report.														
Conduct final supervisor review and incorporate feedback.														
Prepare FYP 2 presentation and final deliverables.														

Figure 3.3 Gantt Chart of FYP 2 (Week 15-Week 28)

CHAPTER 4 System Design

4.1 System Flowchart Diagram

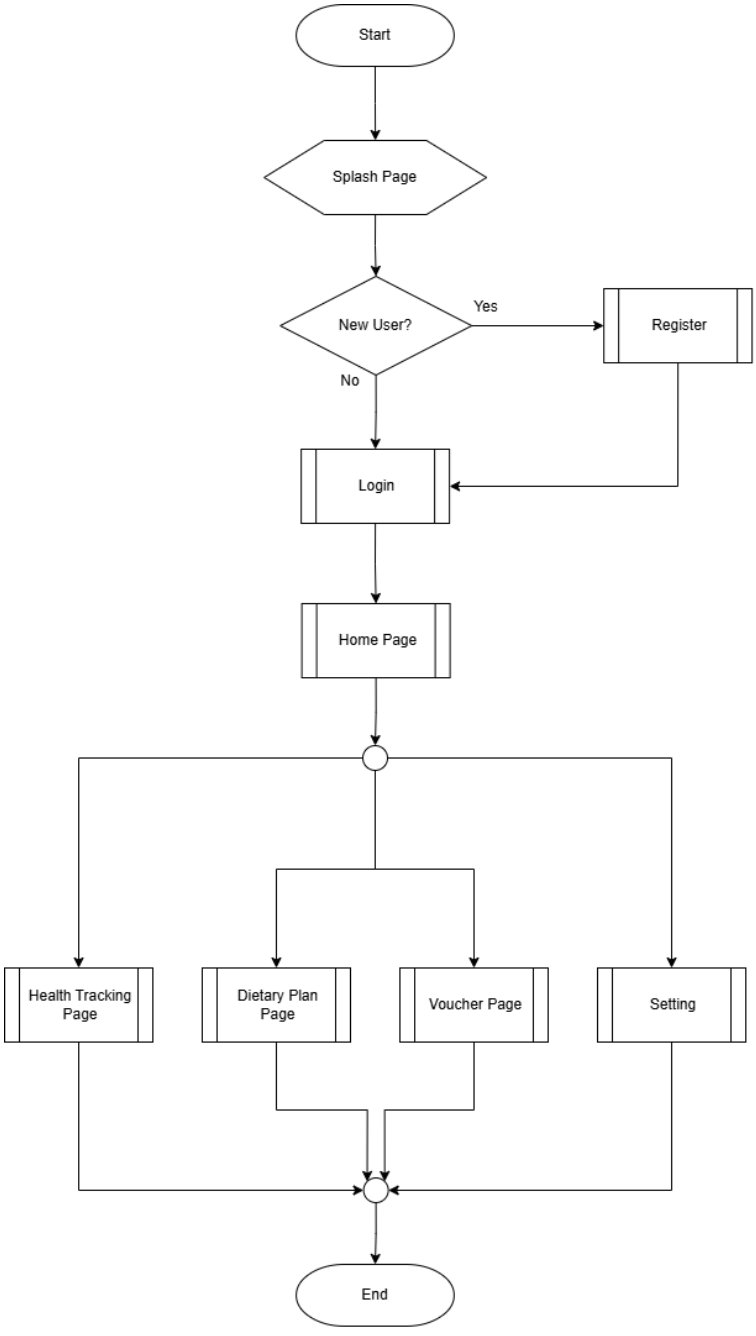


Figure 4.1 Flowchart of whole system module on current work

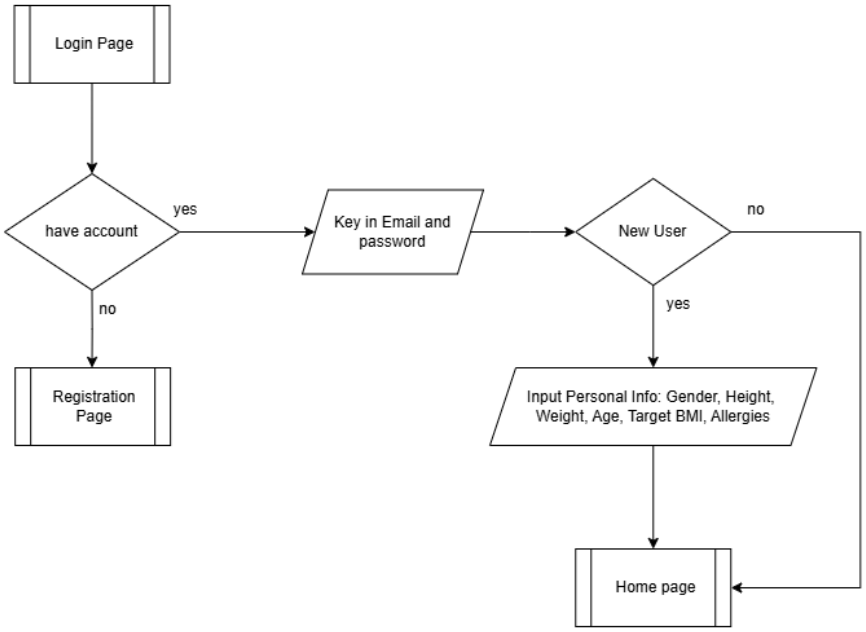


Figure 4.2 Flowchart of Login Page

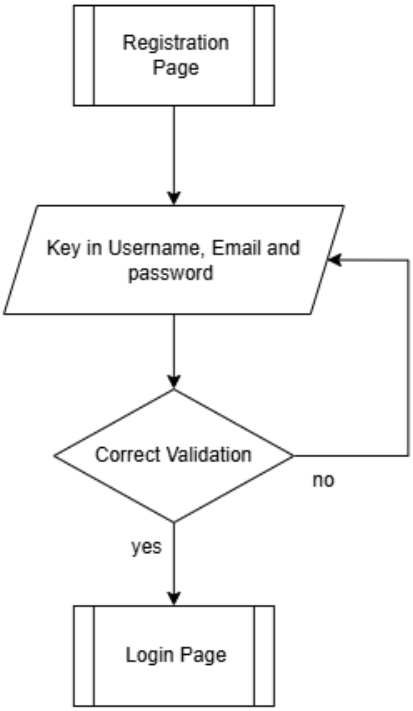


Figure 4.3 Flowchart of Registration Page

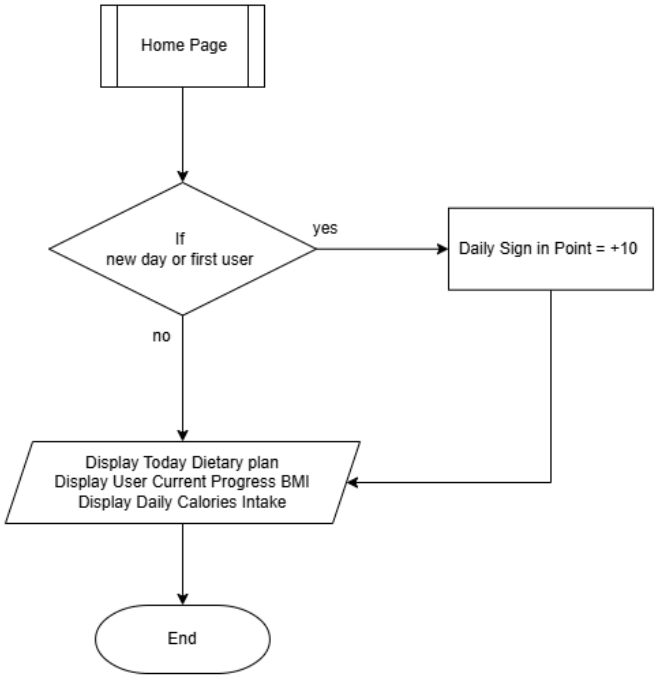


Figure 4.4 Flowchart of Home Page

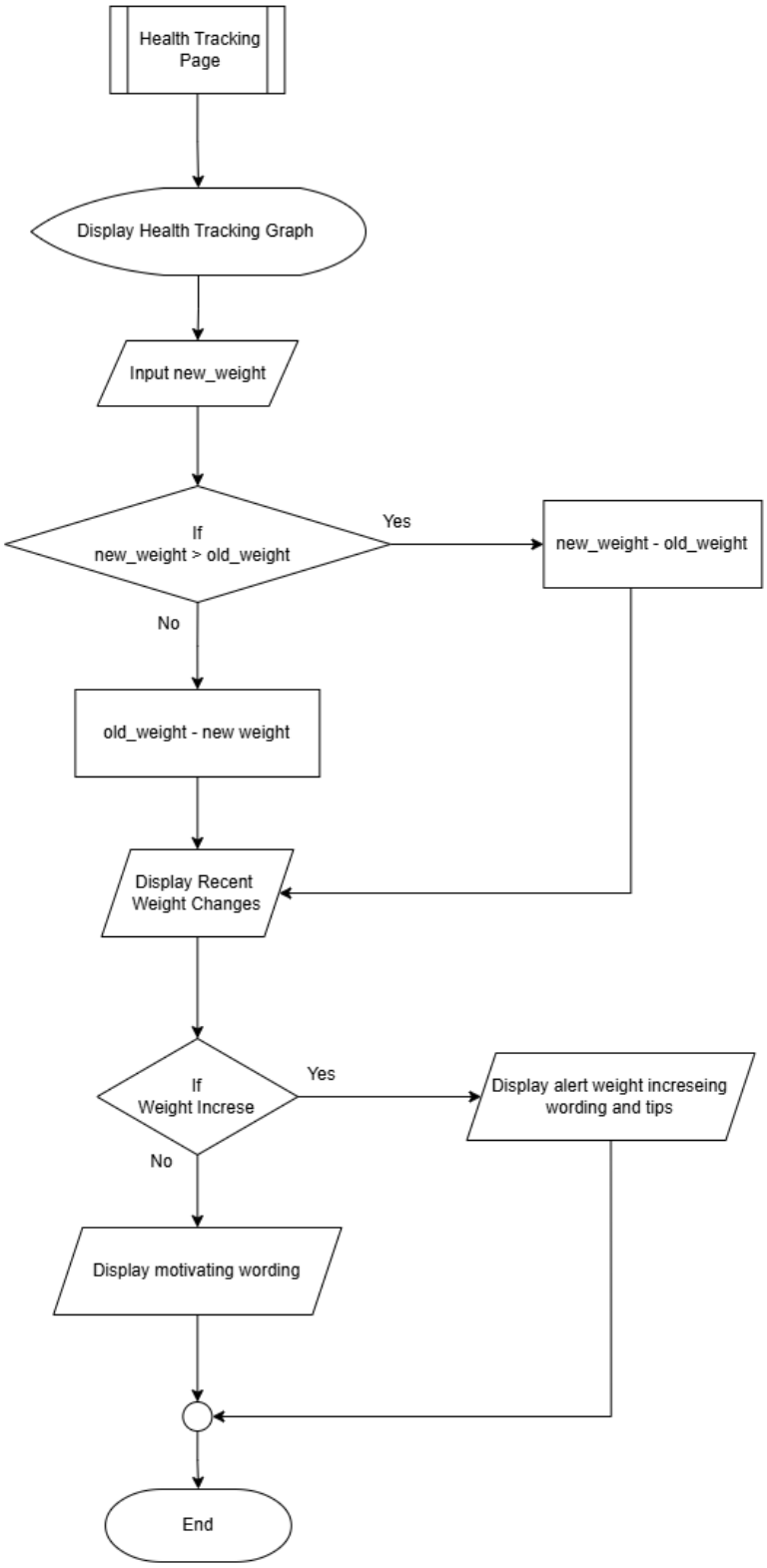


Figure 4.5 Flowchart of Health Tracking Page

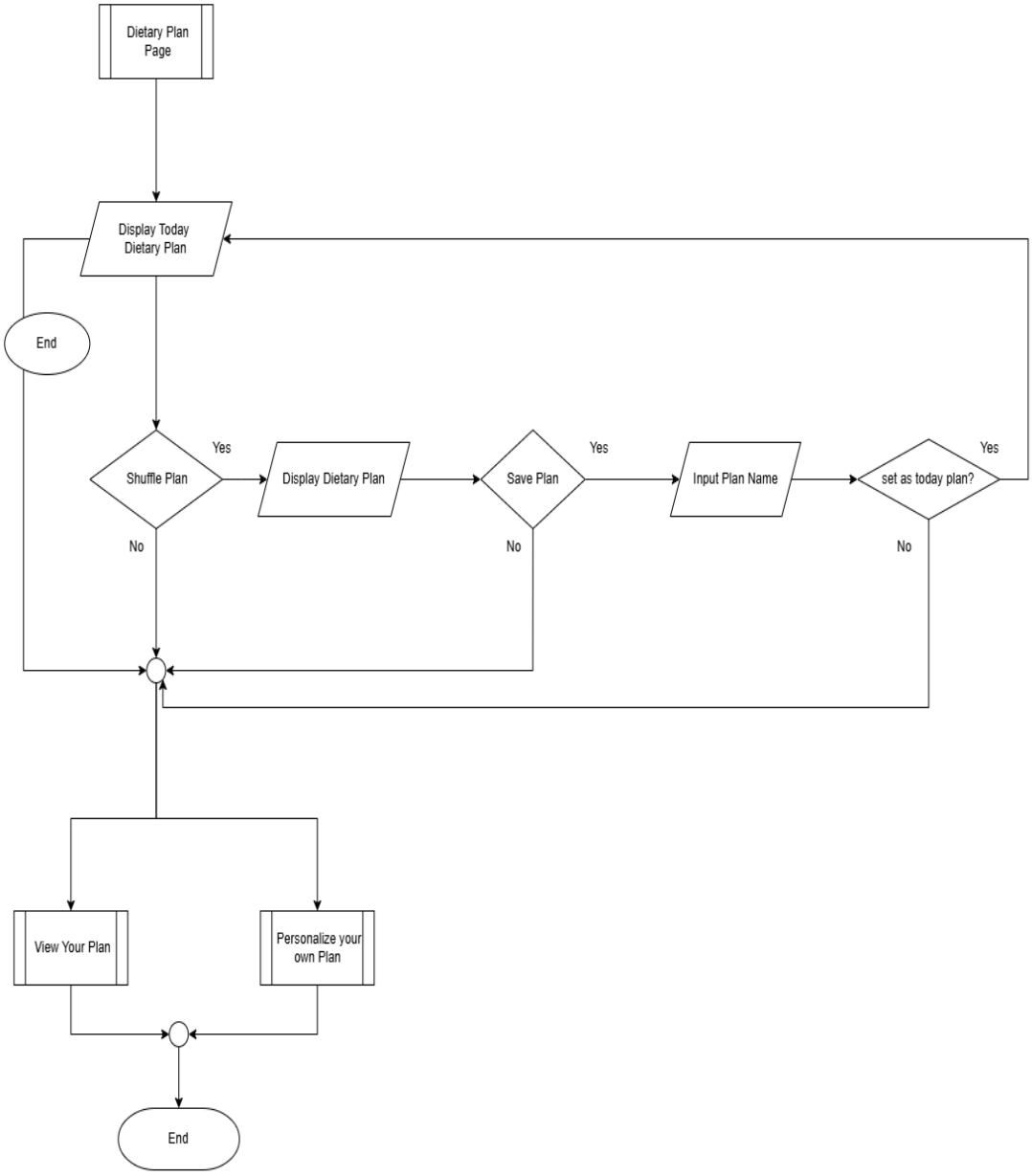


Figure 4.6 Flowchart of Dietary Plan Page

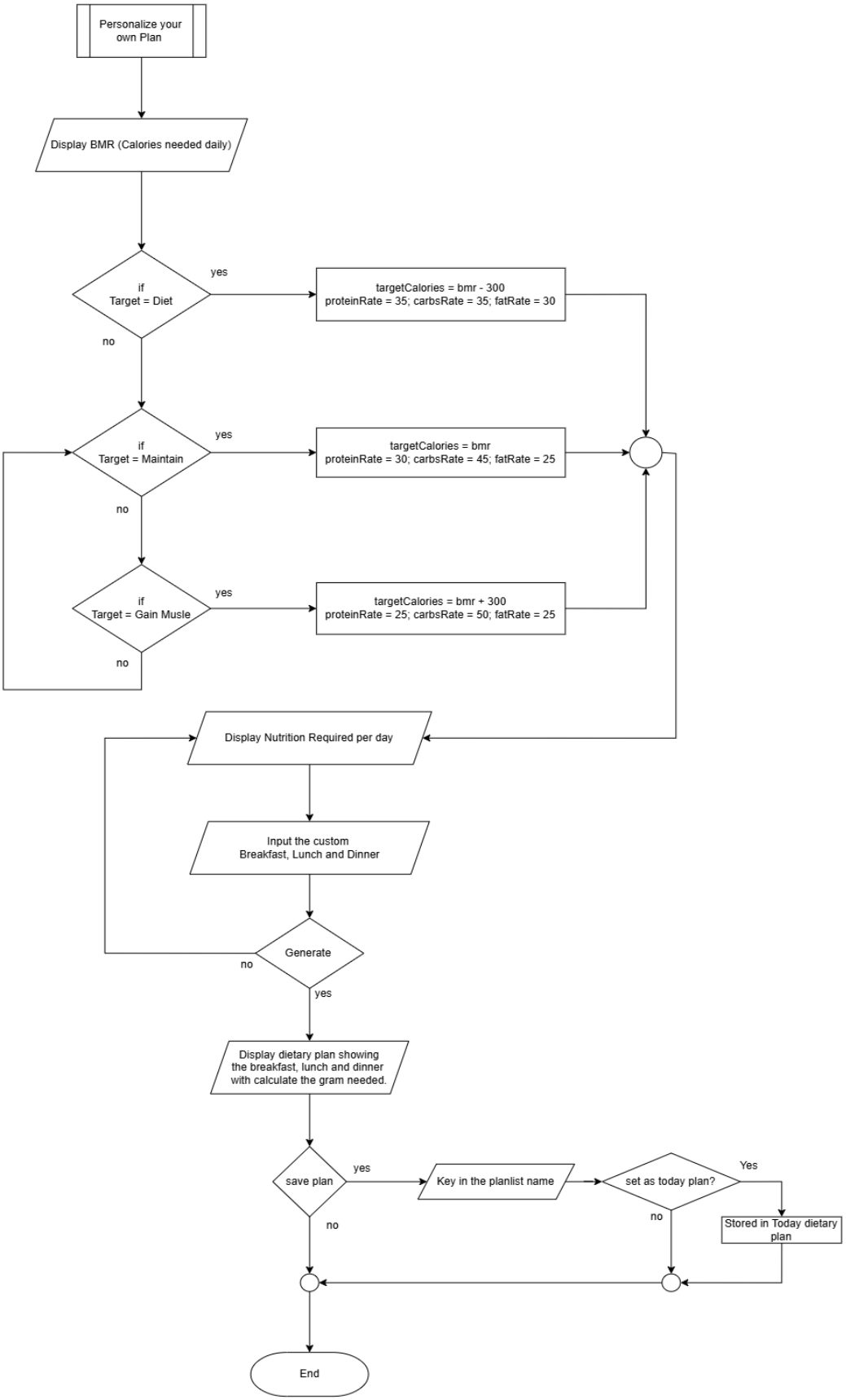


Figure 4.7 Flowchart of Personalize Dietary Plan

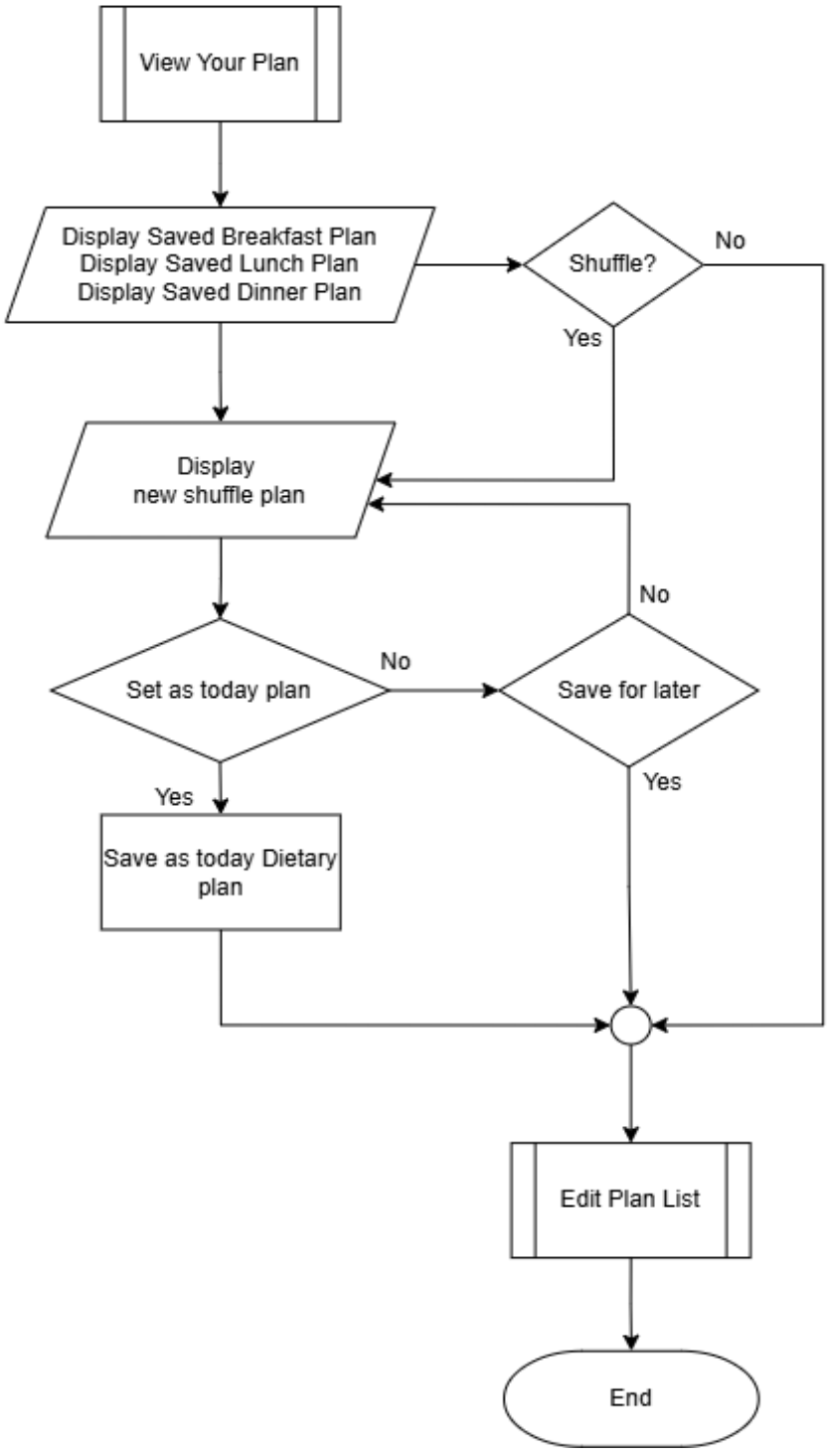


Figure 4.8 Flowchart of View Dietary Plan

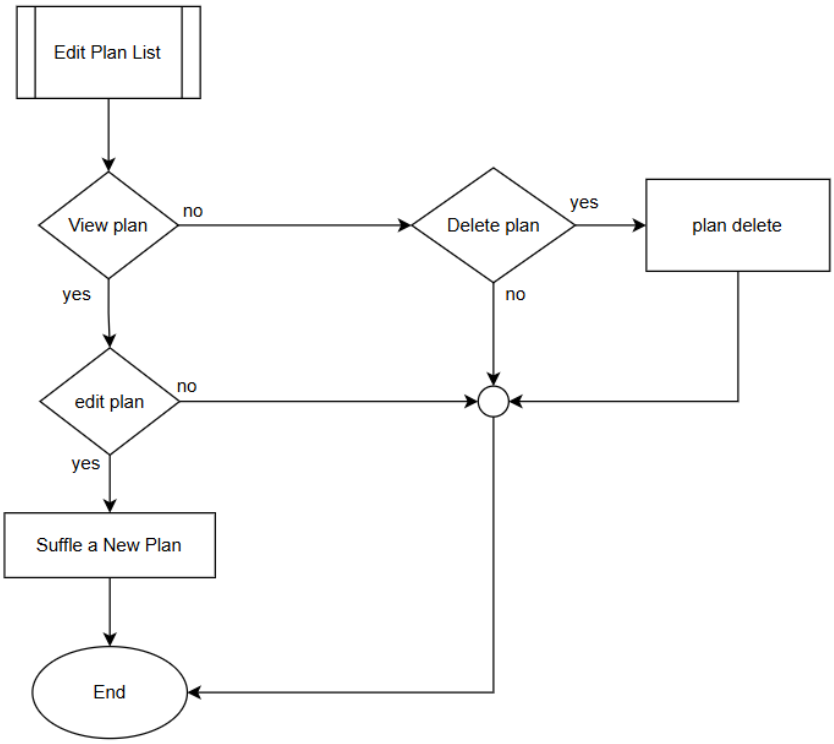


Figure 4.9 Flowchart of Edit Plan List

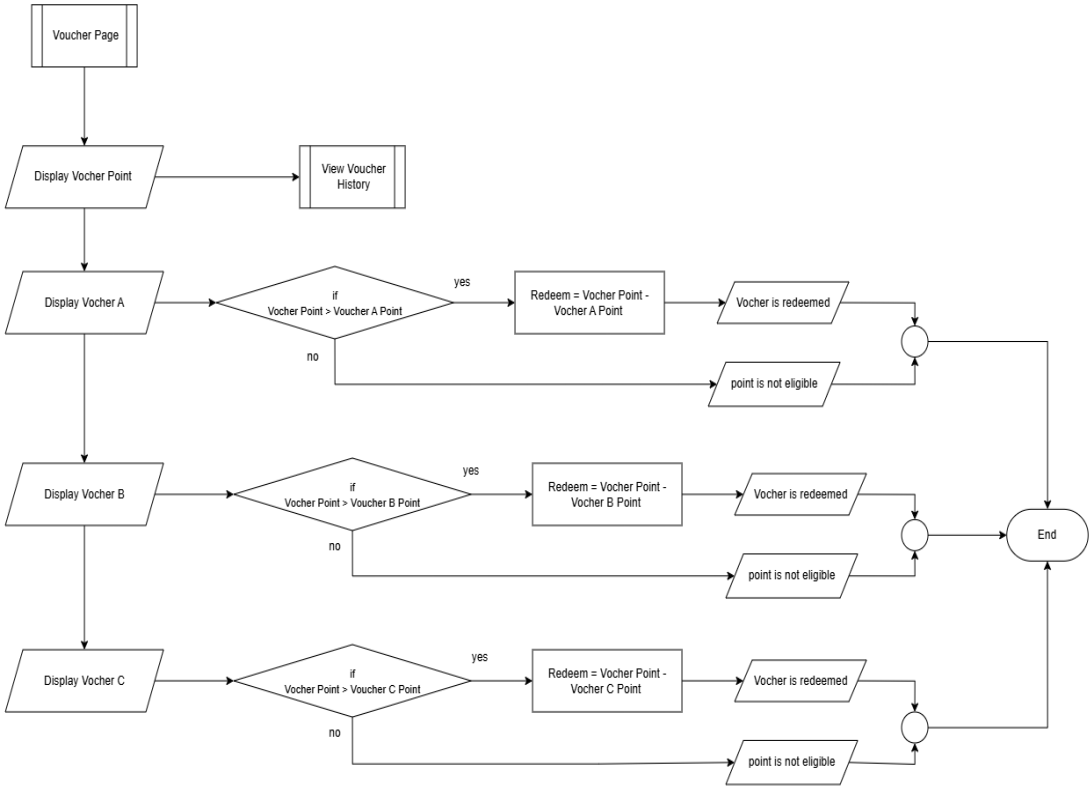


Figure 4.10 Flowchart of Voucher Page

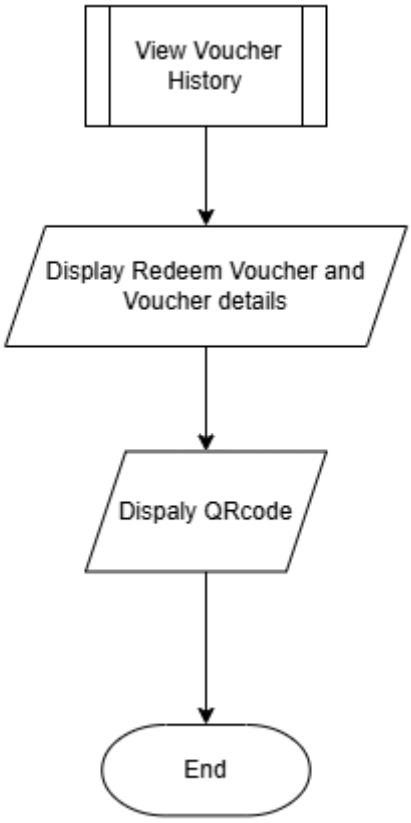


Figure 4.11 Flowchart of View Voucher History

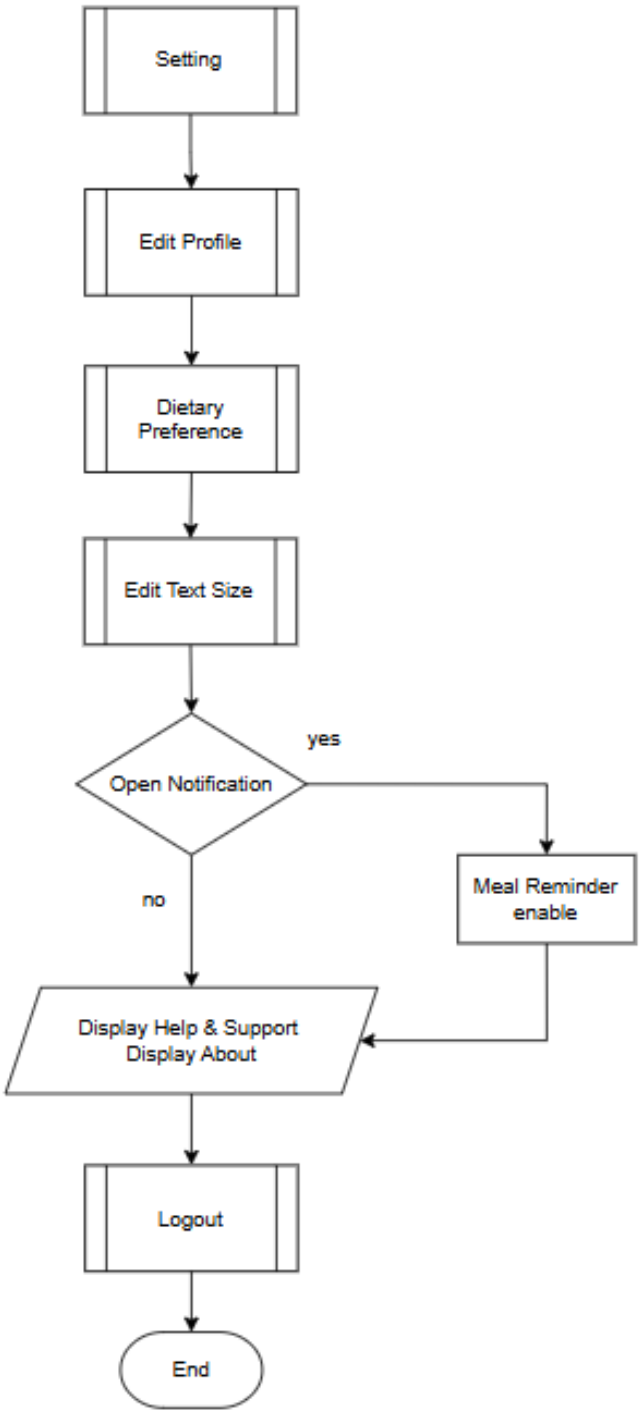


Figure 4.12 Flowchart of Setting

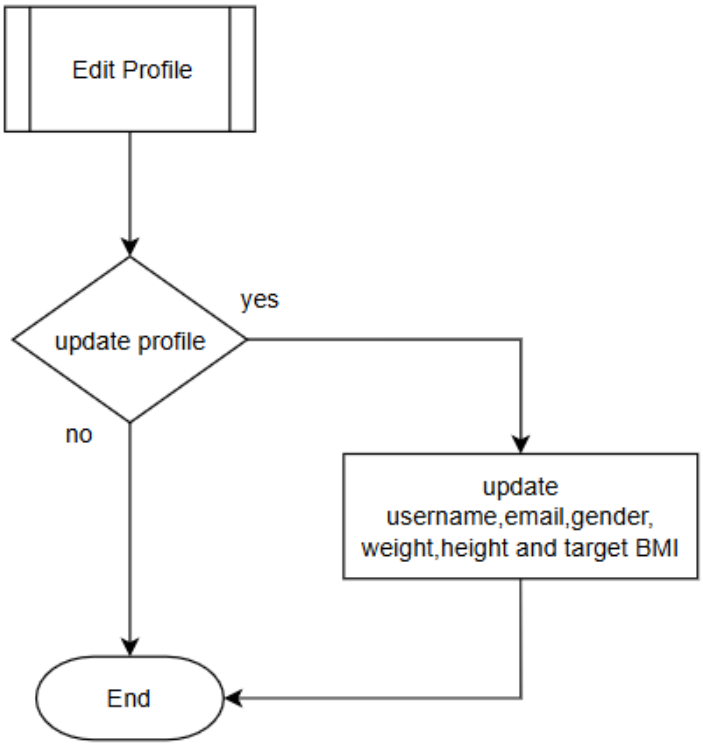


Figure 4.13 Flowchart of Edit Profile

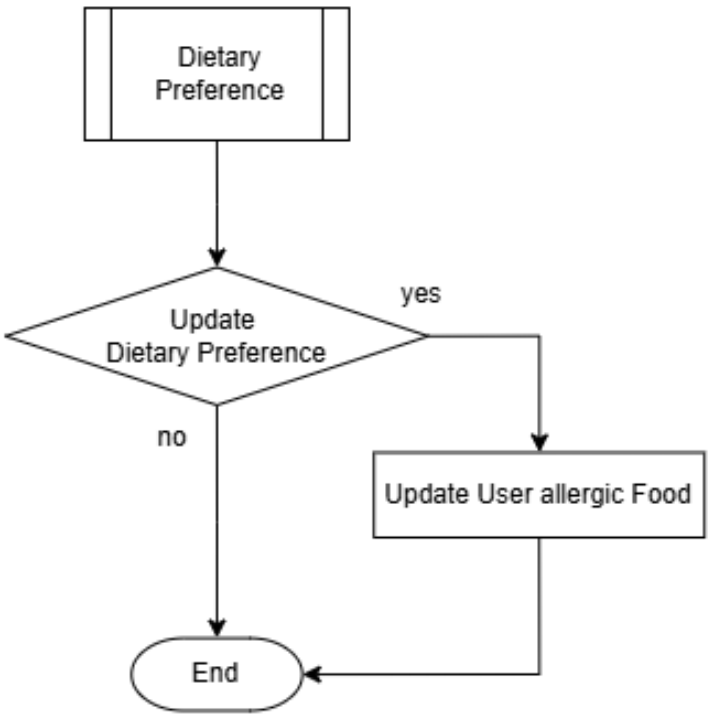


Figure 4.14 Flowchart of Dietary Preferences

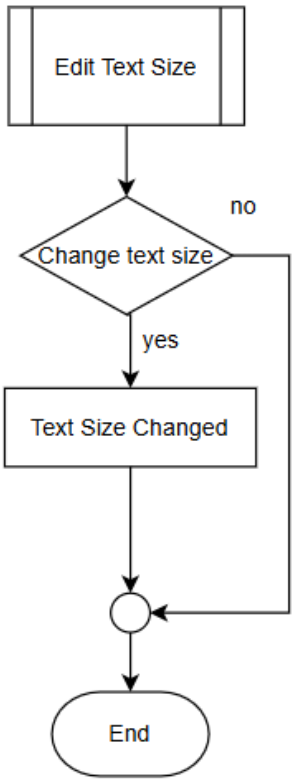


Figure 4.15 Flowchart of Edit Text Size

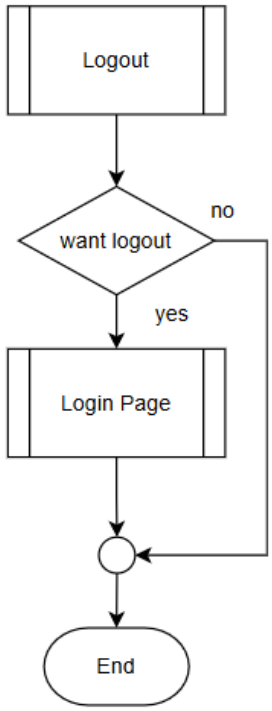


Figure 4.16 Flowchart of Logou

4.2 System ERD Diagram

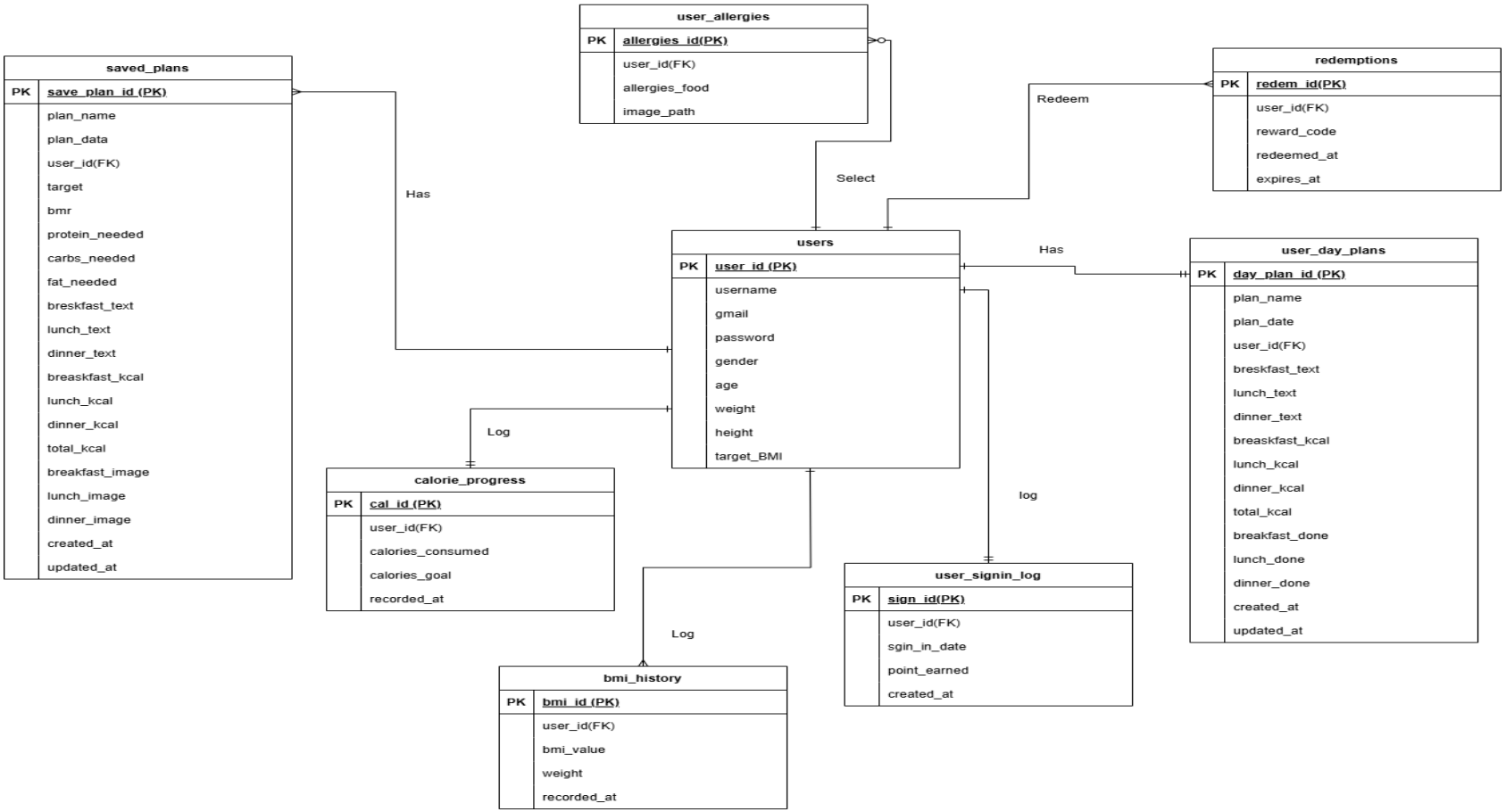


Figure 4.17 System ERD Diagram

The database structure appears in the Entity-Relationship Diagram that specifies how the system stores data

The **users table** in the database has the fields username, password, and email used to authenticate the user and a combination of gender, age, weight, height, bmi, and target_bmi to monitor their health status. Every user of the system is referred to by a unique user_id within the system. The users table of the database is a table of the primary key user_id which is automatically assigned. (Relationships: One user has zero or many allergies, one user has one or many saved plans, one user tracks one or many BMI histories, one user tracks one and only one calorie progresses per day, one user logs one and only one user sign-ins per day, one user has one and only one user day plans, one user redeems one or many redemptions.)

The **saved_plans** table will have plan_id as the primary key and user_id as the single foreign key relating to the users table. The saved_plans table is linked with the users table using this foreign key enabling the two tables to be related in the database system. The saved_plans table employs the user_id foreign key as a way of accessing the information about the meal plan user. In saved_plans table, four attributes will be used, including plan_name, plan_data, user_id and bmi_target to save meal plan and nutritional targets. (Relationship: One user has one or many saved plans.)

The **user_allergies** table has the allergies_id as the primary key, foreign key is the user_id and attribute of allergenic_food and image_path. The user_allergies table serves to store user specific allergies and the image_path to save the image of allergic food as a visual for user. (Relationship: One user has zero or many allergies.)

The table **bmi_history** integrates the bmi_id as the primary key along with a foreign key user_id and the values of bmi_value, weight and the recorded_at. The user_id is the foreign key which is used to access information stored in the users table and it forms the relationship of health records to individual users. The table helps in record the health

tracking system by logging the user weight changed on the recorded date. Relationship: One user track one or many BMI histories.)

The **calorie_progress** table has cal_id as the primary key and user_id as its foreign key with other attributes such as calories_consumed, calories_goal, recorded_at and updated_at. user_id foreign key aids in retrieving the data in the users table to create the progress linking between the individual users. (Relationship: One user tracks one and only one calorie progresses per day)

user_signin_log table is a table with primary key sign_id and foreign key user_id with attributes and sign_in_date, point_earned and created_at. The user_id foreign key will help in accessing the data of the users table to record the sign-in activities of individual users. This table will record the user's daily sign-in process and manage the point earn rewards. (Relationship: One user log one and only one sign-in per day)

The **user_day_plans** table uses day_plan_id as the primary key, user_id as the foreign key to the users table, plan_name, plan_date, breakfast_text, lunch_text, dinner_text, breakfast_kcal, lunch_kcal, dinner_kcal, total_kcal, breakfast_done, lunch_done, dinner_done, created_at, and updated_at as the attributes. The user_id is a foreign key that is linked to the users table to identify the daily plans with specific users. This table will record the user's dietary plan with the plan name, date, calories and nutrition. Besides the breakfast_done, lunch_done, dinner_done is referred to the functional check box to for user to click when they have done taking the meals. (Relationship: One user has one and only one user day plans)

The **redemptions** table has redeem_id as its primary key and user_id as its foreign key, reward_code, redeemed_at and expires_at as its attributes. The user id is a foreign key that can be used to get information of users table to create redemption links by individual users. This table will record the voucher redemption by the user. Users can

view back the voucher they have redeemed and the expiration date of the voucher. (Relationship: One user redeems one or many redemptions.)

4.3 System Use-Case Diagram

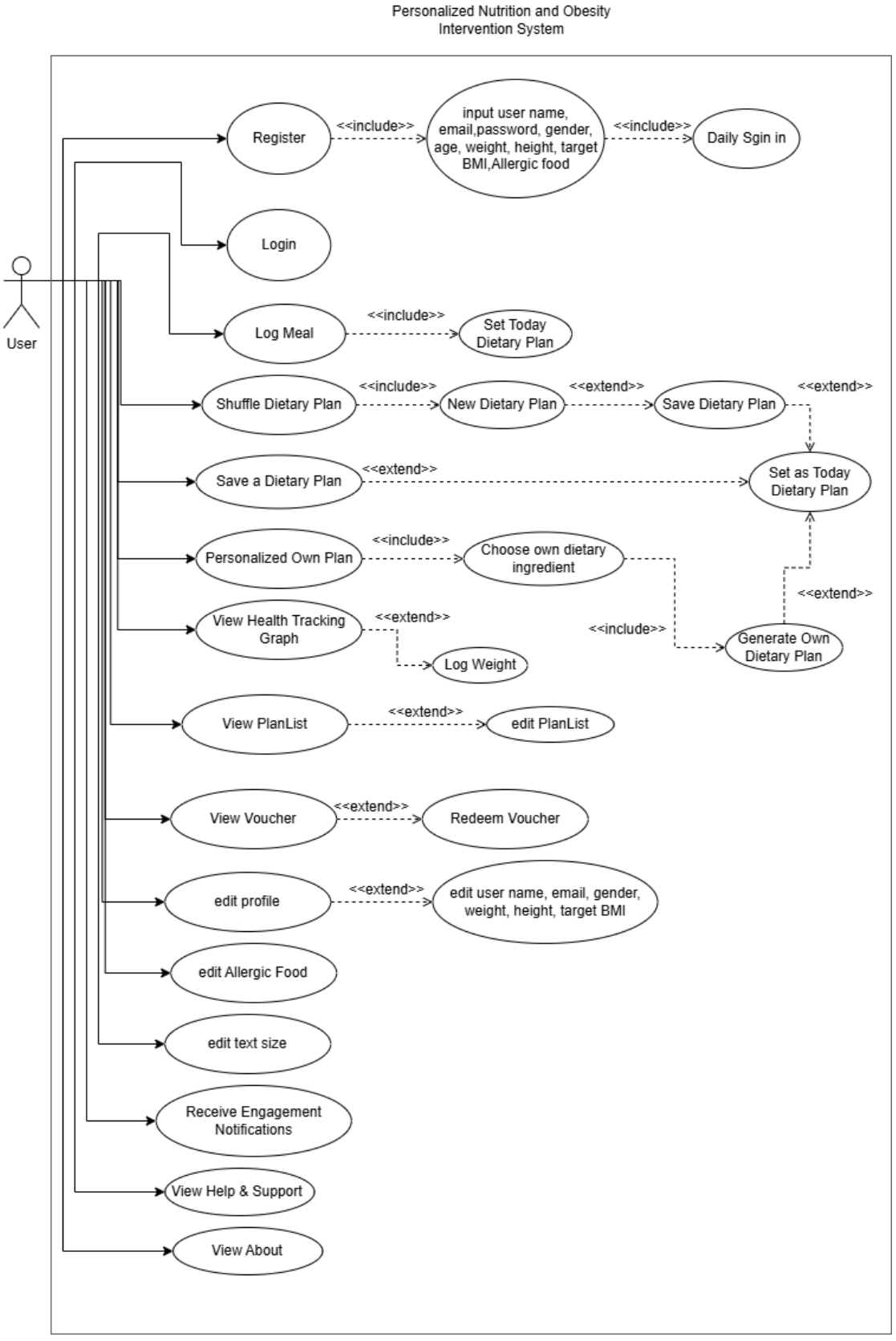


Figure 4.18 System Use-Case Diagram

Through its Use Case Diagram this diagram shows how users interact with the Personalized Nutrition and Obesity Intervention System by illustrating the major features of the Android application for obesity management. The main actor of the system is the User which is depicted as a stick figure who performs activities like registration and login activities as well as BMI calculation and dietary plan generation and health metric tracking and notification receipt. The diagram defines mandatory dependencies through UML notation using the notation <<include>> and establishes optional behaviours through UML notation using <<extend>>.

Use Cases

Register: The User creates an account by inputting username, email, password, gender, age, weight, height, target BMI, and allergic food, stored in the users and user_allergies tables.

Login: The User logs in using email and password, verified against the users table.

Daily Sign In: The User checks in daily, with data logged in the sign_in_log table, extending from the login process.

Log Meal: The User records meals such as breakfast, lunch and dinner with text, images, and calories, updating the day_plans and calorie_progress tables, including the set as today plan.

Shuffle Dietary Plan: The User requests a new randomized plan, including the generation of a dietary plan, updating the saved_plans table.

New Dietary Plan: The system generates a personalized meal plan based on user data such as user BMI and allergies, stored in the saved_plans table, extending to save dietary plan.

Save Dietary Plan: The User saves a generated plan, stored in the saved_plans table, extending to set as today or generate own dietary plan.

Set as Today: The User assigns a saved plan to the current day, populating the user_day_plans table, extending from save dietary plan.

Choose Own Dietary Ingredient: The User selects custom ingredients, included in generating own dietary plan, stored in the saved_plans table.

Generate Own Dietary Plan: The User manually creates a plan with custom ingredients, stored in the saved_plans table, extending from save dietary plan.

View Health Tracking Graph: The User views progress charts using data from bmi_history and calorie_progress tables, including log weight updates.

View Plan List: The User views their saved plans from the saved_plans table, with an optional extend to edit plan list.

Edit Plan List: The User modifies or deletes plans, extending from view plan list, updating the saved_plans table.

View Voucher: The User views available rewards, with data from the redemptions table, extending to redeem voucher.

Redeem Voucher: The User redeems a voucher by using the collected point from daily sign-in system , updating the redemptions table, extending from view voucher.

Edit Profile: The User updates their profile such as username, email, gender, weight, height, target BMI and stored in the user's table.

Edit Allergic Food: The User adds or removes allergies, updating the user_allergies table.

Edit Text Size: The User adjusts the app's text size for accessibility.

Receive Engagement Notifications: The User receives adherence reminders to take their meals on time.

View Help & Support: The User accesses guidance content on static page.

View About: The User views system information on static page.

4.4 System Architecture Diagram

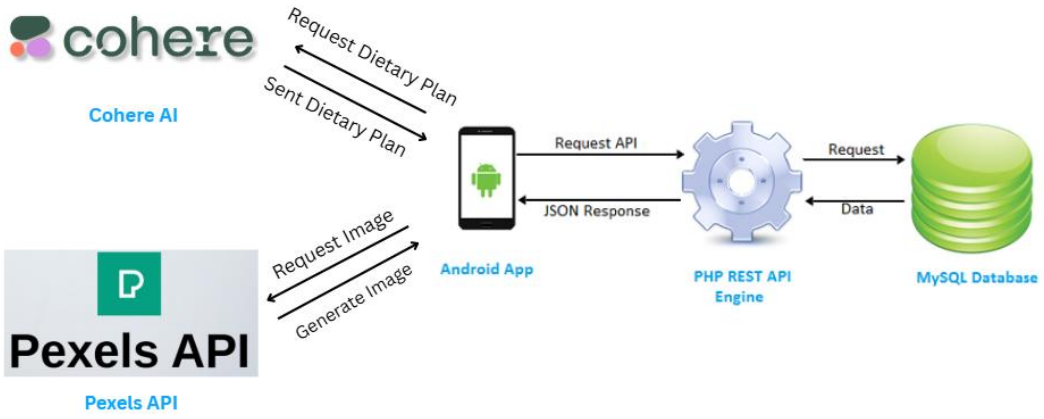


Figure 4.19 System Architecture Diagram

The Personalized Nutrition and Obesity Intervention System features a system architecture diagram that demonstrates the high-level framework of its operation through interaction between Cohere AI, Pexels API and Android App alongside PHP REST API Engine and MySQL Database in Figure 4.19.

The system components work together to realize the necessary features and functionalities of the app, beginning with authentication to customized meals production, image delivery, health monitoring, and engagement notifications as outlined in the mobile design. Data and request flow is illustrated with arrows in the diagram to illustrate how user data is converted into created meal plans and images whilst storing and retrieving stored data.

Cohere AI: The system contains an external AI service which creates customized dietary plans according to user input data consisting of age, weight, height, target BMI along with allergies. The Android App provisions request to Cohere AI for dietary plan suggestions through its “Suggest dietary plan” functionality.

Pexels API: This third party offers image creating features, which increases the visual value of meal plans. The Android App is calling upon Pexels API that produces and supplies the appropriate images to fit the dietary plans.

Android App: The Java-based application in Android Studio acts as the interactive interface for users. Users can perform the process of registration while also managing login access to input dietary choices for meal plan viewing and health tracking along with notification receipt. The app makes a request to Cohere AI to get the dietary plans and to Pexels API to get the images and the response is a JSON file and the generated image, which is processed through the PHP REST API Engine.

PHP REST API Engine: The online platform made with PHP for XAMPP facilitates data exchange between the Android App and MySQL Database. Through its API processing functionality, the system executes authentication requests by generating JSON responses and processes information to generate meal plans and image mergers.

MySQL Database: XAMPP controls persistent storage that stores data in healthyplan database tables including users, saved_plans, user_allergies, bmi_history, calorie_progress, sign_in_log, user_day_plan and redemptions. The PHP REST API Engine processes the data requests and allows registering users, tracing their health, and storing user's plans.

CHAPTER 5 System Implementation

5.1 Setting up

5.1.1 Software setting up

MySQL

1. Download the XAMPP app

Figure 5.1 XAMPP software

XAMPP functions as the development platform for the server-side code since it operates Apache to host PHP REST APIs which contain GetPlanData.php, login.php, InsertPlan.php, UpdatePlanData, get_bmi_and_target.php, get_saved_plans.php, get_today_plan.php, plan_actions.php, rewards.php, save_personal_plan.php, save_plan.php, set_meal_done.php, settings.php, tracking.php, update_meal_done.php. data management interfaces that support user login and dietary input alongside Cohere AI meal plan creation and Pexels API to generate the relevant image to the dietary plan to improve the visualisation.

The entire application relies on MySQL to handle database operations for the healthyplan database by using tables including users, saved_plans, user_allergies, bmi_history, calorie_progress, sign_in_log, user_day_plan and redemptions as well as the phpMyAdmin tool for administrator tasks that facilitate both local testing and development convergence with the Android version.

2. Open the XAMPP and start Apache and MySQL

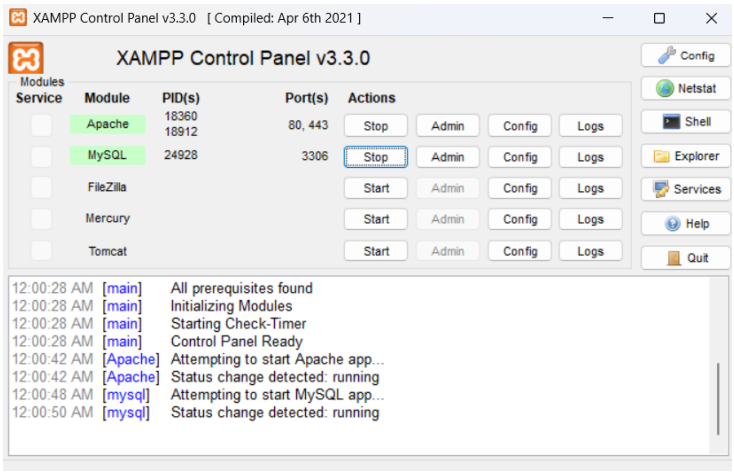


Figure 5.2 XAMPP Control Panel

3. On MySQL Module, Click on Admin and It will lead to the phpMyAdmin localhost page.

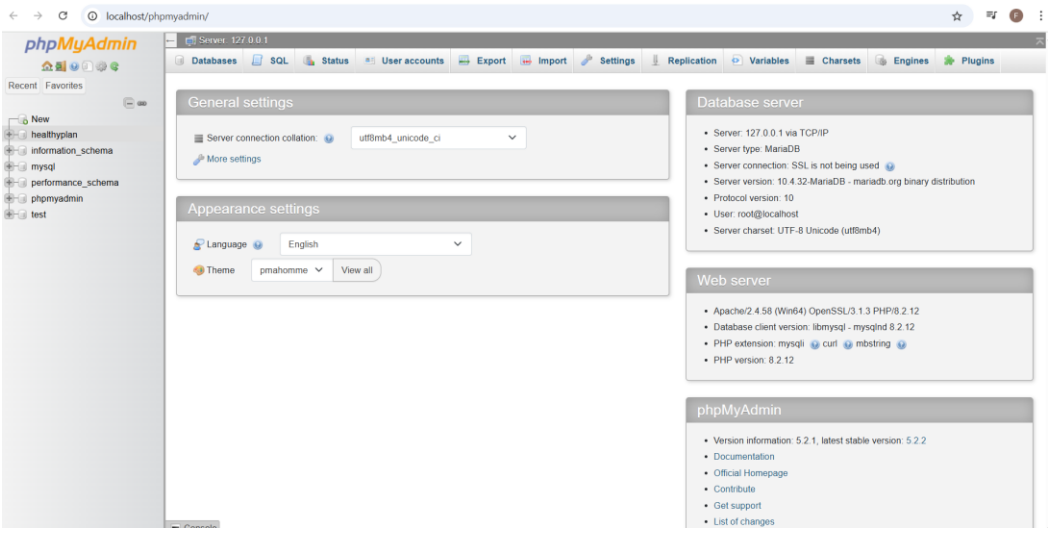


Figure 5.3 phpMyAdmin localhost page

4. Click on new to create a new database and the project database is “healthyplan”.

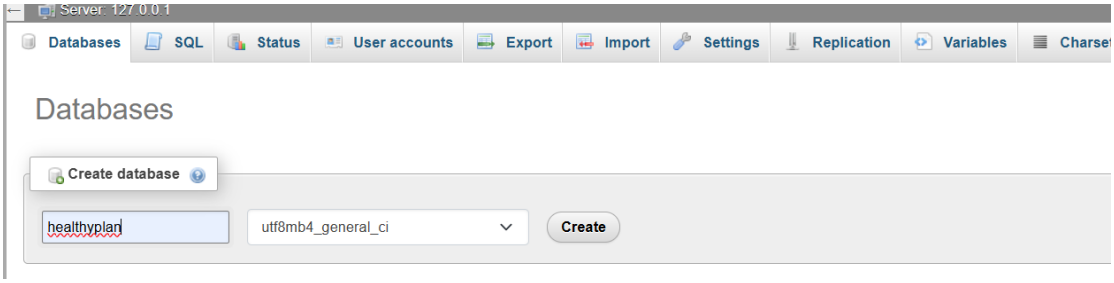


Figure 5.4 Database Create page.

5. After click create, it can start creating the table in the healthyplan.sql database.

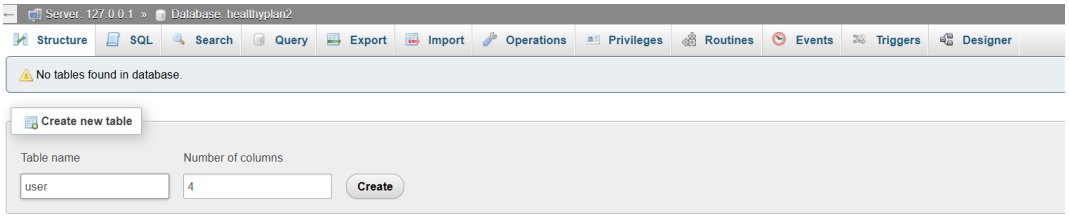


Figure 5.5 Create the database table.

6. After done creating the database, the database and table will appear in the database server.

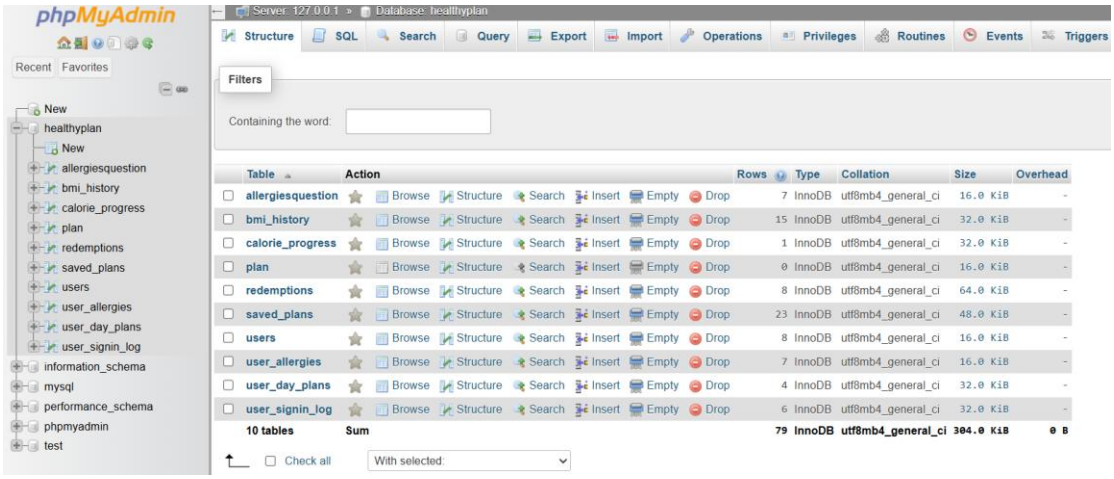


Figure 5.6 “healthyplan” database and the table appeared in the sever.

7.Open File Explorer, create the php file inside the htdocs.

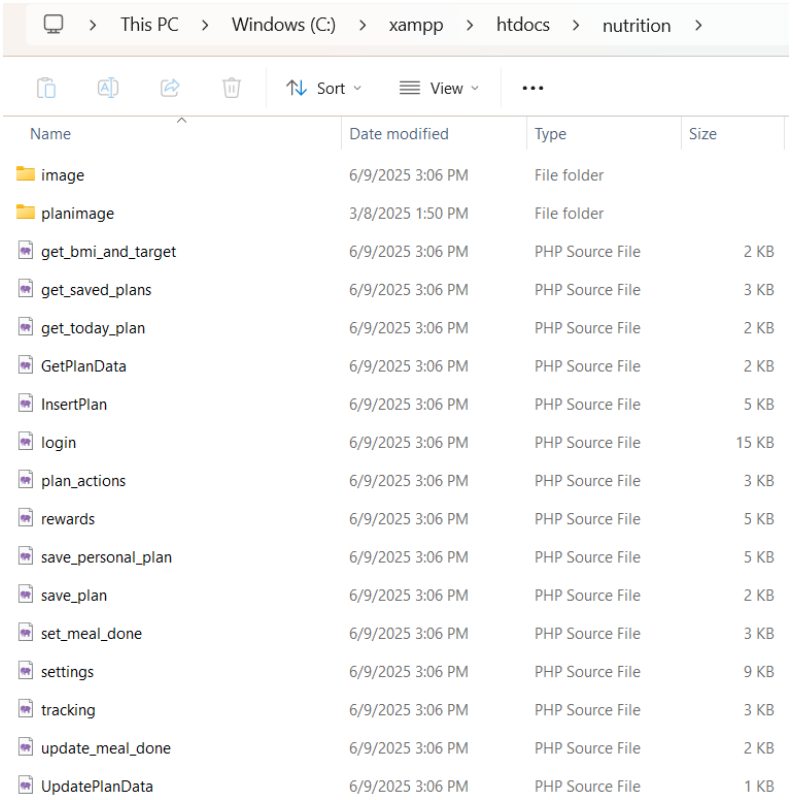


Figure 5.7 php file that created.

4. Inside the php file write the function coding inside the php file.

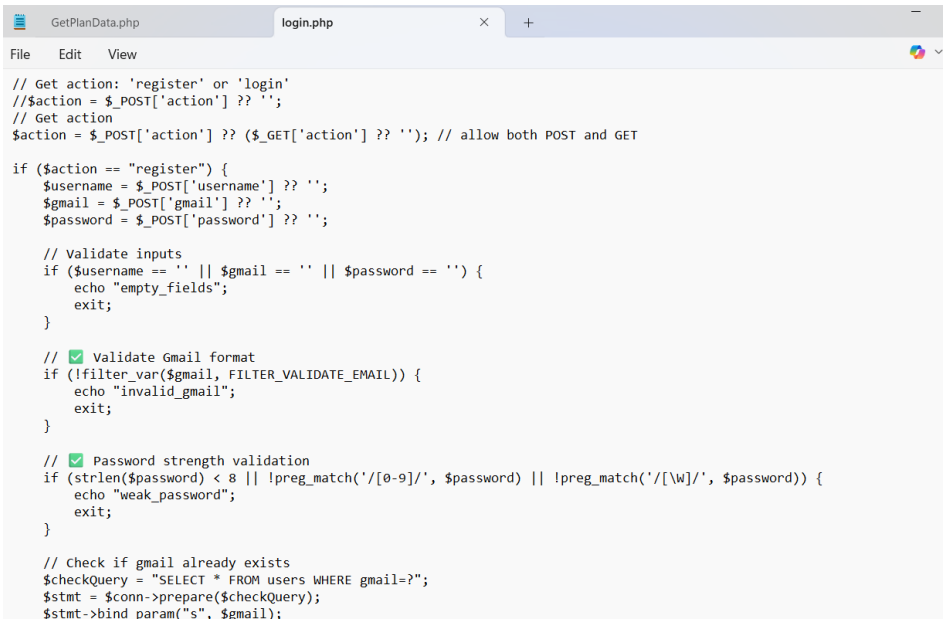


Figure 5.8 function coding in the login.php file.

5. Write this code inside the php file to connect with “healthyplan” database.

```php
<?php
$host = "localhost";
$user = "root";
$password = "";
$db = "healthyplan";

// Connect to DB
$conn = new mysqli($host, $user, $password, $db, 3306);
$userid="";
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
```

Figure 5.9 Shows the coding that need to be key in to connect the database.

6. Inside the Android Studio, coding the URL to call the function inside the php and to save the data into the database.

```
    }

    1 usage
    private void saveUserData(int gender, int age, int height, float weight, float target_BMI ) {
        String url = "http://10.0.2.2/nutrition/login.php?action=save_user";

        int userId = getSharedPreferences( name: "user_prefs", MODE_PRIVATE).getInt( key: "user_id", defValue: -1);

        StringRequest request = new StringRequest(Request.Method.POST, url,
                String response -> Toast.makeText( context: this, text: "User saved", Toast.LENGTH_SHORT).show(),
                VolleyError error -> Toast.makeText( context: this, text: "Error saving user", Toast.LENGTH_SHORT).show()
        ) {
```

Figure 5.10 Shows the coding to call the function inside the php file.

7. After we done any key in on the front end of the mobile system, it will save the data inside the database.

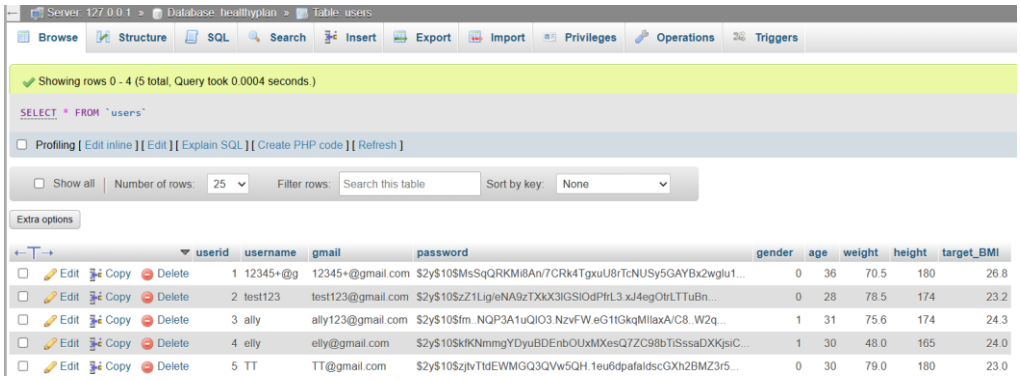


Figure 5.11 Shows the data have been save inside the database.

5.1.2 Cohere AI API setting up

1. Go to the Cohere AI API website “https://docs.cohere.com/”.

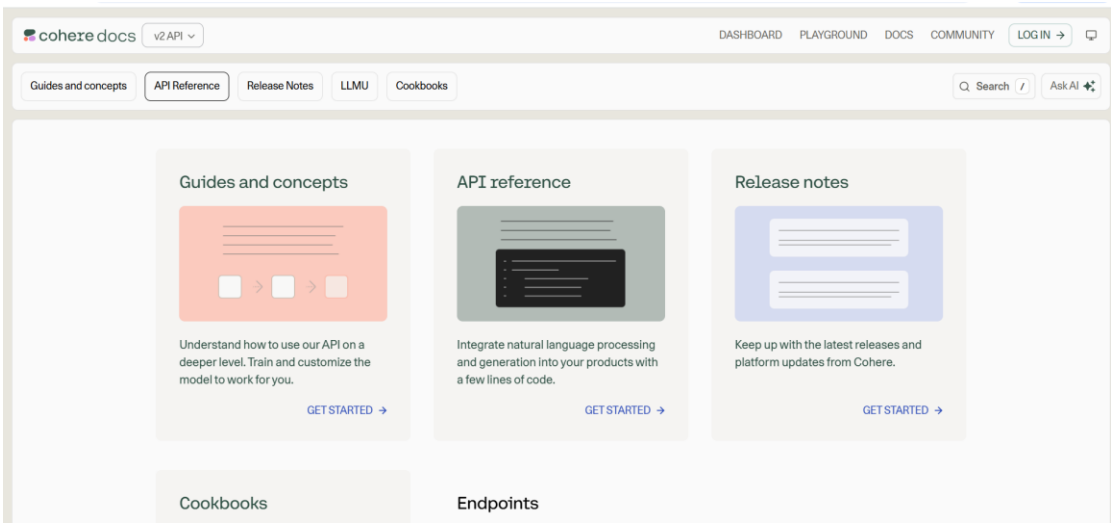


Figure 5.12 Cohere AI website.

2. Login or sign up to the Cohere AI account.

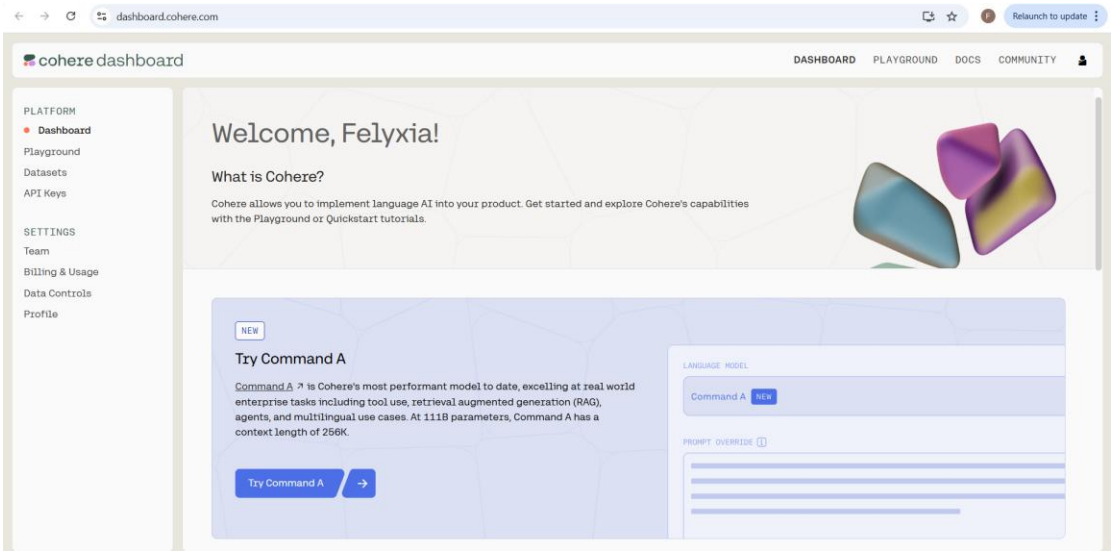


Figure 5.13 Successful login to the Cohere AI account dashboard page.

3. Go to API Keys page, get the free trail API keys.

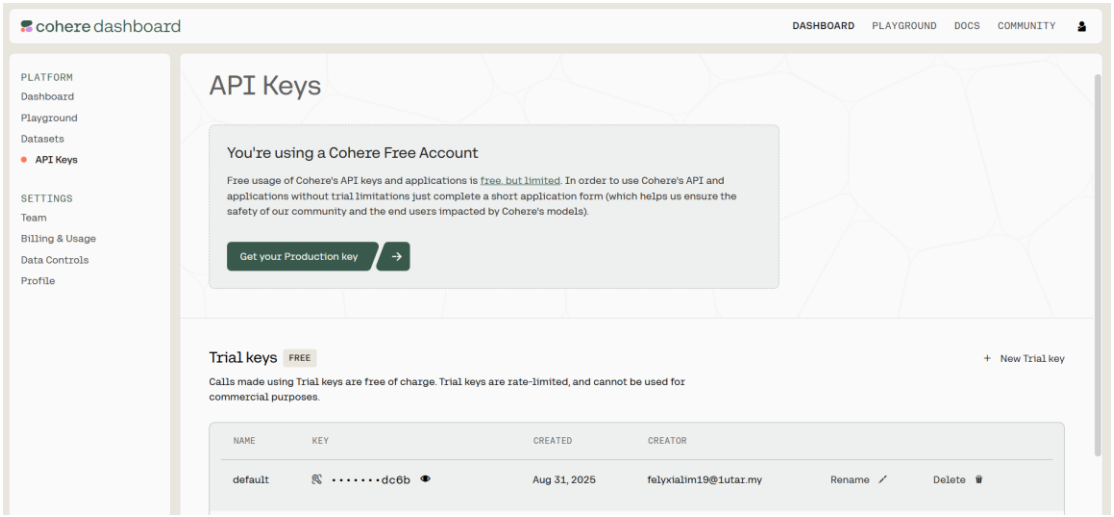


Figure 5.14 API Keys page show the free trail keys for every free user of Cohere AI.

4. Copy and paste the trail API keys into the CohereService.java.

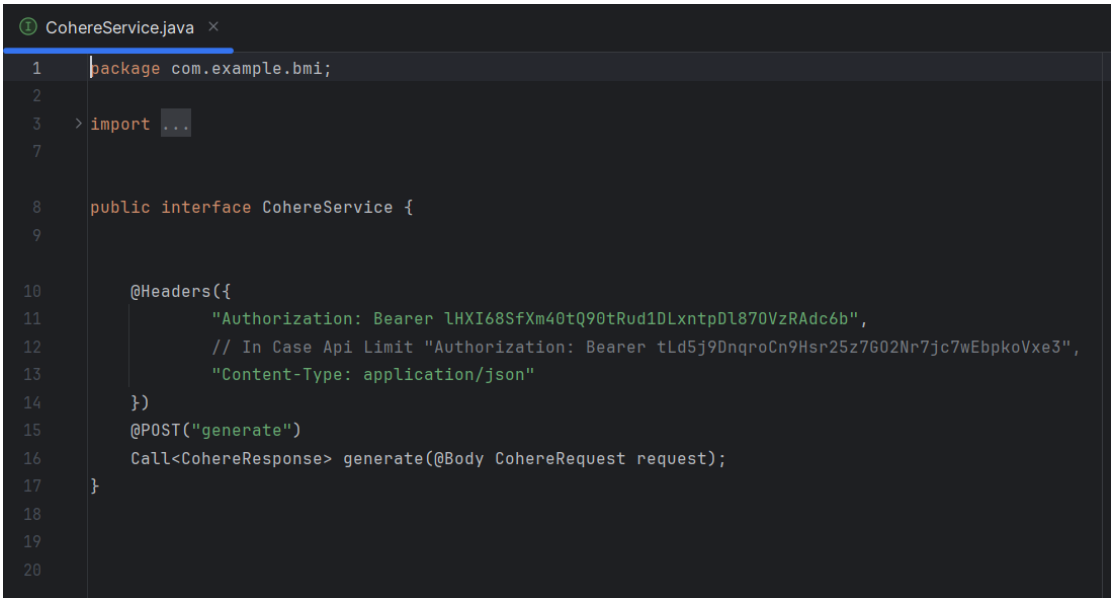


Figure 5.15 Paste the trail Cohere API keys to the CohereService.java to get the Cohere AI response.

5.1.3 Pexels API setting up

1. Go to the Pexels website “https://www.pexels.com/”.

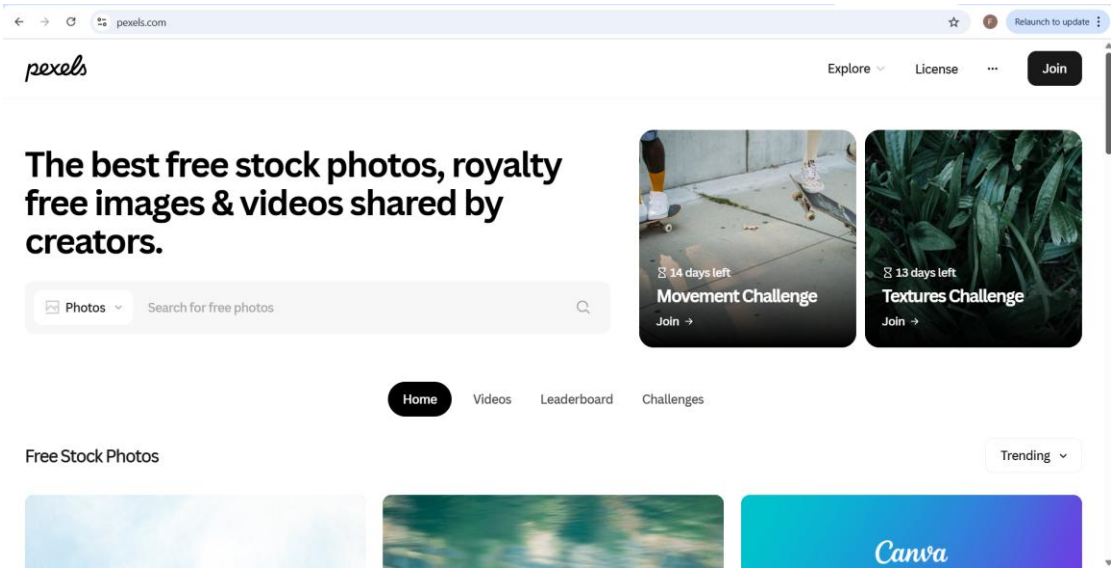


Figure 5.16 Pexels Website

2. Login to own Pexels account and click on the button “Your API Key”.

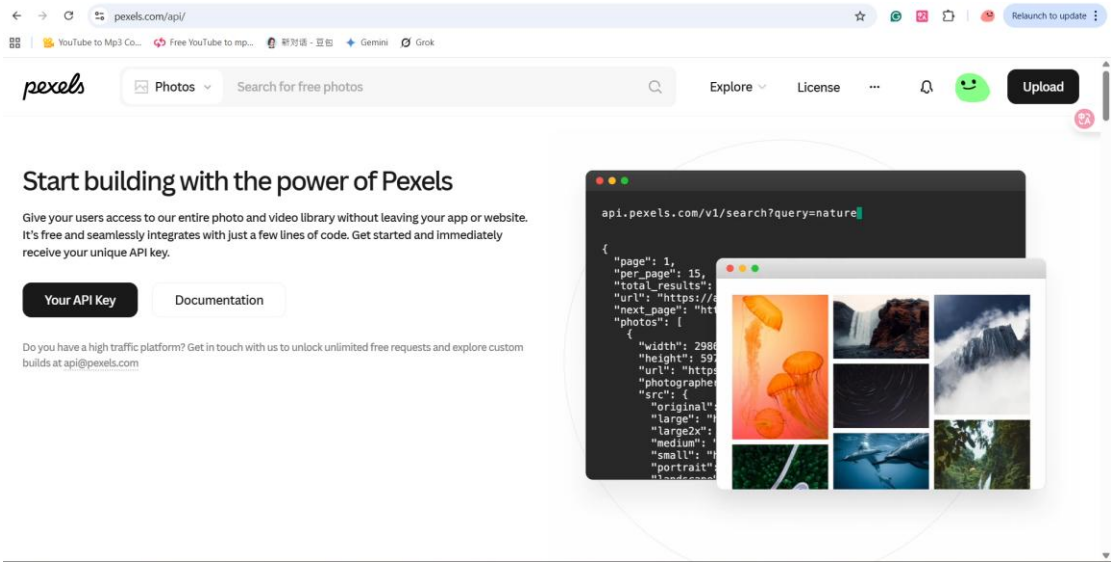


Figure 5.17 Login to Pexels account.

3. Key in the relevant question to generate the API key.

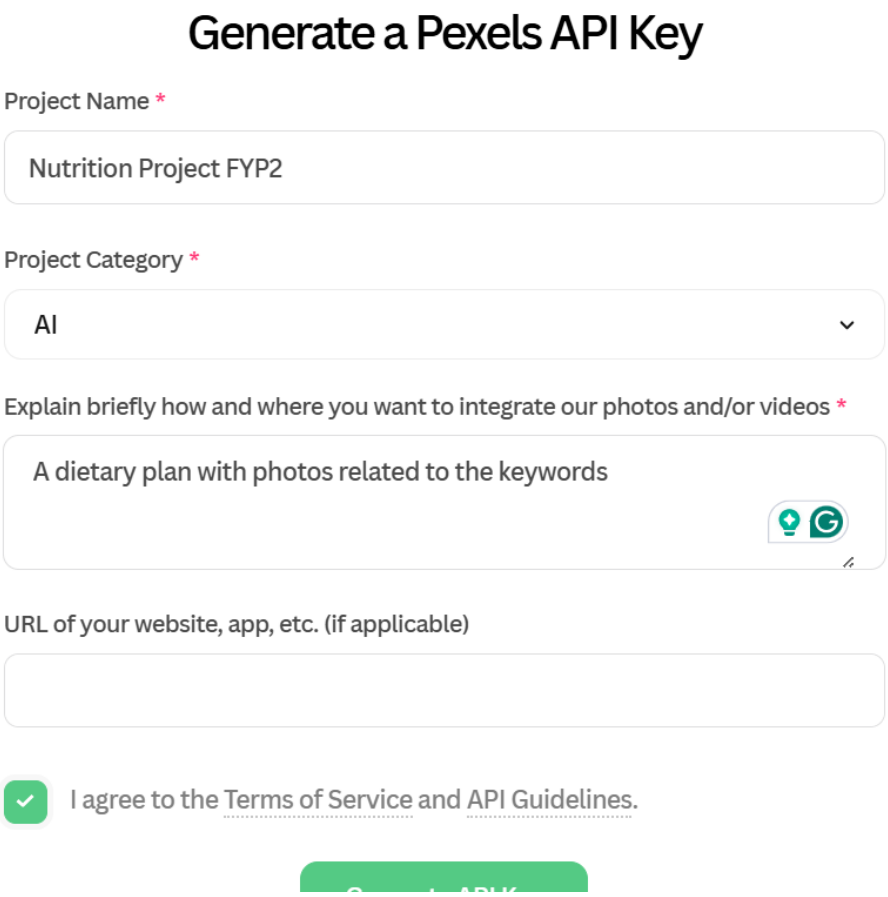


Figure 5.18 Key in the relevant data to generate the API key.

4. After Click on generate button, the API key will be displayed.

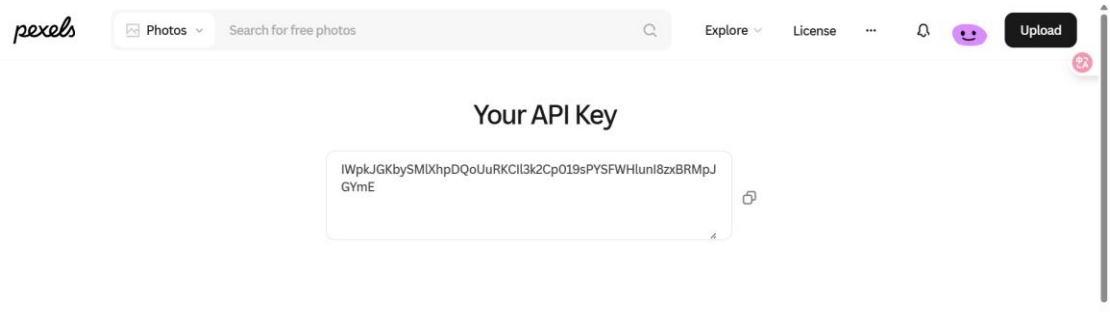


Figure 5.19 API key has successful been generated and ready to be use.

5.2 Screenshot of the System Implementation

5.2.1 Splash Page

Figure 5.20 Splash Page of the System

The splash page will be shown when users start running the system. The splash page will be loaded for 3 seconds. It is employed to entice the attention of the visitor with the use of attractive graphics, videos, or animations to interest the visitor and have a strong first impression. They can also act as a potent brand marketing instrument as they continuously showcase the logo, colors and message of a brand that creates its image in the mind of the visitors [19].

5.2.2 Start Page

Figure 5.21 Start Page

The start page shows the big logo for the application, log in button and sign-up button for user. Users can choose directly from log in to present account or create a new account.

5.2.3 Login Page

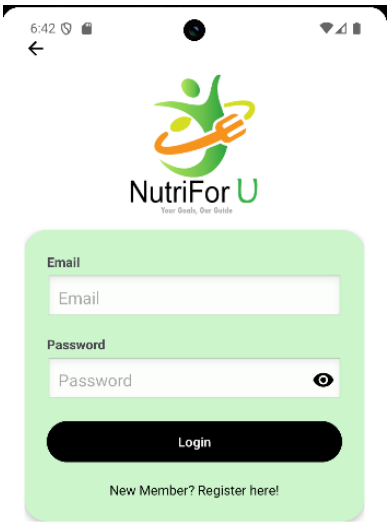


Figure 5.22 Login Page

The Login page allows users to login to their own account, with their own email address and password. If the user is not registered yet, the user can click on the “New Member? Register here!" word for registering an account.

5.2.4 Registration Page

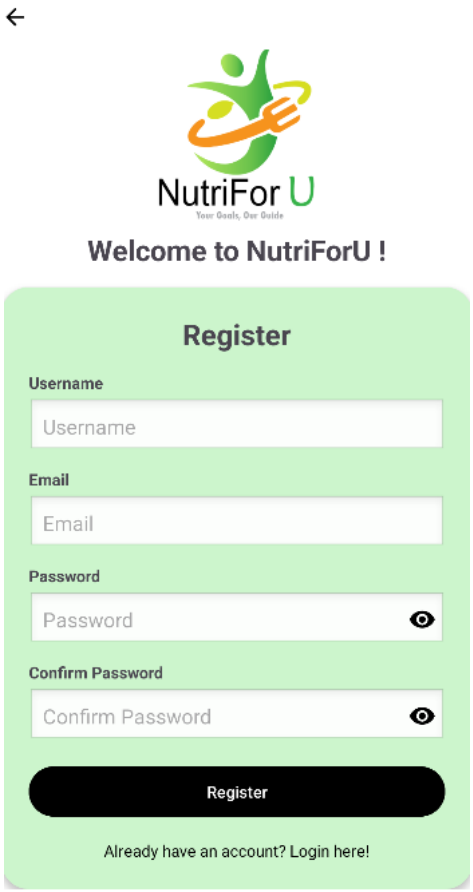


Figure 5.23 Registration Page

The Registration page allows users to create an account by inputting their username, email address and password.

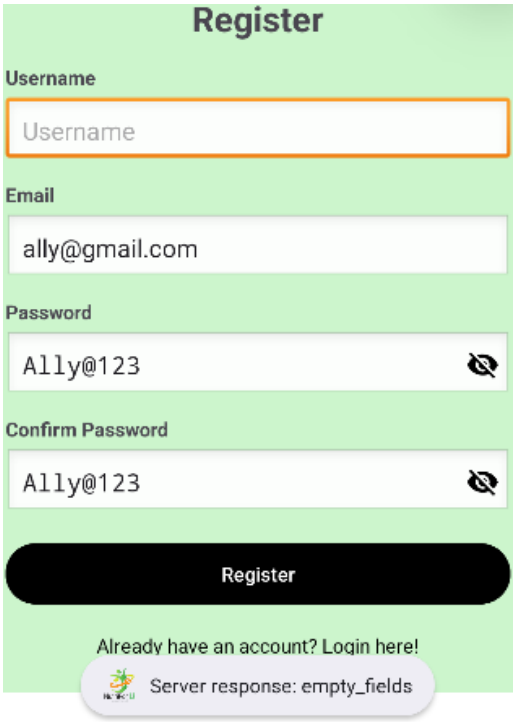


Figure 5.24 Failure of registration prompt when user does not key in their username

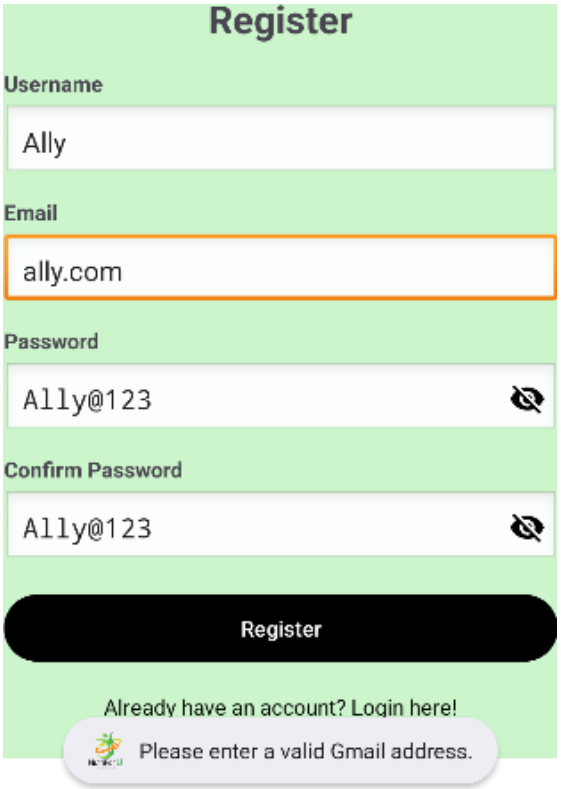


Figure 5.25 Failure of registration prompt when user does not key in their email in correcr format

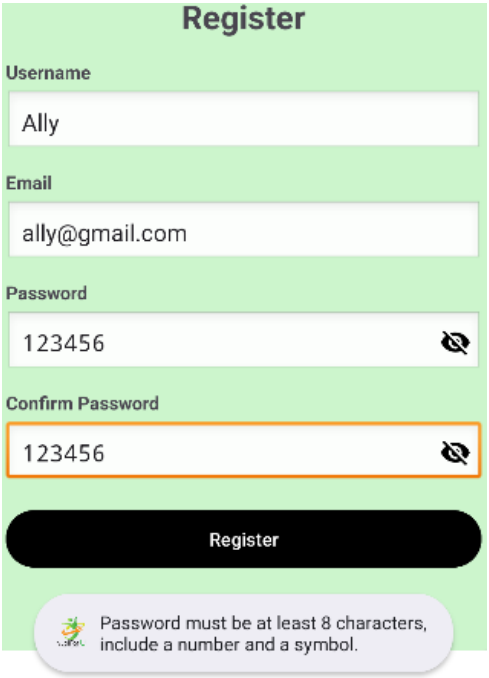


Figure 5.26 Failure of registration prompt when user does key in their password in correct validation format.

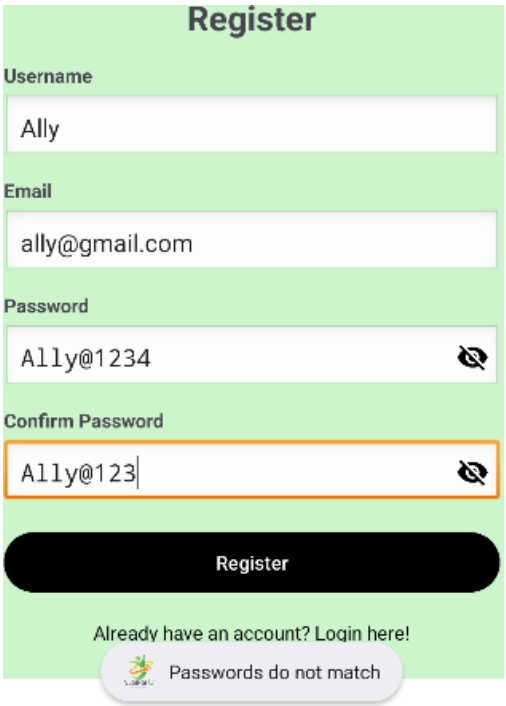


Figure 5.27 Failure of registration prompt when user does key in correct password in confirm password.

In the Registration page, there would be a validation check to alert the user. For example, user does not key in their username and an empty field will alert the user. Next, if user didn't key in the email format correctly, a prompt message will appear says that "Please enter a valid Gmail address". Besides. The password validation also will alert the user when the user key is in wrong format. The system will require a password with 8 characters including number and a symbol. If the user password doesn't meet the requirements, the user account will not register. Last, when confirm password does not match with the user password, it will alert the user password does not match.

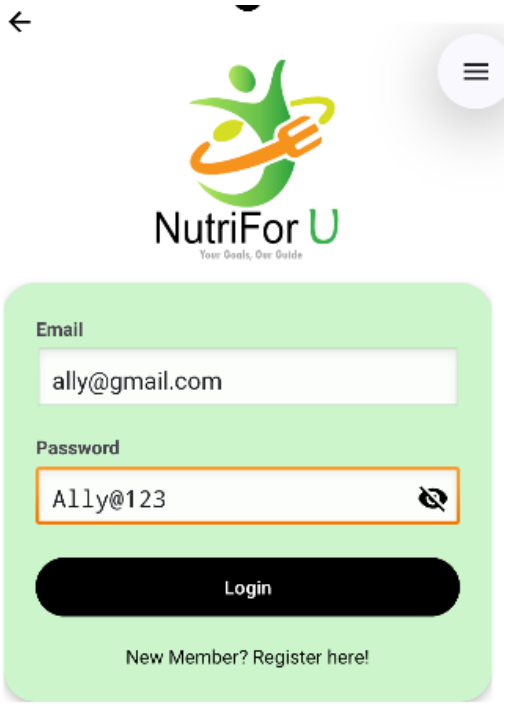


Figure 5.28 Login Account to the System

After registering an account, system will bring back the user to login page to login to the account just created by them to login to the system by entering their email address and password. After entering system, the system will require the user to key in their personal information such as gender, height, weight, target BMI and allergic foods to complete the user profile.

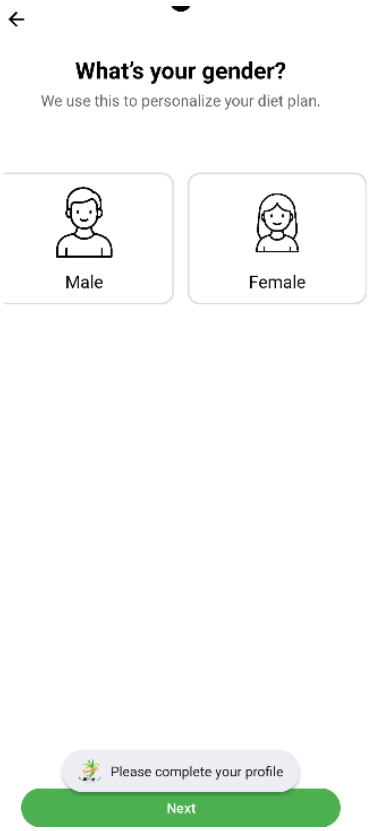


Figure 5.29 Gender selection

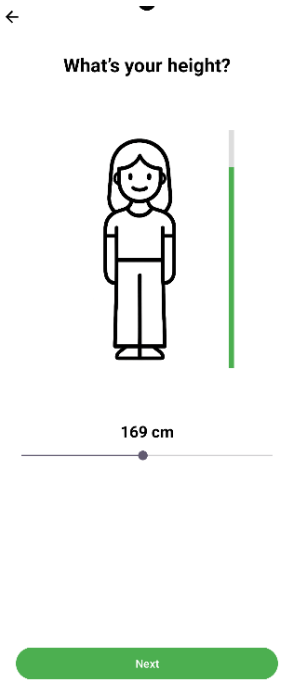


Figure 5.30 Height input

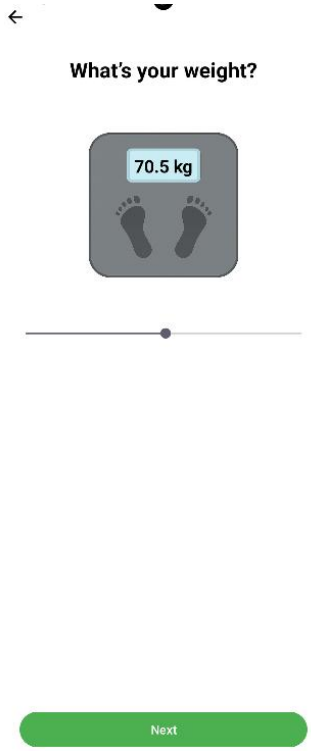


Figure 5.31 Weight input

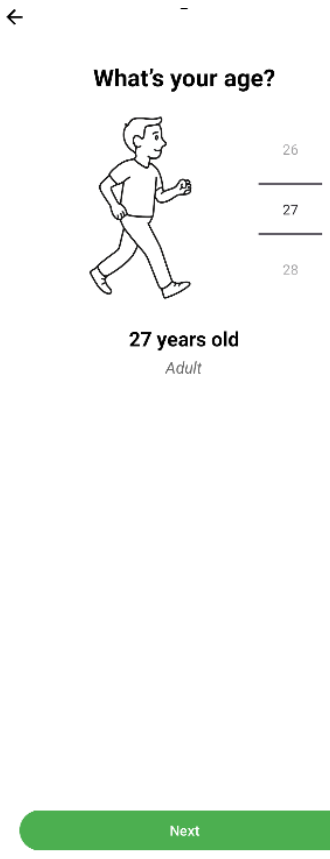


Figure 5.32 Age input

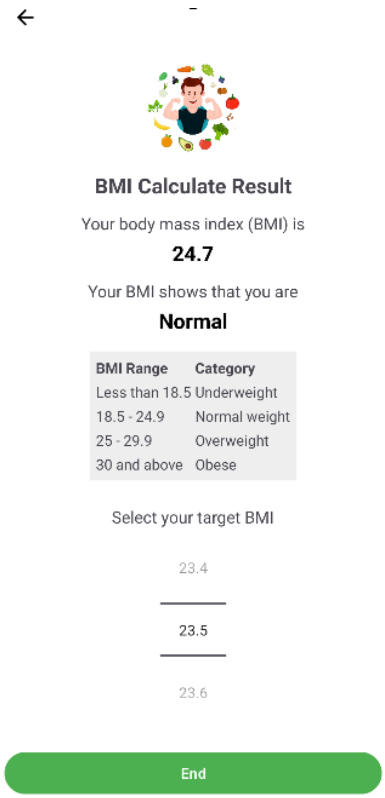


Figure 5.33 Current User BMI will be shown and Target BMI input

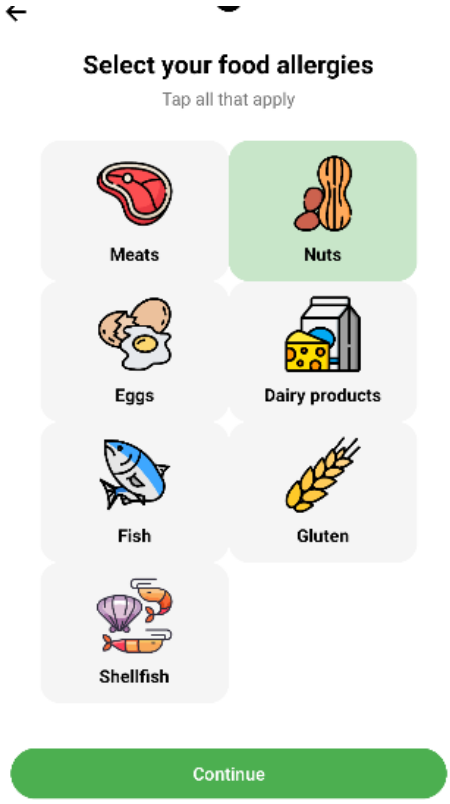


Figure 5.34 Food Allergic Selection

5.2.4 Home Page

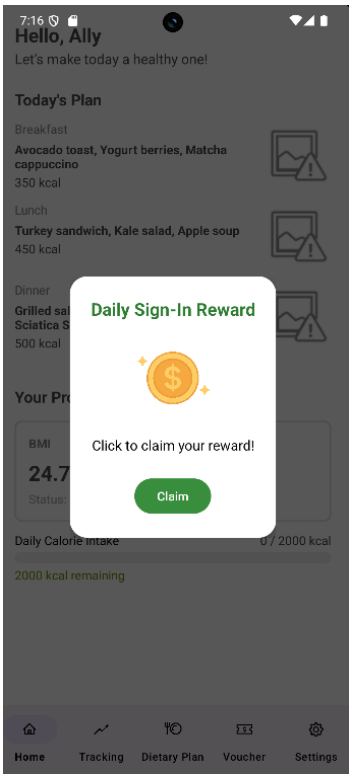


Figure 5.35 Daily Sign-in reward pop up

The system has a daily sign-in reward for user as an engagement tool. Users can claim their reward point daily when they enter the system home page.

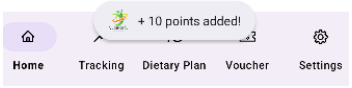


Figure 5.36 After claiming the reward point

A “+10 point added” prompt will be show to the user after claiming the reward point. The system will reward 10 points daily for users to claim for redeem voucher in the system.

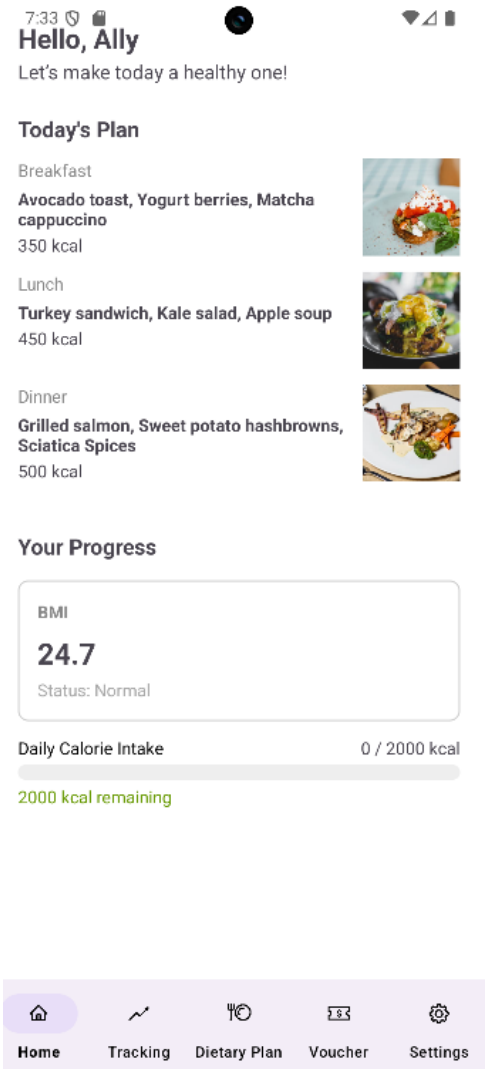


Figure 5.37 Home Page of the system

The home page will be showing a greeting text to the user by having the username and some motivation text at the top. At the middle of the page, will be showing the today dietary plan which is automatically created by the system. The dietary plan includes breakfast, lunch, dinner and calories for each type of meal.

Below the dietary plan have the user progress which shows the user current BMI and daily calorie intakes. The daily calorie intake will help the user to log their calorie intake after taking their meals, to let the users know how many more calories have been taken. A check box will appear for users to log their meal progress after users have selected their today dietary plan.

Bottom of the system, have the system navigation bar which can switch between home page, health tacking page, dietary plan page, voucher page and settings page.

5.2.5 Health Tracking Page

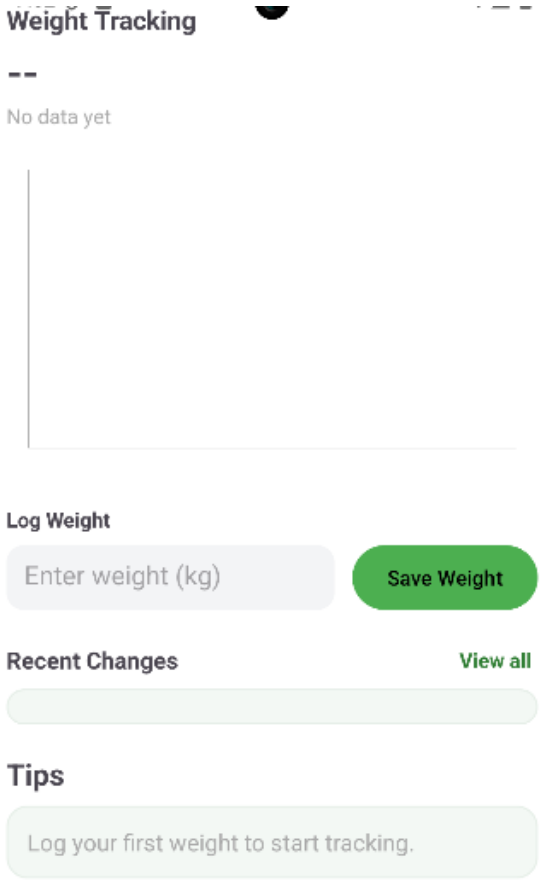


Figure 5.38 Health Tracking Page

The health tracking page is empty since the user hasn't logged any weight yet. The health tracking page shows a better visualization graph for users to better understand their current weight change. Below has the input box for users to log their weight and save weight button. Besides, the recent changes table will be shown to the user to let user know their historical weight change.

To let the system UI to not seem to compact by showing all the historical data on the same page, there is a "view all" button for user to look back their all-historical weight data. Lastly, A tips text motivation box was shown below, to motivate the user. The tips will change based on the weight change situation, whether the weight is increasing, decreasing or reaching the user BMI target.

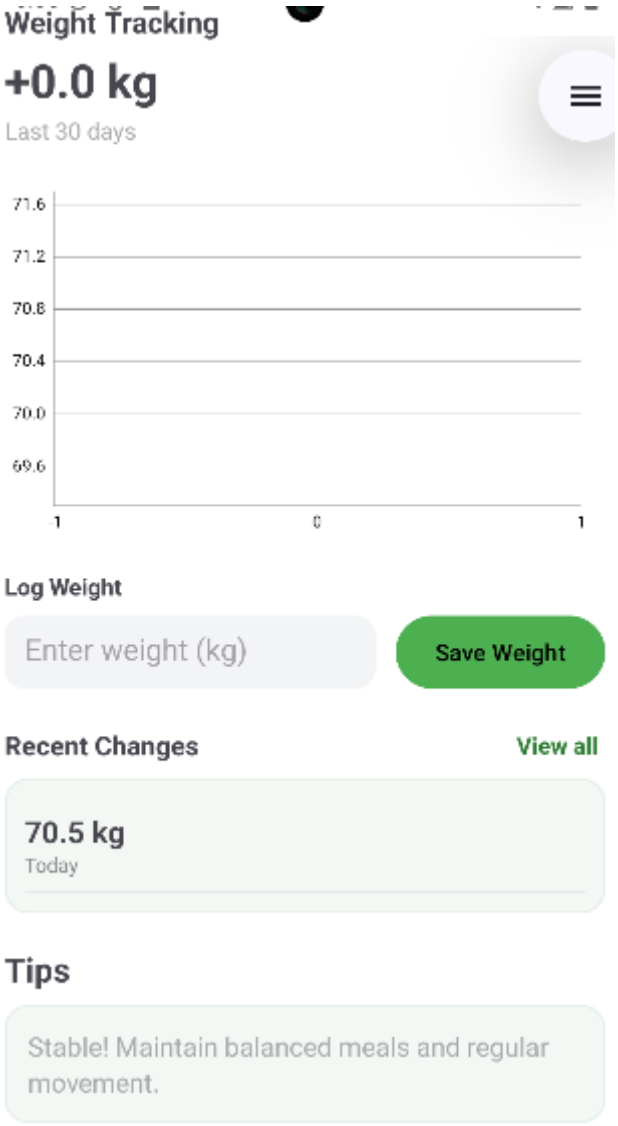


Figure 5.39 Logging weight

The graph table will only show the weight data within 30 days, when it exceeds the 30 days the graph will be renewed.

5.2.6 Dietary Plan

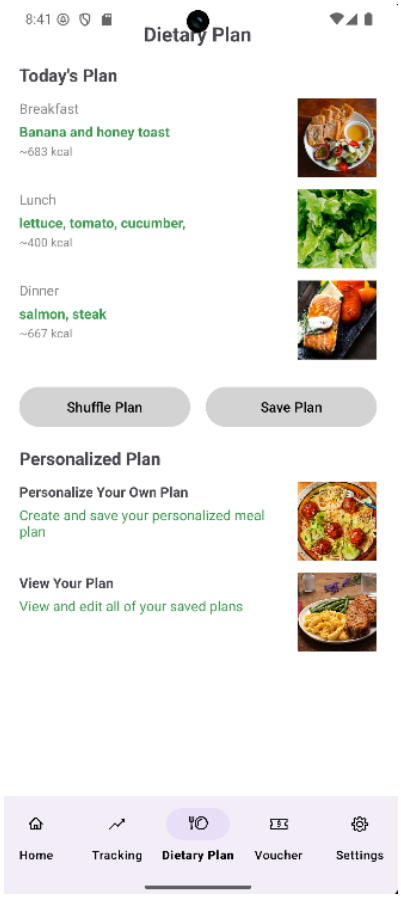


Figure 5.40 Dietary Plan Page

The dietary plan page shows the today plan that generate by the system. A shuffle button for users to shuffle the plan when they are not satisfied with the dietary plan given by the system and a save button for users to save the dietary plan.

Besides, in the dietary plan page also provides the function for users to personalize their own dietary plan and view back the saved dietary plan.

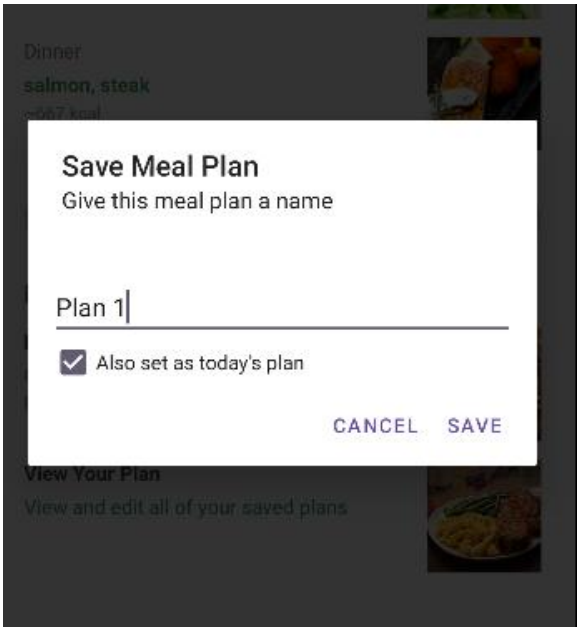


Figure 5.41 Save the dietary plan

Users need to input a meal plan name to save the dietary plan, and users also can choose whether to set the dietary plan as today's plan. In figure 5.41 example shows to set the dietary plan were as today plan.

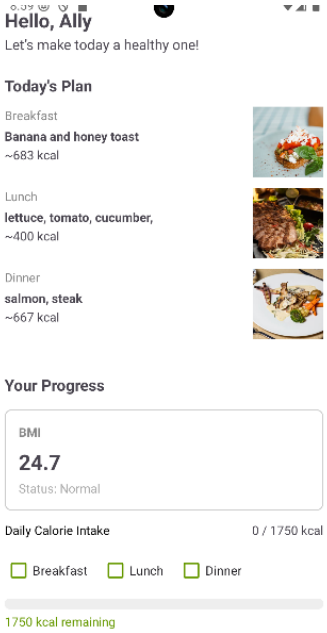


Figure 5.42 Home page today dietary plan updated

After it has been set as today's plan, the home page today dietary plan will change to the user saved dietary plan and the calories progress function will show up for user to log their meals.

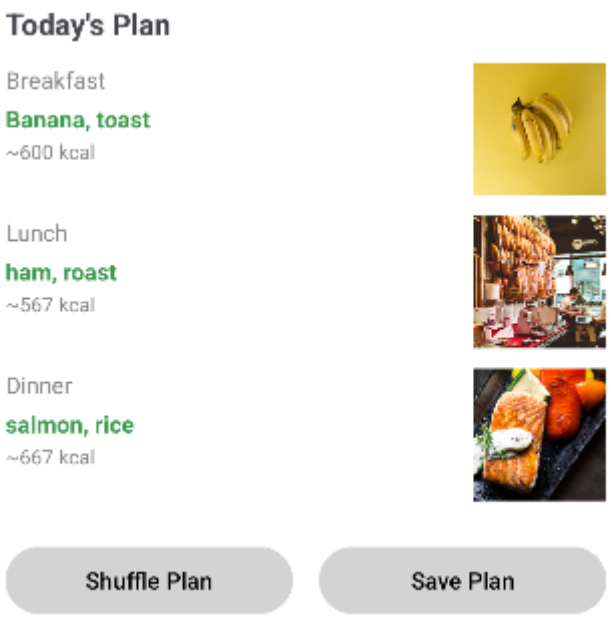


Figure 5.43 Shuffle the Dietary Plan

If user does not stratify with the dietary plan, user can click the shuffle plan button to shuffle the plan.

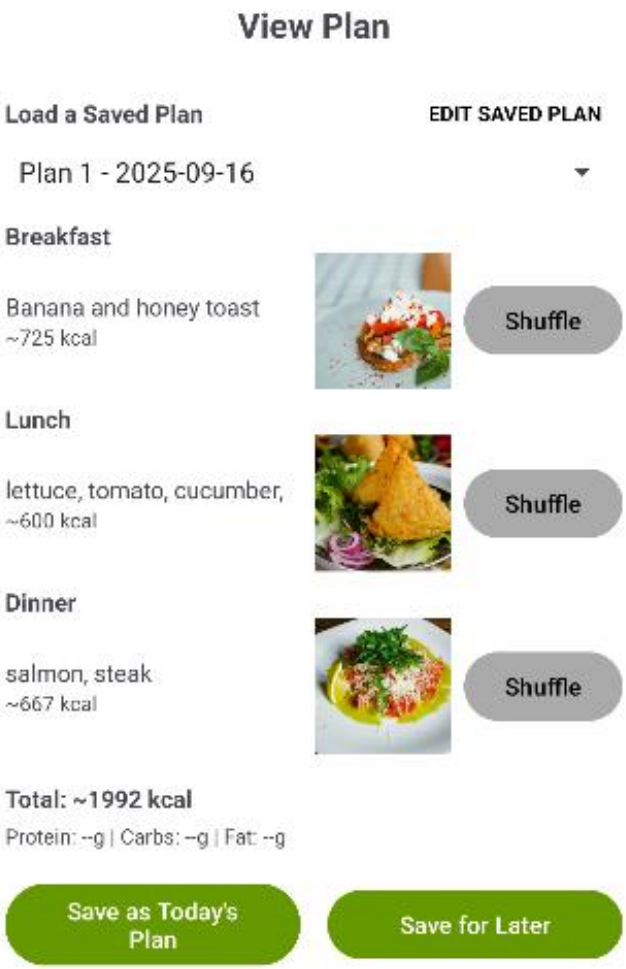


Figure 5.44 View User Plan Page

In view the user plan page will show the saved dietary plan. Users can modify with the saving plan by shuffling each of the meals when it is not satisfied. After being modified, users can save it for later or set as today's plan.

Figure 5.45 Edit Saved Plan

On the top right of the page have the edit saved plan button. When clicked on it, users can view back the saved plan and do modifications by changing the meal plan name or deleting the saved plan.

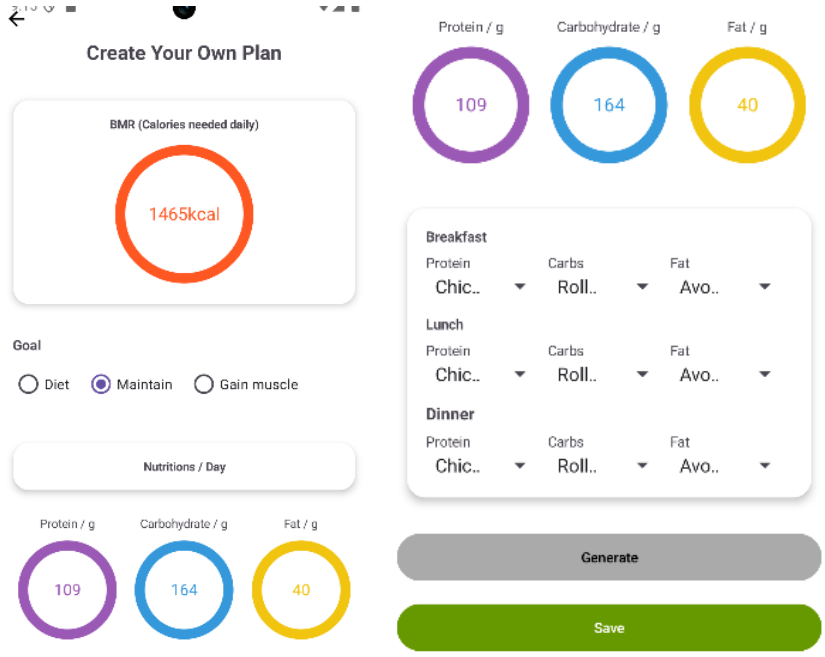


Figure 5.46 Personalize User Own Plan Page

Users can personalize their own plan when they don't want the shuffle dietary plan by the Cohere AI. Users can choose the plan goal such as diet, maintain and gain muscle. Choosing the plan goal will modify the protein, carbohydrate and fat needed in gram. So, the user can choose the dietary plan they like, and system provide the needed nutrition in gram. But since the food in the meal plan is manually key into the system there will be a limit selection for the users.

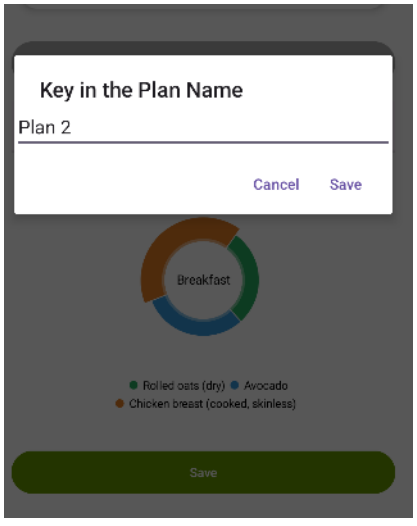


Figure 5.47 Save the personalization plan

After users choose their dietary plan, they can name the plan and save it.

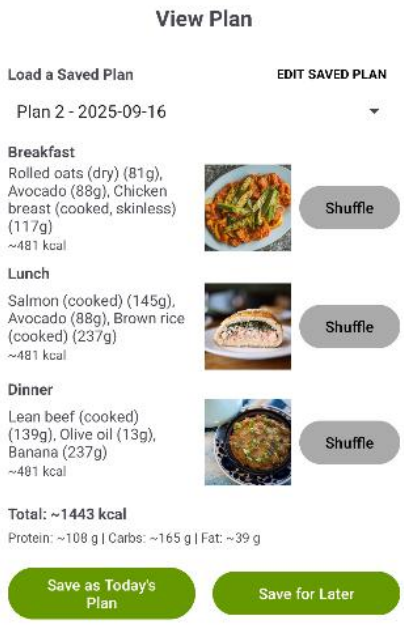


Figure 5.48 View back the saved plan, set as today's plan

Users can view back the saved personalize dietary plan. And let's set the personalized dietary plan as today's plan. When we go back to home page, today's dietary plan is updated.

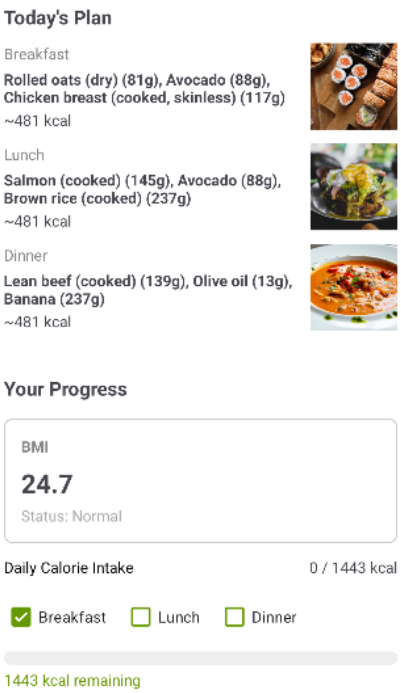


Figure 5.49 Today Dietary plan is updated

When we go back to home page, today's dietary plan is updated.

5.2.7 Voucher Page

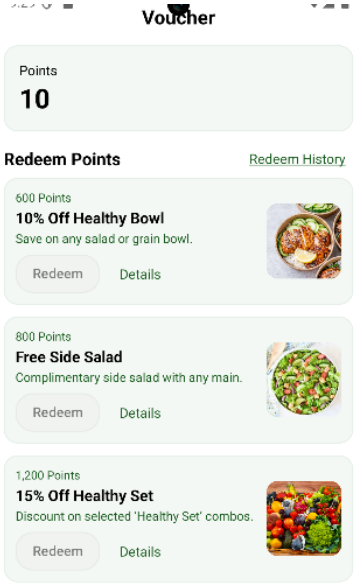


Figure 5.50 Voucher Page

The top of the voucher page will show the collected reward point daily by the users. In the middle of the voucher page, show the available voucher reward for the users to redeem.

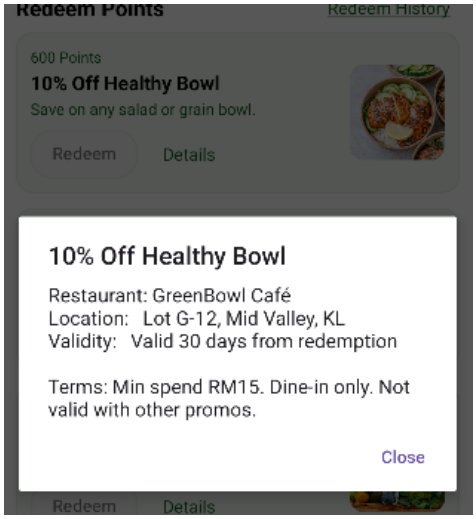


Figure 5.51 Details of the Voucher Rewards

Each of the voucher rewards has the details button for users to see the details of the voucher reward. Details such as restaurant name, location, terms and conditions and expiration dates.

Figure 5.52 Redeem History

When users have redeemed a voucher, users can look back the voucher on the redeem history at the top right corner.

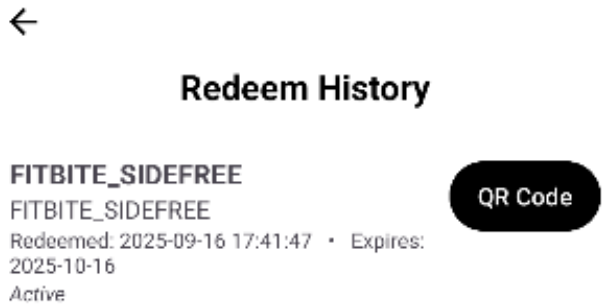


Figure 5.53 Redeem the code by scanning the QR code

For example, if the users have redeemed the code. In the redeem history, the details of the voucher rewards will also show.

Figure 5.54 QR code of the voucher rewards

There will be a QR code button to show the QR code to redeem the code at the restaurant.

5.2.8 Setting Page

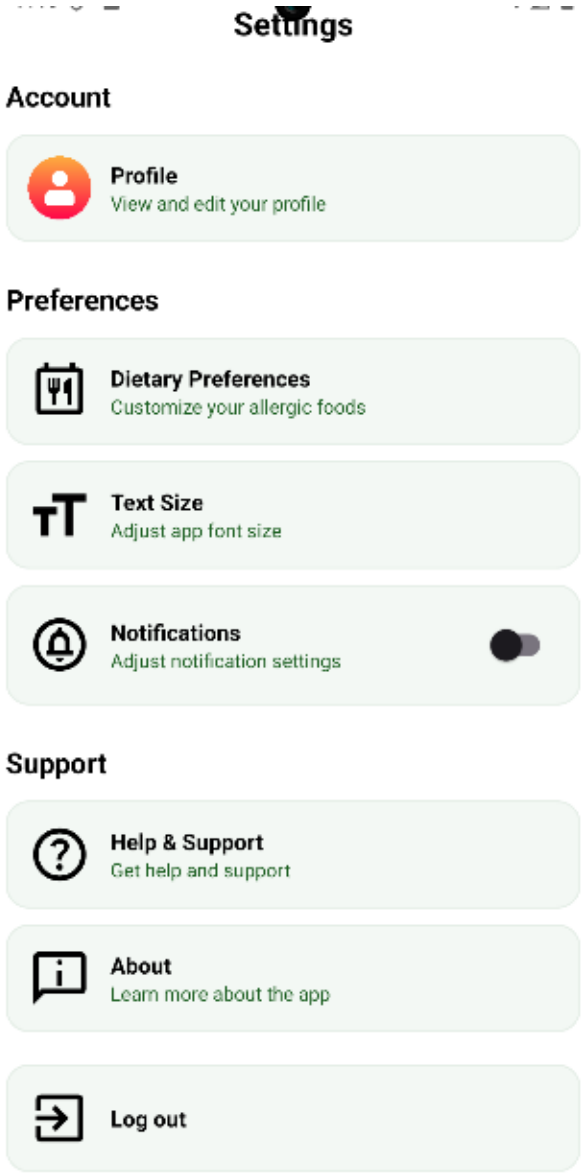


Figure 5.55 Setting Page

In the setting page, users can edit their profile information, edit the dietary preference, text size of the system and the notification settings. Besides, it also provides some support information for the users to read such as help & support and about information of the system.

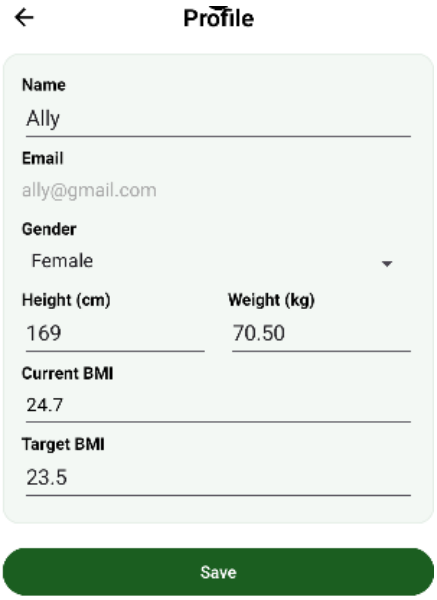


Figure 5.56 Edit Profile

Users can edit their profile by changing their username, gender, height, weight, current BMI and target BMI.

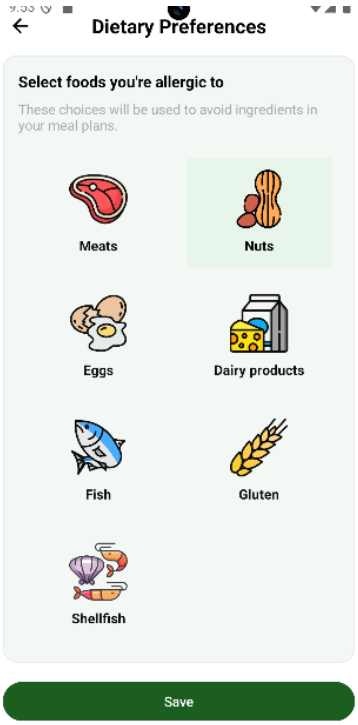


Figure 5.57 Edit Dietary Preferences

Users can change their allergic foods while they can choose more than one food or not choose any of the food.

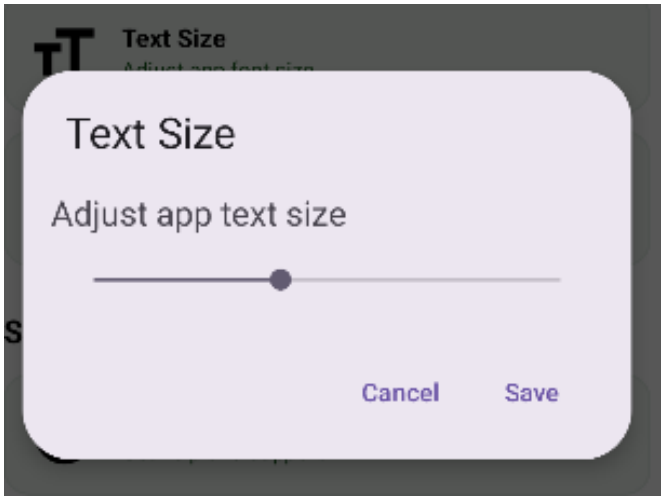


Figure 5.58 Edit Text Size

Users can edit the text size of the system. This function helps the users who have difficulty with eyesight.

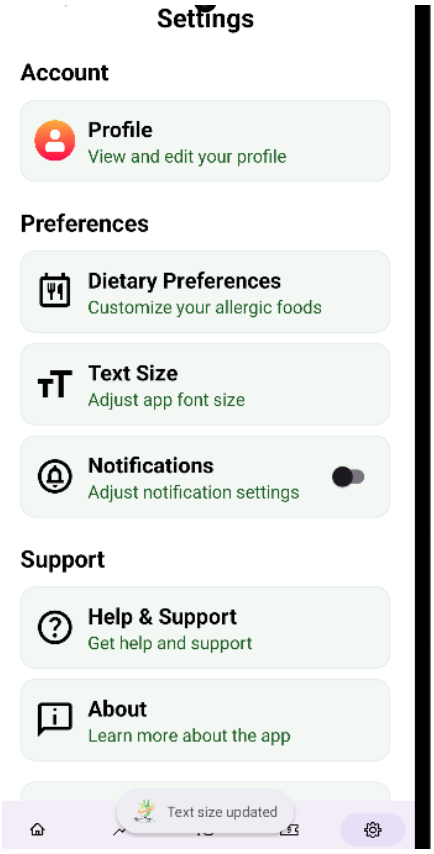


Figure 5.59 System Text Size Changed

After click save, the system text size has changed more bigger than previous system text size.

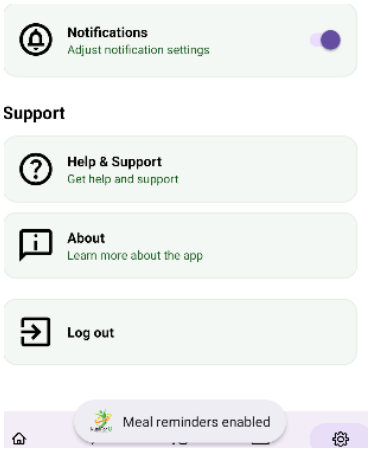


Figure 5.60 Notification Settings

Users can put on and off the notifications setting of the system. The system will send the meal taking time notifications to the user to have their meals on time.

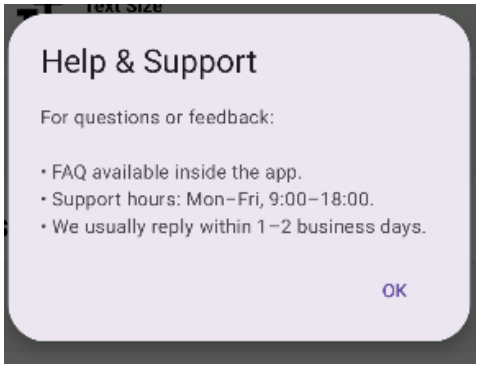


Figure 5.61 Help & Support

Users can view the information help & support of the system.

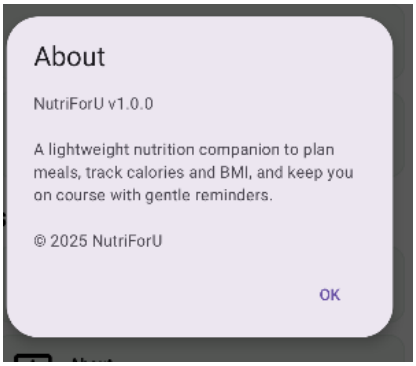


Figure 5.62 About

Users can view the information about the system.

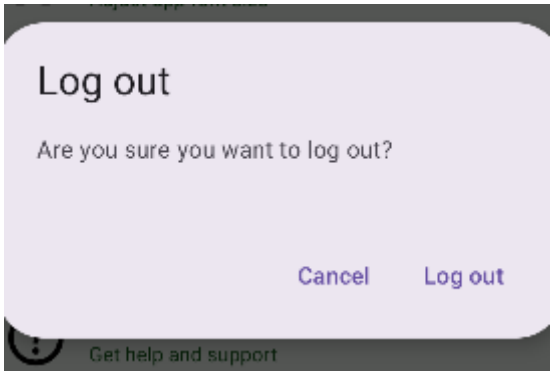


Figure 5.63 Logout system

In setting pages, users can log out the account, after that system will bring back the user to the login page. Users can log in back again.

5.3 Comment and highlight the feasibility of the proposed method

5.3.1 The Dietary Plan Activity

```
2 usages
public class CohereApiClient {
    1 usage
    private static final String BASE_URL = "https://api.cohere.ai/v1/";

    2 usages
    private Retrofit retrofit;

    1 usage
    public CohereApiClient() {
        // Creating a log interceptor
        HttpLoggingInterceptor logging = new HttpLoggingInterceptor();
        logging.setLevel(HttpLoggingInterceptor.Level.BODY);

        // Create OkHttpClient and add interceptor
        OkHttpClient client = new OkHttpClient.Builder()
                .addInterceptor(logging)
                .build();

        // Construct a Retrofit instance, remember to add .client(client)
        retrofit = new Retrofit.Builder()
                .baseUrl(BASE_URL)
                .client(client)  // ✅ Don't forget this line
                .addConverterFactory(GsonConverterFactory.create())
                .build();
    }

    1 usage
    public CohereService createService() { return retrofit.create(CohereService.class); }
}
```

Figure 5.64 Shows the System call the function of Cohere Ai by API

The CohereApiClient class implements Java functionality through which it configures network clients based on Retrofit library requirements for Cohere AI API access. All API calls start from the base Uniform Resource Locator (URL) defined as https://api.cohere.ai/v1/.

Inside its construction phase the HttpLoggingInterceptor creates a debugging tool which logs complete network request and response data. The network client uses an OkHttpClient where this interceptor can be added for handling network communication.

The Retrofit instance requires its build parameters to contain the base URL and OkHttpClient instance and GsonConverterFactory for JSON data conversion. Retrofit through createService() manufactures an interface implementation for CohereService that enables application sections to conduct Cohere API requests effortlessly without direct Hypertext Transfer Protocol (HTTP) management.

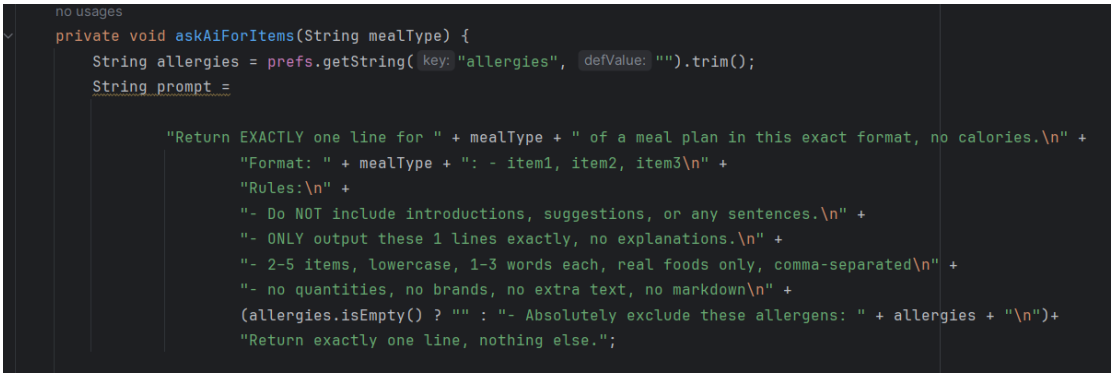


Figure 5.65 Shows the command prompt of dietary plan for Cohere AI

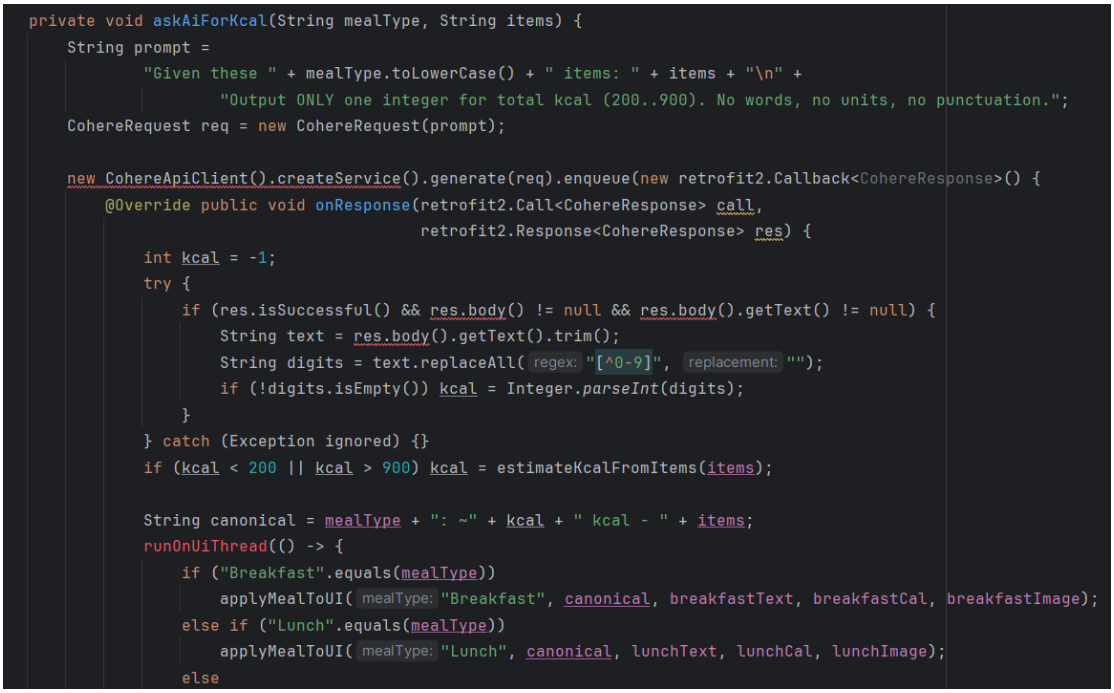
In the system will be using the Cohere AI, the system will generate a prompt message to the Cohere AI server. The Cohere AI server will return a dietary plan to the system. The command prompt required a strictly clear information, otherwise the output might not reach the format looks like.

```java
private int estimateKcalFromItems(String text) {
    // very light heuristic; same feel as PersonalizedplanActivity
    String itemsPart = text;
    int dashIdx = text.indexOf(" - ");
    if (dashIdx >= 0 && dashIdx + 3 < text.length()) itemsPart = text.substring( beginIndex: dashIdx + 3);
    itemsPart = itemsPart.replaceFirst( regex: "(?i)^(breakfast|lunch|dinner)\\s*:\\s*", replacement: "");
    String[] tokens = itemsPart.split( regex: "[,•|]");
    int total = 0, matched = 0;
    Map<String,Integer> HINTS = new HashMap<>();
    HINTS.put("oats",150); HINTS.put("banana",100); HINTS.put("milk",120); HINTS.put("almond milk",40);
    HINTS.put("chicken",220); HINTS.put("breast",165); HINTS.put("rice",200); HINTS.put("brown rice",216);
    HINTS.put("egg",78); HINTS.put("eggs",156); HINTS.put("salad",80); HINTS.put("salmon",230);
    HINTS.put("noodles",300); HINTS.put("noodle",250); HINTS.put("yogurt",120); HINTS.put("apple",95);
    for (String raw : tokens) {
        String item = raw.trim().toLowerCase();
        int best=0;
        for (String k : HINTS.keySet()) if (item.contains(k)) best = Math.max(best, HINTS.get(k));
        if (best > 0) { total += best; matched++; }
    }
    if (matched == 0) return 350;
    return Math.max(150, Math.min(total, 900));
}
```

Figure 5.66 Shows the Estimate Kcal from Item

estimateKcalFromItems is a heuristic-based algorithm in an Android Java program, which estimates the number of calories (kcal) in a meal using a text description of food items. It replicates the item list to be extracted by deleting the text prefix of a meal type and breaking the text into tokens with the help of delimiters such as commas or bullets. The role involves a fixed record known as HINTS that is used to match normal foodstuffs with their approximate calorie content.

It reads the tokens one at a time, compares each one to the HINTS keys and adds up the maximum number of calories corresponding to any food item. When there is a lack of matching items, it will be 350 kcal default. The maximum calorie is limited to 150-900 kcal to guarantee a reasonable production. This role is a backup to estimate the calories in case askAiForKcal AI-driven techniques fail or give untrustworthy results in a BMI/health tracker application.



```
private void askAiForKcal(String mealType, String items) {
    String prompt =
            "Given these " + mealType.toLowerCase() + " items: " + items + "\n" +
                    "Output ONLY one integer for total kcal (200..900). No words, no units, no punctuation.";
    CohereRequest req = new CohereRequest(prompt);

    new CohereApiClient().createService().generate(req).enqueue(new retrofit2.Callback<CohereResponse>() {
        @Override public void onResponse(retrofit2.Call<CohereResponse> call,
                                         retrofit2.Response<CohereResponse> res) {
            int kcal = -1;
            try {
                if (res.isSuccessful() && res.body() != null && res.body().getText() != null) {
                    String text = res.body().getText().trim();
                    String digits = text.replaceAll( regex: "[^0-9]",  replacement: "");
                    if (!digits.isEmpty()) kcal = Integer.parseInt(digits);
                }
            } catch (Exception ignored) {}
            if (kcal < 200 || kcal > 900) kcal = estimateKcalFromItems(items);

            String canonical = mealType + ": ~" + kcal + " kcal - " + items;
            runOnUiThread(() -> {
                if ("Breakfast".equals(mealType))
                    applyMealToUI( mealType: "Breakfast", canonical, breakfastText, breakfastCal, breakfastImage);
                else if ("Lunch".equals(mealType))
                    applyMealToUI( mealType: "Lunch", canonical, lunchText, lunchCal, lunchImage);
                else
```

Figure 5.67 Show ask the AI for Kcal

The askAiForKcal method of this Android Java application approximates the total number of calories in a particular meal using a list of products. It builds an invitation towards an AI service of Cohere API requesting one integer output between 200-900 kcal, extracts the numeric digits in the response text, and reads it as an integer. In case the API call is successful but the value extracted is invalid less than 200 or more than 900 kcal, empty, or completely fails, it resorts to a local estimation technique in estimateKcalFromItems. This makes sure that there is consistent calorie calculation with AI supplement and elegant error management of a BMI/health tracking app.

```
private void fetchMacrosFromAI(String mealType, String itemsDisplay, int kcal) {
    if (itemsDisplay == null || itemsDisplay.trim().isEmpty()) return;

    String prompt =
            "You are a nutritionist. Estimate macronutrients for this single meal.\n" +
                    "Items: " + itemsDisplay + "\n" +
                    "Total energy: " + Math.max(200, kcal) + " kcal\n" +
                    "Return ONLY JSON with INTEGER grams (no prose, no markdown):\n" +
                    "{\"protein_g\": P, \"carbs_g\": C, \"fat_g\": F}\n" +
                    "Constraints: 4*P + 4*C + 9*F must be within 15% of the total kcal.";

    CohereRequest req = new CohereRequest(prompt);
    new CohereApiClient().createService().generate(req).enqueue(new retrofit2.Callback<CohereResponse>(
        @Override public void onResponse(retrofit2.Call<CohereResponse> call,
                                         retrofit2.Response<CohereResponse> res) {
            int p=0,c=0,f=0;
            try {
                if (res.isSuccessful() && res.body() != null) {
                    String raw = res.body().getText();
                    String json = findFirstJsonObject(raw);
                    if (json != null) {
                        JSONObject o = new JSONObject(json);
                        p = Math.max(0, o.optInt( name: "protein_g",  fallback: 0));
                        c = Math.max(0, o.optInt( name: "carbs_g",    fallback: 0));
                        f = Math.max(0, o.optInt( name: "fat_g",      fallback: 0));
                        int energy = 4*p + 4*c + 9*f;
                        if (p <= 0 || c <= 0 || f <= 0 || Math.abs(energy - kcal) > 0.15*kcal) {
                            int[] local = computeMacrosLocal(mealType, kcal);
```

Figure 5.68 Show the Macro Nutrition Calculate from AI

The fetchMacrosFromAI method of this Android Java code approximates the macronutrients in protein, carbs and fat in a meal based on the description of the meal, the total calories and type of meal. It makes a request to an AI service through Cohere API requesting the macronutrient values in the form of a JSON with the condition that the calculated energy formula 4protein + 4carbs + 9fat is not more than 15% of the offered kcal.

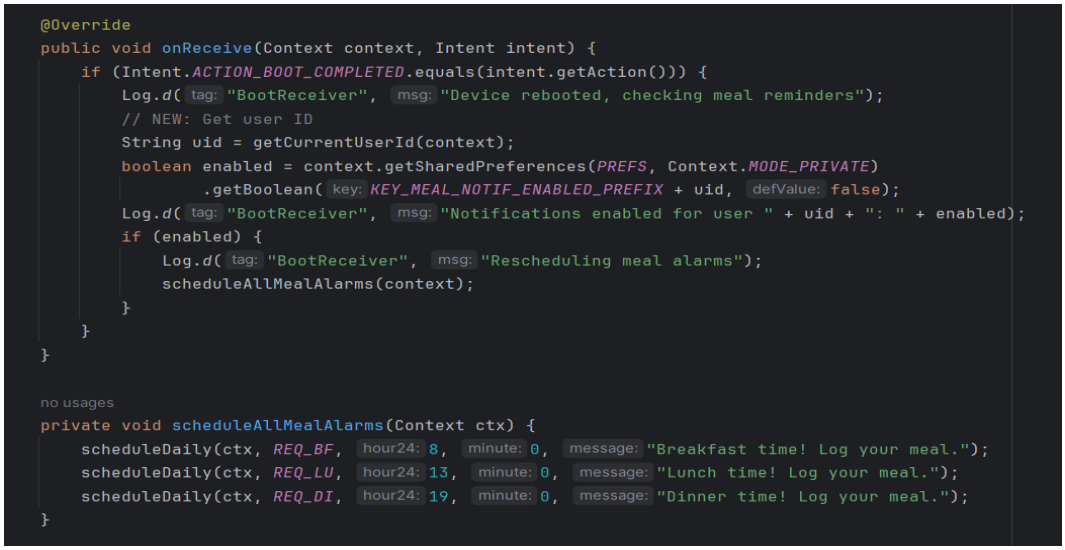
In case the API call is successful and the response is a valid JSON with non-negative values that satisfy the energy constraint, those macronutrients are used; otherwise, a local method of computation is used in computeMacrosLocal. The ensuing macronutrients are stored in an array of the meal on the UI thread that further can be used later. The function treats errors, bad responses or API failure, or makes use of the local fall-back method.

```
            return;
        }
        try {
            String json = response.body().string();
            org.json.JSONObject obj = new org.json.JSONObject(json);
            org.json.JSONArray photos = obj.optJSONArray( name: "photos");
            if (photos == null || photos.length() == 0) return;

            // Pick the first photo whose alt looks like FOOD (not hands/people/drinks)
            int chosen = -1;
            String queryLower = query.toLowerCase();
            for (int i = 0; i < photos.length(); i++) {
                org.json.JSONObject p = photos.getJSONObject(i);
                String alt = p.optString( name: "alt",  fallback: "").toLowerCase();

                boolean looksLikeFood =
                        alt.contains("food") || alt.contains("dish") || alt.contains("meal") ||
                                alt.contains("plate") || alt.contains("bowl") || alt.contains("salad") ||
                                alt.contains("rice")  || alt.contains("noodle") || alt.contains("pasta") ||
                                alt.contains("chicken") || alt.contains("salmon") || alt.contains("egg") ||
                                alt.contains("tofu") || alt.contains("beef") || alt.contains("vegetable") ||
                                alt.contains(queryLower);

                boolean mentionsPeople =
                        alt.matches( regex: ".*\\b(man|woman|people|person|friends|couple|hands)\\b.*");

                boolean drinkOnly =
                        alt.contains("wine") || alt.contains("coffee") || alt.contains("tea") ||
```

Figure 5.69 Show the function of dietary plan fetch meal image

In the dietary plan system, we will use Pexels API to get the dietary plan image into the system. While the Pexels API is not an AI that required a command prompt to understand the question. The dietary image will be generated based on the dietary plan keywords. The keywords must be relevant to the food or otherwise it will appear irrelevant image. Therefore, we need to use If else loop to avoid some irrelevant image such like people, restaurant, scenery and more.

5.3.2 Notification Activity



```
@Override
public void onReceive(Context context, Intent intent) {
    if (Intent.ACTION_BOOT_COMPLETED.equals(intent.getAction())) {
        Log.d( tag: "BootReceiver",  msg: "Device rebooted, checking meal reminders");
        // NEW: Get user ID
        String uid = getCurrentUserId(context);
        boolean enabled = context.getSharedPreferences(PREFS, Context.MODE_PRIVATE)
                .getBoolean( key: KEY_MEAL_NOTIF_ENABLED_PREFIX + uid,  defValue: false);
        Log.d( tag: "BootReceiver",  msg: "Notifications enabled for user " + uid + ": " + enabled);
        if (enabled) {
            Log.d( tag: "BootReceiver",  msg: "Rescheduling meal alarms");
            scheduleAllMealAlarms(context);
        }
    }
}

no usages
private void scheduleAllMealAlarms(Context ctx) {
    scheduleDaily(ctx, REQ_BF,  hour24: 8,  minute: 0,  message: "Breakfast time! Log your meal.");
    scheduleDaily(ctx, REQ_LU,  hour24: 13,  minute: 0,  message: "Lunch time! Log your meal.");
    scheduleDaily(ctx, REQ_DI,  hour24: 19,  minute: 0,  message: "Dinner time! Log your meal.");
}
```

Figure 5.70 Show the Boot Receiver of the system

BootReceiver is an Android BroadcastReceiver that is waiting to receive the ACTION_BOOT_COMPLETED system broadcast which is sent on the device after it has finished booting. It works to reactivate meal reminder alarms to 8:00 AM breakfast, 1:00 PM lunch, and 7:00 PM dinner in an application since alarm is cleared upon reboot.

It verifies user-specific notification preferences with the help of SharedPreferences and, in the case they are enabled, it makes use of AlarmManager to set daily alarms and, naturally, puts messages in them with user-specific messages. The implementation will manage Android version compatibility, multiple person usage and alarms should be set in the devices time zone so that the same reminders are set without any intervention of the user.

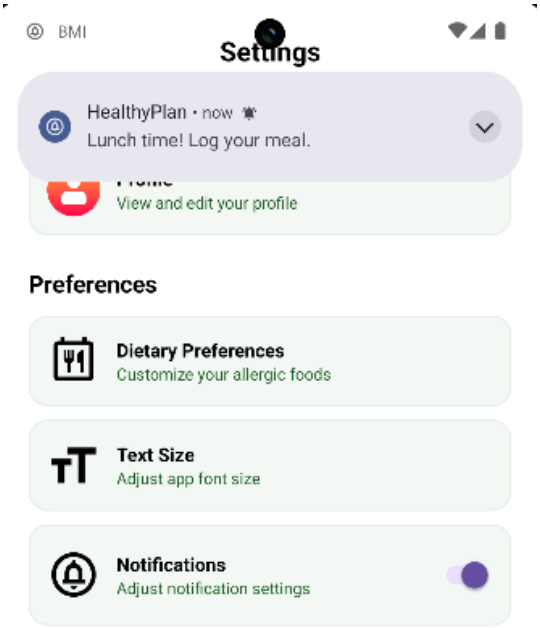


Figure 5.71 Notification Appear when reach specific time.

The meal reminder alarms will be out at 8:00 AM breakfast, 1:00 PM lunch, and 7:00 PM dinner in the application system.

5.3.3 Health Tracking Activity

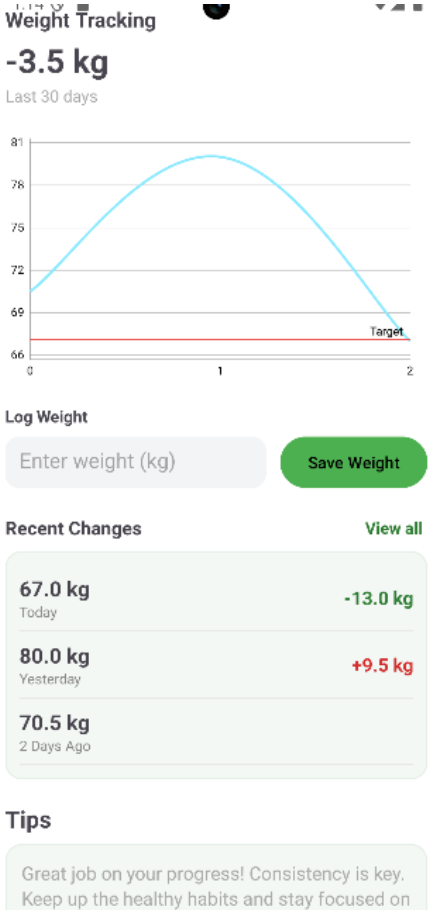


Figure 5.72 Health Tracking Activity

The figure 5.81 health tracking graph example show the weight change in blue line and weight to reach the users target BMI in red line. Besides, when user log their weight, the recent changes will calculate the weight change is increase or decrease in kg.

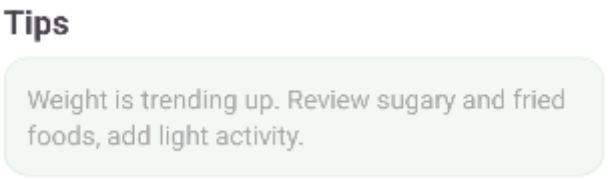


Figure 5.73 Tips motivation text when weight increases

The tips world text will be change when the user's weight is increased and it will alert the users and given some advice information.

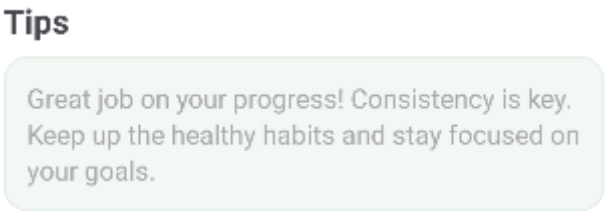


Figure 5.74 Tips motivation text when weight decreases

When the user's weight is decreased. The tips world text will show motivate text to the users to keep the consistency to reach the user's target and some advice tips.

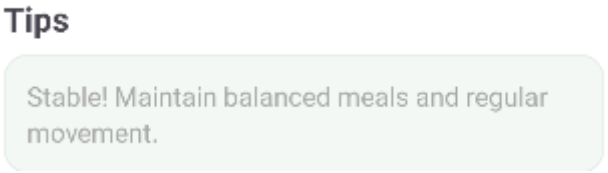


Figure 5.75 Tips motivation text when weight maintain

When the user's weight remains unchanged, but user's current BMI is in normal. The tips world text will show the user's health condition is stable and keep maintaining a stable good health condition.

Chapter 6 System Evaluation and Discuss

The chapter is a review of the Personalize Nutrition and Obesity Intervention System Mobile Application to determine whether it achieves its functional and usability goals or not. It uses three techniques in evaluation that include black box testing, user testing and objective evaluation. Black box testing evaluates the functionality of the system with no information about the inner code of the system, resulting in the evaluation of the input-output behavior. User testing takes the opinions of actual users and can be used to determine usability and user experience. Objective evaluation also compares the system with predetermined objectives, where the outcomes include performance and accuracy. This chapter addresses the methodologies, findings and conclusions of these evaluations, their strengths, weaknesses and improvement opportunities.

6.1 Black Box Testing

6.1.1 Overview

Black box testing was involved to ensure that Personalize Nutrition and Obesity Intervention System Mobile Application works according to what it was meant to but without making any reference to the internal code structure. The testing involved the input given to the app and outputs, and all the features were tested to work properly in different situations. The test cases were created to address all essential features including the feature of login, sign-up, user preferences, home page, health tracking, dietary plan, voucher, and environment setup.

6.1.2 Test Strategy

The test plan was systematic in testing the behaviour of the system:

- **Test Scope**: All user interface features of the mobile application system.

- **Test Levels**: Functional testing at the login page, registration page, health tracking page, dietary plan page, voucher page, setting page and system navigation flow.

- **Tools**: Manual testing using android devices in android studio software, with test cases documented in a spreadsheet.

- **Techniques**: Equivalence partitioning such as testing valid input and invalid input, and boundary value analysis such as testing edge cases.

6.1.3 Test Cases and Results

Below is a summary of test cases and outcomes:

Table 6.1 Summary of test cases and outcomes

Module	Test Case	Input	Expected Output	Actual Output	Status
Login	Valid login credentials	Email: test@gmail.com, Password: Test@123	Redirect to Home page with greeting message	As expected	Pass
Login	Invalid password	Email: test@gmail.com, Password: test@1234 (Wrong)	Error message: "Incorrect email or password"	As expected	Pass
Sign-Up	Complete sign-up with valid preferences	Gender: Male, Weight: 70kg, Height: 170cm, Allergies: None, Target BMI: 22	Redirect to Home page after saving preferences	As expected	Pass
Sign-Up	Missing required fields (e.g., height)	Gender: Female, Weight: 60kg, Height: (empty)	Error message: "Please fill	As expected	Pass

			all required fields"		
Home Page	Display daily dietary plan and BMI	User logs in with saved preferences	Shows greeting, today's plan, current BMI (e.g., 23.5), and calorie intake (e.g., 1800 kcal)	As expected	Pass
Health Tracking	Log weight and update graph	Enter weight: 68kg	Graph updates, shows target BMI line, motivational message (e.g., "Great progress!")	As expected	Pass
Health Tracking	Enter invalid weight (e.g., negative)	Weight: -5kg	Error message: "Invalid weight value"	As expected	Pass
Dietary Plan	Shuffle dietary plan	Click "Shuffle Plan" button	New plan displayed with updated image	As expected	Pass
Dietary Plan	Save personalized plan	Add custom meal: "Grilled Chicken Salad"	Plan saved and displayed in	As expected	Pass

			personalized section		
Voucher	Redeem voucher with sufficient points	Points: 100, Select voucher (50 points)	Voucher redeemed, points reduced to 50, history updated	As expected	Pass
Voucher	Attempt redemption with insufficient points	Points: 20, Select voucher (50 points)	Error message: "Insufficient points"	As expected	Pass
Settings	Update dietary preferences (allergies)	Change allergies to "Peanuts"	Preferences updated, dietary plans exclude peanuts	As expected	Pass
Settings	Adjust text size	Increase text size to "Large"	App text size increases across all pages	As expected	Pass
Settings	The notification toggle	Turn on the notification toggle	Meal reminder enable prompt is shown	As expected	Pass

6.1.4 Discussion

Black box testing was used to ensure that all the modules of Personalize Nutrition and Obesity Intervention System Mobile Application fulfil its functional requirements. The

app was robust in that it allowed valid inputs to be processed and gave reasonable error messages when an invalid input is provided. As an illustration, the login module denied wrong credential and the health tracking module denied invalid weight input. Nevertheless, the test did not leave without minor problems, including the slight lag in the display of the weight graph on older gadgets which can be used to suggest the necessity to optimize the performance. The test cases revealed no major faults and so that the system was stable.

6.2 User Testing

6.2.1 Overview

User testing was also done to evaluate the usability, user experience and satisfaction of Personalize Nutrition and Obesity Intervention System Mobile Application. The participants were put on the app a week where they were asked to complete activities like registering, entering their weight, creating weight-based diets, and voucher collection. The feedback was made through a survey in the form of a Google Form using rating scale 1-5 each on the module as shown in Chapter 6 of this report.

6.2.2 Methodology

- **Participants:** 15 UTAR FICT IB students
- **Tasks:**
  1. Create an account and set up the user profile.
  2. Log the weight and review the health tracking graph.
  3. Shuffle and personalize a dietary plan.
  4. Redeem a voucher using earned points from the daily sign-in reward.
  5.Update the settings such as allergic food and text size.
- **Data Collection:** A 40-question survey with 5 question per module, excluding demographics using a 1-5 rating scale which 1 is poor, 5 is excellent for usability, clarity and usefulness.
- **Analysis:** A mean ratings scores calculated for each question.

6.2.3 Results

Below is a summary of the average rating for each module:

Table 6.2 Summary of the average rating for each module

Module	Question	Mean Rating (1-5)
Overall Experience	Navigation intuitiveness	4.6
	Overall satisfaction	4.3
	Feature set usefulness	4.8
	Likelihood to recommend	4.6
	Design appeal	4.7
Login/Sign-Up	Ease of account creation	5.0
	Clarity of preference setup	5.0
	Smoothness of login	5.0
	Transition to Home page	5.0
	Guidance for accurate data input	5.0
Home Page	Greeting message usefulness	5.0
	Dietary plan relevance	5.0
	BMI/calorie intake clarity	5.0
	Daily sign-in popup appropriateness	5.0
	Layout appeal	5.0
Health Tracking	Ease of weight logging	4.8
	Graph clarity	4.8
	Motivational message effectiveness	4.6
	Progress monitoring usefulness	4.6
	Visual appeal	4.7
Dietary Plan	Today's plan usefulness	4.2
	Shuffle button ease	4.6
	Image helpfulness	3.5

	Personalization options intuitiveness	4.1
	Personalization needs fulfillment	4.2
Voucher	Points/voucher/history clarity	4.8
	Points motivation	4.8
	Voucher appeal	5.0
	Ease of redemption	5.0
	Overall usefulness	4.8
Settings	Easy to update account details	5.0
	Dietary preferences clarity	4.8
	Notification toggle usefulness	5.0
	Text size adjustment effectiveness	4.8
	Settings layout intuitiveness	4.8

6.2.4 Discussion

The analysis of the user acceptance testing of the Personalize Nutrition and Obesity Intervention System Mobile Application shows that the user experience is very positive and the majority of the modules obtain mean rating of more than 4.5 out of 5, which is a good indication of satisfaction with the functionality, usability, and design of the app.

The Home Page and Login/Sign-Up modules received the perfect 5.0 scores on all the reviewed points, which included the simplicity of creating an account, the ability to set preferences, and the catchiness of such features as personalized greetings and the appropriate dieting plans. Other modules that did well were the Health Tracking and Voucher, and the rating was between 4.6 and 5.0, which pointed at good weight logging, motivation factors, and redemption of vouchers.

The Dietary Plan module, however, was rated slightly lower with image helpfulness 3.5 and personalization options 4.1 to 4.2 rating that indicated a problem with image relevance and inclusion of allergic foods occasionally as reported in the weaknesses. These results imply that the personalized nutrition mobile application system has high usability and engagement, but the dietary plan image accuracy and personalization algorithms require improvement to provide the user with the maximum satisfaction.

6.3 Objective Evaluation

6.3.1 Overview

The Personalize Nutrition and Obesity Intervention System Mobile Application was evaluated against established objectives to determine how effective it is when it comes to achieving project objectives. The testing was based on functionality, performance and user interaction, and measured in quantifiable terms.

6.3.2 Objectives and Metrics

1. Functionally: The app accurately generates personalized dietary plans based on user food allergic and target BMI.

- **Result**: 60% of dietary plan excluded allergens which were tested with 30 sample inputs.

2. Performance: The app responds to user actions such as shuffling plans, redeeming vouchers within 2 seconds.

- **Result**: Mean response time for shuffling plans is taking higher respond time and redeeming vouchers taking less than 2 seconds.

3. User Engagement: The daily sign-in points system encourages regular use.

- **Result**: Users will be using the system daily for claiming the daily sign-in points.

4. Accuracy: The BMI calculation and weight tracking graph are accurate.

- **Result**: 0% error rate which tested with 30 weight entries.

6.3.3 Discussion

The objective analysis proved that the Personalize Nutrition and Obesity Intervention System Mobile Application addresses its fundamental goals. This app perfectly personalized diets to the preferences of the users, and there were no cases of allergens

added. There was strong performance with response time of less than 2- seconds of threshold but the dietary plan functions might take more than 2 seconds for respondent.

The numbers indicate that the rate of user engagement is good, as the number of people signing in every day is big, which is supported by the presence of the points-driven system and the existence of other personalize nutrition systems. The reliability is shown by the correct BMI calculation and graph update, but there are some performance lag issues on swapping the eats plan and it may require another optimization. The accessibility was improved with the text size adjustment feature being highly accepted.

6.4 Overall Discussion

The overall outcomes of the black box testing, user testing, and objective evaluation are that Personalize Nutrition and Obesity Intervention System Mobile Application is a functional, user-friendly, and efficient nutrition application. The black box testing confirmed all the essential features with some performance problems with older devices. The user testing noted high usability since the mean ratings of most features were more than 4.0 but found daily sign-in pop-up and greeting message to be areas to be improved. Objective analysis supported the reliability and interest of the app, and all the metrics were in or above the goals.

Strengths:

- Intuitive interface, especially for login, sign-up, and health tracking.
- Accurate calculation with user BMI.
- Engaging points-based voucher system promoting regular use and QR code features to redeem the code.
- A notification meal reminder for user engagement to alert the users log their meal on time.

Weaknesses:

- Slight performance delays on dietary plan shuffling.
- Inaccuracy of dietary plan image.

- Inaccuracy of dietary plan which some of the output result will show the users allergic food.
- Limited variety of redeemable vouchers.

Recommendations:

- Enhance API requests to Cohere AI to reduce response time.
- Check Pexels API response to have relevant images.
- Include fallback images or local image database.
- Improve Cohere AI prompt to filter out user allergies.
- Add dynamic voucher creation on user preference

Chapter 7 Conclusion and Recommendation

7.1 Conclusion

The Personalized Nutrition and Obesity Intervention System was also designed to tackle the growing global obesity pandemic using user-focused and technology-based mobile applications. Incorporated as an Android app in the Android studio, the system is built on the MySQL back-end through XAMPP and leverages the power of Cohere AI to provide highly customized diet plans based on the individual user profile. The app has a wide range of features that enable the users to take charge of their health.

Indeed, secure user authentication and profile management, which have been deployed using classes such as the LoginActivity.java and ProfileActivity.java, make sure that personal information like age, weight, gender and allergies are safely stored and used. The system computes correct health indices, such as the Body Mass Index (BMI) and Basal Metabolic rate (BMR) using Harris-Benedict equation and sends obese alerts in real time, as observed in TrackingActivity.java. One of the tools the application uses is personalized diet planning, which provides three different modes system suggest, AI service shuffle and users custom, allowing Cohere AI to produce meal plans based on the user preferences and dietary limitations, and the nutritional content being displayed as simple pie charts in ViewEditPlanActivity.java.

Also, it allows tracking health by allowing the user to track weight and activity gains with graphical summaries, as done in WeightHistoryActivity.java. Motivational features like points and voucher redemption, administered by VoucherActivity.java, are used to engage users and keep them active within the app and customization within the app like text size, multilingual support stubs, and option-based plan management, allow most users to access the app and make necessary changes. The Agile Scrum technique was used to develop the project, and it helped in binary development in FYP 1 and FYP 2. This strategy enabled a user-feedback loop that was achieved by mini development sprints and led to the inclusion of improved features such as AI-based diet shuffling and a voucher system in FYP 2 that mitigated the weaknesses that were discovered during the FYP 1 prototyping.

During the stages of testing, such as Unit Testing and User Acceptance Testing (UAT), the reliability and performance of the application were guaranteed. This system fills a very important gap in the market because it places nutrition as its core focus at the heart unlike the current applications like Foodvisor or Cronometer which often have nutrition as a secondary attribute. It can enable the users to lead sustainable eating behavior, which can be refined to obesity-related health hazards like diabetes, hypertension, and high cholesterol. The project is highly valuable to the realm of Personalized Nutrition and Obesity Intervention System Mobile Application Development as it illustrates how artificial intelligence and data analytics might be successfully incorporated in the health environment to encourage wellness worldwide.

7.2 Future Plan & Recommendations

Although the Personalized Nutrition and Obesity Intervention System has fulfilled its goals, there are still various channels in which its performance can be improved to optimize its benefits and overcome the existing challenges. The functionalities of the AI can be expanded and this can lead to an improved functionality of the application. As an example, a machine learning model to predict weight patterns using past data or the ability to recommend recipes in real time using a state-of-the-art nutritional analysis would improve on personalization.

Further, cross-platform frameworks such as Flutter or React Native would expand the range of accessibility of the application and eliminate the existing Android-only restriction. Wearables, including Fitbit or Apple Watch, would provide a real-time activity, and it would be possible to investigate the integration of genetic or medical data, including the results of blood tests, to create more accurate recommendations on the type of food an individual should consume. The biometric authentication, e.g. fingerprint login could add another layer of security. The backend should also be migrated to a cloud solution such as AWS or Firebase to increase scalability and performance as more users join the platform, and an administrator dashboard could be invaluable in the form of analytics on user activity and dietary patterns.

Additional features like community discussion boards, leaderboards or camera-based food scanning that logs meals automatically could help increase the level of user engagement and motivation. It would also be important to integrate telemedicine links to consultations with dietitians.

Once the concept of multilingual assistance and full translation of key languages is implemented, the application would be more open to people of various cultural backgrounds, and the features of increased accessibility such as voice recognition or compatibility with screen readers would consider customers with disabilities.

Longitudinal research using a bigger sample of users would present strong information on the effectiveness of the app in terms of weight loss, adherence, and health outcomes in general and may be supported by health organizations to confirm its efficacy.

Lastly, the system may be aligned with global sustainability objectives by adding eco-friendly dietary suggestions, including low-carbon or vegetarian dishes. Such improvements would solve current limitations, including a scarce number of platform support and the minimal scope of AI capabilities, making the application a fully-fledged, scalable obesity management tool. Future versions would also consider collaborating with health food companies or fitness apps in order to increase the voucher system, which would additionally encourage user adoption and encourage them to adopt healthier lifestyles in the long-term.

REFERENCES

[1] World Health Organization, “Obesity and overweight,” www.who.int, Mar. 01, 2024. https://www.who.int/news-room/fact-sheets/detail/obesity-and-overweight#:~:text=Obesity%20is%20a%20chronic%20complex (accessed Feb 28, 2025).

[2] “Pregnancy and obesity: Know the risks,” Mayo Clinic. https://www.mayoclinic.org/healthy-lifestyle/pregnancy-week-by-week/in-depth/pregnancy-and-obesity/art-20044409#:~:text=Having%20a%20high%20BMI%20can (accessed Feb 28, 2025).

[3] S. Shyam et al., “Effect of Personalized Nutrition on Dietary, Physical Activity, and Health Outcomes: a Systematic Review of Randomized Trials,” Nutrients, vol. 14, no. 19, p. 4104, Oct. 2022, doi: https://doi.org/10.3390/nu14194104. (accessed Feb 28, 2025).

[4] M. M. Islam, T. N. Poly, B. A. Walther, and Y.-C. Jack) Li, “Use of Mobile Phone App Interventions to Promote Weight Loss: Meta-Analysis,” JMIR mHealth and uHealth, vol. 8, no. 7, p. e17039, Jul. 2020, doi: https://doi.org/10.2196/17039. (accessed Feb 28, 2025).

[5] M. Herrero, M. Hugas, U. Lele, A. Wirakartakusumah, and M. Torero, “A Shift to Healthy and Sustainable Consumption Patterns,” Science and Innovations for Food Systems Transformation, pp. 59–85, 2023, doi: https://doi.org/10.1007/978-3-031-15703-5_5. (accessed Mar 10, 2025).

[6] T. Armand, Kintoh Allen Nfor, J.-I. Kim, and H.-C. Kim, “Applications of Artificial Intelligence, Machine Learning, and Deep Learning in Nutrition: A Systematic Review,” Nutrients, vol. 16, no. 7, pp. 1073–1073, Apr. 2024, doi: https://doi.org/10.3390/nu16071073. (accessed Mar 10, 2025).

[7] “Foodvisor App | Calorie Counter - Eat Healthy & Lose Weight,” foodvisor.io. https://www.foodvisor.io/en/ (accessed Mar 10, 2025).

[8] “Foodvisor Review: My Firsthand Experience (2023) | Garage Gym Reviews.” https://www.garagegymreviews.com/foodvisor-review (accessed Mar 10, 2025).

[9] Cronometer, “Cronometer: Track nutrition & count calories,” Cronometer.com, 2011. https://cronometer.com/ (accessed Mar 10, 2025).

[10] S. Lappe, “Expert-Tested: Cronometer Review (2024) | Garage Gym Reviews,” Garage Gym Reviews, Mar. 26, 2024. https://www.garagegymreviews.com/cronometer-review#:~:text=Erin%20gave%20Cronometer%20a%203.5 (accessed Mar 14, 2025).

[11] S. M. Dimitratos, J. B. German, and S. E. Schaefer, “Wearable Technology to Quantify the Nutritional Intake of Adults: Validation Study,” JMIR mHealth and uHealth, vol. 8, no. 7, p. e16405, Jul. 2020, doi: https://doi.org/10.2196/16405. (accessed Mar 14, 2025).

[12] “HealthifyMe - Track Nutrition, Plan Your Diet, Get Fitter, Workout from Home,” www.healthifyme.com. https://www.healthifyme.com/in/ (accessed Mar 15, 2025).

[13] “What Is A Continuous Glucose Monitor? - Blog - HealthifyMe,” HealthifyMe, Dec. 16, 2022. https://www.healthifyme.com/blog/continuous-glucose-monitor/ (accessed Mar 15, 2025).

[14] I. Oakley-Girvan and J. Docherty, “A New Approach to Enhancing Engagement in eHealth Applications (Preprint),” Interactive Journal of Medical Research, Apr. 2022, doi: https://doi.org/10.2196/38886. (accessed Mar 15, 2025).

[15] N. Rieke et al., “The future of digital health with federated learning,” npj Digital Medicine, vol. 3, no. 1, pp. 1–7, Sep. 2020, doi: https://doi.org/10.1038/s41746-020-00323-1. (accessed Mar 21, 2025).

[16] X. Li et al., “Digital Health: Tracking Physiomes and Activity Using Wearable Biosensors Reveals Useful Health-Related Information,” PLOS Biology, vol. 15, no. 1, p. e2001402, Jan. 2017, doi: https://doi.org/10.1371/journal.pbio.2001402. (accessed Mar 21, 2025).

[17] S. Patel, H. Park, P. Bonato, L. Chan, and M. Rodgers, “A review of wearable sensors and systems with application in rehabilitation,” Journal of NeuroEngineering and Rehabilitation, vol. 9, no. 1, p. 21, 2012, doi: https://doi.org/10.1186/1743-0003-9-21. (accessed Mar 21, 2025).

[18] D. Johnson, S. Deterding, K.-A. Kuhn, A. Staneva, S. Stoyanov, and L. Hides, “Gamification for health and wellbeing: A systematic review of the literature,” Internet Interventions, vol. 6, pp. 89–106, Nov. 2016, doi: https://doi.org/10.1016/j.invent.2016.10.002. (accessed Mar 21, 2025).

[19] Jordana Alexandrea, “What is a splash page? Your guide to getting started,” Hostinger Tutorials, Jul. 07, 2025. https://www.hostinger.com/tutorials/what-is-a-splash-page (accessed Sep. 16, 2025).

APPENDIX

Code Sample

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">


    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission
android:name="android.permission.POST_NOTIFICATIONS" />


    <uses-permission
android:name="android.permission.SCHEDULE_EXACT_ALARM" />


    <uses-permission
android:name="android.permission.RECEIVE_BOOT_COMPLETED" />

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@drawable/nutriforu_logo"
        android:label="@string/app_name"
        android:networkSecurityConfig="@xml/network_security_config"
        android:roundIcon="@drawable/nutriforu_logo"
        android:supportsRtl="true"
        android:theme="@style/Theme.BMI"
        android:usesCleartextTraffic="true"
        tools:targetApi="31">


        <activity
            android:name=".CalculatorActivity"
            android:exported="false" />
        <activity
            android:name=".PlanListActivity"
            android:exported="false" />
        <activity
            android:name=".HomeActivity"
            android:exported="false" />
        <activity
            android:name=".TrackingActivity"
            android:exported="false"
            android:label="Healthy Tracking"
            android:windowSoftInputMode="adjustResize" />
        <activity
            android:name=".WeightHistoryActivity"
            android:exported="false" />
        <activity
            android:name=".SettingsActivity"
            android:exported="false" />
```

```
        <activity
            android:name=".ProfileEditActivity"
            android:exported="false" />
        <activity
            android:name=".EditPreferenceActivity"
            android:exported="false" />


        <receiver
            android:name=".MealReminderReceiver"
            android:exported="false">
            <intent-filter>
                <action android:name="com.example.bmi.MEAL_REMINDER"
/>
            </intent-filter>
        </receiver>


        <receiver
            android:name=".BootReceiver"
            android:exported="true">
            <intent-filter>
                <action
android:name="android.intent.action.BOOT_COMPLETED" />
            </intent-filter>
        </receiver>


        <activity
            android:name=".VoucherActivity"
            android:exported="false" />
        <activity
            android:name=".AddonPersonalizedPlanActivity"
            android:exported="false" />
        <activity
            android:name=".RedemptionHistoryActivity"
            android:exported="false" />
        <activity
            android:name=".EditSavedPlanActivity"
            android:exported="false" />
        <activity
            android:name=".ProfileActivity"
            android:exported="false" />
        <activity
            android:name=".AllergiesActivity"
            android:exported="false" />
        <activity
            android:name=".AgeActivity"
            android:exported="false" />


        <activity
            android:name=".QuestionActivity"
            android:exported="false" />
        <activity
            android:name=".SuggestPlanActivity"
            android:exported="false" />
        <activity
            android:name=".PersonalizedplanActivity"
            android:exported="false" />
```

```
        <activity
            android:name=".ViewEditPlanActivity"
            android:exported="false" />


        <activity
            android:name=".DietaryActivity"
            android:exported="false" />
        <activity
            android:name=".BMIActivity"
            android:exported="false" />
        <activity
            android:name=".WeightActivity"
            android:exported="false" />
        <activity
            android:name=".HeightActivity"
            android:exported="false" />
        <activity
            android:name=".GenderActivity"
            android:exported="false" />
        <activity
            android:name=".RegisterActivity"
            android:exported="false" />
        <activity
            android:name=".LoginActivity"
            android:exported="false" />
        <activity
            android:name=".SplashActivity"
            android:exported="true"
            android:theme="@style/Theme.BMI.SplashActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:theme="@style/Theme.BMI" />

    </application>

</manifest>
```

build.gradle.kts(:app)

```
plugins {
    alias(libs.plugins.android.application)

}

android {
    namespace = "com.example.bmi"
    compileSdk = 35

    defaultConfig {
        applicationId = "com.example.bmi"
        minSdk = 24
        targetSdk = 35
```

```
        versionCode = 1
        versionName = "1.0"

        testInstrumentationRunner =
"androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-
optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_11
        targetCompatibility = JavaVersion.VERSION_11
    }
}

dependencies {
    implementation ("com.android.volley:volley:1.2.1")
    implementation ("com.squareup.okhttp3:logging-interceptor:4.9.3")
    implementation("com.github.lzyzsd:circleprogress:1.2.1")
    implementation("com.squareup.retrofit2:retrofit:2.9.0")
    implementation("com.squareup.retrofit2:converter-gson:2.9.0")
    implementation("com.squareup.okhttp3:okhttp:4.9.0")
    implementation ("com.github.PhilJay:MPAndroidChart:v3.1.0")
    implementation("com.github.bumptech.glide:glide:4.15.1")
    implementation("com.google.android.material:material:1.11.0")
    implementation ("com.journeyapps:zxing-android-embedded:4.3.0")
    implementation ("com.google.zxing:core:3.5.2")
    implementation ("androidx.core:core:1.13.1") // or latest version
    implementation ("com.google.android.material:material:1.12.0")
    implementation(libs.core.splashscreen)
    // Volley library
    implementation("com.android.volley:volley:1.2.1")
    implementation(libs.appcompat)
    implementation(libs.material)
    implementation(libs.activity)
    implementation(libs.constraintlayout)
    testImplementation(libs.junit)
    androidTestImplementation(libs.ext.junit)
    androidTestImplementation(libs.espresso.core)
}
```

AddonPersonalizedPlanActivity.java

```
package com.example.bmi;

import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.os.Bundle;
import android.text.InputType;
import android.util.Log;
```

```
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.LinearLayout;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;
import android.graphics.Rect;
import android.graphics.Typeface;
import android.view.ViewGroup;
import androidx.recyclerview.widget.RecyclerView;
import androidx.viewpager2.widget.ViewPager2;
import com.google.android.material.tabs.TabLayout;
import com.google.android.material.tabs.TabLayoutMediator;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import com.github.lzyzsd.circleprogress.DonutProgress;
import com.github.mikephil.charting.charts.PieChart;
import com.github.mikephil.charting.data.PieData;
import com.github.mikephil.charting.data.PieDataSet;
import com.github.mikephil.charting.data.PieEntry;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
import java.util.regex.Pattern;

public class AddonPersonalizedPlanActivity extends AppCompatActivity
{
    private double weight = 90;
    private int height = 180;
    private String gender = "Male";
    private int age = 21;
    private double bmr = 0;
    private String target="Diet";
    private DonutProgress bmr_progressbar;
    private DonutProgress protein_progressbar;
    private DonutProgress carbonhydrate_progressbar;
    private DonutProgress fat_progressbar;
    private int proteinRate;
    private int carbsRate;
    private int fatRate;
    private int proteinNeeded;
    private int carbsNeeded;
    private int fatNeeded;
```

```
    private double proteinEveryMeal;
    private double carbsEveryMeal;
    private double fatEveryMeal;
    private double targetCalories;
    private String planName;
    private LinearLayout chartsContainer;
    private String answer;
    private int userId;
    private String function;
    private Button GenerateButton;
    private Button SaveButton;
    private Spinner spinnerProtein1;
    private Spinner spinnerCarbs1;
    private Spinner spinnerFat1;
    private Spinner spinnerProtein2;
    private Spinner spinnerCarbs2;
    private Spinner spinnerFat2;
    private Spinner spinnerProtein3;
    private Spinner spinnerCarbs3;
    private Spinner spinnerFat3;
    private ViewPager2 mealPager;
    private TabLayout mealTabs;

    private RadioGroup rgTarget;
    private RadioButton rbDiet, rbMaintain, rbGain;

    // Maps to store the nutrition data
    private Map<String, Integer> dailyMacros = new HashMap<>();
    private Map<String, Map<String, Integer>> mealFoods = new
LinkedHashMap<>();

    private ArrayList<String> oriproteinFoods = new ArrayList<>();
    private ArrayList<String> oricarbsFoods = new ArrayList<>();

    private ArrayList<String> orifatFoods = new ArrayList<>();

    private ArrayList<String> proteinFoods = new ArrayList<>();
    private ArrayList<Double> proteinFoodsInGram = new ArrayList<>();
    private ArrayList<String> carbsFoods = new ArrayList<>();
    private ArrayList<Double> carbsFoodsInGram = new ArrayList<>();
    private ArrayList<String> fatFoods = new ArrayList<>();
    private ArrayList<Double> fatFoodsInGram = new ArrayList<>();
    private String allergy;
    private ImageButton backButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.addonpersonalizedplanactivity);

        rgTarget   = findViewById(R.id.rgTarget);
        rbDiet     = findViewById(R.id.rbDiet);
        rbMaintain = findViewById(R.id.rbMaintain);
        rbGain     = findViewById(R.id.rbGain);


        target = "Maintain";
        rbMaintain.setChecked(true);
        rgTarget.setOnCheckedChangeListener((group, checkedId) -> {
            if (checkedId == R.id.rbDiet) {
```

```
                target = "Diet";
            } else if (checkedId == R.id.rbGain) {
                target = "Gain more muscle";
            } else {
                target = "Maintain";
            }
            computeAndBindMacros(); // refresh rings when user
switches goal
        });

        mealPager = findViewById(R.id.mealPager);
        mealTabs  = findViewById(R.id.mealTabs);


        mealPager.setVisibility(View.GONE);
        mealTabs.setVisibility(View.GONE);


        mealPager.setOffscreenPageLimit(1);
        mealPager.setPageTransformer((page, position) -> {
            page.setScaleY(0.95f + (1 - Math.abs(position)) * 0.05f);
        });


        mealPager.addItemDecoration(new
EdgesMarginDecoration(dp(8)));

        SharedPreferences sharedPreferences =
getSharedPreferences("user_prefs", MODE_PRIVATE);

allergy=sharedPreferences.getString("allergies","[Eggs,Meats]");
        int size=sharedPreferences.getInt("text_size",12);
        changeTextSize(size);

loadFoodsFromFile(R.raw.protein,proteinFoods,proteinFoodsInGram);
        loadFoodsFromFile(R.raw.carbs,carbsFoods,carbsFoodsInGram);
        loadFoodsFromFile(R.raw.fat,fatFoods,fatFoodsInGram);
        Intent intent = new Intent();
        spinnerProtein1=findViewById(R.id.spinnerProtein1);
        spinnerCarbs1=findViewById(R.id.spinnerCarbs1);
        spinnerFat1=findViewById(R.id.spinnerFat1);
        spinnerProtein2=findViewById(R.id.spinnerProtein2);
        spinnerCarbs2=findViewById(R.id.spinnerCarbs2);
        spinnerFat2=findViewById(R.id.spinnerFat2);
        spinnerProtein3=findViewById(R.id.spinnerProtein3);
        spinnerCarbs3=findViewById(R.id.spinnerCarbs3);
        spinnerFat3=findViewById(R.id.spinnerFat3);
        setupSpinner(spinnerProtein1, proteinFoods);
        setupSpinner(spinnerCarbs1, carbsFoods);
        setupSpinner(spinnerFat1, fatFoods);
        setupSpinner(spinnerProtein2, proteinFoods);
        setupSpinner(spinnerCarbs2, carbsFoods);
        setupSpinner(spinnerFat2, fatFoods);
        setupSpinner(spinnerProtein3, proteinFoods);
        setupSpinner(spinnerCarbs3, carbsFoods);
        setupSpinner(spinnerFat3, fatFoods);
        GenerateButton=findViewById(R.id.GenerateButton);
        GenerateButton.setOnClickListener(new View.OnClickListener()
{
            @Override
```

```
            public void onClick(View v) {
                parseData();

            }
        });

        backButton=findViewById(R.id.backButton123);
        backButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new
Intent(AddonPersonalizedPlanActivity.this, DietaryActivity.class);
                startActivity(intent);

            }
        });
        SaveButton=findViewById(R.id.SaveButton);;
        SaveButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                AlertDialog.Builder builder = new
AlertDialog.Builder(AddonPersonalizedPlanActivity.this);
                builder.setTitle("Key in the Plan Name");


                final EditText input = new
EditText(AddonPersonalizedPlanActivity.this);
                input.setInputType(InputType.TYPE_CLASS_TEXT);
                input.setHint("Exp:Diet Plan");
                builder.setView(input);


                builder.setPositiveButton("Save", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int
which) {
                        planName = input.getText().toString().trim();
                        if (!planName.isEmpty()) {

                            SendtoDatabase();

Toast.makeText(AddonPersonalizedPlanActivity.this, "Saved: " +
planName, Toast.LENGTH_SHORT).show();
                        } else {

Toast.makeText(AddonPersonalizedPlanActivity.this, "Plan name should
not be empty", Toast.LENGTH_SHORT).show();
                        }
                    }
                });


                builder.setNegativeButton("Cancel", null);


                builder.show();
            }
        });
        SaveButton.setEnabled(false);
```

```
        userId = sharedPreferences.getInt("user_id", 2);
        function = sharedPreferences.getString("function",
"Suggest");
        height = sharedPreferences.getInt("height", 180);
        gender = sharedPreferences.getString("gender", "Male");
        weight = sharedPreferences.getFloat("weight", 78);
        age = sharedPreferences.getInt("age", 22);

        Log.e("Personalized",String.valueOf(userId));
        Log.e("Personalized",String.valueOf(function));
        Log.e("Personalized",String.valueOf(height));
        Log.e("Personalized",String.valueOf(gender));
        Log.e("Personalized",String.valueOf(weight));
        Log.e("Personalized",String.valueOf(age));
        Log.e("Personalized",String.valueOf(allergy));

        protein_progressbar = findViewById(R.id.protein_progressbar);
        carbonhydrate_progressbar =
findViewById(R.id.carbonhydrate_progressbar);
        fat_progressbar = findViewById(R.id.fat_progressbar);
        bmr_progressbar = findViewById(R.id.bmr_progressbar);
        computeAndBindMacros();




    }
    private void parseData() {

        String breakfastProtein =
spinnerProtein1.getSelectedItem().toString();
        String breakfastCarbs =
spinnerCarbs1.getSelectedItem().toString();
        String breakfastFat =
spinnerFat1.getSelectedItem().toString();

        String lunchProtein =
spinnerProtein2.getSelectedItem().toString();
        String lunchCarbs =
spinnerCarbs2.getSelectedItem().toString();
        String lunchFat = spinnerFat2.getSelectedItem().toString();

        String dinnerProtein =
spinnerProtein3.getSelectedItem().toString();
        String dinnerCarbs =
spinnerCarbs3.getSelectedItem().toString();
        String dinnerFat = spinnerFat3.getSelectedItem().toString();


        Map<String, Integer> breakfast = new HashMap<>();
        Map<String, Integer> lunch = new HashMap<>();
        Map<String, Integer> dinner = new HashMap<>();


        breakfast.put(breakfastProtein,
calculateGrams(breakfastProtein, proteinFoods,
proteinFoodsInGram,proteinEveryMeal));
        breakfast.put(breakfastCarbs, calculateGrams(breakfastCarbs,
carbsFoods, carbsFoodsInGram,carbsEveryMeal));
```

```
        breakfast.put(breakfastFat, calculateGrams(breakfastFat,
fatFoods, fatFoodsInGram,fatEveryMeal));

        lunch.put(lunchProtein, calculateGrams(lunchProtein,
proteinFoods, proteinFoodsInGram,proteinEveryMeal));
        lunch.put(lunchCarbs, calculateGrams(lunchCarbs, carbsFoods,
carbsFoodsInGram,carbsEveryMeal));
        lunch.put(lunchFat, calculateGrams(lunchFat, fatFoods,
fatFoodsInGram,fatEveryMeal));

        dinner.put(dinnerProtein, calculateGrams(dinnerProtein,
proteinFoods, proteinFoodsInGram,proteinEveryMeal));
        dinner.put(dinnerCarbs, calculateGrams(dinnerCarbs,
carbsFoods, carbsFoodsInGram,carbsEveryMeal));
        dinner.put(dinnerFat, calculateGrams(dinnerFat, fatFoods,
fatFoodsInGram,fatEveryMeal));


        mealFoods.put("Breakfast", breakfast);
        mealFoods.put("Lunch", lunch);
        mealFoods.put("Dinner", dinner);


        displayResults();
    }


    private int calculateGrams(String foodName, ArrayList<String>
foodList, ArrayList<Double> gramsList,Double nutritionNeeded) {
        int index = foodList.indexOf(foodName);
        if (index != -1) {
            double grams = gramsList.get(index);

            return (int) (100 * (nutritionNeeded / (float) grams));
        }
        return 0;
    }
    private void loadFoodsFromFile(int rawResId, ArrayList<String>
foodList, ArrayList<Double> gramsList) {
        try {
            InputStream inputStream =
getResources().openRawResource(rawResId);
            BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream));


            allergy = allergy.replaceAll("[\\[\\]\\s]", "");
            List<String> allergyList =
Arrays.asList(allergy.split(","));

            String line;
            while ((line = reader.readLine()) != null) {
                String[] parts = line.split("-");


                if (parts.length >= 2) {
                    String foodName = parts[0].trim();
                    double grams =
Double.parseDouble(parts[1].trim());
```

```
                    boolean hasAllergy = false;

                    if (parts.length == 3 && parts[2].contains("["))
{
                        // 提取过敏项
                        String allergiesInItem =
parts[2].replaceAll("[\\[\\]\\s]", "");
                        List<String> itemAllergies =
Arrays.asList(allergiesInItem.split(","));

                        for (String allergy : allergyList) {
                            if (itemAllergies.contains(allergy)) {
                                hasAllergy = true;
                                break;
                            }
                        }
                    }

                    if (!hasAllergy) {
                        foodList.add(foodName);
                        gramsList.add(grams);
                    }
                }
            }

            reader.close();
            inputStream.close();
        } catch (IOException e) {
            e.printStackTrace();
        }

        Log.e("List", foodList.toString());
        Log.e("List", gramsList.toString());
    }


    private void setupSpinner(Spinner spinner, ArrayList<String>
items) {
        ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_item, items);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropd
own_item);
        spinner.setAdapter(adapter);
    }


    private class PiePagerAdapter extends
RecyclerView.Adapter<PiePagerAdapter.VH> {
        private final List<String> mealOrder;
        private final Map<String, Map<String, Integer>> data;

        PiePagerAdapter(List<String> mealOrder, Map<String,
Map<String, Integer>> data) {
            this.mealOrder = mealOrder; this.data = data;
        }

        class VH extends RecyclerView.ViewHolder {
            LinearLayout root;
            PieChart pie;
```

```
            TextView title;
            VH(LinearLayout root) {
                super(root);
                this.root = root;
                this.title = (TextView) root.getChildAt(0);
                this.pie   = (PieChart) root.getChildAt(1);
            }
        }

        @Override public VH onCreateViewHolder(ViewGroup parent, int
viewType) {
            // layout: Title (TextView) + PieChart
            LinearLayout col = new LinearLayout(parent.getContext());
            col.setOrientation(LinearLayout.VERTICAL);
            col.setLayoutParams(new RecyclerView.LayoutParams(
                    ViewGroup.LayoutParams.MATCH_PARENT,
ViewGroup.LayoutParams.MATCH_PARENT));
            col.setPadding(dp(8), dp(8), dp(8), dp(8));
            col.setGravity(android.view.Gravity.CENTER_HORIZONTAL);

            TextView title = new TextView(parent.getContext());
            title.setTextSize(16f);
            title.setTypeface(null, Typeface.BOLD);
            title.setPadding(0, dp(4), 0, dp(8));
            title.setGravity(android.view.Gravity.CENTER_HORIZONTAL);
            col.addView(title, new LinearLayout.LayoutParams(
                    ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT));

            PieChart pie = new PieChart(parent.getContext());
            pie.setLayoutParams(new LinearLayout.LayoutParams(
                    ViewGroup.LayoutParams.MATCH_PARENT, dp(200)));
            stylePie(pie);
            col.addView(pie);

            return new VH(col);
        }

        @Override public void onBindViewHolder(VH h, int position) {
            String mealName = mealOrder.get(position);
            h.title.setText(mealName);

            Map<String, Integer> foods = data.get(mealName);
            List<PieEntry> entries = new ArrayList<>();
            if (foods != null) {
                for (Map.Entry<String, Integer> e : foods.entrySet())
{
                    entries.add(new PieEntry(e.getValue(),
e.getKey()));
                }
            }

            PieDataSet ds = new PieDataSet(entries, "");
            ds.setDrawValues(false);
            ds.setSliceSpace(1f);
            ds.setColors(getThemeSliceColors());

            h.pie.setCenterText(mealName);
            h.pie.setData(new PieData(ds));
```

```
            com.github.mikephil.charting.components.Legend L =
h.pie.getLegend();
            L.setEnabled(true);

L.setVerticalAlignment(com.github.mikephil.charting.components.Legend
.LegendVerticalAlignment.BOTTOM);

L.setHorizontalAlignment(com.github.mikephil.charting.components.Lege
nd.LegendHorizontalAlignment.CENTER);

L.setOrientation(com.github.mikephil.charting.components.Legend.Legen
dOrientation.HORIZONTAL);
            L.setDrawInside(false);
            L.setWordWrapEnabled(true);

L.setForm(com.github.mikephil.charting.components.Legend.LegendForm.C
IRCLE);
            L.setTextSize(11f);

            h.pie.invalidate();
        }

        @Override public int getItemCount() { return
mealOrder.size(); }
    }

    private void computeAndBindMacros() {
        // --- BMR ---
        if ("Male".equals(gender)) {
            bmr = (10 * weight) + (6.25 * height) - (5 * age) + 5;
        } else {
            bmr = (10 * weight) + (6.25 * height) - (5 * age) - 161;
        }
        bmr_progressbar.setMax((int) bmr);
        bmr_progressbar.setProgress((int) bmr);
        bmr_progressbar.setText((int) bmr + "kcal");

        // --- Macro split & target calories ---
        // You can tune these % if you want.
        if ("Gain more muscle".equals(target)) {
            targetCalories = bmr + 300;
            proteinRate = 25; carbsRate = 50; fatRate = 25;
        } else if ("Diet".equals(target)) {
            targetCalories = bmr - 300;
            proteinRate = 35; carbsRate = 35; fatRate = 30;
        } else { // Maintain
            targetCalories = bmr;
            proteinRate = 30; carbsRate = 45; fatRate = 25;
        }

        // kcal -> grams (P=4, C=4, F=9)
        proteinNeeded = (int) ((targetCalories * proteinRate / 100.0)
/ 4.0);
        carbsNeeded   = (int) ((targetCalories * carbsRate   / 100.0)
/ 4.0);
        fatNeeded     = (int) ((targetCalories * fatRate     / 100.0)
/ 9.0); // IMPORTANT: fat ÷9

        proteinEveryMeal = proteinNeeded / 3.0;
        carbsEveryMeal   = carbsNeeded   / 3.0;
```

```
        fatEveryMeal     = fatNeeded     / 3.0;

        // --- Bind to the 3 rings ---
        protein_progressbar.setMax(proteinNeeded);
        protein_progressbar.setProgress(proteinNeeded);
        protein_progressbar.setText(String.valueOf(proteinNeeded));

        carbonhydrate_progressbar.setMax(carbsNeeded);
        carbonhydrate_progressbar.setProgress(carbsNeeded);

carbonhydrate_progressbar.setText(String.valueOf(carbsNeeded));

        fat_progressbar.setMax(fatNeeded);
        fat_progressbar.setProgress(fatNeeded);
        fat_progressbar.setText(String.valueOf(fatNeeded));
    }

    private void displayResults() {
        SaveButton.setEnabled(true);

        // order of pages
        final List<String> mealOrder = Arrays.asList("Breakfast",
"Lunch", "Dinner");

        // set the pager adapter
        mealPager.setAdapter(new PiePagerAdapter(mealOrder,
mealFoods));

        // show pager + tabs
        mealPager.setVisibility(View.VISIBLE);
        mealTabs.setVisibility(View.VISIBLE);

        // connect tabs to pager
        new TabLayoutMediator(mealTabs, mealPager,
                (tab, position) ->
tab.setText(mealOrder.get(position))
        ).attach();
    }


    private int getRandomColor() {
        // Generate random RGB values for a color
        int red = (int) (Math.random() * 256);
        int green = (int) (Math.random() * 256);
        int blue = (int) (Math.random() * 256);

        // Return the color in RGB format
        return Color.rgb(red, green, blue);
    }
    private boolean isValidFormat(String response) {
        Pattern validPattern = Pattern.compile(

"\\(Breakfast: .*\\(\\d+g\\), .*\\(\\d+g\\), .*\\(\\d+g\\)\\)\\s*" +

"\\(Lunch: .*\\(\\d+g\\), .*\\(\\d+g\\), .*\\(\\d+g\\)\\)\\s*" +

"\\(Dinner: .*\\(\\d+g\\), .*\\(\\d+g\\), .*\\(\\d+g\\)\\)",
                Pattern.DOTALL
        );
        return validPattern.matcher(response).find();
```

```
    }

    private void SendtoDatabase() {
        String url = "http://10.0.2.2/nutrition/InsertPlan.php";
        Log.e("Connection123", "Start");

        RequestQueue queue = Volley.newRequestQueue(this);
        Log.e("Connection123", "Start1");
        StringRequest stringRequest = new
StringRequest(Request.Method.POST, url,
                response -> {
                    Log.e("Connection123", "Response: " + response);
                    Intent intent = new
Intent(AddonPersonalizedPlanActivity.this, HomeActivity.class);
                    startActivity(intent);

                },
                error -> {
                    Log.e("Connection123", "Volley Error: " +
error.toString());
                }) {
            @Override
            protected Map<String, String> getParams() {
                Log.e("Connection123", "Start2");
                Map<String, String> params = new HashMap<>();
                params.put("plan_name", planName);
                params.put("plan_data", mealFoods.toString());
                params.put("user_id", String.valueOf(userId));
                params.put("bmr", String.valueOf(bmr));
                params.put("proteinNeeded",
String.valueOf(proteinNeeded));
                params.put("carbsNeeded",
String.valueOf(carbsNeeded));
                params.put("fatNeeded", String.valueOf(fatNeeded));
                params.put("target", String.valueOf(target));
                return params;
            }

            @Override
            public String getBodyContentType() {
                return "application/x-www-form-urlencoded;
charset=UTF-8";
            }
        };

        queue.add(stringRequest);
    }
    private void changeTextSize(float size) {

        TextView[] textViews = new TextView[] {
                findViewById(R.id.textView),
                findViewById(R.id.textView2),
                findViewById(R.id.textView6),
                findViewById(R.id.protein_text),
                findViewById(R.id.carbonhydrate_text),
                findViewById(R.id.Fat_text),
                findViewById(R.id.textView),
                findViewById(R.id.textView5),
                findViewById(R.id.textView7),
                findViewById(R.id.textView8),
```

```
                findViewById(R.id.textView3),
                findViewById(R.id.textView9),
                findViewById(R.id.textView10),
                findViewById(R.id.textView11),
                findViewById(R.id.textView12),
                findViewById(R.id.textView13),
                findViewById(R.id.textView14),
                findViewById(R.id.questionText),
                findViewById(R.id.totalquestionText),
                findViewById(R.id.planNameText),
                findViewById(R.id.sdgshfh),
                // Add more TextViews here as needed
        };


        Button[] buttons = new Button[] {
                //findViewById(R.id.Link_Suggest),
                //findViewById(R.id.Link_Personalized),
                //findViewById(R.id.Link_Shuffle),
                //findViewById(R.id.resultBMI),
                findViewById(R.id.GenerateButton),
                findViewById(R.id.SaveButton),
                findViewById(R.id.btnProfile),
                findViewById(R.id.btnPlanList),
                findViewById(R.id.textButton),
                findViewById(R.id.btnLogout),
                findViewById(R.id.nextButton),
                findViewById(R.id.viewButton),
                findViewById(R.id.deleteButton),
                findViewById(R.id.GenerateButton1),
                findViewById(R.id.backbutton1),
                // Add more Buttons here as needed
        };


        for (TextView textView : textViews) {
            if (textView != null) {
                textView.setTextSize(size);
            }
        }


        for (Button button : buttons) {
            if (button != null) {
                button.setTextSize(size);
            }
        }
    }

    private List<Integer> getThemeSliceColors() {
        // keep it tidy: green, blue, orange, yellow, purple (in
case >3 items)
        return Arrays.asList(
                Color.parseColor("#27AE60"), // green
                Color.parseColor("#3498DB"), // blue
                Color.parseColor("#E67E22"), // orange
                Color.parseColor("#F1C40F"), // yellow
                Color.parseColor("#7D3C98")  // purple
        );
    }
```

```
    private int dp(int value) {
        return (int) (value *
getResources().getDisplayMetrics().density);
    }

    // Styles an MPAndroidChart PieChart to look like your theme
    private void stylePie(PieChart chart) {
        chart.getDescription().setEnabled(false);
        chart.getLegend().setEnabled(false);     // hide tiny color
squares
        chart.setDrawEntryLabels(false);         // no labels on
slices
        chart.setUsePercentValues(false);

        chart.setDrawCenterText(true);
        chart.setCenterTextSize(12f);

        // Donut look
        chart.setHoleRadius(70f);
        chart.setTransparentCircleRadius(75f);

        // Extra padding so edges don't get clipped
        chart.setMinOffset(8f);
        chart.setExtraOffsets(6f, 6f, 6f, 6f);
    }

    private static class EdgesMarginDecoration extends
RecyclerView.ItemDecoration {
        private final int margin;
        EdgesMarginDecoration(int marginPx) { this.margin =
marginPx; }

        @Override
        public void getItemOffsets(Rect outRect, View view,
RecyclerView parent, RecyclerView.State state) {
            outRect.left  = margin;
            outRect.right = margin;
        }
    }

}
```

AgeActivity.java

```
package com.example.bmi;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.NumberPicker;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.activity.EdgeToEdge;
import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.contract.ActivityResultContracts;
```

```
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class AgeActivity extends AppCompatActivity {
    NumberPicker numberPicker;
    Button continueButton;
    ImageView ageStageImage;
    TextView ageText;
    TextView stageLabel;
    private float selectedWeight = 40.0f; // Default weight
    private int selectedHeight = 140; // Default height
    private int gender = -1;
    private int selectedAge = 20;
    private ActivityResultLauncher<Intent> AllergiesActivityLauncher;



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //EdgeToEdge.enable(this);
        setContentView(R.layout.activity_age);

        ageStageImage = findViewById(R.id.ageStageImage);
        ageText = findViewById(R.id.ageText);
        stageLabel = findViewById(R.id.stageLabel);

        // Register the launcher inside onCreate
        AllergiesActivityLauncher = registerForActivityResult(
                new ActivityResultContracts.StartActivityForResult(),
result -> {
                    if (result.getResultCode() == RESULT_OK &&
result.getData() != null) {
                        selectedAge =
result.getData().getIntExtra("age", 20);
                        updateAgeSelection();
                    }
                });

        numberPicker = findViewById(R.id.agePicker);
        continueButton = findViewById(R.id.endAll);


        numberPicker.setMinValue(10);
        numberPicker.setMaxValue(100);
        numberPicker.setValue(25);

        ageText.setText(numberPicker.getValue() + " years old"); //
initial text

        int initialVal = numberPicker.getValue();
        ageText.setText(initialVal + " years old");

        if (initialVal <= 2) {
            ageStageImage.setImageResource(R.drawable.baby);
            stageLabel.setText("Baby");
        } else if (initialVal <= 6) {
            ageStageImage.setImageResource(R.drawable.toddler);
```

```
            stageLabel.setText("Toddler");
        } else if (initialVal <= 19) {
            ageStageImage.setImageResource(R.drawable.teenager);
            stageLabel.setText("Teenager");
        } else if (initialVal <= 59) {
            ageStageImage.setImageResource(R.drawable.adult);
            stageLabel.setText("Adult");
        } else {
            ageStageImage.setImageResource(R.drawable.elder);
            stageLabel.setText("Elder");
        }

        numberPicker.setOnValueChangedListener((picker, oldVal,
newVal) -> {
            selectedAge = newVal;
            ageText.setText(newVal + " years old");

            if (newVal <= 2) {
                ageStageImage.setImageResource(R.drawable.baby);
                stageLabel.setText("Baby");
            } else if (newVal <= 6) {
                ageStageImage.setImageResource(R.drawable.toddler);
                stageLabel.setText("Toddler");
            } else if (newVal <= 19) {
                ageStageImage.setImageResource(R.drawable.teenager);
                stageLabel.setText("Teenager");
            } else if (newVal <= 59) {
                ageStageImage.setImageResource(R.drawable.adult);
                stageLabel.setText("Adult");
            } else {
                ageStageImage.setImageResource(R.drawable.elder);
                stageLabel.setText("Elder");
            }
        });

        ImageButton backButton = findViewById(R.id.agebackButton);
        selectedHeight = getIntent().getIntExtra("height", 140);
        selectedWeight = getIntent().getFloatExtra("weight", 40.0f);
        gender = getIntent().getIntExtra("gender", -1);

        // Back button click event
        backButton.setOnClickListener(v -> {
            Intent resultIntent = new Intent();
            resultIntent.putExtra("gender", gender);
            resultIntent.putExtra("height", selectedHeight);
            resultIntent.putExtra("weight", selectedWeight);
            setResult(RESULT_OK, resultIntent);
            finish();
        });

        // Continue button click event
        continueButton.setOnClickListener(v -> {
            selectedAge = numberPicker.getValue();
            Intent intent = new Intent(AgeActivity.this,
BMIActivity.class);
            intent.putExtra("gender", gender);
            intent.putExtra("height", selectedHeight);
            intent.putExtra("weight", selectedWeight);
            intent.putExtra("age", selectedAge);
            SharedPreferences.Editor editor =
```

```
getSharedPreferences("user_prefs", MODE_PRIVATE).edit();
            editor.putInt("height",selectedHeight);
            String gender_Text="";
            if(gender==0){
                gender_Text="Male";
            }else if(gender==1){
                gender_Text="Female";
            }
            editor.putString("gender",gender_Text);
            editor.putFloat("weight",selectedWeight);
            editor.putInt("age",selectedAge);
            AllergiesActivityLauncher.launch(intent);
        });
    }
    private void updateAgeSelection() {
        if (selectedAge >= 20 && selectedAge <= 100.0) {
            numberPicker.setValue(selectedAge);
        }
    }


}
```

AllergiesActivity.java

```
package com.example.bmi;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.Button;
import android.widget.GridLayout;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import com.bumptech.glide.Glide;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

public class AllergiesActivity extends AppCompatActivity {
    GridLayout allergyGrid;
```

```
    ArrayList<String> selectedAllergies = new ArrayList<>();
    LayoutInflater inflater;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_allergies);

        allergyGrid = findViewById(R.id.allergyGrid);
        inflater = LayoutInflater.from(this);

        ImageButton backButton = findViewById(R.id.backButton);
        backButton.setOnClickListener(v -> {
            Intent intent = new Intent(AllergiesActivity.this,
HomeActivity.class);
            startActivity(intent);
            finish();
        });

        Button submitButton = findViewById(R.id.nextButton);
        submitButton.setOnClickListener(v -> saveAllergiesData());

        sendGetRequest();
    }

    private void sendGetRequest() {
        String url =
"http://10.0.2.2/nutrition/login.php?action=get_allergies_question";

        StringRequest stringRequest = new
StringRequest(Request.Method.GET, url,
                response -> {
                    try {
                        JSONArray jsonArray = new
JSONArray(response);
                        for (int i = 0; i < jsonArray.length(); i++)
{
                            JSONObject obj =
jsonArray.getJSONObject(i);
                            String name = obj.getString("question");
                            String imageUrl =
obj.getString("image_path"); // e.g. "images/eggs.png"

                            View allergyItem =
inflater.inflate(R.layout.item_allergy, allergyGrid, false);

                            TextView label =
allergyItem.findViewById(R.id.allergyLabel);
                            ImageView icon =
allergyItem.findViewById(R.id.allergyIcon);

                            label.setText(name);

                            Glide.with(this)
                                    .load("http://10.0.2.2/nutrition/
" + imageUrl)
                                    .placeholder(R.drawable.default_a
llergy_icon)
                                    .into(icon);
```

```
                            // Make the whole item selectable
                            allergyItem.setOnClickListener(v -> {
                                boolean isSelected =
allergyItem.isSelected();
                                allergyItem.setSelected(!isSelected);
                                if (!isSelected) {
                                    selectedAllergies.add(name);
                                } else {
                                    selectedAllergies.remove(name);
                                }
                            });

                            allergyGrid.addView(allergyItem);
                        }
                    } catch (JSONException e) {
                        e.printStackTrace();
                        Toast.makeText(this, "Parsing error",
Toast.LENGTH_SHORT).show();
                    }
                },
                error -> Toast.makeText(this, "Volley error: " +
error.getMessage(), Toast.LENGTH_SHORT).show()
        );

        RequestQueue queue = Volley.newRequestQueue(this);
        queue.add(stringRequest);
    }

    private void saveAllergiesData() {
        saveAllergyToDB(selectedAllergies);

        // Save allergy names to shared preferences
        SharedPreferences.Editor editor =
getSharedPreferences("user_prefs", MODE_PRIVATE).edit();
        try {
            JSONArray allergyArray = new
JSONArray(selectedAllergies);
            editor.putString("allergies", allergyArray.toString());
            Log.d("StoredAllergies", allergyArray.toString());
        } catch (Exception e) {
            e.printStackTrace();
        }
        editor.apply();

        Toast.makeText(this, "Allergies saved",
Toast.LENGTH_SHORT).show();

        startActivity(new Intent(AllergiesActivity.this,
HomeActivity.class));
        finish();
    }

    private void saveAllergyToDB(ArrayList<String> allergiesfood) {
        String url =
"http://10.0.2.2/nutrition/login.php?action=save_allergy";
        int userId = getSharedPreferences("user_prefs",
MODE_PRIVATE).getInt("user_id", 1);

        StringRequest request = new
StringRequest(Request.Method.POST, url,
```

```
                response -> {},
                error -> Toast.makeText(this, "Failed to save
allergy", Toast.LENGTH_SHORT).show()
        ) {
            @Override
            protected Map<String, String> getParams() {
                Map<String, String> params = new HashMap<>();
                params.put("userid", String.valueOf(userId));
                params.put("allergiesfood",
allergiesfood.toString());

                SharedPreferences.Editor editor =
getSharedPreferences("user_prefs", MODE_PRIVATE).edit();
                editor.putString("allergyFood",
allergiesfood.toString());
                editor.apply();

                Log.d("ParamsDebug", "Params Allergies: " + params);
                return params;
            }
        };

        Volley.newRequestQueue(this).add(request);
    }
}
```

BMIActivity.java

```
package com.example.bmi;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.ImageButton;
import android.widget.Button;
import android.widget.NumberPicker;
import android.widget.TextView;
import android.content.Intent;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

import com.android.volley.Request;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;

import org.json.JSONArray;

import java.util.HashMap;
import java.util.Map;

public class BMIActivity extends AppCompatActivity {
```

```
    NumberPicker numberPicker;
    private float selectedWeight = 40.0f; // Default weight
    private float targetBMI = 40.0f; // Default weight
    private int selectedHeight = 140; // Default height
    private int selectedAge = 20; // Default height
    private int gender = -1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //EdgeToEdge.enable(this);
        setContentView(R.layout.activity_bmi);

        // Find UI elements
        Button endButton = findViewById(R.id.endButton);
        TextView bmiResultText = findViewById(R.id.bmiResult); // Add
this TextView in your layout
        TextView bmiMessageText = findViewById(R.id.bmiMessage); //
Also add this

        SharedPreferences prefs = getSharedPreferences("user_prefs",
MODE_PRIVATE);

        // Try to load from SharedPreferences
        selectedWeight = prefs.getFloat("weight", -1.0f);
        selectedHeight = prefs.getInt("height", -1);
        selectedAge = prefs.getInt("age", -1);
        String genderStr = prefs.getString("gender", null);
        Log.e("Save123",genderStr);
        // If not found, fallback to Intent extras]
        if (selectedWeight == -1.0f) {
            selectedWeight = getIntent().getFloatExtra("weight",
40.0f);
        }
        if (selectedHeight == -1) {
            selectedHeight = getIntent().getIntExtra("height", 140);
        }
        if (selectedAge == -1) {
            selectedAge = getIntent().getIntExtra("age", 20);
        }
        if (genderStr == null) {
            gender = getIntent().getIntExtra("gender", -1);
        } else {
            gender = genderStr.equalsIgnoreCase("Male") ? 0 : 1;
        }


        // Calculate BMI
        float heightInMeters = selectedHeight / 100.0f;
        float bmi = selectedWeight / (heightInMeters *
heightInMeters);

        // Determine message
        String bmi_message;
        if (bmi < 18.5f) {
            bmi_message = "Underweight";
        } else if (bmi < 25f) {
            bmi_message = "Normal";
        } else if (bmi < 30f) {
            bmi_message = "Overweight";
```

```
        } else {
            bmi_message = "Obese";
        }

        // Display result
        bmiResultText.setText(String.format("%.1f", bmi));
        bmiMessageText.setText(bmi_message);

        targetBMI = prefs.getFloat("target_BMI", -1.0f);
        NumberPicker numberPicker = findViewById(R.id.targetBMI);

// Create decimal values as strings
        int count = 111; // from 19.0 to 30.0 = (30.0 - 19.0) / 0.1 +
1
        String[] bmiValues = new String[count];
        for (int i = 0; i < count; i++) {
            float value = 19.0f + (i * 0.1f);
            bmiValues[i] = String.format("%.1f", value);
        }

        numberPicker.setMinValue(0);
        numberPicker.setMaxValue(bmiValues.length - 1);
        numberPicker.setDisplayedValues(bmiValues);

        if (targetBMI == -1.0f) {
            targetBMI = 23.0f;
        }

        // Now apply targetBMI to NumberPicker
        int index = (int) ((targetBMI - 19.0f) / 0.1f);
        if (index >= 0 && index < bmiValues.length) {
            numberPicker.setValue(index);
        } else {
            numberPicker.setValue((int) ((23.0f - 19.0f) / 0.1f)); //
safe fallback
        }

        // End BMI Calculate and go back to main page
        endButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String selected = bmiValues[numberPicker.getValue()];
                float selectedBMI = Float.parseFloat(selected);

                SharedPreferences prefs =
getSharedPreferences("user_prefs", MODE_PRIVATE);
                float currentStoredBMI = prefs.getFloat("target_BMI",
-1.0f);

                // Check if selected BMI differs from stored
                if (Math.abs(selectedBMI - currentStoredBMI) >
0.001f) {
                    // Only save if it changed
                    saveUserData(gender, selectedAge, selectedHeight,
selectedWeight, selectedBMI);
                } else {
                    Log.d("BMIActivity", "Target BMI is unchanged.
Skipping DB update.");
                }
```

```
                String allergies = prefs.getString("allergies",
"[]");
                Intent intent;

                if (allergies.equals("")) {
                    intent = new Intent(BMIActivity.this,
AllergiesActivity.class);
                } else {
                    intent = new Intent(BMIActivity.this,
HomeActivity.class);
                }

                startActivity(intent);
            }
        });
    }

    private void saveUserData(int gender, int age, int height, float
weight, float target_BMI ) {
        String url =
"http://10.0.2.2/nutrition/login.php?action=save_user";

        int userId = getSharedPreferences("user_prefs",
MODE_PRIVATE).getInt("user_id", -1);

        StringRequest request = new
StringRequest(Request.Method.POST, url,
                response -> Toast.makeText(this, "User saved",
Toast.LENGTH_SHORT).show(),
                error -> Toast.makeText(this, "Error saving user",
Toast.LENGTH_SHORT).show()
        ) {
            @Override
            protected Map<String, String> getParams() {
                Map<String, String> params = new HashMap<>();
                params.put("userid", String.valueOf(userId)); // <--
send user ID to PHP
                params.put("gender", String.valueOf(gender));
                params.put("age", String.valueOf(age));
                params.put("height", String.valueOf(height));
                params.put("weight", String.valueOf(weight));
                params.put("target_BMI", String.valueOf(target_BMI));
                Log.d("ParamsDebug", "Params: " + params.toString());

                return params;
            }
        };

        Volley.newRequestQueue(this).add(request);

        String genderStr = "";
        if (gender == 0) {
            genderStr = "Male";
        } else if (gender == 1) {
            genderStr = "Female";
        }

        // Save individual fields to SharedPreferences
        SharedPreferences.Editor editor =
```

```
getSharedPreferences("user_prefs", MODE_PRIVATE).edit();
        editor.putString("gender", genderStr);
        editor.putInt("age", age);
        editor.putInt("height", height);
        editor.putFloat("weight", weight);
        editor.putFloat("target_BMI", target_BMI);
        Log.e("SaveBMI",genderStr);
        Log.e("SaveBMI",String.valueOf(age));
        Log.e("SaveBMI",String.valueOf(height));
        Log.e("SaveBMI",String.valueOf(weight));
        Log.e("SaveBMI",String.valueOf(target_BMI));

        editor.apply();
    }
}
```

BootReceiver.java

```
package com.example.bmi;

import android.app.AlarmManager;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences; // NEW: Added for
preferences
import android.os.Build;
import android.util.Log;

import java.util.Calendar;
import java.util.TimeZone;

public class BootReceiver extends BroadcastReceiver {
    private static final int REQ_BF = 21001;
    private static final int REQ_LU = 21002;
    private static final int REQ_DI = 21003;
    private static final String PREFS = "user_prefs"; // NEW: Added
for clarity
    private static final String KEY_MEAL_NOTIF_ENABLED_PREFIX =
"meal_notif_enabled_user_"; // NEW: User-specific key

    @Override
    public void onReceive(Context context, Intent intent) {
        if (Intent.ACTION_BOOT_COMPLETED.equals(intent.getAction()))
{
            Log.d("BootReceiver", "Device rebooted, checking meal
reminders");
            // NEW: Get user ID
            String uid = getCurrentUserId(context);
            boolean enabled = context.getSharedPreferences(PREFS,
Context.MODE_PRIVATE)
                    .getBoolean(KEY_MEAL_NOTIF_ENABLED_PREFIX + uid,
false);
            Log.d("BootReceiver", "Notifications enabled for user " +
uid + ": " + enabled);
            if (enabled) {
                Log.d("BootReceiver", "Rescheduling meal alarms");
```

```
                scheduleAllMealAlarms(context);
            }
        }
    }

    private void scheduleAllMealAlarms(Context ctx) {
        scheduleDaily(ctx, REQ_BF, 8, 0, "Breakfast time! Log your
meal.");
        scheduleDaily(ctx, REQ_LU, 13, 0, "Lunch time! Log your
meal.");
        scheduleDaily(ctx, REQ_DI, 19, 0, "Dinner time! Log your
meal.");
    }

    private void scheduleDaily(Context ctx, int requestCode, int
hour24, int minute, String message) {
        AlarmManager am = (AlarmManager)
ctx.getSystemService(Context.ALARM_SERVICE);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S
&& !am.canScheduleExactAlarms()) {
            Log.e("BootReceiver", "Cannot schedule exact alarms!");
            return;
        }

        Intent i = new Intent(ctx, MealReminderReceiver.class);
        i.setAction("com.example.bmi.MEAL_REMINDER");
        i.putExtra("message", message);
        i.putExtra("requestCode", requestCode);

        PendingIntent pi = PendingIntent.getBroadcast(
                ctx, requestCode, i,
                PendingIntent.FLAG_UPDATE_CURRENT |
PendingIntent.FLAG_IMMUTABLE
        );

        Calendar cal = Calendar.getInstance(TimeZone.getDefault());
        cal.set(Calendar.SECOND, 0);
        cal.set(Calendar.MILLISECOND, 0);
        cal.set(Calendar.HOUR_OF_DAY, hour24);
        cal.set(Calendar.MINUTE, minute);
        if (cal.getTimeInMillis() <= System.currentTimeMillis()) {
            cal.add(Calendar.DAY_OF_YEAR, 1);
        }
        Log.d("BootReceiver", "Scheduling alarm for requestCode: " +
requestCode + " at " + cal.getTime());
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            am.setExactAndAllowWhileIdle(AlarmManager.RTC_WAKEUP,
cal.getTimeInMillis(), pi);
        } else {
            am.setExact(AlarmManager.RTC_WAKEUP,
cal.getTimeInMillis(), pi);
        }
    }

    // NEW: Reused getCurrentUserId
    private String getCurrentUserId(Context context) {
        SharedPreferences sp =
context.getSharedPreferences("session_prefs", Context.MODE_PRIVATE);
        String uid = sp.getString("user_id", null);
        return (uid == null || uid.trim().isEmpty()) ? "guest" :
```

```
uid.trim();
    }
}
```

CalculatorActivity.java

```
package com.example.bmi;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.RadioGroup;
import android.widget.TextView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import java.util.Locale;

public class CalculatorActivity extends AppCompatActivity {

    private EditText weightCal, heightCal, ageCal;
    private TextView indexBMI, statusBMI;
    private Button calculateButton;
    private ImageButton backButton;

    private RadioGroup genderGroup;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_calculator);

        weightCal = findViewById(R.id.weightCal);
        heightCal = findViewById(R.id.heightCal);
        indexBMI = findViewById(R.id.indexBMI);
        statusBMI = findViewById(R.id.statusBMI);
        calculateButton = findViewById(R.id.calculateButton);
        backButton = findViewById(R.id.weightbackButton2);

        ageCal = findViewById(R.id.ageCalculator);
        genderGroup = findViewById(R.id.genderGroup);
        int selectedId = genderGroup.getCheckedRadioButtonId();

        String gender = "";
        if (selectedId == R.id.radioMale) {
            gender = "Male";
        } else if (selectedId == R.id.radioFemale) {
            gender = "Female";
        }

        // Refresh the page and show result
        calculateButton.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View view) {
                calculateAndShowBMI();
            }
```

```
        });

        // Turn back to home page
        backButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                onBackPressed();
            }
        });
    }

    private void calculateAndShowBMI() {
        String wStr = weightCal.getText().toString().trim();
        String hStr = heightCal.getText().toString().trim();
        String ageStr = ageCal.getText().toString().trim();

        if (wStr.isEmpty() || hStr.isEmpty() || ageStr.isEmpty()) {
            Toast.makeText(this, "Please enter age, weight and
height", Toast.LENGTH_SHORT).show();
            return;
        }

        float weight;
        int height;
        int age;

        // Parse and validate numeric input
        try {
            age = Integer.parseInt(ageStr);
        } catch (NumberFormatException e) {
            Toast.makeText(this, "Age must be a valid number.",
Toast.LENGTH_SHORT).show();
            return;
        }

        try {
            weight = Float.parseFloat(wStr);
        } catch (NumberFormatException e) {
            Toast.makeText(this, "Weight must be a valid number",
Toast.LENGTH_SHORT).show();
            return;
        }
        try {
            height = Integer.parseInt(hStr);
        } catch (NumberFormatException e) {
            Toast.makeText(this, "Height must be a valid number.",
Toast.LENGTH_SHORT).show();
            return;
        }

        // Range check for age
        if (age < 2 || age > 100) {
            Toast.makeText(this, "Please enter an age between 2-
100.", Toast.LENGTH_SHORT).show();
            return;
        }

        // Range check for weight
        if (weight < 40f || weight > 120f) {
            Toast.makeText(this, "Please enter a weight between 40-
```

```
120 kg.", Toast.LENGTH_SHORT).show();
            return;
        }

        // Range check for height
        if (height < 140 || height > 200) {
            Toast.makeText(this, "Please enter a height between 140-
200 cm.", Toast.LENGTH_SHORT).show();
            return;
        }


        // Convert cm to meters
        float heightM = height / 100f;

        // BMI formula
        float bmi = weight / (heightM * heightM);
        String bmiText = String.format(Locale.getDefault(), "%.2f",
bmi);
        indexBMI.setText(bmiText);

        // Determine category
        String category;
        if (bmi < 18.5f) {
            category = "Underweight";
        } else if (bmi < 25f) {
            category = "Normal weight";
        } else if (bmi < 30f) {
            category = "Overweight";
        } else {
            category = "Obese";
        }
        statusBMI.setText(category);
    }
}
```

CohereApiClient.java

```
package com.example.bmi;

import okhttp3.OkHttpClient;
import okhttp3.logging.HttpLoggingInterceptor;
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public class CohereApiClient {
    private static final String BASE_URL =
"https://api.cohere.ai/v1/";

    private Retrofit retrofit;

    public CohereApiClient() {
        HttpLoggingInterceptor logging = new
HttpLoggingInterceptor();
        logging.setLevel(HttpLoggingInterceptor.Level.BODY);

        OkHttpClient client = new OkHttpClient.Builder()
                .addInterceptor(logging)
```

```
                .build();


        retrofit = new Retrofit.Builder()
                .baseUrl(BASE_URL)
                .client(client)  //
                .addConverterFactory(GsonConverterFactory.create())
                .build();
    }

    public CohereService createService() {
        return retrofit.create(CohereService.class);
    }
}
```

CohereRequest.java

```
package com.example.bmi;

public class CohereRequest {
    private String model;
    private String prompt;
    private int max_tokens;
    private double temperature;

    public CohereRequest(String prompt) {
        this.model = "command-light";
        this.prompt = prompt;
        this.max_tokens = 300;
        this.temperature = 0.7;
    }


}
```

CohereResponse.java

```
package com.example.bmi;

import java.util.List;

public class CohereResponse {
    private List<Generation> generations;

    public static class Generation {
        public String text;
    }

    public String getText() {
        if (generations != null && !generations.isEmpty()) {
            return generations.get(0).text.trim();
        }
        return "No response from AI.";
    }
}
```

CohereService.java

```
package com.example.bmi;

import retrofit2.Call;
import retrofit2.http.Body;
import retrofit2.http.Headers;
import retrofit2.http.POST;

public interface CohereService {

    @Headers({
//            "Authorization: Bearer
lHXI68SfXm40tQ90tRud1DLxntpDl87OVzRAdc6b",
            // In Case Api Limit
//            "Authorization: Bearer
3iaqWQlCOFRvT7bqp9KrwTviyqzi5Z1lIov3obYG".
            "Authorization: Bearer
tLd5j9DnqroCn9Hsr25z7GO2Nr7jc7wEbpkoVxe3",
            "Content-Type: application/json"
    })
    @POST("generate")
    Call<CohereResponse> generate(@Body CohereRequest request);
}
```

Dietary Activity.java

```
package com.example.bmi;

import android.app.AlertDialog;
import android.app.Dialog;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Bitmap;
import android.graphics.drawable.BitmapDrawable;
import android.os.Bundle;
import android.text.InputType;
import android.util.Base64;
import android.util.Log;
import android.view.View;
import android.widget.*;

import androidx.appcompat.app.AppCompatActivity;

import com.bumptech.glide.Glide;
import
com.google.android.material.bottomnavigation.BottomNavigationView;

import org.json.JSONArray;
import org.json.JSONObject;

import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
```

```
import okhttp3.Call;
import okhttp3.Callback;
import okhttp3.FormBody;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;

public class DietaryActivity extends AppCompatActivity {

    private TextView breakfastText, lunchText, dinnerText;
    private TextView breakfastCal, lunchCal, dinnerCal;
    private ImageView breakfastImage, lunchImage, dinnerImage;
    private Button shuffleButton, saveButton;
    private LinearLayout viewPlanSection;

    private SharedPreferences prefs;

    // cache kcal for total/macros dialog
    private int lastBreakfastKcal = 0, lastLunchKcal = 0,
lastDinnerKcal = 0;
    // cache macros for the popup
    private int[] lastBreakfastMacros = new int[]{0,0,0};
    private int[] lastLunchMacros     = new int[]{0,0,0};
    private int[] lastDinnerMacros    = new int[]{0,0,0};

    // simple kcal extractor
    private static final Pattern KCAL_PATTERN =
            Pattern.compile("(~?\\s*)(\\d{2,4})\\s*kcal",
Pattern.CASE_INSENSITIVE);

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_dietary);

        prefs = getSharedPreferences("user_prefs", MODE_PRIVATE);

        breakfastText = findViewById(R.id.meal1_name);
        lunchText     = findViewById(R.id.meal2_name);
        dinnerText    = findViewById(R.id.meal3_name);
        breakfastCal  = findViewById(R.id.meal1_calories);
        lunchCal      = findViewById(R.id.meal2_calories);
        dinnerCal     = findViewById(R.id.meal3_calories);
        breakfastImage = findViewById(R.id.meal1_image);
        lunchImage     = findViewById(R.id.meal2_image);
        dinnerImage    = findViewById(R.id.meal3_image);
        shuffleButton  = findViewById(R.id.shuffleButton);
        saveButton     = findViewById(R.id.saveButton);
        viewPlanSection = findViewById(R.id.personalizedPlanSection);

        // Go to personalized page
        viewPlanSection.setOnClickListener(v ->
                startActivity(new Intent(DietaryActivity.this,
PersonalizedplanActivity.class)));

        LinearLayout customPlanSection =
findViewById(R.id.customPlanSection);
        customPlanSection.setOnClickListener(v -> {
            Intent i = new Intent(DietaryActivity.this,
AddonPersonalizedPlanActivity.class);
```

```
            startActivity(i);
        });


        // Bottom nav (unchanged)
        BottomNavigationView bottomNav =
findViewById(R.id.bottom_navigation);
        bottomNav.setSelectedItemId(R.id.nav_diet);
        bottomNav.setOnItemSelectedListener(item -> {
            int id = item.getItemId();
            if (id == R.id.nav_home)     startActivity(new
Intent(this, HomeActivity.class));
            if (id == R.id.nav_tracking) startActivity(new
Intent(this, TrackingActivity.class));
            if (id == R.id.nav_voucher)  startActivity(new
Intent(this, VoucherActivity.class));
            if (id == R.id.nav_settings) startActivity(new
Intent(this, SettingsActivity.class));
            return true;
        });

        // Show macros popup when clicking name or image
        View.OnClickListener bkListener = v ->
showMealPopup("Breakfast");
        View.OnClickListener lnListener = v ->
showMealPopup("Lunch");
        View.OnClickListener dnListener = v ->
showMealPopup("Dinner");
        breakfastText.setOnClickListener(bkListener);
        breakfastImage.setOnClickListener(bkListener);
        lunchText.setOnClickListener(lnListener);
        lunchImage.setOnClickListener(lnListener);
        dinnerText.setOnClickListener(dnListener);
        dinnerImage.setOnClickListener(dnListener);

        shuffleButton.setOnClickListener(v -> generateFromAI_All());
        saveButton.setOnClickListener(v -> promptPlanNameAndSave());

        // 1) Try to load today's plan; if none -> fallback to AI
        int userId = prefs.getInt("user_id", -1);
        if (userId > 0) {
            loadTodayPlanOrFallback(userId);
        } else {
            generateFromAI_All();
        }
    }

    /* ========================= today plan fetch
========================= */

    private void loadTodayPlanOrFallback(int userId) {
        String url =
"http://10.0.2.2/nutrition/get_today_plan.php?userid=" + userId;
        OkHttpClient client = new OkHttpClient();
        Request req = new Request.Builder().url(url).build();
        client.newCall(req).enqueue(new Callback() {
            @Override public void onFailure(Call call, IOException e)
{

runOnUiThread(DietaryActivity.this::generateFromAI_All);
```

```
            }
            @Override public void onResponse(Call call, Response
response) throws IOException {
                String json = response.body().string();
                try {
                    JSONObject obj = new JSONObject(json);
                    if
(!"success".equalsIgnoreCase(obj.optString("status"))) {

runOnUiThread(DietaryActivity.this::generateFromAI_All);
                        return;
                    }
                    String b = obj.optString("breakfast_text", "");
                    String l = obj.optString("lunch_text", "");
                    String d = obj.optString("dinner_text", "");
                    int bk = obj.optInt("breakfast_kcal", 0);
                    int lk = obj.optInt("lunch_kcal", 0);
                    int dk = obj.optInt("dinner_kcal", 0);

                    runOnUiThread(() -> {
                        if (!b.isEmpty()) applyMealToUI("Breakfast",
buildCanonicalLine("Breakfast", bk, b),
                                breakfastText, breakfastCal,
breakfastImage);
                        if (!l.isEmpty()) applyMealToUI("Lunch",
buildCanonicalLine("Lunch", lk, l),
                                lunchText, lunchCal, lunchImage);
                        if (!d.isEmpty()) applyMealToUI("Dinner",
buildCanonicalLine("Dinner", dk, d),
                                dinnerText, dinnerCal, dinnerImage);
                        Toast.makeText(DietaryActivity.this, "Loaded
today's saved plan", Toast.LENGTH_SHORT).show();
                    });
                } catch (Exception e) {

runOnUiThread(DietaryActivity.this::generateFromAI_All);
                }
            }
        });
    }

    private String buildCanonicalLine(String meal, int kcal, String
itemsOnly) {
        String items = itemsOnly.replaceAll("^\\s*-\\s*", "").trim();
        return meal + ": ~" + Math.max(200, kcal > 0 ? kcal :
estimateKcalFromItems(items)) + " kcal - " + items;
    }

    /* ========================= AI path (like Personalized)
========================= */

    private void generateFromAI_All() {
        generateMeal("Breakfast");
        generateMeal("Lunch");
        generateMeal("Dinner");
    }

    private void generateMeal(String type) {
        askAiForItems(type);
    }
```

```
    private void askAiForItems(String mealType) {
        String allergies = prefs.getString("allergies", "").trim();
        String prompt =

                "Return EXACTLY one line for " + mealType + " of a
meal plan in this exact format, no calories.\n" +
                        "Format: " + mealType + ": - item1, item2,
item3\n" +
                        "Rules:\n" +
                        "- Do NOT include introductions, suggestions,
or any sentences.\n" +
                        "- ONLY output these 1 lines exactly, no
explanations.\n" +
                        "- 2-5 items, lowercase, 1-3 words each, real
foods only, comma-separated\n" +
                        "- no quantities, no brands, no extra text,
no markdown\n" +
                        (allergies.isEmpty() ? "" : "- Absolutely
exclude these allergens: " + allergies + "\n")+
                        "Return exactly one line, nothing else.";


//        "Return EXACTLY one line for " + mealType + " with ONLY
items, no calories.\n" +
//                        "Format: " + mealType + ": - item1, item2,
item3\n" +
//                        "Rules:\n" +
//                        "- 2-5 items, lowercase, 1-3 words each,
real foods only, comma-separated\n" +
//                        "- no quantities, no brands, no extra text,
no markdown\n" +
//                        (allergies.isEmpty() ? "" : "- exclude
allergens: " + allergies + "\n") +
//                        "Return exactly one line, nothing else.";

        CohereRequest req = new CohereRequest(prompt);
        new
CohereApiClient().createService().generate(req).enqueue(new
retrofit2.Callback<CohereResponse>() {
            @Override public void
onResponse(retrofit2.Call<CohereResponse> call,

retrofit2.Response<CohereResponse> res) {
                if (!res.isSuccessful() || res.body() == null ||
res.body().getText() == null) return;

                String line = res.body().getText().trim();
                String items = line;
                int dash = line.indexOf(" - ");
                if (dash >= 0 && dash + 3 < line.length()) items =
line.substring(dash + 3);
                else {
                    int colon = line.indexOf(':');
                    if (colon >= 0 && colon + 1 < line.length())
items = line.substring(colon + 1);
                }
                items = items.replaceAll("^\\s*-\\s*", "").trim();
                items = items.replaceAll("\\s*[,•|]\\s*", ", ");
                if (items.isEmpty()) items = "oats, banana, yogurt";
```

```
                final String itemsFinal = items;
                runOnUiThread(() -> askAiForKcal(mealType,
itemsFinal));
            }
            @Override public void
onFailure(retrofit2.Call<CohereResponse> call, Throwable t) { /*
silent */ }
        });
    }

    private void askAiForKcal(String mealType, String items) {
        String prompt =
                "Given these " + mealType.toLowerCase() + " items: "
+ items + "\n" +
                        "Output ONLY one integer for total kcal
(200..900). No words, no units, no punctuation.";
        CohereRequest req = new CohereRequest(prompt);

        new
CohereApiClient().createService().generate(req).enqueue(new
retrofit2.Callback<CohereResponse>() {
            @Override public void
onResponse(retrofit2.Call<CohereResponse> call,

retrofit2.Response<CohereResponse> res) {
                int kcal = -1;
                try {
                    if (res.isSuccessful() && res.body() != null &&
res.body().getText() != null) {
                        String text = res.body().getText().trim();
                        String digits = text.replaceAll("[^0-9]",
"");
                        if (!digits.isEmpty()) kcal =
Integer.parseInt(digits);
                    }
                } catch (Exception ignored) {}
                if (kcal < 200 || kcal > 900) kcal =
estimateKcalFromItems(items);

                String canonical = mealType + ": ~" + kcal + " kcal -
" + items;
                runOnUiThread(() -> {
                    if ("Breakfast".equals(mealType))
                        applyMealToUI("Breakfast", canonical,
breakfastText, breakfastCal, breakfastImage);
                    else if ("Lunch".equals(mealType))
                        applyMealToUI("Lunch", canonical, lunchText,
lunchCal, lunchImage);
                    else
                        applyMealToUI("Dinner", canonical,
dinnerText, dinnerCal, dinnerImage);
                });
            }
            @Override public void
onFailure(retrofit2.Call<CohereResponse> call, Throwable t) {
                int kcal = estimateKcalFromItems(items);
                String canonical = mealType + ": ~" + kcal + " kcal -
" + items;
                runOnUiThread(() -> {
```

```
                    if ("Breakfast".equals(mealType))
                        applyMealToUI("Breakfast", canonical,
breakfastText, breakfastCal, breakfastImage);
                    else if ("Lunch".equals(mealType))
                        applyMealToUI("Lunch", canonical, lunchText,
lunchCal, lunchImage);
                    else
                        applyMealToUI("Dinner", canonical,
dinnerText, dinnerCal, dinnerImage);
                });
            }
        });
    }

    /* ========================= UI helpers (kcal, image, macros)
========================= */

    private void applyMealToUI(String mealType, String rawLine,
                               TextView mealTextView, TextView
kcalTextView, ImageView mealImageView) {
        String itemsDisplay = extractItemsDisplay(rawLine);
        mealTextView.setText(itemsDisplay.isEmpty() ? rawLine :
itemsDisplay);

        int kcal = extractKcalIfAny(rawLine);
        if (kcal <= 0) kcal = estimateKcalFromItems(rawLine);
        kcalTextView.setText("~" + kcal + " kcal");

        switch (mealType) {
            case "Breakfast": lastBreakfastKcal = kcal; break;
            case "Lunch":     lastLunchKcal = kcal;     break;
            case "Dinner":    lastDinnerKcal = kcal;    break;
        }

        // kick off image + macros
        fetchMealImage(keywordForImage(itemsDisplay), mealImageView);
        fetchMacrosFromAI(mealType, itemsDisplay, kcal);
    }

    private String extractItemsDisplay(String text) {
        int dashIdx = text.indexOf(" - ");
        String items = (dashIdx >= 0) ? text.substring(dashIdx + 3) :
text;
        items =
items.replaceFirst("(?i)^(breakfast|lunch|dinner)\\s*:\\s*", "");
        items = KCAL_PATTERN.matcher(items).replaceAll("");
        items = items.replaceAll("\\s*[--]\\s*$", "").trim();
        items = items.replaceAll("\\s*[,•|]\\s*", ", ");
        return items;
    }

    private int extractKcalIfAny(String text) {
        Matcher m = KCAL_PATTERN.matcher(text);
        if (m.find()) {
            try { return Integer.parseInt(m.group(2)); } catch
(Exception ignored) {}
        }
        return -1;
    }
```

```
    private int estimateKcalFromItems(String text) {
        // very light heuristic; same feel as
PersonalizedplanActivity
        String itemsPart = text;
        int dashIdx = text.indexOf(" - ");
        if (dashIdx >= 0 && dashIdx + 3 < text.length()) itemsPart =
text.substring(dashIdx + 3);
        itemsPart =
itemsPart.replaceFirst("(?i)^(breakfast|lunch|dinner)\\s*:\\s*", "");
        String[] tokens = itemsPart.split("[,•|]");
        int total = 0, matched = 0;
        Map<String,Integer> HINTS = new HashMap<>();
        HINTS.put("oats",150); HINTS.put("banana",100);
HINTS.put("milk",120); HINTS.put("almond milk",40);
        HINTS.put("chicken",220); HINTS.put("breast",165);
HINTS.put("rice",200); HINTS.put("brown rice",216);
        HINTS.put("egg",78); HINTS.put("eggs",156);
HINTS.put("salad",80); HINTS.put("salmon",230);
        HINTS.put("noodles",300); HINTS.put("noodle",250);
HINTS.put("yogurt",120); HINTS.put("apple",95);
        for (String raw : tokens) {
            String item = raw.trim().toLowerCase();
            int best=0;
            for (String k : HINTS.keySet()) if (item.contains(k))
best = Math.max(best, HINTS.get(k));
            if (best > 0) { total += best; matched++; }
        }
        if (matched == 0) return 350;
        return Math.max(150, Math.min(total, 900));
    }

    private String keywordForImage(String itemsDisplay) {
        if (itemsDisplay == null || itemsDisplay.trim().isEmpty())
return "healthy meal";
        String[] parts = itemsDisplay.split(",");
        for (String p : parts) {
            String s = p.trim().toLowerCase();
            if (s.contains("chicken") || s.contains("salmon") ||
s.contains("egg") ||
                    s.contains("beef") || s.contains("tofu") ||
s.contains("fish") ||
                    s.contains("prawn") || s.contains("shrimp"))
return s;
        }
        for (String p : parts) {
            String s = p.trim().toLowerCase();
            if (s.contains("rice") || s.contains("noodle") ||
s.contains("pasta") || s.contains("oat")) return s;
        }
        return parts[0].trim();
    }

    private void fetchMealImage(String query, ImageView imageView) {
        try {
            // URL-encode
            String q = java.net.URLEncoder.encode(query, "UTF-8");
            String url = "https://api.pexels.com/v1/search?query=" +
q +
                    "&per_page=12&orientation=landscape&size=medium";
```

```
            okhttp3.OkHttpClient client = new okhttp3.OkHttpClient();
            okhttp3.Request request = new okhttp3.Request.Builder()
                    .url(url)
                    .addHeader("Authorization",
"aGuhprX2IJ2guNswKqnQyhR08nW2MK5x0hbKmRCC58D44vFyCqvJp8ZU")
                    .build();

            client.newCall(request).enqueue(new okhttp3.Callback() {
                @Override public void onFailure(okhttp3.Call call,
java.io.IOException e) {
                    Log.e("PEXELS", "Failed: " + e.getMessage());
                }

                @Override public void onResponse(okhttp3.Call call,
okhttp3.Response response) throws java.io.IOException {
                    if (!response.isSuccessful() || response.body()
== null) {
                        Log.e("PEXELS", "HTTP " + response.code());
                        return;
                    }
                    try {
                        String json = response.body().string();
                        org.json.JSONObject obj = new
org.json.JSONObject(json);
                        org.json.JSONArray photos =
obj.optJSONArray("photos");
                        if (photos == null || photos.length() == 0)
return;

                        // Pick the first photo whose alt looks like
FOOD (not hands/people/drinks)
                        int chosen = -1;
                        String queryLower = query.toLowerCase();
                        for (int i = 0; i < photos.length(); i++) {
                            org.json.JSONObject p =
photos.getJSONObject(i);
                            String alt = p.optString("alt",
"").toLowerCase();

                            boolean looksLikeFood =
                                    alt.contains("food") ||
alt.contains("dish") || alt.contains("meal") ||
                                            alt.contains("plate") ||
alt.contains("bowl") || alt.contains("salad") ||
                                            alt.contains("rice")  ||
alt.contains("noodle") || alt.contains("pasta") ||
                                            alt.contains("chicken")
|| alt.contains("salmon") || alt.contains("egg") ||
                                            alt.contains("tofu") ||
alt.contains("beef") || alt.contains("vegetable") ||
                                            alt.contains(queryLower);

                            boolean mentionsPeople =

alt.matches(".*\\b(man|woman|people|person|friends|couple|hands)\\b.*
");

                            boolean drinkOnly =
                                    alt.contains("wine") ||
alt.contains("coffee") || alt.contains("tea") ||
```

```
                                                alt.contains("cocktail")
|| alt.contains("juice");

                                if (looksLikeFood && !mentionsPeople
&& !drinkOnly) {
                                    chosen = i;
                                    break;
                                }
                            }
                            if (chosen == -1) chosen = 0; // fallback
first image

                            String imageUrl =
photos.getJSONObject(chosen).getJSONObject("src").getString("medium")
;
                            runOnUiThread(() ->
Glide.with(DietaryActivity.this).load(imageUrl).into(imageView));
                        } catch (Exception e) {
                            Log.e("PEXELS", "Parse error: " +
e.getMessage());
                        }
                    }
                });
            } catch (Exception ex) {
                Log.e("PEXELS", "Build URL error: " + ex.getMessage());
            }
        }

        private void fetchMacrosFromAI(String mealType, String
itemsDisplay, int kcal) {
            if (itemsDisplay == null || itemsDisplay.trim().isEmpty())
return;

            String prompt =
                    "You are a nutritionist. Estimate macronutrients for
this single meal.\n" +
                            "Items: " + itemsDisplay + "\n" +
                            "Total energy: " + Math.max(200, kcal) + "
kcal\n" +
                            "Return ONLY JSON with INTEGER grams (no
prose, no markdown):\n" +
                            "{\"protein_g\": P, \"carbs_g\": C,
\"fat_g\": F}\n" +
                            "Constraints: 4*P + 4*C + 9*F must be within
15% of the total kcal.";

            CohereRequest req = new CohereRequest(prompt);
            new
CohereApiClient().createService().generate(req).enqueue(new
retrofit2.Callback<CohereResponse>() {
                @Override public void
onResponse(retrofit2.Call<CohereResponse> call,

retrofit2.Response<CohereResponse> res) {
                    int p=0,c=0,f=0;
                    try {
                        if (res.isSuccessful() && res.body() != null) {
                            String raw = res.body().getText();
                            String json = findFirstJsonObject(raw);
                            if (json != null) {
```

```
                            JSONObject o = new JSONObject(json);
                            p = Math.max(0, o.optInt("protein_g",
0));
                            c = Math.max(0, o.optInt("carbs_g",
0));
                            f = Math.max(0, o.optInt("fat_g",
0));
                            int energy = 4*p + 4*c + 9*f;
                            if (p <= 0 || c <= 0 || f <= 0 ||
Math.abs(energy - kcal) > 0.15*kcal) {
                                int[] local =
computeMacrosLocal(mealType, kcal);
                                p = local[0]; c = local[1]; f =
local[2];
                            }
                        } else {
                            int[] local =
computeMacrosLocal(mealType, kcal);
                            p = local[0]; c = local[1]; f = local[2];
                        }
                    } else {
                        int[] local = computeMacrosLocal(mealType,
kcal);
                        p = local[0]; c = local[1]; f = local[2];
                    }
                } catch (Exception e) {
                    int[] local = computeMacrosLocal(mealType, kcal);
                    p = local[0]; c = local[1]; f = local[2];
                }
                final int fp=p, fc=c, ff=f;
                runOnUiThread(() -> {
                    switch (mealType) {
                        case "Breakfast": lastBreakfastMacros = new
int[]{fp,fc,ff}; break;
                        case "Lunch":     lastLunchMacros     = new
int[]{fp,fc,ff}; break;
                        case "Dinner":    lastDinnerMacros    = new
int[]{fp,fc,ff}; break;
                    }
                });
            }
            @Override public void
onFailure(retrofit2.Call<CohereResponse> call, Throwable t) {
                int[] local = computeMacrosLocal(mealType, kcal);
                final int fp=local[0], fc=local[1], ff=local[2];
                runOnUiThread(() -> {
                    switch (mealType) {
                        case "Breakfast": lastBreakfastMacros = new
int[]{fp,fc,ff}; break;
                        case "Lunch":     lastLunchMacros     = new
int[]{fp,fc,ff}; break;
                        case "Dinner":    lastDinnerMacros    = new
int[]{fp,fc,ff}; break;
                    }
                });
            }
        });
    }

    private String findFirstJsonObject(String s) {
```

```
        if (s == null) return null;
        int start = s.indexOf('{');
        int end = s.lastIndexOf('}');
        return (start >= 0 && end > start) ? s.substring(start, end +
1) : null;
    }

    private int[] computeMacrosLocal(String mealType, int kcal) {
        double pRatio, cRatio, fRatio;
        switch (mealType) {
            case "Breakfast": pRatio = 0.25; cRatio = 0.55; fRatio =
0.20; break;
            case "Lunch":     pRatio = 0.30; cRatio = 0.50; fRatio =
0.20; break;
            default:          pRatio = 0.35; cRatio = 0.45; fRatio =
0.20; break;
        }
        int p = Math.max(0, (int)Math.round((kcal * pRatio) / 4.0));
        int c = Math.max(0, (int)Math.round((kcal * cRatio) / 4.0));
        int f = Math.max(0, (int)Math.round((kcal * fRatio) / 9.0));
        int total = 4*p + 4*c + 9*f;
        if (total > 0) {
            double scale = (double)kcal / total;
            p = Math.max(0, (int)Math.round(p * scale));
            c = Math.max(0, (int)Math.round(c * scale));
            f = Math.max(0, (int)Math.round(f * scale));
        }
        return new int[]{p, c, f};
    }

    private int parseKcalLabel(String label) {
        if (label == null) return 0;
        Matcher m = Pattern.compile("(\\d{2,4})\\s*kcal",
Pattern.CASE_INSENSITIVE).matcher(label);
        return m.find() ? Integer.parseInt(m.group(1)) : 0;
    }

    /* ========================= Save plan with name
========================= */

    private void promptPlanNameAndSave() {
        // dialog with EditText + "set as today" checkbox
        LinearLayout container = new LinearLayout(this);
        container.setOrientation(LinearLayout.VERTICAL);
        int pad = (int) (16 *
getResources().getDisplayMetrics().density);
        container.setPadding(pad, pad, pad, 0);

        final EditText input = new EditText(this);
        input.setHint("e.g., High-Protein Plan");
        input.setInputType(InputType.TYPE_CLASS_TEXT |
InputType.TYPE_TEXT_FLAG_CAP_WORDS);

        final CheckBox cbToday = new CheckBox(this);
        cbToday.setText("Also set as today's plan");
        cbToday.setChecked(false);

        container.addView(input);
        container.addView(cbToday);
```

```
        new AlertDialog.Builder(this)
                .setTitle("Save Meal Plan")
                .setMessage("Give this meal plan a name")
                .setView(container)
                .setNegativeButton("Cancel", null)
                .setPositiveButton("Save", (d, w) -> {
                    String name = input.getText().toString().trim();
                    if (name.isEmpty()) {
                        name = "Plan " + new
java.text.SimpleDateFormat("yyyy-MM-dd HH:mm",

java.util.Locale.getDefault()).format(new java.util.Date());
                    }
                    savePlan(name, cbToday.isChecked());
                })
                .show();
    }

    private void savePlan(String planName, boolean setAsToday) {
        int userId = prefs.getInt("user_id", -1);
        if (userId == -1) {
            Toast.makeText(this, "User not logged in",
Toast.LENGTH_SHORT).show();
            return;
        }

        String breakfast = breakfastText.getText().toString();
        String lunch     = lunchText.getText().toString();
        String dinner    = dinnerText.getText().toString();
        String img1 = encodeImage(breakfastImage);
        String img2 = encodeImage(lunchImage);
        String img3 = encodeImage(dinnerImage);

        int bk = parseKcalLabel(breakfastCal.getText().toString());
        int ln = parseKcalLabel(lunchCal.getText().toString());
        int dn = parseKcalLabel(dinnerCal.getText().toString());

        FormBody body = new FormBody.Builder()
                .add("userid", String.valueOf(userId))
                .add("plan_name", planName)
                .add("breakfast", breakfast)
                .add("lunch", lunch)
                .add("dinner", dinner)
                .add("breakfast_image", img1)
                .add("lunch_image", img2)
                .add("dinner_image", img3)
                .add("breakfast_kcal", String.valueOf(bk))
                .add("lunch_kcal",     String.valueOf(ln))
                .add("dinner_kcal",    String.valueOf(dn))
                .add("set_as_today", setAsToday ? "1" : "0")
                .build();

        Request req = new Request.Builder()
                .url("http://10.0.2.2/nutrition/save_personal_plan.ph
p") // unified PHP you used earlier
                .post(body)
                .build();

        new OkHttpClient().newCall(req).enqueue(new Callback() {
            @Override public void onFailure(Call call, IOException e)
```

```
{
                runOnUiThread(() ->
Toast.makeText(DietaryActivity.this, "Save failed",
Toast.LENGTH_SHORT).show());
            }
            @Override public void onResponse(Call call, Response
response) {
                runOnUiThread(() ->
Toast.makeText(DietaryActivity.this, "Plan saved",
Toast.LENGTH_SHORT).show());
            }
        });
    }

    private String encodeImage(ImageView view) {
        try {
            Bitmap bmp = ((BitmapDrawable)
view.getDrawable()).getBitmap();
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            bmp.compress(Bitmap.CompressFormat.JPEG, 100, baos);
            return Base64.encodeToString(baos.toByteArray(),
Base64.DEFAULT);
        } catch (Exception e) {
            return "";
        }
    }

    /* ========================= Macros popup
========================= */

    private void showMealPopup(String mealType) {
        View dialogView =
getLayoutInflater().inflate(R.layout.dialog_meal_details, null);

        TextView titleTv  = dialogView.findViewById(R.id.meal_title);
        TextView itemsTv  = dialogView.findViewById(R.id.meal_items);
        TextView kcalTv   = dialogView.findViewById(R.id.meal_kcal);
        TextView protTv   =
dialogView.findViewById(R.id.meal_protein);
        TextView carbsTv  = dialogView.findViewById(R.id.meal_carbs);
        TextView fatTv    = dialogView.findViewById(R.id.meal_fat);
        Button   closeBtn = dialogView.findViewById(R.id.close_btn);

        String items;
        int kcal;
        int[] macros;

        switch (mealType) {
            case "Breakfast":
                items  = breakfastText.getText() != null ?
breakfastText.getText().toString() : "--";
                kcal   = parseKcalLabel(breakfastCal.getText() !=
null ? breakfastCal.getText().toString() : "0 kcal");
                macros = lastBreakfastMacros;
                break;
            case "Lunch":
                items  = lunchText.getText() != null ?
lunchText.getText().toString() : "--";
                kcal   = parseKcalLabel(lunchCal.getText() != null ?
lunchCal.getText().toString() : "0 kcal");
```

```
                macros = lastLunchMacros;
                break;
            default:
                items  = dinnerText.getText() != null ?
dinnerText.getText().toString() : "--";
                kcal   = parseKcalLabel(dinnerCal.getText() != null ?
dinnerCal.getText().toString() : "0 kcal");
                macros = lastDinnerMacros;
                break;
        }

        titleTv.setText(mealType);
        itemsTv.setText(items);
        kcalTv.setText("~" + Math.max(0, kcal) + " kcal");

        String proteinStr = (macros[0] > 0) ? (macros[0] + " g") : "-
- g";
        String carbsStr   = (macros[1] > 0) ? (macros[1] + " g") : "-
- g";
        String fatStr     = (macros[2] > 0) ? (macros[2] + " g") : "-
- g";
        protTv.setText("Protein: " + proteinStr);
        carbsTv.setText("Carbs: " + carbsStr);
        fatTv.setText("Fat: " + fatStr);

        Dialog dialog = new
AlertDialog.Builder(this).setView(dialogView).create();
        closeBtn.setOnClickListener(v -> dialog.dismiss());
        dialog.show();
    }
}
```

EditPreferenceActivity.java

```
package com.example.bmi;

import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.Button;
import android.widget.GridLayout;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import com.bumptech.glide.Glide;
import com.google.android.material.card.MaterialCardView;
```

```
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

public class EditPreferenceActivity extends AppCompatActivity {

    private static final String BASE = "http://10.0.2.2/nutrition/";


    private static final String URL_CATALOG = BASE +
"login.php?action=get_allergies_question";


    private static final String URL_GET_SELECTED = BASE +
"settings.php?action=get_user_allergies";
    private static final String URL_SAVE         = BASE +
"settings.php?action=save_user_allergies";

    private GridLayout allergyGrid;
    private View progress;
    private Button btnSave;

    private final ArrayList<String> catalogNames = new ArrayList<>();
    private final ArrayList<String> catalogImagePaths = new
ArrayList<>();
    private final Set<String> selected = new HashSet<>();

    private LayoutInflater inflater;
    private RequestQueue queue;
    private int userId;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_edit_preference);

        inflater = LayoutInflater.from(this);
        queue = Volley.newRequestQueue(this);

        SharedPreferences prefs = getSharedPreferences("user_prefs",
MODE_PRIVATE);
        userId = prefs.getInt("user_id", -1);

        ImageButton btnBack = findViewById(R.id.btnBack);
        btnBack.setOnClickListener(v -> onBackPressed());

        allergyGrid = findViewById(R.id.allergyGrid);
        progress    = findViewById(R.id.progress);
        btnSave     = findViewById(R.id.btnSave);

        btnSave.setOnClickListener(v -> saveToServer());
```

```
        fetchSelectedThenCatalog();
    }

    private void fetchSelectedThenCatalog() {
        showLoading(true);

        if (userId <= 0) {
            String cached = getSharedPreferences("user_prefs",
MODE_PRIVATE).getString("allergyFood", "");
            parseAndFillSelectedFromString(cached);
            fetchCatalog();
            return;
        }

        StringRequest req = new StringRequest(Request.Method.POST,
URL_GET_SELECTED,
                resp -> {
                    try {
                        JSONObject o = new JSONObject(resp);
                        if
("success".equalsIgnoreCase(o.optString("status"))) {

                            JSONArray arr =
o.optJSONArray("selected");
                            if (arr != null) {
                                selected.clear();
                                for (int i = 0; i < arr.length();
i++) selected.add(arr.getString(i).trim());
                            } else {

parseAndFillSelectedFromString(o.optString("allergiesfood", ""));
                            }
                        } else {

                            parseAndFillSelectedFromString(

getSharedPreferences("user_prefs", MODE_PRIVATE)
                                            .getString("allergyFood",
"")
                            );
                        }
                    } catch (Exception e) {
                        parseAndFillSelectedFromString(
                                getSharedPreferences("user_prefs",
MODE_PRIVATE)
                                        .getString("allergyFood", "")
                        );
                    }
                    fetchCatalog();
                },
                err -> {
                    parseAndFillSelectedFromString(
                            getSharedPreferences("user_prefs",
MODE_PRIVATE)
                                    .getString("allergyFood", "")
                    );
                    fetchCatalog();
                }) {
            @Override
            protected Map<String, String> getParams() {
```

```
                Map<String, String> p = new HashMap<>();
                p.put("userid", String.valueOf(userId));
                return p;
            }
        };
        queue.add(req);
    }

    private void parseAndFillSelectedFromString(String raw) {
        if (raw == null) return;
        // try JSON array first
        try {
            JSONArray a = new JSONArray(raw);
            selected.clear();
            for (int i = 0; i < a.length(); i++)
selected.add(a.getString(i).trim());
            return;
        } catch (Exception ignore) {}
        // try "[Gluten, Fish]" or "Gluten,Fish"
        String cleaned = raw.replace("[", "").replace("]", "");
        if (!cleaned.trim().isEmpty()) {
            selected.clear();

selected.addAll(Arrays.asList(cleaned.split("\\s*,\\s*")));
        }
    }


    private void fetchCatalog() {
        StringRequest req = new StringRequest(Request.Method.GET,
URL_CATALOG,
                response -> {
                    try {
                        JSONArray arr = new JSONArray(response);
                        catalogNames.clear();
                        catalogImagePaths.clear();
                        for (int i = 0; i < arr.length(); i++) {
                            JSONObject o = arr.getJSONObject(i);

catalogNames.add(o.optString("question"));

catalogImagePaths.add(o.optString("image_path"));
                        }
                        buildGrid();
                    } catch (JSONException e) {
                        Toast.makeText(this, "Parsing error",
Toast.LENGTH_SHORT).show();
                        showLoading(false);
                    }
                },
                error -> {
                    Toast.makeText(this, "Failed to load catalog",
Toast.LENGTH_SHORT).show();
                    showLoading(false);
                });
        queue.add(req);
    }


    private void buildGrid() {
```

```
        allergyGrid.removeAllViews();
        for (int i = 0; i < catalogNames.size(); i++) {
            String name = catalogNames.get(i);
            String imgPath = catalogImagePaths.get(i);

            View card = inflater.inflate(R.layout.item_allergy,
allergyGrid, false);

            TextView label = card.findViewById(R.id.allergyLabel);
            ImageView icon = card.findViewById(R.id.allergyIcon);
            label.setText(name);

            Glide.with(this)
                    .load(BASE + imgPath)       // e.g. image/egg.png
                    .placeholder(R.drawable.default_allergy_icon)
                    .into(icon);

            boolean isSel = selected.contains(name);
            setCardSelected(card, isSel);

            card.setOnClickListener(v -> {
                boolean newState = !v.isSelected();
                setCardSelected(v, newState);
                if (newState) selected.add(name); else
selected.remove(name);
            });

            GridLayout.LayoutParams lp = new
GridLayout.LayoutParams();
            lp.width = 0;
            lp.columnSpec = GridLayout.spec(GridLayout.UNDEFINED,
1f);
            lp.setMargins(dp(8), dp(8), dp(8), dp(8));
            card.setLayoutParams(lp);

            allergyGrid.addView(card);
        }
        showLoading(false);
    }

    private void setCardSelected(View v, boolean selectedState) {
        v.setSelected(selectedState);
        if (v instanceof MaterialCardView) {
            MaterialCardView m = (MaterialCardView) v;
            m.setStrokeWidth(selectedState ? dp(2) : dp(1));
            m.setStrokeColor(selectedState ?
Color.parseColor("#2E7D32") : Color.parseColor("#DDE6DD"));
            m.setCardBackgroundColor(selectedState ?
Color.parseColor("#E7F5EA") : Color.WHITE);
        } else {
            v.setAlpha(selectedState ? 1f : 0.95f);
            v.setBackgroundColor(selectedState ?
Color.parseColor("#E7F5EA") : Color.TRANSPARENT);
        }
    }

    private int dp(int dps) {
        float scale = getResources().getDisplayMetrics().density;
        return Math.round(dps * scale);
    }
```

```
    private void showLoading(boolean show) {
        if (progress != null) progress.setVisibility(show ?
View.VISIBLE : View.GONE);
        if (btnSave != null) btnSave.setEnabled(!show);
    }


    private void saveToServer() {
        showLoading(true);


        JSONArray json = new JSONArray();
        for (String s : selected) json.put(s);
        final String jsonArrayString = json.toString();     //
["Gluten","Fish"]
        final String csvString       = String.join(",", selected); //
Gluten,Fish

        StringRequest req = new StringRequest(Request.Method.POST,
URL_SAVE,
                response -> {
                    showLoading(false);
                    try {
                        JSONObject o = new JSONObject(response);
                        String st = o.optString("status", "");
                        if ("saved".equalsIgnoreCase(st) ||
                                "success".equalsIgnoreCase(st) ||
                                "updated".equalsIgnoreCase(st)) {

                            // cache locally in JSON form
                            getSharedPreferences("user_prefs",
MODE_PRIVATE)
                                    .edit().putString("allergyFood",
jsonArrayString).apply();

                            Toast.makeText(this, "Saved",
Toast.LENGTH_SHORT).show();
                            finish();
                        } else {
                            Toast.makeText(this, "Save error",
Toast.LENGTH_SHORT).show();
                            Log.d("ALLERGY_SAVE", "Server said: " +
response);
                        }
                    } catch (Exception e) {
                        Toast.makeText(this, "Bad server response",
Toast.LENGTH_SHORT).show();
                        Log.e("ALLERGY_SAVE", "Parse error: " +
e.getMessage());
                    }
                },
                error -> {
                    showLoading(false);
                    Toast.makeText(this, "Network error",
Toast.LENGTH_SHORT).show();
                }) {
            @Override
            protected Map<String, String> getParams() {
                Map<String, String> p = new HashMap<>();
```

```
                p.put("userid", String.valueOf(userId));

                p.put("selected", jsonArrayString);
                p.put("allergiesfood", csvString);
                return p;
            }
        };

        queue.add(req);
    }

    @Override
    public void onBackPressed() {
        startActivity(new Intent(this, SettingsActivity.class));
        finish();
    }
}
```

EditSavedPlanActivity.java

```
package com.example.bmi;

import android.content.SharedPreferences;
import android.os.Bundle;
import android.text.InputType;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.*;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import okhttp3.*;

import org.json.JSONArray;
import org.json.JSONObject;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class EditSavedPlanActivity extends AppCompatActivity {

    private RecyclerView plansRecycler;
    private PlansAdapter adapter;
    private final List<Plan> data = new ArrayList<>();
    private SharedPreferences prefs;

    private static final String BASE = "http://10.0.2.2/nutrition/";
    private static final OkHttpClient HTTP = new OkHttpClient();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_edit_saved_plan);

        prefs = getSharedPreferences("user_prefs", MODE_PRIVATE);
```

```
        ImageButton back = findViewById(R.id.backButton);
        back.setOnClickListener(v -> finish());

        plansRecycler = findViewById(R.id.plansRecycler);
        plansRecycler.setLayoutManager(new
LinearLayoutManager(this));
        adapter = new PlansAdapter();
        plansRecycler.setAdapter(adapter);

        fetchPlans();
    }

    private void fetchPlans() {
        int userId = prefs.getInt("user_id", -1);
        if (userId <= 0) {
            Toast.makeText(this, "Missing user id",
Toast.LENGTH_SHORT).show();
            return;
        }

        String url = BASE + "get_saved_plans.php?userid=" + userId;
        Request req = new Request.Builder().url(url).build();

        HTTP.newCall(req).enqueue(new Callback() {
            @Override public void onFailure(@NonNull Call call,
@NonNull IOException e) {
                runOnUiThread(() ->
                        Toast.makeText(EditSavedPlanActivity.this,
"Failed to load plans", Toast.LENGTH_SHORT).show());
            }

            @Override public void onResponse(@NonNull Call call,
@NonNull Response response) throws IOException {
                try {
                    String body = response.body() != null ?
response.body().string() : "[]";
                    JSONArray arr = new JSONArray(body);

                    List<Plan> temp = new ArrayList<>();
                    for (int i = 0; i < arr.length(); i++) {
                        JSONObject o = arr.getJSONObject(i);
                        Plan p = new Plan();
                        p.id = o.optInt("id", 0);
                        p.planName = o.optString("plan_name", "");
                        p.createdAt = o.optString("created_at", "");
                        temp.add(p);
                    }

                    runOnUiThread(() -> {
                        data.clear();
                        data.addAll(temp);
                        adapter.notifyDataSetChanged();
                    });
                } catch (Exception e) {
                    Log.e("PLANS_PARSE", e.getMessage());
                    runOnUiThread(() ->

Toast.makeText(EditSavedPlanActivity.this, "Parse error",
Toast.LENGTH_SHORT).show());
                }
```

```
            }
        });
    }

    /** Rename plan via API, then refresh list item. */
    private void renamePlan(Plan plan, String newName) {
        int userId = prefs.getInt("user_id", -1);
        if (userId <= 0) {
            Toast.makeText(this, "Missing user id",
Toast.LENGTH_SHORT).show();
            return;
        }

        RequestBody body = new FormBody.Builder()
                .add("action", "rename")
                .add("id", String.valueOf(plan.id))
                .add("userid", String.valueOf(userId))
                .add("plan_name", newName)
                .build();

        Request req = new Request.Builder()
                .url(BASE + "plan_actions.php")
                .post(body)
                .build();

        HTTP.newCall(req).enqueue(new okhttp3.Callback() {
            @Override public void onFailure(okhttp3.Call call,
IOException e) {
                runOnUiThread(() ->
                        Toast.makeText(EditSavedPlanActivity.this,
"Rename failed: network", Toast.LENGTH_SHORT).show());
                Log.e("PLAN_RENAME", "HTTP fail: " + e.getMessage());
            }

            @Override public void onResponse(okhttp3.Call call,
okhttp3.Response response) throws IOException {
                String respStr = response.body() != null ?
response.body().string() : "";
                Log.d("PLAN_RENAME", "code=" + response.code() + "
body=" + respStr);

                if (!response.isSuccessful()) {
                    runOnUiThread(() ->

Toast.makeText(EditSavedPlanActivity.this, "Rename failed: " +
response.code(), Toast.LENGTH_SHORT).show());
                    return;
                }

                try {
                    org.json.JSONObject o = new
org.json.JSONObject(respStr);
                    boolean ok = o.optBoolean("ok", false);
                    if (!ok) {
                        String err = o.optString("error", "unknown");
                        runOnUiThread(() ->

Toast.makeText(EditSavedPlanActivity.this, "Rename failed: " + err,
Toast.LENGTH_SHORT).show());
                        return;
```

```
                    }

                    runOnUiThread(() -> {
                        plan.planName = newName;
                        adapter.notifyDataSetChanged();
                        Toast.makeText(EditSavedPlanActivity.this,
"Name updated", Toast.LENGTH_SHORT).show();
                    });
                } catch (Exception ex) {
                    Log.e("PLAN_RENAME", "Parse error: " +
ex.getMessage());
                    runOnUiThread(() ->

Toast.makeText(EditSavedPlanActivity.this, "Rename failed: bad
response", Toast.LENGTH_SHORT).show());
                }
            }
        });
    }



    private void deletePlan(Plan plan, int position) {
        int userId = prefs.getInt("user_id", -1);
        if (userId <= 0) {
            Toast.makeText(this, "Missing user id",
Toast.LENGTH_SHORT).show();
            return;
        }

        RequestBody body = new FormBody.Builder()
                .add("action", "delete")
                .add("id", String.valueOf(plan.id))
                .add("userid", String.valueOf(userId))
                .build();

        Request req = new Request.Builder()
                .url(BASE + "plan_actions.php")
                .post(body)
                .build();

        HTTP.newCall(req).enqueue(new okhttp3.Callback() {
            @Override public void onFailure(okhttp3.Call call,
IOException e) {
                runOnUiThread(() ->
                        Toast.makeText(EditSavedPlanActivity.this,
"Delete failed: network", Toast.LENGTH_SHORT).show());
                Log.e("PLAN_DELETE", "HTTP fail: " + e.getMessage());
            }

            @Override public void onResponse(okhttp3.Call call,
okhttp3.Response response) throws IOException {
                String respStr = response.body() != null ?
response.body().string() : "";
                Log.d("PLAN_DELETE", "code=" + response.code() + "
body=" + respStr);

                if (!response.isSuccessful()) {
                    runOnUiThread(() ->
```

```
Toast.makeText(EditSavedPlanActivity.this, "Delete failed: " +
response.code(), Toast.LENGTH_SHORT).show());
                    return;
                }

                try {
                    org.json.JSONObject o = new
org.json.JSONObject(respStr);
                    boolean ok = o.optBoolean("ok", false);
                    if (!ok) {
                        String err = o.optString("error", "unknown");
                        runOnUiThread(() ->

Toast.makeText(EditSavedPlanActivity.this, "Delete failed: " + err,
Toast.LENGTH_SHORT).show());
                        return;
                    }

                    runOnUiThread(() -> {
                        data.remove(position);
                        adapter.notifyItemRemoved(position);
                        Toast.makeText(EditSavedPlanActivity.this,
"Plan deleted", Toast.LENGTH_SHORT).show();
                    });
                } catch (Exception ex) {
                    Log.e("PLAN_DELETE", "Parse error: " +
ex.getMessage());
                    runOnUiThread(() ->

Toast.makeText(EditSavedPlanActivity.this, "Delete failed: bad
response", Toast.LENGTH_SHORT).show());
                }
            }
        });
    }

    // --- RecyclerView ---

    private class PlansAdapter extends
RecyclerView.Adapter<PlansAdapter.VH> {

        @NonNull @Override
        public VH onCreateViewHolder(@NonNull android.view.ViewGroup
parent, int viewType) {
            View v =
LayoutInflater.from(parent.getContext()).inflate(R.layout.item_saved_
plan, parent, false);
            return new VH(v);
        }

        @Override
        public void onBindViewHolder(@NonNull VH h, int pos) {
            Plan p = data.get(pos);

            // label: "plan name - YYYY-MM-DD" OR "Plan #id - YYYY-
MM-DD"
            String date = (p.createdAt != null &&
p.createdAt.length() >= 10) ? p.createdAt.substring(0, 10) : "";
            String name = (p.planName != null) ? p.planName.trim() :
"";
```

```
            String label = (!name.isEmpty() ? name : ("Plan #" +
p.id)) + (date.isEmpty() ? "" : " - " + date);
            h.label.setText(label);

            // edit
            h.btnEditName.setOnClickListener(v -> {
                final EditText input = new
EditText(EditSavedPlanActivity.this);
                input.setInputType(InputType.TYPE_CLASS_TEXT |
InputType.TYPE_TEXT_FLAG_CAP_WORDS);
                input.setHint("Plan name");
                input.setText(name);

                int pad = (int) (16 *
getResources().getDisplayMetrics().density);
                LinearLayout container = new
LinearLayout(EditSavedPlanActivity.this);
                container.setPadding(pad, pad, pad, 0);
                container.addView(input);

                new
android.app.AlertDialog.Builder(EditSavedPlanActivity.this)
                        .setTitle("Rename plan")
                        .setView(container)
                        .setNegativeButton("Cancel", null)
                        .setPositiveButton("Save", (d, w) -> {
                            String newName =
input.getText().toString().trim();
                            if (newName.isEmpty()) {

Toast.makeText(EditSavedPlanActivity.this, "Name can't be empty",
Toast.LENGTH_SHORT).show();
                                return;
                            }
                            renamePlan(p, newName);
                        }).show();
            });

            // delete
            h.btnDelete.setOnClickListener(v -> {
                new
android.app.AlertDialog.Builder(EditSavedPlanActivity.this)
                        .setTitle("Delete plan")
                        .setMessage("Delete this plan permanently?")
                        .setNegativeButton("Cancel", null)
                        .setPositiveButton("Delete", (d, w) ->
deletePlan(p, h.getAdapterPosition()))
                        .show();
            });
        }

        @Override
        public int getItemCount() { return data.size(); }

        class VH extends RecyclerView.ViewHolder {
            TextView label, btnEditName, btnDelete;
            VH(@NonNull View itemView) {
                super(itemView);
                label = itemView.findViewById(R.id.planLabel);
                btnEditName =
```

```
itemView.findViewById(R.id.btnEditName);
                btnDelete = itemView.findViewById(R.id.btnDelete);
            }
        }
    }

    // --- Model ---
    static class Plan {
        int id;
        String planName;
        String createdAt;
    }
}
```

GenderActivity.java

```
package com.example.bmi;

import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.ImageButton;
import android.widget.Button;
import android.content.Intent;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.contract.ActivityResultContracts;
import androidx.appcompat.app.AppCompatActivity;

public class GenderActivity extends AppCompatActivity {
    private View maleButton, femaleButton, backButton;
    private Button continueButton;
    private int selectedGender = -1; // 0 for Male, 1 for Female

    private final ActivityResultLauncher<Intent>
heightActivityLauncher =
            registerForActivityResult(new
ActivityResultContracts.StartActivityForResult(), result -> {
                if (result.getResultCode() == RESULT_OK &&
result.getData() != null) {
                    selectedGender =
result.getData().getIntExtra("gender", -1);
                    updateGenderSelection();
                }
            });

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //EdgeToEdge.enable(this);
        setContentView(R.layout.activity_gender);

        maleButton = findViewById(R.id.maleOption);
        femaleButton = findViewById(R.id.femaleOption);
        continueButton = findViewById(R.id.weightcontinueButton);
```

```
        maleButton.setOnClickListener(v -> {

maleButton.setBackgroundResource(R.drawable.selected_gender);
            femaleButton.setBackgroundResource(0);
            selectedGender = 0;
        });

        femaleButton.setOnClickListener(v -> {

femaleButton.setBackgroundResource(R.drawable.selected_gender);
            maleButton.setBackgroundResource(0);
            selectedGender = 1;
        });

        continueButton.setOnClickListener(v -> {
            if (selectedGender != -1) {
                Intent intent = new Intent(GenderActivity.this,
HeightActivity.class);
                intent.putExtra("gender", selectedGender);
                SharedPreferences.Editor editor =
getSharedPreferences("user_prefs", MODE_PRIVATE).edit();
                String gender_Text="";
                if(selectedGender==0){
                    gender_Text="Male";
                }else if(selectedGender==1){
                    gender_Text="Female";
                }
                Log.e("Gender",gender_Text);
                editor.putString("gender",gender_Text);
                editor.apply();

                heightActivityLauncher.launch(intent);
            } else {
                Toast.makeText(this, "请选择您的性别",
Toast.LENGTH_SHORT).show();
            }
        });


        // Find the TextView
        backButton = findViewById(R.id.weightbackButton);

        // Set click listener
        backButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                Intent intent = new Intent(GenderActivity.this,
MainActivity.class);
                startActivity(intent);
            }
        });
    }

    private void updateGenderSelection() {
        if (selectedGender == 0) {

maleButton.setBackgroundResource(R.drawable.selected_gender);
            femaleButton.setBackgroundResource(0);
```

```
        } else if (selectedGender == 1) {

femaleButton.setBackgroundResource(R.drawable.selected_gender);
            maleButton.setBackgroundResource(0);
        }
    }
}
```

HeightActivity.java

```
package com.example.bmi;

import android.content.SharedPreferences;
import android.os.Bundle;

import androidx.activity.EdgeToEdge;
import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.contract.ActivityResultContracts;
import androidx.appcompat.app.AppCompatActivity;
import android.view.View;
import android.view.ViewGroup;


import android.widget.Button;
import android.widget.ImageView;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.ImageButton;
import android.content.Intent;

public class HeightActivity extends AppCompatActivity {
    private ImageView genderImage;
    private SeekBar heightSeekBar;
    private TextView heightText;
    private TextView heightCmText;
    private Button continueButton;
    private View rulerFill;

    private int selectedHeight = 140; // Default height
    private int gender = -1; // Default gender
    private ActivityResultLauncher<Intent> weightActivityLauncher;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //EdgeToEdge.enable(this);
        setContentView(R.layout.activity_height);

        // Register the launcher inside onCreate
        weightActivityLauncher = registerForActivityResult(
                new ActivityResultContracts.StartActivityForResult(),
result -> {
                    if (result.getResultCode() == RESULT_OK &&
result.getData() != null) {
                        selectedHeight =
result.getData().getIntExtra("height", 140);
                        updateHeightSelection();
                    }
```

```
                });

        // Find UI elements
        ImageButton backButton = findViewById(R.id.heightbackButton);
        genderImage = findViewById(R.id.genderImage);
        heightSeekBar = findViewById(R.id.heightSeekBar);
        heightCmText = findViewById(R.id.heightCmText); // Actual
value shown (e.g. 172 cm)
        continueButton = findViewById(R.id.heightcontinueButton);
        rulerFill = findViewById(R.id.rulerFill);

        // Retrieve gender from previous activity
        gender = getIntent().getIntExtra("gender", -1);
        if (gender == 0) {
            genderImage.setImageResource(R.drawable.male_height);
        } else if (gender == 1) {
            genderImage.setImageResource(R.drawable.female_height);
        }

        // Set SeekBar range (140 to 200 cm)
        heightSeekBar.setMax(60); // Max range: 200 - 140 = 60
        heightSeekBar.setProgress(0); // Default position at 140 cm
        heightCmText.setText(selectedHeight + " cm"); // Initial
display

        heightSeekBar.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int
progress, boolean fromUser) {
                selectedHeight = 140 + progress;
                heightCmText.setText(selectedHeight + " cm");

                // Scale ruler height (max 300dp based on layout)
                int maxHeightPx = (int) (300 *
getResources().getDisplayMetrics().density); // Convert 300dp to
pixels
                float percentage = (selectedHeight - 0) / 200f;
                int fillHeight = (int) (maxHeightPx * percentage);

                ViewGroup.LayoutParams params =
rulerFill.getLayoutParams();
                params.height = fillHeight;
                rulerFill.setLayoutParams(params);
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {}

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {}
        });

        // Back button
        backButton.setOnClickListener(v -> {
            Intent resultIntent = new Intent();
            resultIntent.putExtra("gender", gender);
            setResult(RESULT_OK, resultIntent);
            finish();
        });
```

```
        // Continue button
        continueButton.setOnClickListener(v -> {
            Intent intent = new Intent(HeightActivity.this,
WeightActivity.class);
            SharedPreferences.Editor editor =
getSharedPreferences("user_prefs", MODE_PRIVATE).edit();
            editor.putInt("height", selectedHeight);

            String gender_Text = "";
            if (gender == 0) {
                gender_Text = "Male";
            } else if (gender == 1) {
                gender_Text = "Female";
            }
            editor.putString("gender", gender_Text);

            intent.putExtra("gender", gender);
            intent.putExtra("height", selectedHeight);
            weightActivityLauncher.launch(intent);
        });
    }

    private void updateHeightSelection() {
        if (selectedHeight >= 140 && selectedHeight <= 200) {
            heightCmText.setText(selectedHeight + " cm");
        }
    }
}
```

HomeActivity.java

```
package com.example.bmi;

import android.os.Bundle;
import android.util.Log;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import
com.google.android.material.bottomnavigation.BottomNavigationView;

import java.util.HashMap;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;
import com.bumptech.glide.Glide;
import android.content.SharedPreferences;
import android.view.View;
import android.graphics.BitmapFactory;
import android.util.Base64;
```

```
import android.widget.CheckBox;
import android.widget.CompoundButton;



public class HomeActivity extends AppCompatActivity {

    private TextView greetingText, motivationText;
    private TextView breakfastText, breakfastCal;
    private TextView lunchText, lunchCal;
    private TextView dinnerText, dinnerCal;
    private ImageView breakfastImage, lunchImage, dinnerImage;
    private TextView bmiValue;
    private TextView calorieRemaining, calorieConsumed;
    private android.widget.ProgressBar calorieProgress;
    private TextView bmiStatus;
    private boolean hasTodayPlan = false;
    private int todayBreakfastKcal = 0, todayLunchKcal = 0,
todayDinnerKcal = 0;
    private CheckBox cbBreakfast, cbLunch, cbDinner;
    private View mealDoneRow;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home);

        // Greeting and motivation
        greetingText = findViewById(R.id.greetingText);
        motivationText = findViewById(R.id.motivationText);
        greetingText = findViewById(R.id.greetingText);
        motivationText = findViewById(R.id.motivationText);

        // Meal views
        breakfastText = findViewById(R.id.meal1_name);
        breakfastCal = findViewById(R.id.meal1_calories);
        breakfastImage = findViewById(R.id.meal1_image);

        lunchText = findViewById(R.id.meal2_name);
        lunchCal = findViewById(R.id.meal2_calories);
        lunchImage = findViewById(R.id.meal2_image);

        dinnerText = findViewById(R.id.meal3_name);
        dinnerCal = findViewById(R.id.meal3_calories);
        dinnerImage = findViewById(R.id.meal3_image);
        bmiStatus = findViewById(R.id.bmiStatus);
        bmiValue = findViewById(R.id.bmiValue);

        calorieProgress = findViewById(R.id.calorieProgress);
        calorieRemaining = findViewById(R.id.calorieRemaining);
        calorieConsumed = findViewById(R.id.calorieConsumed);



        // Bottom nav
        BottomNavigationView bottomNav =
findViewById(R.id.bottom_navigation);
        bottomNav.setSelectedItemId(R.id.nav_home);
```

```
        bottomNav.setOnItemSelectedListener(item -> {
            int id = item.getItemId();
            if (id == R.id.nav_home) return true;
            if (id == R.id.nav_tracking) startActivity(new
android.content.Intent(this, TrackingActivity.class));
            if (id == R.id.nav_diet) startActivity(new
android.content.Intent(this, DietaryActivity.class));
            if (id == R.id.nav_voucher) startActivity(new
android.content.Intent(this, VoucherActivity.class));
            if (id == R.id.nav_settings) startActivity(new
android.content.Intent(this, SettingsActivity.class));
            return true;
        });

        mealDoneRow  = findViewById(R.id.meal_done_row);
        cbBreakfast  = findViewById(R.id.cbBreakfast);
        cbLunch      = findViewById(R.id.cbLunch);
        cbDinner     = findViewById(R.id.cbDinner);
        mealDoneRow.setVisibility(View.GONE);

        SharedPreferences prefs = getSharedPreferences("user_prefs",
MODE_PRIVATE);
        int userId = prefs.getInt("user_id", -1);
        loadTodayPlan(userId);

        String username = prefs.getString("username", "User");

        greetingText.setText("Hello, " + username);
        fetchBMIFromServer(userId);

        greetingText.setText("Hello, " + username);

        if (userId != -1) {
            fetchCalorieProgressFromServer(userId);
        } else {
            Toast.makeText(this, "User ID not found",
Toast.LENGTH_SHORT).show();
        }

        checkSignInStatus(userId);

    }

    private void loadTodayPlan(int userId) {
        if (userId == -1) {
            // No user, just go to AI fallback
            generateMealPlanFromAI();
            return;
        }

        String url =
"http://10.0.2.2/nutrition/get_today_plan.php?userid=" + userId;
        okhttp3.OkHttpClient client = new okhttp3.OkHttpClient();
        okhttp3.Request request = new
okhttp3.Request.Builder().url(url).build();

        client.newCall(request).enqueue(new okhttp3.Callback() {
            @Override public void onFailure(okhttp3.Call call,
java.io.IOException e) {
                runOnUiThread(() -> {
```

```
                    Log.e("TODAY_PLAN", "HTTP fail: " +
e.getMessage());
                    // Fallback to AI if server not reachable
                    generateMealPlanFromAI();
                });
            }

            @Override public void onResponse(okhttp3.Call call,
okhttp3.Response response) throws java.io.IOException {
                String json = response.body().string();
                Log.d("TODAY_PLAN_JSON", json);

                try {
                    org.json.JSONObject obj = new
org.json.JSONObject(json);
                    String status = obj.optString("status", "error");

                    if (!"success".equalsIgnoreCase(status)) {
                        // No plan for today -> fallback to AI
                        runOnUiThread(() ->
generateMealPlanFromAI());
                        return;
                    }

                    String b = obj.optString("breakfast_text", "");
                    String l = obj.optString("lunch_text", "");
                    String d = obj.optString("dinner_text", "");
                    int bk = obj.optInt("breakfast_kcal", 0);
                    int lk = obj.optInt("lunch_kcal", 0);
                    int dk = obj.optInt("dinner_kcal", 0);

                    final int breakfastDone =
obj.optInt("breakfast_done", 0);
                    final int lunchDone     =
obj.optInt("lunch_done", 0);
                    final int dinnerDone    =
obj.optInt("dinner_done", 0);


                    Log.d("TODAY_PLAN_FIELDS",
                            "status=" + obj.optString("status") +
                                    ", b=" + b + ", l=" + l + ", d="
+ d +
                                    ", bk=" + bk + ", lk=" + lk + ",
dk=" + dk);

                    hasTodayPlan = true;
                    todayBreakfastKcal = bk;
                    todayLunchKcal = lk;
                    todayDinnerKcal = dk;

                    runOnUiThread(() -> {
                        // Update UI
                        if (!b.isEmpty()) {
                            breakfastText.setText(b);
                            breakfastCal.setText("~" + (bk > 0 ? bk :
350) + " kcal");

fetchMealImage(extractFirstFoodKeyword(b), breakfastImage);
                        }
```

```
                        if (!l.isEmpty()) {
                            lunchText.setText(l);
                            lunchCal.setText("~" + (lk > 0 ? lk :
450) + " kcal");

fetchMealImage(extractFirstFoodKeyword(l), lunchImage);
                        }
                        if (!d.isEmpty()) {
                            dinnerText.setText(d);
                            dinnerCal.setText("~" + (dk > 0 ? dk :
500) + " kcal");

fetchMealImage(extractFirstFoodKeyword(d), dinnerImage);
                        }

                        mealDoneRow.setVisibility(View.VISIBLE);

                        // Avoid triggering listeners while we set
initial state
                        cbBreakfast.setOnCheckedChangeListener(null);
                        cbLunch.setOnCheckedChangeListener(null);
                        cbDinner.setOnCheckedChangeListener(null);

                        // Initial state from server
                        cbBreakfast.setChecked(breakfastDone == 1);
                        cbLunch.setChecked(lunchDone == 1);
                        cbDinner.setChecked(dinnerDone == 1);

                        // Hook listeners after initial state
                        CompoundButton.OnCheckedChangeListener
listener = (button, isChecked) -> {
                            String meal;
                            if (button == cbBreakfast)       meal =
"breakfast";
                            else if (button == cbLunch)      meal =
"lunch";
                            else                              meal =
"dinner";

                            // update DB
                            pushMealDoneToServer(userId, meal,
isChecked ? 1 : 0);

                            // update the progress bar immediately
                            recalcAndShowCalorieProgressFromChecks();
                        };


cbBreakfast.setOnCheckedChangeListener(listener);
                        cbLunch.setOnCheckedChangeListener(listener);

cbDinner.setOnCheckedChangeListener(listener);

                        // Kick the progress bar once with current
checks + kcals
                        recalcAndShowCalorieProgressFromChecks();

                        Toast.makeText(HomeActivity.this, "Loaded
today's plan", Toast.LENGTH_SHORT).show();
                    });
```

```
                } catch (Exception e) {
                    Log.e("TODAY_PLAN", "Parse error: " +
e.getMessage());
                    runOnUiThread(() -> generateMealPlanFromAI());
                }
            }
        });
    }


    private void generateMealPlanFromAI() {
        String prompt =
                "Generate only 3 lines of meal plan in this exact
format and nothing else:\n" +
                        "Breakfast: - item1, item2, item3\n" +
                        "Lunch: - item1, item2, item3\n" +
                        "Dinner: - item1, item2, item3\n\n" +
                        "Rules:\n" +
                        "- Do NOT include introductions, suggestions,
or any sentences.\n" +
                        "- ONLY output these 3 lines exactly, no
explanations.\n" +
                        "- Each line must begin with 'Breakfast:',
'Lunch:', or 'Dinner:' followed by a dash and comma-separated food
items.\n" +
                        "- Do NOT add anything else. End after
'Dinner' line.";

        CohereRequest request = new CohereRequest(prompt);
        CohereApiClient apiClient = new CohereApiClient();
        CohereService service = apiClient.createService();

        service.generate(request).enqueue(new
Callback<CohereResponse>() {
            @Override
            public void onResponse(Call<CohereResponse> call,
Response<CohereResponse> response) {
                if (response.isSuccessful() && response.body() !=
null) {
                    parseMealResponse(response.body().getText());
                } else {
                    Toast.makeText(HomeActivity.this, "Cohere error:
" + response.message(), Toast.LENGTH_SHORT).show();
                }
            }

            @Override
            public void onFailure(Call<CohereResponse> call,
Throwable t) {
                Toast.makeText(HomeActivity.this, "Failed to connect
to AI", Toast.LENGTH_SHORT).show();
            }
        });
    }

    private void parseMealResponse(String text) {
        Log.d("CohereResponse", text);

        Map<String, String> meals = new HashMap<>();
```

```
        // Extract text blocks by keyword
        Pattern pattern =
Pattern.compile("(Breakfast|Lunch|Dinner):\\s*(.*?)(?=(?:\\n(?:Breakf
ast|Lunch|Dinner):|$))", Pattern.DOTALL);
        Matcher matcher = pattern.matcher(text);

        while (matcher.find()) {
            String mealType = matcher.group(1);
            String mealDetails = matcher.group(2).trim();
            meals.put(mealType, mealDetails);
        }

        if (meals.containsKey("Breakfast")) {
            breakfastText.setText(meals.get("Breakfast"));
            breakfastCal.setText("350 kcal");

fetchMealImage(extractFirstFoodKeyword(meals.get("Breakfast")),
breakfastImage);
        }

        if (meals.containsKey("Lunch")) {
            lunchText.setText(meals.get("Lunch"));
            lunchCal.setText("450 kcal");

fetchMealImage(extractFirstFoodKeyword(meals.get("Lunch")),
lunchImage);
        }

        if (meals.containsKey("Dinner")) {
            dinnerText.setText(meals.get("Dinner"));
            dinnerCal.setText("500 kcal");

fetchMealImage(extractFirstFoodKeyword(meals.get("Dinner")),
dinnerImage);
        }
    }

    private String formatMeal(String raw) {
        String[] foods = raw.split(",");
        StringBuilder result = new StringBuilder();
        for (String food : foods) {
            result.append(food.trim()).append("\n");
        }
        return result.toString().trim();
    }

    private String getRandomMotivation() {
        String[] quotes = {
                "Let's make today a healthy one!",
                "You're one meal away from a better day.",
                "Fuel your body. Feed your mind.",
                "Strong body, strong mind.",
                "Eat well, feel amazing."
        };
        return quotes[(int) (Math.random() * quotes.length)];
    }

    // REPLACE the whole method with this improved version
    private void fetchMealImage(String query, ImageView imageView) {
```

```
        try {
            // URL-encode
            String q = java.net.URLEncoder.encode(query, "UTF-8");
            String url = "https://api.pexels.com/v1/search?query=" +
q +
                    "&per_page=12&orientation=landscape&size=medium";

            okhttp3.OkHttpClient client = new okhttp3.OkHttpClient();
            okhttp3.Request request = new okhttp3.Request.Builder()
                    .url(url)
                    .addHeader("Authorization",
"aGuhprX2IJ2guNswKqnQyhR08nW2MK5x0hbKmRCC58D44vFyCqvJp8ZU")
                    .build();

            client.newCall(request).enqueue(new okhttp3.Callback() {
                @Override public void onFailure(okhttp3.Call call,
java.io.IOException e) {
                    Log.e("PEXELS", "Failed: " + e.getMessage());
                }

                @Override public void onResponse(okhttp3.Call call,
okhttp3.Response response) throws java.io.IOException {
                    if (!response.isSuccessful() || response.body()
== null) {
                        Log.e("PEXELS", "HTTP " + response.code());
                        return;
                    }
                    try {
                        String json = response.body().string();
                        org.json.JSONObject obj = new
org.json.JSONObject(json);
                        org.json.JSONArray photos =
obj.optJSONArray("photos");
                        if (photos == null || photos.length() == 0)
return;

                        // Pick the first photo whose alt looks like
FOOD (not hands/people/drinks)
                        int chosen = -1;
                        String queryLower = query.toLowerCase();
                        for (int i = 0; i < photos.length(); i++) {
                            org.json.JSONObject p =
photos.getJSONObject(i);
                            String alt = p.optString("alt",
"").toLowerCase();

                            boolean looksLikeFood =
                                    alt.contains("food") ||
alt.contains("dish") || alt.contains("meal") ||
                                    alt.contains("plate") ||
alt.contains("bowl") || alt.contains("salad") ||
                                    alt.contains("rice")  ||
alt.contains("noodle") || alt.contains("pasta") ||
                                    alt.contains("chicken")
|| alt.contains("salmon") || alt.contains("egg") ||
                                    alt.contains("tofu") ||
alt.contains("beef") || alt.contains("vegetable") ||
                                    alt.contains(queryLower);

                            boolean mentionsPeople =
```

```
alt.matches(".*\\b(man|woman|people|person|friends|couple|hands)\\b.*
");

                            boolean drinkOnly =
                                    alt.contains("wine") ||
alt.contains("coffee") || alt.contains("tea") ||
                                            alt.contains("cocktail")
|| alt.contains("juice");

                            if (looksLikeFood && !mentionsPeople
&& !drinkOnly) {
                                chosen = i;
                                break;
                            }
                        }
                        if (chosen == -1) chosen = 0; // fallback
first image

                        String imageUrl =
photos.getJSONObject(chosen).getJSONObject("src").getString("medium")
;
                        runOnUiThread(() ->
Glide.with(HomeActivity.this).load(imageUrl).into(imageView));
                    } catch (Exception e) {
                        Log.e("PEXELS", "Parse error: " +
e.getMessage());
                    }
                }
            });
        } catch (Exception ex) {
            Log.e("PEXELS", "Build URL error: " + ex.getMessage());
        }
    }


    private String extractFirstFoodKeyword(String raw) {
        if (raw == null) return "healthy meal";

        // Keep the first 2 food items to make the query more
specific
        String[] parts = raw.split(",");
        java.util.List<String> cleaned = new java.util.ArrayList<>();
        for (int i = 0; i < parts.length && cleaned.size() < 2; i++)
{
            String s = parts[i]
                    .replaceAll("\\(.*?\\)", "")   // drop
parentheses
                    .replaceAll("[^a-zA-Z0-9 ]", " ")
                    .toLowerCase()
                    .trim();
            if (s.isEmpty()) continue;
            // skip tiny/stop words for the query
            if (s.matches("(?i)^(and|with|on|in|of|the|a)$"))
continue;
            cleaned.add(s.replaceAll("\\s+", " "));
        }
        if (cleaned.isEmpty()) return "healthy meal";

        // Join first two items to guide Pexels
```

```
        String base = String.join(" ", cleaned);

        // Enrich to bias toward plated food photos
        return base + " food dish meal recipe on plate close-up";
    }


    private void showCalorieProgress(int consumed, int goal) {
        calorieProgress.setMax(goal);
        calorieProgress.setProgress(consumed);
        calorieRemaining.setText((goal - consumed) + " kcal
remaining");
        calorieConsumed.setText(consumed + " / " + goal + " kcal");
    }

    private void fetchCalorieProgressFromServer(int userid) {
        okhttp3.OkHttpClient client = new okhttp3.OkHttpClient();
        okhttp3.RequestBody formBody = new okhttp3.FormBody.Builder()
                .add("action", "get_calorie_progress")
                .add("userid", String.valueOf(userid))
                .build();

        okhttp3.Request request = new okhttp3.Request.Builder()
                .url("http://10.0.2.2/nutrition/login.php")
                .post(formBody)
                .build();

        client.newCall(request).enqueue(new okhttp3.Callback() {
            @Override
            public void onFailure(okhttp3.Call call,
java.io.IOException e) {
                runOnUiThread(() -> Toast.makeText(HomeActivity.this,
"Failed to fetch calories", Toast.LENGTH_SHORT).show());
            }

            @Override
            public void onResponse(okhttp3.Call call,
okhttp3.Response response) throws java.io.IOException {
                String json = response.body().string();
                Log.d("CALORIE_JSON", json);
                try {
                    org.json.JSONObject obj = new
org.json.JSONObject(json);
                    if (obj.getString("status").equals("success") ||
obj.getString("status").equals("no_data")) {
                        int consumed =
obj.getInt("calories_consumed");
                        int goal = obj.getInt("calories_goal");
                        int savedGoal = todayBreakfastKcal +
todayLunchKcal + todayDinnerKcal;
                        if (savedGoal > 0) goal = savedGoal;
                        final int consumedFinal = consumed;
                        final int goalFinal = goal;
                        runOnUiThread(() ->
showCalorieProgress(consumedFinal, goalFinal));
                    } else {
                        runOnUiThread(() ->
Toast.makeText(HomeActivity.this, "Progress not available",
Toast.LENGTH_SHORT).show());
                    }
```

```
                } catch (Exception e) {
                    Log.e("CALORIE", "JSON error: " +
e.getMessage());
                }
            }
        });
    }

    private void fetchBMIFromServer(int userId) {
        okhttp3.OkHttpClient client = new okhttp3.OkHttpClient();
        okhttp3.RequestBody formBody = new okhttp3.FormBody.Builder()
                .add("action", "get_bmi")
                .add("userid", String.valueOf(userId))
                .build();

        okhttp3.Request request = new okhttp3.Request.Builder()
                .url("http://10.0.2.2/nutrition/login.php") // update
if needed
                .post(formBody)
                .build();

        client.newCall(request).enqueue(new okhttp3.Callback() {
            @Override
            public void onFailure(okhttp3.Call call,
java.io.IOException e) {
                runOnUiThread(() ->
                        Toast.makeText(HomeActivity.this, "Failed to
fetch BMI", Toast.LENGTH_SHORT).show());
            }

            @Override
            public void onResponse(okhttp3.Call call,
okhttp3.Response response) throws java.io.IOException {
                String json = response.body().string();
                try {
                    org.json.JSONObject obj = new
org.json.JSONObject(json);
                    if (obj.getString("status").equals("success")) {
                        double bmi = obj.getDouble("bmi");
                        runOnUiThread(() -> showBMIProgress(bmi));
                    }
                } catch (Exception e) {
                    Log.e("BMI_FETCH", "Error parsing BMI JSON: " +
e.getMessage());
                }
            }
        });
    }

    private void showBMIProgress(double bmi) {
        bmiValue.setText(String.format("%.1f", bmi));

        String condition;
        String suggestion = "";

        if (bmi < 18.5) {
            condition = "Underweight";
            suggestion = "Consider a balanced, calorie-rich diet.";
        } else if (bmi < 24.9) {
            condition = "Normal";
```

```
        } else if (bmi < 29.9) {
            condition = "Overweight";
            suggestion = "Try moderating sugar and fried food
intake.";
        } else {
            condition = "Obese";
            suggestion = "Limit fast food and increase vegetable
intake.";
        }

        String finalMessage = "Status: " + condition;
        if (!suggestion.isEmpty()) {
            finalMessage += "\n" + suggestion;
        }

        bmiStatus.setText(finalMessage);
    }

    private void checkSignInStatus(int userId) {
        okhttp3.OkHttpClient client = new okhttp3.OkHttpClient();
        okhttp3.RequestBody formBody = new okhttp3.FormBody.Builder()
                .add("action", "signin_reward")
                .add("userid", String.valueOf(userId))
                .build();

        okhttp3.Request request = new okhttp3.Request.Builder()
                .url("http://10.0.2.2/nutrition/login.php")
                .post(formBody)
                .build();

        client.newCall(request).enqueue(new okhttp3.Callback() {
            @Override
            public void onFailure(okhttp3.Call call,
java.io.IOException e) {
                runOnUiThread(() ->
                        Toast.makeText(HomeActivity.this, "Failed to
check sign-in reward", Toast.LENGTH_SHORT).show());
            }

            @Override
            public void onResponse(okhttp3.Call call,
okhttp3.Response response) throws java.io.IOException {
                String json = response.body().string();
                try {
                    org.json.JSONObject obj = new
org.json.JSONObject(json);
                    String status = obj.getString("status");

                    Log.d("SIGNIN_FLOW", "Server status: " + status);

                    if (status.equals("already_claimed")) {
                        // Do nothing
                    } else if (status.equals("no_record") ||
status.equals("needs_update")) {
                        runOnUiThread(() -> {
                            Log.d("SIGNIN_FLOW", "Triggering popup
for user: " + userId);
                            showSignInPopup(userId);
                        });
                    } else {
```

```
                        Log.d("SIGNIN_FLOW", "Unexpected status: " +
status);
                    }

                } catch (Exception e) {
                    Log.e("SIGNIN_REWARD", "JSON error: " +
e.getMessage());
                }
            }
        });
    }

    private void showSignInPopup(int userId) {
        android.app.Dialog dialog = new android.app.Dialog(this);
        dialog.setContentView(R.layout.popup_signin);
        dialog.getWindow().setBackgroundDrawable(
                new
android.graphics.drawable.ColorDrawable(android.graphics.Color.TRANSP
ARENT)
        );
        dialog.setCancelable(false);

        TextView title = dialog.findViewById(R.id.title);
        TextView message = dialog.findViewById(R.id.message);
        ImageView coinIcon = dialog.findViewById(R.id.coinIcon);
        android.widget.Button claimButton =
dialog.findViewById(R.id.claimButton);

        title.setText("Daily Sign-In Reward");
        message.setText("Click to claim your reward!");
        coinIcon.setImageResource(R.drawable.coin_reward);

        claimButton.setOnClickListener(v -> {
            // Send sign-in reward request
            okhttp3.OkHttpClient client = new okhttp3.OkHttpClient();
            okhttp3.RequestBody formBody = new
okhttp3.FormBody.Builder()
                    .add("action", "claim_signin_reward")
                    .add("userid", String.valueOf(userId))
                    .build();

            okhttp3.Request request = new okhttp3.Request.Builder()
                    .url("http://10.0.2.2/nutrition/login.php")
                    .post(formBody)
                    .build();

            client.newCall(request).enqueue(new okhttp3.Callback() {
                @Override
                public void onFailure(okhttp3.Call call,
java.io.IOException e) {
                    runOnUiThread(() -> {
                        Toast.makeText(HomeActivity.this, "Failed to
claim reward", Toast.LENGTH_SHORT).show();
                        dialog.dismiss();
                    });
                }

                @Override
                public void onResponse(okhttp3.Call call,
okhttp3.Response response) throws java.io.IOException {
```

```
                    String json = response.body().string();
                    try {
                        org.json.JSONObject obj = new
org.json.JSONObject(json);
                        if
(obj.getString("status").equals("reward_claimed")) {
                            int points = obj.getInt("points");
                            runOnUiThread(() -> {
                                Toast.makeText(HomeActivity.this, "+
" + points + " points added!", Toast.LENGTH_SHORT).show();
                                dialog.dismiss();
                            });
                        } else if
(obj.getString("status").equals("already_claimed")) {
                            runOnUiThread(() -> {
                                Toast.makeText(HomeActivity.this,
"Already claimed today", Toast.LENGTH_SHORT).show();
                                dialog.dismiss();
                            });
                        } else {
                            runOnUiThread(() -> {
                                Toast.makeText(HomeActivity.this,
"Reward error", Toast.LENGTH_SHORT).show();
                                dialog.dismiss();
                            });
                        }
                    } catch (Exception e) {
                        Log.e("SIGNIN_CLAIM", "Error: " +
e.getMessage());
                    }
                }
            });
        });

        dialog.show();
    }

    private void recalcAndShowCalorieProgressFromChecks() {
        int goal = Math.max(0, todayBreakfastKcal) + Math.max(0,
todayLunchKcal) + Math.max(0, todayDinnerKcal);
        int consumed = 0;
        if (cbBreakfast.isChecked()) consumed += Math.max(0,
todayBreakfastKcal);
        if (cbLunch.isChecked())     consumed += Math.max(0,
todayLunchKcal);
        if (cbDinner.isChecked())    consumed += Math.max(0,
todayDinnerKcal);
        showCalorieProgress(consumed, goal);
    }

    private void pushMealDoneToServer(int userId, String meal, int
done) {
        okhttp3.OkHttpClient client = new okhttp3.OkHttpClient();
        okhttp3.RequestBody form = new okhttp3.FormBody.Builder()
                .add("user_id", String.valueOf(userId))   // <-- keep
user_id here
                .add("meal", meal)                        //
"breakfast" | "lunch" | "dinner"
                .add("done", String.valueOf(done))        // 0 or 1
                .build();
```

```
        okhttp3.Request req = new okhttp3.Request.Builder()
                .url("http://10.0.2.2/nutrition/update_meal_done.php"
)
                .post(form)
                .build();

        client.newCall(req).enqueue(new okhttp3.Callback() {
            @Override public void onFailure(okhttp3.Call call,
java.io.IOException e) {
                runOnUiThread(() -> Toast.makeText(HomeActivity.this,
"Update failed", Toast.LENGTH_SHORT).show());
            }
            @Override public void onResponse(okhttp3.Call call,
okhttp3.Response response) throws java.io.IOException {
                String json = response.body().string();
                try {
                    org.json.JSONObject obj = new
org.json.JSONObject(json);
                    if ("success".equals(obj.optString("status"))) {
                        int consumed = obj.optInt("consumed", 0);
                        int goal = obj.optInt("goal", 0);
                        runOnUiThread(() ->
showCalorieProgress(consumed, goal));
                    } else {
                        runOnUiThread(() ->
Toast.makeText(HomeActivity.this, "Server rejected update",
Toast.LENGTH_SHORT).show());
                    }
                } catch (Exception ignore) {}
            }
        });
    }
}
```

LoginActivity.java

```
package com.example.bmi;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.TextView;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.EditText;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;

import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import android.text.InputType;

import org.json.JSONArray;
```

```
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;


public class LoginActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        // Initialize UI components
        EditText userName = findViewById(R.id.userName);
        EditText userPassword = findViewById(R.id.userPassword);
        Button loginButton = findViewById(R.id.loginButton);

        ImageButton togglePassword =
findViewById(R.id.togglePasswordVisibility);
        final boolean[] isPasswordVisible = {false}; // false =
password hidden

        togglePassword.setOnClickListener(v -> {
            if (isPasswordVisible[0]) {
                // Hide password
                userPassword.setInputType(InputType.TYPE_CLASS_TEXT |
InputType.TYPE_TEXT_VARIATION_PASSWORD);

togglePassword.setImageResource(R.drawable.ic_visibility_on_24); //
show "eye open" when hidden
            } else {
                // Show password
                userPassword.setInputType(InputType.TYPE_CLASS_TEXT |
InputType.TYPE_TEXT_VARIATION_VISIBLE_PASSWORD);

togglePassword.setImageResource(R.drawable.visibility_off); // show
"eye with slash" when visible
            }

            isPasswordVisible[0] = !isPasswordVisible[0]; // toggle
state

userPassword.setSelection(userPassword.getText().length()); // keep
cursor at end
        });


        loginButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String enteredGmail =
userName.getText().toString().trim();
                String enteredPassword =
userPassword.getText().toString().trim();

                if (enteredGmail.isEmpty() ||
```

```
enteredPassword.isEmpty()) {
                    Toast.makeText(LoginActivity.this, "Please enter
all fields", Toast.LENGTH_SHORT).show();
                    return;
                }

                // Send login request
                String url = "http://10.0.2.2/nutrition/login.php";
// Change if using real device or different IP

                StringRequest stringRequest = new
StringRequest(Request.Method.POST, url,
                        response -> {
                            try {
                                JSONObject jsonObject = new
JSONObject(response);
                                String status =
jsonObject.getString("status");

                            Log.e("123123123",response);
                                if ("login_success".equals(status)) {
                                    int userId =
jsonObject.getInt("userid");
                                    String username =
jsonObject.optString("username", "User");

                                    // Get profile fields from server
                                    String genderStr =
jsonObject.optString("gender", "");
                                    String ageStr =
jsonObject.optString("age", "");
                                    String heightStr =
jsonObject.optString("height", "");
                                    String weightStr =
jsonObject.optString("weight", "");
                                    String targetBMIStr =
jsonObject.optString("target_BMI", "");
                                    String
allergies=jsonObject.optString("allergies", "");
                                    // Convert gender (0 or 1) to
string label
                                    String gender;
                                    if ("0".equals(genderStr)) {
                                        gender = "Male";
                                    } else if ("1".equals(genderStr))
{
                                        gender = "Female";
                                    } else {
                                        gender = "";  // empty or
"null"
                                    }

                                    // Convert other fields safely
                                    int age = ("null".equals(ageStr)
|| ageStr.isEmpty()) ? -1 : Integer.parseInt(ageStr);
                                    int height =
("null".equals(heightStr) || heightStr.isEmpty()) ? -1 :
Integer.parseInt(heightStr);
                                    float weight =
("null".equals(weightStr) || weightStr.isEmpty()) ? -1f :
```

```
Float.parseFloat(weightStr);
                                    float targetBMI =
("null".equals(targetBMIStr) || targetBMIStr.isEmpty()) ? -1f :
Float.parseFloat(targetBMIStr);

                                    // Get allergies array


                                    // Save to SharedPreferences
                                    SharedPreferences prefs =
getSharedPreferences("user_prefs", MODE_PRIVATE);
                                    SharedPreferences.Editor editor =
prefs.edit();
                                    editor.putInt("user_id", userId);
                                    editor.putString("username",
username);

                                    editor.putString("gender",
gender);

                                    editor.putInt("age", age);
                                    editor.putInt("height", height);
                                    editor.putFloat("weight",
weight);
                                    editor.putFloat("target_BMI",
targetBMI);

                                    editor.putString("allergies",
allergies);
//                                    Log.e("123123123",allergies);

Log.e("Login",String.valueOf(userId));
                                    Log.e("Login",gender);

Log.e("Login",String.valueOf(age));

Log.e("Login",String.valueOf(height));

Log.e("Login",String.valueOf(weight));

Log.e("Login",String.valueOf(targetBMI));
                                    Log.e("Login",allergies);

                                    editor.apply();


                                    // Debug log
//                                    Log.d("LoginPrefs", "userid: "
+ userId +
//                                            ", gender: " + gender +
//                                            ", age: " + age +
//                                            ", height: " + height +
//                                            ", weight: " + weight +
//                                            ", target_BMI: " +
targetBMI +
//                                            ", allergies: " +
allergies);


                                    // Check if profile is incomplete
                                    if (gender.isEmpty() || age == -1
|| height == -1 || weight == -1f) {
```

```
Toast.makeText(LoginActivity.this, "Please complete your profile",
Toast.LENGTH_SHORT).show();
                                        prefs =
getSharedPreferences("user_prefs", MODE_PRIVATE);
                                        editor = prefs.edit();
                                        editor.putInt("user_id",
userId);
                                        startActivity(new
Intent(LoginActivity.this, GenderActivity.class));
                                    } else {

Toast.makeText(LoginActivity.this, "Login successful",
Toast.LENGTH_SHORT).show();
                                        startActivity(new
Intent(LoginActivity.this, HomeActivity.class));
                                    }

                                    finish();
                                } else {

Toast.makeText(LoginActivity.this, "Invalid email or password",
Toast.LENGTH_SHORT).show();
                                }
                            } catch (Exception e) {
                                e.printStackTrace();
                                Toast.makeText(LoginActivity.this,
"An error occurred", Toast.LENGTH_SHORT).show();
                            }

                        },
                        error -> Toast.makeText(LoginActivity.this,
"Error: " + error.getMessage(), Toast.LENGTH_LONG).show()
                ) {
                    @Override
                    protected Map<String, String> getParams() {
                        Map<String, String> params = new HashMap<>();
                        params.put("action", "login");
                        params.put("gmail", enteredGmail);
                        params.put("password", enteredPassword);
                        return params;
                    }
                };

                RequestQueue queue =
Volley.newRequestQueue(LoginActivity.this);
                queue.add(stringRequest);
            }
        });

        // Find the TextView
        TextView registerLink = findViewById(R.id.registerLink);

        // Set click listener
        registerLink.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Navigate to RegisterActivity
                Intent intent = new Intent(LoginActivity.this,
RegisterActivity.class);
```

```
                startActivity(intent);
            }
        });

        // Find the TextView
        ImageButton loginbackButton =
findViewById(R.id.loginbackButton);

        // Set click listener
        loginbackButton.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View v) {
                // Navigate to RegisterActivity
                Intent intent = new Intent(LoginActivity.this,
MainActivity.class);
                startActivity(intent);
            }
        });
    }
}
```

MainActivity.java

```
package com.example.bmi;

import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.splashscreen.SplashScreen;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        SplashScreen.installSplashScreen(this);
        super.onCreate(savedInstanceState);


        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);



ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
(v, insets) -> {
            Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom);
            return insets;
```

```
        });


        Button loginButton = findViewById(R.id.btnLogin);
        Button signUpButton = findViewById(R.id.btnSignUp);

        loginButton.setOnClickListener(v -> {
            Intent intent = new Intent(MainActivity.this,
LoginActivity.class);
            startActivity(intent);
        });

        signUpButton.setOnClickListener(v -> {
            Intent intent = new Intent(MainActivity.this,
RegisterActivity.class);
            startActivity(intent);
        });
    }
}
```

MealReminderReceiver.java

```
package com.example.bmi;

import android.app.AlarmManager;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Build;
import android.util.Log;

import androidx.core.app.NotificationCompat;
import androidx.core.app.NotificationManagerCompat;

import java.util.Calendar;
import java.util.TimeZone;

public class MealReminderReceiver extends BroadcastReceiver {

    private static final String CH_ID = "meal_reminders";
    private static final int REQ_BF = 21001;
    private static final int REQ_LU = 21002;
    private static final int REQ_DI = 21003;
    private static final String PREFS = "user_prefs"; // NEW: Added
for clarity
    private static final String KEY_MEAL_NOTIF_ENABLED_PREFIX =
"meal_notif_enabled_user_"; // NEW: User-specific key

    @Override
    public void onReceive(Context context, Intent intent) {
        // NEW: Get user ID (assuming same logic as SettingsActivity)
        String uid = getCurrentUserId(context);
        Log.d("MealReminderReceiver", "onReceive called with
requestCode: " + intent.getIntExtra("requestCode", -1));
```

```
        // CHANGED: Use user-specific key
        boolean enabled = context.getSharedPreferences(PREFS,
Context.MODE_PRIVATE)
                .getBoolean(KEY_MEAL_NOTIF_ENABLED_PREFIX + uid,
false);
        Log.d("MealReminderReceiver", "Notifications enabled for user
" + uid + ": " + enabled);
        if (!enabled) return;

        String msg = intent.getStringExtra("message");
        int id = intent.getIntExtra("requestCode", 0);

        createChannel(context);

        NotificationCompat.Builder nb = new
NotificationCompat.Builder(context, CH_ID)
                .setSmallIcon(R.drawable.ic_notifications)
                .setContentTitle("HealthyPlan")
                .setContentText(msg != null ? msg : "Meal time!")
                .setAutoCancel(true)
                .setPriority(NotificationCompat.PRIORITY_HIGH);

        NotificationManagerCompat.from(context).notify(id,
nb.build());

        int hour24 = 8, minute = 0;
        if (id == REQ_LU) { hour24 = 13; }
        else if (id == REQ_DI) { hour24 = 19; }
        scheduleDaily(context, id, hour24, minute, msg);
    }

    private void createChannel(Context ctx) {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            NotificationChannel ch = new NotificationChannel(
                    CH_ID, "Meal Reminders",
                    NotificationManager.IMPORTANCE_HIGH
            );
            NotificationManager nm =
ctx.getSystemService(NotificationManager.class);
            if (nm == null) {
                Log.e("MealReminderReceiver", "NotificationManager is
null!");
                return;
            }
            nm.createNotificationChannel(ch);
        }
    }

    private void scheduleDaily(Context ctx, int requestCode, int
hour24, int minute, String message) {
        AlarmManager am = (AlarmManager)
ctx.getSystemService(Context.ALARM_SERVICE);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S
&& !am.canScheduleExactAlarms()) {
            Log.e("MealReminderReceiver", "Cannot schedule exact
alarms!");
            return;
        }
```

```
        Intent i = new Intent(ctx, MealReminderReceiver.class);
        i.setAction("com.example.bmi.MEAL_REMINDER");
        i.putExtra("message", message);
        i.putExtra("requestCode", requestCode);

        PendingIntent pi = PendingIntent.getBroadcast(
                ctx, requestCode, i,
                PendingIntent.FLAG_UPDATE_CURRENT |
PendingIntent.FLAG_IMMUTABLE
        );

        Calendar cal = Calendar.getInstance(TimeZone.getDefault());
        cal.set(Calendar.SECOND, 0);
        cal.set(Calendar.MILLISECOND, 0);
        cal.set(Calendar.HOUR_OF_DAY, hour24);
        cal.set(Calendar.MINUTE, minute);
        if (cal.getTimeInMillis() <= System.currentTimeMillis()) {
            cal.add(Calendar.DAY_OF_YEAR, 1);
        }
        Log.d("MealReminderReceiver", "Scheduling alarm for
requestCode: " + requestCode + " at " + cal.getTime());
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            am.setExactAndAllowWhileIdle(AlarmManager.RTC_WAKEUP,
cal.getTimeInMillis(), pi);
        } else {
            am.setExact(AlarmManager.RTC_WAKEUP,
cal.getTimeInMillis(), pi);
        }
    }

    // NEW: Reused getCurrentUserId from SettingsActivity
    private String getCurrentUserId(Context context) {
        SharedPreferences sp =
context.getSharedPreferences("session_prefs", Context.MODE_PRIVATE);
        String uid = sp.getString("user_id", null);
        return (uid == null || uid.trim().isEmpty()) ? "guest" :
uid.trim();
    }
}
```

PersonalizedplanActivity.java

```
package com.example.bmi;

import android.content.SharedPreferences;
import android.graphics.Bitmap;
import android.graphics.drawable.BitmapDrawable;
import android.os.Bundle;
import android.util.Base64;
import android.util.Log;
import android.view.View;
import android.widget.*;
import androidx.appcompat.app.AppCompatActivity;
import com.bumptech.glide.Glide;
import org.json.JSONArray;
import org.json.JSONObject;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
```

```
import java.util.ArrayList;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import android.text.InputType;
import okhttp3.Call;
import okhttp3.Callback;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;
import java.util.Map;
import java.util.HashMap;
import android.content.Intent;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;

public class PersonalizedplanActivity extends AppCompatActivity {

    Spinner planSpinner;
    ArrayList<SavedPlan> savedPlans = new ArrayList<>();
    ArrayAdapter<SavedPlan> plansAdapter;
    private Integer lastSelectedPlanId = null;

    TextView breakfastText, lunchText, dinnerText, breakfastCal,
lunchCal, dinnerCal;
    ImageView breakfastImg, lunchImg, dinnerImg;
    TextView totalCaloriesText, nutritionSummaryText;
    Button shuffleBreakfastBtn, shuffleLunchBtn, shuffleDinnerBtn;
    Button saveTodayBtn, saveLaterBtn;
    ImageButton backButton;
    SharedPreferences prefs;

    double currentBMI = -1;
    double targetBMI = -1;

    private static final int COHERE_MAX_RETRIES = 3;
    private static final long COHERE_RETRY_BACKOFF_MS = 600;
    private static final String COHERE_STRICT_FORMAT =
            "Return EXACTLY one line in this regex format: " +
                    "^(Breakfast|Lunch|Dinner): ~[2-9][0-
9]{2}?\\s*kcal - [a-z][a-z0-9 ,&/()-]{5,}$";
    private retrofit2.Call<CohereResponse> inflightBreakfastCall;
    private retrofit2.Call<CohereResponse> inflightLunchCall;
    private retrofit2.Call<CohereResponse> inflightDinnerCall;


    private static final Pattern KCAL_PATTERN =
            Pattern.compile("(~?\\s*)(\\d{2,4})\\s*kcal",
Pattern.CASE_INSENSITIVE);
    private String q;

    private int parseKcalLabel(String label) {
        if (label == null) return 0;
        Matcher m = Pattern.compile("(\\d{2,4})\\s*kcal",
Pattern.CASE_INSENSITIVE).matcher(label);
        return m.find() ? Integer.parseInt(m.group(1)) : 0;
    }


    private static final Map<String, Integer> FOOD_KCAL_HINTS = new
```

```
HashMap<String, Integer>() {{
        put("oats", 150); put("oat", 150);
        put("banana", 100);
        put("almond milk", 40); put("milk", 120);
        put("chicken", 220); put("breast", 165);
        put("rice", 200); put("brown rice", 216);
        put("egg", 78); put("eggs", 156);
        put("salad", 80);
        put("salmon", 230);
        put("noodles", 300); put("noodle", 250);
        put("yogurt", 120);
        put("apple", 95);
    }};

    // Cache last kcals for total
    private int lastBreakfastKcal = 0, lastLunchKcal = 0,
lastDinnerKcal = 0;

    private int[] lastBreakfastMacros = new int[]{0,0,0};
    private int[] lastLunchMacros     = new int[]{0,0,0};
    private int[] lastDinnerMacros    = new int[]{0,0,0};

    private String coerceToOneLine(String meal, String raw) {
        if (raw == null) return "";
        String[] lines = raw.split("\\r?\\n");
        // Prefer a line that starts with the label, e.g., "Lunch:"
        for (String L : lines) {
            String s = L.trim();
            if (s.toLowerCase().startsWith(meal.toLowerCase() + ":"))
return s;
        }
        // Otherwise pick the first line that has 'kcal'
        for (String L : lines) {
            String s = L.trim();
            if (s.toLowerCase().contains("kcal")) return s;
        }
        // Fallback: collapse everything into one line
        return raw.replaceAll("\\s+", " ").trim();
    }

    // accepts: "Lunch: ~520 kcal - item1, item2, item3"
    private static final Pattern STRICT_LINE =

Pattern.compile("^(Breakfast|Lunch|Dinner):\\s*~(\\d{2,4})\\s*kcal\\s
*-\\s*([a-z].*)$",
                    Pattern.CASE_INSENSITIVE);

    // remove banned words like "about", "approximately" if they
appear before kcal or in items
    private String sanitizeMealLine(String meal, String line) {
        String s =
line.replaceAll("(?i)\\b(about|approximately|around|roughly|estimate|
estimated|circa)\\b\\.?\\s*", "");
        // collapse spaces
        s = s.replaceAll("\\s+", " ").trim();
        // ensure it starts with the meal label
        if (!s.toLowerCase().startsWith(meal.toLowerCase() + ":")) {
            s = meal + ": " + s;
        }
        return s;
```

```
    }

    // final guard: if line doesn't match the strict pattern, return
null (invalid)
    private boolean isValidMealLine(String meal, String line) {
        if (line == null) return false;
        java.util.regex.Matcher m = STRICT_LINE.matcher(line.trim());
        if (!m.matches()) return false;
        // also require it starts with the correct meal label
        return
line.trim().toLowerCase().startsWith(meal.toLowerCase() + ":");
    }

    /** Extract kcal if present; else -1. */
    private int extractKcalIfAny(String text) {
        Matcher m = KCAL_PATTERN.matcher(text);
        if (m.find()) {
            try { return Integer.parseInt(m.group(2)); } catch
(Exception ignored) {}
        }
        return -1;
    }

    /** Estimate kcal from item list (comma-separated or after " -
"). */
    private int estimateKcalFromItems(String text) {
        String itemsPart = text;
        int dashIdx = text.indexOf(" - ");
        if (dashIdx >= 0 && dashIdx + 3 < text.length()) itemsPart =
text.substring(dashIdx + 3);
        itemsPart =
itemsPart.replaceFirst("(?i)^(breakfast|lunch|dinner)\\s*:\\s*", "");

        String[] tokens = itemsPart.split("[,•|]");
        int total = 0, matched = 0;
        for (String raw : tokens) {
            String item = raw.trim().toLowerCase();
            if (item.isEmpty()) continue;
            int best = 0;
            for (String key : FOOD_KCAL_HINTS.keySet()) {
                if (item.contains(key)) best = Math.max(best,
FOOD_KCAL_HINTS.get(key));
            }
            if (best > 0) { total += best; matched++; }
        }
        if (matched == 0) return 350;                  // fallback if
nothing matched
        return Math.max(150, Math.min(total, 900));    // clamp to a
sane range
    }

    /** Normalize items for display (strip labels/kcal, keep "oats,
banana, milk"). */
    private String extractItemsDisplay(String text) {
        int dashIdx = text.indexOf(" - ");
        String items = (dashIdx >= 0) ? text.substring(dashIdx + 3) :
text;
        items =
items.replaceFirst("(?i)^(breakfast|lunch|dinner)\\s*:\\s*", "");
        items = KCAL_PATTERN.matcher(items).replaceAll("");  // drop
```

```
any "xxx kcal"
        items = items.replaceAll("\\s*[--]\\s*$", "").trim();
        items = items.replaceAll("\\s*[,•|]\\s*", ", ");
        return items;
    }

    private String findFirstJsonObject(String s) {
        if (s == null) return null;
        int start = s.indexOf('{');
        int end = s.lastIndexOf('}');
        return (start >= 0 && end > start) ? s.substring(start, end +
1) : null;
    }

    /** First keyword to fetch image for. */
    private String keywordForImage(String itemsDisplay) {
        if (itemsDisplay == null) return "healthy meal";
        String[] parts = itemsDisplay.split(",");
        ArrayList<String> cleaned = new ArrayList<>();
        for (String p : parts) {
            String s = p.trim().toLowerCase();
            if (s.length() < 2) continue;
            if (s.matches("(?i)^(and|with|on|in|of|the|a)$"))
continue;
            cleaned.add(s.replaceAll("[^a-z0-9 ]", "").trim());
        }
        if (cleaned.isEmpty()) return "healthy meal";
        // prefer a protein or main carb as the keyword if present
        for (String s : cleaned) {
            if (s.contains("chicken") || s.contains("salmon") ||
s.contains("egg") || s.contains("beef") ||
                    s.contains("tofu") || s.contains("fish") ||
s.contains("prawn") || s.contains("shrimp")) {
                return s;
            }
        }
        for (String s : cleaned) {
            if (s.contains("rice") || s.contains("noodle") ||
s.contains("pasta") || s.contains("oat")) {
                return s;
            }
        }
        return cleaned.get(0);
    }


    /** One path to set UI (AI or saved plan): items + kcal
(parse/estimate) + image + total. */
    private void applyMealToUI(
            String mealType,        // "Breakfast" | "Lunch" |
"Dinner"
            String rawLine,         // raw text from AI or saved plan
            TextView mealTextView,  // items text view
            TextView kcalTextView,  // kcal text view
            ImageView mealImageView // image view
    ) {
        String itemsDisplay = extractItemsDisplay(rawLine);
        mealTextView.setText(itemsDisplay.isEmpty() ? rawLine :
itemsDisplay);
```

```
        int kcal = extractKcalIfAny(rawLine);
        if (kcal <= 0) kcal = estimateKcalFromItems(rawLine);
        kcalTextView.setText("~" + kcal + " kcal");
        fetchMacrosFromAI(mealType, itemsDisplay, kcal);

        String keyword = keywordForImage(itemsDisplay);
        if (keyword.length() < 2 ||
keyword.matches("(?i)^(about|approximately|around)$")) {
            keyword = mealType.toLowerCase() + " healthy meal";
        }
        fetchImage(keyword, mealImageView);

        switch (mealType) {
            case "Breakfast": lastBreakfastKcal = kcal; break;
            case "Lunch":     lastLunchKcal = kcal;     break;
            case "Dinner":    lastDinnerKcal = kcal;    break;
        }
        updateTotalKcal();
    }

    /** Updates the Total kcal label using cached values. */
    private void updateTotalKcal() {
        int sum = 0;
        if (lastBreakfastKcal > 0) sum += lastBreakfastKcal;
        if (lastLunchKcal > 0) sum += lastLunchKcal;
        if (lastDinnerKcal > 0) sum += lastDinnerKcal;
        if (totalCaloriesText != null && sum > 0) {
            totalCaloriesText.setText("Total: ~" + sum + " kcal");
        }
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.personalizedplan);

        prefs = getSharedPreferences("user_prefs", MODE_PRIVATE);

        planSpinner = findViewById(R.id.plan_spinner);
        plansAdapter = new ArrayAdapter<>(
                PersonalizedplanActivity.this,
                android.R.layout.simple_spinner_item,
                savedPlans
        );

plansAdapter.setDropDownViewResource(android.R.layout.simple_spinner_
dropdown_item);
        planSpinner.setAdapter(plansAdapter);

        planSpinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override public void onItemSelected(AdapterView<?>
parent, View view, int position, long id) {
                if (position >= 0 && position < savedPlans.size()) {
                    SavedPlan selected = savedPlans.get(position);
                    lastSelectedPlanId = selected.id;   // remember
selection
                    loadPlanIntoLayout(selected);
                }
            }
```

```
            @Override public void onNothingSelected(AdapterView<?>
parent) {}
        });


        breakfastText = findViewById(R.id.breakfast_name);
        lunchText = findViewById(R.id.lunch_name);
        dinnerText = findViewById(R.id.dinner_name);
        breakfastCal = findViewById(R.id.breakfast_cal);
        lunchCal = findViewById(R.id.lunch_cal);
        dinnerCal = findViewById(R.id.dinner_cal);
        breakfastImg = findViewById(R.id.breakfast_img);
        lunchImg = findViewById(R.id.lunch_img);
        dinnerImg = findViewById(R.id.dinner_img);
        totalCaloriesText = findViewById(R.id.total_calories);
        nutritionSummaryText = findViewById(R.id.nutrition_summary);

        LinearLayout rowBreakfast = findViewById(R.id.row_breakfast);
        LinearLayout rowLunch     = findViewById(R.id.row_lunch);
        LinearLayout rowDinner    = findViewById(R.id.row_dinner);

        TextView editSavedPlanBtn =
findViewById(R.id.edit_saved_plan_btn);
        editSavedPlanBtn.setOnClickListener(v -> {
            Intent i = new Intent(PersonalizedplanActivity.this,
EditSavedPlanActivity.class);
            startActivity(i);
        });

        rowBreakfast.setOnClickListener(v ->
showMealPopup("Breakfast"));
        rowLunch.setOnClickListener(v -> showMealPopup("Lunch"));
        rowDinner.setOnClickListener(v -> showMealPopup("Dinner"));
        shuffleBreakfastBtn = findViewById(R.id.shuffle_breakfast);
        shuffleLunchBtn = findViewById(R.id.shuffle_lunch);
        shuffleDinnerBtn = findViewById(R.id.shuffle_dinner);
        saveTodayBtn = findViewById(R.id.save_today_btn);
        saveLaterBtn = findViewById(R.id.save_later_btn);
        backButton = findViewById(R.id.backButton);

        backButton.setOnClickListener(v -> finish());

        shuffleBreakfastBtn.setOnClickListener(v ->
generateMeal("Breakfast"));
        shuffleLunchBtn.setOnClickListener(v ->
generateMeal("Lunch"));
        shuffleDinnerBtn.setOnClickListener(v ->
generateMeal("Dinner"));

        saveTodayBtn.setOnClickListener(v ->
promptPlanNameAndSave("today"));
        saveLaterBtn.setOnClickListener(v ->
promptPlanNameAndSave("later"));

        loadSavedPlans();
        fetchBMIValues();
    }

    @Override
    protected void onResume() {
```

```
        super.onResume();
        // Re-fetch plans every time we return to this page
        loadSavedPlans();
    }

    private void fetchBMIValues() {
        int userId = prefs.getInt("user_id", -1);
        if (userId == -1) return;

        String url =
"http://10.0.2.2/nutrition/get_bmi_and_target.php?userid=" + userId;
        OkHttpClient client = new OkHttpClient();
        Request request = new Request.Builder().url(url).build();

        client.newCall(request).enqueue(new Callback() {
            @Override public void onFailure(Call call, IOException e)
{
                Log.e("BMI_FETCH", "Failed to fetch BMI data");
            }

            @Override public void onResponse(Call call, Response
response) throws IOException {
                try {
                    JSONObject obj = new
JSONObject(response.body().string());
                    currentBMI = obj.optDouble("current_bmi", -1);
                    targetBMI = obj.optDouble("target_bmi", -1);

                    runOnUiThread(() -> {
                        SavedPlan sel = getCurrentSelectedPlan();
                        if (sel == null || sel.needAi) {
                            // No plan selected yet OR this plan
explicitly needs AI
                            generateMeal("Breakfast");
                            generateMeal("Lunch");
                            generateMeal("Dinner");
                        }
                        // else: a saved plan is already selected; do
not overwrite it
                    });

                } catch (Exception e) {
                    Log.e("BMI_PARSE", "Error parsing BMI response: "
+ e.getMessage());
                }
            }
        });
    }

    private void generateMeal(String type) {
        if (suppressAI) return;
        cancelInFlight(type);
        askAiForItems(type);
    }

    /** 1) Ask AI for ITEMS ONLY (one line). Then we chain kcal +
macros. */
    private void askAiForItems(String mealType) {
        String allergies = prefs.getString("allergies", "").trim();
```

```
        String prompt =
                "Return EXACTLY one line for " + mealType + " with
ONLY items, no calories.\n" +
                "Format: " + mealType + ": - item1, item2,
item3\n" +
                "Rules:\n" +
                "- 2-5 items, lowercase, 1-3 words each, real
foods only, comma-separated\n" +
                "- no quantities, no brands, no extra text,
no markdown\n" +
                (allergies.isEmpty() ? "" : "- exclude
allergens: " + allergies + "\n") +
                "Return exactly one line, nothing else.";

        CohereRequest req = new CohereRequest(prompt);
        new
CohereApiClient().createService().generate(req).enqueue(new
retrofit2.Callback<CohereResponse>() {
            @Override public void
onResponse(retrofit2.Call<CohereResponse> call,

retrofit2.Response<CohereResponse> res) {
                if (!res.isSuccessful() || res.body() == null ||
res.body().getText() == null) return;

                String line = res.body().getText().trim();
                // Pull just the items part after first '-' or ':'
and normalize
                String items = line;
                int dash = line.indexOf(" - ");
                if (dash >= 0 && dash + 3 < line.length()) items =
line.substring(dash + 3);
                else {
                    int colon = line.indexOf(':');
                    if (colon >= 0 && colon + 1 < line.length())
items = line.substring(colon + 1);
                }
                items = items.replaceAll("^\\s*-\\s*", "").trim();
                items = items.replaceAll("\\s*[,•|]\\s*", ", ");

                if (items.isEmpty()) {
                    items = "oats, banana, yogurt"; // ultra-safe
minimal fallback
                }

                final String itemsFinal = items;
                runOnUiThread(() -> {
                    // 1) Put items on the correct row + image
                    switch (mealType) {
                        case "Breakfast":
                            breakfastText.setText(itemsFinal);
                            fetchImage(keywordForImage(itemsFinal),
breakfastImg);
                            break;
                        case "Lunch":
                            lunchText.setText(itemsFinal);
                            fetchImage(keywordForImage(itemsFinal),
lunchImg);
                            break;
                        case "Dinner":
```

```
                            dinnerText.setText(itemsFinal);
                            fetchImage(keywordForImage(itemsFinal),
dinnerImg);
                            break;
                    }
                    // 2) Ask AI for kcal, then macros
                    askAiForKcal(mealType, itemsFinal);
                });
            }
            @Override public void
onFailure(retrofit2.Call<CohereResponse> call, Throwable t) {
                Log.e("AI_ITEMS", "Call failed: " + t.getMessage());
            }
        });
    }

    /** 2) Ask AI for KCAL ONLY (a single integer). If bad, estimate
locally. Then update UI + macros. */
    private void askAiForKcal(String mealType, String items) {
        String prompt =
                "Given these " + mealType.toLowerCase() + " items: "
+ items + "\n" +
                        "Output ONLY one integer for total kcal
(200..900). No words, no units, no punctuation.";

        CohereRequest req = new CohereRequest(prompt);
        new
CohereApiClient().createService().generate(req).enqueue(new
retrofit2.Callback<CohereResponse>() {
            @Override public void
onResponse(retrofit2.Call<CohereResponse> call,

retrofit2.Response<CohereResponse> res) {
                int kcal = -1;
                try {
                    if (res.isSuccessful() && res.body() != null &&
res.body().getText() != null) {
                        String text = res.body().getText().trim();
                        // keep only digits
                        String digits = text.replaceAll("[^0-9]",
"");
                        if (!digits.isEmpty()) {
                            kcal = Integer.parseInt(digits);
                        }
                    }
                } catch (Exception ignored) {}

                if (kcal < 200 || kcal > 900) {
                    kcal = estimateKcalFromItems(items);
                }

                final int kcalFinal = kcal;
                runOnUiThread(() -> {
                    // Update kcal label + cache
                    switch (mealType) {
                        case "Breakfast":
                            breakfastCal.setText("~" + kcalFinal + "
kcal");
                            lastBreakfastKcal = kcalFinal;
                            break;
```

```
                        case "Lunch":
                            lunchCal.setText("~" + kcalFinal + "
kcal");
                            lastLunchKcal = kcalFinal;
                            break;
                        case "Dinner":
                            dinnerCal.setText("~" + kcalFinal + "
kcal");
                            lastDinnerKcal = kcalFinal;
                            break;
                    }

                    updateTotalKcal();
                    // Ask for macros (you already had this)
                    fetchMacrosFromAI(mealType, items, kcalFinal);
                });
            }
            @Override public void
onFailure(retrofit2.Call<CohereResponse> call, Throwable t) {
                Log.e("AI_KCAL", "Call failed: " + t.getMessage());
                // If API fully fails, estimate locally so UI never
stalls
                int kcal = estimateKcalFromItems(items);
                runOnUiThread(() -> {
                    switch (mealType) {
                        case "Breakfast":
                            breakfastCal.setText("~" + kcal + "
kcal"); lastBreakfastKcal = kcal; break;
                        case "Lunch":
                            lunchCal.setText("~" + kcal + " kcal");
lastLunchKcal = kcal;     break;
                        case "Dinner":
                            dinnerCal.setText("~" + kcal + " kcal");
lastDinnerKcal = kcal;    break;
                    }
                    updateTotalKcal();
                    fetchMacrosFromAI(mealType, items, kcal);
                });
            }
        });
    }


    private void updateMealFromText(String meal, String line) {
        Log.d("AI_RESPONSE", line);
        String oneLine = coerceToOneLine(meal, line);
        oneLine = sanitizeMealLine(meal, oneLine);

        if (!isValidMealLine(meal, oneLine)) {
            String items = extractItemsDisplay(oneLine);
            if (items == null || items.trim().isEmpty()) items =
"healthy oats, banana, yogurt"; // minimal fallback text
            int kcal = extractKcalIfAny(oneLine);
            if (kcal <= 0) kcal = estimateKcalFromItems(items);
            oneLine = meal + ": ~" + Math.max(200, Math.min(kcal,
900)) + " kcal - " + items;
        }

        switch (meal) {
            case "Breakfast":
```

```
                applyMealToUI("Breakfast", oneLine, breakfastText,
breakfastCal, breakfastImg);
                break;
            case "Lunch":
                applyMealToUI("Lunch", oneLine, lunchText, lunchCal,
lunchImg);
                break;
            case "Dinner":
                applyMealToUI("Dinner", oneLine, dinnerText,
dinnerCal, dinnerImg);
                break;
        }
        updateSummary();
    }

    private void updateSummary() {
        updateTotalKcal();

        int p = lastBreakfastMacros[0] + lastLunchMacros[0] +
lastDinnerMacros[0];
        int c = lastBreakfastMacros[1] + lastLunchMacros[1] +
lastDinnerMacros[1];
        int f = lastBreakfastMacros[2] + lastLunchMacros[2] +
lastDinnerMacros[2];

        if (p + c + f > 0) {
            nutritionSummaryText.setText("Protein: ~" + p + " g |
Carbs: ~" + c + " g | Fat: ~" + f + " g");
        } else {
            nutritionSummaryText.setText("Protein: --g | Carbs: --g |
Fat: --g");
        }
    }


    private void loadSavedPlans() {
        int userId = prefs.getInt("user_id", -1);
        if (userId == -1) return;

        String url =
"http://10.0.2.2/nutrition/get_saved_plans.php?userid=" + userId;
        OkHttpClient client = new OkHttpClient();
        Request request = new Request.Builder().url(url).build();

        client.newCall(request).enqueue(new Callback() {
            @Override
            public void onFailure(Call call, IOException e) {
                runOnUiThread(() ->
                        Toast.makeText(PersonalizedplanActivity.this,
"Failed to load plans", Toast.LENGTH_SHORT).show()
                );
            }

            @Override
            public void onResponse(Call call, Response response)
throws IOException {
                try {
                    JSONArray arr = new
JSONArray(response.body().string());
```

```
                    ArrayList<SavedPlan> newList = new ArrayList<>();
                    for (int i = 0; i < arr.length(); i++) {
                        JSONObject obj = arr.getJSONObject(i);
                        SavedPlan plan = new SavedPlan();
                        plan.id        = obj.getInt("id");
                        plan.name      = obj.optString("plan_name",
"");
                        plan.breakfast = obj.optString("breakfast",
"");
                        plan.lunch     = obj.optString("lunch", "");
                        plan.dinner    = obj.optString("dinner", "");
                        plan.createdAt = obj.optString("created_at",
"");
                        plan.needAi    = obj.optInt("need_ai", 0) ==
1;
                        plan.breakfast_kcal =
obj.optInt("breakfast_kcal", 0);
                        plan.lunch_kcal     =
obj.optInt("lunch_kcal", 0);
                        plan.dinner_kcal    =
obj.optInt("dinner_kcal", 0);
                        plan.breakfast_p_g =
obj.optInt("breakfast_p_g", 0);
                        plan.breakfast_c_g =
obj.optInt("breakfast_c_g", 0);
                        plan.breakfast_f_g =
obj.optInt("breakfast_f_g", 0);
                        plan.lunch_p_g = obj.optInt("lunch_p_g", 0);
                        plan.lunch_c_g = obj.optInt("lunch_c_g", 0);
                        plan.lunch_f_g = obj.optInt("lunch_f_g", 0);
                        plan.dinner_p_g = obj.optInt("dinner_p_g",
0);
                        plan.dinner_c_g = obj.optInt("dinner_c_g",
0);
                        plan.dinner_f_g = obj.optInt("dinner_f_g",
0);
                        newList.add(plan);
                    }

                    runOnUiThread(() -> {
                        // 1) Replace list contents
                        savedPlans.clear();
                        savedPlans.addAll(newList);
                        plansAdapter.notifyDataSetChanged();

                        // 2) If no plans exist, clear UI
                        if (savedPlans.isEmpty()) {
                            planSpinner.setSelection(0);
                            breakfastText.setText("--");
                            lunchText.setText("--");
                            dinnerText.setText("--");
                            breakfastCal.setText("-- kcal");
                            lunchCal.setText("-- kcal");
                            dinnerCal.setText("-- kcal");
                            totalCaloriesText.setText("Total: --
kcal");
                            nutritionSummaryText.setText("Protein: --
g | Carbs: --g | Fat: --g");
                            return;
                        }
```

```
                        // ☑ Only auto-select the first plan if
nothing was selected before
                        if (lastSelectedPlanId == null) {
                            lastSelectedPlanId =
savedPlans.get(0).id;
                            planSpinner.setSelection(0);
                        }
                        // Else: do nothing — let user's selection
trigger onItemSelected()
                    });

                } catch (Exception e) {
                    Log.e("SavedPlans", "Error: " + e.getMessage());
                }
            }
        });
    }


    private void loadPlanIntoLayout(SavedPlan plan) {
        // --- 1) Put the saved ITEMS on screen immediately (no AI
changes to items) ---
        // Text
        breakfastText.setText(plan.breakfast == null ? "--" :
plan.breakfast);
        lunchText.setText(plan.lunch == null ? "--" : plan.lunch);
        dinnerText.setText(plan.dinner == null ? "--" : plan.dinner);

        // Pictures (keyword from saved items)
        fetchImage(keywordForImage(plan.breakfast), breakfastImg);
        fetchImage(keywordForImage(plan.lunch),     lunchImg);
        fetchImage(keywordForImage(plan.dinner),    dinnerImg);

        // --- 2) KCAL: use saved if present; otherwise ask AI ONLY
for kcal (do not change items) ---
        if (plan.breakfast_kcal > 0) {
            breakfastCal.setText("~" + plan.breakfast_kcal + "
kcal");
            lastBreakfastKcal = plan.breakfast_kcal;
        } else {
            breakfastCal.setText("~… kcal");
            // Ask AI for kcal from the existing items; this call
will also trigger macros fetch
            askAiForKcal("Breakfast", plan.breakfast);
        }

        if (plan.lunch_kcal > 0) {
            lunchCal.setText("~" + plan.lunch_kcal + " kcal");
            lastLunchKcal = plan.lunch_kcal;
        } else {
            lunchCal.setText("~… kcal");
            askAiForKcal("Lunch", plan.lunch);
        }

        if (plan.dinner_kcal > 0) {
            dinnerCal.setText("~" + plan.dinner_kcal + " kcal");
            lastDinnerKcal = plan.dinner_kcal;
        } else {
            dinnerCal.setText("~… kcal");
```

```
            askAiForKcal("Dinner", plan.dinner);
        }

        // Keep the running total up-to-date using whatever kcal we
have so far
        updateTotalKcal();

        // --- 3) MACROS: if saved -> use them. If missing -> ask AI
using the items + kcal. ---
        if (hasMacros(plan.breakfast_p_g, plan.breakfast_c_g,
plan.breakfast_f_g)) {
            lastBreakfastMacros = new int[]{ plan.breakfast_p_g,
plan.breakfast_c_g, plan.breakfast_f_g };
        } else if (plan.breakfast_kcal > 0) {
            // Only fetch now if we already have kcal; otherwise
askAiForKcal() above will fetch macros later
            fetchMacrosFromAI("Breakfast", plan.breakfast,
plan.breakfast_kcal);
        }

        if (hasMacros(plan.lunch_p_g, plan.lunch_c_g,
plan.lunch_f_g)) {
            lastLunchMacros = new int[]{ plan.lunch_p_g,
plan.lunch_c_g, plan.lunch_f_g };
        } else if (plan.lunch_kcal > 0) {
            fetchMacrosFromAI("Lunch", plan.lunch, plan.lunch_kcal);
        }

        if (hasMacros(plan.dinner_p_g, plan.dinner_c_g,
plan.dinner_f_g)) {
            lastDinnerMacros = new int[]{ plan.dinner_p_g,
plan.dinner_c_g, plan.dinner_f_g };
        } else if (plan.dinner_kcal > 0) {
            fetchMacrosFromAI("Dinner", plan.dinner,
plan.dinner_kcal);
        }

        // --- 4) Update the footer summary ---
        updateSummary();
    }




    private String extractKeyword(String raw) {
        String[] parts = raw.split(",");
        return parts.length > 0 ? parts[0].trim() : "healthy food";
    }

    private void fetchImage(String keyword, ImageView view) {
        try {
            String key = (keyword == null ? "" :
keyword.trim().toLowerCase());
            if (key.isEmpty() ||
key.matches("(?i)^(about|approximately|around)$")) {
                key = "healthy meal";
            }
            final String keyFinal = key;

            // Build a more food-photography centric query
```

```
            String enriched = key + " food dish meal recipe on plate
close-up overhead";
            String q = java.net.URLEncoder.encode(enriched, "UTF-8");
            String url = "https://api.pexels.com/v1/search?query=" +
q +
                    "&per_page=12&orientation=landscape&size=medium";

            OkHttpClient client = new OkHttpClient();
            Request request = new Request.Builder()
                    .url(url)
                    .addHeader("Authorization",
"aGuhprX2IJ2guNswKqnQyhR08nW2MK5x0hbKmRCC58D44vFyCqvJp8ZU")
                    .build();

            client.newCall(request).enqueue(new Callback() {
                @Override public void onFailure(Call call,
IOException e) {
                    Log.e("PEXELS", "HTTP fail: " + e.getMessage());
                }
                @Override public void onResponse(Call call, Response
res) throws IOException {
                    if (!res.isSuccessful() || res.body() == null) {
                        Log.e("PEXELS", "HTTP code: " + res.code());
                        return;
                    }
                    try {
                        String json = res.body().string();
                        JSONObject obj = new JSONObject(json);
                        JSONArray photos =
obj.optJSONArray("photos");
                        if (photos == null || photos.length() == 0)
return;

                        int chosen = -1;
                        for (int i = 0; i < photos.length(); i++) {
                            JSONObject p = photos.getJSONObject(i);
                            String alt = p.optString("alt",
"").toLowerCase();

                            boolean looksLikeFood =
                                    alt.contains("food") ||
alt.contains("dish") || alt.contains("meal") ||
                                            alt.contains("plate") ||
alt.contains("bowl") || alt.contains("salad") ||
                                            alt.contains(keyFinal);

                            boolean mentionsPeople =

alt.matches(".*\\b(man|woman|people|person|friends|couple|hands)\\b.*
");

                            boolean notDrinkOnly =
                                    !(alt.contains("wine") ||
alt.contains("coffee") || alt.contains("tea") ||
                                            alt.contains("cocktail")
|| alt.contains("juice"));

                            if (looksLikeFood && !mentionsPeople &&
notDrinkOnly) {
                                chosen = i;
```

```
                                break;
                            }
                        }
                        if (chosen == -1) chosen = 0;

                        String imageUrl =
photos.getJSONObject(chosen).getJSONObject("src").getString("large");
                        runOnUiThread(() ->
Glide.with(PersonalizedplanActivity.this).load(imageUrl).into(view));
                    } catch (Exception e) {
                        Log.e("PEXELS", "Parse error: " +
e.getMessage());
                    }
                }
            });
        } catch (Exception ex) {
            Log.e("PEXELS", "Build URL error: " + ex.getMessage());
        }
    }


    /** Ask AI for macros for one meal and store them, then refresh
the summary. */
    private void fetchMacrosFromAI(String mealType, String
itemsDisplay, int kcal) {
        if (itemsDisplay == null || itemsDisplay.trim().isEmpty())
return;

        String prompt =
                "You are a nutritionist. Estimate macronutrients for
this single meal.\n" +
                        "Items: " + itemsDisplay + "\n" +
                        "Total energy: " + Math.max(200, kcal) + "
kcal\n" +
                        "Return ONLY JSON with INTEGER grams (no
prose, no markdown):\n" +
                        "{\"protein_g\": P, \"carbs_g\": C,
\"fat_g\": F}\n" +
                        "Constraints: 4*P + 4*C + 9*F must be within
15% of the total kcal. " +
                        "Do NOT copy any example values. Choose
realistic integers that satisfy the constraint.";

        CohereRequest req = new CohereRequest(prompt);
        new
CohereApiClient().createService().generate(req).enqueue(new
retrofit2.Callback<CohereResponse>() {
            @Override public void
onResponse(retrofit2.Call<CohereResponse> call,

retrofit2.Response<CohereResponse> res) {
                if (!res.isSuccessful() || res.body() == null) {
                    // fallback if API returned nothing useful
                    int[] m = computeMacrosLocal(mealType, kcal);
                    postStore(mealType, m);
                    return;
                }

                String raw = res.body().getText();
                try {
```

```
                    String json = findFirstJsonObject(raw);
                    if (json == null) {
                        int[] m = computeMacrosLocal(mealType, kcal);
                        postStore(mealType, m);
                        return;
                    }

                    org.json.JSONObject o = new
org.json.JSONObject(json);
                    // --- parse ---
                    int p = Math.max(0, o.optInt("protein_g", 0));
                    int c = Math.max(0, o.optInt("carbs_g",   0));
                    int f = Math.max(0, o.optInt("fat_g",     0));

                    // --- <<< PLACE THE VALIDATION HERE >>> ---
                    int energyCheck = 4*p + 4*c + 9*f;
                    if (p <= 0 || c <= 0 || f <= 0 ||
Math.abs(energyCheck - kcal) > 0.15 * kcal) {
                        int[] m = computeMacrosLocal(mealType, kcal);
                        p = m[0]; c = m[1]; f = m[2];
                    }
                    final int pFinal = p, cFinal = c, fFinal = f;
                    // ----------------------------------------

                    runOnUiThread(() -> {
                        //switch (mealType) {
                        //    case "Breakfast": lastBreakfastMacros =
new int[]{pFinal, cFinal, fFinal}; break;
                        //    case "Lunch":     lastLunchMacros     =
new int[]{pFinal, cFinal, fFinal}; break;
                         //   case "Dinner":    lastDinnerMacros    =
new int[]{pFinal, cFinal, fFinal}; break;
                        //}
                        updateSummary();
                    });
                } catch (Exception e) {
                    // parsing error -> fallback
                    int[] m = computeMacrosLocal(mealType, kcal);
                    postStore(mealType, m);
                }
            }

            @Override public void
onFailure(retrofit2.Call<CohereResponse> call, Throwable t) {
                // network/API failure -> fallback
                int[] m = computeMacrosLocal(mealType, kcal);
                postStore(mealType, m);
            }

            private void postStore(String mealType, int[] m) {
                runOnUiThread(() -> {
                    switch (mealType) {
                        case "Breakfast": lastBreakfastMacros = m;
break;
                        case "Lunch":     lastLunchMacros     = m;
break;
                        case "Dinner":    lastDinnerMacros    = m;
break;
                    }
                    updateSummary();
```

```
                });
            }
        });
    }


    private void savePlan(String mode, String planName) {
        int userId = prefs.getInt("user_id", -1);
        if (userId == -1) return;

        String breakfast = breakfastText.getText().toString();
        String lunch = lunchText.getText().toString();
        String dinner = dinnerText.getText().toString();
        String img1 = encodeImage(breakfastImg);
        String img2 = encodeImage(lunchImg);
        String img3 = encodeImage(dinnerImg);

        int bk = parseKcalLabel(breakfastCal.getText().toString());
        int ln = parseKcalLabel(lunchCal.getText().toString());
        int dn = parseKcalLabel(dinnerCal.getText().toString());

        okhttp3.FormBody body = new okhttp3.FormBody.Builder()
                .add("userid", String.valueOf(userId))
                .add("plan_name", planName)                         //
<-- NEW
                .add("breakfast", breakfast)
                .add("lunch", lunch)
                .add("dinner", dinner)
                .add("breakfast_image", img1)
                .add("lunch_image", img2)
                .add("dinner_image", img3)
                .add("breakfast_kcal", String.valueOf(bk))
                .add("lunch_kcal",     String.valueOf(ln))
                .add("dinner_kcal",    String.valueOf(dn))
                .add("set_as_today", mode.equals("today") ? "1" :
"0")
                .build();

        Request req = new Request.Builder()
                .url("http://10.0.2.2/nutrition/save_personal_plan.ph
p")
                .post(body).build();

        new OkHttpClient().newCall(req).enqueue(new Callback() {
            @Override public void onFailure(Call call, IOException e)
{
                runOnUiThread(() ->
Toast.makeText(PersonalizedplanActivity.this, "Save failed",
Toast.LENGTH_SHORT).show());
            }
            @Override public void onResponse(Call call, Response
response) {
                runOnUiThread(() ->
Toast.makeText(PersonalizedplanActivity.this, "Plan saved",
Toast.LENGTH_SHORT).show());
                // Optionally reload plans so the spinner shows the
new name immediately
                loadSavedPlans();
            }
        });
```

```
    }

    private String encodeImage(ImageView view) {
        try {
            Bitmap bmp = ((BitmapDrawable)
view.getDrawable()).getBitmap();
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            bmp.compress(Bitmap.CompressFormat.JPEG, 100, baos);
            return Base64.encodeToString(baos.toByteArray(),
Base64.DEFAULT);
        } catch (Exception e) {
            return "";
        }
    }

    public static class SavedPlan {
        public int id;
        public String name;
        public String breakfast, lunch, dinner, createdAt;
        public boolean needAi;
        public int breakfast_kcal, lunch_kcal, dinner_kcal;
        public int breakfast_p_g, breakfast_c_g, breakfast_f_g;
        public int lunch_p_g, lunch_c_g, lunch_f_g;
        public int dinner_p_g, dinner_c_g, dinner_f_g;

        @Override
        public String toString() {
            String date = (createdAt != null && createdAt.length() >=
10)
                    ? createdAt.substring(0, 10)
                    : "";
            String cleanName = (name != null) ? name.trim() : "";
            if (!cleanName.isEmpty()) {
                // show "Name - YYYY-MM-DD"
                return cleanName + (date.isEmpty() ? "" : " - " +
date);
            }
            // fallback: "Plan #id - YYYY-MM-DD"
            return "Plan #" + id + (date.isEmpty() ? "" : " - " +
date);
        }
    }


    private void cancelInFlight(String type) {
        try {
            if ("Breakfast".equals(type) && inflightBreakfastCall !=
null) inflightBreakfastCall.cancel();
            if ("Lunch".equals(type) && inflightLunchCall != null)
inflightLunchCall.cancel();
            if ("Dinner".equals(type) && inflightDinnerCall != null)
inflightDinnerCall.cancel();
        } catch (Exception ignored) {}
    }

    private String buildUserHintsForMeal(String type) {
        // Optional: bias the model per meal to reduce ambiguity and
speed up returns
        if ("Breakfast".equals(type)) return "- prefer
grains/eggs/yogurt/fruit; avoid soups and steaks";
```

```
        if ("Lunch".equals(type))     return "- prefer
rice/noodles/protein/veggies; avoid breakfast pastries";
        if ("Dinner".equals(type))    return "- prefer balanced
protein + veg + carbs; avoid desserts";
        return "";
    }

    private void cohereGenerateWithRetry(String type, String prompt,
int attempt) {
        CohereRequest req = new CohereRequest(prompt);

        retrofit2.Call<CohereResponse> call = new
CohereApiClient().createService().generate(req);
        if ("Breakfast".equals(type)) inflightBreakfastCall = call;
        if ("Lunch".equals(type))     inflightLunchCall = call;
        if ("Dinner".equals(type))    inflightDinnerCall = call;

        call.enqueue(new retrofit2.Callback<CohereResponse>() {
            @Override public void
onResponse(retrofit2.Call<CohereResponse> c,

retrofit2.Response<CohereResponse> res) {
                if (!res.isSuccessful() || res.body() == null) {
                    retryOrFail(type, prompt, attempt);
                    return;
                }
                String text = res.body().getText() == null ? "" :
res.body().getText().trim();
                String normalized = normalizeCohereLine(type, text);
                if (normalized != null) {
                    String oneLine = sanitizeMealLine(type,
normalized);
                    runOnUiThread(() -> updateMealFromText(type,
oneLine));
                    return;
                }
                cohereSingleLineItems(type);
            }

            @Override public void
onFailure(retrofit2.Call<CohereResponse> c, Throwable t) {
                if (c.isCanceled()) return; // canceled due to a new
request
                retryOrFail(type, prompt, attempt);
            }
        });
    }

    private void retryOrFail(String type, String prompt, int attempt)
{
        if (attempt >= COHERE_MAX_RETRIES) {
            runOnUiThread(() -> Toast.makeText(this, "AI timeout,
used fallback for " + type, Toast.LENGTH_SHORT).show());
            // Fallback quick pattern: safer than showing nothing
            String fallbackItems = ("Breakfast".equals(type)) ?
"oatmeal, banana, yogurt"
                    : ("Lunch".equals(type)) ? "chicken breast, rice,
salad"
                    : "salmon, brown rice, veggies";
            String safe = type + ": ~" +
```

```
(("Breakfast".equals(type)) ? 420 : ("Lunch".equals(type)) ? 550 :
520)
                    + " kcal - " + fallbackItems;
            runOnUiThread(() -> updateMealFromText(type, safe));
            return;
        }
        getWindow().getDecorView().postDelayed(
                () -> cohereGenerateWithRetry(type, prompt, attempt +
1),
                COHERE_RETRY_BACKOFF_MS * attempt
        );
    }

    /** If strict-ish normalization failed, get items only, then
compute kcal locally. */
    private void cohereSingleLineItems(String mealType) {
        String allergies = prefs.getString("allergies", "").trim();
        String prompt =
                "Output exactly ONE LINE for " + mealType + " with
only items, no calories.\n" +
                        "Format: " + mealType + ": - item1, item2,
item3\n" +
                        "Rules: items 2-5, lowercase, 1-3 words each,
real foods only, comma-separated, " +
                        "no quantities, no brands. No extra text, no
markdown." +
                        (allergies.isEmpty() ? "" : "\nExclude
allergens: " + allergies);

        CohereRequest req2 = new CohereRequest(prompt);
        new
CohereApiClient().createService().generate(req2).enqueue(new
retrofit2.Callback<CohereResponse>() {
            @Override public void
onResponse(retrofit2.Call<CohereResponse> c2,

retrofit2.Response<CohereResponse> r2) {
                if (!r2.isSuccessful() || r2.body() == null ||
r2.body().getText() == null) return;
                String line = coerceToOneLine(mealType,
r2.body().getText().trim());
                // Pull just the items part after the first '-' or
':' and estimate kcal
                String items = line;
                int dash = line.indexOf(" - ");
                if (dash >= 0 && dash + 3 < line.length()) items =
line.substring(dash + 3);
                else {
                    int colon = line.indexOf(':');
                    if (colon >= 0 && colon + 1 < line.length())
items = line.substring(colon + 1);
                }
                items = items.replaceAll("^\\s*-\\s*", "").trim();
                // Build the canonical string ourselves
                int kcal = estimateKcalFromItems(items);
                final String canonical = mealType + ": ~" + kcal + "
kcal - " + items;
                runOnUiThread(() -> updateMealFromText(mealType,
canonical));
            }
```

```
            @Override public void
onFailure(retrofit2.Call<CohereResponse> c2, Throwable t2) {
                // Silent: keep UI as-is if even this fails (rare).
You can toast if you prefer.
                Log.e("AI_SIMPLE", "Fallback items call failed: " +
t2.getMessage());
            }
        });
    }

    private String normalizeCohereLine(String meal, String raw) {
        if (raw == null) return null;

        String s = coerceToOneLine(meal, raw).trim();

        // force meal label at start
        if (!s.toLowerCase().startsWith(meal.toLowerCase() + ":")) {
            int idx = s.toLowerCase().indexOf(meal.toLowerCase());
            if (idx >= 0) s = s.substring(idx);
            if (!s.toLowerCase().startsWith(meal.toLowerCase() +
":")) {
                s = meal + ": " + s;
            }
        }

        // extract kcal number in many formats
        Matcher km =
Pattern.compile("(?i)(~?\\s*)(\\d{2,4})\\s*(k?cal(?:ories)?)?").match
er(s);
        int kcal = -1;
        if (km.find()) {
            try { kcal = Integer.parseInt(km.group(2)); } catch
(Exception ignored) {}
        }
        if (kcal < 200 || kcal > 900) {
            if (kcal > 0) kcal = Math.max(200, Math.min(kcal, 900));
        }

        // items part
        String itemsPart = s;
        int dash = s.indexOf(" - ");
        if (dash >= 0) itemsPart = s.substring(dash + 3);
        else {
            int colon = s.indexOf(':');
            if (colon >= 0 && colon + 1 < s.length()) itemsPart =
s.substring(colon + 1);
        }

        // clean items
        String[] rawItems = itemsPart.split(",");
        ArrayList<String> items = new ArrayList<>();
        for (String it : rawItems) {
            String t = it.trim().toLowerCase().replaceAll("[^a-z0-9
&/()-]", "");
            t = t.replaceAll("\\s+", " ");
            if (t.isEmpty()) continue;
            if
(t.matches("(?i)^(and|with|on|in|of|the|a|cal|kcal|calories)$"))
continue;
            items.add(t);
```

```
            if (items.size() == 5) break;
        }

        if (items.isEmpty()) {
            String fallbackItems = extractItemsDisplay(s);
            if (fallbackItems != null
&& !fallbackItems.trim().isEmpty()) {
                return meal + ": ~" + (kcal > 0 ? kcal :
estimateKcalFromItems(fallbackItems)) +
                        " kcal - " + fallbackItems;
            }
            return null;
        }

        String itemsJoined = String.join(", ", items);

        if (kcal <= 0) {
            kcal = estimateKcalFromItems(itemsJoined);
        }

        return meal + ": ~" + kcal + " kcal - " + itemsJoined;
    }

    private void showMacrosPopup() {
        // Inflate the custom layout
        View dialogView =
getLayoutInflater().inflate(R.layout.dialog_macros, null);

        TextView proteinDetail =
dialogView.findViewById(R.id.protein_detail);
        TextView carbsDetail   =
dialogView.findViewById(R.id.carbs_detail);
        TextView fatDetail     =
dialogView.findViewById(R.id.fat_detail);
        Button closeBtn        =
dialogView.findViewById(R.id.close_btn);

        // Fill in values from your totals
        int totalProtein = lastBreakfastMacros[0] +
lastLunchMacros[0] + lastDinnerMacros[0];
        int totalCarbs   = lastBreakfastMacros[1] +
lastLunchMacros[1] + lastDinnerMacros[1];
        int totalFat     = lastBreakfastMacros[2] +
lastLunchMacros[2] + lastDinnerMacros[2];

        proteinDetail.setText("Protein: ~" + totalProtein + " g");
        carbsDetail.setText("Carbs: ~" + totalCarbs + " g");
        fatDetail.setText("Fat: ~" + totalFat + " g");

        // Build the dialog
        android.app.AlertDialog dialog = new
android.app.AlertDialog.Builder(this)
                .setView(dialogView)
                .create();

        // Close button action
        closeBtn.setOnClickListener(v -> dialog.dismiss());

        dialog.show();
    }
```

```
    private void showMealPopup(String mealType) {
        // Inflate the custom dialog layout
        View dialogView =
getLayoutInflater().inflate(R.layout.dialog_meal_details, null);

        TextView titleTv  = dialogView.findViewById(R.id.meal_title);
        TextView itemsTv  = dialogView.findViewById(R.id.meal_items);
        TextView kcalTv   = dialogView.findViewById(R.id.meal_kcal);
        TextView protTv   =
dialogView.findViewById(R.id.meal_protein);
        TextView carbsTv  = dialogView.findViewById(R.id.meal_carbs);
        TextView fatTv    = dialogView.findViewById(R.id.meal_fat);
        Button   closeBtn = dialogView.findViewById(R.id.close_btn);

        String items;
        int kcal;
        int[] macros;

        switch (mealType) {
            case "Breakfast":
                items  = breakfastText.getText() != null ?
breakfastText.getText().toString() : "--";
                kcal   = parseKcalLabel(breakfastCal.getText() !=
null ? breakfastCal.getText().toString() : "0 kcal");
                macros = lastBreakfastMacros;
                break;
            case "Lunch":
                items  = lunchText.getText() != null ?
lunchText.getText().toString() : "--";
                kcal   = parseKcalLabel(lunchCal.getText() != null ?
lunchCal.getText().toString() : "0 kcal");
                macros = lastLunchMacros;
                break;
            default: // "Dinner"
                items  = dinnerText.getText() != null ?
dinnerText.getText().toString() : "--";
                kcal   = parseKcalLabel(dinnerCal.getText() != null ?
dinnerCal.getText().toString() : "0 kcal");
                macros = lastDinnerMacros;
                break;
        }

        titleTv.setText(mealType);
        itemsTv.setText(items);
        kcalTv.setText("~" + Math.max(0, kcal) + " kcal");

        // If macros are still zero (AI not returned yet), show
placeholders politely
        String proteinStr = (macros[0] > 0) ? (macros[0] + " g") : "-
- g";
        String carbsStr   = (macros[1] > 0) ? (macros[1] + " g") : "-
- g";
        String fatStr     = (macros[2] > 0) ? (macros[2] + " g") : "-
- g";

        protTv.setText("Protein: " + proteinStr);
        carbsTv.setText("Carbs: " + carbsStr);
        fatTv.setText("Fat: " + fatStr);
```

```
        android.app.AlertDialog dialog = new
android.app.AlertDialog.Builder(this)
                .setView(dialogView)
                .create();

        closeBtn.setOnClickListener(v -> dialog.dismiss());
        dialog.show();
    }

    private int[] computeMacrosLocal(String mealType, int kcal) {
        double pRatio, cRatio, fRatio;
        switch (mealType) {
            case "Breakfast": pRatio = 0.25; cRatio = 0.55; fRatio =
0.20; break;
            case "Lunch":     pRatio = 0.30; cRatio = 0.50; fRatio =
0.20; break;
            default:          pRatio = 0.35; cRatio = 0.45; fRatio =
0.20; break; // Dinner
        }
        int p = Math.max(0, (int)Math.round((kcal * pRatio) / 4.0));
        int c = Math.max(0, (int)Math.round((kcal * cRatio) / 4.0));
        int f = Math.max(0, (int)Math.round((kcal * fRatio) / 9.0));

        int total = 4*p + 4*c + 9*f;
        if (total > 0) {
            double scale = (double)kcal / total;
            p = Math.max(0, (int)Math.round(p * scale));
            c = Math.max(0, (int)Math.round(c * scale));
            f = Math.max(0, (int)Math.round(f * scale));
        }
        return new int[]{p, c, f};
    }

    private void promptPlanNameAndSave(String mode) {
        final EditText input = new EditText(this);
        input.setHint("e.g., High-Protein Plan");
        input.setInputType(InputType.TYPE_CLASS_TEXT |
InputType.TYPE_TEXT_FLAG_CAP_WORDS);

        int pad = (int) (16 *
getResources().getDisplayMetrics().density);
        LinearLayout container = new LinearLayout(this);
        container.setPadding(pad, pad, pad, 0);
        container.addView(input);

        String title = mode.equals("today") ? "Save as Today's
Plan" : "Save Plan";

        new android.app.AlertDialog.Builder(this)
                .setTitle(title)
                .setMessage("Give this meal plan a name")
                .setView(container)
                .setNegativeButton("Cancel", null)
                .setPositiveButton("Save", (d, w) -> {
                    String name = input.getText().toString().trim();
                    if (name.isEmpty()) {
                        name = "Plan " + new
java.text.SimpleDateFormat("yyyy-MM-dd HH:mm",

java.util.Locale.getDefault()).format(new java.util.Date());
```

```
                    }
                    savePlan(mode, name);
                })
                .show();
    }

    // Stop AI from touching the UI while we are showing a saved plan
    private boolean suppressAI = false;

    // Helper: fetch the currently selected SavedPlan (if any)
    private SavedPlan getCurrentSelectedPlan() {
        if (lastSelectedPlanId == null) return null;
        for (SavedPlan p : savedPlans) if (p.id ==
lastSelectedPlanId) return p;
        return null;
    }

    private boolean hasMacros(int p, int c, int f) {
        return p > 0 || c > 0 || f > 0;
    }

}
```

PlanListActivity.java

```
package com.example.bmi;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;

import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.HashMap;
import java.util.Map;

public class PlanListActivity extends AppCompatActivity {

    private LinearLayout questionContainer;
    int planId;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_plan_list);
        SharedPreferences sharedPreferences =
getSharedPreferences("user_prefs", MODE_PRIVATE);
        int size=sharedPreferences.getInt("text_size",12);
        changeTextSize(size);
        // Get the container from the main layout
        questionContainer = findViewById(R.id.planContainer);

        // Fetch plans from the server
        fetchPlansFromServer();

        // Find the TextView
        ImageButton backButton = findViewById(R.id.planbackButton);

        // Set click listener
        backButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Navigate to RegisterActivity
                Intent intent = new Intent(PlanListActivity.this,
HomeActivity.class);
                startActivity(intent);
            }
        });
    }
    private void changeTextSize(float size) {

        TextView[] textViews = new TextView[] {
                findViewById(R.id.textView),
                findViewById(R.id.textView2),
                findViewById(R.id.textView6),
                findViewById(R.id.protein_text),
                findViewById(R.id.carbonhydrate_text),
                findViewById(R.id.Fat_text),
                findViewById(R.id.textView),
                findViewById(R.id.textView5),
                findViewById(R.id.textView7),
                findViewById(R.id.textView8),
                findViewById(R.id.textView3),
                findViewById(R.id.textView9),
                findViewById(R.id.textView10),
                findViewById(R.id.textView11),
                findViewById(R.id.textView12),
                findViewById(R.id.textView13),
                findViewById(R.id.textView14),
                findViewById(R.id.questionText),
                findViewById(R.id.totalquestionText),
                findViewById(R.id.planNameText),
                findViewById(R.id.sdgshfh),
                findViewById(R.id.asdfsaf),
                // Add more TextViews here as needed
        };
```

```
        Button[] buttons = new Button[] {
                //findViewById(R.id.Link_Suggest),
                //findViewById(R.id.Link_Personalized),
                //findViewById(R.id.Link_Shuffle),
                //findViewById(R.id.resultBMI),
                findViewById(R.id.GenerateButton),
                findViewById(R.id.SaveButton),
                findViewById(R.id.btnProfile),
                findViewById(R.id.btnPlanList),
                findViewById(R.id.textButton),
                findViewById(R.id.btnLogout),
                findViewById(R.id.nextButton),
                findViewById(R.id.viewButton),
                findViewById(R.id.deleteButton),
                findViewById(R.id.GenerateButton1),
                findViewById(R.id.backbutton1),
                // Add more Buttons here as needed
        };


        for (TextView textView : textViews) {
            if (textView != null) {
                textView.setTextSize(size);
            }
        }


        for (Button button : buttons) {
            if (button != null) {
                button.setTextSize(size);
            }
        }
    }
    private void fetchPlansFromServer() {
        // Example GET URL - add the `userid` to the URL as a query
parameter
        SharedPreferences sharedPreferences =
getSharedPreferences("user_prefs", MODE_PRIVATE);
        int userId = sharedPreferences.getInt("user_id", 1); // Get
user ID from shared preferences
        String url =
"http://10.0.2.2/nutrition/login.php?action=get_plans&userid=" +
userId; // Append the userId as a query parameter

        StringRequest stringRequest = new
StringRequest(Request.Method.POST, url,
                response -> {
                    try {
                        // Handle the response as you already did
                        JSONArray jsonArray = new
JSONArray(response);
                        for (int i = 0; i < jsonArray.length(); i++)
{
                            JSONObject obj =
jsonArray.getJSONObject(i);
                            int planId = obj.getInt("plan_id");
                            String planName =
obj.getString("plan_name");
                            // Inflate a new row using the XML
template and populate it
```

```
                            Log.e("123555", String.valueOf(planId));
                            inflatePlanRow(planId, planName);
                        }
                    } catch (JSONException e) {
                        e.printStackTrace();
                        Toast.makeText(PlanListActivity.this,
                                "Parsing error",
Toast.LENGTH_SHORT).show();
                    }
                },
                error -> Toast.makeText(PlanListActivity.this,
                        "Volley error: " + error.getMessage(),
Toast.LENGTH_SHORT).show()
        ) {
            @Override
            protected Map<String, String> getParams() {
                Map<String, String> params = new HashMap<>();
                SharedPreferences sharedPreferences =
getSharedPreferences("user_prefs", MODE_PRIVATE);
                int userId = sharedPreferences.getInt("user_id", 1);
                params.put("userid", String.valueOf(userId)); // Add
userid to the params
                return params;
            }
        };
        RequestQueue queue = Volley.newRequestQueue(this);
        queue.add(stringRequest);
    }


    /**
     * Inflates a row layout from XML and populates it with the plan
data.
     * The inflated layout already contains the View and Delete
buttons.
     *
     * @param planId   the ID of the plan.
     * @param planName the name of the plan.
     */
    private void inflatePlanRow(int planId, String planName) {
        // Inflate the row layout from XML file (row_plan_list.xml)
        LayoutInflater inflater = LayoutInflater.from(this);
        View rowView = inflater.inflate(R.layout.row_plan_list,
questionContainer, false);

        // Get references to the views within the inflated layout.
        TextView planNameText =
rowView.findViewById(R.id.planNameText);
        Button viewButton = rowView.findViewById(R.id.viewButton);
        Button deleteButton =
rowView.findViewById(R.id.deleteButton);

        // Set the plan name in the TextView.
        planNameText.setText(planName);

        // Set up a click listener for the View button.
        viewButton.setOnClickListener(view -> {

            Intent intent = new Intent(PlanListActivity.this,
ViewEditPlanActivity.class);
```

```
            intent.putExtra("plan_id", planId); // Use the
appropriate key and variable
            startActivity(intent);
            // You can add further logic to launch another activity,
etc.
        });

        // Store planId in the Delete button's tag so that you can
retrieve it on click.
        deleteButton.setTag(planId);
        deleteButton.setOnClickListener(view -> {
            int currentPlanId = (int) view.getTag();
            // Send a request to delete the plan and remove this row
from the container.
            deletePlanFromDB(currentPlanId, rowView);
        });

        // Add the fully populated rowView to the container.
        questionContainer.addView(rowView);
    }

    /**
     * Sends a POST request to delete the plan from the database and
removes the row from the UI on success.
     *
     * @param planId  the ID of the plan to be deleted.
     * @param rowView the row view to remove after deletion.
     */
    private void deletePlanFromDB(int planId, View rowView) {
        String url =
"http://10.0.2.2/nutrition/login.php?action=delete_plan";

        // Retrieve userId from SharedPreferences if needed.
        SharedPreferences prefs = getSharedPreferences("user_prefs",
MODE_PRIVATE);
         int userId = prefs.getInt("user_id", -1);

        StringRequest request = new
StringRequest(Request.Method.POST, url,
                response -> {
                    // On success, remove the row from the container.
                    questionContainer.removeView(rowView);
                    Log.e("123123213321",response+"WAHA:"+planId);
                },
                error -> Toast.makeText(PlanListActivity.this,
                        "Failed to delete plan: " +
error.getMessage(), Toast.LENGTH_SHORT).show()
        ) {
            @Override
            protected Map<String, String> getParams() {
                Map<String, String> params = new HashMap<>();
                params.put("plan_id", String.valueOf(planId)); // Add
userid to the params
                return params;
            }
        };

        RequestQueue queue = Volley.newRequestQueue(this);
        queue.add(request);
```

```
    }
}
```

ProfileActivity.java

```
package com.example.bmi;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;

import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;

import org.json.JSONException;
import org.json.JSONObject;

public class ProfileActivity extends AppCompatActivity {

    private TextView usernameText;
    private TextView gmailText;
    private TextView heightText;
    private TextView weightText;
    private TextView currentBMIText;
    private TextView currentBMIStatusText;
    private EditText targetBMIInput;
    private Button updateButton;
    private ImageButton backButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_profile);

        int userId = getSharedPreferences("user_prefs",
MODE_PRIVATE).getInt("user_id", -1);

        // Initialize UI elements
        usernameText = findViewById(R.id.nameProfile);
        gmailText = findViewById(R.id.emailProfile);
        heightText = findViewById(R.id.heightProfile);
        weightText = findViewById(R.id.weightProfile);
```

```
        currentBMIText = findViewById(R.id.currentBMIText);
        currentBMIStatusText = findViewById(R.id.currentBMIStatus);
        targetBMIInput = findViewById(R.id.targetBMIInput);  // now
it's an EditText
        backButton = findViewById(R.id.profilebackButton);
        updateButton = findViewById(R.id.Updatebutton);

        fetchUserInfo(ProfileActivity.this, userId);

        backButton.setOnClickListener(v -> {
            Intent intent = new Intent(ProfileActivity.this,
HomeActivity.class);
            startActivity(intent);
        });

        updateButton.setOnClickListener(v -> {
            updateUserProfile(userId);
        });
    }

    private void fetchUserInfo(Context context, int userId) {
        String url =
"http://10.0.2.2/nutrition/login.php?action=get_user_info&userid=" +
userId;

        StringRequest stringRequest = new
StringRequest(Request.Method.GET, url,
                response -> {
                    try {
                        JSONObject json = new JSONObject(response);
                        if
(json.getString("status").equals("success")) {
                            JSONObject user =
json.getJSONObject("user");

                            String username =
user.getString("username");
                            String gmail = user.getString("gmail");
                            int genderInt = user.optInt("gender", -
1);
                            String gender = genderInt == 0 ? "Male" :
genderInt == 1 ? "Female" : "Unknown";
                            int age = user.optInt("age", -1);
                            int height = user.optInt("height", 100);
                            float weight = (float)
user.optDouble("weight", 40.0);
                            float target_BMI = (float)
user.optDouble("target_BMI", 20.0f);

                            float currentBMI = weight / ((height /
100f) * (height / 100f));
                            String bmiStatus =
getBMIStatus(currentBMI);

                            // Gender radio buttons
                            RadioGroup genderGroup =
findViewById(R.id.genderRadioGroup);
                            RadioButton radioMale =
findViewById(R.id.radioMale);
                            RadioButton radioFemale =
```

```
findViewById(R.id.radioFemale);

                            if (gender.equals("Male")) {
                                radioMale.setChecked(true);
                            } else if (gender.equals("Female")) {
                                radioFemale.setChecked(true);
                            }

                            // Set UI texts
                            usernameText.setText(username);
                            gmailText.setText(gmail);

heightText.setText(String.valueOf(height));
                            weightText.setText(String.format("%.1f",
weight));

currentBMIText.setText(String.format("%.1f", currentBMI));
                            currentBMIStatusText.setText(bmiStatus);

targetBMIInput.setText(String.format("%.1f", target_BMI));

                        } else {
                            Log.e("UserInfo", "User not found or
error.");
                        }
                    } catch (JSONException e) {
                        e.printStackTrace();
                        Log.e("UserInfo", "JSON parsing error.");
                    }
                },
                error -> Log.e("UserInfo", "Volley error: " +
error.getMessage())
        );

        RequestQueue queue = Volley.newRequestQueue(context);
        queue.add(stringRequest);
    }

    private void updateUserProfile(int userId) {
        String url =
"http://10.0.2.2/nutrition/login.php?action=update_user";

        RadioGroup genderGroup = findViewById(R.id.genderRadioGroup);
        RadioButton selectedGender =
findViewById(genderGroup.getCheckedRadioButtonId());
        String genderStr =
selectedGender.getText().toString().equals("Male") ? "0" : "1";

        String heightStr = heightText.getText().toString().trim();
        String weightStr = weightText.getText().toString().trim();
        String usernameStr =
usernameText.getText().toString().trim();
        String targetBMIStr =
targetBMIInput.getText().toString().trim();

        if (usernameStr.isEmpty() || weightStr.isEmpty() ||
heightStr.isEmpty() || targetBMIStr.isEmpty()) {
            Toast.makeText(this, "Please fill in all fields.",
Toast.LENGTH_SHORT).show();
            return;
```

```
        }

        if (usernameStr.length() < 4 || usernameStr.length() > 10) {
            Toast.makeText(this, "Username must be 4 to 10
characters.", Toast.LENGTH_SHORT).show();
            return;
        }

        float weight, targetBMI;
        int height;

        try {
            weight = Float.parseFloat(weightStr);
            targetBMI = Float.parseFloat(targetBMIStr);
        } catch (NumberFormatException e) {
            Toast.makeText(this, "Weight and Target BMI must be valid
numbers.", Toast.LENGTH_SHORT).show();
            return;
        }

        try {
            height = Integer.parseInt(heightStr);
        } catch (NumberFormatException e) {
            Toast.makeText(this, "Height must be a valid number.",
Toast.LENGTH_SHORT).show();
            return;
        }

        if (weight < 20 || weight > 100 || height < 140 || height >
200 || targetBMI < 10 || targetBMI > 40) {
            Toast.makeText(this, "Please enter valid ranges.",
Toast.LENGTH_SHORT).show();
            return;
        }

        StringRequest stringRequest = new
StringRequest(Request.Method.POST, url,
                response -> {
                    Toast.makeText(this, "Profile updated
successfully!", Toast.LENGTH_SHORT).show();

                    // ☑ Jump to another activity after successful
update
                    Intent intent = new Intent(ProfileActivity.this,
HomeActivity.class); // change to your target Activity
                    startActivity(intent);
                    finish(); // optional: close current activity
                },
                error -> Toast.makeText(this, "Update failed. Please
try again.", Toast.LENGTH_SHORT).show()
        ) {
            @Override
            protected java.util.Map<String, String> getParams() {
                java.util.Map<String, String> params = new
java.util.HashMap<>();
                params.put("userid", String.valueOf(userId));
                params.put("username", usernameStr);
                params.put("gender", genderStr);
                params.put("height", String.valueOf(height));
                params.put("weight", String.valueOf(weight));
```

```
                params.put("target_BMI", String.valueOf(targetBMI));
                return params;
            }
        };

        Volley.newRequestQueue(this).add(stringRequest);
    }

    private String getBMIStatus(float bmi) {
        if (bmi < 18.5f) return "Underweight";
        else if (bmi < 24.9f) return "Normal";
        else if (bmi < 29.9f) return "Overweight";
        else return "Obese";
    }
}
```

ProfileEditActivity.java

```
package com.example.bmi;

import android.content.SharedPreferences;
import android.os.Bundle;
import android.text.Editable;
import android.text.InputFilter;
import android.text.TextWatcher;
import android.text.Spanned;
import android.util.Log;
import android.view.View;
import android.widget.*;

import androidx.appcompat.app.AppCompatActivity;

import org.json.JSONObject;

import java.io.IOException;
import java.util.Locale;

import okhttp3.FormBody;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;
import okhttp3.Call;
import okhttp3.Callback;

public class ProfileEditActivity extends AppCompatActivity {

    private EditText etName, etHeight, etWeight, etTargetBmi;
    private TextView tvEmail, tvCurrentBmi;
    private Spinner spGender;
    private ProgressBar progress;

    private int userId;
    private final OkHttpClient client = new OkHttpClient();

    private static final String PROFILE_URL =
"http://10.0.2.2/nutrition/settings.php";
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_profile_edit);

        ImageButton btnBack = findViewById(R.id.btnBack);
        btnBack.setOnClickListener(v -> finish());

        etName       = findViewById(R.id.etName);
        tvEmail      = findViewById(R.id.tvEmail);
        spGender     = findViewById(R.id.spGender);
        etHeight     = findViewById(R.id.etHeight);
        etWeight     = findViewById(R.id.etWeight);
        tvCurrentBmi = findViewById(R.id.tvCurrentBmi);
        etTargetBmi  = findViewById(R.id.etTargetBmi);
        progress     = findViewById(R.id.progress);
        Button btnSave = findViewById(R.id.btnSave);

        // Spinner gender options
        ArrayAdapter<CharSequence> adapter =
ArrayAdapter.createFromResource(
                this, R.array.gender_options,
android.R.layout.simple_spinner_item);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropd
own_item);
        spGender.setAdapter(adapter);

        // Limit BMI input decimals
        etTargetBmi.setFilters(new InputFilter[]{new
DecimalDigitsInputFilter(2)});

        // ---- Prefill from SharedPreferences safely ----
        SharedPreferences prefs = getSharedPreferences("user_prefs",
MODE_PRIVATE);
        userId = prefs.getInt("user_id", -1);
        Log.d("ProfileEdit", "userId=" + userId);

        String namePref   = getPrefAsString(prefs, "username",
                getPrefAsString(prefs, "name", ""));
        String emailPref  = getPrefAsString(prefs, "gmail",
                getPrefAsString(prefs, "email", "-"));
        String heightPref = getPrefAsString(prefs, "height",
                getPrefAsString(prefs, "height_cm", ""));
        String weightPref = getPrefAsString(prefs, "weight",
                getPrefAsString(prefs, "weight_kg", ""));
        String targetPref = getPrefAsString(prefs, "target_BMI",
                getPrefAsString(prefs, "target_bmi", ""));
        String currentBmiPref = getPrefAsString(prefs, "current_bmi",
"—");

        // gender may be stored as int (0/1) or text
        String genderTextPref = "Male";
        if (prefs.contains("gender")) {
            Object gVal = prefs.getAll().get("gender");
            if (gVal instanceof Integer) {
                genderTextPref = ((Integer) gVal == 1) ? "Female" :
"Male";
            } else if (gVal instanceof String) {
                genderTextPref = (String) gVal;
```

```
            }
        } else {
            genderTextPref = getPrefAsString(prefs, "gender_text",
"Male");
        }

        etName.setText(namePref);
        tvEmail.setText(emailPref);

spGender.setSelection("Female".equalsIgnoreCase(genderTextPref) ? 1 :
0);
        etHeight.setText(heightPref);
        etWeight.setText(weightPref);
        etTargetBmi.setText(targetPref);
        tvCurrentBmi.setText(currentBmiPref); // show last known BMI
from prefs (if any)



        fetchProfile();               // load from server (fills
email & current_bmi)
        btnSave.setOnClickListener(v -> saveProfile());
    }

    // Optional watcher if you ever want live BMI again
    private final TextWatcher simpleWatcher = new TextWatcher() {
        @Override public void beforeTextChanged(CharSequence s, int
start, int count, int after) {}
        @Override public void onTextChanged(CharSequence s, int
start, int before, int count) { calcAndShowBmi(); }
        @Override public void afterTextChanged(Editable s) {}
    };

    /** Safely read any value from SharedPreferences as String
(handles int/float/String) */
    private String getPrefAsString(SharedPreferences prefs, String
key, String fallback) {
        if (!prefs.contains(key)) return fallback;
        try {
            Object v = prefs.getAll().get(key);
            if (v == null) return fallback;
            return String.valueOf(v);
        } catch (ClassCastException e) {
            return fallback;
        }
    }

    private void calcAndShowBmi() {
        try {
            double h =
Double.parseDouble(etHeight.getText().toString().trim());
            double w =
Double.parseDouble(etWeight.getText().toString().trim());
            if (h > 0) {
                double m = h / 100.0;
                double bmi = w / (m * m);

tvCurrentBmi.setText(String.format(Locale.getDefault(), "%.1f",
bmi));
                return;
```

```
            }
        } catch (Exception ignored) {}
        tvCurrentBmi.setText("—");
    }

    private void setLoading(boolean show) {
        progress.setVisibility(show ? View.VISIBLE : View.GONE);
    }

    private void fetchProfile() {
        if (userId <= 0) {
            Toast.makeText(this, "No userId in SharedPreferences",
Toast.LENGTH_LONG).show();
            Log.e("ProfileEdit", "Missing userId. Did you save it at
login?");
            return;
        }
        setLoading(true);

        RequestBody body = new FormBody.Builder()
                .add("action", "get_profile")
                .add("userid", String.valueOf(userId))
                .build();

        Request req = new
Request.Builder().url(PROFILE_URL).post(body).build();
        client.newCall(req).enqueue(new Callback() {
            @Override public void onFailure(Call call, IOException e)
{
                Log.e("ProfileEdit", "Network error", e);
                runOnUiThread(() -> {
                    setLoading(false);
                    Toast.makeText(ProfileEditActivity.this, "Network
error: " + e.getMessage(), Toast.LENGTH_LONG).show();
                });
            }

            @Override public void onResponse(Call call, Response
response) throws IOException {
                final String json = response.body() != null ?
response.body().string() : "";
                Log.d("ProfileEdit", "Server JSON: " + json);

                runOnUiThread(() -> {
                    try {
                        JSONObject o = new JSONObject(json);
                        String status = o.optString("status", "");
                        if ("success".equalsIgnoreCase(status)) {
                            String name   = o.optString("name", "");
                            String email  = o.optString("email", "-
");

                            String gender = o.optString("gender",
"Male");

                            String height = o.optString("height_cm",
"");

                            String weight = o.optString("weight_kg",
"");
                            if (weight.isEmpty()) {
                                weight =
etWeight.getText().toString();
```

```
                            }
                            String target = o.optString("target_bmi",
"");
                            String current =
o.optString("current_bmi", "—");

                            etName.setText(name);
                            tvEmail.setText(email);

spGender.setSelection("Female".equalsIgnoreCase(gender) ? 1 : 0);
                            etHeight.setText(height);
                            etWeight.setText(weight);
                            etTargetBmi.setText(target);
                            tvCurrentBmi.setText(current); // ← use
server's latest BMI

                            // Cache to prefs for quick load next
time
                            SharedPreferences.Editor ed =
getSharedPreferences("user_prefs", MODE_PRIVATE).edit();
                            ed.putString("username", name);
                            ed.putString("name", name);
                            ed.putString("gmail", email);
                            ed.putString("email", email);
                            ed.putInt("gender",
"Female".equalsIgnoreCase(gender) ? 1 : 0);
                            ed.putString("gender_text", gender);
                            ed.putString("height", height);
                            ed.putString("height_cm", height);
                            ed.putString("weight", weight);
                            ed.putString("weight_kg", weight);
                            ed.putString("target_BMI", target);
                            ed.putString("target_bmi", target);
                            ed.putString("current_bmi", current);
                            ed.apply();

                        } else {
                            String msg = o.optString("message",
"Server error");
                            Toast.makeText(ProfileEditActivity.this,
"Server: " + status + " - " + msg, Toast.LENGTH_LONG).show();
                        }
                    } catch (Exception ex) {
                        Log.e("ProfileEdit", "Parse error", ex);
                        Toast.makeText(ProfileEditActivity.this, "Bad
JSON from server", Toast.LENGTH_LONG).show();
                    } finally {
                        setLoading(false);
                    }
                });
            }
        });
    }

    private void saveProfile() {
        String name = etName.getText().toString().trim();
        String heightStr = etHeight.getText().toString().trim();
        String weightStr = etWeight.getText().toString().trim();
        String targetStr = etTargetBmi.getText().toString().trim();
        String gender = spGender.getSelectedItemPosition() == 1 ?
```

```
"Female" : "Male";

        if (name.isEmpty() || heightStr.isEmpty() ||
weightStr.isEmpty()) {
            Toast.makeText(this, "Please fill name, height and
weight.", Toast.LENGTH_SHORT).show();
            return;
        }

        setLoading(true);

        RequestBody body = new FormBody.Builder()
                .add("action", "update_profile")
                .add("userid", String.valueOf(userId))
                .add("name", name)
                .add("gender", gender)
                .add("height_cm", heightStr)
                .add("weight_kg", weightStr)
                .add("target_bmi", targetStr)
                .build();

        Request req = new
Request.Builder().url(PROFILE_URL).post(body).build();
        client.newCall(req).enqueue(new Callback() {
            @Override public void onFailure(Call call, IOException e)
{
                Log.e("ProfileEdit", "Save failed", e);
                runOnUiThread(() -> {
                    setLoading(false);
                    Toast.makeText(ProfileEditActivity.this, "Save
failed: " + e.getMessage(), Toast.LENGTH_SHORT).show();
                });
            }

            @Override public void onResponse(Call call, Response
response) throws IOException {
                String json = response.body() != null ?
response.body().string() : "";
                Log.d("ProfileEdit", "Save response: " + json);
                boolean ok = false;
                try {
                    JSONObject o = new JSONObject(json);
                    ok =
"updated".equalsIgnoreCase(o.optString("status"))
                            ||
"success".equalsIgnoreCase(o.optString("status"));
                } catch (Exception ignored) {}

                boolean finalOk = ok;
                runOnUiThread(() -> {
                    setLoading(false);
                    if (finalOk) {
                        // Persist (compat keys)
                        SharedPreferences.Editor ed =
getSharedPreferences("user_prefs", MODE_PRIVATE).edit();
                        ed.putString("username", name);
                        ed.putString("name", name);
                        ed.putInt("gender",
"Female".equalsIgnoreCase(gender) ? 1 : 0);
                        ed.putString("gender_text", gender);
```

```
                        ed.putString("height", heightStr);
                        ed.putString("height_cm", heightStr);
                        ed.putString("weight", weightStr);
                        ed.putString("weight_kg", weightStr);
                        ed.putString("target_BMI", targetStr);
                        ed.putString("target_bmi", targetStr);


                        String latestBmi = "";
                        String latestWeight = weightStr; // fallback
                        try {
                            JSONObject o = new JSONObject(json);
                            latestBmi    = o.optString("current_bmi",
"");
                            latestWeight = o.optString("weight_kg",
latestWeight);
                        } catch (Exception ignored) {}

                        if (!latestBmi.trim().isEmpty()) {
                            ed.putString("current_bmi", latestBmi);
                            tvCurrentBmi.setText(latestBmi);
                        }

                        ed.putString("weight", latestWeight);
                        ed.putString("weight_kg", latestWeight);
                        ed.apply();

                        Toast.makeText(ProfileEditActivity.this,
"Saved", Toast.LENGTH_SHORT).show();
                        finish();
                    } else {
                        Toast.makeText(ProfileEditActivity.this,
"Save error", Toast.LENGTH_SHORT).show();
                    }
                });
            }
        });
    }


    static class DecimalDigitsInputFilter implements InputFilter {
        private final int decimalDigits;
        DecimalDigitsInputFilter(int decimalDigits)
{ this.decimalDigits = decimalDigits; }
        @Override
        public CharSequence filter(CharSequence source, int start,
int end,
                                   Spanned dest, int dstart, int
dend) {
            String before = dest.subSequence(0, dstart).toString();
            String insert = source.subSequence(start,
end).toString();
            String after  = dest.subSequence(dend,
dest.length()).toString();
            String result = before + insert + after;

            if (result.isEmpty()) return null;
            int dot = result.indexOf('.');
            if (dot == -1) return null;
            if (result.indexOf('.', dot + 1) != -1) return ""; //
```

```
reject 2nd dot
            int digitsAfter = result.length() - dot - 1;
            return (digitsAfter > decimalDigits) ? "" : null;
        }
    }
}
```

QuestionActivity.java

```
package com.example.bmi;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Collections;

public class QuestionActivity extends AppCompatActivity {

    private RadioGroup answerRadioGroup;
    private TextView questionText;
    private Button nextButton;
    private ImageButton backButton;
    private TextView totalquestionText;
    private int totalQuestions = 5;
    private ArrayList<ArrayList<String>> answersMultiple = new
ArrayList<>(Collections.nCopies(totalQuestions, null));
    private int currentQuestion_Answer = 1;

    private ArrayList<String> answers = new
ArrayList<>(Collections.nCopies(totalQuestions, ""));

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.question);

        answerRadioGroup = findViewById(R.id.answerRadioGroup);
        questionText = findViewById(R.id.questionText);
        nextButton = findViewById(R.id.nextButton);
        backButton = findViewById(R.id.backButton);
        totalquestionText= findViewById(R.id.totalquestionText);
        totalquestionText.setText(currentQuestion_Answer-
```

```
1+"/"+totalQuestions);

        nextButton.setEnabled(false);


        loadQuestionFromFile(currentQuestion_Answer);
        loadRadioButtonsFromFile(currentQuestion_Answer);
        SharedPreferences sharedPreferences =
getSharedPreferences("user_prefs", MODE_PRIVATE);
        int size=sharedPreferences.getInt("text_size",12);
        changeTextSize(size);
        Log.e("size",String.valueOf(size));

        answerRadioGroup.setOnCheckedChangeListener(new
RadioGroup.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(RadioGroup group, int
checkedId) {
                nextButton.setEnabled(checkedId != -1);
            }
        });


        nextButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (currentQuestion_Answer == 5) {

                    ArrayList<String> selectedOptions = new
ArrayList<>();
                    for (int i = 0; i <
answerRadioGroup.getChildCount(); i++) {
                        View child = answerRadioGroup.getChildAt(i);
                        if (child instanceof android.widget.CheckBox)
{
                            android.widget.CheckBox checkBox =
(android.widget.CheckBox) child;
                            if (checkBox.isChecked()) {

selectedOptions.add(checkBox.getText().toString());
                            }
                        }
                    }
                    answersMultiple.set(0, selectedOptions);
                    Toast.makeText(QuestionActivity.this, "多选结果: "
+ selectedOptions.toString(), Toast.LENGTH_SHORT).show();

                    Intent intent = new Intent(QuestionActivity.this,
SuggestPlanActivity.class);


                    Log.e("MainActivity123", answers.toString());
                    intent.putStringArrayListExtra("singleAnswers",
answers);


                    SharedPreferences prefs =
getSharedPreferences("user_prefs", MODE_PRIVATE);
                    SharedPreferences.Editor editor = prefs.edit();
                    editor.putString("answersMultiple",
```

```
answersMultiple.toString());
                    editor.apply();
                    startActivity(intent);
                    finish();
                }


                int selectedId =
answerRadioGroup.getCheckedRadioButtonId();
                if (selectedId != -1) {
                    RadioButton selectedRadio =
findViewById(selectedId);
                    String selectedText =
selectedRadio.getText().toString();
                    answers.set(currentQuestion_Answer - 1,
selectedText);
                    Toast.makeText(QuestionActivity.this,
selectedText, Toast.LENGTH_SHORT).show();
                }

                if (currentQuestion_Answer < totalQuestions) {
                    currentQuestion_Answer++;
                    totalquestionText.setText(currentQuestion_Answer
- 1 + "/" + totalQuestions);
                    loadQuestionFromFile(currentQuestion_Answer);
                    loadRadioButtonsFromFile(currentQuestion_Answer);
                    nextButton.setEnabled(false);
                }

            }
        });



        backButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (currentQuestion_Answer > 1) {
                    currentQuestion_Answer--;
                    totalquestionText.setText(currentQuestion_Answer-
1+"/5");

                    loadQuestionFromFile(currentQuestion_Answer);
                    loadRadioButtonsFromFile(currentQuestion_Answer);
                    nextButton.setEnabled(false);
                }else if(currentQuestion_Answer == 1){
                    Intent intent = new Intent(QuestionActivity.this,
HomeActivity.class);
                    startActivity(intent);
                }
            }
        });
    }


    private void loadQuestionFromFile(int questionNumber) {
        InputStream inputStream =
getResources().openRawResource(R.raw.questionmcq);
        BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream));
```

```
        String line;
        try {
            while ((line = reader.readLine()) != null) {
                if (line.startsWith(String.valueOf(questionNumber)))
{
                    questionText.setText(line.substring( 1));
                    break;
                }
            }
            reader.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    private boolean hasAtLeastOneCheckboxChecked() {
        for (int i = 0; i < answerRadioGroup.getChildCount(); i++) {
            View child = answerRadioGroup.getChildAt(i);
            if (child instanceof android.widget.CheckBox) {
                if (((android.widget.CheckBox) child).isChecked()) {
                    return true;
                }
            }
        }
        return false;
    }


    private void loadRadioButtonsFromFile(int questionNumber) {
        answerRadioGroup.removeAllViews();
        InputStream inputStream =
getResources().openRawResource(R.raw.answermcq);
        BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream));
        String line;
        try {
            while ((line = reader.readLine()) != null) {
                if (line.startsWith(String.valueOf(questionNumber)))
{
                    String answerText = line.substring(1);

                    if (questionNumber == 5) {

                        android.widget.CheckBox checkBox = new
android.widget.CheckBox(this);
                        checkBox.setText(answerText);

                        checkBox.setId(View.generateViewId());
                        answerRadioGroup.addView(checkBox);


                        ArrayList<String> previousSelections =
answersMultiple.get(4);
                        if (previousSelections != null &&
previousSelections.contains(answerText)) {
                            checkBox.setChecked(true);
                        }


                        checkBox.setOnClickListener(new
View.OnClickListener() {
```

```
                            @Override
                            public void onClick(View v) {
                                if (hasAtLeastOneCheckboxChecked()) {
                                    nextButton.setEnabled(true);
                                } else {
                                    nextButton.setEnabled(false);
                                }
                            }
                        });

                    } else {

                        RadioButton radioButton = new
RadioButton(this);
                        radioButton.setText(answerText);
                        radioButton.setId(View.generateViewId());
                        answerRadioGroup.addView(radioButton);


                        if (answers.get(questionNumber -
1).equals(answerText)) {
                            radioButton.setChecked(true);
                            nextButton.setEnabled(true);
                        }
                    }
                }
            }
            reader.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    private void changeTextSize(float size) {

        TextView[] textViews = new TextView[] {
                findViewById(R.id.textView),
                findViewById(R.id.textView2),
                findViewById(R.id.textView6),
                findViewById(R.id.protein_text),
                findViewById(R.id.carbonhydrate_text),
                findViewById(R.id.Fat_text),
                findViewById(R.id.textView),
                findViewById(R.id.textView5),
                findViewById(R.id.textView7),
                findViewById(R.id.textView8),
                findViewById(R.id.textView3),
                findViewById(R.id.textView9),
                findViewById(R.id.textView10),
                findViewById(R.id.textView11),
                findViewById(R.id.textView12),
                findViewById(R.id.textView13),
                findViewById(R.id.textView14),
                findViewById(R.id.questionText),
                findViewById(R.id.totalquestionText),
                findViewById(R.id.planNameText),
                findViewById(R.id.sdgshfh),
                findViewById(R.id.asdfsaf),

        };
```

```
        Button[] buttons = new Button[] {
                //findViewById(R.id.Link_Suggest),
                //findViewById(R.id.Link_Personalized),
                //findViewById(R.id.Link_Shuffle),
                //findViewById(R.id.resultBMI),
                findViewById(R.id.GenerateButton),
                findViewById(R.id.SaveButton),
                findViewById(R.id.btnProfile),
                findViewById(R.id.btnPlanList),
                findViewById(R.id.textButton),
                findViewById(R.id.btnLogout),
                findViewById(R.id.nextButton),
                findViewById(R.id.viewButton),
                findViewById(R.id.deleteButton),
                findViewById(R.id.GenerateButton1),
                findViewById(R.id.backbutton1),
                // Add more Buttons here as needed
        };


        for (TextView textView : textViews) {
            if (textView != null) {
                textView.setTextSize(size);
            }
        }

        for (Button button : buttons) {
            if (button != null) {
                button.setTextSize(size);
            }
        }
    }
}
```

RedemptionHistoryActivity.java

```
package com.example.bmi;

import android.app.AlertDialog;
import android.content.SharedPreferences;
import android.graphics.Bitmap;
import android.graphics.Color;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import org.json.JSONArray;
import org.json.JSONObject;
```

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;

import okhttp3.Call;
import okhttp3.Callback;
import okhttp3.FormBody;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;

public class RedemptionHistoryActivity extends AppCompatActivity {

    private LinearLayout historyList;
    private int userId;
    private final OkHttpClient client = new OkHttpClient();
    private static final String REWARDS_URL =
"http://10.0.2.2/nutrition/rewards.php";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_redeem_history);

        // Back button
        ImageButton backBtn = findViewById(R.id.btnBack);
        backBtn.setOnClickListener(v -> finish());

        historyList = findViewById(R.id.historyList);

        SharedPreferences prefs = getSharedPreferences("user_prefs",
MODE_PRIVATE);
        userId = prefs.getInt("user_id", -1);

        fetchHistory();
    }

    private void fetchHistory() {
        if (userId <= 0) { Toast.makeText(this, "No user id",
Toast.LENGTH_SHORT).show(); return; }

        RequestBody body = new FormBody.Builder()
                .add("action", "get_history")
                .add("userid", String.valueOf(userId))
                .build();

        Request req = new
Request.Builder().url(REWARDS_URL).post(body).build();

        client.newCall(req).enqueue(new Callback() {
            @Override public void onFailure(Call call,
java.io.IOException e) {
                runOnUiThread(() ->

Toast.makeText(RedemptionHistoryActivity.this, "Load failed",
Toast.LENGTH_SHORT).show());
            }
```

```
            @Override public void onResponse(Call call, Response res)
throws java.io.IOException {
                String json = res.body().string();
                try {
                    JSONObject o = new JSONObject(json);
                    if
(!"success".equalsIgnoreCase(o.optString("status"))) {
                        runOnUiThread(() ->

Toast.makeText(RedemptionHistoryActivity.this, "No history",
Toast.LENGTH_SHORT).show());
                        return;
                    }

                    JSONArray items = o.optJSONArray("items");
                    runOnUiThread(() -> render(items));
                } catch (Exception ignore) {
                    runOnUiThread(() ->

Toast.makeText(RedemptionHistoryActivity.this, "Bad response",
Toast.LENGTH_SHORT).show());
                }
            }
        });
    }

    private void render(JSONArray items) {
        historyList.removeAllViews();
        if (items == null || items.length() == 0) {
            TextView empty = new TextView(this);
            empty.setText("No redemptions yet.");
            historyList.addView(empty);
            return;
        }

        LayoutInflater inf = LayoutInflater.from(this);
        for (int i = 0; i < items.length(); i++) {
            JSONObject it = items.optJSONObject(i);
            if (it == null) continue;

            View row = inf.inflate(R.layout.item_redeem_history,
historyList, false);

            String code   = it.optString("reward_code", "");
            String title  = it.optString("reward_title",
mapTitle(code));
            String when   = it.optString("redeemed_at", "");
            String expire = it.optString("expires_at", "");
            String status = it.optString("status",
computeStatus(expire));

            ((TextView)
row.findViewById(R.id.hTitle)).setText(title);
            ((TextView) row.findViewById(R.id.hCode)).setText(code);
            ((TextView) row.findViewById(R.id.hDates)).setText(
                    "Redeemed: " + when + "   •   Expires: " +
(expire.isEmpty() ? "-" : expire)
            );
            ((TextView)
row.findViewById(R.id.hStatus)).setText(status);
```

```
            // QR Code button
            Button qrBtn = row.findViewById(R.id.btnQr);
            qrBtn.setOnClickListener(v -> showQrDialog(code));

            historyList.addView(row);
        }
    }

    private String mapTitle(String code) {
        switch (code) {
            case "SMOOTHIE_FREE": return "Free Smoothie";
            case "DISC_10":       return "Discount on Next Order";
            case "VEGGIE_BOX":    return "Fruit & Veggie Basket";
            default:              return code;
        }
    }

    private String computeStatus(String expiresDate) {
        if (expiresDate == null || expiresDate.isEmpty()) return
"Active";
        try {
            Date exp = new SimpleDateFormat("yyyy-MM-dd",
Locale.getDefault()).parse(expiresDate);
            return (exp != null && exp.before(new Date())) ?
"Expired" : "Active";
        } catch (ParseException e) {
            return "Active";
        }
    }

    private void showQrDialog(String code) {
        com.google.zxing.qrcode.QRCodeWriter writer = new
com.google.zxing.qrcode.QRCodeWriter();
        try {
            int size = 500;

            // Custom message + code
            String qrContent = "Thanks for using this voucher ^-
^\nVoucher Code: " + code;

            com.google.zxing.common.BitMatrix bitMatrix =
                    writer.encode(qrContent,
com.google.zxing.BarcodeFormat.QR_CODE, size, size);

            Bitmap bmp = Bitmap.createBitmap(size, size,
Bitmap.Config.RGB_565);
            for (int x = 0; x < size; x++) {
                for (int y = 0; y < size; y++) {
                    bmp.setPixel(x, y, bitMatrix.get(x, y) ?
Color.BLACK : Color.WHITE);
                }
            }

            ImageView qrImage = new ImageView(this);
            qrImage.setImageBitmap(bmp);
            qrImage.setPadding(50, 50, 50, 50);

            new AlertDialog.Builder(this)
                    .setTitle("QR Code")
```

```
                    .setView(qrImage)
                    .setPositiveButton("Close", null)
                    .show();

        } catch (Exception e) {
            Toast.makeText(this, "QR generation failed",
Toast.LENGTH_SHORT).show();
        }
    }
}
```

RegistrationActivity.java

```
package com.example.bmi;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.TextView;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import android.text.InputType;
import android.widget.ImageButton;



import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

import java.util.HashMap;
import java.util.Map;

public class RegisterActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_register);

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.register)
, (v, insets) -> {
            Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
```

```
            v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom);
            return insets;
        });

        // Find the TextView
        TextView registerLink = findViewById(R.id.loginLink);

        ImageButton registerBackButton =
findViewById(R.id.registerBackButton);
        registerBackButton.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(RegisterActivity.this,
MainActivity.class);
                startActivity(intent);
            }
        });

        // Set click listener
        registerLink.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Navigate to RegisterActivity
                Intent intent = new Intent(RegisterActivity.this,
LoginActivity.class);
                startActivity(intent);
            }
        });

        Button registerButton = findViewById(R.id.registerButton);
        EditText username = findViewById(R.id.username);
        EditText email = findViewById(R.id.email);
        EditText password = findViewById(R.id.password);
        EditText cfpassword = findViewById(R.id.cfpassword);
        ImageButton togglePassword =
findViewById(R.id.togglePassword);
        ImageButton toggleCfPassword =
findViewById(R.id.toggleCfPassword);

        // Track visibility state
        final boolean[] isPasswordVisible = {false};
        final boolean[] isCfPasswordVisible = {false};

        togglePassword.setOnClickListener(v -> {
            if (isPasswordVisible[0]) {
                // Password is currently visible → Hide it
                password.setInputType(InputType.TYPE_CLASS_TEXT |
InputType.TYPE_TEXT_VARIATION_PASSWORD);

togglePassword.setImageResource(R.drawable.ic_visibility_on_24); //
eye open = hidden
            } else {
                // Password is currently hidden → Show it
                password.setInputType(InputType.TYPE_CLASS_TEXT |
InputType.TYPE_TEXT_VARIATION_VISIBLE_PASSWORD);

togglePassword.setImageResource(R.drawable.visibility_off); // eye
with slash = visible
```

```
            }

            isPasswordVisible[0] = !isPasswordVisible[0]; // toggle
AFTER
            password.setSelection(password.getText().length());
        });

        toggleCfPassword.setOnClickListener(v -> {
            if (isCfPasswordVisible[0]) {
                cfpassword.setInputType(InputType.TYPE_CLASS_TEXT |
InputType.TYPE_TEXT_VARIATION_PASSWORD);

toggleCfPassword.setImageResource(R.drawable.ic_visibility_on_24);
            } else {
                cfpassword.setInputType(InputType.TYPE_CLASS_TEXT |
InputType.TYPE_TEXT_VARIATION_VISIBLE_PASSWORD);

toggleCfPassword.setImageResource(R.drawable.visibility_off);
            }

            isCfPasswordVisible[0] = !isCfPasswordVisible[0]; //
toggle AFTER
            cfpassword.setSelection(cfpassword.getText().length());
        });

        registerButton.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View v) {
                String usernameStr =
username.getText().toString().trim();
                String emailStr = email.getText().toString().trim();
                String passwordStr =
password.getText().toString().trim();
                String cfPasswordStr =
cfpassword.getText().toString().trim();

                if (!passwordStr.equals(cfPasswordStr)) {
                    Toast.makeText(RegisterActivity.this, "Passwords
do not match", Toast.LENGTH_SHORT).show();
                    return;
                }

                // Send data to PHP server
                StringRequest stringRequest = new
StringRequest(Request.Method.POST,
"http://10.0.2.2/nutrition/login.php",
                        new Response.Listener<String>() {
                            @Override
                            public void onResponse(String response) {
                                switch (response.trim()) {
                                    case "register_success":
                                        Intent intent = new
Intent(RegisterActivity.this, LoginActivity.class);
                                        startActivity(intent);

Toast.makeText(RegisterActivity.this, "Registration successful!",
Toast.LENGTH_SHORT).show();
                                        break;
                                    case "gmail_exists":
```

```
Toast.makeText(RegisterActivity.this, "Gmail already registered!",
Toast.LENGTH_SHORT).show();
                                        break;
                                    case "invalid_gmail":

Toast.makeText(RegisterActivity.this, "Please enter a valid Gmail
address.", Toast.LENGTH_SHORT).show();
                                        break;
                                    case "weak_password":

Toast.makeText(RegisterActivity.this, "Password must be at least 8
characters, include a number and a symbol.",
Toast.LENGTH_LONG).show();
                                        break;
                                    default:

Toast.makeText(RegisterActivity.this, "Server response: " + response,
Toast.LENGTH_SHORT).show();
                                        Log.e("MYSQL",response);
                                        break;
                                }
                            }
                        },
                        new Response.ErrorListener() {
                            @Override
                            public void onErrorResponse(VolleyError
error) {
                                Toast.makeText(RegisterActivity.this,
"Error: " + error.getMessage(), Toast.LENGTH_SHORT).show();
                            }
                        }) {
                    @Override
                    protected Map<String, String> getParams() {
                        Map<String, String> params = new HashMap<>();
                        params.put("action", "register");
                        params.put("username", usernameStr);
                        params.put("gmail", emailStr);
                        params.put("password", passwordStr);
                        return params;
                    }
                };

                RequestQueue queue =
Volley.newRequestQueue(RegisterActivity.this);
                queue.add(stringRequest);
            }
        });


    }


}
```

RelativeTimeUtil.java

```
package com.example.bmi;

import java.util.Calendar;
import java.util.Locale;

public class RelativeTimeUtil {
    public static String relativeTime(long millis) {
        long d = dayDiffLocal(millis);
        if (d == 0) return "today";
        if (d == 1) return "yesterday";
        if (d < 7)  return d + " days ago";
        if (d < 30) return (d / 7) + " weeks ago";
        return (d / 30) + " months ago";
    }
    private static long dayDiffLocal(long millis) {
        return (midnight(System.currentTimeMillis()) -
midnight(millis)) / 86_400_000L;
    }
    private static long midnight(long millis) {
        Calendar c = Calendar.getInstance();
        c.setTimeInMillis(millis);
        c.set(Calendar.HOUR_OF_DAY, 0);
        c.set(Calendar.MINUTE, 0);
        c.set(Calendar.SECOND, 0);
        c.set(Calendar.MILLISECOND, 0);
        return c.getTimeInMillis();
    }
}
```

SettingsActivity.java

```
package com.example.bmi;

import android.Manifest;
import android.app.AlarmManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.net.Uri; // NEW: Added for battery optimization intent
import android.os.Build;
import android.os.Bundle;
import android.provider.Settings;
import android.util.Log;
import android.widget.LinearLayout;
import android.widget.Toast;
import android.widget.TextView;
import android.util.TypedValue;
import android.os.PowerManager; // NEW: Added for PowerManager

import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.contract.ActivityResultContracts;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;

import
```

```
com.google.android.material.bottomnavigation.BottomNavigationView;
import com.google.android.material.dialog.MaterialAlertDialogBuilder;
import com.google.android.material.switchmaterial.SwitchMaterial;

import java.util.Calendar;
import java.util.TimeZone;

public class SettingsActivity extends AppCompatActivity {

    private static final String PREFS = "user_prefs";
    private static final String KEY_MEAL_NOTIF_ENABLED_PREFIX =
"meal_notif_enabled_user_";
    private static final String KEY_TEXT_SCALE = "text_scale";
    private static final String PREFS_UI = "ui_prefs";
    private static final String PREFS_SESSION = "session_prefs";
    private static final String KEY_TEXT_SCALE_PREFIX =
"text_scale_user_";
    private static final String KEY_SESSION_USER_ID = "user_id";
    private static final int REQ_BF = 21001;
    private static final int REQ_LU = 21002;
    private static final int REQ_DI = 21003;

    private SwitchMaterial swMealReminders;
    private LinearLayout rowTextSize, rowLogout;
    private ActivityResultLauncher<String> notifPermLauncher;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        String uid = getCurrentUserId();
        float savedScale = getSharedPreferences(PREFS_UI,
MODE_PRIVATE)
                .getFloat(KEY_TEXT_SCALE_PREFIX + uid, 1.0f);

        applyTextScale(savedScale);

        setContentView(R.layout.activity_settings);

        rowTextSize = findViewById(R.id.rowTextSize);
        rowLogout = findViewById(R.id.rowLogout);

        BottomNavigationView bottomNav =
findViewById(R.id.bottom_navigation);
        if (bottomNav != null) {
            bottomNav.setSelectedItemId(R.id.nav_settings);
            bottomNav.setOnItemSelectedListener(item -> {
                int id = item.getItemId();
                if (id == R.id.nav_home) {
                    startActivity(new Intent(this,
HomeActivity.class));
                    return true;
                }
                if (id == R.id.nav_diet) {
                    startActivity(new Intent(this,
DietaryActivity.class));
                    return true;
                }
                if (id == R.id.nav_tracking) {
                    startActivity(new Intent(this,
TrackingActivity.class));
```

```
                    return true;
                }
                if (id == R.id.nav_voucher) {
                    startActivity(new Intent(this,
VoucherActivity.class));
                    return true;
                }
                if (id == R.id.nav_settings) {
                    return true;
                }
                return false;
            });
        }

        LinearLayout rowProfile = findViewById(R.id.rowProfile);
        LinearLayout rowDietary = findViewById(R.id.rowDietary);
        LinearLayout rowNotifications =
findViewById(R.id.rowNotifications);
        LinearLayout rowHelp = findViewById(R.id.rowHelp);
        LinearLayout rowAbout = findViewById(R.id.rowAbout);

        rowProfile.setOnClickListener(v ->
                startActivity(new Intent(this,
ProfileEditActivity.class)));

        rowDietary.setOnClickListener(v ->
                startActivity(new Intent(this,
EditPreferenceActivity.class)));



        swMealReminders = findViewById(R.id.swMealReminders);

        String notifKey = KEY_MEAL_NOTIF_ENABLED_PREFIX + uid;
        boolean enabled = getSharedPreferences(PREFS, MODE_PRIVATE)
                .getBoolean(notifKey, false);
        Log.d("SettingsActivity", "User: " + uid + ",
meal_notif_enabled: " + enabled);
        if (swMealReminders != null)
swMealReminders.setChecked(enabled);

        rowNotifications.setOnClickListener(v -> {
            if (swMealReminders != null) swMealReminders.toggle();
        });

        notifPermLauncher = registerForActivityResult(
                new ActivityResultContracts.RequestPermission(),
                granted -> {
                    if (granted) {
                        if (checkExactAlarmPermission(this)) {
                            // NEW: Request battery optimization
exemption
                            requestIgnoreBatteryOptimizations(this);
                            scheduleAllMealAlarms(this);
                            saveToggle(true, uid);
                            Toast.makeText(this, "Meal reminders
enabled", Toast.LENGTH_SHORT).show();
                        } else {
                            swMealReminders.setChecked(false);
                            saveToggle(false, uid);
```

```
                            Toast.makeText(this, "Exact alarm
permission required", Toast.LENGTH_SHORT).show();
                        }
                    } else {
                        if (swMealReminders != null)
swMealReminders.setChecked(false);
                        saveToggle(false, uid);
                        Toast.makeText(this, "Notification permission
required", Toast.LENGTH_SHORT).show();
                    }
                }
        );

        if (swMealReminders != null) {
            swMealReminders.setOnCheckedChangeListener((buttonView,
isChecked) -> {
                Log.d("SettingsActivity", "Toggle changed for user: "
+ uid + ", isChecked: " + isChecked);
                if (isChecked) {
                    if (needsPostNotifPermission() &&
                            ContextCompat.checkSelfPermission(this,
Manifest.permission.POST_NOTIFICATIONS)
                                    !=
PackageManager.PERMISSION_GRANTED) {

notifPermLauncher.launch(Manifest.permission.POST_NOTIFICATIONS);
                    } else if (checkExactAlarmPermission(this)) {
                        // NEW: Request battery optimization
exemption
                        requestIgnoreBatteryOptimizations(this);
                        scheduleAllMealAlarms(this);
                        saveToggle(true, uid);
                        Toast.makeText(this, "Meal reminders
enabled", Toast.LENGTH_SHORT).show();
                    } else {
                        swMealReminders.setChecked(false);
                        saveToggle(false, uid);
                        Toast.makeText(this, "Exact alarm permission
required", Toast.LENGTH_SHORT).show();
                        Intent intent = new
Intent(Settings.ACTION_REQUEST_SCHEDULE_EXACT_ALARM);
                        startActivity(intent);
                    }
                } else {
                    cancelAllMealAlarms(this);
                    saveToggle(false, uid);
                    Toast.makeText(this, "Meal reminders disabled",
Toast.LENGTH_SHORT).show();
                }
            });
        }

        if (enabled && checkExactAlarmPermission(this) &&
                (!needsPostNotifPermission() ||
                        ContextCompat.checkSelfPermission(this,
Manifest.permission.POST_NOTIFICATIONS)
                                ==
PackageManager.PERMISSION_GRANTED)) {
            // NEW: Request battery optimization exemption on startup
if enabled
```

```
            requestIgnoreBatteryOptimizations(this);
            scheduleAllMealAlarms(this);
        }

        rowHelp.setOnClickListener(v -> showHelpDialog());
        rowAbout.setOnClickListener(v -> showAboutDialog());

        if (rowTextSize != null) {
            rowTextSize.setOnClickListener(v ->
showTextSizeDialog());
        }
        if (rowLogout != null) {
            rowLogout.setOnClickListener(v -> confirmLogout());
        }
    }

    private void showHelpDialog() {
        new MaterialAlertDialogBuilder(this)
                .setTitle("Help & Support")
                .setMessage(
                        "For questions or feedback:\n\n" +
                                "• FAQ available inside the app.\n" +
                                "• Support hours: Mon–Fri, 9:00–
18:00.\n" +
                                "• We usually reply within 1–2
business days.")
                .setPositiveButton("OK", null)
                .show();
    }

    private void showAboutDialog() {
        new MaterialAlertDialogBuilder(this)
                .setTitle("About")
                .setMessage(
                        "NutriForU v1.0.0\n\n" +
                                "A lightweight nutrition companion to
plan meals, track calories and BMI, " +
                                "and keep you on course with gentle
reminders.\n\n" +
                                "© 2025 NutriForU")
                .setPositiveButton("OK", null)
                .show();
    }

    private boolean needsPostNotifPermission() {
        return Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU;
    }

    private boolean checkExactAlarmPermission(Context ctx) {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {
            AlarmManager am = (AlarmManager)
ctx.getSystemService(Context.ALARM_SERVICE);
            boolean canSchedule = am.canScheduleExactAlarms();
            Log.d("SettingsActivity", "Can schedule exact alarms: " +
canSchedule);
            return canSchedule;
        }
        return true;
    }
```

```
    private void saveToggle(boolean enabled, String uid) {
        String notifKey = KEY_MEAL_NOTIF_ENABLED_PREFIX + uid;
        Log.d("SettingsActivity", "Saving " + notifKey + ": " +
enabled);
        getSharedPreferences(PREFS, MODE_PRIVATE)
                .edit()
                .putBoolean(notifKey, enabled)
                .apply();
    }

    // NEW: Method to request battery optimization exemption
    private void requestIgnoreBatteryOptimizations(Context ctx) {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            PowerManager pm = (PowerManager)
ctx.getSystemService(Context.POWER_SERVICE);
            if
(!pm.isIgnoringBatteryOptimizations(ctx.getPackageName())) {
                Log.d("SettingsActivity", "Requesting battery
optimization exemption");
                Intent intent = new
Intent(Settings.ACTION_REQUEST_IGNORE_BATTERY_OPTIMIZATIONS);
                intent.setData(Uri.parse("package:" +
ctx.getPackageName()));
                ctx.startActivity(intent);
            } else {
                Log.d("SettingsActivity", "Battery optimization
already disabled");
            }
        }
    }

    private void scheduleAllMealAlarms(Context ctx) {
        scheduleDaily(ctx, REQ_BF, 8, 0, "Breakfast time! Log your
meal.");
        scheduleDaily(ctx, REQ_LU, 13, 0, "Lunch time! Log your
meal.");
        scheduleDaily(ctx, REQ_DI, 19, 0, "Dinner time! Log your
meal.");
    }

    private void cancelAllMealAlarms(Context ctx) {
        cancel(ctx, REQ_BF);
        cancel(ctx, REQ_LU);
        cancel(ctx, REQ_DI);
    }

//    private void scheduleDaily(Context ctx, int requestCode, int
hour24, int minute, String message) {
//        AlarmManager am = (AlarmManager)
ctx.getSystemService(Context.ALARM_SERVICE);
//        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S
&& !am.canScheduleExactAlarms()) {
//            Log.e("SettingsActivity", "Cannot schedule exact
alarms!");
//            Toast.makeText(ctx, "Please allow exact alarms in
settings", Toast.LENGTH_LONG).show();
//            return;
//        }
//
//        Intent i = new Intent(ctx, MealReminderReceiver.class);
```

```
//         i.setAction("com.example.bmi.MEAL_REMINDER");
//         i.putExtra("message", message);
//         i.putExtra("requestCode", requestCode);
//
//         PendingIntent pi = PendingIntent.getBroadcast(
//                 ctx, requestCode, i,
//                 PendingIntent.FLAG_UPDATE_CURRENT |
PendingIntent.FLAG_IMMUTABLE
//         );
//
//         Calendar cal = Calendar.getInstance(TimeZone.getDefault());
//         cal.set(Calendar.SECOND, 0);
//         cal.set(Calendar.MILLISECOND, 0);
//         cal.set(Calendar.HOUR_OF_DAY, hour24);
//         cal.set(Calendar.MINUTE, minute);
//
//         if (cal.getTimeInMillis() <= System.currentTimeMillis()) {
//             cal.add(Calendar.DAY_OF_YEAR, 1);
//         }
//
//         Log.d("SettingsActivity", "Scheduling alarm for
requestCode: " + requestCode + " at " + cal.getTime());
//         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
//             am.setExactAndAllowWhileIdle(AlarmManager.RTC_WAKEUP,
cal.getTimeInMillis(), pi);
//         } else {
//             am.setExact(AlarmManager.RTC_WAKEUP,
cal.getTimeInMillis(), pi);
//         }
//     }

    private void scheduleDaily(Context ctx, int requestCode, int
hour24, int minute, String message) {
        AlarmManager am = (AlarmManager)
ctx.getSystemService(Context.ALARM_SERVICE);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S
&& !am.canScheduleExactAlarms()) {
            Log.e("SettingsActivity", "Cannot schedule exact
alarms!");
            Toast.makeText(ctx, "Please allow exact alarms in
settings", Toast.LENGTH_LONG).show();
            return;
        }

        Intent i = new Intent(ctx, MealReminderReceiver.class);
        i.setAction("com.example.bmi.MEAL_REMINDER");
        i.putExtra("message", message);
        i.putExtra("requestCode", requestCode);

        PendingIntent pi = PendingIntent.getBroadcast(
                ctx, requestCode, i,
                PendingIntent.FLAG_UPDATE_CURRENT |
PendingIntent.FLAG_IMMUTABLE
        );

        Calendar cal = Calendar.getInstance(TimeZone.getDefault());
        cal.set(Calendar.SECOND, 0);
        cal.set(Calendar.MILLISECOND, 0);
        cal.set(Calendar.HOUR_OF_DAY, hour24);
        cal.set(Calendar.MINUTE, minute);
```

```
        if (cal.getTimeInMillis() <= System.currentTimeMillis()) {
            cal.add(Calendar.DAY_OF_YEAR, 1);
        }
        Log.d("SettingsActivity", "Scheduling alarm for requestCode:
" + requestCode + " at " + cal.getTime());
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            am.setExactAndAllowWhileIdle(AlarmManager.RTC_WAKEUP,
cal.getTimeInMillis(), pi);
        } else {
            am.setExact(AlarmManager.RTC_WAKEUP,
cal.getTimeInMillis(), pi);
        }
    }

    private void cancel(Context ctx, int requestCode) {
        AlarmManager am = (AlarmManager)
ctx.getSystemService(Context.ALARM_SERVICE);
        Intent i = new Intent(ctx,
MealReminderReceiver.class).setAction("com.example.bmi.MEAL_REMINDER"
);
        PendingIntent pi = PendingIntent.getBroadcast(
                ctx, requestCode, i,
                PendingIntent.FLAG_UPDATE_CURRENT |
PendingIntent.FLAG_IMMUTABLE
        );
        am.cancel(pi);
    }

    private void showTextSizeDialog() {
        String uid = getCurrentUserId();
        float current = getSharedPreferences(PREFS_UI, MODE_PRIVATE)
                .getFloat(KEY_TEXT_SCALE_PREFIX + uid, 1.0f);
        int progress = (int) ((current - 1.0f) * 100);
        if (progress < 0) progress = 0;
        if (progress > 50) progress = 50;

        LinearLayout container = new LinearLayout(this);
        container.setOrientation(LinearLayout.VERTICAL);
        int pad = (int) (16 *
getResources().getDisplayMetrics().density);
        container.setPadding(pad, pad, pad, pad);

        final TextView tvLive = new TextView(this);
        tvLive.setText("Adjust app text size");
        tvLive.setPadding(0, 0, 0, pad / 2);
        container.addView(tvLive,
                new
LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT,
                        LinearLayout.LayoutParams.WRAP_CONTENT));

        final android.widget.SeekBar seek = new
android.widget.SeekBar(this);
        seek.setMax(50);
        seek.setProgress(progress);
        LinearLayout.LayoutParams sbLp = new
LinearLayout.LayoutParams(
                LinearLayout.LayoutParams.MATCH_PARENT,
                LinearLayout.LayoutParams.WRAP_CONTENT);
        sbLp.setMargins(pad / 2, pad / 2, pad / 2, 0);
        seek.setLayoutParams(sbLp);
```

```
        container.addView(seek);

        float baseSp = 16f;
        tvLive.setTextSize(baseSp * current);
        seek.setOnSeekBarChangeListener(new
android.widget.SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(android.widget.SeekBar s,
int p, boolean fromUser) {
                float scale = 1.0f + (p / 100f);
                tvLive.setTextSize(baseSp * scale);
            }
            @Override
            public void onStartTrackingTouch(android.widget.SeekBar
s) {}
            @Override
            public void onStopTrackingTouch(android.widget.SeekBar s)
{}
        });

        new
com.google.android.material.dialog.MaterialAlertDialogBuilder(this)
                .setTitle("Text Size")
                .setView(container)
                .setNegativeButton("Cancel", null)
                .setPositiveButton("Save", (d, w) -> {
                    float scale = 1.0f + (seek.getProgress() / 100f);
                    getSharedPreferences(PREFS_UI, MODE_PRIVATE)
                            .edit()
                            .putFloat(KEY_TEXT_SCALE_PREFIX + uid,
scale)
                            .apply();
                    applyTextScale(scale);
                    recreate();
                    Toast.makeText(this, "Text size updated",
Toast.LENGTH_SHORT).show();
                })
                .show();
    }

    private void applyTextScale(float scale) {
        android.content.res.Configuration cfg =
                new
android.content.res.Configuration(getResources().getConfiguration());
        cfg.fontScale = scale;

        android.util.DisplayMetrics metrics =
getResources().getDisplayMetrics();
        android.view.WindowManager wm = (android.view.WindowManager)
getSystemService(Context.WINDOW_SERVICE);
        wm.getDefaultDisplay().getMetrics(metrics);
        metrics.scaledDensity = cfg.fontScale * metrics.density;

        getResources().updateConfiguration(cfg, metrics);
    }

    private void confirmLogout() {
        new MaterialAlertDialogBuilder(this)
                .setTitle("Log out")
                .setMessage("Are you sure you want to log out?")
```

```
                .setNegativeButton("Cancel", null)
                .setPositiveButton("Log out", (d, w) -> {
                    cancelAllMealAlarms(this);

                    String uid = getCurrentUserId();
                    getSharedPreferences(PREFS, MODE_PRIVATE)
                            .edit()
                            .remove(KEY_MEAL_NOTIF_ENABLED_PREFIX +
uid)
                            .apply();

                    getSharedPreferences(PREFS_SESSION,
MODE_PRIVATE).edit().clear().apply();

                    Intent i = new Intent(this, MainActivity.class);
                    i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
                    startActivity(i);
                    finish();
                })
                .show();
    }

    private String getCurrentUserId() {
        SharedPreferences sp = getSharedPreferences(PREFS_SESSION,
MODE_PRIVATE);
        String uid = sp.getString(KEY_SESSION_USER_ID, null);
        return (uid == null || uid.trim().isEmpty()) ? "guest" :
uid.trim();
    }
}
```

SplashActivity.java

```
package com.example.bmi;

import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.os.Handler;

import androidx.activity.ComponentActivity;
import androidx.core.splashscreen.SplashScreen;

public class SplashActivity extends ComponentActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // Ensure system splash shows briefly (Android 12+ only)
        SplashScreen splashScreen = null;
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {
            splashScreen = SplashScreen.installSplashScreen(this);
            splashScreen.setKeepOnScreenCondition(() -> false);  //
skip linger
        }

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash);
```

```
        // Delay before launching main screen
        new Handler().postDelayed(() -> {
            startActivity(new Intent(SplashActivity.this,
MainActivity.class));
            finish();
        }, 3000);  // 2 seconds
    }
}
```

SuggestPlanActivity.java

```
package com.example.bmi;

import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.os.Bundle;
import android.text.InputType;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import com.github.lzyzsd.circleprogress.DonutProgress;
import com.github.mikephil.charting.charts.PieChart;
import com.github.mikephil.charting.data.PieData;
import com.github.mikephil.charting.data.PieDataSet;
import com.github.mikephil.charting.data.PieEntry;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class SuggestPlanActivity extends AppCompatActivity {
    private double weight = 90;
    private int height = 180;
    private String gender = "Male";
```

```
    private String mealtext="";
    private int age = 21;
    private double bmr = 0;
    private String target = "Diet";
    private String prefferedCarbs="";
    private String prefferedProtein="";
    private DonutProgress bmr_progressbar;
    private DonutProgress protein_progressbar;
    private DonutProgress carbonhydrate_progressbar;
    private DonutProgress fat_progressbar;
    private TextView sdgshfh;
    private int proteinRate=0;
    private int carbsRate=0;
    private int fatRate=0;
    private int proteinNeeded;
    private int carbsNeeded;
    private int fatNeeded;
    private double targetCalories=0;
    private String planName;
    private LinearLayout chartsContainer;
    private String answer;
    private int userId;
    private String function;
    private Button GenerateButton1;
    private Button SaveButton;


    private Map<String, Integer> dailyMacros = new HashMap<>();
    private Map<String, Map<String, Integer>> mealFoods = new
LinkedHashMap<>();
    private ArrayList<String> singleAnswers;
    private String allergyFood;
    private String multipleAnswers="";
    private ImageButton backButton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.suggestplan);
        Intent intent = new Intent();

        GenerateButton1=findViewById(R.id.GenerateButton1);
        GenerateButton1.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View v) {
               askGptQuestion();
            }
        });
        backButton=findViewById(R.id.backbutton2);
        backButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                Intent intent = new Intent(SuggestPlanActivity.this,
HomeActivity.class);
                startActivity(intent);
            }
        });
        SaveButton=findViewById(R.id.SaveButton);;
        SaveButton.setOnClickListener(new View.OnClickListener() {
```

```
            @Override
            public void onClick(View v) {
                AlertDialog.Builder builder = new
AlertDialog.Builder(SuggestPlanActivity.this);
                builder.setTitle("Key in the Plan Name");


                final EditText input = new
EditText(SuggestPlanActivity.this);
                input.setInputType(InputType.TYPE_CLASS_TEXT);
                input.setHint("Exp:Diet Plan");
                builder.setView(input);


                builder.setPositiveButton("Save", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int
which) {
                        planName = input.getText().toString().trim();
                        if (!planName.isEmpty()) {

                            SendtoDatabase();
                            Toast.makeText(SuggestPlanActivity.this,
"Saved: " + planName, Toast.LENGTH_SHORT).show();
                        } else {
                            Toast.makeText(SuggestPlanActivity.this,
"Plan name should not be empty", Toast.LENGTH_SHORT).show();
                        }
                    }
                });


                builder.setNegativeButton("取消", null);


                builder.show();
            }
        });
        SaveButton.setEnabled(false);

        SharedPreferences sharedPreferences =
getSharedPreferences("user_prefs", MODE_PRIVATE);
        int size=sharedPreferences.getInt("text_size",12);
        changeTextSize(size);
        userId = sharedPreferences.getInt("user_id", 2);
        function = sharedPreferences.getString("function",
"Suggest");
        height = sharedPreferences.getInt("height", 180);
        gender = sharedPreferences.getString("gender", "Male");
        weight = sharedPreferences.getFloat("weight", 78);
        age = sharedPreferences.getInt("age", 22);
        allergyFood= sharedPreferences.getString("allergies",
"[Eggs,Meats]");
        singleAnswers =
getIntent().getStringArrayListExtra("singleAnswers");
        multipleAnswers=
sharedPreferences.getString("answersMultiple", "1");
        Log.e("SuggestPlan",String.valueOf(userId));
```

```
        Log.e("SuggestPlan",String.valueOf(function));
        Log.e("SuggestPlan",String.valueOf(height));
        Log.e("SuggestPlan",String.valueOf(gender));
        Log.e("SuggestPlan",String.valueOf(weight));
        Log.e("SuggestPlan",String.valueOf(age));
        Log.e("SuggestPlan",String.valueOf(allergyFood));
        Log.e("SuggestPlan",String.valueOf(multipleAnswers));




        if (function == "Suggest") {
            if (singleAnswers != null && singleAnswers.size() > 3) {
                target = singleAnswers.get(0);
                prefferedCarbs = singleAnswers.get(2);
                prefferedProtein = singleAnswers.get(3);
                Log.e("ReceivedData", target + prefferedCarbs +
prefferedProtein);
            } else {
                Log.e("ReceivedData", "singleAnswers is null or
doesn't contain enough elements.");
            }


            Log.e("SuggestPlan", "Wahaha " +multipleAnswers);
        }

Log.e("SuggestPlan","Wahaha"+String.valueOf(multipleAnswers));

            protein_progressbar =
findViewById(R.id.protein_progressbar);
            carbonhydrate_progressbar =
findViewById(R.id.carbonhydrate_progressbar);
            fat_progressbar = findViewById(R.id.fat_progressbar);
            bmr_progressbar = findViewById(R.id.bmr_progressbar);
            sdgshfh = findViewById(R.id.sdgshfh);
            chartsContainer = findViewById(R.id.piechartLinear);
            if (gender.equals("Male")) {
                bmr = (10 * weight) + (6.25 * height) - (5 * age) +
5;
                bmr_progressbar.setMax((int) bmr);
                bmr_progressbar.setProgress((int) bmr);
                bmr_progressbar.setText((int) bmr + "kcal");
            } else {
                bmr = (10 * weight) + (6.25 * height) - (5 * age) -
161;
                bmr_progressbar.setMax((int) bmr);
                bmr_progressbar.setProgress((int) bmr);
                bmr_progressbar.setText((int) bmr + "kcal");
            }

            if (target.equals("Gain more muscle")) {
                targetCalories = bmr + 300;
                proteinRate = 25;
                carbsRate = 50;
                fatRate = 25;
                proteinNeeded = (int)((targetCalories / 100 *
proteinRate) / 4);
                carbsNeeded =  (int)((targetCalories / 100 *
carbsRate) / 4);
```

```
                fatNeeded =  (int)((targetCalories / 100 * fatRate) /
4);
            } else if (target.equals("Diet")) {
                targetCalories = bmr - 300;
                proteinRate = 35;
                carbsRate = 35;
                fatRate = 30;
                proteinNeeded = (int)((targetCalories / 100 *
proteinRate) / 4);
                carbsNeeded =  (int)((targetCalories / 100 *
carbsRate) / 4);
                fatNeeded =  (int)((targetCalories / 100 * fatRate) /
4);
            } else {
                targetCalories = bmr - 300;
                proteinRate = 35;
                carbsRate = 35;
                fatRate = 30;
                proteinNeeded = (int)((targetCalories / 100 *
proteinRate) / 4);
                carbsNeeded =  (int)((targetCalories / 100 *
carbsRate) / 4);
                fatNeeded =  (int)((targetCalories / 100 * fatRate) /
4);
            }
            Log.e("Shuffle",function);
            Log.e("Shuffle", String.valueOf(bmr));
            Log.e("Shuffle", String.valueOf(gender));
            Log.e("Shuffle",String.valueOf(height));
            Log.e("Shuffle", String.valueOf(weight));
            Log.e("Shuffle", String.valueOf(age));
        Log.e("Shuffle", String.valueOf(targetCalories));
            protein_progressbar.setMax((int) proteinNeeded);
            protein_progressbar.setProgress((int) proteinNeeded);

protein_progressbar.setText(String.valueOf(proteinNeeded));

            carbonhydrate_progressbar.setMax((int) carbsNeeded);
            carbonhydrate_progressbar.setProgress((int) carbsNeeded);

carbonhydrate_progressbar.setText(String.valueOf(carbsNeeded));

            fat_progressbar.setMax((int) fatNeeded);
            fat_progressbar.setProgress((int) fatNeeded);
            fat_progressbar.setText(String.valueOf(fatNeeded));
            askGptQuestion();

    }
    private void askGptQuestion() {
        Log.e("Running","Running123");

        sdgshfh.setText(""); // 🧹 Clear previous text first
        sdgshfh.setText("Please wait, analyzing...");

        String prompt =
                "You are a meal planning assistant.\n\n" +

                        "I will give you my daily macronutrient
goals. Please help me generate a meal plan for one day (3 meals:
breakfast, lunch, and dinner).\n\n" +
```

```
                        "I have allergies to "+allergyFood+",do not
include these foods into my meal plan"+"\n\n" +

                        "IMPORTANT RULES:\n" +
                        "- For each meal, provide exactly 1 food for
protein, 1 for carbs, and 1 for fat.\n" +
                        "- Each nutrient must be from a different
food. Do NOT mix nutrients in one food item.\n" +
                        "- The number inside the parentheses means:
how many grams of that food I need to eat to get the required amount
of that nutrient.\n" +
                        "- Do NOT write how much protein/carbs/fat
the food contains. Just how many grams of food to eat.\n" +
                        "- DO NOT explain anything. Just return the
result in the strict format below.\n\n" +
                        "- DO NOT simply make a space. Just return
the result in the strict format below.\n\n" +
                        "FORMAT:\n" +
                        "(Breakfast:[ProteinFoodName(amount in grams
to eat),CarbsFoodName(amount in grams to eat),FatFoodName(amount in
grams to eat)])\n" +
                        "(Lunch:[ProteinFoodName(amount in grams to
eat),CarbsFoodName(amount in grams to eat),FatFoodName(amount in
grams to eat)])\n" +
                        "(Dinner:[ProteinFoodName(amount in grams to
eat),CarbsFoodName(amount in grams to eat),FatFoodName(amount in
grams to eat)])\n\n" +

                        "Your answer must follow this format
exactly.\n" +
                        "The total of the day should match
approximately:\n" +
                        "- Protein: 120g\n" +
                        "- Carbs: 200g\n" +
                        "- Fat: 60g\n\n" +

                        "If you output anything outside this format,
I won't be able to parse it in my app.\n" +
                        "Please strictly follow the
instructions.\n\n" +

                        "Let's begin.";

    if(function=="Suggest"){
        Log.e("Function",function );
        prompt =
                "You are a meal planning assistant.\n\n" +

                        "I will give you my daily macronutrient
goals. Please help me generate a meal plan for one day (3 meals:
breakfast, lunch, and dinner).\n\n" +
                        "For carbs food i prefer
"+prefferedCarbs+",and for protein food i prefer
"+prefferedProtein+"\n\n" +
                        "I have allergies to
"+multipleAnswers.toString()+",do not include these foods into my
meal plan"+"\n\n" +
                        "IMPORTANT RULES:\n" +
                        "- For each meal, provide exactly 1 food for
protein, 1 for carbs, and 1 for fat.\n" +
```

```
                        "- Each nutrient must be from a different
food. Do NOT mix nutrients in one food item.\n" +
                        "- The number inside the parentheses means:
how many grams of that food I need to eat to get the required amount
of that nutrient.\n" +
                        "- Do NOT write how much protein/carbs/fat
the food contains. Just how many grams of food to eat.\n" +
                        "- DO NOT explain anything. Just return the
result in the strict format below.\n\n" +
                        "- DO NOT simply make a space. Just return
the result in the strict format below.\n\n" +
                        "FORMAT:\n" +
                        "(Breakfast:[ProteinFoodName(amount in grams
to eat),CarbsFoodName(amount in grams to eat),FatFoodName(amount in
grams to eat)])\n" +
                        "(Lunch:[ProteinFoodName(amount in grams to
eat),CarbsFoodName(amount in grams to eat),FatFoodName(amount in
grams to eat)])\n" +
                        "(Dinner:[ProteinFoodName(amount in grams to
eat),CarbsFoodName(amount in grams to eat),FatFoodName(amount in
grams to eat)])\n\n" +

                        "Your answer must follow this format
exactly.\n" +
                        "The total of the day should match
approximately:\n" +
                        "- Protein: 120g\n" +
                        "- Carbs: 200g\n" +
                        "- Fat: 60g\n\n" +

                        "If you output anything outside this format,
I won't be able to parse it in my app.\n" +
                        "Please strictly follow the
instructions.\n\n" +

                        "Let's begin.";
    }



        CohereRequest request = new CohereRequest(prompt);

        CohereApiClient apiClient = new CohereApiClient();
        CohereService service = apiClient.createService();

        sdgshfh.setText("Please wait, analyzing...");

        service.generate(request).enqueue(new
Callback<CohereResponse>() {
            @Override
            public void onResponse(Call<CohereResponse> call,
Response<CohereResponse> response) {
                if (response.isSuccessful() && response.body() !=
null) {
                    answer = response.body().getText();
                    sdgshfh.setText(answer);

                    if (isValidFormat(answer)) {
                        sdgshfh.setText(answer+"123");
                        parseAndDisplay(answer);
```

```
                    } else {
                        try {
                            Log.e("Running","Running");
                            Thread.sleep(4000);
                            askGptQuestion(); // Retry
                        } catch (InterruptedException e) {
                            throw new RuntimeException(e);
                        }

                    }
                } else {
                    sdgshfh.setText("Request failed: Code " +
response.code() + " - " + response.message());
                }
            }


            @Override
            public void onFailure(Call<CohereResponse> call,
Throwable t) {
                sdgshfh.setText("Request error: " + t.getMessage());
            }
        });
    }



    private void displayResults() {
        StringBuilder result = new StringBuilder();

        // Display daily macros
        result.append("Daily Nutrition:\n");
        result.append(String.format("Protein: %dg\n",
dailyMacros.get("Protein")));
        result.append(String.format("Carbs: %dg\n",
dailyMacros.get("Carbs")));
        result.append(String.format("Fat: %dg\n\n",
dailyMacros.get("Fat")));

        // Clear previous charts
        chartsContainer.removeAllViews();

        // Correct meal order
        String[] mealOrder = {"Breakfast", "Lunch", "Dinner"};

        for (String mealName : mealOrder) {
            Map<String, Integer> foods = mealFoods.get(mealName);
            if (foods == null) continue; // Skip if this meal isn't
present

            // Create a TextView for meal title
            TextView mealTitle = new TextView(this);
            mealTitle.setText(mealName);
            mealTitle.setTextSize(36);
            mealTitle.setTextColor(Color.WHITE);
            chartsContainer.addView(mealTitle);

            // Create pie chart
            PieChart pieChart = new PieChart(this);
            LinearLayout.LayoutParams params = new
```

```
LinearLayout.LayoutParams(
                    LinearLayout.LayoutParams.MATCH_PARENT,
                    600
            );
            params.setMargins(0, 10, 0, 30);
            pieChart.setLayoutParams(params);

            // Prepare data for pie chart
            List<PieEntry> entries = new ArrayList<>();
            List<Integer> colors = new ArrayList<>();

            for (Map.Entry<String, Integer> foodEntry :
foods.entrySet()) {
                String foodName = foodEntry.getKey().replace(",",
"");
                entries.add(new PieEntry(foodEntry.getValue(),
foodName));
                colors.add(getRandomColor());
            }

            PieDataSet dataSet = new PieDataSet(entries, "");
            dataSet.setColors(colors);
            dataSet.setValueTextSize(12f);
            dataSet.setValueTextColor(Color.WHITE);

            PieData pieData = new PieData(dataSet);
            pieChart.setData(pieData);
            pieChart.getDescription().setEnabled(false);
            pieChart.setEntryLabelColor(Color.WHITE);
            pieChart.setEntryLabelTextSize(12f);
            pieChart.setDrawEntryLabels(true);
            pieChart.setUsePercentValues(false);
            pieChart.setDrawHoleEnabled(true);
            pieChart.setHoleColor(Color.TRANSPARENT);
            pieChart.setTransparentCircleRadius(40f);
            pieChart.animateY(1000);
            pieChart.getLegend().setTextColor(Color.WHITE);
            chartsContainer.addView(pieChart);
        }

        SaveButton.setEnabled(true);
    }

    private void parseAndDisplay(String response) {
        mealFoods = new LinkedHashMap<>(); // 🧹 Reset mealFoods
with order
        mealFoods.clear();
        chartsContainer.removeAllViews();
        mealtext = "";  // 🧹 Clear old mealtext too

        Pattern mealPattern = Pattern.compile(
                "\\((Breakfast|Lunch|Dinner):" +

"([^()]+?\\(\\d+g\\)),([^()]+?\\(\\d+g\\)),([^()]+?\\(\\d+g\\)\\))" ,
// 第一个食物        // 第三个食物
                Pattern.DOTALL
        );
        Matcher matcher = mealPattern.matcher(response);
```

```
        while (matcher.find()) {

            String mealName = matcher.group(1);
            mealtext += matcher.group(0) + "\n\n";  // Add two
newlines for better separation

            String foodItems = matcher.group(2) + "," +
matcher.group(3) + "," + matcher.group(4);

            Log.d("MealDebug", "Meal: " + mealName + "\nFood Items: "
+ foodItems);


            Map<String, Integer> foodMap = new HashMap<>();


            Pattern foodPattern =
Pattern.compile("([^()\\n]+?)\\s*\\((\\d+)g\\)");
            Matcher foodMatcher = foodPattern.matcher(foodItems);

            while (foodMatcher.find()) {
                String foodName =
foodMatcher.group(1).replaceAll("[^\\p{L}\\p{N}]", "").trim(); // 只
保留字母和数字
                int grams = Integer.parseInt(foodMatcher.group(2));
                foodMap.put(foodName, grams);
            }


            mealFoods.put(mealName, foodMap);
        }
        Log.e("mealFoods",mealFoods.toString());

        StringBuilder prettyText = new StringBuilder();

        String[] mealOrder = {"Breakfast", "Lunch", "Dinner"};
        for (String mealName : mealOrder) {
            Map<String, Integer> foods = mealFoods.get(mealName);
            if (foods == null) continue;

            prettyText.append(mealName).append(":\n");

            for (Map.Entry<String, Integer> entry : foods.entrySet())
{
                prettyText.append("- ")
                        .append(entry.getKey())
                        .append(": ")
                        .append(entry.getValue())
                        .append("g\n");
            }

            prettyText.append("\n"); // Extra line between meals
        }

        // Finally show it
        sdgshfh.setText(prettyText.toString());

        displayResults();
    }
```

```
    private int getRandomColor() {
        // Generate random RGB values for a color
        int red = (int) (Math.random() * 256);
        int green = (int) (Math.random() * 256);
        int blue = (int) (Math.random() * 256);

        // Return the color in RGB format
        return Color.rgb(red, green, blue);
    }
    private boolean isValidFormat(String response) {
        Pattern validPattern = Pattern.compile(

"\\(Breakfast: .*\\(\\d+g\\), .*\\(\\d+g\\), .*\\(\\d+g\\)\\)\\s*" +

"\\(Lunch: .*\\(\\d+g\\), .*\\(\\d+g\\), .*\\(\\d+g\\)\\)\\s*" +

"\\(Dinner: .*\\(\\d+g\\), .*\\(\\d+g\\), .*\\(\\d+g\\)\\)",
                Pattern.DOTALL
        );
        return validPattern.matcher(response).find();
    }

    private void SendtoDatabase() {
        String url = "http://10.0.2.2/nutrition/InsertPlan.php";
        Log.e("Connection123", "Start");

        RequestQueue queue = Volley.newRequestQueue(this);
        Log.e("Connection123", "Start1");
        StringRequest stringRequest = new
StringRequest(Request.Method.POST, url,
                response -> {
                    Log.e("Connection123", "Response: " + response);
                    Intent intent = new
Intent(SuggestPlanActivity.this, HomeActivity.class);
                    startActivity(intent);
                },
                error -> {
                    Log.e("Connection123", "Volley Error: " +
error.toString());
                }) {
            @Override
            protected Map<String, String> getParams() {
                Log.e("Connection123", "Start2");
                Map<String, String> params = new HashMap<>();
                params.put("plan_name", planName);
                params.put("plan_data", mealFoods.toString());
                params.put("user_id", String.valueOf(userId));
                params.put("bmr", String.valueOf(bmr));
                params.put("proteinNeeded",
String.valueOf(proteinNeeded));
                params.put("carbsNeeded",
String.valueOf(carbsNeeded));
                params.put("fatNeeded", String.valueOf(fatNeeded));
                params.put("target", String.valueOf(target));

                return params;
            }
```

```
            @Override
            public String getBodyContentType() {
                return "application/x-www-form-urlencoded;
charset=UTF-8";
            }
        };

        queue.add(stringRequest);
    }



    private void changeTextSize(float size) {

        TextView[] textViews = new TextView[] {
                findViewById(R.id.textView),
                findViewById(R.id.textView2),
                findViewById(R.id.textView6),
                findViewById(R.id.protein_text),
                findViewById(R.id.carbonhydrate_text),
                findViewById(R.id.Fat_text),
                findViewById(R.id.textView),
                findViewById(R.id.textView5),
                findViewById(R.id.textView7),
                findViewById(R.id.textView8),
                findViewById(R.id.textView3),
                findViewById(R.id.textView9),
                findViewById(R.id.textView10),
                findViewById(R.id.textView11),
                findViewById(R.id.textView12),
                findViewById(R.id.textView13),
                findViewById(R.id.textView14),
                findViewById(R.id.questionText),
                findViewById(R.id.totalquestionText),
                findViewById(R.id.planNameText),
                findViewById(R.id.sdgshfh),
                findViewById(R.id.asdfsaf),

                // Add more TextViews here as needed
        };


        Button[] buttons = new Button[] {
                //findViewById(R.id.Link_Suggest),
                //findViewById(R.id.Link_Personalized),
                //findViewById(R.id.Link_Shuffle),
                //findViewById(R.id.resultBMI),
                findViewById(R.id.GenerateButton),
                findViewById(R.id.SaveButton),
                findViewById(R.id.btnProfile),
                findViewById(R.id.btnPlanList),
                findViewById(R.id.textButton),
                findViewById(R.id.btnLogout),
                findViewById(R.id.nextButton),
                findViewById(R.id.viewButton),
                findViewById(R.id.deleteButton),
                findViewById(R.id.GenerateButton1),
                findViewById(R.id.backbutton1),

                // Add more Buttons here as needed
```

```
        };


        for (TextView textView : textViews) {
            if (textView != null) {
                textView.setTextSize(size);
            }
        }


        for (Button button : buttons) {
            if (button != null) {
                button.setTextSize(size);
            }
        }
    }
}
```

TrackingActivity.java

```
package com.example.bmi;

import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.github.mikephil.charting.charts.LineChart;
import com.github.mikephil.charting.components.Legend;
import com.github.mikephil.charting.components.LimitLine;
import com.github.mikephil.charting.components.XAxis;
import com.github.mikephil.charting.components.YAxis;
import com.github.mikephil.charting.data.Entry;
import com.github.mikephil.charting.data.LineData;
import com.github.mikephil.charting.data.LineDataSet;
import
com.google.android.material.bottomnavigation.BottomNavigationView;

import org.json.JSONArray;
import org.json.JSONObject;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.Locale;
import java.util.List;
import java.util.concurrent.TimeUnit;
```

```
import okhttp3.Call;
import okhttp3.Callback;
import okhttp3.FormBody;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;
import java.util.Calendar;

public class TrackingActivity extends AppCompatActivity {

    // --- UI ---
    private LineChart chart;
    private TextView changeSummary, changeSub, tipsText,
obesityAlert, btnViewAll;
    private LinearLayout recentContainer;
    private EditText etWeight;
    private Button btnSave;

    // --- State ---
    private double heightCm = 0;
    private double targetBMI = 0;
    private int userId;

    private final List<WeightPoint> history = new ArrayList<>();
    private final OkHttpClient client = new OkHttpClient();

    // --- Endpoints ---
    private static final String LOGIN_URL    =
"http://10.0.2.2/nutrition/login.php";
    private static final String TRACKING_URL =
"http://10.0.2.2/nutrition/tracking.php";

    @Override
    protected void onCreate(Bundle b) {
        super.onCreate(b);
        setContentView(R.layout.activity_tracking);

        // Views
        chart           = findViewById(R.id.weightChart);
        changeSummary   = findViewById(R.id.weightChangeSummary);
        changeSub       = findViewById(R.id.weightChangeSub);
        recentContainer = findViewById(R.id.recentContainer);
        tipsText        = findViewById(R.id.tipsText);
        obesityAlert    = findViewById(R.id.obesityAlert);
        etWeight        = findViewById(R.id.etWeight);
        btnSave         = findViewById(R.id.btnSaveWeight);
        btnViewAll      = findViewById(R.id.btnViewAll);

        // Bottom nav
        BottomNavigationView bottomNav =
findViewById(R.id.bottom_navigation);
        if (bottomNav != null) {
            bottomNav.setSelectedItemId(R.id.nav_tracking);
            bottomNav.setOnItemSelectedListener(item -> {
                int id = item.getItemId();
                if (id == R.id.nav_home)     { startActivity(new
Intent(this, HomeActivity.class));     return true; }
                if (id == R.id.nav_tracking) { return true; }
                if (id == R.id.nav_diet)     { startActivity(new
```

```
Intent(this, DietaryActivity.class));  return true; }
                if (id == R.id.nav_voucher)  { startActivity(new
Intent(this, VoucherActivity.class));  return true; }
                if (id == R.id.nav_settings) { startActivity(new
Intent(this, SettingsActivity.class)); return true; }
                return false;
            });
        }

        TextView btnViewAll = findViewById(R.id.btnViewAll);
        btnViewAll.setOnClickListener(v ->
                startActivity(new Intent(this,
WeightHistoryActivity.class)));

        SharedPreferences prefs = getSharedPreferences("user_prefs",
MODE_PRIVATE);
        userId = prefs.getInt("user_id", -1);

        setupChart();
        fetchUserBasicsThenData();

        btnSave.setOnClickListener(v -> {
            String txt = etWeight.getText().toString().trim();
            if (txt.isEmpty()) { Toast.makeText(this, "Enter weight
(kg)", Toast.LENGTH_SHORT).show(); return; }
            double w;
            try { w = Double.parseDouble(txt); } catch (Exception e)
{ w = 0; }
            if (w <= 0) { Toast.makeText(this, "Invalid weight",
Toast.LENGTH_SHORT).show(); return; }
            addWeight(w);
        });
    }


    private void setupChart() {
        chart.getDescription().setEnabled(false);
        chart.setTouchEnabled(false);

        XAxis x = chart.getXAxis();
        x.setDrawGridLines(false);
        x.setPosition(XAxis.XAxisPosition.BOTTOM);
        x.setGranularity(1f);

        chart.getAxisRight().setEnabled(false);

        YAxis y = chart.getAxisLeft();
        y.setDrawGridLines(true);

        Legend l = chart.getLegend();
        l.setEnabled(false);
    }

    private void drawChart() {
        List<Entry> entries = new ArrayList<>();
        for (int i = 0; i < history.size(); i++) {
            entries.add(new Entry(i, (float) history.get(i).weight));
        }

        LineDataSet ds = new LineDataSet(entries, "Weight");
```

```
        ds.setLineWidth(2.2f);
        ds.setDrawCircles(false);
        ds.setMode(LineDataSet.Mode.CUBIC_BEZIER);
        ds.setDrawValues(false);

        chart.setData(new LineData(ds));

        chart.getAxisLeft().removeAllLimitLines();
        if (heightCm > 0 && targetBMI > 0) {
            double targetW = targetBMI * Math.pow(heightCm / 100.0,
2.0);
            LimitLine ll = new LimitLine((float) targetW, "Target");
            ll.setLineWidth(1.5f);
            chart.getAxisLeft().addLimitLine(ll);
        }

        chart.invalidate();
    }

//    private void fillRecent() {
//        recentContainer.removeAllViews();
//
//        LayoutInflater inflater = LayoutInflater.from(this);
//        int shown = 0;
//
//        // newest first, max 3
//        for (int i = history.size() - 1; i >= 0 && shown < 3; i--,
shown++) {
//            WeightPoint w = history.get(i);
//            double delta = 0;
//            if (i < history.size() - 1) {
//                delta = w.weight - history.get(i + 1).weight;
//            }
//
//            View row =
inflater.inflate(R.layout.item_recent_change, recentContainer,
false);
//
//            TextView tvWeight = row.findViewById(R.id.tvWeight);
//            TextView tvWhen   = row.findViewById(R.id.tvWhen);
//            TextView tvDelta  = row.findViewById(R.id.tvDelta);
//
//            tvWeight.setText(String.format(Locale.getDefault(),
"%.1f kg", w.weight));
//            tvWhen.setText(relativeTime(w.time));
//            tvDelta.setText(String.format(Locale.getDefault(),
"%+.1f kg", delta));
//            tvDelta.setTextColor(delta < 0 ?
Color.parseColor("#2E7D32")  // green for loss
//                    : Color.parseColor("#D32F2F")); // red for gain
//
//            recentContainer.addView(row);
//        }
//    }

    private void fillRecent() {
        recentContainer.removeAllViews();

        LayoutInflater inflater = LayoutInflater.from(this);
        int shown = 0;
```

```
        // Newest first, max 3
        for (int i = history.size() - 1; i >= 0 && shown < 3; i--,
shown++) {
            WeightPoint w = history.get(i);
            double delta = 0.0;

            // Delta is the difference from the previous day in the
display order
            if (i > 0) { // If there is a previous day
                delta = w.weight - history.get(i - 1).weight; //
Current day vs. previous day
            }

            View row = inflater.inflate(R.layout.item_recent_change,
recentContainer, false);

            TextView tvWeight = row.findViewById(R.id.tvWeight);
            TextView tvWhen   = row.findViewById(R.id.tvWhen);
            TextView tvDelta  = row.findViewById(R.id.tvDelta);

            tvWeight.setText(String.format(Locale.getDefault(), "%.1f
kg", w.weight));
            tvWhen.setText(relativeTime(w.time));
            if (delta != 0.0) {
                tvDelta.setText(String.format(Locale.getDefault(),
"%+.1f kg", delta));
                tvDelta.setTextColor(delta < 0 ?
Color.parseColor("#2E7D32") : Color.parseColor("#D32F2F"));
            } else {
                tvDelta.setText(""); // No delta for the initial
weight or if no previous weight
            }

            recentContainer.addView(row);
        }
    }

    private void computeChangeAndTips() {
        if (history.isEmpty()) {
            changeSummary.setText("--");
            changeSub.setText("No data yet");
            tipsText.setText("Log your first weight to start
tracking.");
            obesityAlert.setVisibility(View.GONE);
            return;
        }

        // 30-day change
        long past30 = System.currentTimeMillis() - 30L * 24 * 60 * 60
* 1000;

        // ensure oldest->newest
        Collections.sort(history, Comparator.comparingLong(p ->
p.time));

        double first = history.get(0).weight;
        double last = history.get(history.size() - 1).weight;
        for (WeightPoint p : history) {
            if (p.time >= past30) { first = p.weight; break; }
```

```
        }
        double change = last - first;

        changeSummary.setText(String.format(Locale.getDefault(),
"%+.1f kg", change));
        changeSub.setText("Last 30 days");

        // BMI alert
        if (heightCm > 0) {
            double bmi = last / Math.pow(heightCm / 100.0, 2.0);
            if (bmi >= 30) {
                obesityAlert.setText("Obesity risk: consider reducing
fast food and increasing veggies.");
                obesityAlert.setVisibility(View.VISIBLE);
            } else {
                obesityAlert.setVisibility(View.GONE);
            }
        }

        // Tips
        if (change < -0.5) {
            tipsText.setText("Great job on your progress! Consistency
is key. Keep up the healthy habits and stay focused on your goals.");
        } else if (change > 0.5) {
            tipsText.setText("Weight is trending up. Review sugary
and fried foods, add light activity.");
        } else {
            tipsText.setText("Stable! Maintain balanced meals and
regular movement.");
        }
    }


    private void fetchUserBasicsThenData() {
        RequestBody body = new FormBody.Builder()
                .add("action", "get_user_info")
                .add("userid", String.valueOf(userId))
                .build();
        Request req = new Request.Builder()
                .url(LOGIN_URL)
                .post(body)
                .build();

        client.newCall(req).enqueue(new Callback() {
            @Override public void onFailure(Call call,
java.io.IOException e) {
                runOnUiThread(() ->
                        Toast.makeText(TrackingActivity.this, "Failed
to load user", Toast.LENGTH_SHORT).show()
                );
                fetchHistory();
            }

            @Override public void onResponse(Call call, Response
response) throws java.io.IOException {
                String json = response.body().string();
                try {
                    JSONObject o = new JSONObject(json);
                    if ("success".equals(o.optString("status"))) {
                        JSONObject u = o.getJSONObject("user");
```

```
                        heightCm = u.optDouble("height", 0);
                        targetBMI = u.optDouble("target_BMI", 0);
                    }
                } catch (Exception ignore) {}
                fetchHistory();
            }
        });
    }

    private void fetchHistory() {
        RequestBody body = new FormBody.Builder()
                .add("action", "get_weight_history")
                .add("userid", String.valueOf(userId))
                .add("days", "120")
                .build();

        Request req = new Request.Builder()
                .url(TRACKING_URL)
                .post(body)
                .build();

        client.newCall(req).enqueue(new Callback() {
            @Override public void onFailure(Call call,
java.io.IOException e) {
                runOnUiThread(() ->
                        Toast.makeText(TrackingActivity.this, "Failed
to load history", Toast.LENGTH_SHORT).show()
                );
            }

            @Override public void onResponse(Call call, Response res)
throws java.io.IOException {
                String json = res.body().string();
                try {
                    JSONObject o = new JSONObject(json);
                    history.clear();
                    if ("success".equals(o.optString("status"))) {
                        JSONArray arr = o.optJSONArray("items");
                        if (arr == null) arr =
o.optJSONArray("history");
                        if (arr != null) {
                            for (int i = 0; i < arr.length(); i++) {
                                JSONObject it = arr.getJSONObject(i);
                                double w = it.has("w") ?
it.optDouble("w") : it.optDouble("weight");
                                String ts = it.has("t") ?
it.optString("t") : it.optString("recorded_at");
                                history.add(new WeightPoint(w,
parseDateToMillis(ts)));
                            }
                        }
                        // oldest -> newest
                        Collections.sort(history,
Comparator.comparingLong(p -> p.time));
                    }
                } catch (Exception e) { /* ignore & render whatever
we have */ }

                runOnUiThread(() -> {
                    drawChart();
```

```
                    fillRecent();
                    computeChangeAndTips();
                });
            }
        });
    }

    private void addWeight(double weight) {
        RequestBody body = new FormBody.Builder()
                .add("action", "add_weight")
                .add("userid", String.valueOf(userId))
                .add("weight", String.valueOf(weight))
                .build();

        Request req = new Request.Builder()
                .url(TRACKING_URL)
                .post(body)
                .build();

        client.newCall(req).enqueue(new Callback() {
            @Override public void onFailure(Call call,
java.io.IOException e) {
                runOnUiThread(() ->
                        Toast.makeText(TrackingActivity.this, "Save
failed", Toast.LENGTH_SHORT).show()
                );
            }

            @Override public void onResponse(Call call, Response res)
throws java.io.IOException {
                String json = res.body().string();
                try {
                    JSONObject o = new JSONObject(json);
                    if (!"success".equals(o.optString("status"))) {
                        runOnUiThread(() ->
                                Toast.makeText(TrackingActivity.this,
"Server rejected", Toast.LENGTH_SHORT).show()
                        );
                    } else {
                        runOnUiThread(() -> {
                            etWeight.setText("");
                            Toast.makeText(TrackingActivity.this,
"Saved", Toast.LENGTH_SHORT).show();
                        });
                        fetchHistory();
                    }
                } catch (Exception e) {
                    runOnUiThread(() ->
                            Toast.makeText(TrackingActivity.this,
"Bad server response", Toast.LENGTH_SHORT).show()
                    );
                }
            }
        });
    }


    private long parseDateToMillis(String s) {
        String[] patterns = {"yyyy-MM-dd HH:mm:ss", "yyyy-MM-dd"};
        for (String p : patterns) {
```

```
            try {
                return new SimpleDateFormat(p,
Locale.getDefault()).parse(s).getTime();
            } catch (Exception ignore) {}
        }
        return System.currentTimeMillis();
    }

    private String relativeTime(long millis) {
        long d = dayDiffLocal(millis);
        if (d == 0) return "Today";
        if (d == 1) return "Yesterday";
        if (d < 7)  return d + " Days Ago";
        if (d < 30) return (d / 7) + " Weeks Ago";
        return (d / 30) + " Months ago";
    }

    private long dayDiffLocal(long millis) {
        return (midnight(System.currentTimeMillis()) -
midnight(millis)) / 86_400_000L; // 86400000
    }

    private long midnight(long millis) {
        Calendar c = Calendar.getInstance();           // user's
local timezone
        c.setTimeInMillis(millis);
        c.set(Calendar.HOUR_OF_DAY, 0);
        c.set(Calendar.MINUTE, 0);
        c.set(Calendar.SECOND, 0);
        c.set(Calendar.MILLISECOND, 0);
        return c.getTimeInMillis();
    }

    static class WeightPoint {
        final double weight;
        final long time;
        WeightPoint(double w, long t) { weight = w; time = t; }
    }
}
```

ViewEditPlanActivity.java

```
package com.example.bmi;

import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.LinearLayout;
import android.widget.Spinner;
import android.widget.SpinnerAdapter;
import android.widget.TextView;
```

```
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import com.github.lzyzsd.circleprogress.DonutProgress;
import com.github.mikephil.charting.charts.PieChart;
import com.github.mikephil.charting.data.PieData;
import com.github.mikephil.charting.data.PieDataSet;
import com.github.mikephil.charting.data.PieEntry;

import org.json.JSONException;
import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class ViewEditPlanActivity extends AppCompatActivity {
    private String planData;
    private double weight = 90;
    private int height = 180;
    private String gender = "Male";
    private int age = 21;
    private double bmr = 0;
    private String target="Diet";
    private DonutProgress bmr_progressbar;
    private DonutProgress protein_progressbar;
    private DonutProgress carbonhydrate_progressbar;
    private DonutProgress fat_progressbar;
    private int proteinRate;
    private int carbsRate;
    private int fatRate;
    private double proteinNeeded;
    private double carbsNeeded;
    private double fatNeeded;
    private double proteinEveryMeal;
    private double carbsEveryMeal;
    private double fatEveryMeal;
    private double targetCalories;
    private String planName;
    private LinearLayout chartsContainer;
    private String answer;
    private int userId;
    private String function;
    private Button GenerateButton;
    private Button SaveButton;
    private Spinner spinnerProtein1;
    private Spinner spinnerCarbs1;
    private Spinner spinnerFat1;
    private Spinner spinnerProtein2;
```

```
    private Spinner spinnerCarbs2;
    private Spinner spinnerFat2;
    private Spinner spinnerProtein3;
    private Spinner spinnerCarbs3;
    private Spinner spinnerFat3;



    private Map<String, Integer> dailyMacros = new HashMap<>();
    private Map<String, Map<String, Integer>> mealFoods = new
HashMap<>();

    private ArrayList<String> oriproteinFoods = new ArrayList<>();
    private ArrayList<String> oricarbsFoods = new ArrayList<>();

    private ArrayList<String> orifatFoods = new ArrayList<>();

    private ArrayList<String> proteinFoods = new ArrayList<>();
    private ArrayList<Double> proteinFoodsInGram = new ArrayList<>();
    private ArrayList<String> carbsFoods = new ArrayList<>();
    private ArrayList<Double> carbsFoodsInGram = new ArrayList<>();
    private ArrayList<String> fatFoods = new ArrayList<>();
    private ArrayList<Double> fatFoodsInGram = new ArrayList<>();
    private String allergy;
    private int planID;
    private EditText PlanNameText;
    private Button backButton;
    private TextView CurrentFoodList;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.vieweditplan);

        planID=getIntent().getIntExtra("plan_id",8);
        loadDataFromDatabase();
        SharedPreferences sharedPreferences =
getSharedPreferences("user_prefs", MODE_PRIVATE);

allergy=sharedPreferences.getString("allergies","[Eggs,Meats]");

loadFoodsFromFile(R.raw.protein,proteinFoods,proteinFoodsInGram);
        loadFoodsFromFile(R.raw.carbs,carbsFoods,carbsFoodsInGram);
        loadFoodsFromFile(R.raw.fat,fatFoods,fatFoodsInGram);
        Intent intent = new Intent();
        PlanNameText=findViewById(R.id.PlanName);
        CurrentFoodList=findViewById(R.id.CurrentFoodList);
        spinnerProtein1=findViewById(R.id.spinnerProtein1);
        spinnerCarbs1=findViewById(R.id.spinnerCarbs1);
        spinnerFat1=findViewById(R.id.spinnerFat1);
        spinnerProtein2=findViewById(R.id.spinnerProtein2);
        spinnerCarbs2=findViewById(R.id.spinnerCarbs2);
        spinnerFat2=findViewById(R.id.spinnerFat2);
        spinnerProtein3=findViewById(R.id.spinnerProtein3);
        spinnerCarbs3=findViewById(R.id.spinnerCarbs3);
        spinnerFat3=findViewById(R.id.spinnerFat3);

        GenerateButton=findViewById(R.id.GenerateButton);
        GenerateButton.setOnClickListener(new View.OnClickListener()
```

```
{
            @Override
            public void onClick(View v) {
                parseData();
                SaveButton.setEnabled(true);
            }
        });

        backButton=findViewById(R.id.backbutton1);
        backButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(ViewEditPlanActivity.this,
PlanListActivity.class);
                startActivity(intent);

            }
        });
        SaveButton=findViewById(R.id.SaveButton);;
        SaveButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                SendtoDatabase();
            }
        });

        SaveButton.setEnabled(false);
        // 读取 user_id
        userId = sharedPreferences.getInt("user_id", 2);
        function = sharedPreferences.getString("function",
"Suggest");
        height = sharedPreferences.getInt("height", 180);
        gender = sharedPreferences.getString("gender", "Male");
        weight = sharedPreferences.getFloat("weight", 78);
        age = sharedPreferences.getInt("age", 22);
        int size=sharedPreferences.getInt("text_size",12);
        changeTextSize(size);


        protein_progressbar = findViewById(R.id.protein_progressbar);
        carbonhydrate_progressbar =
findViewById(R.id.carbonhydrate_progressbar);
        fat_progressbar = findViewById(R.id.fat_progressbar);
        bmr_progressbar = findViewById(R.id.bmr_progressbar);
        chartsContainer = findViewById(R.id.piechartLinear);



        Log.e("Shuffle",function);
        Log.e("Shuffle", String.valueOf(bmr));
        Log.e("Shuffle", String.valueOf(gender));
        Log.e("Shuffle",String.valueOf(height));
        Log.e("Shuffle", String.valueOf(weight));
        Log.e("Shuffle", String.valueOf(age));
        Log.e("Shuffle", String.valueOf(targetCalories));

        Log.e("Plan123", "carbsEveryMeal: " + proteinEveryMeal);
        Log.e("Plan123", "carbsEveryMeal: " + carbsEveryMeal);
        Log.e("Plan123", "carbsEveryMeal: " + fatEveryMeal);
```

```
    }

    private void loadDataFromDatabase() {
        String url = "http://10.0.2.2/nutrition/GetPlanData.php";
        Log.e("Connection123", "Start");

        RequestQueue queue = Volley.newRequestQueue(this);
        Log.e("Connection123", "Start1");

        StringRequest stringRequest = new
StringRequest(Request.Method.POST, url,
                response -> {
                    Log.e("Connection123", "Response: " + response);

                    try {
                        JSONObject jsonObject = new
JSONObject(response);
                        if
(jsonObject.getString("status").equals("success")) {
                            JSONObject data =
jsonObject.getJSONObject("data");

                             planName = data.getString("plan_name");
                             planData = data.getString("plan_data");
                             bmr = data.getDouble("BMR");
                             proteinNeeded =
data.getDouble("ProteinNeeded");
                             carbsNeeded =
data.getDouble("CarbsNeeded");
                             fatNeeded = data.getDouble("FatNeeded");
                            target=data.getString("target");
                            // 打印或使用数据
                            Log.e("Plan123", "Name: " + planName);
                            Log.e("Plan123", "Data: " + planData);
                            Log.e("Plan123", "BMR: " + bmr);
                            Log.e("Plan123", "Protein: " +
proteinNeeded);
                            Log.e("Plan123", "Carbs: " +
carbsNeeded);
                            Log.e("Plan123", "Fat: " + fatNeeded);
                            Log.e("Plan123", "target: " + target);
                            PlanNameText.setText(planName);
                            bmr_progressbar.setMax((int) bmr);
                            bmr_progressbar.setProgress((int) bmr);
                            bmr_progressbar.setText((int) bmr +
"kcal");
                            protein_progressbar.setMax((int)
proteinNeeded);
                            protein_progressbar.setProgress((int)
proteinNeeded);

protein_progressbar.setText(String.valueOf(proteinNeeded));

                            carbonhydrate_progressbar.setMax((int)
carbsNeeded);

carbonhydrate_progressbar.setProgress((int) carbsNeeded);

carbonhydrate_progressbar.setText(String.valueOf(carbsNeeded));
```

```
                            fat_progressbar.setMax((int) fatNeeded);
                            fat_progressbar.setProgress((int)
fatNeeded);

fat_progressbar.setText(String.valueOf(fatNeeded));

                            showPlanDataWithGrams(planData);
                            proteinEveryMeal=proteinNeeded/3;
                            carbsEveryMeal=carbsNeeded/3;
                            fatEveryMeal=fatNeeded/3;
                            setupSpinner(spinnerProtein1,
proteinFoods);
                            setupSpinner(spinnerCarbs1, carbsFoods);
                            setupSpinner(spinnerFat1, fatFoods);
                            setupSpinner(spinnerProtein2,
proteinFoods);
                            setupSpinner(spinnerCarbs2, carbsFoods);
                            setupSpinner(spinnerFat2, fatFoods);
                            setupSpinner(spinnerProtein3,
proteinFoods);
                            setupSpinner(spinnerCarbs3, carbsFoods);
                            setupSpinner(spinnerFat3, fatFoods);
                            setupSpinner();

                        } else {
                            Log.e("Connection123", "Error in
response: " + jsonObject.getString("message"));
                        }
                    } catch (JSONException e) {
                        e.printStackTrace();
                        Log.e("Connection123", "JSON Error: " +
e.getMessage());
                    }

                },
                error -> {
                    Log.e("Connection123", "Volley Error: " +
error.toString());
                }) {

            @Override
            protected Map<String, String> getParams() {
                Log.e("Connection123", "Start2");
                Map<String, String> params = new HashMap<>();
                params.put("planID", String.valueOf(planID));
                return params;
            }

            @Override
            public String getBodyContentType() {
                return "application/x-www-form-urlencoded;
charset=UTF-8";
            }
        };

        queue.add(stringRequest);
    }


    // 1. First, function to parse planData String into a Map
```

```
    private Map<String, Map<String, Integer>>
parsePlanDataWithGrams(String planDataString) {
        Map<String, Map<String, Integer>> result = new HashMap<>();

        if (planDataString == null || planDataString.length() < 2 ||
                !planDataString.startsWith("{")
|| !planDataString.endsWith("}")) {
            Log.e("ParsePlan", "Invalid planDataString: " +
planDataString);
            return result;
        }

        // Remove outer braces
        String content = planDataString.substring(1,
planDataString.length() - 1).trim();

        List<String> mealEntries = new ArrayList<>();
        int braceCount = 0;
        StringBuilder current = new StringBuilder();

        for (char c : content.toCharArray()) {
            if (c == '{') braceCount++;
            if (c == '}') braceCount--;

            current.append(c);

            if (braceCount == 0 && c == '}') {
                mealEntries.add(current.toString().trim());
                current.setLength(0); // Clear current
            }
        }

        for (String meal : mealEntries) {
            int eqIndex = meal.indexOf('=');
            int braceStart = meal.indexOf('{');
            int braceEnd = meal.lastIndexOf('}');

            if (eqIndex == -1 || braceStart == -1 || braceEnd == -1)
continue;

            String mealName = meal.substring(0,
eqIndex).trim().replaceAll("[,\\s]", "");
            String mealContent = meal.substring(braceStart + 1,
braceEnd);

            Map<String, Integer> foodMap = new HashMap<>();
            String[] foodEntries = mealContent.split(",\\s*");

            for (String foodEntry : foodEntries) {
                String[] parts = foodEntry.split("=");
                if (parts.length == 2) {
                    String foodName = parts[0].trim();
                    int grams = Integer.parseInt(parts[1].trim());
                    foodMap.put(foodName, grams);
                }
            }

            result.put(mealName, foodMap);
        }
```

```
        return result;
    }

    // 2. Then, code to display it nicely with grams
    private void showPlanDataWithGrams(String planDataString) {
        Map<String, Map<String, Integer>> parsedPlanData =
parsePlanDataWithGrams(planDataString);

        StringBuilder prettyText = new StringBuilder();
        String[] mealOrder = {"Breakfast", "Lunch", "Dinner"};
        for (String mealName : mealOrder) {
            Map<String, Integer> foods =
parsedPlanData.get(mealName);
            if (foods == null) continue;

            prettyText.append(mealName).append(":\n");

            for (Map.Entry<String, Integer> entry : foods.entrySet())
{
                prettyText.append("- ")
                        .append(entry.getKey())
                        .append(": ")
                        .append(entry.getValue())
                        .append("g\n");
            }

            prettyText.append("\n"); // Extra line between meals
        }

        // Finally show on the screen
        CurrentFoodList.setText(prettyText.toString());
    }

    private void parseData() {

        String breakfastProtein =
spinnerProtein1.getSelectedItem().toString();
        String breakfastCarbs =
spinnerCarbs1.getSelectedItem().toString();
        String breakfastFat =
spinnerFat1.getSelectedItem().toString();

        String lunchProtein =
spinnerProtein2.getSelectedItem().toString();
        String lunchCarbs =
spinnerCarbs2.getSelectedItem().toString();
        String lunchFat = spinnerFat2.getSelectedItem().toString();

        String dinnerProtein =
spinnerProtein3.getSelectedItem().toString();
        String dinnerCarbs =
spinnerCarbs3.getSelectedItem().toString();
        String dinnerFat = spinnerFat3.getSelectedItem().toString();


        Map<String, Integer> breakfast = new HashMap<>();
        Map<String, Integer> lunch = new HashMap<>();
        Map<String, Integer> dinner = new HashMap<>();
```

```
        breakfast.put(breakfastProtein,
calculateGrams(breakfastProtein, proteinFoods,
proteinFoodsInGram,proteinEveryMeal));
        breakfast.put(breakfastCarbs, calculateGrams(breakfastCarbs,
carbsFoods, carbsFoodsInGram,carbsEveryMeal));
        breakfast.put(breakfastFat, calculateGrams(breakfastFat,
fatFoods, fatFoodsInGram,fatEveryMeal));

        lunch.put(lunchProtein, calculateGrams(lunchProtein,
proteinFoods, proteinFoodsInGram,proteinEveryMeal));
        lunch.put(lunchCarbs, calculateGrams(lunchCarbs, carbsFoods,
carbsFoodsInGram,carbsEveryMeal));
        lunch.put(lunchFat, calculateGrams(lunchFat, fatFoods,
fatFoodsInGram,fatEveryMeal));

        dinner.put(dinnerProtein, calculateGrams(dinnerProtein,
proteinFoods, proteinFoodsInGram,proteinEveryMeal));
        dinner.put(dinnerCarbs, calculateGrams(dinnerCarbs,
carbsFoods, carbsFoodsInGram,carbsEveryMeal));
        dinner.put(dinnerFat, calculateGrams(dinnerFat, fatFoods,
fatFoodsInGram,fatEveryMeal));


        mealFoods.put("Breakfast", breakfast);
        mealFoods.put("Lunch", lunch);
        mealFoods.put("Dinner", dinner);


        displayResults();
    }


    private int calculateGrams(String foodName, ArrayList<String>
foodList, ArrayList<Double> gramsList,Double nutritionNeeded) {
        int index = foodList.indexOf(foodName);
        if (index != -1) {
            double grams = gramsList.get(index);

            return (int) (100 * (nutritionNeeded / (float) grams));
// 如果需要根据 grams 做一些其他的计算可以在这里修改
        }
        return 0;
    }
    private void loadFoodsFromFile(int rawResId, ArrayList<String>
foodList, ArrayList<Double> gramsList) {
        try {
            InputStream inputStream =
getResources().openRawResource(rawResId);
            BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream));


            allergy = allergy.replaceAll("[\\[\\]\\s]", ""); // 去中括
号和空格
            List<String> allergyList =
Arrays.asList(allergy.split(","));

            String line;
            while ((line = reader.readLine()) != null) {
```

```
                String[] parts = line.split("-");


                if (parts.length >= 2) {
                    String foodName = parts[0].trim();
                    double grams =
Double.parseDouble(parts[1].trim());

                    boolean hasAllergy = false;

                    if (parts.length == 3 && parts[2].contains("["))
{

                        String allergiesInItem =
parts[2].replaceAll("[\\[\\]\\s]", "");
                        List<String> itemAllergies =
Arrays.asList(allergiesInItem.split(","));

                        for (String allergy : allergyList) {
                            if (itemAllergies.contains(allergy)) {
                                hasAllergy = true;
                                break;
                            }
                        }
                    }

                    if (!hasAllergy) {
                        foodList.add(foodName);
                        gramsList.add(grams);
                    }
                }
            }

            reader.close();
            inputStream.close();
        } catch (IOException e) {
            e.printStackTrace();
        }

        Log.e("List", foodList.toString());
        Log.e("List", gramsList.toString());
    }
    private void setupSpinner(Spinner spinner, ArrayList<String>
items) {
        ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_item, items);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropd
own_item);
        spinner.setAdapter(adapter);
    }
    private Map<String, ArrayList<String>> parsePlanData(String
planDataString) {
        Map<String, ArrayList<String>> result = new HashMap<>();

        if (planDataString == null || planDataString.length() < 2 ||
                !planDataString.startsWith("{")
|| !planDataString.endsWith("}")) {
            Log.e("ParsePlan", "Invalid planDataString: " +
planDataString);
```

```
            return result;
        }


        String content = planDataString.substring(1,
planDataString.length() - 1).trim();

        List<String> mealEntries = new ArrayList<>();
        int braceCount = 0;
        StringBuilder current = new StringBuilder();

        for (char c : content.toCharArray()) {
            if (c == '{') braceCount++;
            if (c == '}') braceCount--;

            current.append(c);

            if (braceCount == 0 && c == '}') {
                mealEntries.add(current.toString().trim());
                current.setLength(0);


            }
        }
        Log.e("ParsePlan", "mealEntries:"+mealEntries.toString());
        for (String meal : mealEntries) {
            int eqIndex = meal.indexOf('=');
            int braceStart = meal.indexOf('{');
            int braceEnd = meal.lastIndexOf('}');

            if (eqIndex == -1 || braceStart == -1 || braceEnd == -1)
continue;

            String mealName = meal.substring(0,
eqIndex).trim().replaceAll("[,\\s]", "");
            String mealContent = meal.substring(braceStart + 1,
braceEnd);

            ArrayList<String> foodMap = new ArrayList<>();
            String[] foodEntries = mealContent.split(",\\s*");

            for (String foodEntry : foodEntries) {
                String[] parts = foodEntry.split("=");
                if (parts.length == 2) {
                    String foodName = parts[0].trim();
                    foodMap.add(foodName);
                }
            }
            Log.e("ParsePlan", "mealName:"+mealName);
            Log.e("ParsePlan", "foodMap:"+foodMap.toString());
            result.put(mealName, foodMap);
        }

        return result;
    }

    private void setupSpinner() {


        Map<String, ArrayList<String>> planMap =
```

```
parsePlanData(planData);
        Log.e("List", planMap.toString());

        if (planMap.containsKey("Breakfast")) {
            ArrayList<String> breakfastItems = new
ArrayList<>(planMap.get("Breakfast"));
            if (breakfastItems.size() >= 3) {
                Log.e("List", "Came to
Breakfast"+breakfastItems.get(0)+breakfastItems.get(1)+breakfastItems
.get(2));
                setSpinnerSelection(spinnerProtein1,
breakfastItems.get(0));
                setSpinnerSelection(spinnerCarbs1,
breakfastItems.get(1));
                setSpinnerSelection(spinnerFat1,
breakfastItems.get(2));
            }
        }

        if (planMap.containsKey("Lunch")) {
            ArrayList<String> lunchItems = new
ArrayList<>(planMap.get("Lunch"));
            if (lunchItems.size() >= 3) {
                Log.e("List", "Came to
Lunch"+lunchItems.get(0)+lunchItems.get(1)+lunchItems.get(2));

                setSpinnerSelection(spinnerProtein2,
lunchItems.get(0));
                setSpinnerSelection(spinnerCarbs2,
lunchItems.get(1));
                setSpinnerSelection(spinnerFat2, lunchItems.get(2));
            }
        }

        if (planMap.containsKey("Dinner")) {

            ArrayList<String> dinnerItems = new
ArrayList<>(planMap.get("Dinner"));
            if (dinnerItems.size() >= 3) {
                Log.e("List", "Came to
Dinner"+dinnerItems.get(0)+dinnerItems.get(1)+dinnerItems.get(2));

                setSpinnerSelection(spinnerProtein3,
dinnerItems.get(0));
                setSpinnerSelection(spinnerCarbs3,
dinnerItems.get(1));
                setSpinnerSelection(spinnerFat3, dinnerItems.get(2));
            }
        }
    }
    private void setSpinnerSelection(Spinner spinner, String value) {
        SpinnerAdapter adapter = spinner.getAdapter();
        for (int i = 0; i < adapter.getCount(); i++) {
            if (adapter.getItem(i).toString().contains(value)) {
                spinner.setSelection(i);
                break;
            }
        }
    }
```

```
    private void displayResults() {
        StringBuilder result = new StringBuilder();

        // Display daily macros
        result.append("Daily Nutrition:\n");
        result.append(String.format("Protein: %dg\n",
dailyMacros.get("Protein")));
        result.append(String.format("Carbs: %dg\n",
dailyMacros.get("Carbs")));
        result.append(String.format("Fat: %dg\n\n",
dailyMacros.get("Fat")));

        chartsContainer.removeAllViews();

        // Define the correct meal order
        String[] mealOrder = {"Breakfast", "Lunch", "Dinner"};

        for (String mealName : mealOrder) {
            Map<String, Integer> foods = mealFoods.get(mealName);
            if (foods == null) continue; // skip if meal data missing

            // Create a TextView for meal title
            TextView mealTitle = new TextView(this);
            mealTitle.setText(mealName);
            mealTitle.setTextSize(36);
            mealTitle.setTextColor(Color.WHITE);
            chartsContainer.addView(mealTitle);

            // Create pie chart
            PieChart pieChart = new PieChart(this);
            LinearLayout.LayoutParams params = new
LinearLayout.LayoutParams(
                    LinearLayout.LayoutParams.MATCH_PARENT,
                    600
            );
            params.setMargins(0, 10, 0, 30);
            pieChart.setLayoutParams(params);

            // Prepare data
            List<PieEntry> entries = new ArrayList<>();
            List<Integer> colors = new ArrayList<>();

            for (Map.Entry<String, Integer> foodEntry :
foods.entrySet()) {
                String foodName = foodEntry.getKey().replace(",",
"");
                entries.add(new PieEntry(foodEntry.getValue(),
foodName));
                colors.add(getRandomColor());
            }

            PieDataSet dataSet = new PieDataSet(entries, "");
            dataSet.setColors(colors);
            dataSet.setValueTextSize(12f);
            dataSet.setValueTextColor(Color.WHITE);

            PieData pieData = new PieData(dataSet);
            pieChart.setData(pieData);
            pieChart.getDescription().setEnabled(false);
```

```
            pieChart.setEntryLabelColor(Color.WHITE);
            pieChart.setEntryLabelTextSize(12f);
            pieChart.setDrawEntryLabels(true);
            pieChart.setUsePercentValues(false);
            pieChart.setDrawHoleEnabled(true);
            pieChart.setHoleColor(Color.TRANSPARENT);
            pieChart.setTransparentCircleRadius(40f);
            pieChart.animateY(1000);
            pieChart.getLegend().setTextColor(Color.WHITE);
            chartsContainer.addView(pieChart);
        }

        SaveButton.setEnabled(true);
    }

    private int getRandomColor() {
        // Generate random RGB values for a color
        int red = (int) (Math.random() * 256);
        int green = (int) (Math.random() * 256);
        int blue = (int) (Math.random() * 256);

        // Return the color in RGB format
        return Color.rgb(red, green, blue);
    }


    private void SendtoDatabase() {
        String url = "http://10.0.2.2/nutrition/UpdatePlanData.php";
        Log.e("Connection123", "Start");

        RequestQueue queue = Volley.newRequestQueue(this);
        Log.e("Connection123", "Start1");
        StringRequest stringRequest = new
StringRequest(Request.Method.POST, url,
                response -> {
                    Log.e("Connection123", response);
                    Toast.makeText(this, "Updated Successfully",
Toast.LENGTH_SHORT).show();
                    Intent intent = new
Intent(ViewEditPlanActivity.this, PlanListActivity.class);
                    startActivity(intent);
                },
                error -> {
                    Log.e("Connection123", "Volley Error: " +
error.toString());
                }) {
            @Override
            protected Map<String, String> getParams() {
                Log.e("Connection123", "Start2");
                Map<String, String> params = new HashMap<>();
                params.put("planID", String.valueOf(planID));

                params.put("plan_name",
PlanNameText.getText().toString());
                params.put("plan_data", mealFoods.toString());


                return params;
            }
```

```
            @Override
            public String getBodyContentType() {
                return "application/x-www-form-urlencoded;
charset=UTF-8";
            }
        };

        queue.add(stringRequest);
    }

    private void changeTextSize(float size) {

        TextView[] textViews = new TextView[] {
                findViewById(R.id.textView),
                findViewById(R.id.textView2),
                findViewById(R.id.textView6),
                findViewById(R.id.protein_text),
                findViewById(R.id.carbonhydrate_text),
                findViewById(R.id.Fat_text),
                findViewById(R.id.textView),
                findViewById(R.id.textView5),
                findViewById(R.id.textView7),
                findViewById(R.id.textView8),
                findViewById(R.id.textView3),
                findViewById(R.id.textView9),
                findViewById(R.id.textView10),
                findViewById(R.id.textView11),
                findViewById(R.id.textView12),
                findViewById(R.id.textView13),
                findViewById(R.id.textView14),
                findViewById(R.id.questionText),
                findViewById(R.id.totalquestionText),
                findViewById(R.id.planNameText),
                findViewById(R.id.sdgshfh),
                findViewById(R.id.asdfsaf),
                // Add more TextViews here as needed
        };


        Button[] buttons = new Button[] {
                //findViewById(R.id.Link_Suggest),
                //findViewById(R.id.Link_Personalized),
                //findViewById(R.id.Link_Shuffle),
                //findViewById(R.id.resultBMI),
                findViewById(R.id.GenerateButton),
                findViewById(R.id.SaveButton),
                findViewById(R.id.btnProfile),
                findViewById(R.id.btnPlanList),
                findViewById(R.id.textButton),
                findViewById(R.id.btnLogout),
                findViewById(R.id.nextButton),
                findViewById(R.id.viewButton),
                findViewById(R.id.deleteButton),
                findViewById(R.id.GenerateButton1),
                findViewById(R.id.backbutton1),
                // Add more Buttons here as needed
        };


        for (TextView textView : textViews) {
```

```
            if (textView != null) {
                textView.setTextSize(size);
            }
        }


        for (Button button : buttons) {
            if (button != null) {
                button.setTextSize(size);
            }
        }
    }


}
```

VoucherActivity.java

```
package com.example.bmi;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import
com.google.android.material.bottomnavigation.BottomNavigationView;

import org.json.JSONObject;

import java.text.NumberFormat;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;

import okhttp3.Call;
import okhttp3.Callback;
import okhttp3.FormBody;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;
import android.graphics.Paint;

public class VoucherActivity extends AppCompatActivity {

    private TextView tvPoints;
    private LinearLayout voucherList;
```

```
    private int userId;
    private int currentPoints = 0;

    private final OkHttpClient client = new OkHttpClient();
    private static final String REWARDS_URL =
"http://10.0.2.2/nutrition/rewards.php";


    static class RewardItem {
        final String code, title, desc;
        final int cost, imageRes;

        // restaurant details (no fixed date here)
        final String restaurant;
        final String location;
        final String validityNote; // e.g. "Valid 30 days from
redemption"
        final String terms;

        RewardItem(String code, String title, String desc, int cost,
int imageRes,
                   String restaurant, String location, String
validityNote, String terms) {
            this.code = code; this.title = title; this.desc = desc;
this.cost = cost; this.imageRes = imageRes;
            this.restaurant = restaurant; this.location = location;
this.validityNote = validityNote; this.terms = terms;
        }
    }

    private List<RewardItem> rewards() {
        List<RewardItem> list = new ArrayList<>();

        list.add(new RewardItem(
                "GBOWL_10OFF",
                "10% Off Healthy Bowl",
                "Save on any salad or grain bowl.",
                600,
                R.drawable.voucher_discount,
                "GreenBowl Café",
                "Lot G-12, Mid Valley, KL",
                "Valid 30 days from redemption",
                "Min spend RM15. Dine-in only. Not valid with other
promos."
        ));

        list.add(new RewardItem(
                "FITBITE_SIDEFREE",
                "Free Side Salad",
                "Complimentary side salad with any main.",
                800,
                R.drawable.smoothie, // use any image you have
                "FitBite Salad Bar",
                "1F-23, Sunway Pyramid",
                "Valid 30 days from redemption",
                "One redemption per user. Show code at counter."
        ));

        list.add(new RewardItem(
                "FRESHNGO_15SET",
```

```
                "15% Off Healthy Set",
                "Discount on selected 'Healthy Set' combos.",
                1200,
                R.drawable.voucher_plan,
                "Fresh n' Go Grocer",
                "Selected outlets & online",
                "Valid 30 days from redemption",
                "While stocks last. Not exchangeable for cash."
        ));

        return list;
    }


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_voucher);

        TextView btnHistory = findViewById(R.id.btnHistory);
        btnHistory.setPaintFlags(btnHistory.getPaintFlags() |
Paint.UNDERLINE_TEXT_FLAG);
        btnHistory.setOnClickListener(v ->
                startActivity(new Intent(this,
RedemptionHistoryActivity.class)));


        tvPoints = findViewById(R.id.tvPoints);
        voucherList = findViewById(R.id.voucherList);

        BottomNavigationView bottomNav =
findViewById(R.id.bottom_navigation);
        if (bottomNav != null) {
            bottomNav.setSelectedItemId(R.id.nav_voucher);
            bottomNav.setOnItemSelectedListener(item -> {
                int id = item.getItemId();
                if (id == R.id.nav_home)     { startActivity(new
Intent(this, HomeActivity.class));     return true; }
                if (id == R.id.nav_tracking) { startActivity(new
Intent(this, TrackingActivity.class)); return true; }
                if (id == R.id.nav_diet)     { startActivity(new
Intent(this, DietaryActivity.class));  return true; }
                if (id == R.id.nav_voucher)  { return true; }
                if (id == R.id.nav_settings) { startActivity(new
Intent(this, SettingsActivity.class)); return true; }
                return false;
            });
        }


        SharedPreferences prefs = getSharedPreferences("user_prefs",
MODE_PRIVATE);
        userId = prefs.getInt("user_id", -1);


        renderRewards();


        fetchUserPoints();
    }
```

```
    private void renderRewards() {
        LayoutInflater inflater = LayoutInflater.from(this);
        voucherList.removeAllViews();

        for (RewardItem r : rewards()) {
            View row = inflater.inflate(R.layout.item_voucher,
voucherList, false);

            ((TextView)
row.findViewById(R.id.tvCost)).setText(formatPoints(r.cost) + "
Points");
            TextView tvTitle = row.findViewById(R.id.tvTitle);
            TextView tvDesc  = row.findViewById(R.id.tvDesc);
            ImageView img    = row.findViewById(R.id.img);

            tvTitle.setText(r.title);
            tvDesc.setText(r.desc);
            img.setImageResource(r.imageRes);

            // tap any of these to see details (until we add a
dedicated button)
            View.OnClickListener show = v -> showDetailsDialog(r);
            tvTitle.setOnClickListener(show);
            tvDesc.setOnClickListener(show);
            img.setOnClickListener(show);

            Button btn = row.findViewById(R.id.btnRedeem);
            btn.setTag(r.cost);
            btn.setOnClickListener(v -> onRedeemClicked(r));

            TextView btnDetails = row.findViewById(R.id.btnDetails);
            if (btnDetails != null) {
                btnDetails.setOnClickListener(v ->
showDetailsDialog(r));
            }

            voucherList.addView(row);
        }
        refreshRedeemButtons();
    }

    private void onRedeemClicked(RewardItem r) {
        if (currentPoints < r.cost) {
            Toast.makeText(this, "Not enough points",
Toast.LENGTH_SHORT).show();
            return;
        }

        new AlertDialog.Builder(this)
                .setTitle("Redeem \"" + r.title + "\"?")
                .setMessage("This will spend " + r.cost + " points.")
                .setPositiveButton("Redeem", (d, w) ->
redeemReward(r))
                .setNegativeButton("Cancel", null)
                .show();
    }

    private void redeemReward(RewardItem r) {
        if (userId <= 0) { Toast.makeText(this, "No user id",
```

```
Toast.LENGTH_SHORT).show(); return; }

        // Find the button we tapped to disable while loading
        Button tapped = null;
        for (int i = 0; i < voucherList.getChildCount(); i++) {
            View row = voucherList.getChildAt(i);
            TextView title = row.findViewById(R.id.tvTitle);
            if (title != null &&
r.title.equals(title.getText().toString())) {
                tapped = row.findViewById(R.id.btnRedeem);
                break;
            }
        }
        if (tapped != null) tapped.setEnabled(false);

        RequestBody body = new FormBody.Builder()
                .add("action", "redeem")
                .add("userid", String.valueOf(userId))
                .add("reward_code", r.code)
                .add("cost", String.valueOf(r.cost))
                .build();

        Request req = new
Request.Builder().url(REWARDS_URL).post(body).build();

        Button finalTapped = tapped;
        client.newCall(req).enqueue(new Callback() {
            @Override public void onFailure(Call call,
java.io.IOException e) {
                runOnUiThread(() -> {
                    if (finalTapped != null)
finalTapped.setEnabled(true);
                    Toast.makeText(VoucherActivity.this, "Redeem
failed", Toast.LENGTH_SHORT).show();
                });
            }

            @Override public void onResponse(Call call, Response res)
throws java.io.IOException {
                String json = res.body().string();
                try {
                    JSONObject o = new JSONObject(json);
                    String status = o.optString("status");
                    if ("redeemed".equalsIgnoreCase(status)) {
                        final int newPts = o.optInt("new_points",
Math.max(0, currentPoints - r.cost));
                        final String expires =
o.optString("expires_at", "");

                        runOnUiThread(() -> {
                            currentPoints = newPts;

tvPoints.setText(formatPoints(currentPoints));
                            refreshRedeemButtons();
                            if (finalTapped != null)
finalTapped.setEnabled(true);
                            String msg = expires.isEmpty()
                                    ? "Redeemed!"
                                    : "Redeemed! Expires on " +
expires;
```

```
                            Toast.makeText(VoucherActivity.this, msg,
Toast.LENGTH_LONG).show();
                        });
                    } else if
("not_enough_points".equalsIgnoreCase(status)) {
                        runOnUiThread(() -> {
                            if (finalTapped != null)
finalTapped.setEnabled(true);
                            Toast.makeText(VoucherActivity.this, "Not
enough points", Toast.LENGTH_SHORT).show();
                        });
                    } else {
                        runOnUiThread(() -> {
                            if (finalTapped != null)
finalTapped.setEnabled(true);
                            Toast.makeText(VoucherActivity.this,
"Redeem error", Toast.LENGTH_SHORT).show();
                        });
                    }
                } catch (Exception e) {
                    runOnUiThread(() -> {
                        if (finalTapped != null)
finalTapped.setEnabled(true);
                        Toast.makeText(VoucherActivity.this, "Bad
server response", Toast.LENGTH_SHORT).show();
                    });
                }
            }
        });
    }

    private void refreshRedeemButtons() {
        for (int i = 0; i < voucherList.getChildCount(); i++) {
            View row = voucherList.getChildAt(i);
            Button btn = row.findViewById(R.id.btnRedeem);

            int cost = 0;
            Object tag = btn.getTag();
            if (tag instanceof Integer) cost = (Integer) tag;

            boolean enabled = currentPoints >= cost;
            btn.setEnabled(enabled);
            btn.setAlpha(enabled ? 1f : 0.5f);
        }
    }

    private void fetchUserPoints() {
        if (userId <= 0) { tvPoints.setText("—"); return; }

        RequestBody body = new FormBody.Builder()
                .add("action", "get_user_points")
                .add("userid", String.valueOf(userId))
                .build();

        Request req = new
Request.Builder().url(REWARDS_URL).post(body).build();

        client.newCall(req).enqueue(new Callback() {
            @Override public void onFailure(Call call,
java.io.IOException e) {
```

```
                runOnUiThread(() -> tvPoints.setText("—"));
            }

            @Override public void onResponse(Call call, Response res)
throws java.io.IOException {
                String json = res.body().string();
                int points = 0;
                try {
                    JSONObject o = new JSONObject(json);
                    if
("success".equalsIgnoreCase(o.optString("status"))) {
                        points = o.optInt("points", 0);
                    }
                } catch (Exception ignore) {}

                final int finalPts = points;
                runOnUiThread(() -> {
                    currentPoints = finalPts;
                    tvPoints.setText(formatPoints(currentPoints));
                    refreshRedeemButtons();
                });
            }
        });
    }

    private String formatPoints(int p) {
        return
NumberFormat.getIntegerInstance(Locale.getDefault()).format(p);
    }

    private void showDetailsDialog(RewardItem r) {
        String msg =
                "Restaurant: " + r.restaurant + "\n" +
                        "Location:   " + r.location + "\n" +
                        "Validity:   " + r.validityNote + "\n\n" +
                        "Terms: " + r.terms;

        new AlertDialog.Builder(this)
                .setTitle(r.title)
                .setMessage(msg)
                .setNegativeButton("Close", null)

                .show();
    }

}
```

WeightActivity.java

```
package com.example.bmi;

import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.SeekBar;
import android.widget.TextView;
```

```
import androidx.activity.EdgeToEdge;
import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.contract.ActivityResultContracts;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class WeightActivity extends AppCompatActivity {
    private int selectedHeight = 140; // Default height
    private float selectedWeight = 40.0f; // Default weight
    private SeekBar weightSeekBar;
    private TextView weightText;
    private Button continueButton;
    private final int minWeight = 400;  // 40.0 * 10
    private final int maxWeight = 1000; // 100.0 * 10
    private int gender = -1; // Default gender
    private ActivityResultLauncher<Intent> AgeActivityLauncher;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //EdgeToEdge.enable(this);
        setContentView(R.layout.activity_weight);

        selectedHeight = getIntent().getIntExtra("height", 140);
        gender = getIntent().getIntExtra("gender", -1);

        // Register the launcher inside onCreate
        AgeActivityLauncher = registerForActivityResult(
                new ActivityResultContracts.StartActivityForResult(),
result -> {
                    if (result.getResultCode() == RESULT_OK &&
result.getData() != null) {
                        selectedWeight =
result.getData().getFloatExtra("weight", 40.0f);
                        updateWeightSelection();
                    }
                });

        // Find UI elements
        ImageButton backButton = findViewById(R.id.weightbackButton);
        weightSeekBar = findViewById(R.id.weightseekBar);
        weightText = findViewById(R.id.weightText);
        continueButton = findViewById(R.id.weightcontinueButton);

        // Set SeekBar for 0.1 kg steps (40.0 to 100.0)
        weightSeekBar.setMax(maxWeight - minWeight);
        weightSeekBar.setProgress(0);
        weightText.setText(String.format("%.1f kg", selectedWeight));

        weightSeekBar.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int
progress, boolean fromUser) {
                selectedWeight = (minWeight + progress) / 10.0f;
                weightText.setText(String.format("%.1f kg",
selectedWeight));
            }
```

```
            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {}

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {}
        });

        // Back button click event
        backButton.setOnClickListener(v -> {
            Intent resultIntent = new Intent();
            resultIntent.putExtra("gender", gender);
            resultIntent.putExtra("height", selectedHeight);
            setResult(RESULT_OK, resultIntent);
            finish();
        });

        // Continue button click event
        continueButton.setOnClickListener(v -> {
            Intent intent = new Intent(WeightActivity.this,
AgeActivity.class);
            intent.putExtra("gender", gender);
            intent.putExtra("height", selectedHeight);
            intent.putExtra("weight", selectedWeight);
            AgeActivityLauncher.launch(intent);
        });
    }
    private void updateWeightSelection() {
        if (selectedWeight >= 40.0 && selectedWeight <= 100.0) {
            weightText.setText(String.format("%.1f kg",
selectedWeight));
        }
    }
}
```

WeightHistoryAdapter.java

```
package com.example.bmi;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;
import java.util.Locale;

class WeightHistoryAdapter extends
RecyclerView.Adapter<WeightHistoryAdapter.VH> {

    static class Item {
        final double weight;
        final long timeMillis;
        Item(double w, long t){ weight = w; timeMillis = t; }
    }
```

```
    private final List<Item> items;
    private final SimpleDateFormat df = new SimpleDateFormat("yyyy-
MM-dd", Locale.getDefault());
    WeightHistoryAdapter(List<Item> items){ this.items = items; }

    @NonNull @Override public VH onCreateViewHolder(@NonNull
ViewGroup parent, int viewType) {
        View v = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.item_weight_history, parent,
false);
        return new VH(v);
    }

    @Override public void onBindViewHolder(@NonNull VH h, int pos) {
        Item it = items.get(pos);
        h.tvWeight.setText(String.format(Locale.getDefault(), "%.1f
kg", it.weight));
        h.tvDate.setText(df.format(new Date(it.timeMillis)));

h.tvTimeAgo.setText(RelativeTimeUtil.relativeTime(it.timeMillis));
    }

    @Override public int getItemCount() { return items.size(); }

    static class VH extends RecyclerView.ViewHolder {
        final TextView tvDate, tvTimeAgo, tvWeight;
        VH(@NonNull View v){
            super(v);
            tvDate    = v.findViewById(R.id.tvDate);
            tvTimeAgo = v.findViewById(R.id.tvTimeAgo);
            tvWeight  = v.findViewById(R.id.tvWeight);
        }
    }
}
```

WeightHistoryActivity.java

```
package com.example.bmi;

import android.content.SharedPreferences;
import android.os.Bundle;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import com.google.android.material.chip.ChipGroup;
import org.json.JSONArray;
import org.json.JSONObject;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;
import java.util.Locale;
import okhttp3.*;
import com.google.android.material.appbar.MaterialToolbar;
```

```
public class WeightHistoryActivity extends AppCompatActivity {

    private RecyclerView recycler;
    private ChipGroup chipGroup;
    private final List<WeightHistoryAdapter.Item> data = new
ArrayList<>();
    private WeightHistoryAdapter adapter;

    private int userId;
    private final OkHttpClient client = new OkHttpClient();
    private static final String TRACKING_URL =
"http://10.0.2.2/nutrition/tracking.php";

    @Override protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_weight_history);

        MaterialToolbar toolbar = findViewById(R.id.toolbar);
        toolbar.setNavigationOnClickListener(v -> finish());

        SharedPreferences prefs = getSharedPreferences("user_prefs",
MODE_PRIVATE);
        userId = prefs.getInt("user_id", -1);

        recycler = findViewById(R.id.recycler);
        chipGroup = findViewById(R.id.chipGroup);

        recycler.setLayoutManager(new LinearLayoutManager(this));
        adapter = new WeightHistoryAdapter(data);
        recycler.setAdapter(adapter);

        // Load default (All)
        fetchHistory(0);

        chipGroup.setOnCheckedStateChangeListener((group, checkedIds)
-> {
            int days = 0;
            if (checkedIds.contains(R.id.chip_30)) days = 30;
            else if (checkedIds.contains(R.id.chip_60)) days = 60;
            else if (checkedIds.contains(R.id.chip_90)) days = 90;
            else days = 0; // All
            fetchHistory(days);
        });
    }

    private void fetchHistory(int days) {
        FormBody.Builder fb = new FormBody.Builder()
                .add("action", "get_weight_history")
                .add("userid", String.valueOf(userId));
        if (days > 0) fb.add("days", String.valueOf(days));

        Request req = new Request.Builder()
                .url(TRACKING_URL)
                .post(fb.build())
                .build();

        client.newCall(req).enqueue(new Callback() {
            @Override public void onFailure(Call call,
java.io.IOException e) {
                runOnUiThread(() ->
```

```
Toast.makeText(WeightHistoryActivity.this,
                        "Failed to load history",
Toast.LENGTH_SHORT).show());
            }
            @Override public void onResponse(Call call, Response res)
throws java.io.IOException {
                String json = res.body().string();
                List<WeightHistoryAdapter.Item> temp = new
ArrayList<>();
                try {
                    JSONObject o = new JSONObject(json);
                    JSONArray arr = o.optJSONArray("items");
                    if (arr == null) arr = o.optJSONArray("history");
                    if (arr != null) {
                        for (int i = 0; i < arr.length(); i++) {
                            JSONObject it = arr.getJSONObject(i);
                            double w = it.has("w") ?
it.optDouble("w") : it.optDouble("weight");
                            String ts = it.has("t") ?
it.optString("t") : it.optString("recorded_at");
                            temp.add(new WeightHistoryAdapter.Item(w,
parseToMillis(ts)));
                        }
                    }
                    // newest first
                    Collections.sort(temp, Comparator.comparingLong(a
-> a.timeMillis));
                    Collections.reverse(temp);
                } catch (Exception ignore) {}

                runOnUiThread(() -> {
                    data.clear();
                    data.addAll(temp);
                    adapter.notifyDataSetChanged();
                });
            }
        });
    }

    private long parseToMillis(String s) {
        String[] pats = {"yyyy-MM-dd HH:mm:ss", "yyyy-MM-dd"};
        for (String p : pats) {
            try { return new SimpleDateFormat(p,
Locale.getDefault()).parse(s).getTime(); }
            catch (Exception ignore) {}
        }
        return System.currentTimeMillis();
    }
}
```

Google Form – User Testing

NutriForU Mobile Application User Testing Survey

Thank you for participating in the user testing for **NutriForU**, a personalized nutrition mobile application designed to support your health and dietary goals. This survey will take approximately 10-15 minutes to complete. Your feedback is invaluable for improving our Final Year Project (FYP). All responses are anonymous.

* Indicates required question

NutriFor U

Your Goals, Our Guide

Section 1: Demographics

1. 1. What is your age group? *

Mark only one oval.

- ◯ Under 18
- ◯ 18 - 24
- ◯ 25 - 34
- ◯ 35 - 44
- ◯ 45 and above

2. 2. What your gender? *

Mark only one oval.

- ◯ Male
- ◯ Female
- ◯ Prefer not to say

3. 3. How often do you use nutrition or health tracking apps? *

Mark only one oval.

- ◯ Daily
- ◯ Weekly
- ◯ Monthly
- ◯ Rarely/Never

4. 4. What is your primary goal for using NutriForU? *

Mark only one oval.

- ◯ Weight loss
- ◯ Weight maintenance
- ◯ Healthy eating
- ◯ Gain Muscle
- ◯ Other: _____

5. 5. How comfortable are you with using mobile apps? *

Mark only one oval.

- ◯ Very comfortable
- ◯ Somewhat comfortable
- ◯ Neutral
- ◯ Somewhat uncomfortable
- ◯ Very uncomfortable

Section 2: Overall Experience

6. 1. How intuitive was the overall navigation of NutriForU? *

Mark only one oval.

	1	2	3	4	5	
(Ver	◯	◯	◯	◯	◯	(Very Easy)

7. 2. How satisfied were you with NutriForU overall? *

Mark only one oval.

	1	2	3	4	5	
(Ver	◯	◯	◯	◯	◯	(Very Satisfied)

8. 3. How useful was the overall feature set of NutriForU (e.g., Health Tracking, Dietary Plan, Voucher, Settings)? *

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

9. 4. How likely are you to recommend NutriForU to others? *

Mark only one oval.

	1	2	3	4	5	
(Not	◯	◯	◯	◯	◯	(Very Likely)

10. 5. How visually appealing was the app's design? *

Mark only one oval.

	1	2	3	4	5	
(Not	◯	◯	◯	◯	◯	(Very Appealing)

Section 3: Login and Sign-Up Process

11. 1. How easy was it to create an account on NutriForU? *

Mark only one oval.

	1	2	3	4	5	
(Ver	◯	◯	◯	◯	◯	(Very Easy)

12. 2. How clear was the user preference setup (gender, weight, height....) *

Mark only one oval.

	1	2	3	4	5	
(Ver	◯	◯	◯	◯	◯	(Very Clear)

13. 3. How smooth was the login process? *

Mark only one oval.

	1	2	3	4	5	
(Ver	◯	◯	◯	◯	◯	(Very Smooth)

14. 4. How intuitive was the transition from sign-up to the Home page? *

Mark only one oval.

	1	2	3	4	5	
(Ver	◯	◯	◯	◯	◯	(Very Intuitive)

15. 5. How well did the sign-up process guide you to input accurate data? *

Mark only one oval.

	1	2	3	4	5	
(No1	◯	◯	◯	◯	◯	(Very Well)

Section 4: Home Page

16. 1. How useful was the greeting message on the Home page? *

Mark only one oval.

	1	2	3	4	5	
(No1	◯	◯	◯	◯	◯	(Very Useful)

17. 2. How relevant was the daily dietary plan displayed on the Home page? *

Mark only one oval.

	1	2	3	4	5	
(No1	◯	◯	◯	◯	◯	(Very Relevant)

18. 3. How clear was the display of your current BMI and daily calorie intake? *

Mark only one oval.

	1	2	3	4	5	
(Ver	◯	◯	◯	◯	◯	(Very Clear)

19. 4. How appropriate was the daily sign-in popup for collecting points? *

Mark only one oval.

	1	2	3	4	5	
(Ver	◯	◯	◯	◯	◯	(Very Appropriate)

20. 5. How visually appealing was the Home page layout? *

Mark only one oval.

	1	2	3	4	5	
(Not	◯	◯	◯	◯	◯	(Very Appealing)

Section 5: Health Tracking System

21. 1.How easy was it to log your weight manually on the Health Tracking system? *

Mark only one oval.

	1	2	3	4	5	
(Ver	◯	◯	◯	◯	◯	(Very Easy)

22. 2. How clear was the weight change graph (with the target BMI red line)? *

Mark only one oval.

	1	2	3	4	5	
(Ver	◯	◯	◯	◯	◯	(Very Clear)

23. 3. How motivating were the messages or tips displayed after logging your weight? *

Mark only one oval.

	1	2	3	4	5	
(Not	◯	◯	◯	◯	◯	(Very Motivating)

24. 4. How useful was the Health Tracking page for monitoring your progress? *

Mark only one oval.

	1	2	3	4	5	
(Not	◯	◯	◯	◯	◯	(Very Useful)

25. 5. How visually appealing was the Health Tracking page? *

Mark only one oval.

	1	2	3	4	5	
(Not	◯	◯	◯	◯	◯	(Very Appealing)

Section 6: Dietary Plan System

26. 1. How useful was the "Today's Plan" display on the Dietary Plan page? *

Mark only one oval.

	1	2	3	4	5	
(Not	◯	◯	◯	◯	◯	(Very Useful)

27. 2. How easy was it to use the "Shuffle Plan" button to switch dietary plans? *

Mark only one oval.

	1	2	3	4	5	
(Ver	◯	◯	◯	◯	◯	(Very Easy)

28. 3. How helpful were the images displayed beside the dietary plans? *

Mark only one oval.

	1	2	3	4	5	
(Not	◯	◯	◯	◯	◯	(Very Helpful)

29. 4. How intuitive were the options to save, edit, or delete personalized plans? *

Mark only one oval.

	1	2	3	4	5	
(Ver	◯	◯	◯	◯	◯	(Very Intuitive)

30. 5. How well did the Dietary Plan page meet your personalization needs? *

Mark only one oval.

	1	2	3	4	5	
(No1	◯	◯	◯	◯	◯	(Very Well)

Section 6: Voucher Page

31. 1. How clear was the display of collected points, available vouchers, and redeem history? *

Mark only one oval.

	1	2	3	4	5	
(Ver	◯	◯	◯	◯	◯	(Very Clear)

32. 2. How motivating were the daily sign-in points for regular app use? *

Mark only one oval.

	1	2	3	4	5	
(No1	◯	◯	◯	◯	◯	(Very Motivating)

33. 3. How appealing were the available vouchers for redemption? *

Mark only one oval.

	1	2	3	4	5	
(No1	◯	◯	◯	◯	◯	(Very Appealing)

34. 4. How easy was it to redeem vouchers on the Voucher page? *

Mark only one oval.

	1	2	3	4	5	
(Ver	◯	◯	◯	◯	◯	(Very Easy)

35. 5. How useful was the Voucher page overall? *

Mark only one oval.

	1	2	3	4	5	
(Not	◯	◯	◯	◯	◯	(Very Useful)

Section 8: Setting Page

36. 1. How easy was it to update your account details (name, height, weight, gender, target BMI)? *

Mark only one oval.

	1	2	3	4	5	
(Ver	◯	◯	◯	◯	◯	(Very Easy)

37. 2. How clear was the option to update dietary preferences (e.g., allergic foods)? *

Mark only one oval.

	1	2	3	4	5	
(Ver	◯	◯	◯	◯	◯	(Very Clear)

38. 3. How useful was the notification toggle feature? *

Mark only one oval.

	1	2	3	4	5	
(No	◯	◯	◯	◯	◯	(Very Useful)

39. 4. How effective was the text size adjustment feature? *

Mark only one oval.

	1	2	3	4	5	
(No	◯	◯	◯	◯	◯	(Very Effective)

40. 5. How intuitive was the Settings page layout (e.g., help & support, about, logout)? *

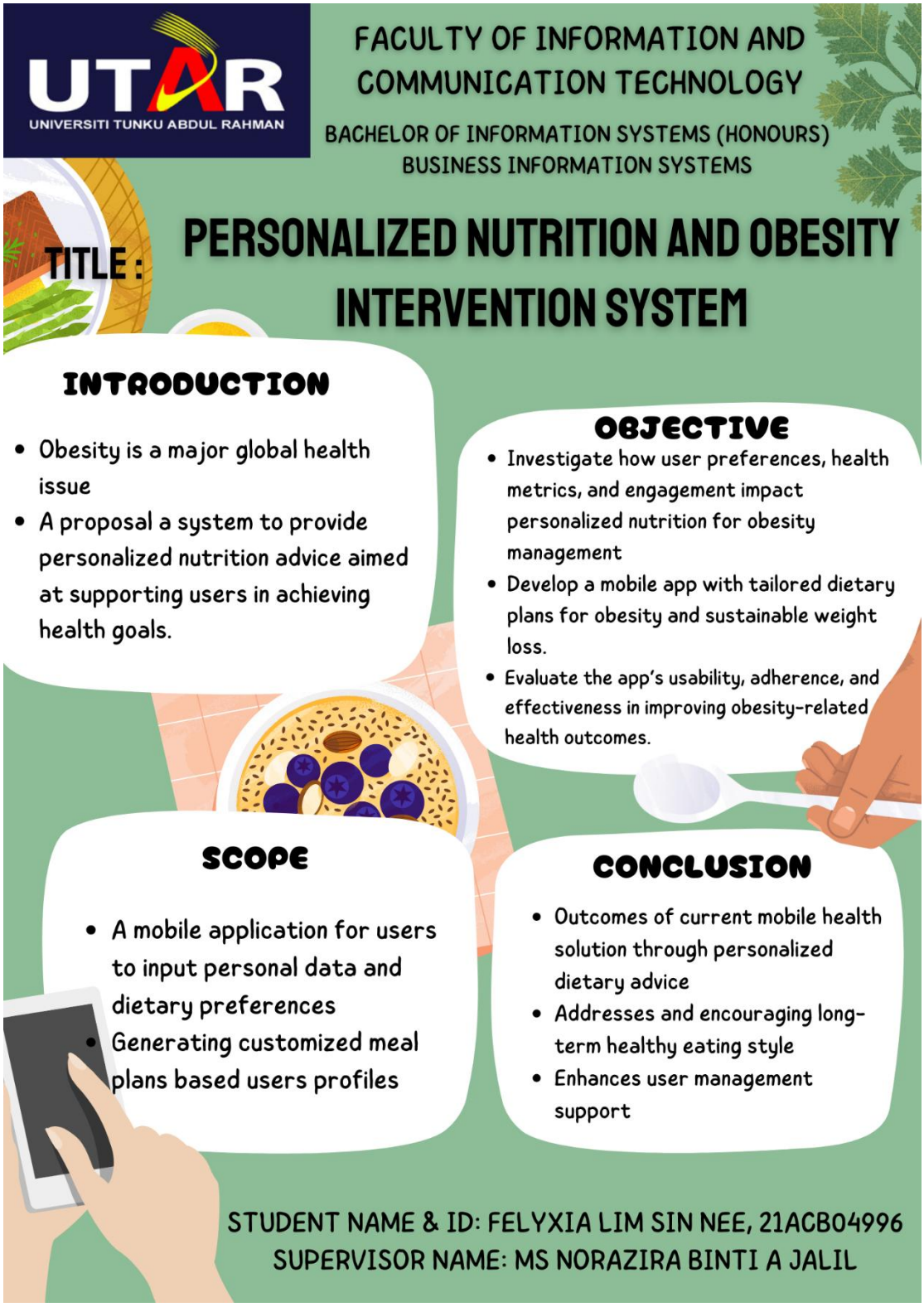
Mark only one oval.

	1	2	3	4	5	
(Ver	◯	◯	◯	◯	◯	(Very Intuitive)

Thank You! ^0^

Thank you for taking the time to complete this survey! Your feedback is crucial for improving **NutriForU** and ensuring it meets your needs. We truly appreciate your help in shaping our app.

Poster


UTAR
UNIVERSITI TUNKU ABDUL RAHMAN
FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
BACHELOR OF INFORMATION SYSTEMS (HONOURS) BUSINESS INFORMATION SYSTEMS
TITLE :
PERSONALIZED NUTRITION AND OBESITY INTERVENTION SYSTEM
INTRODUCTION
Obesity is a major global health issue
A proposal a system to provide personalized nutrition advice aimed at supporting users in achieving health goals.
OBJECTIVE
Investigate how user preferences, health metrics, and engagement impact personalized nutrition for obesity management
Develop a mobile app with tailored dietary plans for obesity and sustainable weight loss.
Evaluate the app's usability, adherence, and effectiveness in improving obesity-related health outcomes.
SCOPE
A mobile application for users to input personal data and dietary preferences
Generating customized meal plans based users profiles
CONCLUSION
Outcomes of current mobile health solution through personalized dietary advice
Addresses and encouraging long-term healthy eating style
Enhances user management support
STUDENT NAME & ID: FELYXIA LIM SIN NEE, 21ACB04996
SUPERVISOR NAME: MS NORAZIRA BINTI A JALIL