PREDICTIVE PERSONALISED WORKOUT AND DIETARY GUIDANCE SYSTEM

BY

ALICIA CHUA XIU WEN

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JUNE 2025

COPYRIGHT STATEMENT

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisors, Ts Dr Mogana a/p Vadiveloo, for her invaluable guidance, continuous support, and expert advice throughout this research. Her insightful feedback and encouragement were instrumental in shaping this project.

I am also sincerely thankful to the Faculty of Information System and Communication Technology (FICT) at Universiti Tunku Abdul Rahman (UTAR) for providing the necessary resources and academic support.

Lastly, I acknowledge all the researchers and developers whose previous work laid the foundation for this project.

ABSTRACT

The COVID-19 pandemic has increased the use of fitness and dietary mobile applications. While existing fitness and dietary applications offer useful functionalities, they often fail to deliver personalised recommendation that account for individual differences. This project proposes the development of the "Predictive Personalised Workout and Dietary Guidance System", a comprehensive mobile application designed to address the shortcomings of existing systems. This application utilises publicly available datasets and integrates artificial intelligence to analyse user data such as weight, height, gender, and age, offering tailored recommendations that evolve with user progress. Deep learning models were integrated and evaluated to predict users' Body Mass Index (BMI) classification. During the development phase in, three predictive models were implemented and evaluated: a Deep Neural Network (DNN), a U-Net-based Convolutional Neural Network (CNN) and a Random Forest. Among them, the CNN model achieved the highest test accuracy of 90.36%, but DNN and Random Forest only achieved a test accuracy of 88.70% and 85.00%, respectively, proving the U-Net-based CNN model is more effective and reliable for BMI classification. This result highlights the advantage of using a U-Net-based CNN architecture for personalised health predictions within the application. Unlike existing systems, which often focus primarily on exercise tracking with minimal dietary support and lack of suitable workout recommendations. The application will include functions such as fitness and dietary tracking, community platform to enhance user engagement and motivation, artificial intelligence (AI) chatbot that support users with personalised guidance, and weight tracking function. In addition, the system implements a personalised dietary module that uses AI to analyse meals' macronutrient intake, enabling users to adopt a more health-oriented diet. With the comprehensive functions offered by the system, users are expected to benefit from more precise and adaptable health guidance, thereby improving long-term commitment and overall health outcomes.

Area of study: Mobile Application Development, Deep Learning

Keywords: Mobile Application, Fitness, Deep Learning, Machine Learning, Firebase, Body Mass Index (BMI) Classification, Artificial Intelligence (AI) Chatbot

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

1D	1-Dimensional
AI	Artificial Intelligence
API	Application Programming Interface
BFP	Body Fat Percentage
BMI	Body Mass Index
CNN	Convolutional Neural Network
DNN	Deep Neural Network
DL	Deep Learning
JSON	JavaScript Object Notation
MET	Metabolic Equivalent of Task
MVC	Model-View-Controller
ReLU	Rectified Linear Unit
SDLC	Software Development Life Cycle
TEE	Total Energy Expenditure
WHO	World Health Organisation

Chapter 1
Introduction

1.1 Background information

In recent decades, global health has faced a major challenge in the form of rising obesity rates and associated lifestyle diseases. The World Health Organisation (WHO) reported that over 2.5 billion adults were overweight in 2022, with 890 million classified as obese, highlighting the need for more effective health interventions tailored to individual needs [1]. Traditional approaches to fitness and nutrition management have often relied on generic advice that fails to consider the user's physical condition, personal preferences, and evolving goals. As a result, such methods have led to limited success in fostering long-term behaviours change and improving public health outcomes [2]. However, despite their popularity, many of these platforms still rely on generalised algorithms and static recommendations that do not adapt to individual progress or changing needs, leading to user dissatisfaction and eventual disengagement [3].

The field of personalised health systems, particularly those using artificial intelligence (AI) techniques, seeks to resolve these shortcomings by offering data-driven insights tailored to each user. Key metrics such as Body Mass Index (BMI), Basal Metabolic Rate (BMR), and Total Energy Expenditure (TEE) are central to delivering effective, personalised recommendations. By learning from user behaviours and physiological data, AI-driven applications can dynamically adapt workout and dietary plans over time. These smart systems surpass conventional apps by providing ongoing feedback, suggesting modifications, and adjusting to user goals in real-time [4][2].

Equally important in driving user success is the social dimension of health platforms. Research shows that community engagement—via peer interaction, group challenges, and progress sharing—can significantly enhance user motivation and commitment [4]. Platforms like Strava [5] have demonstrated the impact of clubs and leaderboards in fostering a sense of belonging, while FITTR [6] and MyNetDiary [7] incorporate features such as AI chatbots and group discussions to provide emotional and informational support. These components are vital in promoting sustained engagement and turning health goals into daily habits.

In summary, the intersection of mobile technology, AI, and health science offers a compelling opportunity to design intelligent, responsive systems that go beyond traditional fitness tracking. The proposed system builds on this foundation by integrating AI-based personalisation, flexible fitness and diet planning, and social interaction features to deliver a user-centric, evolving experience. For readers unfamiliar with the technical aspects, it is essential to understand that this project not only involves app development but also incorporates machine learning techniques to interpret user data and provide meaningful, personalised health recommendations in real-time.

1.2 Problem Statement and Motivation

Due to the lockdown of COVID-19 pandemic, fitness application has led to an increase in demand among people [3]. This is because fitness centres and gyms were forced to close or switch to digital. During the first half of 2020, health and fitness application downloads grew by 46% worldwide, and the daily active users increased by 24% from Q1 to Q2 in year 2020 [8]. Fitness applications have the ability to improve users' exercising consistency, maintain their fitness habits, and encourage regular walking habits [4]. Furthermore, existing similar systems such as Strava [5], MyFitnessPal [9], FITTR [6], and MyNetDiary [7], play a crucial part in guiding users towards a healthy lifestyle by providing personalised fitness routines, progress monitoring, food recommendations, and social aspects that build a sense of community. These features enable users to follow good habits, create achievable exercise goals, and stay inspired on their path to a healthy lifestyle. Therefore, the growing popularity of fitness apps, along with their ability to provide personalised guidance and assistance, has the potential to improve public health and promote better lifestyles around the world.

This project is motivated by the crucial need to solve the limitations of existing fitness and dietary application. As users seek tools that match their individual measurements and exercise goals, the lack of personalised recommendations and flexibility in current applications can lead to frustration, reduced motivation, and ultimately, abandonment of the platform. Furthermore, many fitness applications lack community features, limiting users of the social support and peer motivation that are crucial to maintain long-term dedication to their health goals [10]. Moreover, this project is motivated by a desire to fill these gaps by developing an application that not only provides tailored and evolving guidance but also fosters a supportive community environment, thereby increasing user engagement, satisfaction, and overall success in achieving fitness and dietary goals.

1.3 Project Scope and Direction

This project will deliver an Android based mobile application aimed at health-conscious individuals, fitness enthusiasts, and users seeking personalised dietary and workout guidance. It addresses the lack of personalisation and adaptability in existing fitness and dietary mobile applications by developing a comprehensive data-driven mobile application.

Unlike generic mobile applications, this system combines Artificial Intelligence (AI)-driven personalisation, and community support into a single platform, offering a holistic solution for sustainable health management. To support prediction and recommendation features such as Body Mass Index (BMI) category analysis, the application utilises publicly available datasets, including the *Fitness Exercises using Body Fat Percentage (BFP) & Body Mass Index (BMI)* [11] from Kaggle, ensuring data-driven accuracy and model training integrity.

1.4 Project Objectives

The primary objective of this project is to develop an application that accurately analyses user data to deliver personalised fitness and dietary guidance. This is achieved through the following sub-objectives:

1. **To implement adaptive recommendations**
   To provide workout recommendations based on user preferences.
2. **To integrate tracking tools**
   To enable users to log in their meals, exercises, and anthropometric data, with automated calories and nutrient calculations.
3. **To build a community platform**
   To facilitate social interaction through challenges, leaderboards, and peer support forums.
4. **To implement an Artificial Intelligence (AI) chatbot**
   To provide instant, personalised advice through integration with relevant Application Programming Interface (APIs).

1.5 Impact, Significance, and Contribution

This project addresses a critical gap in digital health tools by combining personalisation, adaptability, and social engagement. Its contributions include:

1. **Improved health outcomes**
   Tailored recommendations increase adherence to fitness plans, directly combating obesity trends.
2. **Technological innovation**
   Integration of deep learning for dynamic adaptation sets a new standard for health mobile applications.
3. **Social impact**
   Community features foster accountability and motivation, addressing the isolation often felt in fitness journeys.
4. **Scalability**
   The use of public datasets ensures the system can adapt to diverse populations and evolving health trends.

By bridging the divide between generic advice and individual care, this system has the potential to revolutionize how users approach health management in the digital age.

1.6 Report Organisation

The details of this report are organised into seven chapters. Chapter 2 presents the literature review of existing related systems and relevant machine learning and deep learning models. Chapter 3 discusses the system design, such as the system block diagram, system flowchart, and database design. Chapter 4 outlines the methodology and approaches used in the system, including the system architecture, system requirements and the project timeline. Chapter 5 shows the system implementation on hardware, software, configurations, and operations. Chapter 6 provides an in-depth evaluation and discussion on the system and the machine learning and deep learning models. Finally, Chapter 7 concludes the report by providing recommendation for further improvements.

Chapter 2

Literature Review

In recent years, the rapid advancement of technology has significantly impacted the fitness and dietary industry, allowing the development of fitness and dietary systems. Fitness and dietary system play an important role in promoting the health and wellbeing of individuals and communities. This chapter will be focus on reviewing and examining the existing fitness and dietary application, evaluating the methods and functionalities of existing application. In addition, this chapter will also introduce machine learning and deep learning models that have been applied to enhance prediction and user interaction in the application.

2.1 Introduction to Machine Learning Models

Machine Learning is a branch of artificial intelligence (AI) that focusses on the development of algorithms capable of learning patterns from data and generating predictions without being explicitly programmed with specific rules. It includes a variety of techniques like as supervised, unsupervised, semi-supervised, and reinforcement learning. In supervised learning, models are trained on input-output pairs to determine feature-label mappings, which are often used in classification and regression problems. Unsupervised learning, on the other hand, seeks to identify hidden structures within unlabelled datasets using techniques such as clustering and dimensionality reduction, whereas semi-supervised learning mixes limited labelled data with a higher proportion of unlabelled data to increase generalisation. Reinforcement learning focusses on sequential decision-making problems, where agents learn to maximise cumulative rewards by interacting with an environment. [12] [13]

Traditional machine learning models are less computationally expensive and can perform well on small or organised datasets. They also provide interpretability, as algorithms like decision trees or logistic regression frequently provide insights into feature significance and decision rules. However, machine learning models have difficulties in handling high-dimensional or unstructured data. Despite these challenges, machine learning is still widely used in many sectors, including healthcare and finance. [12][13]

2.1.1 Random Forest

Random Forest is one of the most widely used ensembles learning algorithms in machine learning, introduced by Breiman as an improvement over individual decision tree [14]. As shown in Figure 2.1.1, it works by building a large number of decision trees during training and integrate their outputs to make more robust and accurate predictions [14].

In the Figure 2.1.1 below, each tree in the random forest is trained on a bootstrap sample of the original dataset, a process known as bagging, which introduces diversity among the trees and reduce variance. At each node split, a randomly chosen subset of features is considered rather than the entire set, minimising correlation between trees and improving generalisation capability. For classification problems, the model predicts the class label on majority voting across trees, but for regression problems, it averages the numerical outputs [14].

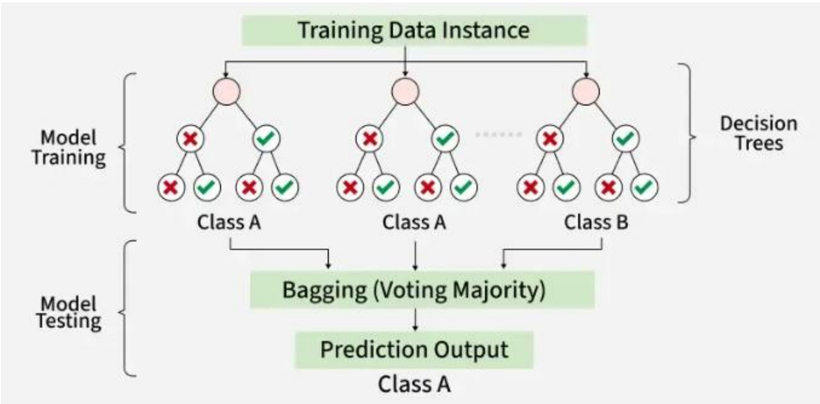


Figure 2.1.1 Visualisation of random forest model

The architecture's benefits in its resilience to overfitting, robustness to noise and outliers, and ability to manage categorical and continuous data adequately [15]. Furthermore, random forest useful by quantifying how much each feature contributes to reducing prediction error, making it valuable for feature selection and interpretation [15].

Despite these advantages, random forest is not without limitations. Training and inference can be computationally expensive when the number of depth of trees is large, which can limit scalability for very high-dimensional datasets. Although random forest is more interpretable than some ensemble methods, the combination of many trees complicates tracing individual decision routes, making it less transparent than simpler models. Its model performance also highly dependent on hyperparameters such as the number of trees, the maximum depth of each tree, the minimum samples required for splits, and the number of features considered at each node. Proper parameter adjustments balance the bias-variance trade-off, preventing the model from underfitting or overfitting. [16] [14]

2.2 Introduction to Deep Learning Models

Deep learning model is a subset of machine learning, which uses artificial neural networks with multiple layers to automatically learns pattern from data [17]. As shown in Figure 2.2.1, these networks consist of three primary layers: an input layer that receives raw data, hidden layers that transform data through mathematical operations, and an output later that generates predictions [17]. Each neuron applies an activation function to introduce non-linearity, enabling the model to learn intricate relationships in the data.

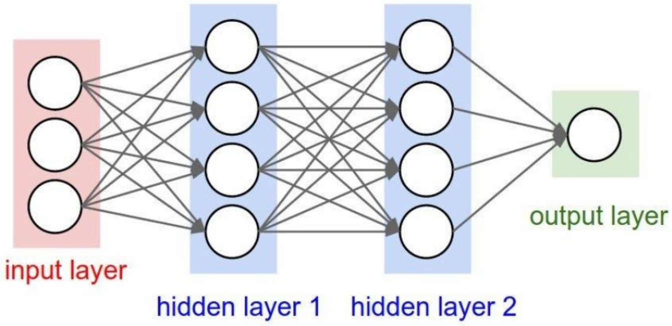


Figure 2.2.1 Visualisation of deep learning model

The training process in deep learning involves two key steps: forward propagation and backpropagation. During forward propagation, data flows through the network, and the model makes prediction [17]. The difference between these predictions and actual outcomes is quantified using a loss function, such as Mean Squared Error. In backpropagation, the model adjusts the weights and biases of its neurons using gradient descent, an optimization algorithm that minimizes prediction errors [17]. This cycle repeats over thousands of iterations or epochs, refining the model's accuracy.

Unlike traditional machine learning, deep learning model offers significant advantages through its ability to autonomously perform feature engineering, generating new, meaningful features from limited or unstructured data without human intervention [18]. Deep learning also scales efficiently with large datasets, optimizing parameters across full cycle learning to improve precision. Its capacity to automate feature extraction reduces time and effort in preprocessing, while its adaptability allows it to tackle diverse data formats and evolving challenges [18]. These strengths make deep learning indispensable for tasks requiring data-driven insights.

2.2.1 Deep Neural Network (DNN)

Deep Neural Network (DNN) is the foundational architecture of deep learning, consisting of multiple layers of interconnected neurons [18]. Unlike shallow networks, DNNs leverage hierarchical layers to learn increasingly abstract representations of data to uncover non-linear relationship.

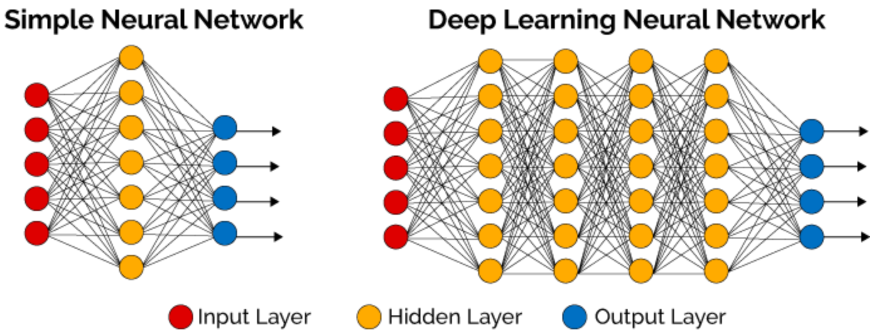


Figure 2.2.2 Comparison between simple neural network and DNN

However, DNNs face significant challenges, including high computational costs, scalability issues, overfitting risks, dependency on large, labelled datasets, lack of interpretability, or even ethical vulnerabilities like adversarial attacks [19]. These limitations complicate deployment in real-world application.

To effectively train a DNN, several hyperparameters must be carefully tuned to optimize performance and prevent common issues such as overfitting. Key hyperparameters including the learning rate, which controls how quickly the model adjusts its weights during training. The number of hidden layers and neurons per layer determine the model's capacity to learn complex patterns as more layers and neurons generally allow for greater expressiveness but increase the risk of overfitting. The batch size dictates how many samples are used in one forward or backward pass, where smaller batch sizes provide more updates but introduce noise. The number of epochs defines how many complete passes through the training dataset the model performs. Additionally, activation functions like Rectified Linear Unit (ReLU) or sigmoid are important for introducing non-linearity into the model, and regularization techniques such as dropout or L2 regularization are often used to mitigate overfitting [20].

2.2.2 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is a specialized class of deep learning models designed for processing grid-like data, such as images, by leveraging spatial hierarchies [21]. CNNs excel at capturing local patterns through its convolutional layers. As in Figure 2.2.3, in the convolutional layers, these layers apply learnable kernels to input data to detect spatial features [22]. Each filter slides over the input, computing dot products to generate feature maps [22]. The pooling operations (e.g., max-pooling) progressively reduce spatial dimensions, enhancing translational invariance [22]. Positioned at the network's end, these layers aggregate high-level features for classification or regression. This also includes the dropout and regularization, which is to prevent overfitting by randomly deactivating neurons during training, ensuring robustness to noise [23].

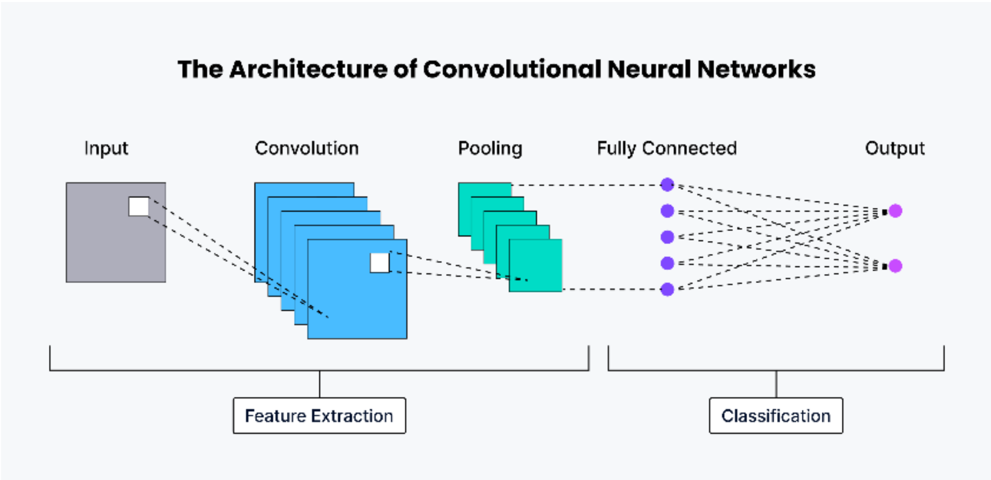


Figure 2.2.3 Architecture of CNN

2.2.2.1 U-Net

The U-Net architecture, a CNN variant, introduces symmetric expanding paths to recover spatial resolution lost during down sampling. U-Net is characterized by its U-shaped encoder-decoder structure with skip connections, as shown in Figure 2.2.4. The encoder extract features from input, and the decoder utilises these features to create a segmentation mask.

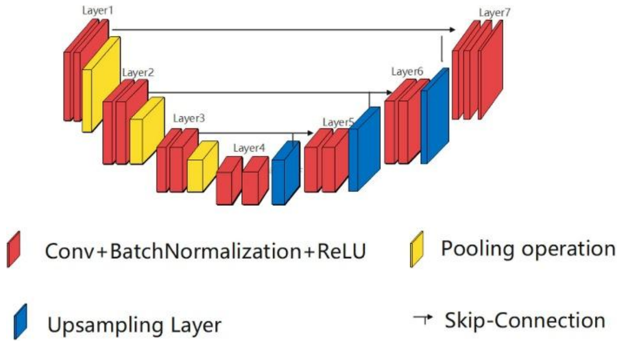


Figure 2.2.4 Architecture of U-Net

The encoder consists of convolutional blocks and pooling operations. Each block in the convolutional blocks uses two consecutive 3x3 convolutions to extract spatial features, batch normalization for stable training, and ReLU activation which introduce nonlinearity while mitigating vanishing gradients [24]. While the decoder recovers spatial resolution through up sampling layers, feature concatenation, and convolutional blocks [24].

One of the critical innovations of U-Net is their skip connections, which allow addressing information loss during down sampling by directly transferring encoder features to decoder [24].

In addition to its architectural strengths, U-Net's performance is significantly influenced by several critical hyperparameters. One of the key parameters is the kernel size in convolutional layers, which determines the receptive field and plays a role in capturing local versus broader spatial features. Another essential consideration is the convolutional layer configuration. For example, whether each block contains one or two convolutional layers before pooling. More complex configurations can extract richer features but also increase training complexity. Additionally, the number of pooling and up-convolutional layers defines the depths of the encoder-decoder structure, which affects how well the model can abstract features [25]. Proper tuning of these parameters significantly improves model performance.

2.3 Evaluation of Existing Similar System

2.3.1 Strava

Strava [5] is a fitness-oriented mobile application available in both Google Play Store and Apple App Store that primarily focuses on tracking, recording, and analysing user's running and cycling activities through GPS device or mobile. Strava is a strong fitness application which records over 30 types of activities, covers 100 million athletes in 195 countries.

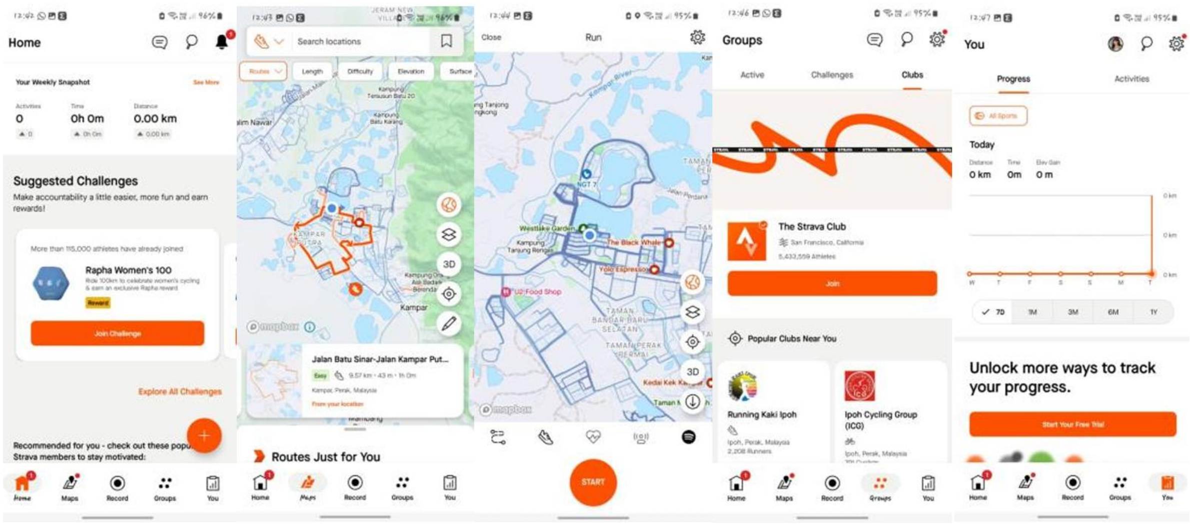


Figure 2.3.1 Main features in Strava

It supports various features including filterable exercise recommendation, real-time exercise tracking and recording, community aspects, and user analytics as shown in Figure 2.3.1. As unlock additional paid features such as smart routing with 3 dimensional (3D) maps, downloading favourite routes to use offline, segment leaderboards, advanced training analysis, design own challenges, progress tracking and more, users have to subscribe to the Strava membership.

The home page of Strava allows user to directly view their analysed weekly report on number of activities, total workout time and total workout distance, suggest available challenges available to users, workout posting by the Strava members, and lastly a button allowing users to post their workout activity to the community.

The filterable exercise route recommendation is the highlighted feature available in Strava. This feature allows user to filter and get new recommended route by length, difficulty, elevation and surface. The system will suggest a new route for the users based on their filtering. However, these features only available to the Strava members. Other than following the exercise as per the given recommended routes, users are also allowed to "run in their own way" using the real-time exercise tracking given. This feature support music playing while exercise, which is connected to the music applications, sports choosing such as run, trail run, walk, ride, swim skate and more, build own custom routes in their website, and heart rate sensing using other devices like health wristband. After the user completed their workout, the workout history will be logged into the history and users are able to view them in the activities analysis.

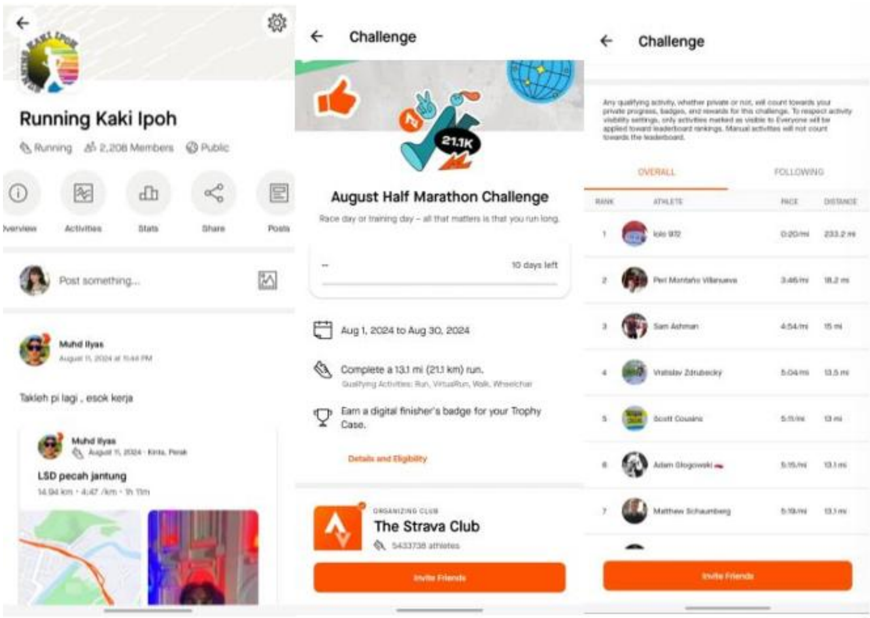


Figure 2.3.2 Clubs and challenges of Strava

Other than physical workout activity tracking and analysing, Strava also provide a platform for users to interact with other Strava users in the community feature. According to Figure 2.3.2, users will be recommended by the nearest available clubs from them. The figure shows the sub-features in the clubs. Users can view an overview of the club, access activity feeds posted by club members, share the club with others, create workout posts, and check weekly statistics. These statistics include the workout distance rankings of club members, the total activities performed by members, the overall workout distance of the club, and more. Club members are encouraged to share their workout activities as a post in the club, and club members are allowed to like and comment on it.

Additionally, Strava also offers various challenges allow users to take part into. As shown in Figure 2.3.2, once user joined a challenge, there will be a progress bar showing user's challenge progress, other details on the challenge, and a leaderboard of the challenge participants. Users will earn a digital finisher's badge once they have completed their challenge. These community feature in Strava increase the Strava user's workout engagement and interaction, enhancing their workout experience and increase user activity.

2.3.2 MyFitnessPal

MyFitnessPal [9] is a well-known nutrition tracking application that helps users create healthy habits with its all-in-one food, exercise, and calorie tracker. This application is featured as the GQ 2022 Fitness Awards "Best Fitness App" [26] and receives 3.7 million 5-star reviews due to its comprehensive workout-tracking tools, including detailed macronutrient information, weight-tracking, and customisable goals.

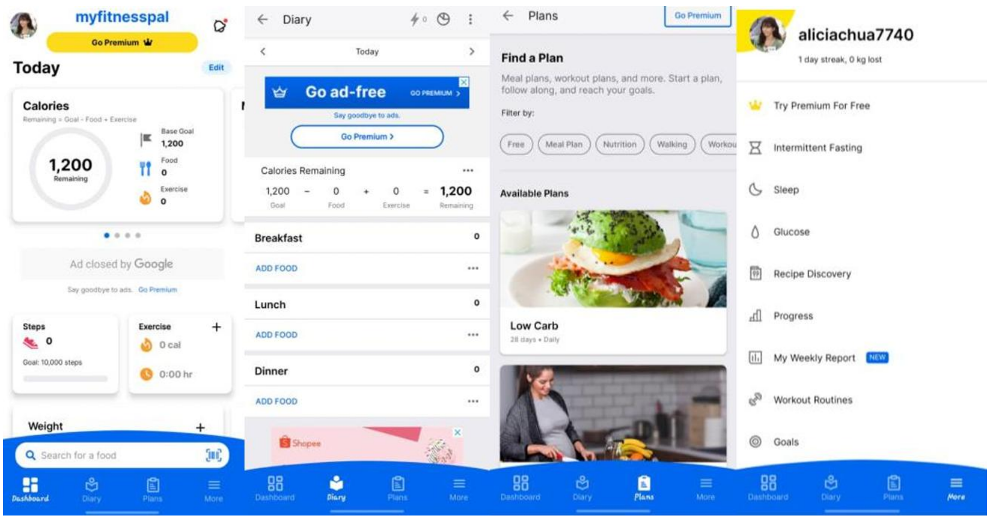


Figure 2.3.3 Main features of MyFitnessPal

There are several main features provided by MyFitnessPal. The highlighted feature in the system is the calories and macronutrient counting. Users are provided with initial goals, by just answering on few questions while creating their user profile, including age, height, weight, gender, goal weight and so on. MyFitnessPal sets the user daily calorie goal in net calories which defines as [calories consumed (food) – calories burned (exercise) = net calories]. In order to obtain user's calories consumption, MyFitnessPal has a wide range of food dataset which contains over 18 million global foods to easily track and calculate the calories consummation.

Based on Figure 2.3.3, other than calorie features, MyFitnessPal also provide exercise recording feature to allow users to log in their exercises including cardiovascular, strength, or workout routines. The exercise logging feature contains up to 300 types of exercise available for user to log and track their calories burned. Furthermore, MyFitnessPal provides plan features, which provide meal plans, workout plans, and more for users to reach their goals. There are more additional features provided by MyFitnessPal under the "more" section, such as intermittent fasting, sleep tracking, glucose tracking, recipe discovery, weekly report and so on.

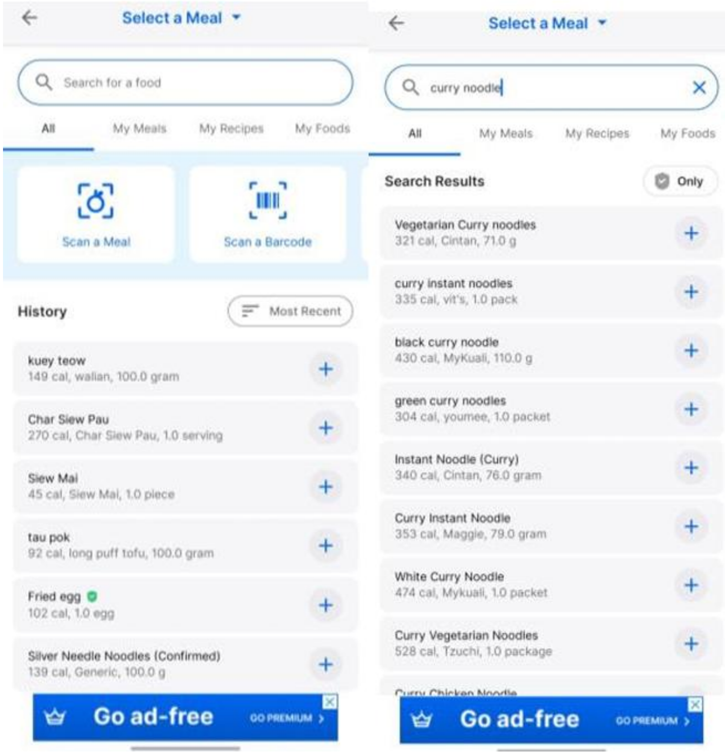


Figure 2.3.4 Dietary tracking of MyFitnessPal

Figure 2.3.4 above shows the dietary tracking feature in MyFitnessPal which contains over 18 million global foods. Users can log their food taken using the search function. For example, user search for curry noodle, there will be results of different types of curry noodles with estimated calories and net weight in grams. This feature also supports food meal scanning or barcode scanning. However, these action only available to MyFitnessPal premium users. Users are also allowed to save their favourite meal and recipe for fast logging and create own food if the food is not inside their database.

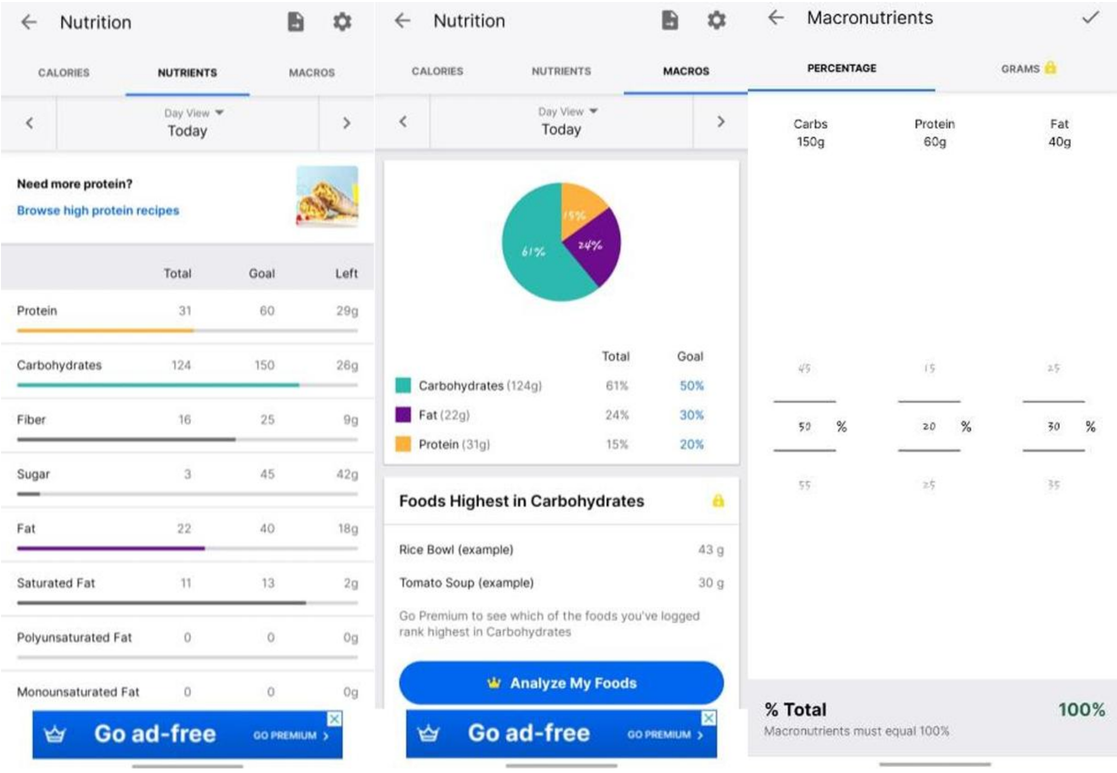


Figure 2.3.5 Nutrients and macronutrients distribution of MyFitnessPal

After logging a food taken, user is able to see the nutrients distribution they have taken such as protein, carbohydrates, sugar fat, or even cholesterol, sodium, potassium and so on as shown in Figure 2.3.5. MyFitnessPal will evaluate the nutrient goal and calculates the nutrient left they should take by subtracting the total nutrient they have taken. Users can also see the macronutrient ratio they have taken, and/or adjust their preferred macronutrient percentage ratio.

2.3.3 FITTR

FITTR [6] is a mobile fitness and coach application, which focuses on virtual coaching to provide tailored plans, fitness guidance, and supportive community to help users to achieve their fitness goals.

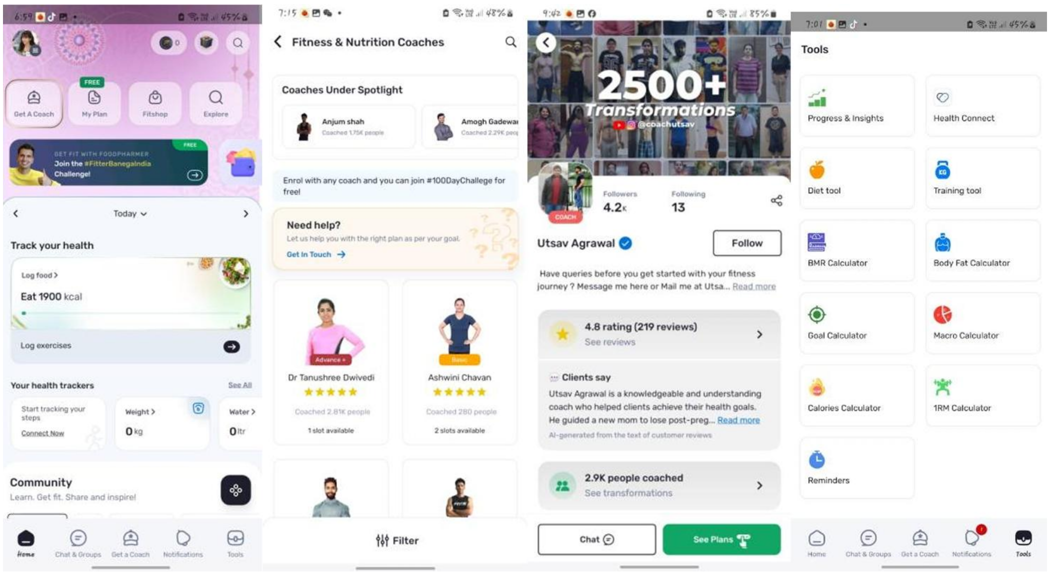


Figure 2.3.6 Main features of FITTR

FITTR contains various features including fitness and nutrition coaches, nutrition and diet logging and tracking, health tracker such as walking steps and weight, and community discussions. Users able to select certified fitness and nutrition coach, online personal training coach, personalised clinical diet planning coach or even mental guidance coach. The system covers a wide range of coaches from physical to mental. As shown in Figure 2.3.6 above, users are able to view available coaches, review their ratings and comments from other users, view coaches' specialties and certificates, and their posts on the community. If users are interested to the particular coach, users are able to chat with the coach or proceed to payment for fitness coaching.

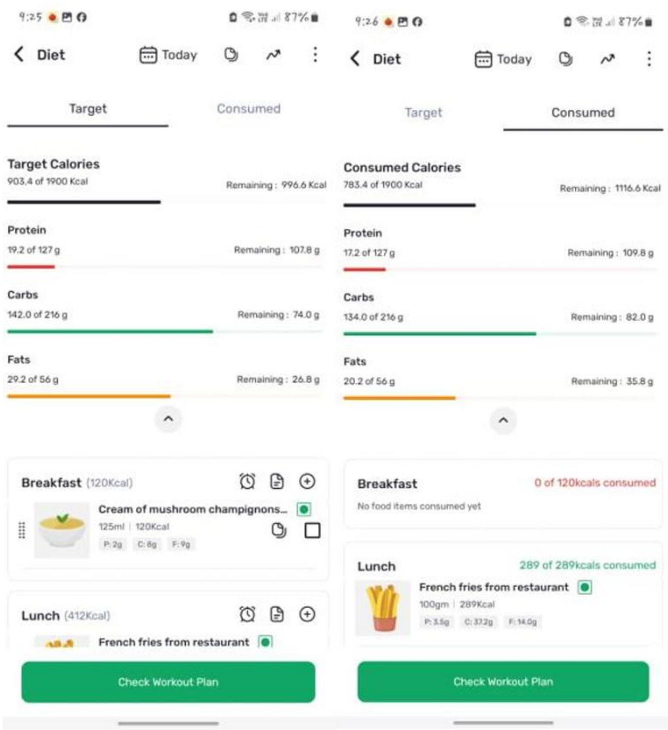


Figure 2.3.7 Dietary and food logging of FITTR

Based on Figure 2.3.7 above, FITTR allow users to plan their targeted meal plan for every day, calculate their targeted calories taken. Once user tick on the checkbox indicating that they have taken the meal, the system will calculate their consumed calorie with the macronutrients. This feature is helpful for users who wants to plan their dietary earlier, tracking how much is their targeted calories taken.

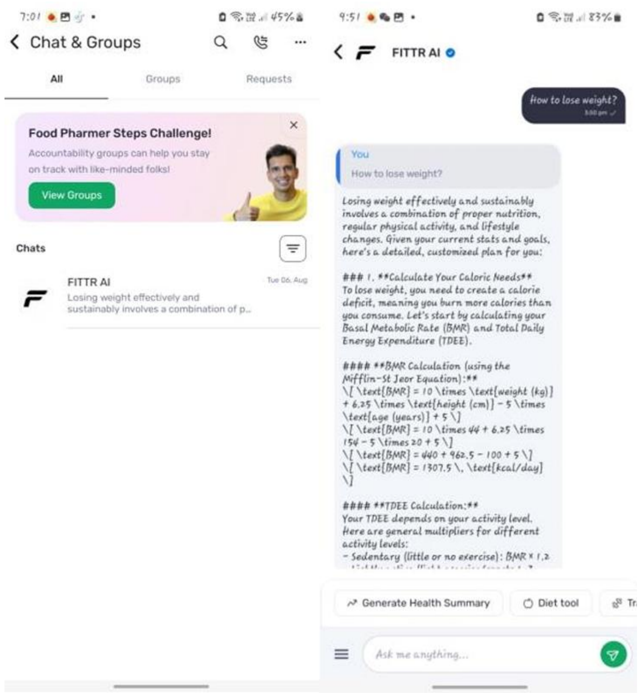


Figure 2.3.8 AI chatbot of FITTR

Other than the common features similar to other fitness app, FITTR has a standout feature which is the FITTR artificial intelligence (AI) chatbot. The FITTR AI chatbot offers personalised guidance providing workout recommendations or diet tips, and 24/7 real-time user feedback and support. Figure 2.3.8 as an example, asking question “How to lose weight?” to the FITTR AI chatbot, the chatbot reply in 5 detailed steps to user including calculate caloric needs with detail calculating steps, diet plan, exercise plan, lifestyle changes, and progress tracking. This AI chatbot can serve as a virtual fitness coach, providing user a personalised support and motivation to help users to achieve their health goals.

2.3.4 MyNetDiary

MyNetDiary [7] is a dietary planning application focuses on calorie counting and exercise tracker with over 20 million members worldwide. The application aims to reach their dietary goals by finding a diet that fits users’ lifestyle, allowing users to set their desired weight loss target, and track food, physical activities, and nutrients for better planning and scheduling.

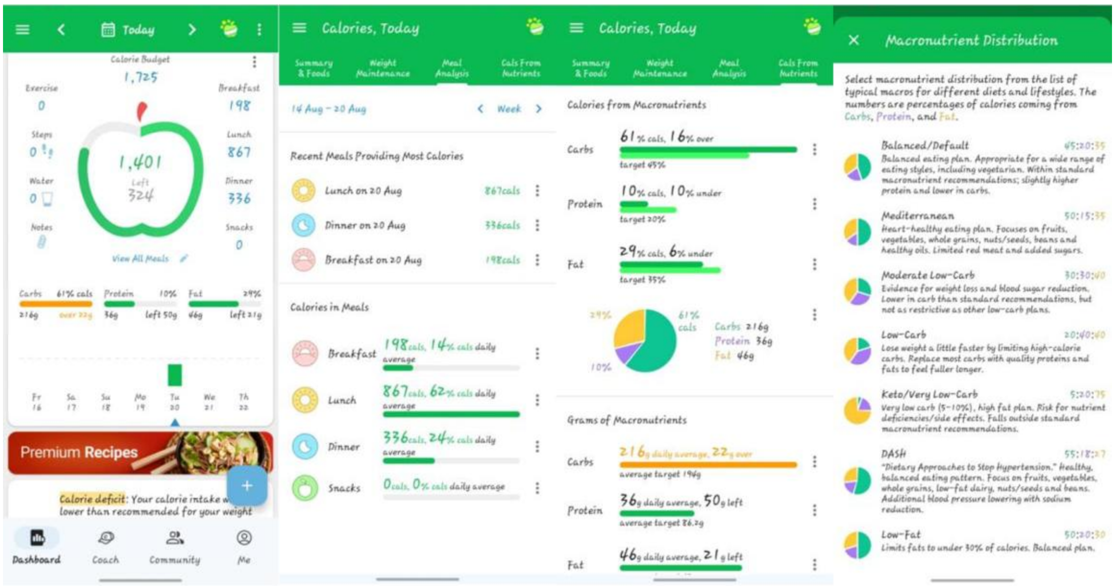


Figure 2.3.9 Calorie counting and tracking of MyNetDiary

The highlighted main feature of MyNetDiary is their calorie counting and tracking feature based on the user’s body mass index (BMI). As shown in Figure 2.3.9, MyNetDiary provide detailed calorie tracking, allowing users to log in their meals taken and calculate their calorie taken. Users are able to view the calorie they have taken for each meal, and the calories form the macronutrients. Users are allowed to customise their macronutrients distribution. As for example, users who wish for faster weight loss may need to follow the low-carb macronutrients distribution to limit high-calorie carb taken by replacing with proteins and fats. However, this feature only available for the premium users.

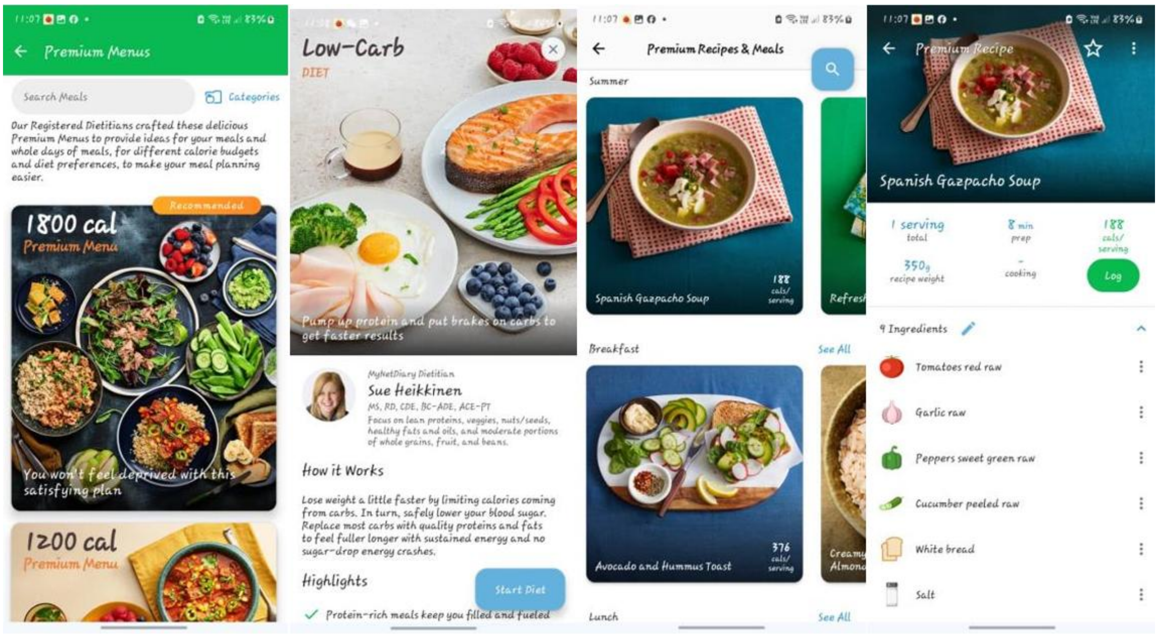


Figure 2.3.10 Dietary planner of MyFitnessPal

Based on Figure2.3.10 above, MyFitnessPal provide a comprehensive of dietary planner, from menu to recipe & meals. The application provides menu planned by their professional dietician, for various type of macronutrients distribution including low carb diet, high-protein diet, low-fat diet and so on. Additionally, the application provides various premium low-calorie recipes and meals, with detailed information such as preparation time, estimated calorie, estimated weight, ingredients needed and so on. This feature convenience to save user's time and effort, providing a well-balanced diet.

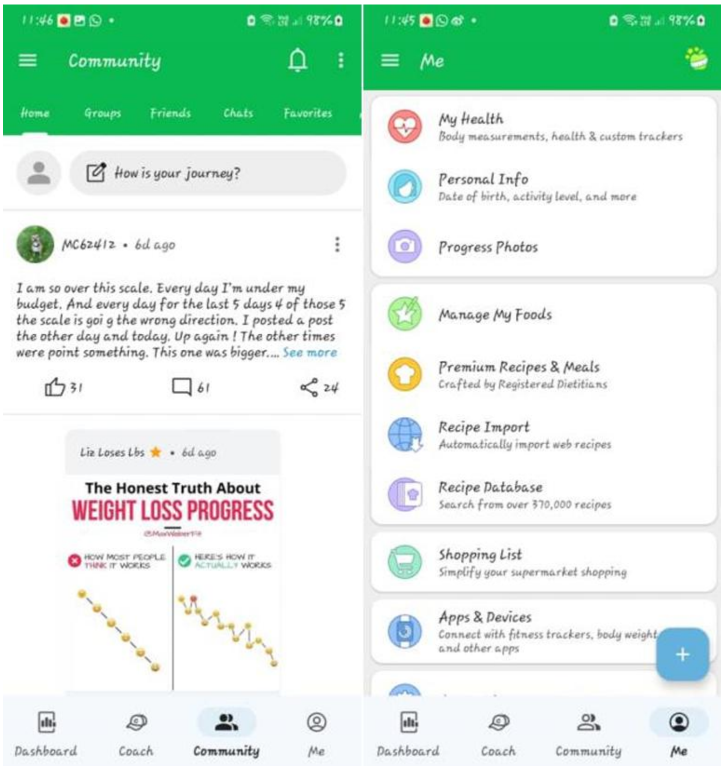


Figure 2.3.11 Additional features of MyNetDiary

Other than the main features, MyNetDiary also provide additional features such as community as illustrated in Figure 2.3.11. Users are allowed to discuss and share their thoughts through the community, joining groups, inviting friends, chat with others, and ask question to the registered dietician (RD). The application also contains features in user profile, allow user to track their body measurements, health and custom data, import recipe from the web, search recipe over the 370,000-recipe database, and so on. The additional feature available make MyNetDiary a comprehensive platform as a dietary planning application.

2.4 Critical Analysis of Existing Related System

Table 2.4.1 Pros and cons of existing related system

Application	Pros	Cons
Strava	- Provide filter-able workout routes recommendation. - Having a strong community aspect, allowing users to participate in challenges.	- Focuses primarily on exercise tracking only, with minimal support for dietary tracking.
MyFitnessPal	- Have wide database of foods and nutritional information. - Offers calorie and macronutrient calculating and tracking.	- Does not provide suitable workout recommendations.
FITTR	- Provide meal plans suggested by nutrition experts. - Offers daily different workout plans. - Allows asking questions and chatting with its AI chatbot.	- Difficult to use interface. - Lack of fitness tracking and recommendation.
MyNetDiary	- Calculate the amount of calorie users can take based on their BMI. - Allow users to log and track their meals, including calories count and nutrition taken. - Allow users to choose their desired customised nutrition distribution.	- Does not suggest dietary meal recommendation. - Does not provide workout suggestions.

Strava [5] performs well in providing filterable exercise route recommendations, allowing users to easily find and select routes that suit their preferences and desired difficulty levels by different aspects including running distance, running elevation and so on. This application's major community feature encourages social connection with other users or friends, by allowing users to join or compete in challenges, and join clubs to find friends, discuss on workouts, or even motivate on another. However, Strava is major in exercise monitoring, with no assistance for diet tracking and recommendation. This constraint may be a disadvantage for users looking for a more comprehensive approach to their fitness journey, as they would need to use a separate app for diet and nutritional tracking and meal planning.

MyFitnessPal [9] has wide database of foods and nutritional information, which makes it an effective application for dietary planning and consumption tracking. It allows users to manage their dietary with the application's calorie and macronutrient calculation and tracking tools. This tool is helpful for users who wants to control their diet in a nutritionally balanced way. It also allows users to log their exercise done to calculate their consumed calories. However, MyFitnessPal lack of personalised workout suggestions. Users have to rely on outside sources for customised exercise routines. This might be a disadvantage for user who is finding for personalised workout recommendation.

FITTR [6] offers various important features, including nutrition expert recommended meal plans, and daily workout training plans, making it a flexible app for diet and fitness. This application also provides other features like body mass ratio (BMR) calculator based on user's BMI and body fat percentage, goal calculator based on user's current weight, desired weight, target week, and total energy expenditure (TEE). Moreover, it includes a highlighted feature which is the AI chatbot, which allows users to communicate and ask questions, to provide personalised assistance and direction for users. Although FITTR performs well in both dietary and workout planning, it lacks fitness tracking and personalised workout suggestions based on user's BMI or BMR. Furthermore, it has a complex layout which packs a lot of information and features into a single screen, which can make the interface appear cluttered. Users may find it difficult to locate specific features or information due to the overwhelming amount of content presented at once.

MyNetDiary [7] is an excellent tool for users who want to keep track of their nutrition taken in detail. This application calculates the number of calories that users can consume depending on their BMI and allows them to log and track their meals, including calorie counts and other nutritional information. Users can also select their preferred customised macronutrient distribution, such as low-carb distribution, low-fat distribution and so on. Furthermore, it provides various premium dietary plans that coached by different dietician experts. Additionally, this application does allow users to find and log their exercise, and to calculate the calories consumed. However, this application obviously focuses on the dietary recommendation only, and it does not provide any workout or exercise recommendation for users.

2.5 The Mifflin St. Jeor Equation

The Mifflin St. Jeor equation is one of the most widely used equations for calculating an individual's Basal Metabolic Rate (BMR), which represents the number of calories the body requires to maintain essential physiological functions [27]. This equation was first introduced by Mifflin and has become a standard in nutritional science due to its accuracy across a population of healthy adults [28].

The equations are defined as follows in e.q (2.1) and e.q (2.2), where weight is measured in kilograms, height in centimetres, and age in years:

- For males:

$$BMR = (10 \times weight) + (6.25 \times height) - (5 \times age) + 5$$

e.q (2.1)

- For females:

$$BMR = (10 \times weight) + (6.25 \times height) - (5 \times age) - 161$$

e.q (2.2)

Compared to other formulas such as the Harris-Benedict equation, the Mifflin St. Jeor equation is considered to provide more accurate predictions of resting energy expenditure. For this reason, it is commonly used in both clinical and fitness application. [28]

2.6 The Exercise Calorie Calculation Equation

In addition to calculating daily calorie requirement through BMR, it is essential to estimate the calories expended during physical activities. Exercise energy expenditure can be calculated using the Metabolic Equivalent of Task (MET) formula, which quantifies the intensity of physical activities relative to resting energy expenditure [29]. The general equation [30] for calculating exercise calories is shown in e.q (2.3):

$$Exercise\ Calories = \frac{(MET\ level\ of\ activity \times 3.5 \times Weight(kg) \times duration(minutes))}{200}$$

e.q (2.3)

Where MET level of activity represents the intensity of the exercise, with 1 MET defined as the energy cost of sitting quietly at reast, Weight is the body weight of the individual in kilograms, and duration is the duration of the exercise session in minutes.

This formula is widely recognized and is recommended by the American College of Sports Medicine (ASCM) for estimating caloric expenditure during various forms of physical activity. The MET values for different exercises can be referenced from standardized compendiums such as the *Compendium of Physical Activities*. [29]

Chapter 3
System Design

This chapter highlights the system design of the project. It explains the overall design of the proposed system, providing a top-down view of the system, beginning with the system architecture, use case diagram, activity diagram, block diagrams, and flow diagrams, followed by specifications of each system component, the design of the database architecture, and the interaction between components

3.1 System Architecture

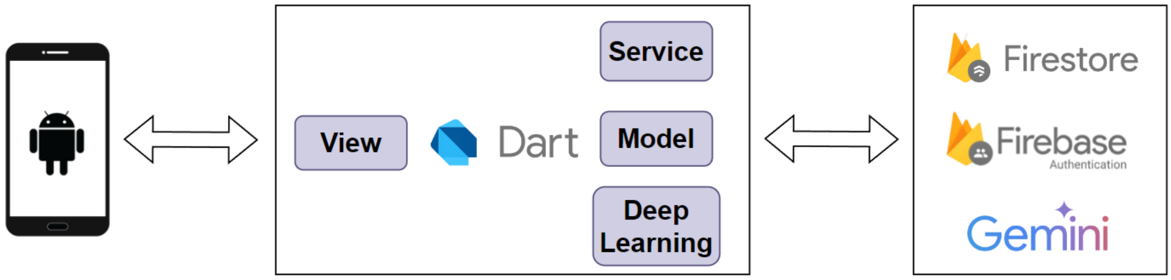


Figure 3.1 System Architecture Diagram

The Figure 3.1 above represents the high-level architecture of the proposed project, an Android based mobile application developed with Flutter and integrated with Firebase for backend services and deep learning (DL) for personalised predictions.

This architecture follows the Model-View-Controller (MVC) design pattern and is structured to ensure maintainability, scalability, and integration of artificial intelligence features.

In the MVC architecture, which is the core of the application. This layer contains the main logic of the application, developed using the Dart programming language. It is divided into multiple functional layers for clarity and modularity:

Table 3.1 MVC architecture overview

View	Handles the User Interface (UI) to display data such as user registration. The view layer is important to collect input for the users and output their goal to them through the view layer.
Service	Acts as a mediator between the view and backend. This layer manages API calls to Firebase Authentication and Firestore Database to perform backend data service.
Model	Defines the structure of logic of application data, which is used to perform data parsing or basic validation. This layer ensures a clean separation of data and logic.
Deep Learning	Responsible for processing user input and generating predictions. The deep learning model is run locally using TensorFlow Lite, by taking data from the model layer and sends predictions back to the service or view for display.

Firebase is used in this project to handle core backend operations, such as user authentication and cloud data storage. Firebase Authentication handles secure user registration and login, which may support various methods including email/password, Google sign-in, or other authentication providers. Firestore Database is a Cloud-based NoSQL database that stores real-time data such as user profiles. The Gemini Application Programming Interface (API) is integrated to provide advanced Artificial Intelligence (AI) powered functionalities, including chatbot interactions and nutrition analysis of custom meals.

3.2 Use Case Diagram and Description

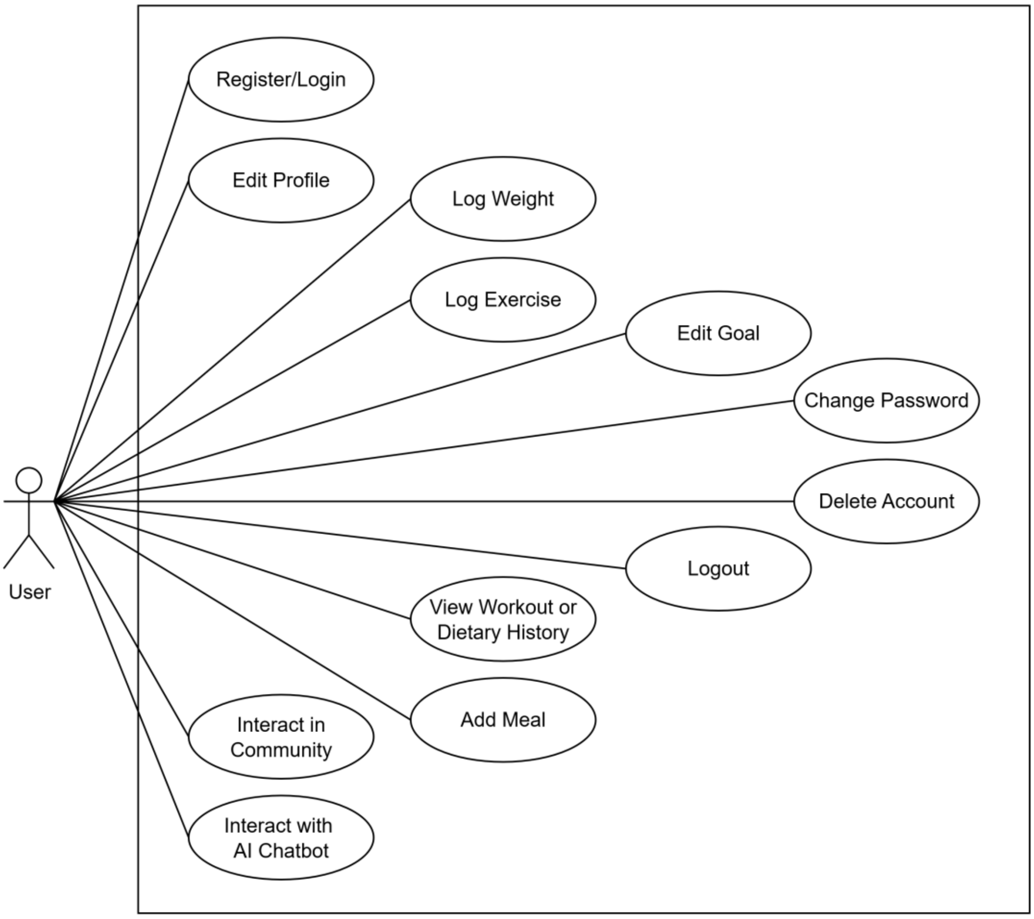


Figure 3.2 Use Case Diagram of the proposed system

Figure 3.2 illustrates the use case diagram of the proposed system. It outlines the interactions between the actor and the system, highlighting the key functionalities available to the actor. The diagram serves as a conceptual framework that guides the development and detailed explanation of the use case descriptions presented in the following section.

3.2.1 Use Case Description

Table 3.2 Use Case Description for "Register/Login"

Use Case ID	UC001	**Version**	1.0
Use Case	Register / Login		
Purpose	To authenticate users and provide access to their account		
Actor	User		
Trigger	User enters the application		
Precondition	-		
Scenario Name	**Step**	**Action**	
Main Flow	1	User enters the application	
	2	System displays welcome screen with options of "Register" or "Login"	
	3	User selects either: • Register: User enters required details (first and last name, email, password) • Login: User enters registered email and password	
Sub Flow – User is already logged in	2a.1	System checks whether user is already logged in	
	2a.2	System directs user to the homepage	
Sub Flow - Profile Completion for User Registration	3a.1	System validates whether the email address is taken for other account user	
	3a.2	System direct user to profile completion after registered with an account	
	3a.3	User fills in basic personal information (height, weight, age, gender)	
	3a.4	System validates input fields	
	3a.5	User selects "Continue"	
	3a.6	System analyses user's BMI category and display suitable goal using pre-trained deep learning model.	
	3a.7	User is allowed to select their desired goal if their BMI category falls under "normal" category	
	3a.8	User selects "Continue"	
	3a.9	System directs user to the homepage	

Sub Flow – Login to Existing Account	3b.1	System validates the credentials against stored account records
	3b.2	System directs user to the homepage
Alternate Flow - Invalid fields	3b.1.1	System displays error message
	3b.1.2	Return to Main Flow Step 3(Login) / Sub Flow 3a.3
Alternate Flow – Email is Already Taken	3a.5.1	System validated that email address is taken
	3a.5.2	System displays error message
	3a.5.3	User is required to re-enter email
Rules	1. A new account can only be created with an email that does not already exist in the system 2. Password must be at least 8 characters. 3. The system uses deep learning model to predict user BMI based on entered height, weight, gender, and age 4. If the user is already logged in, the system should bypass the login screen and redirect to the homepage	

Table 3.3 Use Case Description for "Edit Profile"

Use Case ID	UC002		Version	1.0
Use Case	Edit Profile			
Purpose	To allow users to edit their account profile			
Actor	User			
Trigger	User selects "Edit Profile" under profile settings			
Precondition	User is logged into the system			
Scenario Name	**Step**	**Action**		
Main Flow	1	User selects "Edit Profile" under profile settings		
	2	User edits profile		
	3	System validates input fields		
	4	User selects "Save Changes"		
	5	System update user's profile to the database		
Alternate Flow - Invalid Input	3a.1	System displays error message		
	3a.2	User is required to re-enter input		
Rules	1. User can only edit their own account profile 2. All mandatory fields must be filled before saving changes			

Table 3.4 Use Case Description for "Log Weight"

Use Case ID	UC003	**Version**	1.0
Use Case	Log Weight		
Purpose	To allow users to update their weight data		
Actor	User		
Trigger	User selects "Log Weight" under the weight dashboard from the homepage		
Precondition	User is logged into the system		
Scenario Name	**Step**	**Action**	
Main Flow	1	User selects "Log Weight" under the weight dashboard from the homepage	
	2	User edits current weight input field	
	3	System validates input field	
	4	User selects "Save Weight"	
	5	System updates user's weight to the database	
	6	System navigates user back to the weight dashboard	
Alternate Flow - Invalid Input	3a.1	System displays error message	
	3a.2	User is required to re-enter input	
Rules	1. System should recalculate user's suggested daily calorie intake 2. System should re-predict user's BMI category 3. System should regenerate suitable workout recommendation once BMI category changes due to new weight log		

Table 3.5 Use Case Description for “Log Exercise”

Use Case ID	UC004		**Version**	1.0
Use Case	Log Exercise			
Purpose	To allow users to log exercise based on given exercise list			
Actor	User			
Trigger	User selects “Log Exercise” from the workout page			
Precondition	User is logged into the system			
Scenario Name	**Step**	**Action**		
Main Flow	1	User selects “Log Exercise” from the workout page		
	2	System displays log exercise page with available exercise list		
	3	User selects exercise start time, exercise type, and fills in the exercise duration		
	4	System validates exercise duration validity		
	5	System calculates the estimated calories burned based on the exercise calories formula		
	6	User selects “Log Workout”		
	7	System saves exercise to the database		
	8	System displays confirmation message and navigates user back to the workout page		
Alternate Flow - Invalid Input	4a.1	System displays error message		
	4a.2	User re-enters valid exercise duration		
Rules	1. User must log only exercises they have personally performed. 2. The system must update the user’s workout history with the new log entry			

Table 3.6 Use Case Description for "Edit Goal"

Use Case ID	UC005	**Version**	1.0
Use Case	Edit Goal		
Purpose	To allow users to edit their goal		
Actor	User		
Trigger	User selects "Edit Goal" from the weight dashboard		
Precondition	User is logged into the system		
Scenario Name	**Step**	**Action**	
Main Flow	1	User selects "Edit Goal" from the weight dashboard	
	2	System displays the edit goal page	
	3	User selects their desired goal	
	4	User selects "Save Goal"	
	5	System updates user's goal to the database	
	6	System displays confirmation message and navigates user back to the weight dashboard	
Sub Flow – User's BMI category falls outside "normal"	3a.1	System does not allow user to edit their goal	
Rules	1. Only users with a "normal" BMI category are allowed to edit their goal 2. System should recalculate user's suggested daily calorie intake		

Table 3.7 Use Case Description for "Change Password"

Use Case ID	UC006	**Version**	1.0
Use Case	Change Password		
Purpose	To allow users to change the account password		
Actor	User		
Trigger	User selects "Change Password" in profile setting		
Precondition	User is logged into the system		
Scenario Name	**Step**	**Action**	
Main Flow	1	User selects "Change Password" in profile setting	
	2	User enters current password, new password and confirm password	
	3	User selects "Change Password"	
	4	System validates user input	
	5	System updates user password to firebase authentication	
	6	System displays confirmation message and navigates user back to profile setting	
Alternate Flow – Password Less Than 8 Characters/ Confirm Password Do Not Match/ Current Password Wrong	5a.1	System displays error message	
	5a.2	User is required to re-enter input fields	
Rules	-		

Table 3.8 Use Case Description for "Delete Account"

Use Case ID	UC007	**Version**	1.0
Use Case	Delete Account		
Purpose	To allow		
Actor	User		
Trigger	User selects "Delete Account" in profile setting		
Precondition	User is logged into the system		
Scenario Name	**Step**	**Action**	
Main Flow	1	User selects "Delete Account" in profile setting	
	2	System displays delete account page	
	3	User enters current account password	
	4	User type confirmation phrase	
	5	User tick required acknowledgement	
	6	User selects "Delete My Account Forever"	
	7	System displays final warning message	
	8	User selects "Yes, Delete My Account"	
	9	System validates user password, confirmation phrase, and acknowledgement	
	10	System deletes user account	
	11	System displays confirmation message and navigates user to registration/login page	
Alternate Flow - Invalid Password/ Invalid confirmation phrase/ Not acknowledged	10a.1	System displays error message	
	10a.2	User is required to enter valid password/ enter valid confirmation phrase / tick acknowledgement	
Rules	1. Once deleted, the account and all associated data are permanently removed and cannot be recovered		

Table 3.9 Use Case Description for "Logout"

Use Case ID	UC008	**Version**	1.0
Use Case	Logout		
Purpose	To allow users to logout of current account		
Actor	User		
Trigger	User selects "Log Out" from profile setting		
Precondition	User is logged into the system		
Scenario Name	**Step**	**Action**	
Main Flow	1	User selects "Log Out" from profile setting	
	2	System ask confirmation for account logout	
	3	User selects confirm	
	4	System logout current account and navigate user to login page	
Rules	1. System should remove user saved credentials to avoid automatically login		

Table 3.10 Use Case Description for "Add Meal"

<table>
<tr><td>Use Case ID</td><td colspan="2">UC009</td><td>Version</td><td>1.0</td></tr>
<tr><td>Use Case</td><td colspan="4">Add Meal</td></tr>
<tr><td>Purpose</td><td colspan="4">To allow users to add meal to their today's meal</td></tr>
<tr><td>Actor</td><td colspan="4">User</td></tr>
<tr><td>Trigger</td><td colspan="4">User selects "Add Meal" from the diet page</td></tr>
<tr><td>Precondition</td><td colspan="4">User is logged into the system</td></tr>
<tr><td>Scenario Name</td><td>Step</td><td colspan="3">Action</td></tr>
<tr><td rowspan="6">Main Flow</td><td>1</td><td colspan="3">User selects "Add Meal" from the diet page</td></tr>
<tr><td>2</td><td colspan="3">System displays add meal page</td></tr>
<tr><td>3</td><td colspan="3">User selects mealtime and adds meal</td></tr>
<tr><td>4</td><td colspan="3">User selects either to add meals from the available recipe library or customise their meal</td></tr>
<tr><td>5</td><td colspan="3">System validates whether the mealtime is added with meal</td></tr>
<tr><td>6</td><td colspan="3">System adds the meal to database</td></tr>
<tr><td rowspan="4">Sub Flow – User Selects Meals from Recipe Library</td><td>4a.1</td><td colspan="3">User selects "Recipe Library" from the tab</td></tr>
<tr><td>4a.2</td><td colspan="3">System displays list of recipes from available recipe library</td></tr>
<tr><td>4a.3</td><td colspan="3">User selects add button on desired meal</td></tr>
<tr><td>4a.4</td><td colspan="3">Return to Main Flow Step 5</td></tr>
<tr><td rowspan="5">Sub Flow – User Customise their Meal</td><td>4b.1</td><td colspan="3">User selects "Custom Meal" from the tab</td></tr>
<tr><td>4b.2</td><td colspan="3">System displays meal customisation screen</td></tr>
<tr><td>4b.3</td><td colspan="3">User enters the meal name</td></tr>
<tr><td>4b.4</td><td colspan="3">User can choose either to analyse nutrition values using AI or manually insert nutrition values. If user selects analyse using AI, system should pass meal name to API to analyse nutrition breakdown, and display on the screen</td></tr>
<tr><td>4b.5</td><td colspan="3">Return to Main Flow Step 5</td></tr>
<tr><td rowspan="3">Alternate Flow – Selected Mealtime is added with meal</td><td>6a.1</td><td colspan="3">System displays replace message and ask user do they want to replace with new meal</td></tr>
<tr><td>6a.2</td><td colspan="3">User selects either replace or cancel</td></tr>
<tr><td>6a.3</td><td colspan="3">Return to Main Flow Step 3 / Step 5</td></tr>
<tr><td>Rules</td><td colspan="4">1. Each mealtime must have only one meal entry (if a new meal is added, the system must ask whether to replace the existing one)</td></tr>
</table>

Table 3.11 Use Case Description for "View Workout or Dietary History"

<table>
<tr><td>Use Case ID</td><td colspan="2">UC010</td><td>Version</td><td>1.0</td></tr>
<tr><td>Use Case</td><td colspan="4">View Workout or Dietary History</td></tr>
<tr><td>Purpose</td><td colspan="4">To allow users to view their history on workout and diet</td></tr>
<tr><td>Actor</td><td colspan="4">User</td></tr>
<tr><td>Trigger</td><td colspan="4">User selects 'calendar' button on workout or dietary page</td></tr>
<tr><td>Precondition</td><td colspan="4">User is logged into the system</td></tr>
<tr><td>Scenario Name</td><td>Step</td><td colspan="3">Action</td></tr>
<tr><td rowspan="2">Main Flow</td><td>1</td><td colspan="3">User selects 'calendar' button on workout or dietary page</td></tr>
<tr><td>2</td><td colspan="3">System displays workout or dietary history:

<u>For workout history</u>
System displays workout history with statistical data and filterable tags (by exercise category and time)
<u>For dietary history</u>
System displays workout history with statistical data, filterable by custom date, weekly, or monthly</td></tr>
<tr><td>Rules</td><td colspan="4">System must retrieve and display accurate historical data from the database
If no history is available, the system must display an empty state message (e.g., "No history found")</td></tr>
</table>

Table 3.12 Use Case Description for "Interact in Community"

Use Case ID	UC011		**Version**	1.0
Use Case	Interact in Community			
Purpose	To allow user to interact with other users in community page			
Actor	User			
Trigger	User selects "Community" from the bottom navigation bar			
Precondition	User is logged into the system			
Scenario Name	**Step**	**Action**		
Main Flow	1	User selects "Community" from the bottom navigation bar		
	2	System displays the community page		
	3	User selects the 'add' button on the bottom right corner to post new feed		
	4	System displays create post page		
	5	User selects desired channel to post		
	6	User writes on their thoughts		
	7	User clicks on "Share" button		
	8	System store user's post to the database		
	9	System displays confirmation message and navigates user back to the community page		
Sub Flow – Like and Comment	3a.1	User likes or comment on posts by hitting the like button or comment button to write a comment		
Sub Flow – View Own Posts	3b.1	User selects 'card' icon on the top right corner of community page		
	3b.2	System displays all posts posted by the user with likes and comments count		
Rules	1. User can only post under one channel at a time 2. User can only delete own posts			

Table 3.13 Use Case Description for "Interact with AI Chatbot"

Use Case ID	UC012		**Version**	1.0
Use Case	Interact with AI Chatbot			
Purpose	To allow user to interact with the AI Chatbot			
Actor	User			
Trigger	User selects "Ask AI" widget from the homepage			
Precondition	User is logged into the system			
Scenario Name	**Step**	**Action**		
Main Flow	1	User selects "Ask AI" widget from the homepage		
	2	System display the 'FitFuel AI' page		
	3	User enters prompt and sends		
	4	System responds to the user's prompt		
Alternate Flow – Irrelevant Prompt	3a.1	User enters prompt that is irrelevant with fitness, nutrition, or wellness.		
	3a.2	System responds with: "I'm specialized in fitness and nutrition advice. Please ask me about workouts, exercise routines, diet plans, meal planning, or health-related nutrition questions!"		
Rules	1. All chatbot interactions must be displayed in a conversational format (user prompt → chatbot response)			

3.3 Activity Diagram

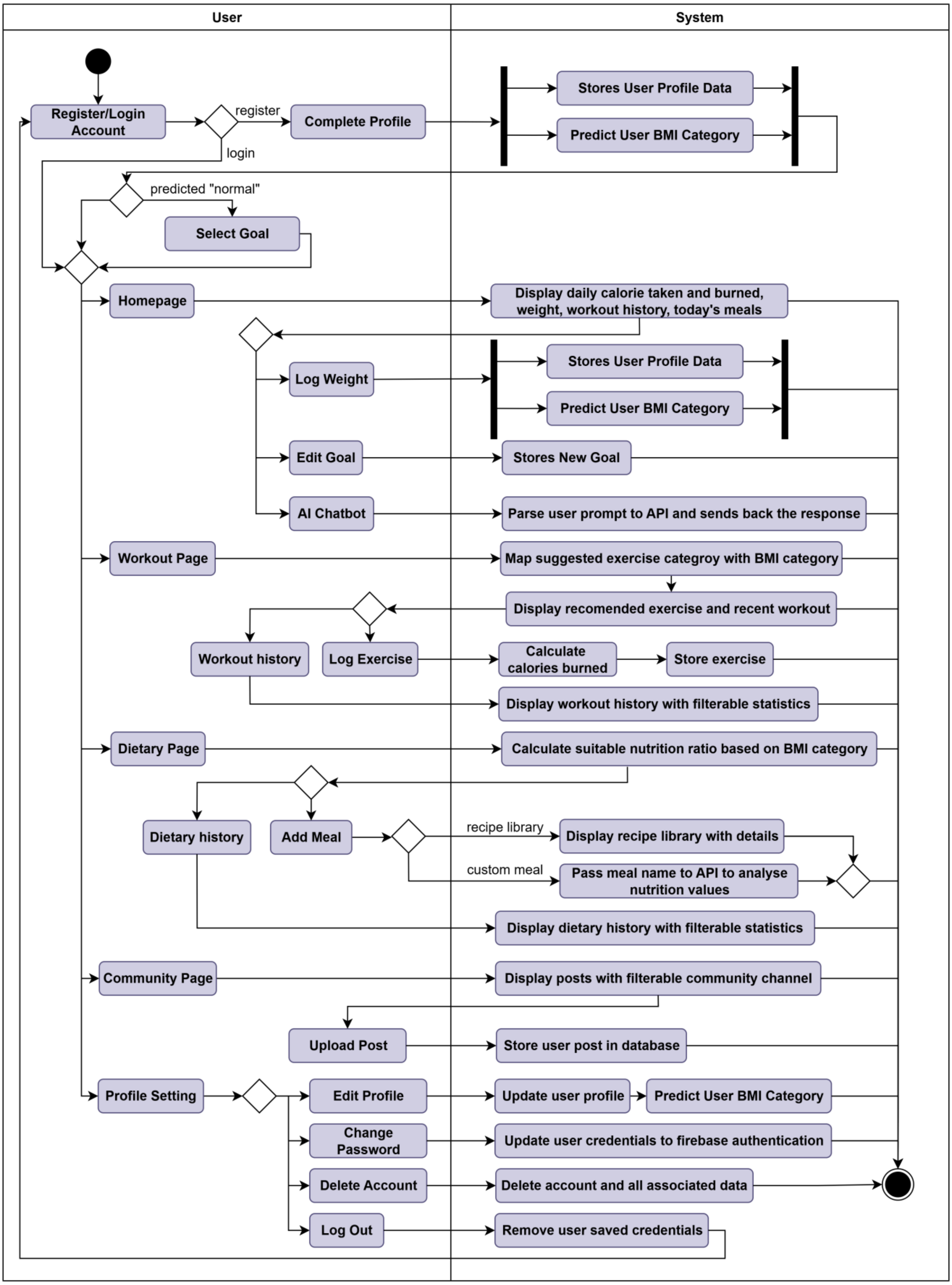


Figure 3.3 Activity Diagram of the proposed system

The activity diagram in Figure 3.3 provides a detailed representation of the operational flow within the proposed system. It models the sequence of activities performed by both the user and the system, structured into two swim lanes. This separation of responsibilities shows the interaction between user actions and system processes.

Once a user has registered an account using an email address and password, the user proceeds to complete their profile by providing essential information such as gender, age, weight, and height. The system will then store the user profile data in the database and initiates the prediction of the user's BMI category using a pre-trained deep learning model. Based on the predicted BMI category, users are directed to different goal-setting screens. Users with a *normal* BMI category are free to select their goal (lose weight, maintain weight, or gain weight), whereas users classified as *underweight* are automatically assigned to gain weight, and those classified as *overweight* are assigned to lose weight.

Upon completion of the goal-setting stage, users are redirected to the homepage of the application, which functions as the central navigation page. The homepage connects with a bottom navigation bar that grants access to the workout page, dietary page, and the community page. The profile settings menu is accessible via the top-right corner of the homepage.

Within the homepage itself, three primary functionalities are available. First, users may log their weight, which allows the system to update the stored data, and re-predict the user's BMI category. Secondly, users with a *normal* BMI category are permitted to edit their goal. Lastly, users may interact with an Artificial Intelligence (AI) chatbot, which is supported by the Gemini API (gemini-2.0-flash). This feature enables users to submit prompts related to fitness, diet, or general wellness, which are parsed by the system to the API and returned with contextually relevant responses.

The workout page presents users with a list of recommended exercises, that are mapped according to the user's BMI category, selected goal, and suggested exercise categories. Based on these mappings, suitable exercises are displayed. Users may log exercises by selecting an activity and specifying the duration. The system will then estimate the calories burned by calculating using predefined formula and stores the data into the database. A workout history feature allows users to review past exercise records and filter them for statistical analysis.

The dietary page works by calculating a suitable daily nutritional distribution based on the user's goal. Within the add meal function, users can either select meals from a predefined recipe library or create a custom meal entry. For customised meals, the system offers two approaches: manual input of nutritional values or automated nutritional analysis via the Gemini API. The analysis provides estimates for calories, proteins, carbohydrates, and fats. In the dietary page, users may also access their dietary history, with options to filter and analyse their nutritional intake over time.

The community page serves as a social engagement platform within the application. Here, users can view posts categorised into community channels, such as general community, fitness enthusiasts, and healthy nutrition. Users may also upload posts, as well as interact with others through likes and comments, thereby fostering peer engagement and motivation.

Finally, the profile settings section provides four key functionalities: edit profile, change password, delete account, and log out. The edit profile function enables users to update their personal information such as first and last name, age, gender, and height. This will prompt the system to re-predict the user's BMI category based on the updated personal information. The change password function will update user credentials securely through Firebase Authentication. The delete account option ensures that all user data and associated records are permanently removed from the database. The log out function clears saved user credentials, thereby preventing automatic re-login from prior sessions.

In summary, the activity diagram shows the entire lifecycle of the user interaction, from registration to ongoing fitness and dietary management, supported by backend systems. This ensures that the application provides a data-driven experience that aligns with users' health goals while also offering opportunities for community support and interaction.

3.4 System Block Diagram

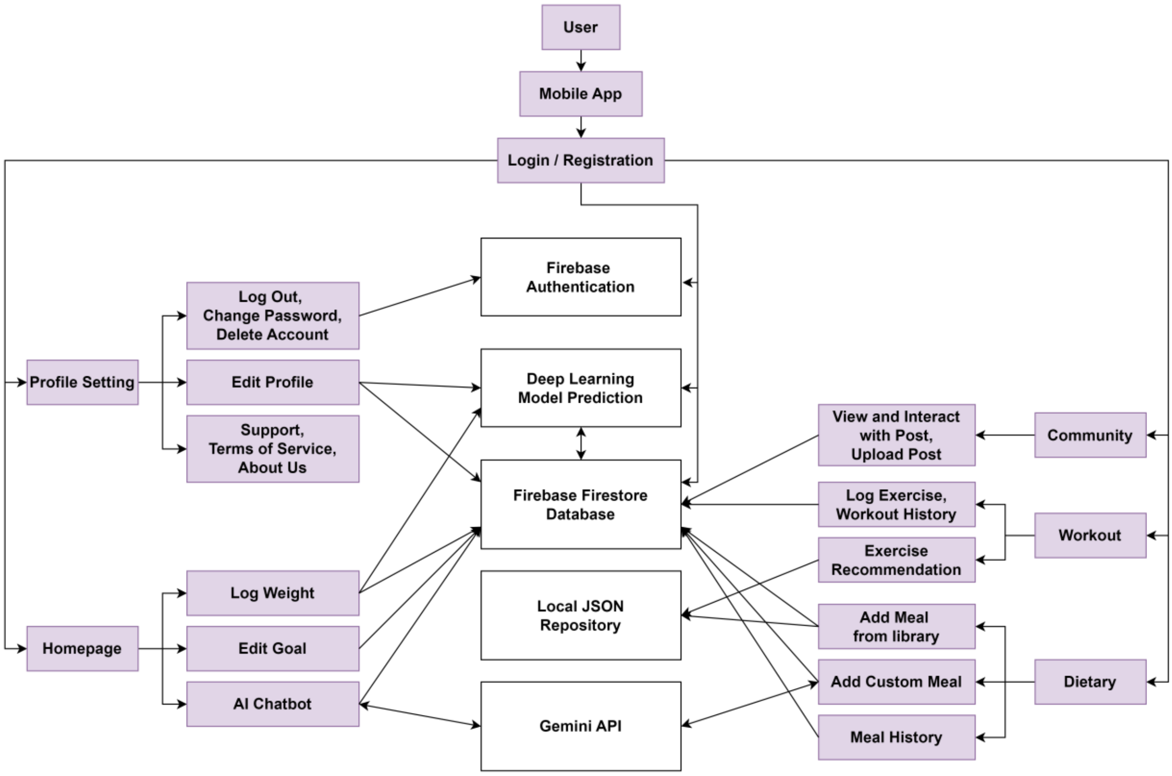


Figure 3.4 System Block Diagram

The system block diagram, as illustrated in Figure 3.4, provides an overview of the main components, functions, and data flows within the system. The design follows a modular architecture that integrates both frontend user interactions and backend services.

At the highest level, the system begins with the user, who interacts with the mobile application through the login and registration interface. The user registration process interacts with the deep learning model prediction and the Firestore Database. The User authentication is managed using the **Firebase Authentication**, which validates user credentials and ensures authorised access to the system.

Once authenticated, user profile data and subsequent activities are stored in the **Firebase Firestore Database**, which serves as the central database for user information, exercise logs, dietary records, and community interactions. The Firestore database operates in real time, ensuring that updates such as weight logs are immediately reflected in the application.

The system also integrates a **Deep Learning Model** that predicts the user's Body Mass Index (BMI) category based on personal data (age, gender, weight, and height). This model is invoked whenever a user updates their weight or profile details, and the prediction results are stored back into the Firestore database for subsequent use in goal setting, workout recommendations, and dietary calculations.

To support content-driven features, the system utilises a **Local JSON Repository**, which stores smalls size static datasets such as predefined recommended exercise categories based on user BMI category and goal, and recipe lists with nutrition breakdown. These repositories enable efficient retrieval of structured information without requiring constant remote API calls.

For advanced functionality, particularly natural language processing in AI chatbot and nutritional analysis, the system integrates the **Gemini API**. The AI chatbot uses Gemini to interpret user queries related to fitness, diet, and wellness, while the dietary module uses the same API to analyse custom meal entries, returning estimated calories, protein, carbohydrates, and fats.

Overall, the system block diagram demonstrates how the mobile application connects frontend user interactions with backend services and intelligent APIs, ensuring secure authentication and efficient data storage.

3.5 System Components Specifications

The system consists of multiple interconnected components that work together on user interaction, data processing, storage and external services. The main components are:

3.5.1 User Interface Components

Table 3.14 below shows the visible modules within the mobile application where the user interacts with the system.

Table 3.14 Function modules with description

Function Module	Description
Login/Registration	Provides account creation, authentication, and secure access to the system.
Homepage	Central hub that provides access to logging weight, editing goals, and interacting with the AI chatbot. It also connects to other function modules such as workout and dietary.

Workout	Supports exercise recommendations, exercise logging/deleting, calorie burn calculation, and workout history tracking.
Dietary	Enables users to add/delete meals from the recipe library or by creating custom meal, with dietary history and nutritional statistics accessible for analysis.
Community	Allows users to view, upload, and interact with posts categorised into channels, supporting engagement. It also sends notifications to the user once user receive interaction from other users.
Profile Settings	Offers functions such as editing profile information, changing passwords, deleting accounts, logging out, and accessing support pages including Support, Terms of Service and About Us.

3.5.2 Processing and Logic Components

This layer represents the computational core of the system, where user inputs are transformed into meaningful outputs through a combination of predictive modelling, decision rules, and data-driven calculations. The processing and logic components are described as in Table 3.15:

Table 3.15 Processing and logic components with description

Component	Description
BMI Prediction Logic	A pre-trained deep learning model to predict user's Body Mass Index (BMI). The model classifies users based on parameters such as height, weight, age, and gender into categories of severe thinness, mild thinness, moderate thinness, normal, overweight, obese, or severely obese. This prediction serves as the foundation for subsequent recommendation rules.
Goal Assignment Logic	The system incorporates rule-based logic to determine user's achievable goals according to their predicted BMI category. Users in the normal BMI range are permitted to select their own fitness goals (lose, maintain, or gain weight). By contrast, underweight users are restricted to a "gain weight" goal, while overweight users are assigned to a "lose weight" goal. This enforces medically appropriate guidance and safeguards against contradictory user decisions.
Calories Burn Processing	The workout module applies formula-based calculation to estimate calories burned during physical exercises. When a user logs an exercise along with its duration, the system utilises standard metabolic equivalent (MET)

	values to calculate energy expenditure. These results are stored as calorie burned in the database and contribute to the user's workout history and statistical analysis.
Exercise Recommendation Rules	A recommended exercise categories mapping rules aligns user BMI categories and goals. The mapping is supported by static data stored in local JSON repository, which contains predefined recommended exercise categories, and exercise lists. This logic enables the system to adapt its suggestions according to user-specific needs.
Suggested Nutrition Breakdown Processing	The dietary module computes a daily nutritional distribution tailored to the user's goal. Macronutrient allocations for calories, protein, carbohydrates, and fats are dynamically generated. Specifically: • Users with 'lose weight' goal: Target calories are set 10% lower than the calculated maintenance calories, with macronutrient ratios of 0.33 protein, 0.45 carbohydrates, and 0.22 fats. • Users with 'gain weight' goal: Target calories are set 10% higher than maintenance calories, with macronutrient ratios of 0.24 protein, 0.48 carbohydrates, and 0.28 fats. • Users with 'maintain weight' goal: Target calories are equivalent to maintenance calories, with macronutrient ratios of 0.28 protein, 0.44 carbohydrates, and 0.28 fats.
Meal Customisation	For custom meal entries, the Gemini API is invoked to automatically analyse nutritional values, whereas users also retain the option to input values manually. This ensures that dietary tracking remains both flexible and accurate.
Chatbot Query Handling	The Artificial Intelligence chatbot integrates with the Gemini API to provide interactive responses to user prompts. Queries related to fitness, dietary guidance, or general wellness are parsed and transmitted to the API. The system then processes the returned response to be displayed in the user interface.

3.5.3 Data Management and Backend Components

The data management and backend components are responsible for securely handling user data, maintaining application state, and supporting real-time interactions across the system. These components ensure that information is stored persistently, synchronised efficiently, and retrieved accurately whenever it is required. The major backend elements are described as in the Table 3.16 below:

Table 3.16 Data management and backend components with description

Components	Description
Firebase Authentication	Firebase Authentication provides a secure mechanism for user identity management. It supports functions such as registration, login, password reset, and account deletion. Credentials are verified using Firebase's cloud-based service, avoiding unauthorised access. Authentication tokens are generated and validated during each session, enhancing secure communication between frontend and backend.
Firebase Firestore Database	The system utilises Firebase Firestore, a cloud-based NoSQL database, to store structured user data in real time. Firestore manages information such as user profiles, weight logs, workout logs, meal logs, and community posts. Data is organised into collections and documents, allowing for scalability and efficient querying. Firestore's real-time synchronisation feature ensures that updates made by users are immediately reflected across devices.
Local JSON Repository	A static JSON repository is included within the application to store predefined content such as exercise lists, recommended exercise categories, and recipe libraries. The approach reduces dependency on external APIs while ensuring fast access to reference data.
Integration with External Services	The backend layer also interacts with external APIs such as Gemini. The Gemini API processes natural language queries and nutritional analysis requests, while the backend ensures smooth communication by formatting requests and storing relevant responses when necessary.

3.6 Circuits and Components Design

This section describes the detailed design of the system components, focusing on how the backend, storage, and user interface are structured and interact with one another. The design ensures proper data management, scalability, and integration between the different functional modules of the mobile application.

3.6.1 Firestore Database Design

The backend storage of the system is managed using Google Firestore, a NoSQL cloud database. The database is structured into two main top-level collections, which is the Users and Community. As shown in Figure 3.5 below, each of them contains documents and sub-collections that support the functionality of the application.

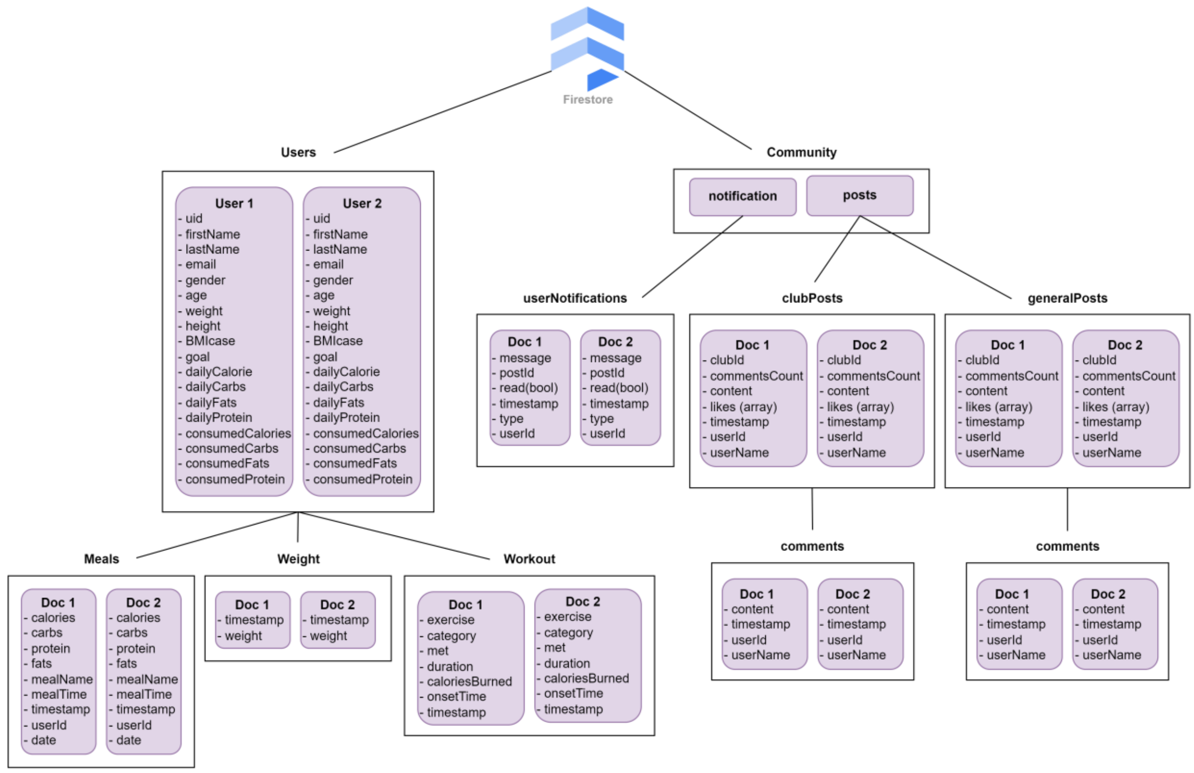


Figure 3.5 Firestore Database Design

The Users collection stores individual user profiles, identified by their unique user Id (uid). Each user document contains personal attributes such as age, gender, weight, height, first and last name. To support functional features, each user document includes 3 sub-collections:

- **Meals**: records daily meal intake with nutritional values for each meal
- **Weight**: stores timestamped user's weight history
- **Workout**: records user's exercise logs with calories burned

The Community collection manages user interactions and social features. It is divided into two main parts:

- **Notifications**: which contains the userNotifications sub-collection used to store alerts such as post likes and comments
- **Posts**: which consists of generalPosts and clubPosts sub-collections. Each post document stores content, author information, timestamp, likes and comments count. Posts also include a comments sub-collection for user replies.

This design supports scalability by separating personal data from community data, while maintaining a clear hierarchy of collections and sub collections. The use of sub-collections ensures efficient data retrieval, allowing the application to quickly query specific subsets of information, such as a user's meal history.

3.6.2 JSON Repository Design

The system integrates statis JSON files as lightweight repositories for predefined dataset or rules. These enable fast local access to structured information, reducing the need for frequent database queries or API calls. The 3 key JSON files used in the system are described as shown in Table 3.17 below:

Table 3.17 JSON Repository Design

File	Description	Example

Exercise list	This file stores a categorised list of exercises grouped into four categories: Cardiovascular, Strength Training, HIIT, and Low-Impact Exercise. Each entry specifies the exercise name and its corresponding Metabolic Equivalent of Task (MET) value, which is used to calculate estimated calories burned	"Cardiovascular": [
		{
		"name": "Jogging ",
		"met": 8.0
		}
		]

Exercise recommendation	This file defines the mapping between BMI categories, user goals, and recommended exercise categories. Each BMI categories is associated with one or more exercise categories tailored to user's health goal. This mapping serves as the foundation for the workout recommendation logic.	"normal": [
		{
		"goal": "lose weight",
		"exercise_category": "cardiovascular, HIIT"
		}
		]

Recipes	This file stores a library of recipes with detailed nutritional information. Each recipe entry contains metadata such as preparation time, cooking time, servings, and even step-by-step directions. Nutritional values, including calories, fats, carbohydrates, and protein, are also included to support meal tracking and logging.	{
		"name": "Chicken Stir-Fry",
		"prep_time": "20 mins",
		"cook_time": "20 mins",
		"total_time": "40 mins",
		"calorie": 344,
		"fats": 7,
		"carbs": 45,
		"protein": 25
		}

3.7 System Components Interaction Operations

This section explains how the different system components interact during the user operations. Each operation involves coordinated communication between the user interface, processing logic, and backend storage.

3.7.1 User Registration and Authentication

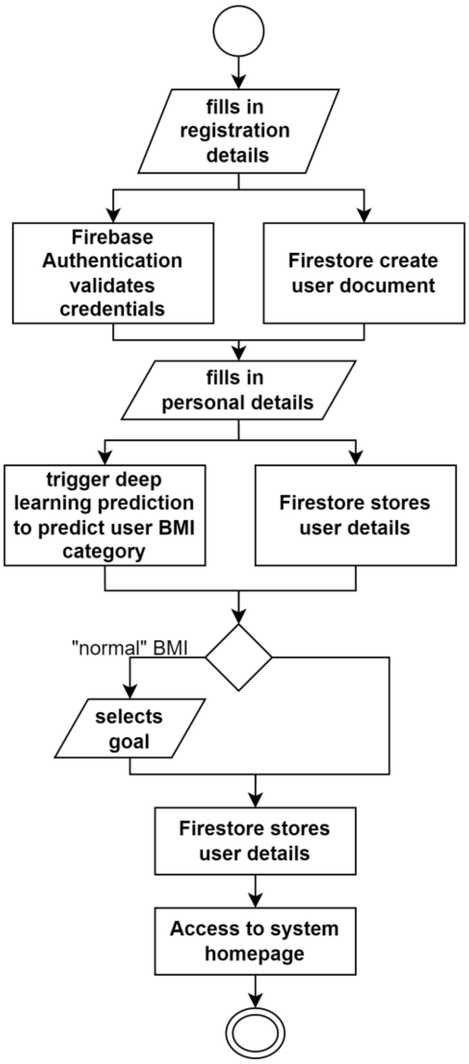


Figure 3.6 Flowchart of User Registration and Authentication

Figure 3.6 shows the flow of user registration and authentication. The process begins when a new user provides registration details such as email and password. The Firebase Authentication will validate the credentials whether the email is taken. If successful, a user document is created in Firestore.

Next, the user fills in personal details (age, gender, height, and weight). At this stage, the deep learning model is triggered to predict the BMI category, which is then stored in Firestore. If the BMI category is *normal*, the system prompts the user to select their desired goal (gain, maintain, or lose weight). This goal is stored alongside the BMI category. Finally, once all details are recorded, the user gains access to the system homepage.

3.7.2 Logging Weight

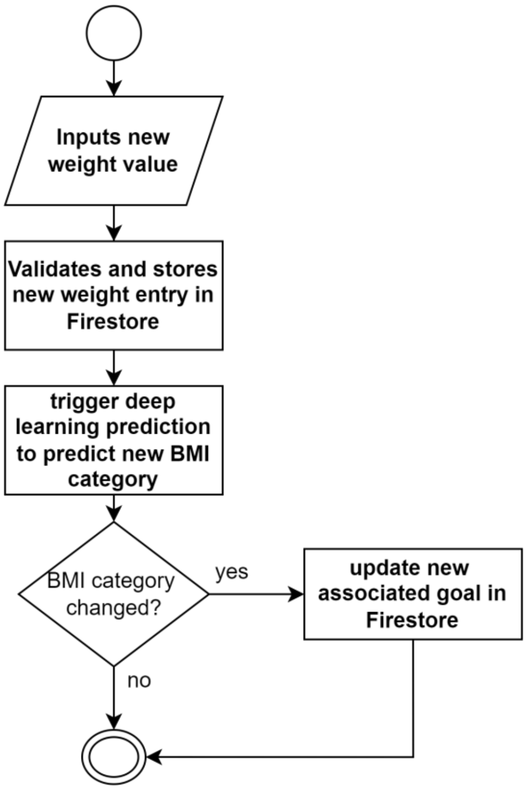


Figure 3.7 Flowchart of Logging Weight

Figure 3.7 shows the flow of logging new weight. The process begins with inputting new weight values by the user. The system will then validate the input and create a new weight document to store new weight timestamp in the weight collection in Firestore. Meanwhile, the system will trigger the deep learning model to re-predict user's latest BMI category using their new logged weight. If the BMI category changes, the system will assign new goal that is associated with their new BMI category.

3.7.3 Editing Goal

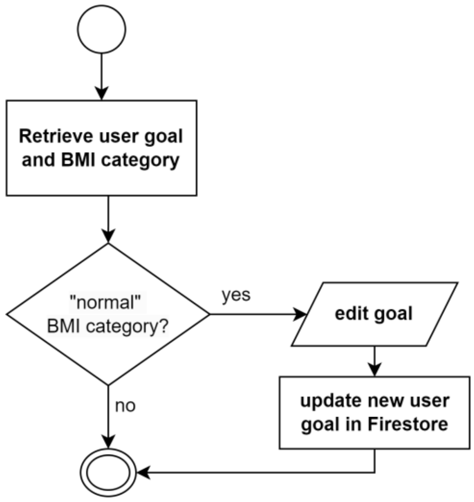


Figure 3.8 Flowchart of Editing Goal

Figure 3.8 shows the flow of editing goal. The system will first retrieve the user's current BMI category. If user's BMI category falls under "normal" category, users are allowed to edit their goal, and the system will update user's new goal to the Firestore. Else, users are not allowed to modify their goal.

3.7.4 AI Chatbot

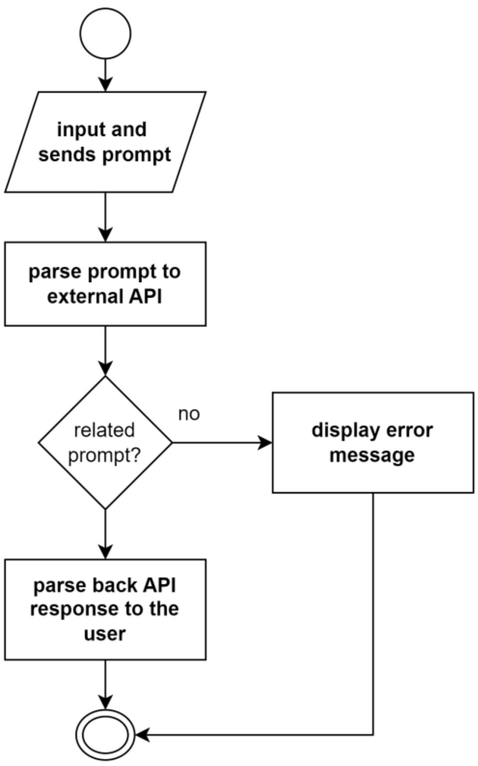


Figure 3.9 Flowchart of AI Chatbot

Figure 3.9 shows the flow of AI Chatbot interaction. The process starts with user input and then sends in the prompt to the system. The system will then parse the user input to the external Gemini API. If user's input is related to fitness, diet, or general wellness, the system will parse back the API response to the user. Else, the system should display error message to tell user to send in only fitness, diet, or general wellness related prompt.

3.7.5 Logging Exercise

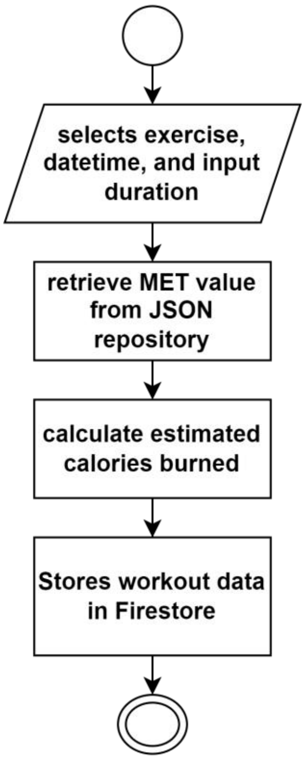


Figure 3.10 Flowchart of Logging Exercise

Figure 3.10 shows the flow of logging exercise. The process starts with selecting an exercise, datetime, and input exercise duration. The system will then calculate calories burned using formula: MET value × weight × duration / 200 and stores user workout data in Firestore under the workout collection.

3.7.6 Adding Meals

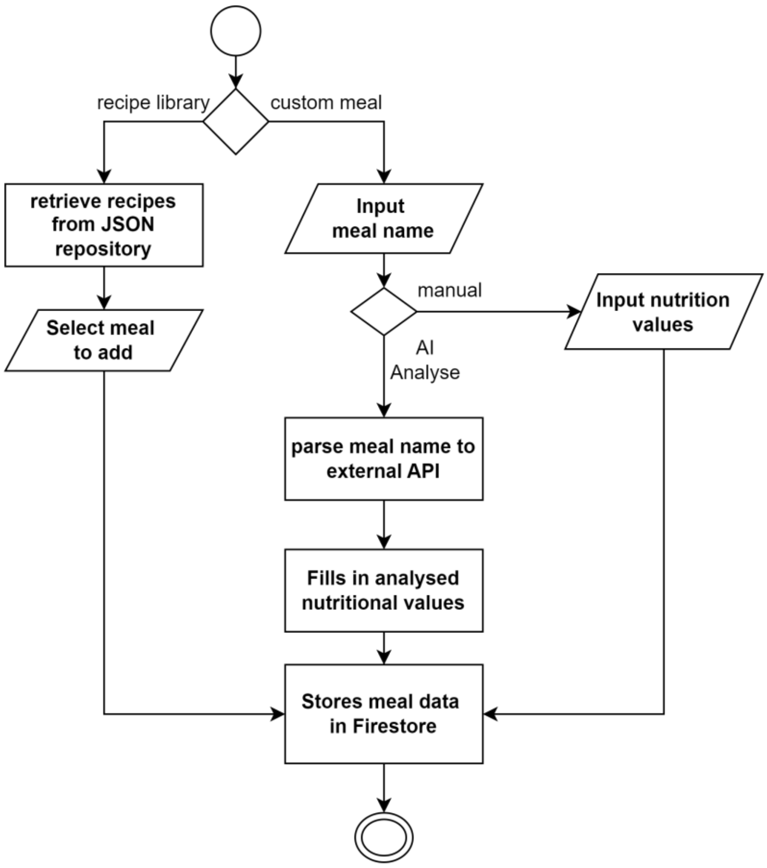


Figure 3.11 Flowchart of Adding Meals

Figure 3.11 shows the flows of adding meals. The process starts with allowing users to select whether they want to add meal from the available recipe library, or to customise their meal. For adding meals from the recipe library, the system will retrieve recipes from the JSON repository to allow users to select meal from the library. For meal customisation, users will first require inputting meal name. Then users may choose to manually input nutrition values or analyse nutrition values using AI. For AI nutrition analysis, the system will first parse the meal name to the external API. The system will then receive response from the external API, then automatically fills in the analysed nutritional values provided by the external API. Once all done, the meal data will be stored in Firestore, under the meals collection.

3.7.7 Community Interactions

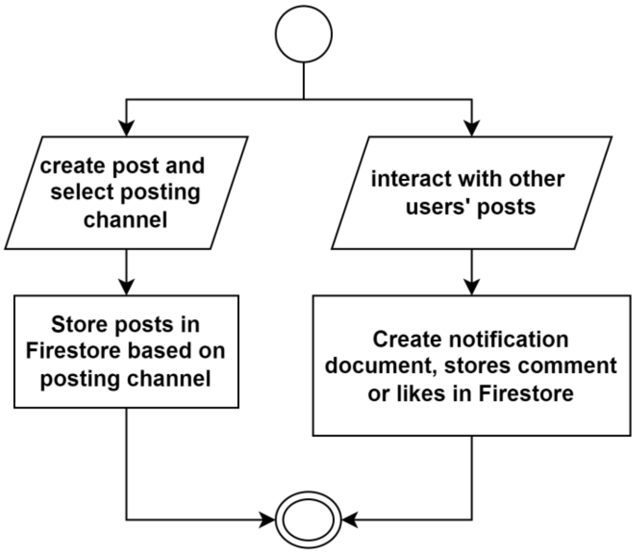


Figure 3.12 Flowchart of Community Interactions

Figure 3.12 shows the flows of community interactions. There are 2 main user functions in the community. Firstly, users may create post and select posting channel. The system will then store the user post in Firestore based on their selected posting channel. Secondly, users may interact with other user's post by tapping like or commenting. The system will then store the user interaction in Firestore and create notification documents to be sent to the post author.

Chapter 4

System Methodology/Approach

This chapter presents the methodology and approach used in the system, system specifications, and timeline of the project.

4.1 Methodology Used

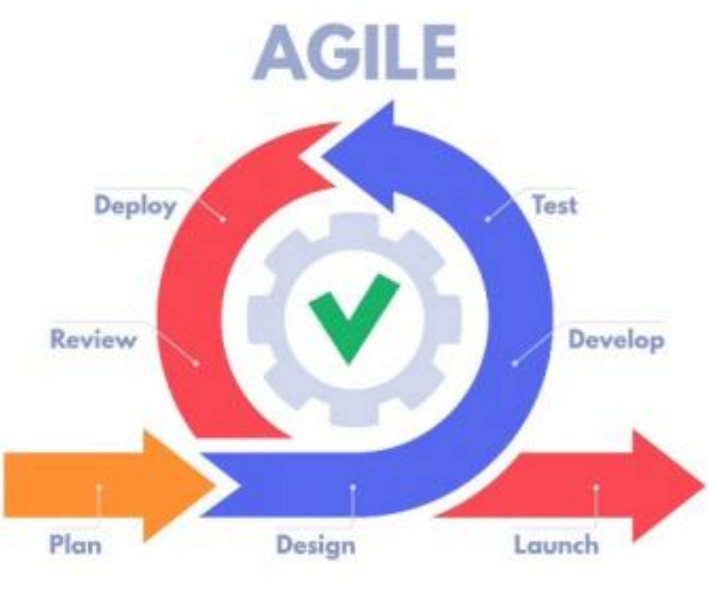


Figure 4.1 Stages in Agile Methodology

In this project, the system will follow the Agile methodology, complemented by the Software Development Life Cyle (SDLC) framework. Agile methodology is an iterative and incremental approach to software development that priorities flexibility, customer collaboration, and adaptability to changes [31]. Unlike traditional methodologies such as waterfall approach, which requires a linear and sequential process, while Agile allows for a more fluid and dynamic development cycle. According to the studies of the Standish Group's SHAOS Report (2015) [32], Agile methodology has a higher success rates compared to waterfall approach. In the waterfall approach, each phase of the project must be completed before moving on to the next, making it difficult to adapt to changes or new requirements once development has begun [33]. This can create challenges in addressing unforeseen issues that arise during the development process. In contrast, Agile breaks the project into smaller, manageable iterations [31]. Each iteration involves planning, designing, development, testing, and deployment, allowing us to deliver functional software more frequently and gather feedback earlier in the process. This methodology is the most suitable for projects that require adaptability and user-centric design, making it an excellent fit for the proposed system.

4.2 System Requirements

4.2.1 Hardware Requirements

Table 4.1 Laptop Specifications

Laptop Specifications	
Operating System	Windows 11 Home 64-bit
Central Processing Unit (CPU)	AMD Ryzen 7 PRO 5850U with Radeon Graphics 1.90 GHz
Memory (RAM)	16GB
Manufacturer	Lenovo

Table 4.2 Smartphone Specifications

Smartphone Specifications	
Operating System	Android 14
Processor	Exynos 2200
Display Resolution	1080 x 2340 pixels
Memory (RAM)	8GB
Manufacturer	Samsung Electronics

4.2.2 Software Requirements

Table 4.3 Software Specifications

Development Tools	• **Visual Studio Code**: The main Integrated Development Environment (IDE) for writing, debugging, and managing the Flutter and Dart codebase • **Android Studio**: Used primarily for Android SDK integration and emulator testing • **Google Colab**: Cloud-based environment for deep learning model training and experimentation
Programming Languages and Frameworks	• **Flutter**: The main framework for developing the system • **Dart**: Programming language for application logic and UI development • **Python**: Used to build and train the deep learning model before deployment to the system.
SDKs and Libraries	• **Android SDK**: Provides necessary libraries and build tools for Android app development • **TensorFlow Lite**: Lightweight machine learning framework for deploying trained deep learning models on mobile devices
Backend Services	• **Firebase Authentication**: Provides secure user login and registration • **Firebase Firestore Database**: A NoSQL cloud database for storing system data
External APIs	• **Gemini API**: Provide AI-driven natural language processing to power the chatbot feature in the system

4.3 Timeline

4.3.1 Timeline of the FYP1

Table 4.4 Timeline of the FYP1

Weeks Progress	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Complete user interface design														
Dataset collection														
System front end design														
Set up Google Firebase														
Complete system functions in VSCode														
Deep learning model training														
Integrating model into the system														
System testing														
Report Writing														
Presentation														

The timeline for the FYP1 has been carefully planned and divided into weekly tasks over a 14-week period, as illustrated in Table 4.4. In the initial, the focus is on completing the user interface design using Figma, ensuring a clear and intuitive layout for the application. Dataset collection is planned to begin in Week 3, followed by system front-end design in Week 4 and 5, meanwhile setting up the Google Firebase. While the system flow will be developed in Week 6 to 8 since it needed much effort on coding multiple pages with Google Firebase backend implementation. Deep learning model training is initiated in Week 9 and continues through Week 10; this model is developed specifically to classify user's Body Mass Index

(BMI) category based on user input data. During this period, model optimization and evaluation are carried out. Integration of the trained model into the system is slated for Week 11, allowing ample time for system testing and report writing in Week 11 and 12. The final stage is the preparation for the project presentation in Week 13 and 14. This timeline ensures a structured and progressive development approach, allowing for continuous iteration and refinement throughout the semester.

4.3.2 Timeline of the FYP2

Table 4.5 Timeline of the FYP2

Weeks / **Progress**	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Dataset collection														
Completing Workout Module														
Completing Dietary Module														
Completing Profile Setting														
Integrate AI chatbot feature														
Integrate community feature														
System refinement														
System testing and deployment														
Report Writing														
Presentation														

The main tasks to be completed in FYP2 is as shown in Table 4.5, begins with dataset collection in Week 1, and the workout module will be started after the dataset collection in Week 3. Then, I dietary module is scheduled to be completed during Week 5 and 6, focusing on developing a model that provides calorie intake suggestion based on user input data. Integration of the trained model will take place in Week 5 and 6, followed by the implementation of the AI chatbot feature in Week 6. The community feature will be developed between Week 7 and 8. System refinement is prioritized in Week 9 and 10 to ensure optimal performance and functionality before proceeding to testing and deployment in Week 10 and 11. The final phases include report writing in Week 11 and 12, and presentation preparations in Week 13 and 14.

Chapter 5

System Implementation

5.1 Integrating Google Firebase

Google Firebase is a cloud-based platform developed by Google to provide backend services for mobile and web applications. It offers powerful tools such as user authentication, real-time databases, and cloud storage. In this project, Firebase is used to handle core backend functionalities including Firebase Authentication and Firestore Databases.

Before starting with Firebase Authentication and Firestore Databases, ensure the Firebase Command Line Interface (CLI) is installed to allow signing in to Firebase account, to create and configure a project.

```
static const FirebaseOptions android = FirebaseOptions(
  apiKey: 'AIzaSyCpnO7lge2n8TXFjIlGAG6KkWfIKwoG1QU',
  appId: '1:583729978530:android:08116ea08800e65c858bfb',
  messagingSenderId: '583729978530',
  projectId: 'fitfuelf0edc',
  storageBucket: 'fitfuelf0edc.firebasestorage.app',
);
```

Figure 5.1.1 Firebase options in Flutter

5.1.1 Firebase Authentication

To allow users to register and log in securely, Firebase Authentication was integrated into the system. Firebase Authentication provides a simple and secure method to manage user sign-up, sign-in, and authentication processes using email and password credentials.

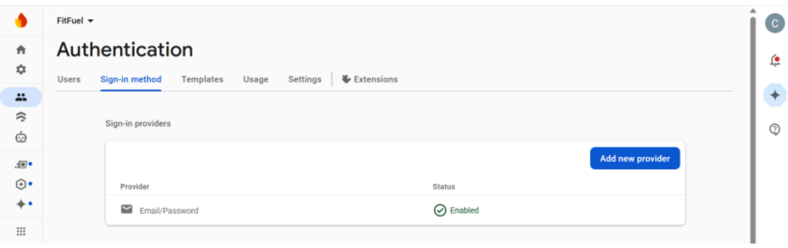


Figure 5.1.2 Firebase Authentication setup

During the setup, the authentication method was enabled in the Firebase Console, as shown in Figure 5.1.2. Users can create a new account by providing their email and password, which are then securely verified and stored within Firebase Authentication. To integrate the Firebase

Authentication in the system, a plugin called 'firebase_auth' is required to use the Firebase Authentication API. Detailed integrations are shown in Figure 5.1.3 and Figure 5.1.4. The integration ensures that users' credentials are securely managed without having to manually handle sensitive data, enhancing the overall security and user experience of the application.

```
lib > view > SignIn&SignUp > firebase_auth_implementation > firebase_auth_servi
import 'package:firebase_auth/firebase_auth.dart';

class FirebaseAuthServices {
  FirebaseAuth _auth = FirebaseAuth.instance;

  Future<User?> signUpWithEmailAndPassword(
    String email,
    String password,
  ) async {
    try {
      UserCredential credential = await _auth.createUserWithEmailAndPassword(
        email: email,
        password: password,
      );
      return credential.user;
    } catch (e) {
      print('Some error occured');
    }

    return null;
  }

  Future<User?> signInWithEmailAndPassword(
    String email,
    String password,
  ) async {
    try {
      UserCredential credential = await _auth.signInWithEmailAndPassword(
        email: email,
        password: password,
      );
      return credential.user;
    } catch (e) {
      print('Some error occured');
    }
```

```
void _signUp() async {
  String firstName = _firstNameController.text;
  String lastName = _lastNameController.text;
  String email = _emailController.text;
  String password = _passwordController.text;

  if (!isCheck) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text("Please accept the Privacy Policy and Terms of Use"),
      ), // SnackBar
    );
    return;
  }

  try {
    User? user = await _auth.signUpWithEmailAndPassword(email, password);

    if (user != null) {
      // Store additional user data in Firestore
      await FirebaseFirestore.instance.collection('users').doc(user.uid).set({
        'firstName': firstName,
        'lastName': lastName,
        'email': email,
        'createdAt': FieldValue.serverTimestamp(),
        'uid': user.uid,
      });

      Navigator.push(
        // ignore: use_build_context_synchronously
        context,
        MaterialPageRoute(builder: (context) => CompleteProfileView()),
      );
    }
  } catch (e) {
    // ignore: use_build_context_synchronously
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Error during sign up: ${e.toString()}')),
    );
  }
}
}
```

Figure 5.1.3 Code snippets of signing up using Firebase Authentication

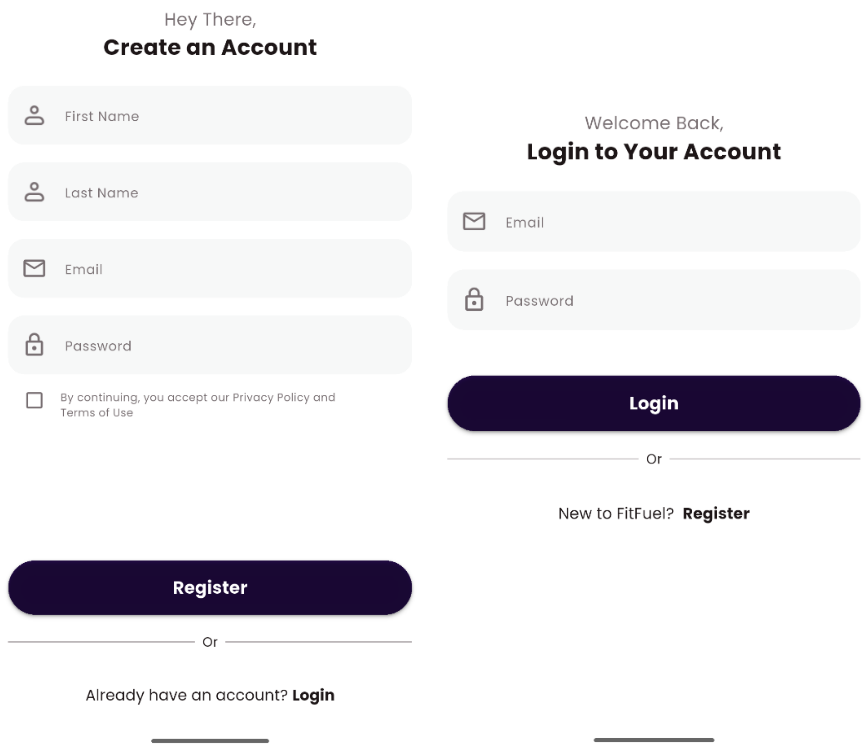


Figure 5.1.4 Sign Up and Sign In page of the application

5.1.2 Firestore Database

To store user data such as personal information, Firestore Database was integrated into the system. Firestore is a scalable and flexible NoSQL cloud database. In this project, once a user registered their profile, details such as first name, last name, weight, are stored under their unique User ID (UID) in the Firestore Database. To implement Firestore integration, the 'cloud_firestore' plugin was added to the Flutter project. The configuration and examples of how user data is stored are illustrated in Figure 5.1.5 and Figure 5.1.6. Firestore's real-time updating capability ensures that any changes made by the user are instantly reflected in the application, enhancing data consistency and system responsiveness.

```
await FirebaseFirestore.instance
    .collection("users")
    .doc(
      user.uid,
    ) //referrence from previous signup uid
    .update({
      'gender': _genderController.text,
      'age': _ageController.text,
      'weight': _weightController.text,
      'height': _heightController.text,
    });
```

Figure 5.1.5 Example of code snippets of updating data to Firestore Database

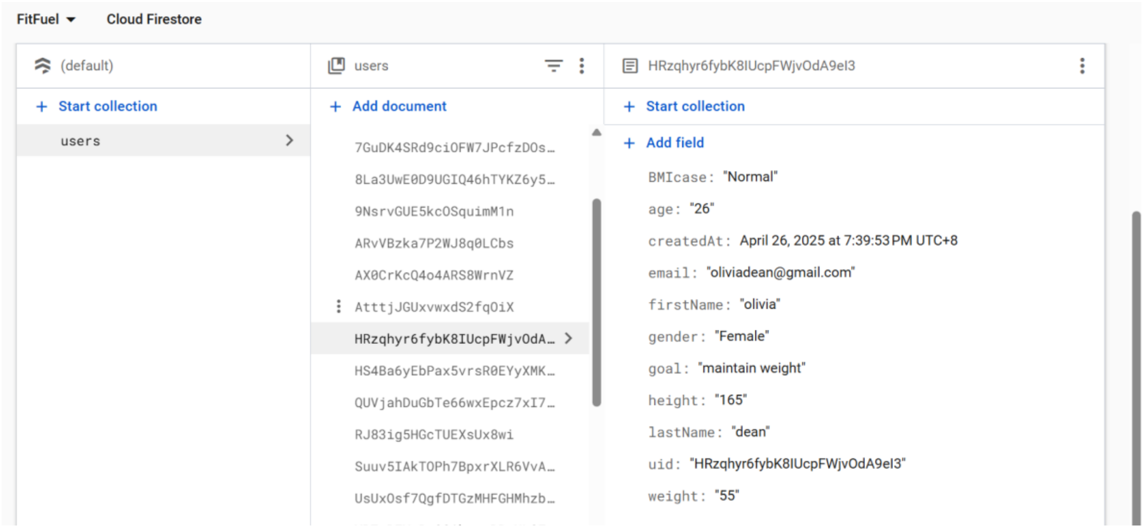


Figure 5.1.6 Document stored under User collection after user completed user profile

5.2 Google Gemini API Implementation

The Google Gemini Application Programming Interface (API) was integrated into the system to provide artificial intelligence (AI) support for the system which deliver natural and context-aware responses to users and enhancing interactivity.

To begin the integration, an API key was generated from the Google AI Studio platform as shown in Figure 5.2.1, and it is securely stored in the project environment file to prevent unauthorised access. Hardcoding the API key directly into the source code raise security risks, therefore, developers are encouraged to store API keys in secure locations to ensure safe and reliable integration.

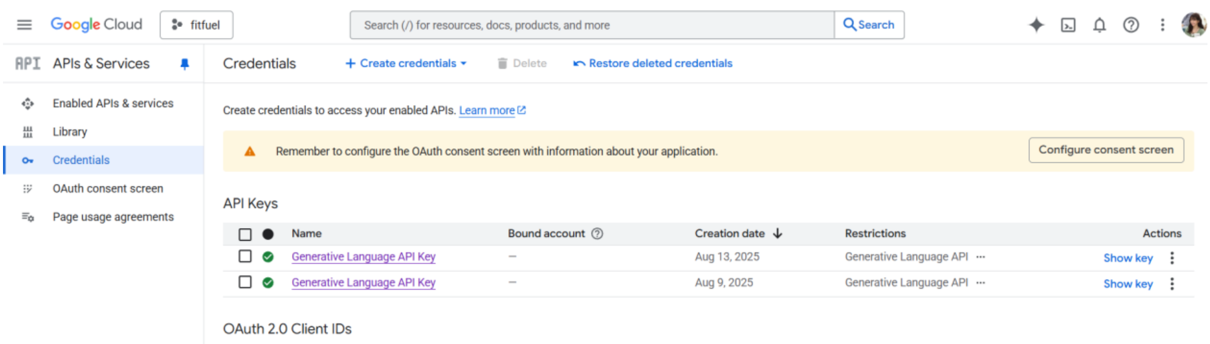


Figure 5.2.1 Setup of the Gemini API key within the project

In the Flutter project, the HTTP package was installed to facilitate the API communication between the system and the Gemini API endpoint. With this setup, the system is able to send user input as requests and receive responses from Gemini for further processing.

5.3 Creating a BMI Category Prediction Model

A Body Mass Index (BMI) category prediction model was developed to predict user’s BMI category within the application. The model predicts whether a user falls into categories such as severe thinness, mild thinness, moderate thinness, normal, overweight, obese, or severely obese, based on input features including weight, height, age, and gender.

To build the prediction model, a publicly available dataset [11] was pre-processed using **Pandas** for data manipulation and **scikit-learn** for data normalisation, label encoding, and handling class imbalance with the **SMOTE** technique. This ensured a balanced dataset for training, validation, and testing. The features were scaled using **MinMaxScaler**, while categorical features were one-hot encoded to prepare them for classification tasks.

The model itself was implemented in **TensorFlow** and **Keras**, where a U-Net based Convolutional Neural Network (CNN) was constructed to classify BMI cases. The training was carried out in Google Colab, which provided a cloud-based GPU environment for faster computation. Training, validation, and test datasets were created using an 80:10:10 split, and the model was trained with early stopping to prevent overfitting. The details of the model hyperparameters will be further discussed in **Chapter 6.1.2**.

Finally, the trained model was converted into **TensorFlow Lite (TFLite)** format, enabling deployment on mobile devices. This ensures that the prediction tool runs efficiently within the application, allowing users to receive real time BMI category predictions directly from the application.

5.4 User Registration

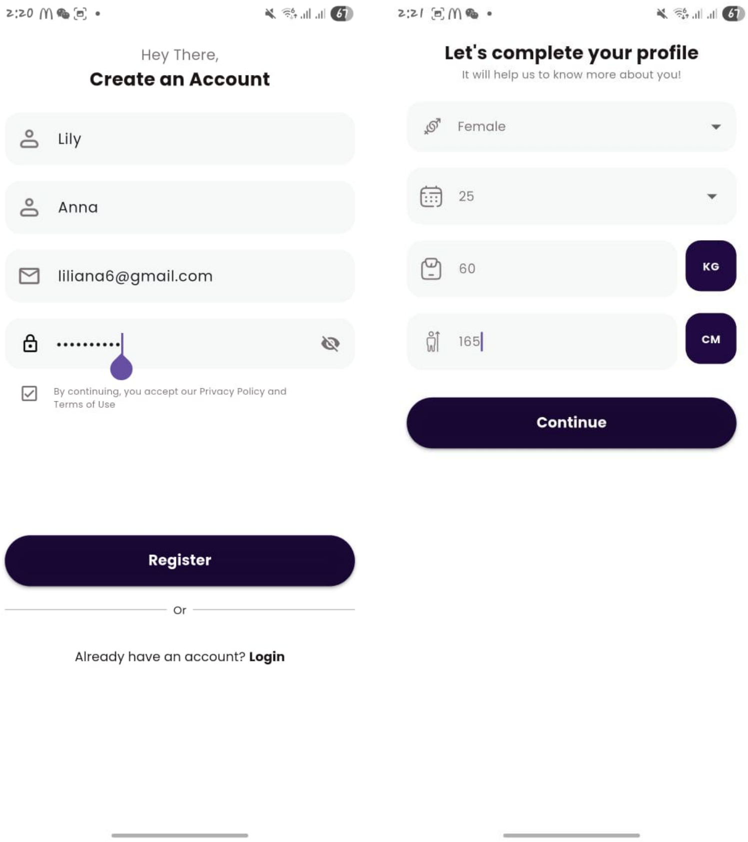


Figure 5.4.1 User Registration steps of the application

For each new user registration, users are required to create a new account using unregistered email address and complete their profile by inputting basic information including gender, age, weight, and height. Each time user registers a new account, the system will create a new user document with unique user identifier (uid) in the Firebase Authentication and stores user data in the Firestore database. Figure 5.4.2 and Figure 5.4.3 below shows the implementation steps and results in Firebase Authentication and Firestore Database.

```dart
try {
  User? user = await _auth.signUpWithEmailAndPassword(email, password);

  if (user != null) {
    // Store additional user data in Firestore
    await FirebaseFirestore.instance.collection('users').doc(user.uid).set({
      'firstName': firstName,
      'lastName': lastName,
      'email': email,
      'createdAt': FieldValue.serverTimestamp(),
      'uid': user.uid,
    });
```

```dart
Future<User?> signUpWithEmailAndPassword(
  String email,
  String password,
) async {
  try {
    UserCredential credential = await _auth.createUserWithEmailAndPassword(
      email: email,
      password: password,
    );
    return credential.user;
  } catch (e) {
    print('Some error occured');
  }

  return null;
}
```

Figure 5.4.2 Code snippets of handling new user registration

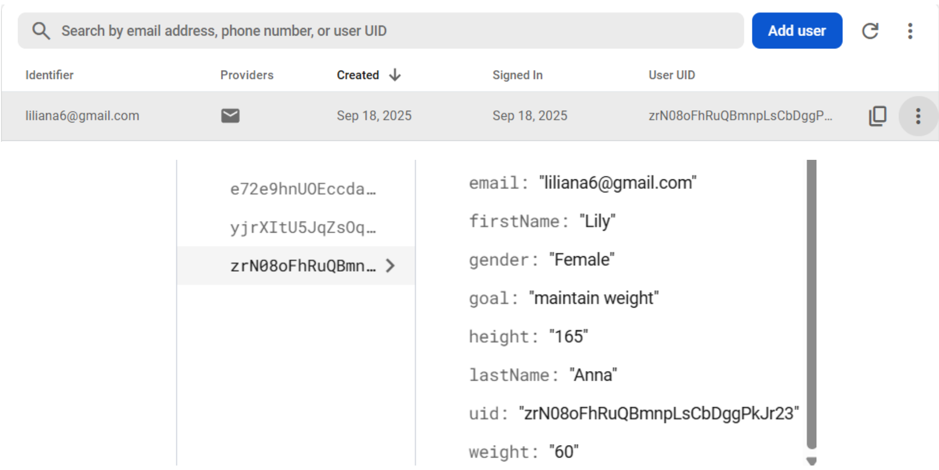


Figure 5.4.3 New user document creation in Firebase Authentication and Firestore Database

5.4.1 Model Prediction

Upon successful registration, after users complete their profile by providing their height, weight, age, and gender, the system immediately invokes the integrated deep learning model to predict the user's initial Body Mass Index (BMI) case. This prediction help classify the user's body condition into specific BMI categories, such as Severe Thinness, Moderate Thinness, Mild Thinness, Normal, Overweight, Obese, Severe Obese.

As shown in Figure 5.4.4, the system dynamically outputs a corresponding goal for the user:

- If the user's BMI case is classified as **Normal**, the system allows the user to freely select their desired goal, choosing between **maintaining weight, gaining weight, or losing weight** based on their personal preference.
- However, if the user's BMI case falls into **underweight** categories, the system automatically sets the goal to **gain weight**.
- Similarly, if the BMI case is categorized as **overweight or obese**, the system automatically sets to goal to **lose weight** to promote healthier outcomes.

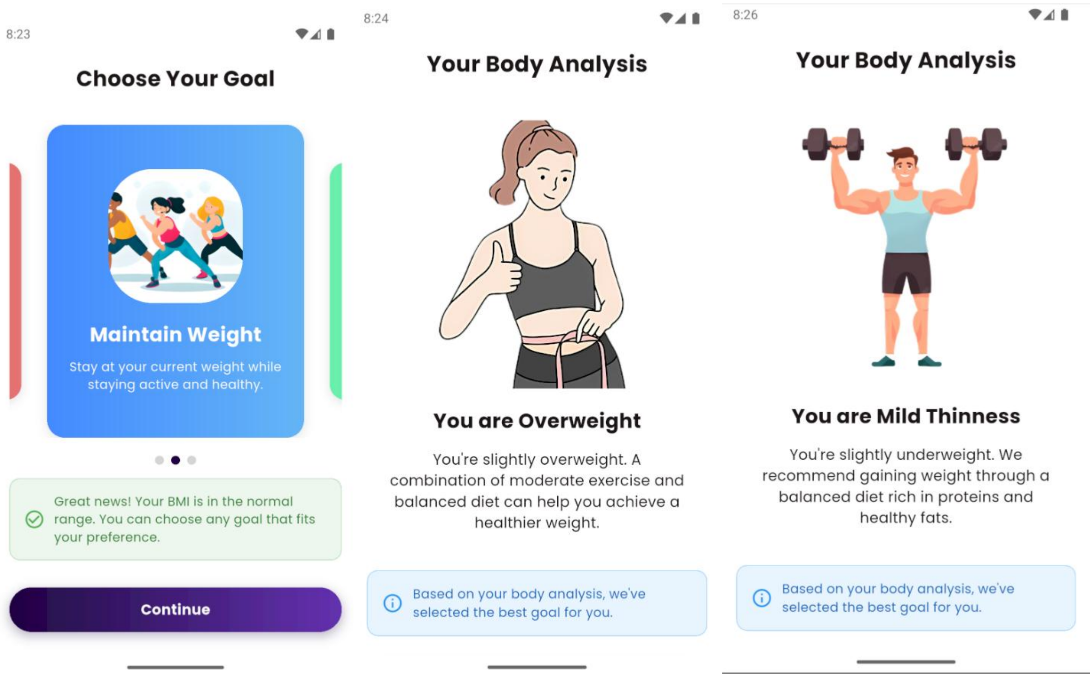


Figure 5.4.4 Different outcome for different BMI categories in initial goal setting page.

To implement the BMI prediction in the system, a prediction model in discussed previously in Chapter 5.3, that has been converted into TensorFlow Lite (.tflite) format has to be ready in the project's assets folder. Figure 5.4.5 below shows the steps to invoke a prediction model to predict user's BMI category. The implementation works by using the interpreter to interpret BMI category using user's height, weight, age, and gender inserted previously in the user profile completion page. The system will then predict and map user's predicted BMI category

```
try {
  final modelFile = await _getModelFile('assets/model/BMI_model.tflite');
  final interpreterOptions = InterpreterOptions();
  _interpreter = await Interpreter.fromFile(
    modelFile,
    options: interpreterOptions,
  );
  _modelLoaded = true;
} catch (e) {
```

```
// BMI case mapping
final Map<int, String> _bmiCaseMapping = {
  0: "Overweight",
  1: "Normal",
  2: "Obese",
  3: "Severe Obese",
  4: "Severe Thinness",
  5: "Mild Thinness",
  6: "Moderate Thinness",
};
```

```
// Get the prediction result
final predictionList = outputs[0];
final maxIndex = predictionList.indexOf(
  predictionList.reduce((curr, next) => curr > next ? curr : next),
);
final predictionResult = _bmiCaseMapping[maxIndex] ?? "Unknown BMI Case";

return {
  'success': true,
  'bmiCase': predictionResult,
```

Figure 5.4.5 Code snippets of integrating model prediction into the system

A comprehensive explanations and discussions regarding the prediction mechanism, architecture, and performance of the deep learning model used in this application are provided in Chapter 6 to provide a clearer understanding of how user data is utilised to generate accurate Body Mass Index (BMI) category outcomes.

5.5 User Profile

The system features a user profile functionality, which allows users to view and manage their personal information within the application. The user data will first show in the profile page by retrieving user data from Firestore Database based on their UID. Users may edit their profile and save their new profile. Once new profile is saved, the new user data will be updated to the Firestore Database. Details of implementation are shown in Figure 5.5.1 and Figure 5.5.2.

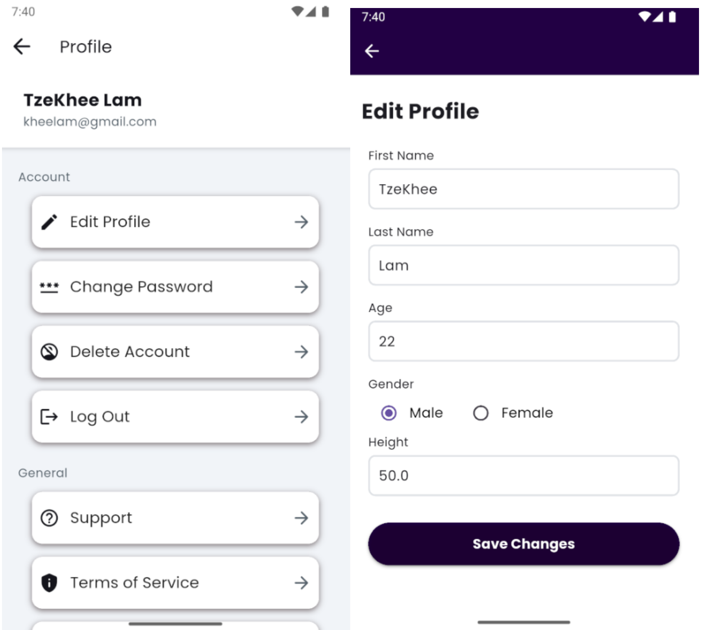


Figure 5.5.1 User profile of the application

```
Future<void> _saveProfile() async {
  if (userId == null) return;

  setState(() {
    _isLoading = true;
  });

  try {
    await FirebaseFirestore.instance.collection('users').doc(userId).update({
      'firstName': _firstNameController.text.trim(),
      'lastName': _lastNameController.text.trim(),
      'age': int.tryParse(_ageController.text.trim()) ?? 0,
      'height': double.tryParse(_heightController.text.trim()) ?? 0.0,
      'gender': groupValue,
    });

    setState(() {
      _isLoading = false;
    });

    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text("Your profile has been updated"),
        duration: Duration(seconds: 2),
      ), // SnackBar
    );
    Navigator.pop(context);
```

Figure 5.5.2 Code snippets of updating new user profile

5.6 Log Weight

The system provides a log weight feature that enables users to conveniently track their weight progress over time. On the Home Page, a dedicated 'Weight' section is available where the user's most recently logged weight is displayed. This allows users to monitor their latest weight status at a glance immediately upon launching the application.

Users can access the weight dashboard by clicking on the weight section. Inside the dashboard, users are able to log a new weight entry. As shown in Figure 5.6.1, a simple and user-friendly page is provided for users to input their new weight value. After entering and confirming new weight, the system updates the user's weight data in the Firestore Database under their unique user ID (UID). Each time the user updated their new weight, the deep learning model will predict again user's new BMI case and update to the Firestore Database. Once the new weight is successfully saved, the updated weight is immediately reflected on the Home Page in the weight section.

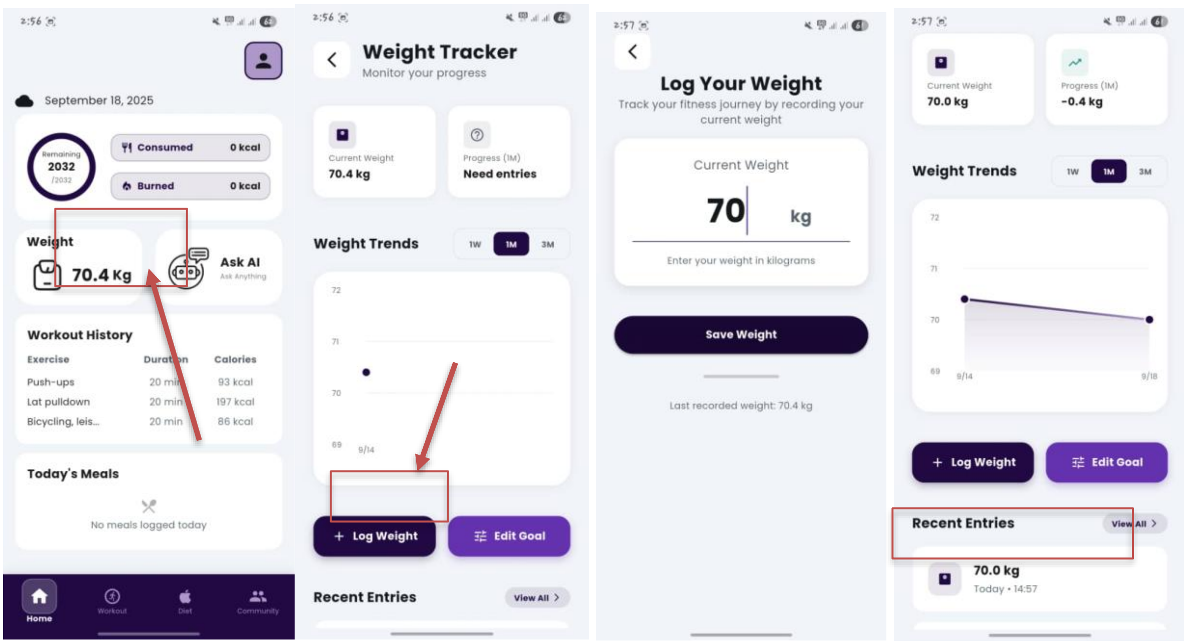


Figure 5.6.1 Weight dashboard and log weight screen of the application

5.7 Edit Goal

The system allows users to edit their goal from the weight dashboard. As shown in Figure 5.7.1, the user is provided with a page which shows their current goal, and goal editing section. However, users who currently BMI category prediction falls outside the "normal" range are not allowed to edit their goal.

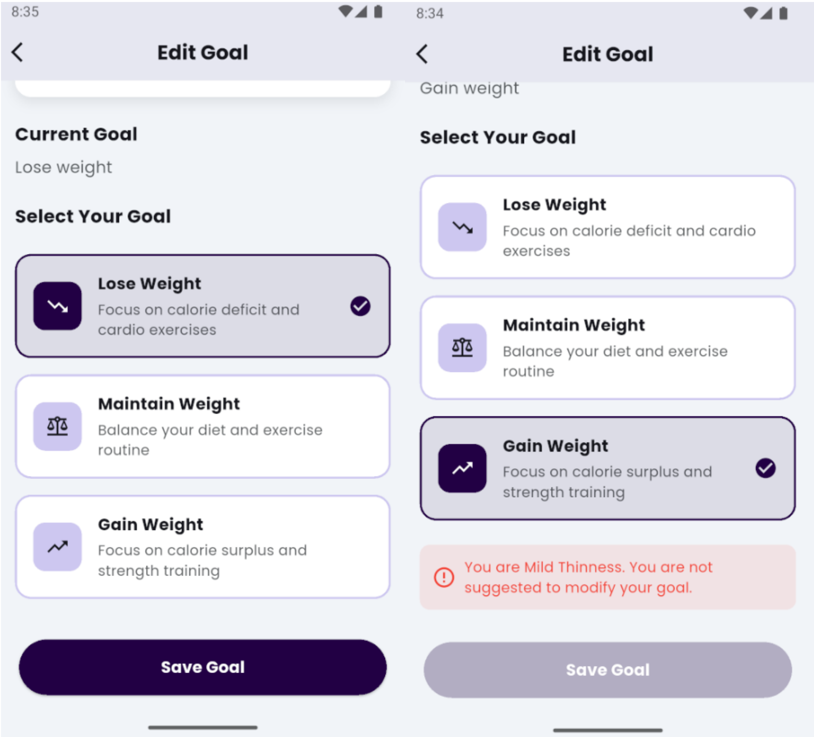


Figure 5.7.1 Editing goal page of the application

5.8 AI Chatbot

The AI chatbot provides interactive assistance, acting as a health companion for user throughout the application. The chatbot can be accessed via the homepage, where users are presented with a conversational interface.

As shown in Figure 5.8.1, users can input free-text questions such as "How to lose weight?" or "where do you live?". The system sends the query to the Gemini API, which processes the request and returns a context-aware response. However, if user's text questions are not related to fitness, dietary, or general wellness, the system will response with error message as shown in Figure 5.8.1. For clearer explanation, Figure 5.8.2 below shows the system instruction that is given to the Gemini API. The Chatbot should only respond to related topics.

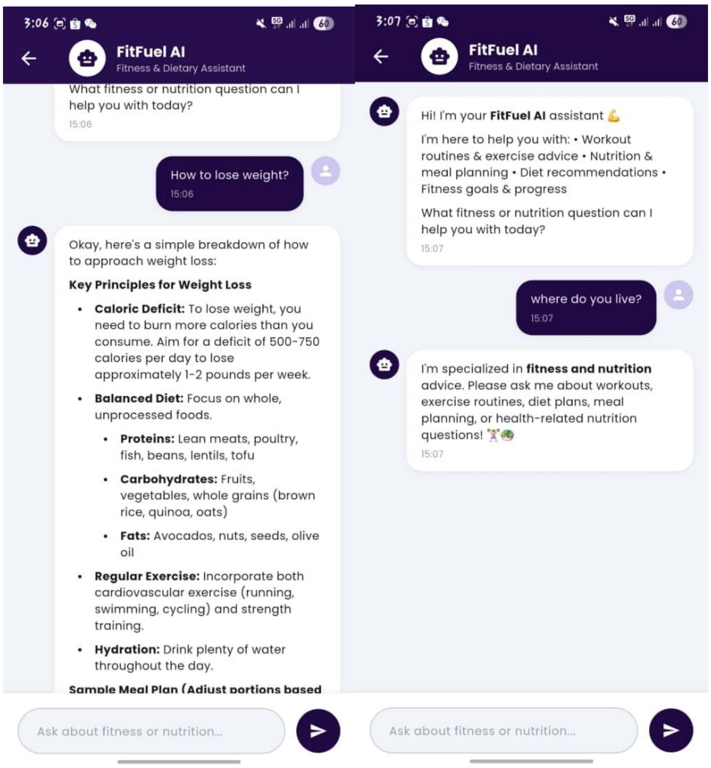


Figure 5.8.1 AI Chatbot screen of the application

```
apiKey: _apiKey,
systemInstruction: Content.system('''
  You are FitFuel AI, a specialized fitness and nutrition assistant. You ONLY provide advice and answers related t

  1. FITNESS: Exercise routines, workout plans, strength training, cardio, flexibility, sports, physical activitie
  2. NUTRITION: Diet plans, meal planning, calorie counting, macronutrients, micronutrients, healthy recipes, weig
  3. WELLNESS: Sleep, hydration, stress management as they relate to fitness and nutrition

  IMPORTANT RULES:
  - Only respond to fitness and nutrition-related questions
  - If a question is not related to fitness/nutrition, politely redirect the user to ask about fitness or dietary
  - Provide practical, evidence-based advice
  - Always remind users to consult healthcare professionals for medical concerns
  - Keep responses concise but helpful
  - Be encouraging and motivational
  - Use markdown formatting for better readability (use **bold** for emphasis, bullet points, numbered lists, etc.
  - Structure your responses with clear headings and formatting when appropriate

  If someone asks about topics outside fitness/nutrition, respond with: "I'm specialized in fitness and nutrition
'''),
```

Figure 5.8.2 System instruction to the Gemini API

5.9 Log Exercise

The log exercise feature enables user to document workouts for calorie burn tracking. On the workout page, a “Log Exercise” button is provided, redirecting users to a log exercise page where they can select an exercise and input exercise details including start time and duration. As shown in Figure 5.9.1 below, users will select an exercise from the exercise list listed on the log exercise page. The list of exercises is categorised into different exercise categories including Cardiovascular, Strength Training, HIIT, and Low-Impact Exercise. After choosing desired exercise and input exercise duration. The system will calculate the estimated calories burned from the exercise using e.q (2.3): *MET x 3.5 x weight x duration / 200*.

To show the list of exercises on the screen, an exercise list data in JSON file format is ready in the system. As shown in Figure 5.9.2, exercises with MET values are listed under its corresponding exercise category.

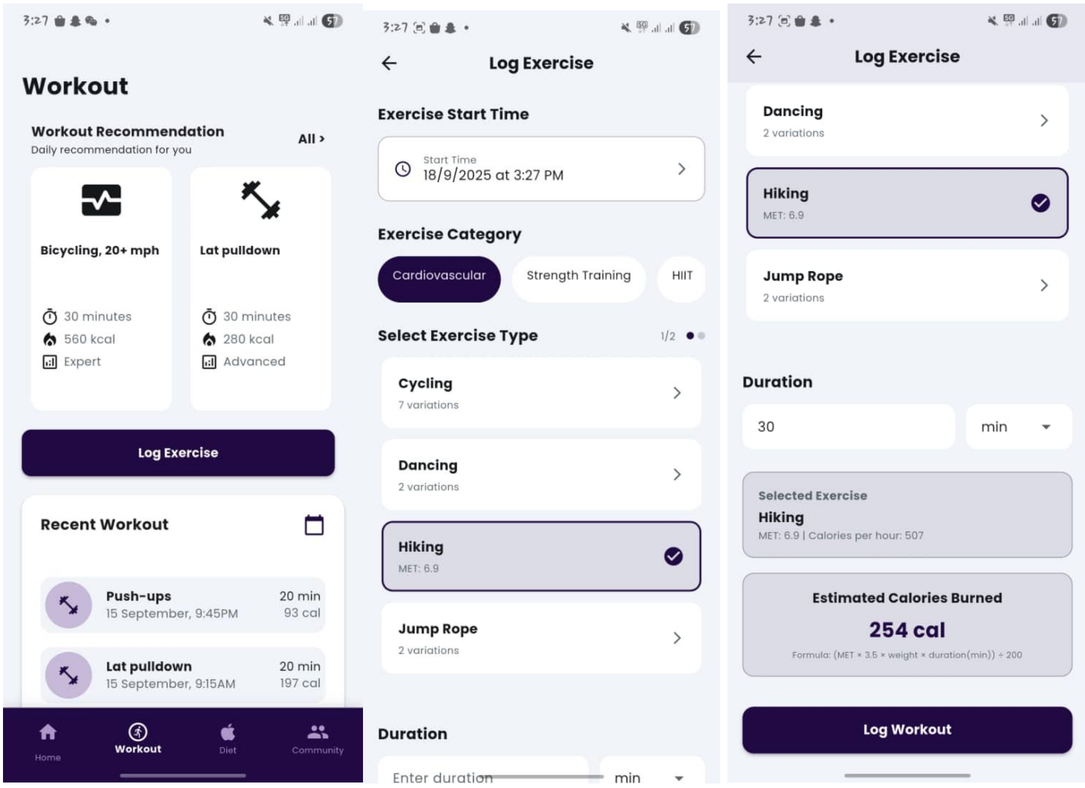


Figure 5.9.1 Log exercise screen of the application

```
    "Strength Training": [
        {
            "name": "Push-ups",
            "met": 3.8
        },
        {
            "name": "Pull-ups",
            "met": 3.8
        },
        {
            "name": "Squats",
            "met": 5.0
        },
        {
            "name": "Lunges",
            "met": 3.8
        },
        {
```

Figure 5.9.2 Design of exercise list data in JSON file format

Once user logs the workout, the Firestore stores data in the workout collection with fields of category, duration, MET, calories burned, and timestamp. The exercise is further used for exercise history viewing.

5.10 View Workout History

The system also provides a comprehensive workout history page with statistical data, where users can track all past exercises in chronological order. As shown in Figure 5.10.1, each workout entry displays details such as exercise name, duration, calories burned, and timestamp.

The history page retrieves records dynamically from Firestore. The interface is designed with clear filtering options, including filtering by workout categories, or time period. This allows users to review their progress over specific time periods or view specific data on selected exercise category.

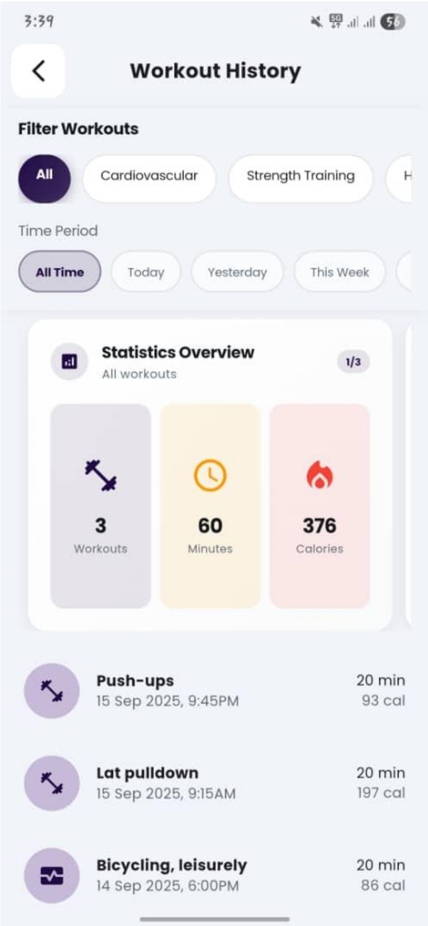


Figure 5.10.1 View workout history of the application

5.11 Workout Recommendation

The system delivers workout recommendations based on the user's BMI case. As shown in Figure 5.11.1, when the user navigates to the workout recommendation section, the system will display a list of exercises that is suitable for their predicted BMI category and goal.

The recommendation works by obtaining the user's BMI category and goal from the Firestore, the system will then map user's BMI category and goal with exercise recommendation. The details of the exercise recommendation JSON file format design are shown in Figure 5. 11.2. Once user is mapped with suitable exercise category, the system will obtain list of suitable exercises from the exercise list shown previously in Figure 5.7.2 and display all recommended exercises as shown in Figure 5. 11.1. For example, a user with BMI category of "normal" who wants to "gain weight" will be suggested to do strength training or low-impact exercise. The system will then display all suitable exercises from strength training and low-impact exercise category on the screen.

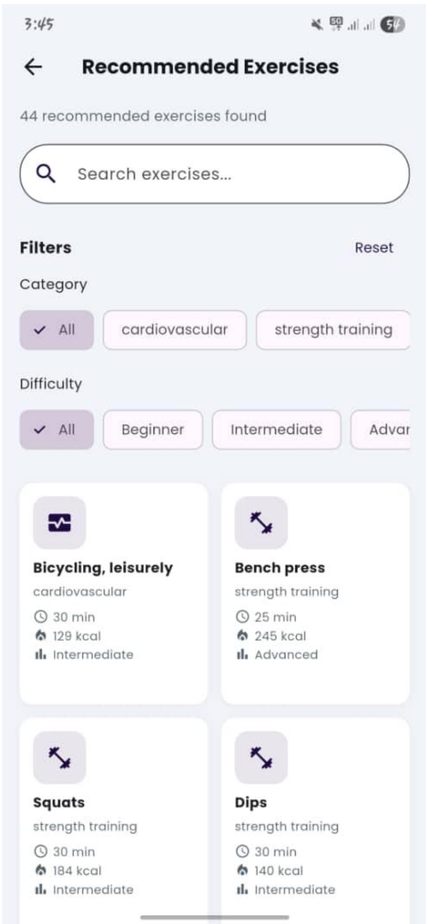


Figure 5.11.1 Workout Recommendation of the application

```
    ],
    "mild thinness": [
        {
            "goal": "gain weight",
            "exercise_category": "strength training, low-impact exercise"
        }
    ],
    "normal": [
        {
            "goal": "gain weight",
            "exercise_category": "strength training, low-impact exercise"
        },
        {
            "goal": "maintain weight",
            "exercise_category": "cardiovascular, strength training"
        },
        {
            "goal": "lose weight",
            "exercise_category": "cardiovascular, HIIT"
        }
    ],
    "over weight": [
        {
            "goal": "lose weight",
            "exercise_category": "HIIT, strength training"
        }
```

Figure 5.11.2 Code snippets on the design of workout recommendation in JSON file format

5.12 Add Meal

The add meal feature allows users to record their daily food intake for nutritional monitoring. On the dietary homepage, users can access to the add meal page. Where the add meal page allows user to either add meal from the recipe library or customise their meal, as shown in Figure 5.12.1.

For the recipe library, it works by obtaining recipes that is predefined and stored as JSON file in the system. As shown in Figure 5.12.2, each recipe comes with details including recipe name, preparation time, cooking time, total time, servings, ingredients, directions, calories, fats, protein, and carbs.

For the meal customisation, it works by allowing user to input the meal name and even the nutritional values manually. Users are also allowed to use the ‘AI Analyse’ function which will parse the meal name to the Gemini API. As shown in Figure 5.12.3, the Gemini API will response with JSON object of calories, protein, fats, and carbs, which is used to reflect on the custom meal page. The nutritional analysis result is shown in Figure 5.12.4.

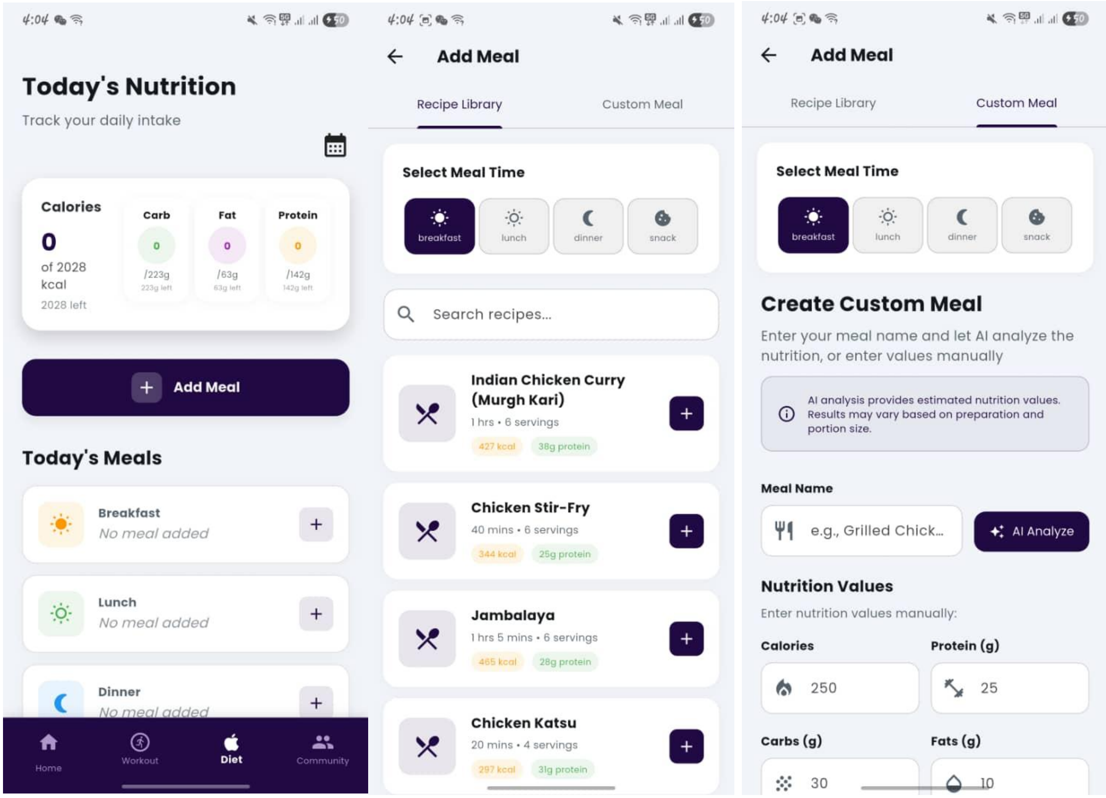


Figure 5.12.1 Add meal function of the application

```json
[
  {
    "name": "Indian Chicken Curry (Murgh Kari)",
    "prep_time": "20 mins",
    "cook_time": "40 mins",
    "total_time": "1 hrs",
    "servings": 6,
    "ingredients": [
      {
        "quantity": "2",
        "unit": "pounds",
        "name": "skinless, boneless chicken breast halves"
      },
      {
        "quantity": "2",
        "unit": "teaspoons",
        "name": "salt"
      },
      {
        "quantity": "½",
        "unit": "cup",
        "name": "cooking oil"
      },
```

Figure 5.12.2 Predefined recipe list in JSON file format

```dart
final String prompt = '''
  Please analyze the nutritional content of this meal: "$mealName"

  Provide estimated nutritional values for a single serving size of this dish.
  Consider the main ingredients and typical preparation methods.

  Respond with ONLY a JSON object in this exact format (no other text):
  {
    "calories": [number],
    "protein": [number],
    "carbs": [number],
    "fats": [number]
  }
```

Figure 5.12.3 System instruction to the Gemini API

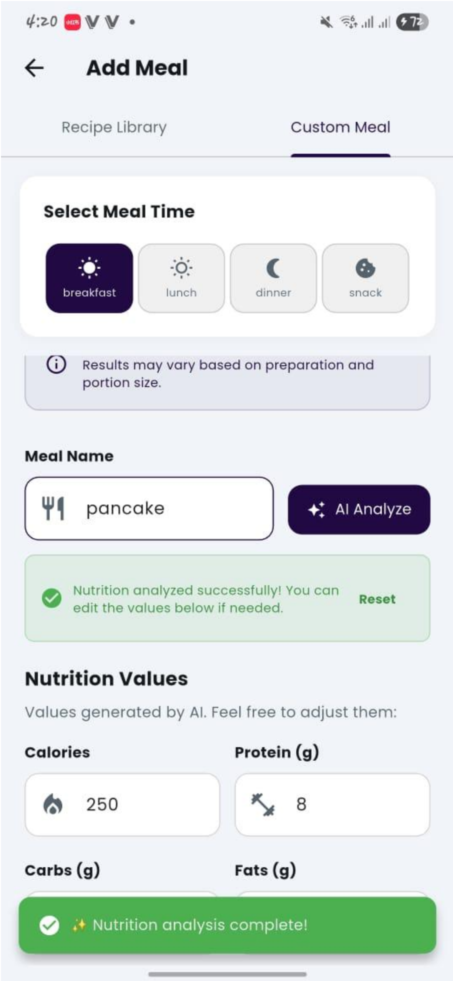


Figure 5.12.4 AI nutrition analysis result of the application

5.13 View Dietary History

The dietary history page provides an overview of all logged meals. User can view entries organised by date, with each entry displaying calories and macronutrient breakdown. As shown in Figure 5.13.1, user may choose to view dietary statistical data by selecting custom date from calendar, or to view weekly or monthly statistical data. The dietary data is dynamically retrieved from the Firestore and sorted by date and mealtime.

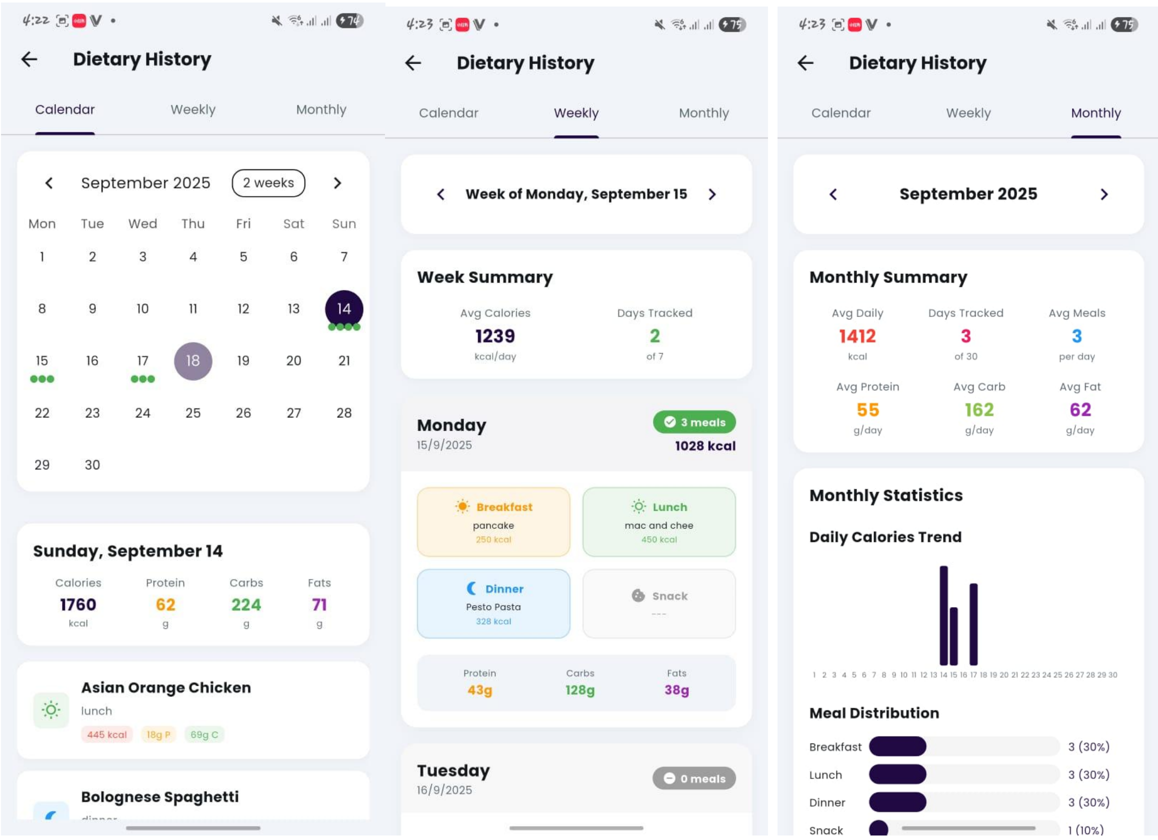


Figure 5.13.1 Dietary history of the application

5.14 Create Post in Community

The community module allows users to share updates, achievements, and questions with others. On the Community Page, users can tap on the "+" button on the bottom right corner to draft new content. As shown in Figure 5.14.1, users may compose text posts and select their desired channel (general community, fitness enthusiasts, or healthy nutrition) to be posted on. Once submitted, the post is stored in the community collection under either the generalPosts or clubPosts in the Firestore. The post is instantly published to the community feed. This feature enhances the social interaction and creates a supportive environment for users pursuing similar goals.

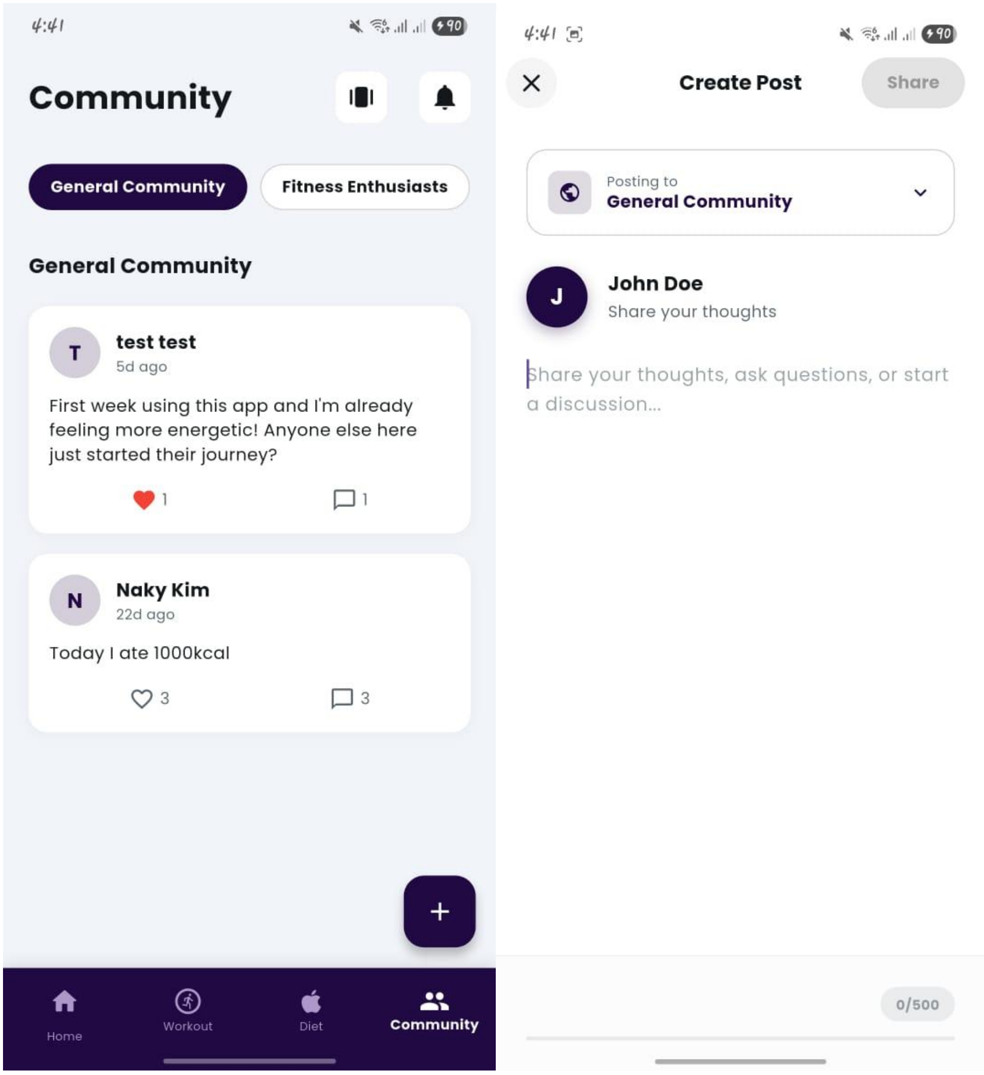


Figure 5.14.1 Creating post in the application

5.15 Interact with Others' Post in Community

The system allows users to engage with posts by liking or commenting. When a user selects a post, they are presented with interaction options as shown in Figure 5.15.1. Liking a post increments the like counter in Firestore (Figure 5.15.2), while commenting adds a new document under the comments collection of the selected post (Figure 5.15.3). All interactions are updated in real time, ensuring the post reflects the latest engagement activity. For every interaction, the system will create a notification document which is used to notify the post owner on the interaction (Figure 5.15.4).

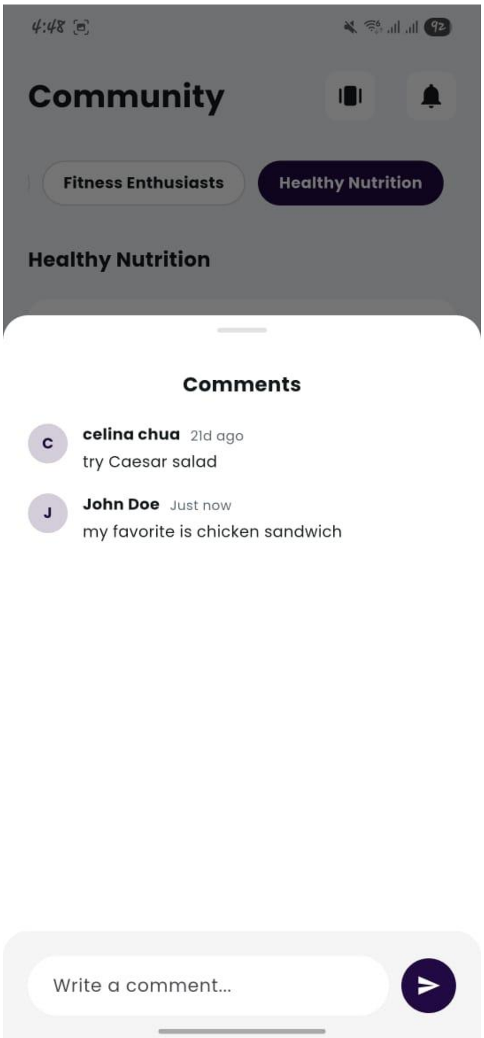


Figure 5.15.1 Commenting on post in the application

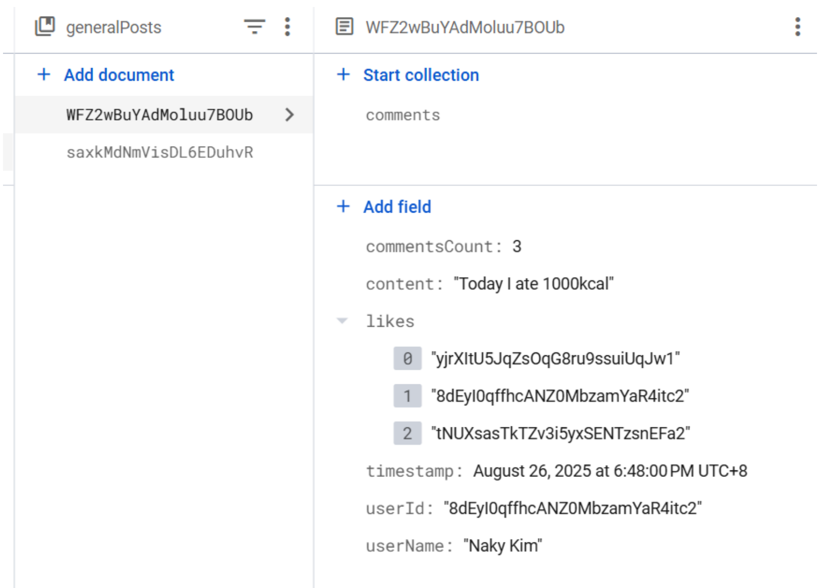


Figure 5.15.2 Storing Likes in Firestore

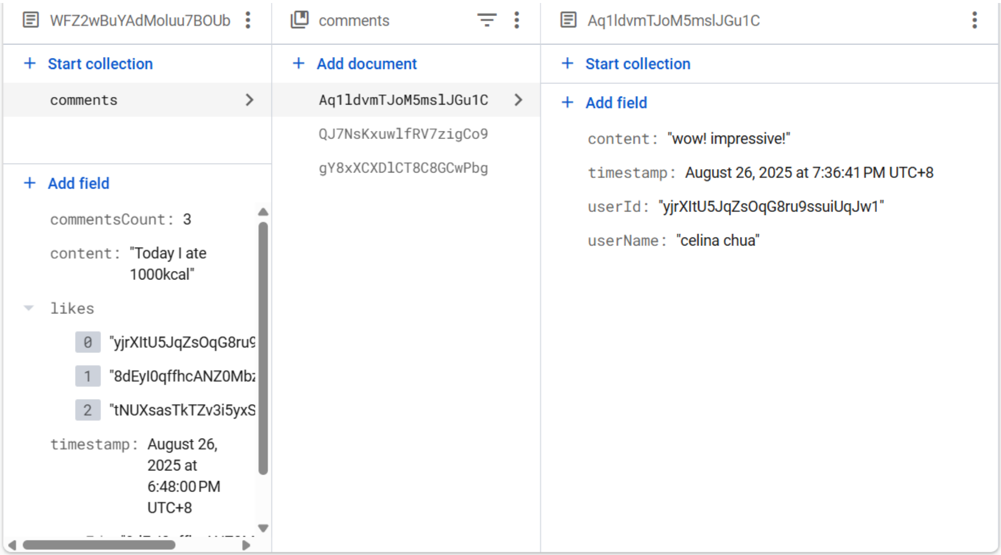


Figure 5.15.3 Comment document in Firestore

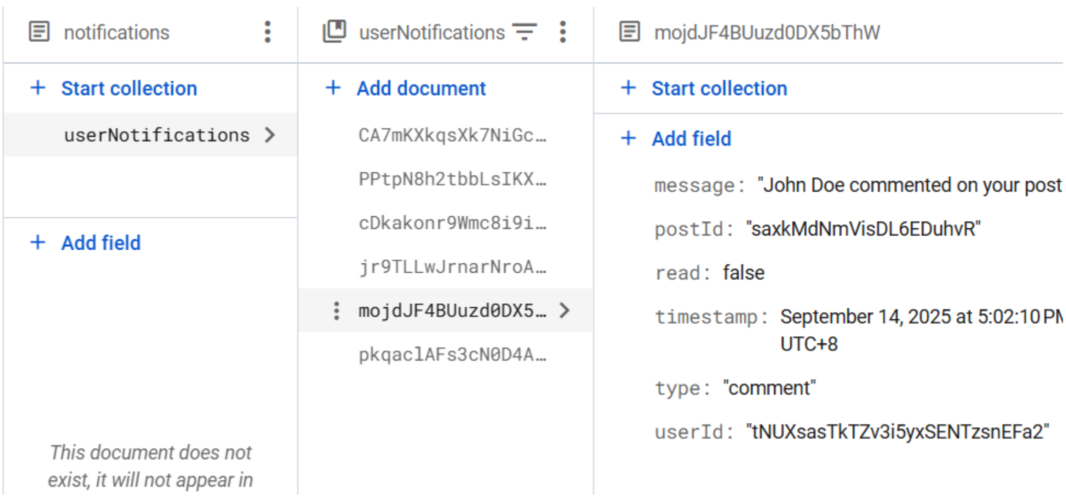


Figure 5.15.4 Notification document in Firestore

Chapter 6
System Evaluation and Discussion

6.1 Comparing Overall Performance Between Models

This section presents a comprehensive evaluation and comparison between three trained predictive models developed for BMI case classification: the Deep Neural Network (DNN) [20], the U-Net, a Convolutional Neural Network (CNN) architecture [25], and the Random Forest [14]. The objective was to determine the most effective model that not only achieve high prediction accuracy but also generalise well to the data, minimizing the risk of overfitting. The evaluation was conducted using a pre-processed Body Mass Index (BMI) dataset [11] in which only essential features including height, weight, age, and gender were retained for model input. This ensures that the models focus exclusively on core features to predict accurate BMI classifications.

6.1.1 Deep Neural Network (DNN)

The DNN model [20] was designed as the initial baseline for BMI case classification. The architecture was relatively deep, consisting of three hidden dense layers, progressively reducing in size to enable hierarchical feature learning. The architecture and training configuration are as follows:

Table 6.1 Training configuration of DNN

Input layer	4 neurons
Hidden layers	First Dense layer: 256 neurons with ReLU activation Second Dense layer:128 neurons with ReLU activation Third Dense layer: 64 neurons with ReLU activation
Output layer	7 neurons with Softmax activation for multi-class classification
Optimizer	Adam optimizer with a learning rate of 0.001
Loss function	Categorical crossentropy
Epochs	150
Batch size	64

6.1.2 Convolutional Neural Network (CNN)

The U-Net architecture [25] was adapted in this project to process tabular data by applying 1-Dimensional (1D) convolutions, allowing efficient feature extraction from the input attributes. The adapted U-Net CNN architecture consisted of three major components: an encoder, a bottleneck, and a decoder, followed by a fully connected classification head. The key components of the model architecture are as follows:

Tabel 6.2 Training configuration of U-Net based CNN

Input Layer	Input shape (4,1), treating each feature as a single channel 1D input.
Encoder Path	Three encoder blocks, each consisting of: • 1D convolutional layer (kernel size=2, ReLU activation) • Batch normalization • MaxPooling1D The number of filters increased progressively (8, 16, 32) across the encoder blocks to capture more complex features at deeper layers.
Bottleneck	A convolutional layer with 64 filters and batch normalization
Decoder Path	Three decoder blocks, each consisting of: • UpSamling1D • 1D convolutional layer • Batch normalization • Skip connections from corresponding encoder layers to enhance feature learning The number of filters decreased symmetrically (32, 16, 8) during decoding.
Classification head	• Flattened the output of the final decoder layer • Fully connected dense layer with 32 neurons • Dropout layer (rate = 0.5) • Output dense layer with 7 neurons using softmax activation
Training configuration	• Loss function: categorical crossentropy • Optimizer: Adam with learning rate of 0.001
Epochs	200
Batch size	64

6.1.3 Random Forest

The random forest algorithm [14] was implemented in this project as one of the benchmarks models for BMI category classification. Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the class selected by the majority of tress [14]. The model training configurations are shown in the table 6.3 below:

Table 6.3 Training configuration for Random Forest Model

Input features	Weight, Height, Gender (encoded), Age
Number of tress (n_estimators)	BMIcase (with 7 categories: mild thinness, moderate thinness, severe thinness, normal, overweight, obese, severe obese)
Train-Test split	80:20
Hyperparameter tuning	• RandomizedSearchCV with 50iterations • 5-fold cross-validation • scoring=accuracy
Best parameters found	• n_estimators=351 • max_depth=25 • max_features=log2 • min_samples_split=8 • min_samples_leaf=3 • bootstrap=True
optimizer	Randomized hyperparameter search across predefined ranges
Random state	42

6.1.4 Model Performance Evaluation

The performance of DNN, U-Net based CNN, and Random Forest are evaluated across various metrics, including accuracy, precision, recall, and F1-score [34], as summarised in Table 6.4. These metrics provide a comprehensive view of each model's performance and allow comparison between different models.

The accuracy measures the overall proportion of correctly classified samples against the total number of samples. It is calculated using the Equation (6.1), where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

e.q (6.1)

Precision focuses on the correctness of positive predictions, representing the proportion of correctly predicted positive instances among all predicted positives. It is defined using Equation (6.2), where a higher precision value indicates that the model produces fewer false positives.

$$Precision = \frac{TP}{TP + FP}$$

e.q (6.2)

Recall measures the ability of the model to correctly identify positive instances, expressed as the ratio of correctly predicted positives to the total actual positives. It is defined using Equation (6.3), where a higher recall value indicates that the model is effective at capturing actual positives, with fewer false negatives.

$$Recall = \frac{TP}{TP + FN}$$

e.q (6.3)

The F1-score provides a harmonic mean between precision and recall, offering a balanced evaluation when both metrics are important. It is calculated as shown in Equation (6.4). Unlike accuracy, which only measures overall correctness, the F1-score evaluates how well the model balances precise predictions with capturing relevant cases.

$$F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

e.q (6.4)

Table 6.4 Evaluation score of DNN, U-Net based CNN and Random Forest

Metric	**DNN**	**U-Net CNN**	**Random Forest**
Training Accuracy	85.58%	92.85%	95.30%
Validation Accuracy	88.41%	91.14%	83.35%
Test Accuracy	88.70%	90.36%	85.00%
Precision	88.99%	90.46%	83.80%
Recall	88.70%	90.36%	83.35%
F1-score	88.71%	90.38%	83.43%

The Deep Neural Network (DNN) achieved a training accuracy of 85.58%, suggesting a relatively steady learning process without severe overfitting. Its validation accuracy of 88.41% and test accuracy of 88.70 indicate strong generalization ability, as the model performed better on unseen data than during training. This could cause by the effective regularisation and the model's capacity to capture the essential structure of the dataset without overfitting to noise.

Furthermore, the DNN demonstrates a precision of 88.99% and recall of 88.70%, resulting in an F1-score of 88.71%. These balanced scores confirm that the model is effective at minimising both false positive and false negatives. However, while the DNN produced stable results, its overall performance was slightly lower compared to the U-Net based Convolutional Neural Network (CNN), suggesting that it captured feature in the data less effectively.

The U-Net based CNN outperformed the other models across most metrics. It achieved a training accuracy of 92.85%, with validation accuracy of 91.14% and test accuracy of 90.36%. Unlike Random Forest, the gap between training and validation accuracy was relatively small, indicating strong generalization.

The CNN also produced the highest precision of 90.46%, recall of 90.36% and F1-score of 90.38%, demonstrating its effectiveness in consistently identifying BMI cases while maintaining predictive correctness. The architecture's encoder-decoder structure, coupled with skip connections, likely allowed it to capture both global and local feature patterns from the input data, which enhanced its predictive performance. Overall, the U-Net CNN established itself as the most robust and generalisable model in this study, showing an excellent balance between accuracy and class sensitivity.

The Random Forest model achieved the highest training accuracy at 95.30%, suggesting that it learned the training dataset extremely well. However, this strong performance raised concerns about overfitting. The validation accuracy dropped to 83.35%, and the test accuracy further settled at 85.00%, both significantly lower than the training accuracy. This highlights the model's limited ability to generalise to unseen data, a common issue when ensemble models overfit to specific patterns or noise in the training set.

In terms of other metrics, the Random Forest achieved precision of 83.80%, recall of 83.35%, and F1-score of 83.34%. Despite being acceptable, these results fall below of those of the DNN and U-Net CNN, suggesting that the Random Forest produced more false positives and false negatives overall. This suggests that while Random Forest was effective in capturing relationships within the training data, it struggled to maintain consistency when faced with new samples.

To further support the evaluation, proving U-Net CNN achieve better model performance than the DNN, the training and validation accuracy curves for both DNN and U-Net CNN were plotted, as shown in Figure 6.1 and Figure 6.2.

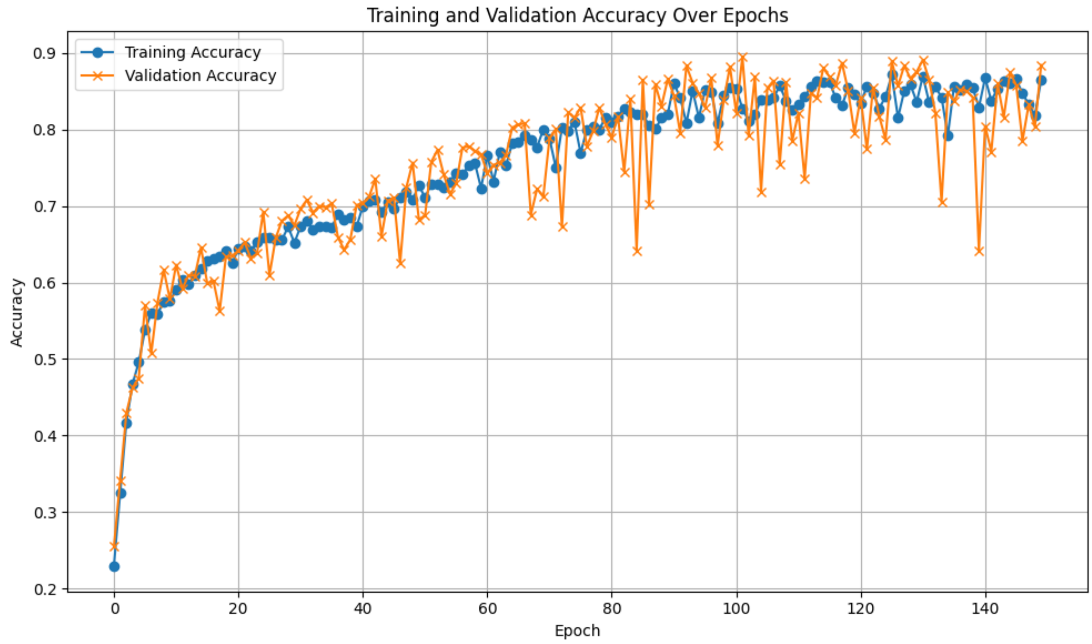


Figure 6.1 Training and validation accuracy curves of DNN model

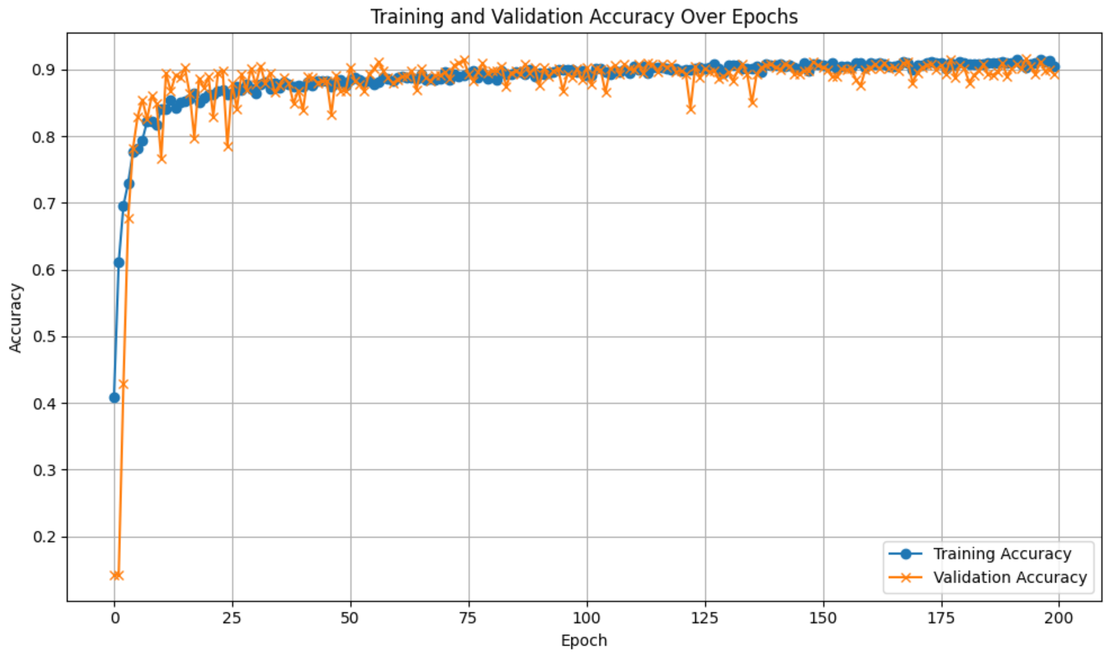


Figure 6.2 Training and validation accuracy curves of U-Net CNN model

In the DNN model (Figure 6.1), the training accuracy showed a steady increase throughout the epochs. However, the validation accuracy fluctuated significantly, with irregular sharp drops and recoveries. This instability indicates that the DNN model had difficulties generalizing to unseen validation data and was prone to slight overfitting, as reflected by the divergence between training and validation accuracy curves after approximately 80 epochs.

On the other hand, the U-Net CNN model (Figure 6.2) demonstrated a much smoother and more stable learning curve. The training and validation accuracies increased simultaneously with minimal fluctuation and remained closely aligned throughout the training process. This close alignment suggests that the U-Net-based CNN model generalised well to new data, reducing the risk of overfitting and ensuring consistent performance.

Overall, these results confirm that the U-Net CNN model demonstrates better generalization ability, higher predictive performance, and greater robustness compared to the DNN model.

In summary, among the three models, the U-Net CNN achieved the best balance between training, validation, and test performance, making it the most reliable model for BMI case classification. The DNN also performed competitively, showing consistent results across datasets without significant signs of overfitting. In contrast, the Random Forest, despite its high training accuracy, showed weaker generalisation, underlining the risks of overfitting in ensemble methods. The evaluation confirms that deep learning models, particularly the U-Net based CNN, are most suitable for this dataset, as they leverage feature representation learning to achieve higher model performance compared to traditional machine learning approaches.

6.2 Model Architecture Analysis

6.2.1 Analysis between DNN and U-Net CNN

Although the DNN model achieved high accuracy during training, its performance was consistently lower than the U-Net-based CNN model across validation and test datasets.

<u>Lack of Feature Extraction Capability</u>

The DNN model consists of dense layers. While dense layers can model complex non-linear relationships, they are not specialized for feature extraction. Unlike convolutional layers, dense layers do not learn spatial or local patterns effectively. In contrast, the U-Net CNN architecture utilises 1D convolutional layers in the encoder path, which enables the model to automatically learn hierarchical and localized features from the input data. As a result, the CNN is able to capture meaningful patterns that the DNN may overlook. [17]

No Encoding-Decoding Structure in DNN

The U-Net-based CNN employs an encoder-decoder structure with skip connections. This allows the network to learn more comprehensive structure for accurate classification [35]. On the other hand, the DNN lacks such an encoding-decoding mechanism and simply stacks dense layers, which can cause the model to struggle in capturing complex features relationship necessary for BMI case classification.

Insufficient Depth and Parameter Efficiency

Although the DNN architecture used multiple hidden layers, it was still relatively shallow compared to the deeper feature extraction performed by the U-Net CNN's convolutional blocks. CNN models are parameter-efficient because convolutional layers share weights across spatial locations, enabling deeper architectures without massive parameter explosion. Dense layer, however, require independent weights for each connection, learning to less efficient learning. This makes DNNs more prone to overfitting while training [36].

Overfitting Behaviour

Finally, the training curves in Figure 6.1 showed that the DNN model began to overfit after approximately 80 epochs, as reflected by the increasing divergence between training and validation accuracy. This suggests that despite high training performance, the DNN failed to generalise to new data. In contrast, the U-Net CNN maintained a tight match between training and validation performance in Figure 6.2, indicating better control over overfitting due to its regularization and architectural advantages.

Thus, the architectural limitations of DNN have collectively explained why the U-Net-based CNN model achieved better performance for BMI case classification in this application.

6.2.2 Analysis between Deep Learning and Machine Learning Models

While deep learning models such as DNN and U-Net CNN leverage layered architectures to automatically extract and transform features, the Random Forest model follows a fundamentally different machine learning approach.

Ensemble-Based Feature Splitting

Random Forest is built upon an ensemble of decision trees, where each of the trees trained on bootstrapped samples with randomised feature subsets. This design reduces variance and improves robustness, explaining its strong training performance. However, its recursive splitting process relies on predefined thresholds and does not extract hierarchical feature interactions as deep learning do. This makes Random Forest effective for structured tabular data but less capable of learning layered patterns.

Generalisation Challenges

Compared to U-Net CNN, Random Forest struggle to maintain performance across validation and test sets despite achieving high training accuracy. The absence of architectural mechanisms such as feature extraction or encoding decoding structure constrained its generalisation ability. Deep learning models, particularly the U-Net CNN, were able to balance complexity with efficiency, while Random Forest showed signs of memorising training specific patterns

Overall, machine learning like Random Forest provided strong baseline results, but deep learning models like U-Net CNN captured more complex feature patterns, making them better suited for this dataset.

Chapter 7
Conclusion and Recommendation

7.1 Conclusion

The project has designed, developed, and evaluated a personalised workout and dietary guidance mobile application that integrates deep learning to predict Body Mass Index (BMI) cases, recommend workouts and nutrition, and foster community engagement for healthier lifestyle management. The system was implemented using Flutter as the primary development framework, Firebase for backend services, and TensorFlow Lite for deploying predictive models. Additional support was provided through external resources such as the Gemini API for chatbot interactions and publicly available datasets for nutrition library.

The project successfully achieved its objective by combining several core modules including user registration and login, weight logging, goal management, meal tracking, exercise logging and recommendations, and community interaction features. The user of Firestore collections allows efficient organisation of user-specific data. Meanwhile, the predictive model for BMI case classification enhances the application's practical value beyond simple data logging.

The project has implemented and evaluated three predictive approaches including DNN, U-Net based CNN, and Random Forest, where the U-Net CNN model achieved the strongest model performance due to its ability to capture localized feature interactions through convolutional and skip-connection structures.

With the comprehensive solutions of the system, users will experience not only accurate BMI category predictions but also meaningful lifestyle support through structured workout plans, nutritional guidance, and progress monitoring. By providing recommendations and tracking, the system empowers users to make informed decisions in their daily health routines, fostering consistency and long-term behavioural changes.

Ultimately, the system is designed to not only as a predictive tool, but as a holistic digital companion for health management, supporting users in achieving sustainable well-being and fitness outcomes.

7.2 Recommendation

Although the project successfully achieved the proposed objectives, but it still remains opportunities for further enhancement and refinement. The following recommendations are suggested for future work:

1. **Model Improvement and Expansion**

   While the implemented models performed reasonably well, their robustness could be improved by training on a larger dataset with broader demographic coverage. This would increase the generalisability of predictions across diverse user populations.

2. **System Features and Functionality**

   The system may be integrated with smartwatches or fitness trackers, which allow real-time data collection on steps, heart rate, and calories burned, reducing reliance on manual inputs.

3. **User Experience and Engagement**

   The system could be expanded with gamification elements by adding achievement badges, progress streaks, or leaderboards, which could further motivate users to engage consistently with the application.

4. **Technical Enhancements**

   The system can be further enhanced by introducing offline support for core features, ensuring that the system remains both adaptive and accessible for a wide range of use.

REFERENCES

[1] World Health Organisation, “Obesity and overweight,” World Health Organisation, 2024. [Online]. Available: https://www.who.int/news room/fact-sheets/detail/obesity-and-overweight.

[2] L. K. Herrmann and J. Kim, “The fitness of apps: a theory-based examination of mobile fitness app usage over 5 months,” mHealth, vol. 3, pp. 2–2, Jan. 2017. [Abstract]. Available: mHealth,

[3] Sidharthan, Chinta. "Did the use of sports and fitness apps change before and after the COVID-19 pandemic?". News-Medical, Aug. 23, 2023. [Online]. Available: https://www.news-medical.net/news/20230823/Did-the-use-of sports-and-fitness-apps-change-before-and-after-the-COVID-19 pandemic.aspx

[4] B. Trukeschitz, S. Eisenberg, C. Schneider, and U. Schneider, “Exploring the effectiveness of a fitness-app prototype for home care service users in Austria and Italy,” Health & Social Care in the Community, Jan. 2022. [Abstract]. Available: Wiley Online Library, https://doi.org/10.1111/hsc.13733.

[5] “Running App and Cycling App | Strava,” www.strava.com. [Online]. Available: https://www.strava.com/mobile.

[6] “Fittr - World’s Largest Fitness & Nutrition Community,” www.fittr.com. [Online]. Available: https://www.fittr.com/

[7] MyNetDiary, “MyNetDiary - Free Calorie Counter and Diet Assistant,” MyNetDiary. [Online]. Available: https://www.mynetdiary.com/

[8] C. Ang, “Fitness apps grew by nearly 50% during the first half of 2020, study finds,” World Economic Available: Forum, Sep. 15, 2020. [Online]. https://www.weforum.org/agenda/2020/09/fitness-apps-gym health-downloads/

[9] “Free Android Calorie Counter, Android Calorie Tracker | MyFitnessPal.com,” Myfitnesspal.com, 2016. [Online]. Available: https://www.myfitnesspal.com/mobile/android

[10] “Why do users abandon fitness apps?,” autentika.com. https://autentika.com/blog/why-do-users-abandon-fitness-apps

[11] M. Mahmoud, “fitness exercises using BFP & BMI,” Kaggle.com, 2024. https://www.kaggle.com/datasets/mustafa20635/fitness-exercises-using-bfp-and-bmi/data

[12] T. Silva, “Understanding Types of Machine Learning Models,” Aug. 27, 2024. https://www.clicdata.com/blog/machine-learning-models-types/

[13] IBM, “What Is Machine learning?,” IBM, Sep. 22, 2021. https://www.ibm.com/think/topics/machine-learning

[14] L. Breiman, “Random Forests,” Jan. 2001. Available: https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf

[15] E. Scornet, G. Biau, and J.-P. Vert, “Consistency of random forests,” The Annals of Statistics, vol. 43, no. 4, pp. 1716–1741, Aug. 2015, doi: https://doi.org/10.1214/15-aos1321.

[16] T. M. Oshiro, P. S. Perez, and J. A. Baranauskas, “How Many Trees in a Random Forest?,” Machine Learning and Data Mining in Pattern Recognition, vol. 7376, pp. 154–168, 2012, doi: https://doi.org/10.1007/978-3-642-31537-4_13.

[17] Y. LeCun, Y. Bengio, and G. Hinton, “Deep Learning,” Nature, vol. 521, no. 7553, pp. 436–444, May 2015.

[18] Shams Forruque Ahmed et al., “Deep learning modelling techniques: current progress, applications, advantages, and challenges,” Artificial Intelligence Review, vol. 56, no. 1, Apr. 2023, doi: https://doi.org/10.1007/s10462-023-10466-8.

[19] I. Goodfellow, Y. Bengio, and A. Courville, “Deep Learning,” MIT Press, Nov. 18, 2016. https://mitpress.mit.edu/9780262035613/deep-learning/

[20] Alessio Gozzoli, “Practical Guide to Hyperparameters Optimization for Deep Learning Models,” FloydHub Blog, Sep. 05, 2018. https://floydhub.ghost.io/guide-to-hyperparameters-search-for-deep-learning-models/

[21] “Gradient-based learning applied to document recognition - IEEE Journals & Magazine,” Ieee.org, 2019. https://ieeexplore.ieee.org/document/726791

[22] S. Sharma and A. Kaur, “Prediction and Detection of Epilepsy Seizures Using Deep Learning Based Convolutional Neural Networks Models,” pp. 1–8, Dec. 2023, doi: https://doi.org/10.1109/icaiihi57871.2023.10489738.

[23] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” Journal of Machine Learning Research, vol. 15, pp. 1929–1958, 2014

[24] D. Nam and A. Pak, “OVERVIEW OF TRANSFORMER-BASED MODELS FOR MEDICAL IMAGE SEGMENTATION,” Scientific Journal of Astana IT University, pp. 64–75, Mar. 2023, doi: https://doi.org/10.37943/13BKBF2003.

[25] Y. Lee, W. Sim, J. Park, and J. Lee, "Evaluation of hyperparameter combinations of the U-net model for land cover classification," Forests, vol. 13, no. 11, p. 1813, 2022. doi:10.3390/f13111813.

[26] The and C. Moore, “The 2022 GQ Fitness Awards: The Best Workout Clothes, Gear, Tech, and Accessories,” GQ, Sep. 26, 2022. [Online]. Available: https://www.gq.com/story/gq-fitness-awards-2022

[27] M. D. Mifflin, S. T. St Jeor, L. A. Hill, B. J. Scott, S. A. Daugherty, and Y. O. Koh, “A new predictive equation for resting energy expenditure in healthy individuals,” The American Journal of Clinical Nutrition, vol. 51, no. 2, pp. 241–247, Feb. 1990, doi: https://doi.org/10.1093/ajcn/51.2.241.

[28] D. Frankenfield, L. Roth-Yousey, and C. Compher, “Comparison of Predictive Equations for Resting Metabolic Rate in Healthy Nonobese and Obese Adults: A Systematic Review,” Journal of the American Dietetic Association, vol. 105, no. 5, pp. 775–789, May 2005.

[29] American College of Sports Medicine, “Physical Activity Guidelines,” ACSM, 2025. https://acsm.org/education-resources/trending-topics-resources/physical-activity-guidelines/

[30] D. Hall, “This handout lists the intensity -the MET level -of various physical activities. What level is right for you?,” 2008. Available: https://media.hypersites.com/clients/1235/filemanager/MHC/METs.pdf

[31] Kumar, G., & Bhatia, P. K., "Impact of agile methodology on software development process," International Journal of Computer Technology and Electronics Engineering (IJCTEE), 2(4), 46-50, July, 2012

[32] S. Hastie and S. Wojewoda, “Standish Group 2015 Chaos Report – Q&A with Jennifer Lynch,” October, 2015. [Online]. Available: https://www.infoq.com/articles/standish-chaos-2015/

[33] R. Paredes, “Waterfall Methodology: The Pros and Cons,” SafetyCulture, May 25, 2023. [Online]. Available:https://safetyculture.com/topics/waterfall methodology/

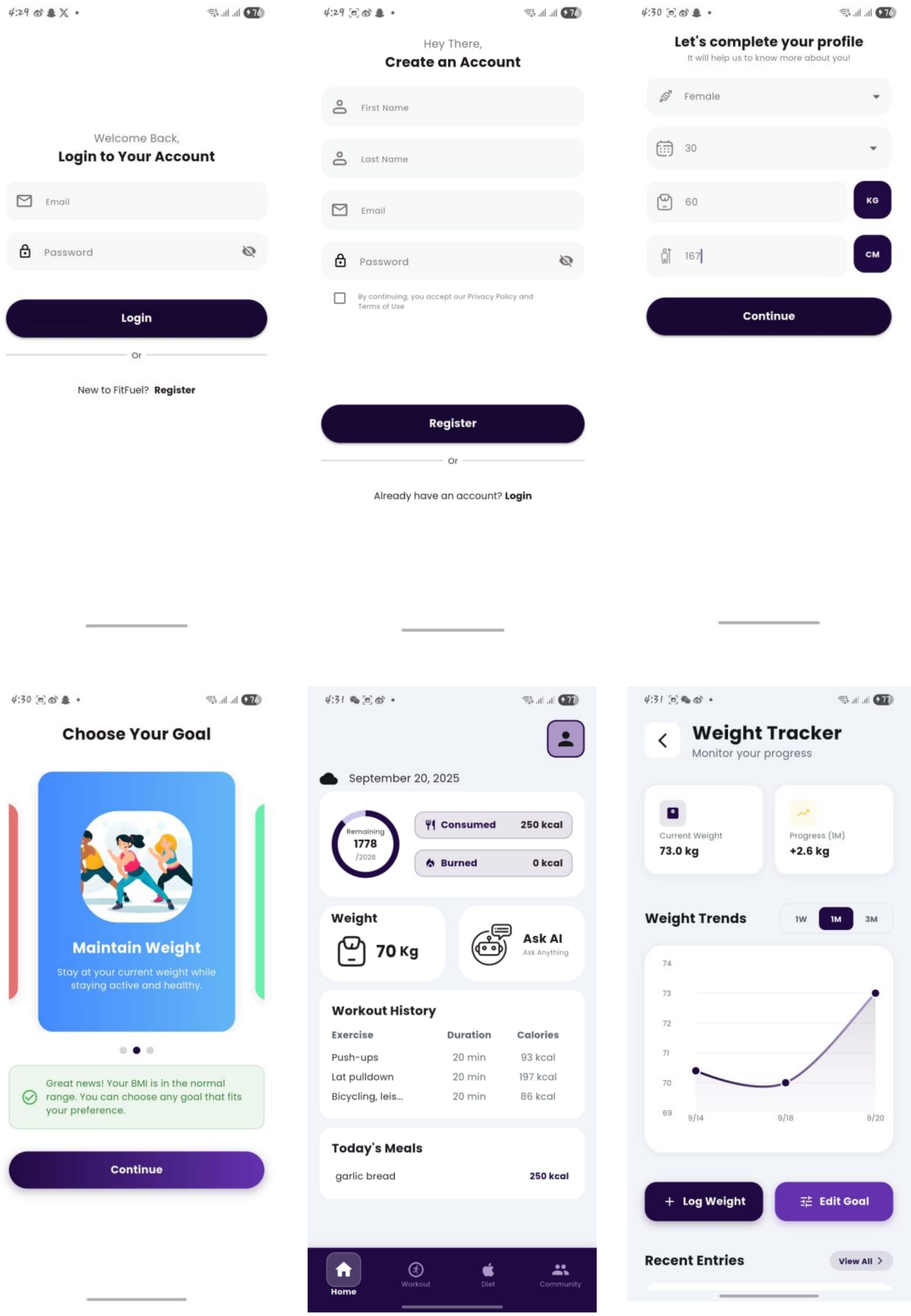
[34] google, “Classification: Accuracy, recall, precision, and related metrics,” Google for Developers, 2024. https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall

[35] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," May 2015. Available: https://arxiv.org/pdf/1505.04597

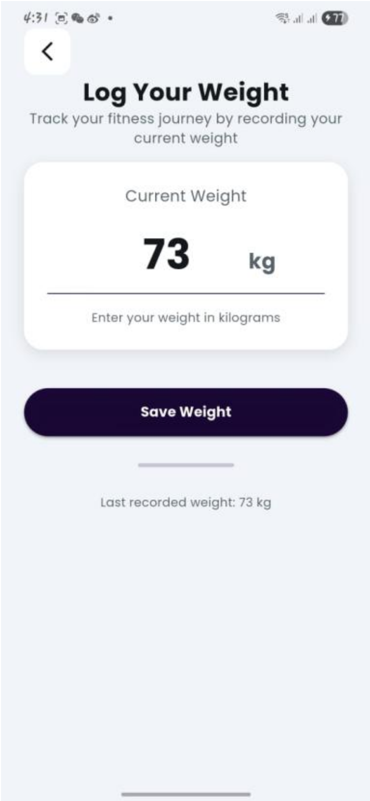
[36] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning," MIT Press, Nov. 18, 2016. https://mitpress.mit.edu/9780262035613/deep-learning/

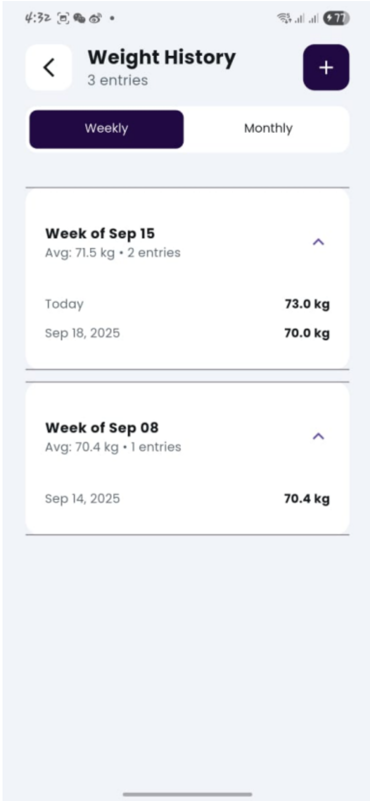
APPENDIX 1

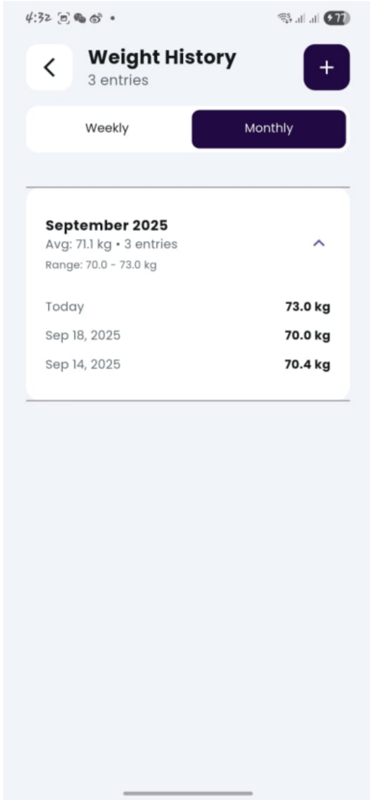
Application Interface Design

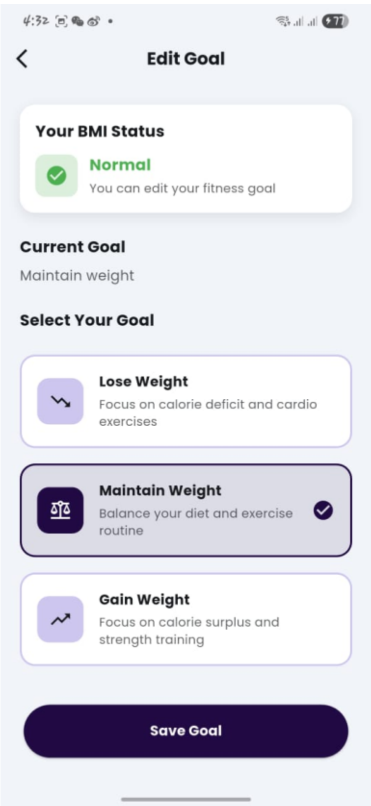


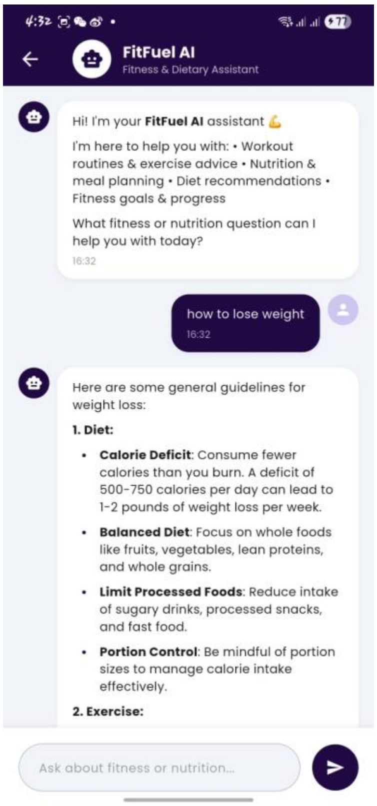
APPENDIX

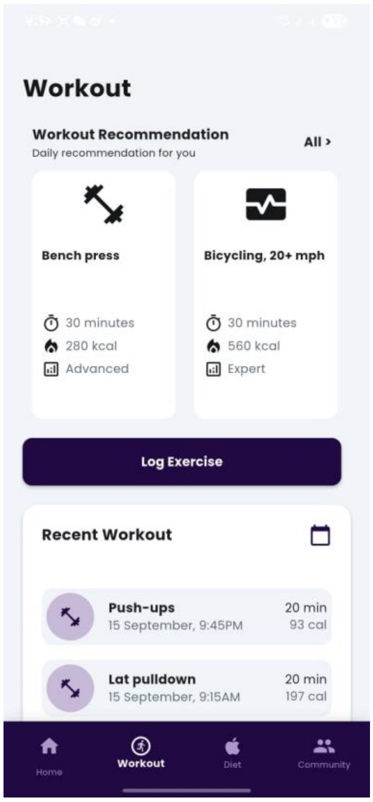

Log Your Weight
Track your fitness journey by recording your current weight
Current Weight
73
kg
Enter your weight in kilograms
Save Weight
Last recorded weight: 73 kg

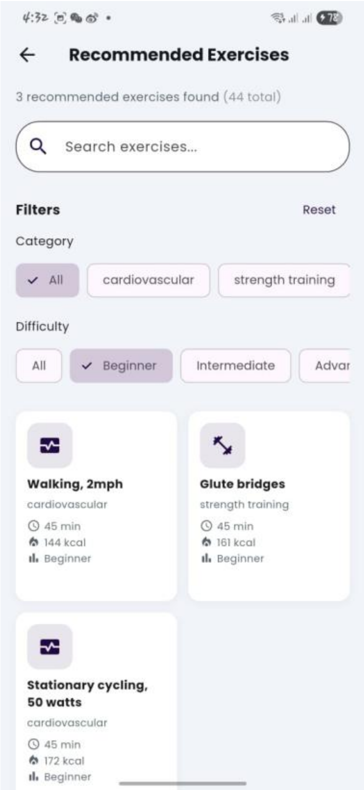

Weight History
3 entries
Weekly
Monthly
Week of Sep 15
Avg: 71.5 kg • 2 entries
Today
73.0 kg
Sep 18, 2025
70.0 kg
Week of Sep 08
Avg: 70.4 kg • 1 entries
Sep 14, 2025
70.4 kg

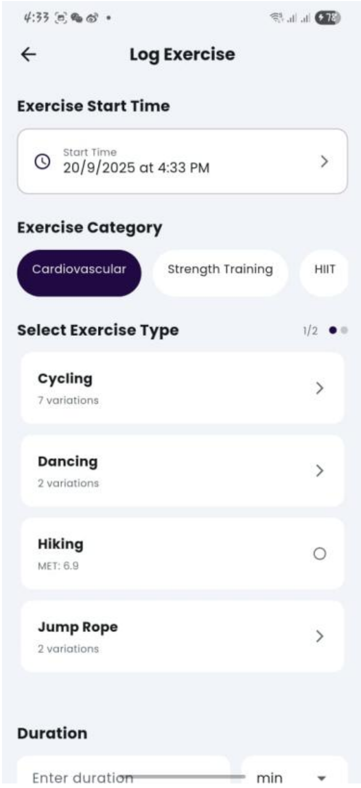

Weight History
3 entries
Weekly
Monthly
September 2025
Avg: 71.1 kg • 3 entries
Range: 70.0 - 73.0 kg
Today
73.0 kg
Sep 18, 2025
70.0 kg
Sep 14, 2025
70.4 kg

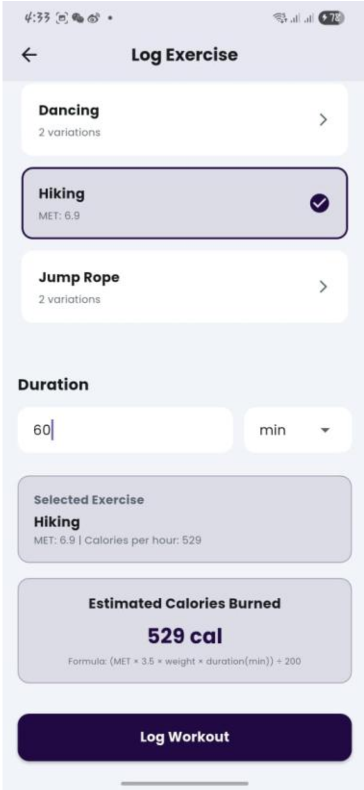

Edit Goal
Your BMI Status
Normal
You can edit your fitness goal
Current Goal
Maintain weight
Select Your Goal
Lose Weight
Focus on calorie deficit and cardio exercises
Maintain Weight
Balance your diet and exercise routine
Gain Weight
Focus on calorie surplus and strength training
Save Goal

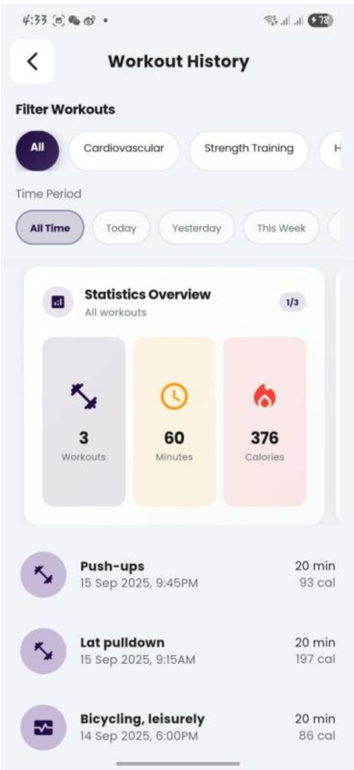

FitFuel AI
Fitness & Dietary Assistant
Hi! I'm your FitFuel AI assistant 💪
I'm here to help you with: • Workout routines & exercise advice • Nutrition & meal planning • Diet recommendations • Fitness goals & progress
What fitness or nutrition question can I help you with today?
16:32
how to lose weight
16:32
Here are some general guidelines for weight loss:
1. Diet:
Calorie Deficit: Consume fewer calories than you burn. A deficit of 500-750 calories per day can lead to 1-2 pounds of weight loss per week.
Balanced Diet: Focus on whole foods like fruits, vegetables, lean proteins, and whole grains.
Limit Processed Foods: Reduce intake of sugary drinks, processed snacks, and fast food.
Portion Control: Be mindful of portion sizes to manage calorie intake effectively.
2. Exercise:
Ask about fitness or nutrition...

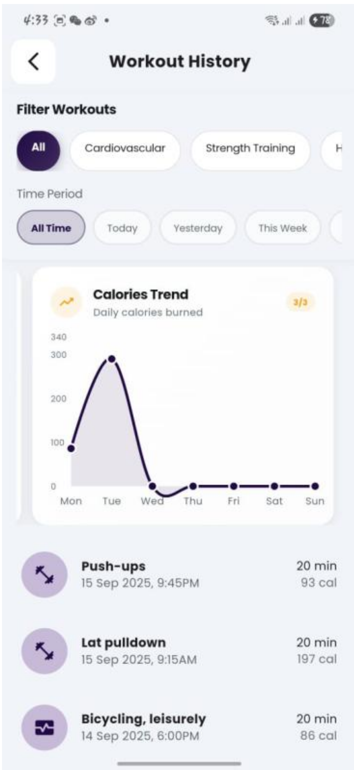

Workout
Workout Recommendation
All >
Daily recommendation for you
Bench press
30 minutes
280 kcal
Advanced
Bicycling, 20+ mph
30 minutes
560 kcal
Expert
Log Exercise
Recent Workout
Push-ups
15 September, 9:45PM
20 min
93 cal
Lat pulldown
15 September, 9:15AM
20 min
197 cal
Home
Workout
Diet
Community

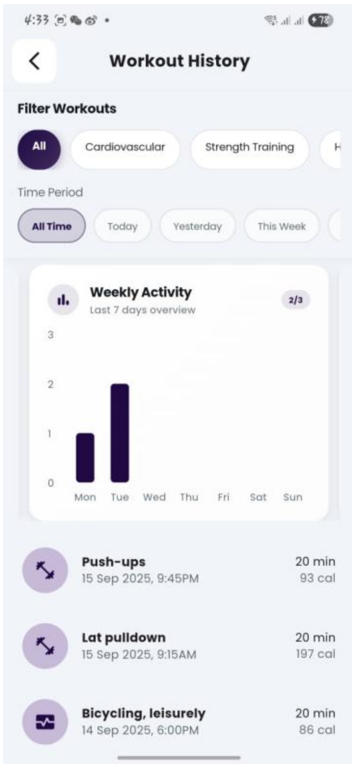

Recommended Exercises
3 recommended exercises found (44 total)
Search exercises...
Filters
Reset
Category
All
cardiovascular
strength training
Difficulty
All
Beginner
Intermediate
Walking, 2mph
cardiovascular
45 min
144 kcal
Beginner
Glute bridges
strength training
45 min
161 kcal
Beginner
Stationary cycling, 50 watts
cardiovascular
45 min
172 kcal
Beginner


Log Exercise
Exercise Start Time
Start Time
20/9/2025 at 4:33 PM
Exercise Category
Cardiovascular
Strength Training
HIIT
Select Exercise Type
1/2
Cycling
7 variations
Dancing
2 variations
Hiking
MET: 6.9
Jump Rope
2 variations
Duration
Enter duration
min

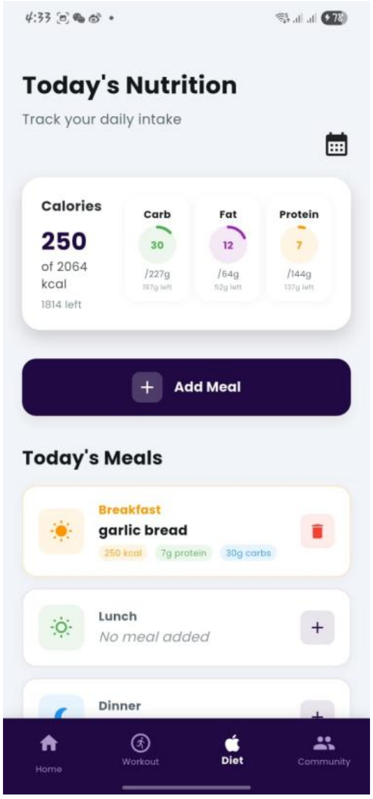

Log Exercise
Dancing
2 variations
Hiking
MET: 6.9
Jump Rope
2 variations
Duration
60
min
Selected Exercise
Hiking
MET: 6.9 | Calories per hour: 529
Estimated Calories Burned
529 cal
Log Workout

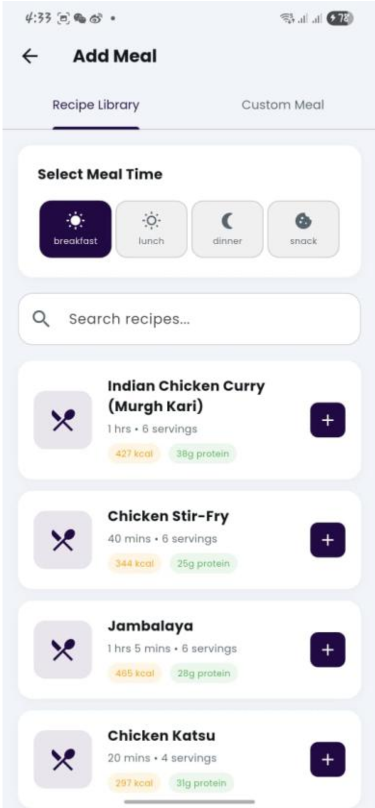

Workout History
Filter Workouts
All
Cardiovascular
Strength Training
Time Period
All Time
Today
Yesterday
This Week
Statistics Overview
All workouts
1/3
3
Workouts
60
Minutes
376
Calories
Push-ups
15 Sep 2025, 9:45PM
20 min
93 cal
Lat pulldown
15 Sep 2025, 9:15AM
20 min
197 cal
Bicycling, leisurely
14 Sep 2025, 6:00PM
20 min
86 cal

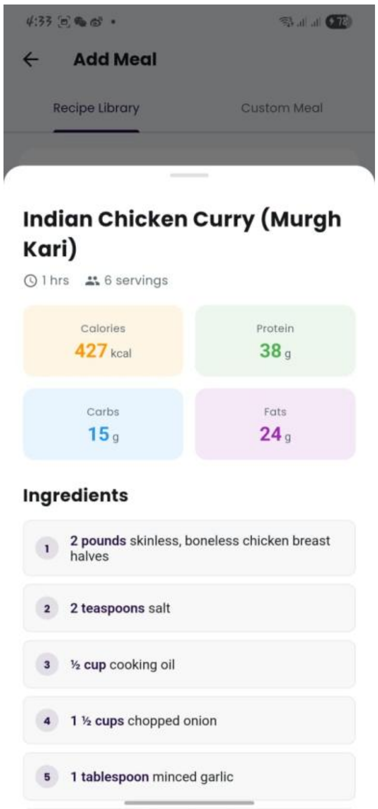

Workout History
Filter Workouts
All
Cardiovascular
Strength Training
Time Period
All Time
Today
Yesterday
This Week
Calories Trend
Daily calories burned
3/3
340
300
200
100
0
Mon
Tue
Wed
Thu
Fri
Sat
Sun
Push-ups
15 Sep 2025, 9:45PM
20 min
93 cal
Lat pulldown
15 Sep 2025, 9:15AM
20 min
197 cal
Bicycling, leisurely
14 Sep 2025, 6:00PM
20 min
86 cal

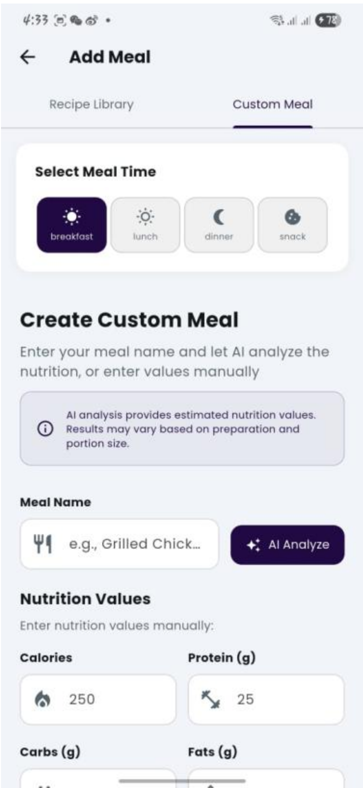

Workout History
Filter Workouts
All
Cardiovascular
Strength Training
Time Period
All Time
Today
Yesterday
This Week
Weekly Activity
Last 7 days overview
2/3
3
2
1
0
Mon
Tue
Wed
Thu
Fri
Sat
Sun
Push-ups
15 Sep 2025, 9:45PM
20 min
93 cal
Lat pulldown
15 Sep 2025, 9:15AM
20 min
197 cal
Bicycling, leisurely
14 Sep 2025, 6:00PM
20 min
86 cal

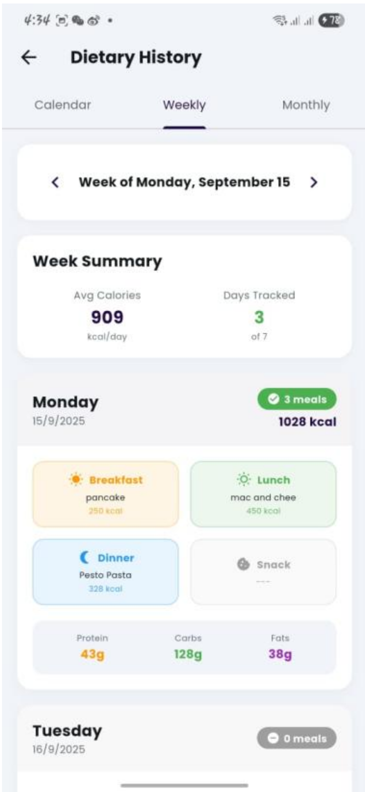
APPENDIX

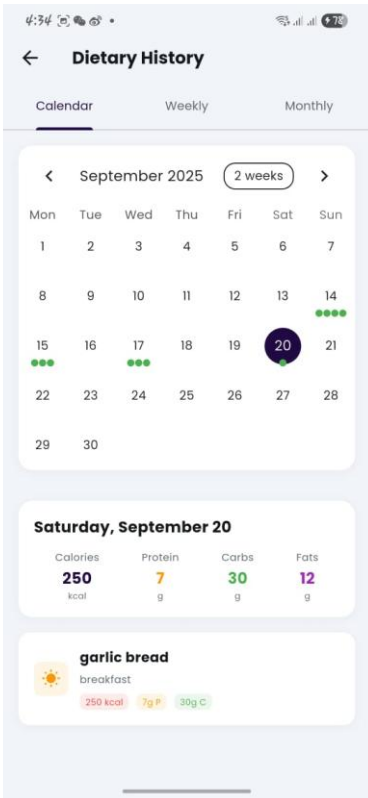

Today's Nutrition
Track your daily intake
Calories
250
of 2064
kcal
1814 left
Carb
30
/227g
Fat
12
/64g
Protein
7
/144g
Add Meal
Today's Meals
Breakfast
garlic bread
250 kcal
7g protein
30g carbs
Lunch
No meal added
Dinner
Home
Workout
Diet
Community


Add Meal
Recipe Library
Custom Meal
Select Meal Time
breakfast
lunch
dinner
snack
Search recipes...
Indian Chicken Curry (Murgh Kari)
1 hrs • 6 servings
427 kcal
38g protein
Chicken Stir-Fry
40 mins • 6 servings
344 kcal
25g protein
Jambalaya
1 hrs 5 mins • 6 servings
465 kcal
28g protein
Chicken Katsu
20 mins • 4 servings
297 kcal
31g protein

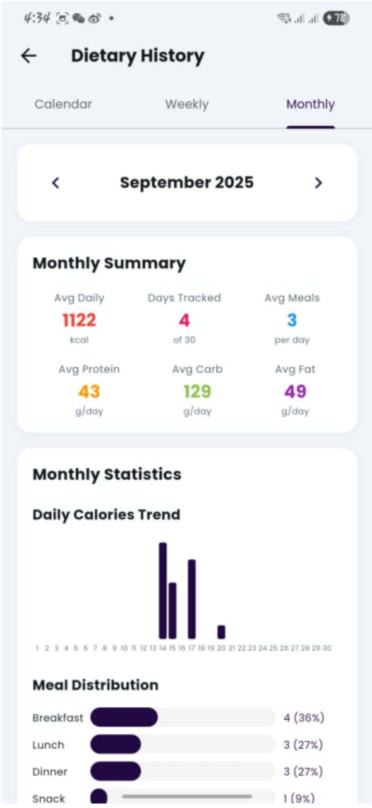

Add Meal
Recipe Library
Custom Meal
Indian Chicken Curry (Murgh Kari)
1 hrs
6 servings
Calories
427 kcal
Protein
38 g
Carbs
15 g
Fats
24 g
Ingredients
1 2 pounds skinless, boneless chicken breast halves
2 2 teaspoons salt
3 ½ cup cooking oil
4 1 ½ cups chopped onion
5 1 tablespoon minced garlic

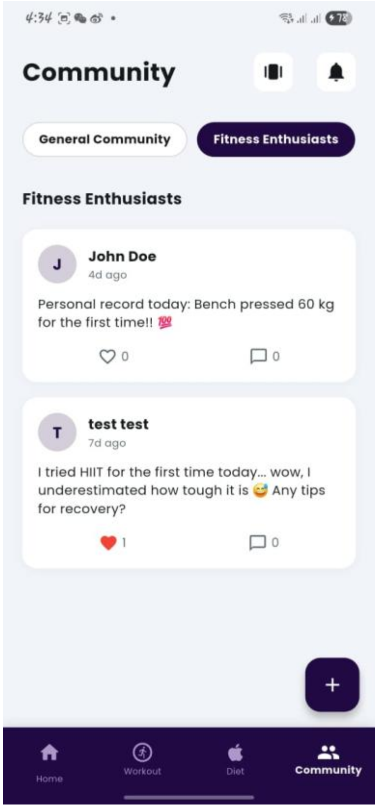

Add Meal
Recipe Library
Custom Meal
Select Meal Time
breakfast
lunch
dinner
snack
Create Custom Meal
Enter your meal name and let AI analyze the nutrition, or enter values manually
AI analysis provides estimated nutrition values. Results may vary based on preparation and portion size.
Meal Name
e.g., Grilled Chick...
AI Analyze
Nutrition Values
Enter nutrition values manually:
Calories
250
Protein (g)
25
Carbs (g)
Fats (g)

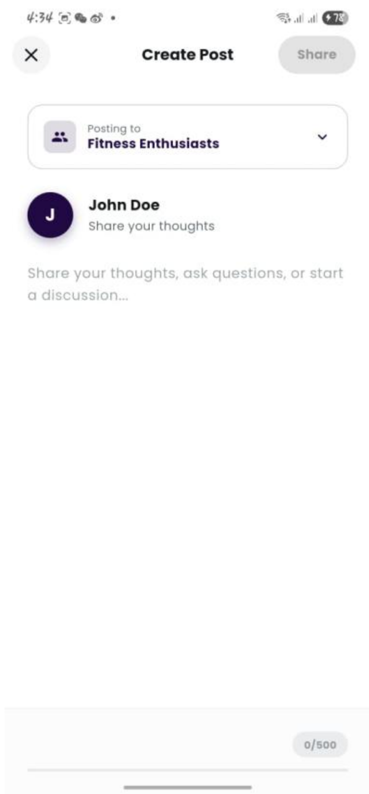

Dietary History
Calendar
Weekly
Monthly
Week of Monday, September 15
Week Summary
Avg Calories
909
kcal/day
Days Tracked
3
of 7
Monday
15/9/2025
3 meals
1028 kcal
Breakfast
pancake
250 kcal
Lunch
mac and chee
450 kcal
Dinner
Pesto Pasta
328 kcal
Snack
Protein
43g
Carbs
128g
Fats
38g
Tuesday
16/9/2025
0 meals

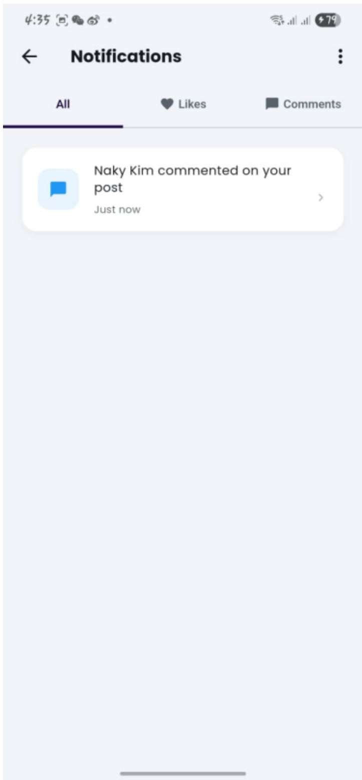

Dietary History
Calendar
Weekly
Monthly
September 2025
2 weeks
Mon
Tue
Wed
Thu
Fri
Sat
Sun
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30
Saturday, September 20
Calories
250
kcal
Protein
7
g
Carbs
30
g
Fats
12
g
garlic bread
breakfast
250 kcal
7g P
30g C

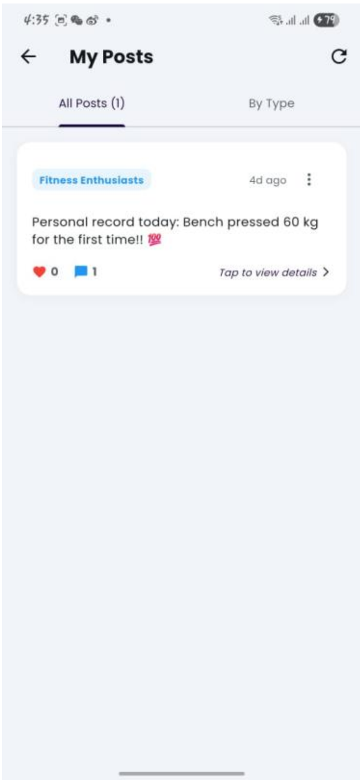
APPENDIX

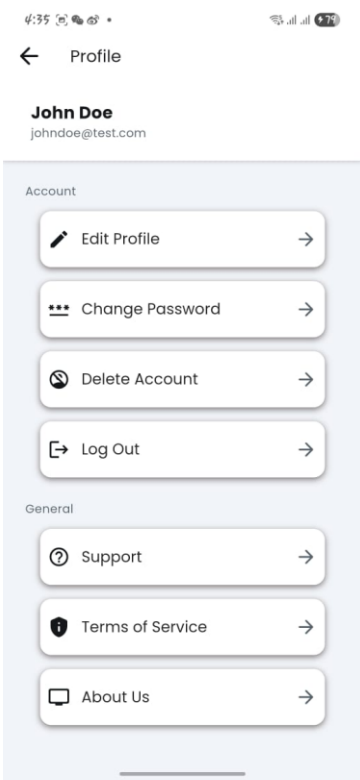

Dietary History
Calendar
Weekly
Monthly
September 2025
Monthly Summary
Avg Daily
1122
kcal
Days Tracked
4
of 30
Avg Meals
3
per day
Avg Protein
43
g/day
Avg Carb
129
g/day
Avg Fat
49
g/day
Monthly Statistics
Daily Calories Trend
Meal Distribution
Breakfast
4 (36%)
Lunch
3 (27%)
Dinner
3 (27%)
Snack
1 (9%)

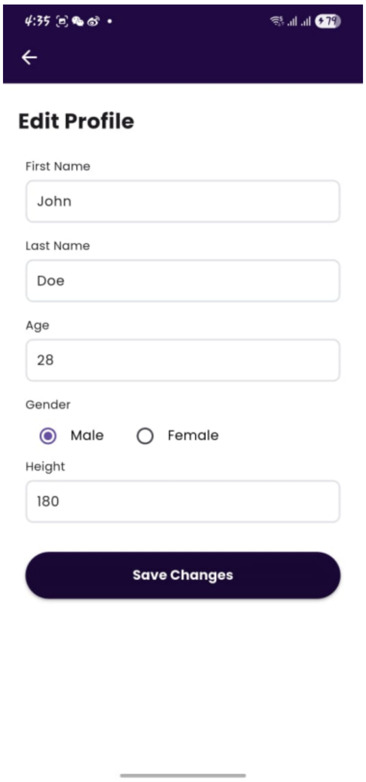

Community
General Community
Fitness Enthusiasts
Fitness Enthusiasts
John Doe
4d ago
Personal record today: Bench pressed 60 kg for the first time!! 💯
0
0
test test
7d ago
I tried HIIT for the first time today... wow, I underestimated how tough it is 😅 Any tips for recovery?
1
0
Home
Workout
Diet
Community

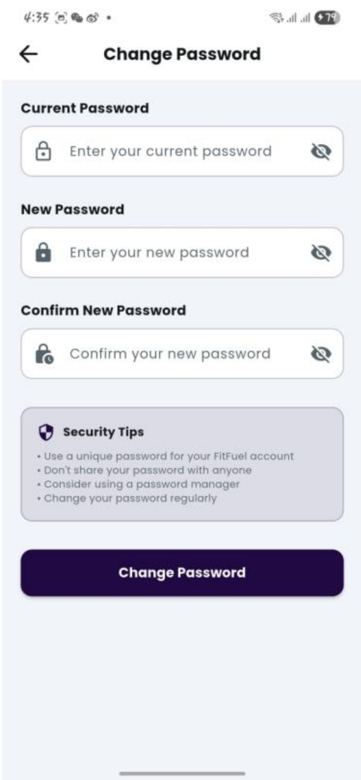

Create Post
Share
Posting to
Fitness Enthusiasts
John Doe
Share your thoughts
Share your thoughts, ask questions, or start a discussion...
0/500

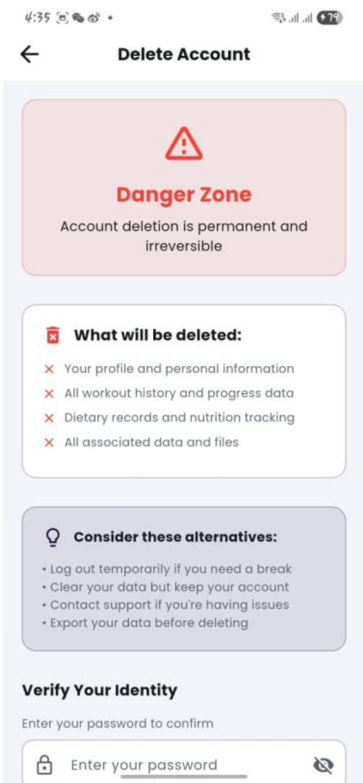

Notifications
All
Likes
Comments
Naky Kim commented on your post
Just now

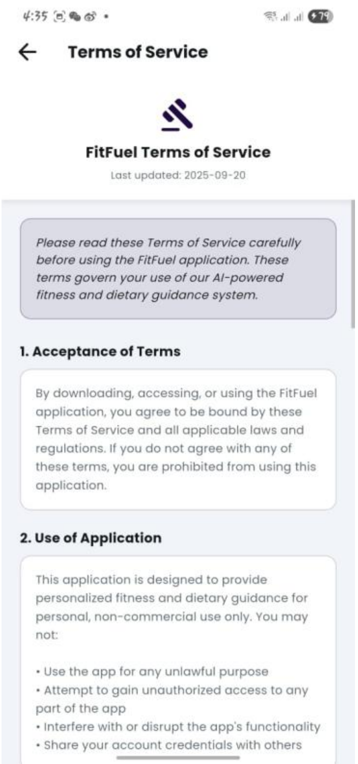

My Posts
All Posts (1)
By Type
Fitness Enthusiasts
4d ago
Personal record today: Bench pressed 60 kg for the first time!! 💯
0
1
Tap to view details

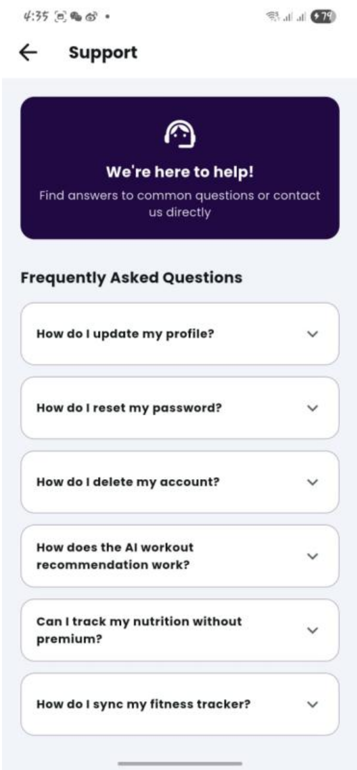

Profile
John Doe
johndoe@test.com
Account
Edit Profile
Change Password
Delete Account
Log Out
General
Support
Terms of Service
About Us

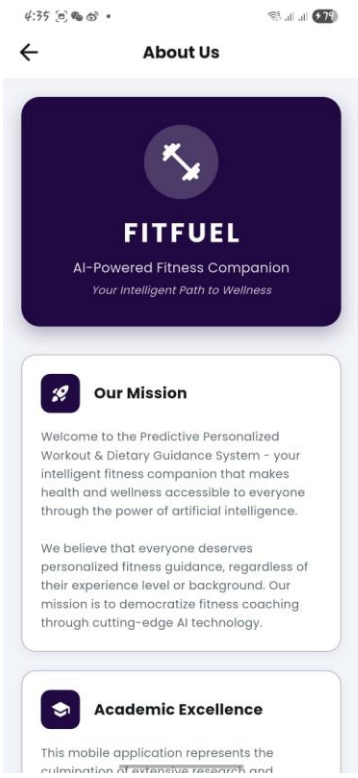

Edit Profile
First Name
John
Last Name
Doe
Age
28
Gender
Male
Female
Height
180
Save Changes


Change Password
Current Password
Enter your current password
New Password
Enter your new password
Confirm New Password
Confirm your new password
Security Tips
• Use a unique password for your FitFuel account
• Don't share your password with anyone
• Consider using a password manager
• Change your password regularly
Change Password


Delete Account
Danger Zone
Account deletion is permanent and irreversible
What will be deleted:
Your profile and personal information
All workout history and progress data
Dietary records and nutrition tracking
All associated data and files
Consider these alternatives:
• Log out temporarily if you need a break
• Clear your data but keep your account
• Contact support if you're having issues
• Export your data before deleting
Verify Your Identity
Enter your password to confirm
Enter your password


Terms of Service
FitFuel Terms of Service
Last updated: 2025-09-20
Please read these Terms of Service carefully before using the FitFuel application. These terms govern your use of our AI-powered fitness and dietary guidance system.
1. Acceptance of Terms
By downloading, accessing, or using the FitFuel application, you agree to be bound by these Terms of Service and all applicable laws and regulations. If you do not agree with any of these terms, you are prohibited from using this application.
2. Use of Application
This application is designed to provide personalized fitness and dietary guidance for personal, non-commercial use only. You may not:
• Use the app for any unlawful purpose
• Attempt to gain unauthorized access to any part of the app
• Interfere with or disrupt the app's functionality
• Share your account credentials with others


Support
We're here to help!
Find answers to common questions or contact us directly
Frequently Asked Questions
How do I update my profile?
How do I reset my password?
How do I delete my account?
How does the AI workout recommendation work?
Can I track my nutrition without premium?
How do I sync my fitness tracker?


About Us
FITFUEL
AI-Powered Fitness Companion
Your Intelligent Path to Wellness
Our Mission
Welcome to the Predictive Personalized Workout & Dietary Guidance System - your intelligent fitness companion that makes health and wellness accessible to everyone through the power of artificial intelligence.
We believe that everyone deserves personalized fitness guidance, regardless of their experience level or background. Our mission is to democratize fitness coaching through cutting-edge AI technology.
Academic Excellence
This mobile application represents the

APPENDIX 2

POSTER