

MEAL LOGGING AND NUTRIENT TRACKING APPLICATION

BY

WONG MEI KUAN

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)

FEBRUARY 2026

COPYRIGHT STATEMENT

© 2026 Wong Mei Kuan. All rights reserved.

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

ACKNOWLEDGEMENTS

I would like to express thanks and appreciation to my supervisor, Ts. Dr. Lim Seng Poh and my moderator, Puan Fatin Natasya binti Shuhaimi who have given me a golden opportunity to involve in this project. Besides that, they have given me a lot of guidance in order to complete this project. When I was facing problems in this project, the advice from them always assists me in overcoming the problems. Again, a million thanks to my supervisor and moderator.

Other than that, I would like to thank my friend, Chong Jia Hui who has provided a lot of assistance to me when completing this project. Although both of us are having different project and task scope, she is still willing to support me when I faced difficulties in developing this project.

ABSTRACT

In this project, a mobile application named FoodWise was developed for logging meals, tracking nutrient intake, and receiving personalised meal recommendations. Many individuals struggle to maintain balanced nutrition, especially those with health conditions, due to irregular eating habits and the lack of personalised dietary guidance. In the literature review, MyFitnessPal, YAZIO, HealthifyMe, Eatr, Foodvisor, and Lifesum were reviewed. These existing applications focus mainly on calorie counting and general nutrient tracking, and personalised meal recommendations based on health conditions are not available. As a result, balanced nutrient intake is difficult to manage. Therefore, the objective of this project is to develop a mobile application that can log meals, track nutrients, and generate personalised meal recommendations. The scope includes user authentication, health information input, daily nutrient breakdown, and personalised meal recommendations for five categories which are diabetes, hypertension, obesity, heart disease, and no chronic disease. The methodology was carried out using the Agile methodology. Flutter and Firebase were used to develop the mobile application, while FastAPI was used to deploy the recommendation model. A Linear Regression model was trained to generate recommended protein, carbohydrate, and fat intake values. The model achieved strong performance with an R^2 score of 0.936 and lower error values compared to Random Forest Regressor and Multilayer Perceptron Regressor models. In the conclusion, this project successfully developed a meal logging and nutrient tracking application that assists users in monitoring their daily nutrient intake while receiving personalised meal recommendation based on their health conditions. Other functions such as custom meal creation, community forum, personalised nutritional advice, and reminders were successfully developed for users. Moreover, the management of user accounts, admin registrations, meals, and reported posts and comments were also developed for admins. For future work, a longer period of nutrient progress tracking, meal recognition using image processing, integration of wearable devices, and expansion of meal database will be completed.

Area of Study (Maximum 2): Mobile Application

Keywords (Maximum 5): Mobile Application, Nutrient Tracking, Meal Recommendations, Meal Logging, Health Conditions

TABLE OF CONTENTS

TITLE PAGE	i
COPYRIGHT STATEMENT	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
TABLE OF CONTENTS	v
LIST OF FIGURES	viii
LIST OF TABLES	xiii
LIST OF SYMBOLS	xvi
LIST OF ABBREVIATIONS	xvii
CHAPTER 1 INTRODUCTION	1
1.1 Problem Statement and Motivation	1
1.2 Objectives	2
1.3 Project Scope	3
1.4 Contributions	4
1.5 Report Organisation	4
CHAPTER 2 LITERATURE REVIEW	5
2.1 Overview	5
2.2 MyFitnessPal	5
2.3 YAZIO	9
2.4 HealthifyMe	11
2.5 Eatr	13
2.6 Foodvisor	15
2.7 Lifesum	16
2.8 Comparison of the Previous Works	18
2.9 Limitation of Previous Works	18
2.10 Recommendation Models	19
2.11 Proposed Solutions	21

CHAPTER 3 SYSTEM METHODOLOGY	22
3.1 Design Specifications	22
3.1.1 Methodologies and General Work Procedures	22
3.2 System Design Diagram	25
3.2.1 System Architecture Diagram	25
3.2.2 Use Case Diagram	26
3.2.3 Use Case Description	27
3.2.4 Activity Diagram	29
3.3 Development of the Recommendation Models for Daily Nutrient Intake	30
3.3.1 Data Collection	30
3.3.2 Dataset Preparation	31
3.3.3 Models Training	31
3.4 Meal Database	33
3.5 Timeline	34
3.5.1 FYP1 Timeline	34
3.5.2 FYP2 Timeline	35
 CHAPTER 4 SYSTEM DESIGN	 36
4.1 System Block Diagram	36
4.2 System Components Specifications	36
4.3 Database Design	39
 CHAPTER 5 SYSTEM IMPLEMENTATION	 40
5.1 Hardware Setup	40
5.2 Software Setup	41
5.3 Setting and Configuration	42
5.4 System Operation	45
5.5 Recommendation Models for Daily Nutrients Intake	68
5.6 Implementation Issues and Challenges	70
5.7 Concluding Remark	71
 CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION	 72
6.1 System Testing and Performance Metrics	72

6.1.1	System Testing	72
6.1.2	System Performance Definition	72
6.1.3	Verification Plan	73
6.2	Testing Setup and Result	76
6.2.1	Recommendation Model Result Testing	76
6.2.2	Functional Testing of Proposed Application Modules	78
6.3	Project Challenges	80
6.4	Objectives Evaluation	80
6.5	Concluding Remark	81
CHAPTER 7	CONCLUSION AND RECOMMENDATION	82
7.1	Conclusion	82
7.2	Recommendation	82
REFERENCES		84
APPENDIX A		A-1
A.1	Use Case Description	A-1
A.2	Activity Diagram	A-24
A.3	Database Design	A-38
A.4	Functional Testing of Proposed Application Modules	A-51
APPENDIX B	POSTER	B-1

LIST OF FIGURES

Figure Number	Title	Page
Figure 2.2.1	Main Interface in MyFitnessPal	6
Figure 2.2.2	Meal Searching in MyFitnessPal	6
Figure 2.2.3	Meal Logging in MyFitnessPal	6
Figure 2.2.4	Custom Meal in MyFitnessPal	7
Figure 2.2.5	Progress Tracking in MyFitnessPal	8
Figure 2.2.6	Exercise Tracking in MyFitnessPal	8
Figure 2.3.1	Main Interface in YAZIO	9
Figure 2.3.2	Meal Searching in YAZIO	9
Figure 2.3.3	Meal Logging in YAZIO	9
Figure 2.3.4	Calorie Consumption and Nutrient Intake Tracking in YAZIO	10
Figure 2.3.5	Intermittent Fasting Page in YAZIO	10
Figure 2.4.1	Main Interface in HealthifyMe	11
Figure 2.4.2	Meal Searching in HealthifyMe	11
Figure 2.4.3	Meal Logging in HealthifyMe	12
Figure 2.4.4	AI Coach Ria in HealthifyMe	12
Figure 2.4.5	Other Trackers in HealthifyMe	12
Figure 2.4.6	Calorie Tracking in HealthifyMe	13
Figure 2.4.7	Progress Report in HealthifyMe	13
Figure 2.5.1	Main Interface in Eatr	14
Figure 2.5.2	Meal Searching in Eatr	14
Figure 2.5.3	Meal Logging in Eatr	14
Figure 2.5.4	Nutrient Tracking in Eatr	14
Figure 2.5.5	Custom Meal in Eatr	14
Figure 2.5.6	Meal Planning in Eatr	14
Figure 2.6.1	Main Interface in Foodvisor	15
Figure 2.6.2	Meal Searching and Logging in Foodvisor	15
Figure 2.6.3	Nutrient Tracking in Foodvisor	15
Figure 2.6.4	Custom Meal in Foodvisor	16

Figure 2.6.5	Water Tracking in Foodvisor	16
Figure 2.7.1	Main Interface in Lifesum	17
Figure 2.7.2	Meal Searching and Logging in Lifesum	17
Figure 2.7.3	Nutrient Tracking in Lifesum	17
Figure 2.7.4	Water, Veggie, and Fruit Count Tracking in Lifesum	17
Figure 3.1.1.1	Agile Methodology Phases	22
Figure 3.2.1.1	System Architecture Diagram	25
Figure 3.2.2.1	Use Case Diagram	26
Figure 3.2.4.1	Activity Diagram for Register Account	29
Figure 3.3.3.1	Random Forest Regressor Model Training	31
Figure 3.3.3.2	MLP Model Training	32
Figure 3.3.3.3	Linear Regression Model Training	33
Figure 3.4.1	Malaysian Food Composition Database	33
Figure 3.5.1.1	FYP1 Timeline	34
Figure 3.5.2.1	FYP2 Timeline	35
Figure 4.1.1	System Block Diagram	36
Figure 5.3.1	Android Studio Configured with Flutter Project and Emulator	42
Figure 5.3.2	Application Permissions Configured in AndroidManifest.xml	43
Figure 5.3.3	Firebase Project Configured for the Proposed Application	43
Figure 5.3.4	Backend API Successfully Deployed on Render	44
Figure 5.4.1	Welcome Page	45
Figure 5.4.2	User and Admin Registration Page	46
Figure 5.4.3	Login Page	46
Figure 5.4.4	Email Account Selection	46
Figure 5.4.5	Password Recovery Page	47
Figure 5.4.6	Password Reset Email	47
Figure 5.4.7	Password Reset Page	47
Figure 5.4.8	Email is Registered in Registration Page	48
Figure 5.4.9	Incorrect Format Email in Registration Page	48
Figure 5.4.10	Password Length Error in Registration Page	48
Figure 5.4.11	Health Conditions in User Home Page	49

Figure 5.4.12	Health Conditions Information in User Home Page	49
Figure 5.4.13	Nutrient Information in User Home Page	49
Figure 5.4.14	Progress Bars in Diary Page	50
Figure 5.4.15	Date Selector in Diary Page	50
Figure 5.4.16	Meal Periods and Water in Diary Page	50
Figure 5.4.17	Water Intake Logging in Diary Page	51
Figure 5.4.18	First Time Logging in Diary Page	51
Figure 5.4.19	Missing Yesterday Meals in Diary Page	51
Figure 5.4.20	Upper View of the Meal Details Page	51
Figure 5.4.21	Lower View of the Meal Details Page	51
Figure 5.4.22	Recommended Meals in Meal Logging Page	52
Figure 5.4.23	Favourite Meals in Meal Logging Page	52
Figure 5.4.24	Custom Meals in Meal Logging Page	52
Figure 5.4.25	Serving Size Input in Meal Logging Page	53
Figure 5.4.26	Exceeds Limit in Meal Logging Page	53
Figure 5.4.27	Nutrient Breakdown in Meal Logging Page	53
Figure 5.4.28	Quick Meal Tab in Custom Meal Page	53
Figure 5.4.29	Quick Meal Tab in Custom Meal Page	53
Figure 5.4.30	Nutrition Label Extraction in Custom Meal Page	53
Figure 5.4.31	“Full Recipe” Tab in Custom Meal Page	54
Figure 5.4.32	Ingredient Selection Page	54
Figure 5.4.33	Ingredient Weight Input in Ingredient Selection Page	54
Figure 5.4.34	“My List” Tab in Custom Meal Page	55
Figure 5.4.35	Delete Confirmation in Custom Meal Page	55
Figure 5.4.36	Nutrition Trends Page	55
Figure 5.4.37	AI Nutrition Assistant Page	56
Figure 5.4.38	List of Forum Posts in Community Forum Page	57
Figure 5.4.39	“Load more posts” Button in Community Forum Page	57
Figure 5.4.40	No More Posts in Community Forum Page	57
Figure 5.4.41	Post Details Page	58
Figure 5.4.42	Report Post or Comment in Post Details Page	58
Figure 5.4.43	Edit Post or Comment in Post Details Page	58
Figure 5.4.44	New Post Page	59

Figure 5.4.45	Empty Fields in New Post Page	59
Figure 5.4.46	Reminders Page	60
Figure 5.4.47	New Reminder in Reminders Page	60
Figure 5.4.48	Time Picker in Reminders Page	60
Figure 5.4.49	Upper View of Profile Page	61
Figure 5.4.50	Lower View of Profile Page	61
Figure 5.4.51	Upper View of Input and Update Profile Page	62
Figure 5.4.52	Lower View of Input and Update Profile Page	62
Figure 5.4.53	Empty Field and Incorrect Data Format in Input and Update Profile Page	62
Figure 5.4.54	Admin Home Page	63
Figure 5.4.55	Active User List in User Management Page	64
Figure 5.4.56	Banned User List in User Management Page	64
Figure 5.4.57	Sorting Options in User Management Page	64
Figure 5.4.58	Active User Details in User Management Page	64
Figure 5.4.59	Banned User Details in User Management Page	64
Figure 5.4.60	Banning Timeline in User Management Page	64
Figure 5.4.61	Admin List in Admin Management Page	65
Figure 5.4.62	Sorting Options in Admin Management Page	65
Figure 5.4.63	Admin Registration Details in Admin Management Page	65
Figure 5.4.64	Meal Management Page	66
Figure 5.4.65	Meal Details in Meal Management Page	66
Figure 5.4.66	Edit Meal in Meal Management Page	66
Figure 5.4.67	Upper View of Add Meal Window in Meal Management Page	67
Figure 5.4.68	Lower View of Add Meal Window in Meal Management Page	67
Figure 5.4.69	Empty Fields and Incorrect Data Format in Meal Management Page	67
Figure 5.4.70	Reported Posts and Comments Management Page	68
Figure 5.4.71	Report History Page	68
Figure 5.5.1	Best Model of Random Forest Regressor Model	68
Figure 5.5.2	Random Forest Regressor Results	69

Figure 5.5.3	Multilayer Perceptron Regressor Results	69
Figure 5.5.4	Linear Regression Results	69
Figure 6.2.1.1	Nutrients Recommendation for Obesity	76
Figure 6.2.1.2	Nutrients Recommendation for Hypertension	77
Figure 6.2.1.3	Nutrients Recommendation for Diabetes	77
Figure 6.2.1.4	Nutrients Recommendation for Health Disease	77
Figure A.2.1	Activity Diagram for Login Account	A-24
Figure A.2.2	Activity Diagram for Input Profile	A-24
Figure A.2.3	Activity Diagram for View Profile	A-25
Figure A.2.4	Activity Diagram for Update Profile	A-25
Figure A.2.5	Activity Diagram for View Daily Nutrient Intake Recommendations	A-26
Figure A.2.6	Activity Diagram for View Diary	A-27
Figure A.2.7	Activity Diagram for View Logged Meals	A-28
Figure A.2.8	Activity Diagram for View Meal Recommendations	A-28
Figure A.2.9	Activity Diagram for Log Meals	A-29
Figure A.2.10	Activity Diagram for Add Custom Meals	A-30
Figure A.2.11	Activity Diagram for Track Nutrient Progress	A-31
Figure A.2.12	Activity Diagram for Get Personalised Nutrition Advice	A-31
Figure A.2.13	Activity Diagram for View Community Forum	A-32
Figure A.2.14	Activity Diagram for Receive Reminders	A-33
Figure A.2.15	Activity Diagram for Manage User Accounts	A-34
Figure A.2.16	Activity Diagram for Manage Admin Registration	A-35
Figure A.2.17	Activity Diagram for Manage Meals	A-36
Figure A.2.18	Activity Diagram for Manage Reported Posts and Comments	A-37

LIST OF TABLES

Table Number	Title	Page
Table 2.11.1	Comparison of the Previous Works with Proposed Application	21
Table 3.2.3.1	Use Case Description for Register Account	27
Table 3.3.1.1	Attributes Used for Model Training	30
Table 4.3.1	Database Design for the diary Collection	39
Table 5.1.1	Specifications of Laptop	40
Table 5.1.2	Specifications of Smartphone	40
Table 6.2.1.1	Model Result Testing Based on Health Condition Requirements	76
Table 6.2.2.1	Functional Test Cases for the Login Module	78
Table A.1.1	Use Case Description for Login Account	A-1
Table A.1.2	Use Case Description for Input Profile	A-2
Table A.1.3	Use Case Description for View Profile	A-3
Table A.1.4	Use Case Description for Update Profile	A-4
Table A.1.5	Use Case Description for View Daily Nutrient Intake Recommendations	A-5
Table A.1.6	Use Case Description for View Diary	A-6
Table A.1.7	Use Case Description for View Logged Meals	A-8
Table A.1.8	Use Case Description for View Meal Recommendations	A-9
Table A.1.9	Use Case Description for Log Meals	A-10
Table A.1.10	Use Case Description for Add Custom Meals	A-12
Table A.1.11	Use Case Description for Track Nutrient Progress	A-14
Table A.1.12	Use Case Description for Get Personalised Nutrition Advice	A-15
Table A.1.13	Use Case Description for View Community Forum	A-16
Table A.1.14	Use Case Description for Receive Reminders	A-18
Table A.1.15	Use Case Description for Manage User Accounts	A-19
Table A.1.16	Use Case Description for Manage Admin Registration	A-20
Table A.1.17	Use Case Description for Manage Meals	A-21

Table A.1.18	Use Case Description for Manage Reported Posts and Comments	A-23
Table A.3.1	Database Design for the users Collection	A-38
Table A.3.2	Database Design for the chats Collection	A-38
Table A.3.3	Database Design for the chat Subcollection	A-39
Table A.3.4	Database Design for the mealLogs Collection	A-39
Table A.3.5	Database Design for the customMeal Collection	A-40
Table A.3.6	Database Design for the meals Subcollection	A-40
Table A.3.7	Database Design for the meals Collection	A-41
Table A.3.8	Database Design for the posts Collection	A-42
Table A.3.9	Database Design for the comments Subcollection	A-44
Table A.3.10	Database Design for the likes Subcollection	A-44
Table A.3.11	Database Design for the reminders Collection	A-45
Table A.3.12	Database Design for the recommendations Collection	A-45
Table A.3.13	Database Design for the dates Subcollection	A-46
Table A.3.14	Database Design for the reports Collection	A-47
Table A.3.15	Database Design for the savedPosts Collection	A-48
Table A.3.16	Database Design for the posts Subcollection	A-48
Table A.3.17	Database Design for the usersInfo Collection	A-49
Table A.3.18	Database Design for the viewedPosts Collection	A-49
Table A.3.19	Database Design for the posts Subcollection	A-50
Table A.3.20	Database Design for the waterLogs Collection	A-50
Table A.4.1	Functional Test Cases for the Registration Module	A-51
Table A.4.2	Functional Test Cases for the Password Recovery Module	A-52
Table A.4.3	Functional Test Cases for the User Home Module	A-53
Table A.4.4	Functional Test Cases for the Diary Module	A-53
Table A.4.5	Functional Test Cases for the Nutrition Trends Module	A-56
Table A.4.6	Functional Test Cases for the Meal Logging Module	A-56
Table A.4.7	Functional Test Cases for the Custom Meal Module	A-58
Table A.4.8	Functional Test Cases for the Ingredient Selecting Module	A-61
Table A.4.9	Functional Test Cases for the Meal Details Module	A-62
Table A.4.10	Functional Test Cases for the AI Nutrition Assistant Module	A-62

Table A.4.11	Functional Test Cases for the Community Forum Module	A-63
Table A.4.12	Functional Test Cases for the Post Details Module	A-64
Table A.4.13	Functional Test Cases for the Post Creation Module	A-66
Table A.4.14	Functional Test Cases for the Reminders Module	A-67
Table A.4.15	Functional Test Cases for the Profile Module	A-68
Table A.4.16	Functional Test Cases for the Input and Update Profile Module	A-68
Table A.4.17	Functional Test Cases for the Admin Home Module	A-69
Table A.4.18	Functional Test Cases for the User Management Module	A-69
Table A.4.19	Functional Test Cases for the Admin Management Module	A-70
Table A.4.20	Functional Test Cases for the Meal Management Module	A-71
Table A.4.21	Functional Test Cases for the Reported Posts and Comments Management Module	A-73
Table A.4.22	Functional Test Cases for the Report History Module	A-74

LIST OF SYMBOLS

%	Percentage
:	Colon
+	Add
>	Greater than
<	Less than
@	At

LIST OF ABBREVIATIONS

<i>API</i>	Application Programming Interface
<i>AI</i>	Artificial Intelligence
<i>MLP</i>	Multilayer Perceptron
<i>HTTP</i>	Hypertext Transfer Protocol
<i>JSON</i>	JavaScript Object Notation
<i>BMI</i>	Body Mass Index
<i>ReLU</i>	Rectified Linear Unit
<i>MSE</i>	Mean Squared Error
<i>MyFCD</i>	Malaysian Food Composition Database
<i>NoSQL</i>	Not Only SQL
<i>IDE</i>	Integrated Development Environment
<i>GPU</i>	Graphics Processing Unit
<i>SDK</i>	Software Development Kit
<i>ml</i>	Milliliter
<i>g</i>	Gram
<i>AM</i>	Ante Meridiem
<i>PM</i>	Post Meridiem
<i>R²</i>	R-squared
<i>MAE</i>	Mean Absolute Error
<i>RMSE</i>	Root Mean Squared Error
<i>UI</i>	User Interface
<i>LD</i>	Low-Density Lipoprotein
<i>UTC</i>	Coordinated Universal Time
<i>Oct</i>	October

Chapter 1

Introduction

The problem statement, motivation, project objectives, project scope, contributions, and report organisation of this project are presented in this chapter.

In the fast-paced world of today, many individuals often find themselves rushing through their daily routines and neglecting the essential act of having regular meals [1]. Meal skipping has become a common practice for various reasons, including time constraints, weight management goals, or simply forgetting to eat.

According to [2], the skipping rate for breakfast, lunch, and dinner are 14-88%, 8-57%, and 4-57% respectively. This proves that there are more individuals who skip their breakfast compared to lunch and dinner. Regular omission of breakfast has been associated with a postprandial rise in glucose levels after lunch, which is linked to elevated pre-lunch plasma-free fatty acids. Furthermore, habitual breakfast skipping may increase the risk of obesity, diabetes, and cardiovascular disease [3]. Hence, it is crucial to understand that regularly skipping meals can have detrimental effects on our physical and mental well-being.

Meals are important as meals provide energy for daily routines [1]. This energy is derived from nutrients such as carbohydrates, proteins, and fats, which are metabolised by the body. Skipping meals can result in low blood sugar levels, leading to fatigue and dizziness. Nutritional values are different for every meal, so it is hard to obtain all the nutrients a person need from a single meal [4]. Therefore, the logging of meals and tracking of nutrient intake are essential to evaluate whether daily nutritional requirements are met.

1.1 Problem Statement and Motivation

Nutrients are the components present in any dish consumed, and the amount of each nutrient varies depending on the meal. Certain nutrients are essential for the human body to maintain healthy organ function and overall well-being [5]. However, the nutritional value of meal cannot be judged solely by its appearance or taste. For many people, especially those with chronic diseases such as diabetes or high blood pressure, the meals to be avoided is hard to

identify as personalised dietary information is not available [6]. This problem is getting worse with age, as natural declines in organ function increase the risk of nutrient imbalances [7].

Although unusual nutrient levels found in blood tests are usually managed by hospital through medication, dietary changes are considered an important non-medical approach to improve health [8]. However, tools for tracking and monitoring nutrient intake are not widely available. The function of giving personalised meal recommendations based on both health conditions and nutrient restrictions is not available in the existing mobile applications for nutrient tracking and personalised meal recommendations [9].

Therefore, the motivation for this project is to develop a mobile application which the nutrient intake would be tracked and personalised meal recommendations would be provided. The current gaps in dietary monitoring would be addressed, and healthier eating habits would be promoted while improving nutrient balance based on individual health conditions.

1.2 Objectives

The project objectives are stated as follows:

- i. To develop a mobile application that can record daily meals of the users and track their nutrient intake.**

This project will implement user registration and authentication to ensure a secure and personalised experience which allows each user to create and verify their account. Once logged in, users can record meal details by searching for available meals or by adding custom entries. The proposed application will then analyse these records to generate a breakdown of key nutritional values, including calories, macronutrients, and micronutrients. The proposed application will include the nutrient progress tracking function, enabling users to monitor their dietary habits and measure their progress toward health goals over time. Moreover, the proposed application will include additional functions such as custom meal creation, personalised nutritional advice using Google AI, community forum, and reminders for the users. On the other hand, the proposed application will include the management of user accounts, admin registration, meals, and reported posts and comments for the admins.

ii. To integrate a model to recommend suitable meals to users based on their health conditions.

The meal recommendation function will consider health conditions such as hypertension, diabetes, high cholesterol, or obesity to ensure that recommendations are both safe and beneficial. By analysing the health condition and nutrient intake history of the users, the model was trained for generating personalised meal options designed to encourage healthier eating habits. This function will help users make informed dietary decisions that align with their nutritional requirements and health conditions.

1.3 Project Scope

The scope of this project is focused on developing a mobile application to ensure users able to log their meals and track their nutrient and calorie intake depending on their health conditions and diet requirements. The user can also get personalised meal recommendations and alert when their consumption is beyond healthy limits.

The major functions will be:

- Manual meal logging with nutritional breakdown
- Weekly nutrition summaries
- Custom meal creation
- Alerts when nutrient intake is excessive
- Personalised meal recommendations based on individual health data

The project scope is limited to functions that exist in free versions of available nutrition-tracking applications, and no premium Application Programming Interface (API) or paid subscriptions will be required. Barcode scanning, step or exercise sync, and meals photo logging functions will not be considered in this project. Moreover, the health conditions are limited to five categories which are hypertension, diabetes, obesity, heart disease, and no chronic disease. For nutrition label extraction, the nutrition label is limited to products made from Malaysia only. Additionally, the nutritional details of the meals are provided as a reference, since the exact amount of nutrients is difficult to determine.

1.4 Contributions

The main contribution of this project is addressing the limitations of existing nutrition applications by offering a more personalised and health-oriented approach. While many applications provide calorie counting and nutrient tracking, these applications often lack the ability to adapt recommendations based on individual health conditions. This project contributes by integrating health conditions such as diabetes into the project, allowing the personalised meal recommendations to be tailored accordingly.

Another significant contribution is the integration of nutrition label extraction into the nutritional values input process. This function enables the application to extract the text on the nutrition label and updates the fields with the extracted values. This function reduces the nutritional values input time and decreases the possibility of entering incorrect values.

In addition, the project provides a proactive monitoring application through alerts and reminders. Users will be notified when their daily nutrient intake is insufficient or if a particular nutrient is being consumed in excess. This ensures users receive timely guidance rather than only passive tracking.

Overall, the project moves beyond simple calorie counting by delivering personalised meal recommendations grounded in real health data. By combining health conditions and continuous monitoring, the project contributes a more comprehensive, user-focused solution compared to existing nutrition application.

1.5 Report Organisation

The details of this project are presented across several chapters. In Chapter 1, the project background, problem statement, objectives, scope, and contributions are introduced. In Chapter 2, the existing applications are reviewed and their limitations are discussed. In Chapter 3, the methodology, system design, and development approach used in the project are explained. In Chapter 4, the system design including architecture and component details is presented. In Chapter 5, the system implementation, setup, and operational process are described. In Chapter 6, the system performance and testing results are evaluated while the challenges faced are discussed. Finally, the project is concluded and the recommendations for future improvements are provided in Chapter 7.

Chapter 2

Literature Review

2.1 Overview

A significant increase has been observed recently in public awareness regarding health and nutrition, resulting in a growing demand for tools designed to monitor and improve dietary habits [10]. The importance of accurate nutrient intake tracking is recognised for its role in supporting health goals, managing weight, and addressing health conditions influenced by diet. Mobile applications have been developed to enable the logging of meal intake, analysis of nutrient composition, and setting of dietary targets to meet these needs. Healthier lifestyles are promoted through such applications, as quick access to nutritional data is provided and informed dietary choices are facilitated. Several existing applications have been identified as notable examples of established nutrition tracking applications created to meet these demands.

2.2 MyFitnessPal

MyFitnessPal [11] is a meal logging and nutrition tracking mobile application which is used to record and monitor daily nutrients intake of the users. Figure 2.2.1 shows the main interface of this application which allows users to register or log into the application while the login credentials are stored in the database. This application uses a comprehensive meal database which enables the users to log their meal by searching, scanning barcodes, or adding custom meals as shown in Figure 2.2.2, Figure 2.2.3, and Figure 2.2.4. Nutrient breakdown for each meal item, such as calories, macronutrients, and micronutrients, are listed for users to check their dietary patterns.

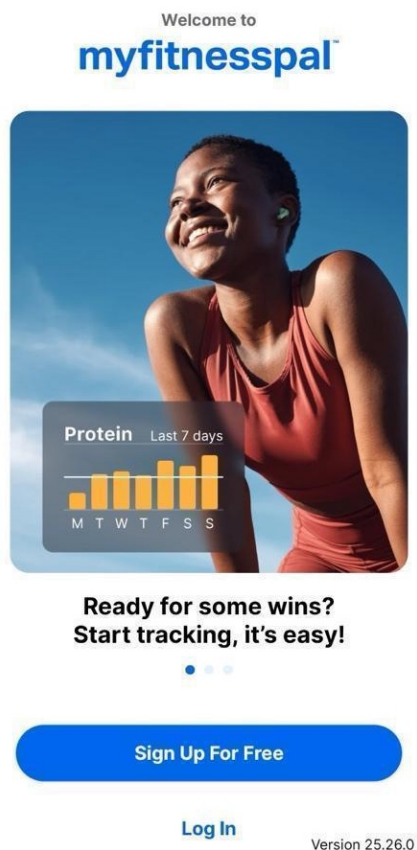


Figure 2.2.1 Main Interface in MyFitnessPal

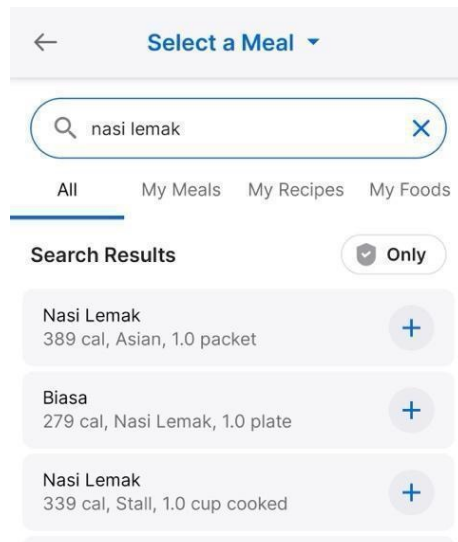


Figure 2.2.2 Meal Searching in MyFitnessPal

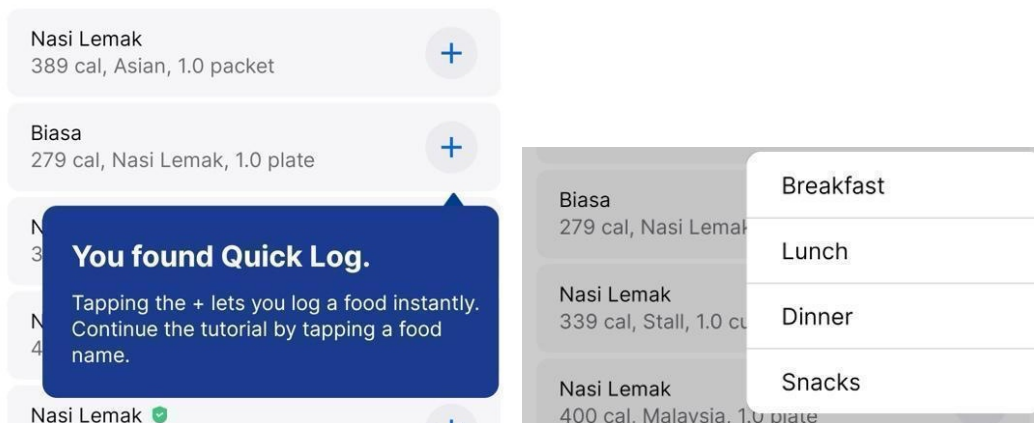


Figure 2.2.3 Meal Logging in MyFitnessPal

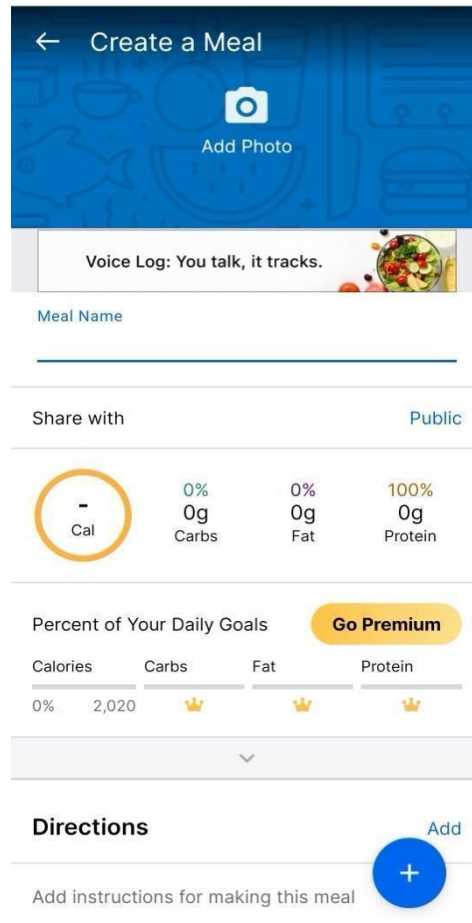


Figure 2.2.4 Custom Meal in MyFitnessPal

Furthermore, progress tracking is facilitated through detailed charts and summaries as shown in Figure 2.2.5. Exercise tracking is implemented to allow users to input their exercising details and track their calories burned as shown in Figure 2.2.6. A community forum is included, enabling the sharing of tips, recipes, and motivational support among users. Integration with wearable fitness devices is supported, ensuring synchronisation of health and activity data.

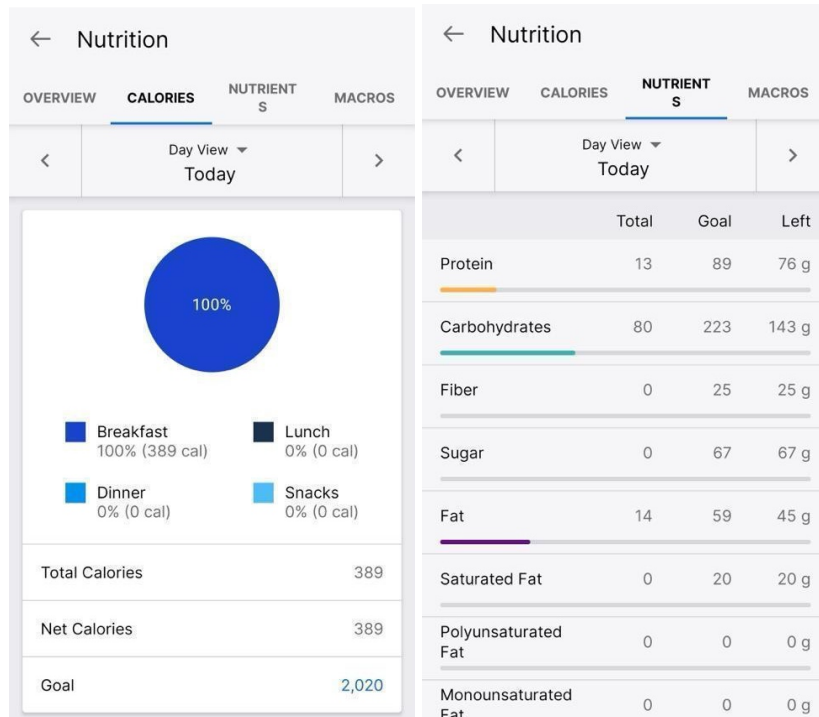


Figure 2.2.5 Progress Tracking in MyFitnessPal

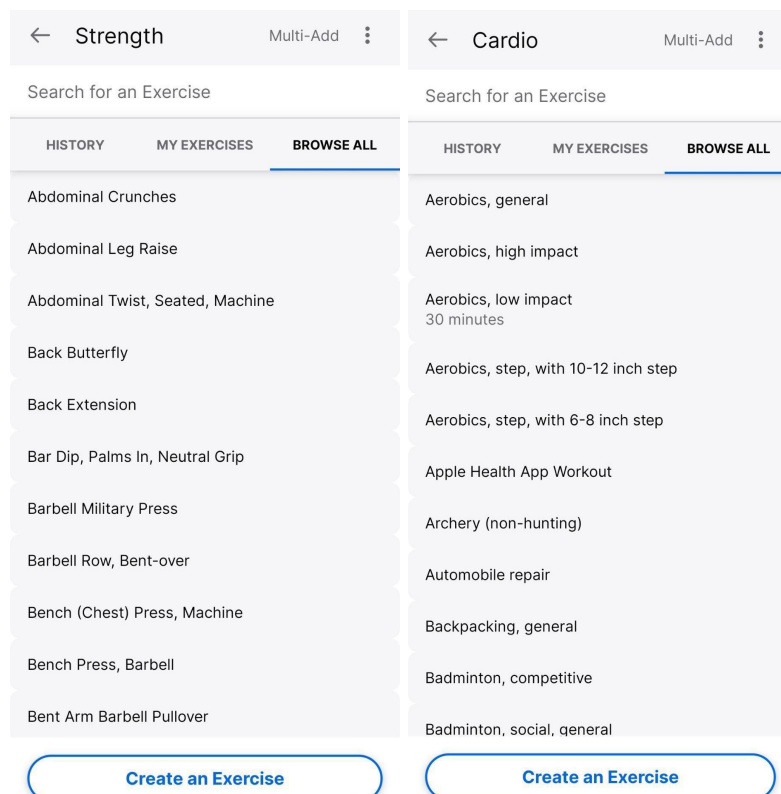


Figure 2.2.6 Exercise Tracking in MyFitnessPal

2.3 YAZIO

YAZIO [12] is a mobile application developed for meal logging and nutrition tracking. This application aims to help users get a healthier life by losing weight, building muscle, and eating healthier. The main interface of YAZIO is made for users to register their accounts and the information of their account will be stored in the database as shown in Figure 2.3.1. Figure 2.3.2 and Figure 2.3.3 shows the meals logging and the meals searching function which will retrieve the meal information from the meal database.



Figure 2.3.1 Main Interface in YAZIO

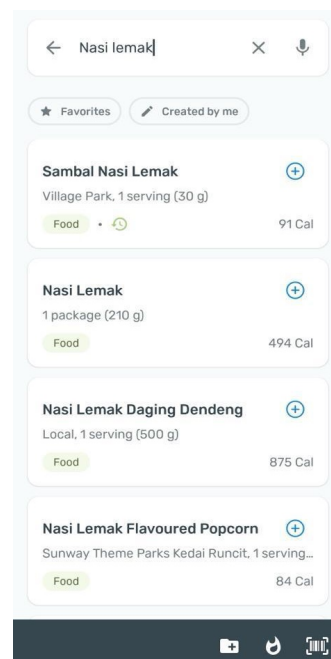


Figure 2.3.2 Meal Searching in YAZIO

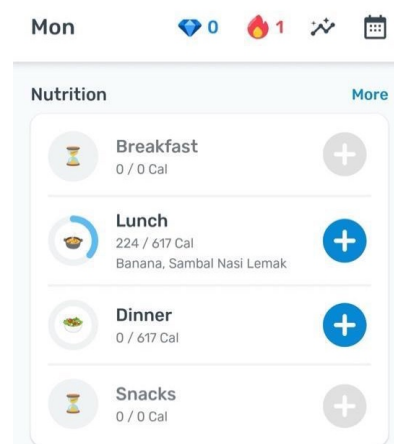


Figure 2.3.3 Meal Logging in YAZIO

The functions, such as calorie consumption tracking, and nutrient intake monitoring, is provided in this application to help users reach the goal as shown in Figure 2.3.4. Daily nutritional intakes are calculated based on the meal consumed and presented using graphs and charts. This program facilitates the tracking of intermittent fasting schedules, enabling the monitoring of fasting durations and progress as shown in Figure 2.3.5. Moreover, nutritional reports can also be produced to discern patterns and indicate areas for enhancement. Integration with additional fitness applications and devices is facilitated for thorough lifestyle monitoring.

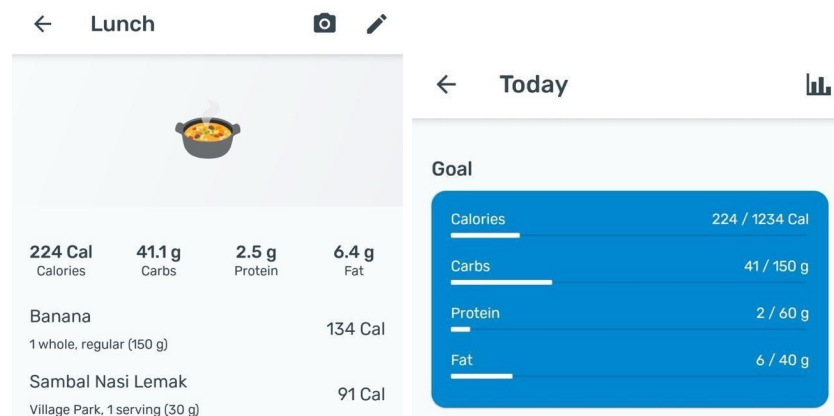


Figure 2.3.4 Calorie Consumption and Nutrient Intake Tracking in YAZIO

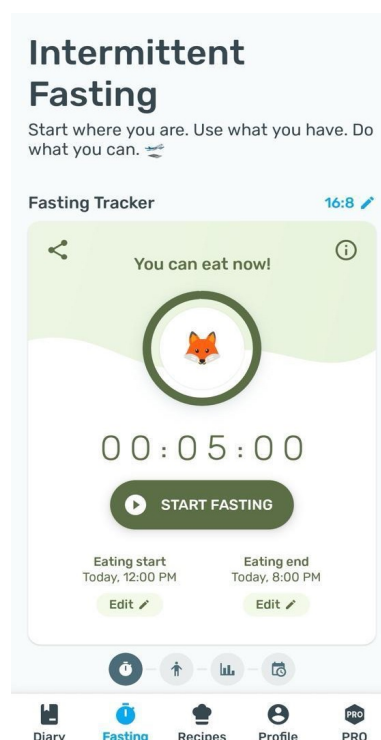


Figure 2.3.5 Intermittent Fasting Page in YAZIO

2.4 HealthifyMe

HealthifyMe [13] is a health and nutrition application designed to assist in managing diet, fitness, and weight goals. Figure 2.4.1 shows the main interface of HealthifyMe. A database containing Indian and international food is provided, enabling meals to be logged through manual search or barcode scanning as shown in Figure 2.4.2 and Figure 2.4.3. Personalised diet plans are created by Artificial Intelligence (AI) or professional nutritionists, based on user health conditions, preferences, and goals as shown in Figure 2.4.4.

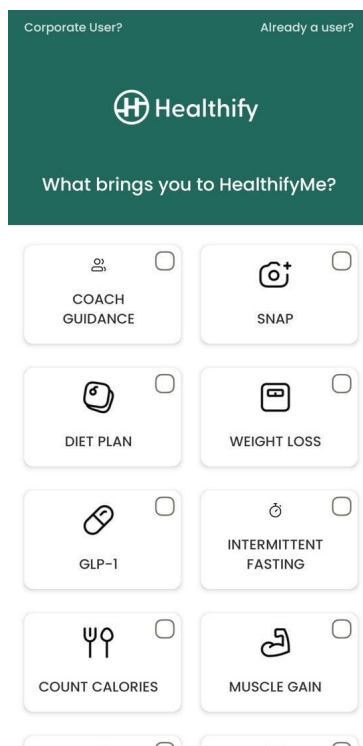


Figure 2.4.1 Main Interface in HealthifyMe

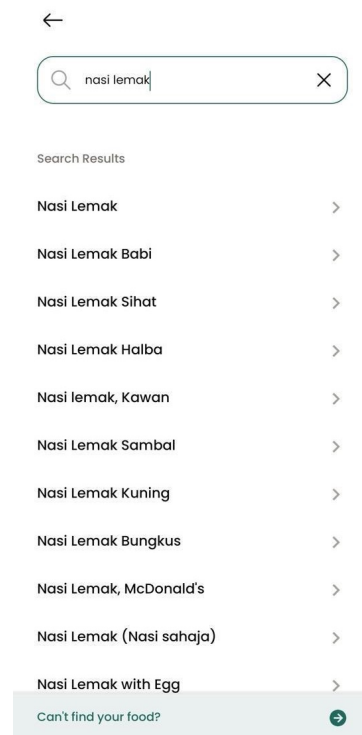


Figure 2.4.2 Meal Searching in HealthifyMe

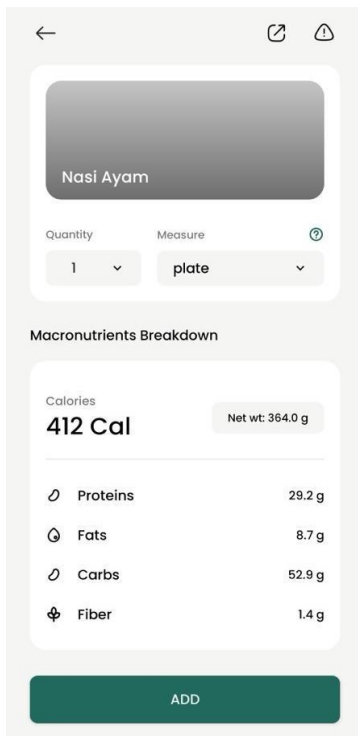


Figure 2.4.3 Meal Logging in HealthifyMe



Figure 2.4.4 AI Coach Ria in HealthifyMe

Other than that, exercise routines, step counts, and hydration levels can be tracked as shown in Figure 2.4.5. Additionally, a calorie tracker is provided to assess energy consumption and expenditure, offering real-time feedback on nutrient equilibrium as shown in Figure 2.4.6. Figure 2.4.7 shows detailed reports are generated to assess progress and adherence to dietary guidelines over time. The integration of wearable gadgets and fitness monitors enables seamless data collection. A coaching component is accessible, in which certified health professionals provide guidance and encouragement.

Your Trackers

-  Calorie Tracker
-  Water Tracker
-  Steps Tracker
-  Workout Tracker
-  Weight Tracker

Figure 2.4.5 Other Trackers in HealthifyMe

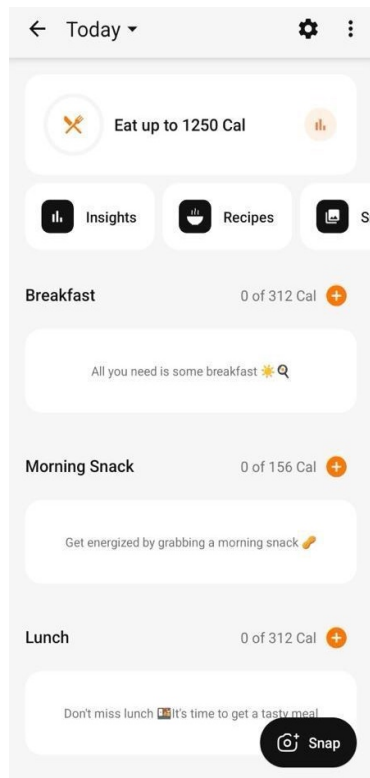


Figure 2.4.6 Calorie Tracking in HealthifyMe

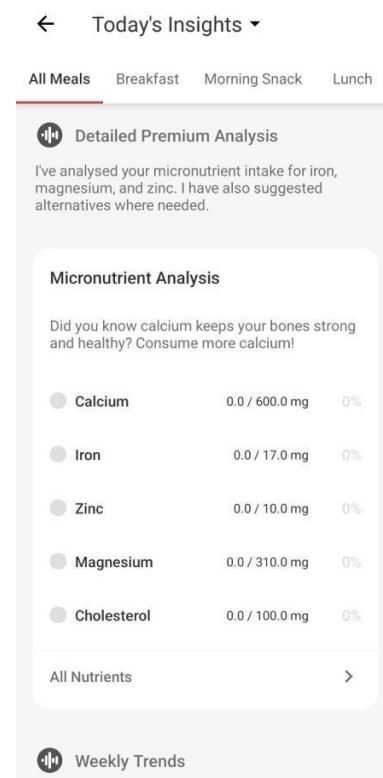


Figure 2.4.7 Progress Report in HealthifyMe

2.5 Eatr

Eatr [14] is a nutrition tracking and meal planning application designed for calorie monitoring and macronutrient management. Figure 2.5.1 shows the main interface of the application. The application supports meal logging, calorie estimation, and tracking of carbohydrate, protein, and fat intake. Users may search for their meal using keywords as shown in Figure 2.5.2 and log their meal for a specific meal period as shown in Figure 2.5.3. Figure 2.5.4 shows the daily nutrient tracking which displays the total nutrient of the meal logged. Users may create their own custom meal if the meal is not available in the meal database as shown in Figure 2.5.5. The meal planning function provides personalised meal and recipe recommendation based on dietary preferences and dietary goals as shown in Figure 2.5.6.



Figure 2.5.1 Main Interface in Eatr

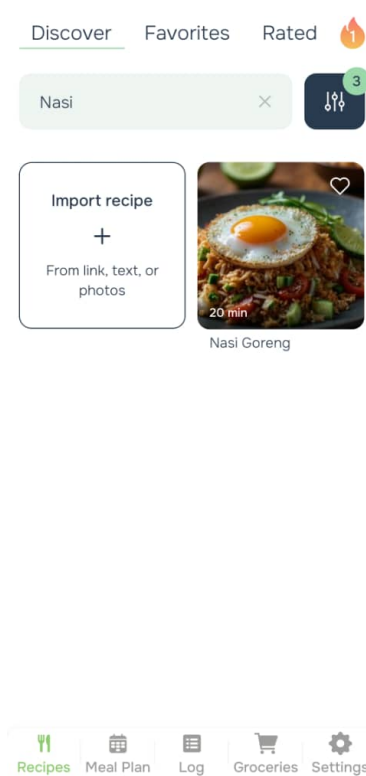


Figure 2.5.2 Meal Searching in Eatr

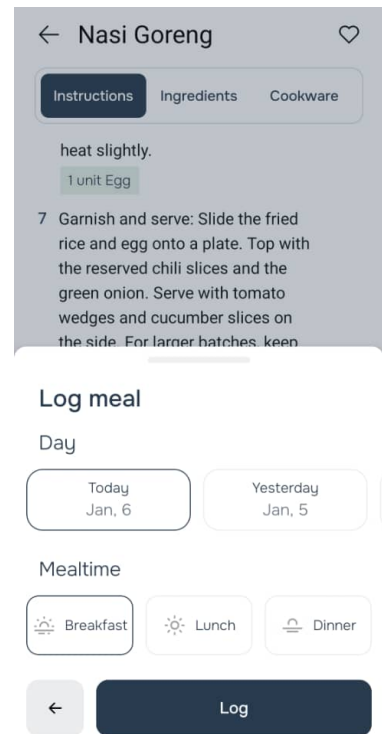


Figure 2.5.3 Meal Logging in Eatr

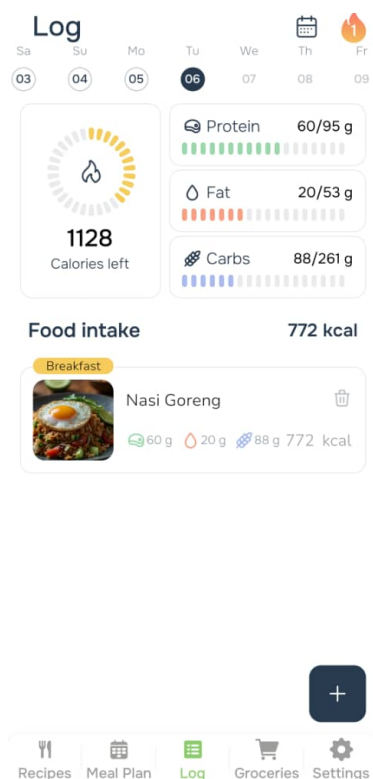


Figure 2.5.4 Nutrient Tracking in Eatr

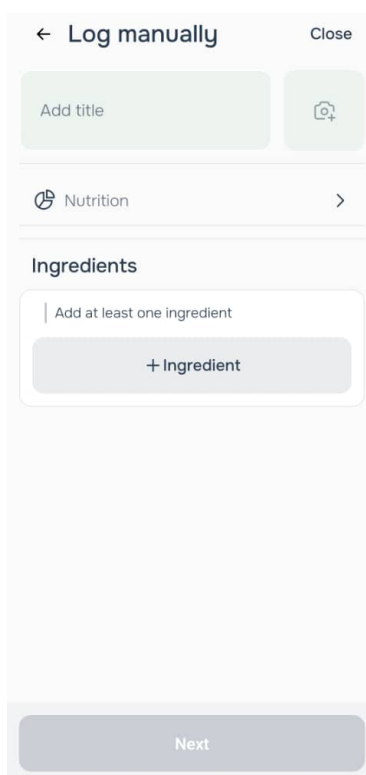


Figure 2.5.5 Custom Meal in Eatr

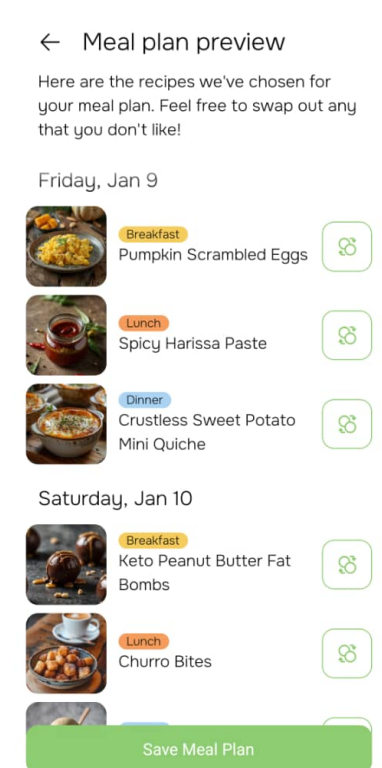


Figure 2.5.6 Meal Planning in Eatr

2.6 Foodvisor

Foodvisor [15] is a meal logging and nutrient tracking application focused on caloric intake tracking and macronutrient evaluation. Figure 2.6.1 shows the main interface of the application. The application supports meal logging through photo capture, barcode scanning, and manual logging, and applies computer vision model to identify foods and estimate calories, carbohydrates, proteins, and fats from recognised items as shown in Figure 2.6.2. The daily total nutrient intake is displayed in the journal page as shown in Figure 2.6.3. The application also provides the function of creating custom meal if the meal is not available as shown in Figure 2.6.4. Additionally, the application allows users to track their water intake as displayed in Figure 2.6.5.



Figure 2.6.1 Main Interface in Foodvisor

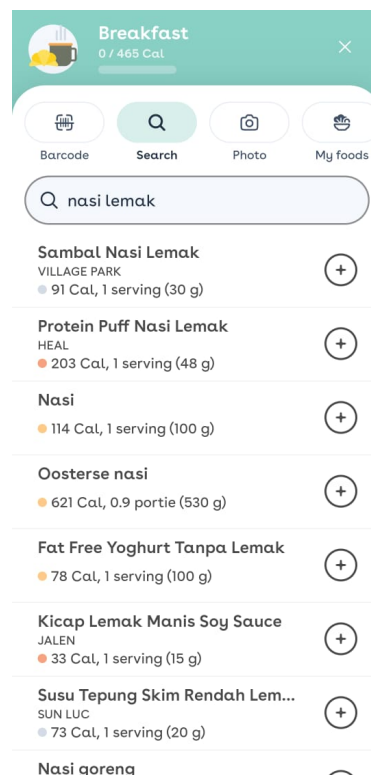


Figure 2.6.2 Meal Searching and Logging in Foodvisor

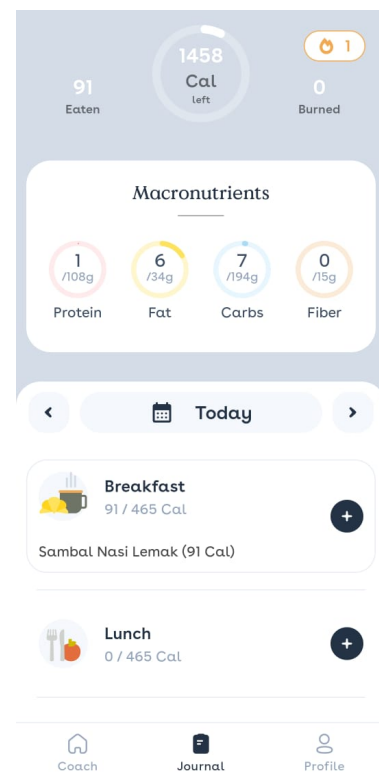


Figure 2.6.3 Nutrient Tracking in Foodvisor

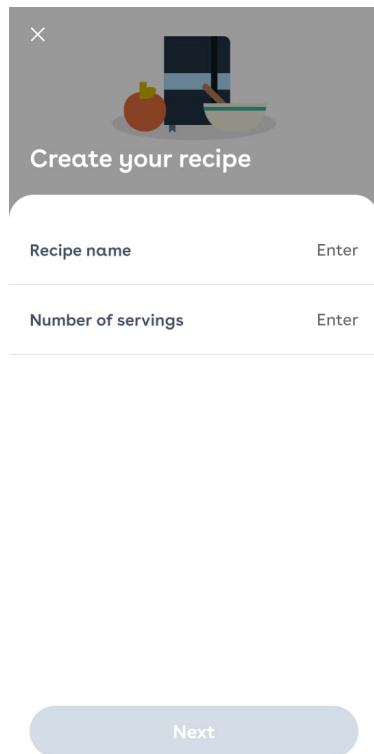


Figure 2.6.4 Custom Meal in Foodvisor

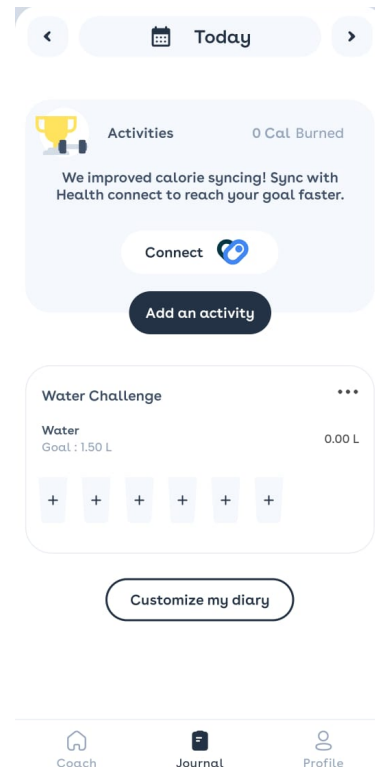


Figure 2.6.5 Water Tracking in Foodvisor

2.7 Lifesum

Lifesum [16] is a nutrient tracking and diet management application. Figure 2.7.1 shows the main interface of the application. The application supports meal logging, calorie counting, and macronutrient tracking for daily food intake as shown in Figure 2.7.2 and Figure 2.7.3. Other than that, the application also tracks water intake, veggie, and fruit count as shown in Figure 2.7.4. The application provides meal plans and guided diet programs to support weight control and healthy eating habits. It also includes food-database search, barcode scanning, and integration with external health and activity platforms to synchronize user data and monitoring.

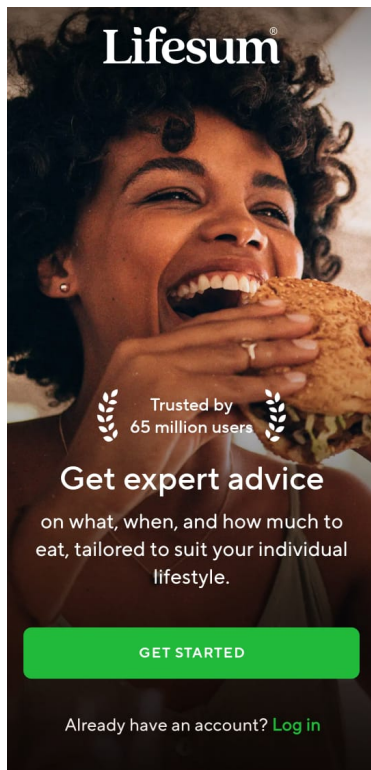


Figure 2.7.1 Main Interface in Lifesum

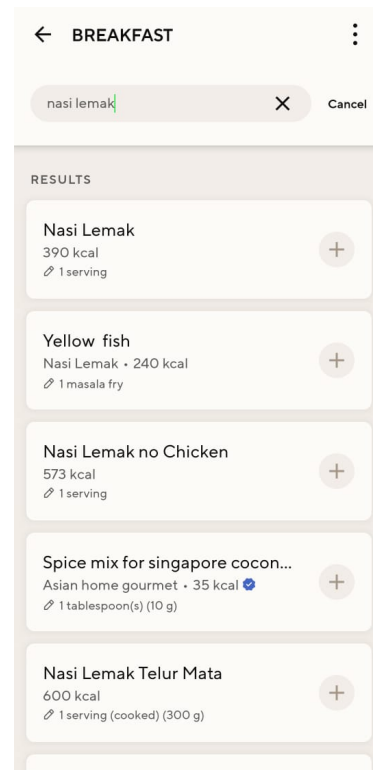


Figure 2.7.2 Meal Searching and Logging in Lifesum

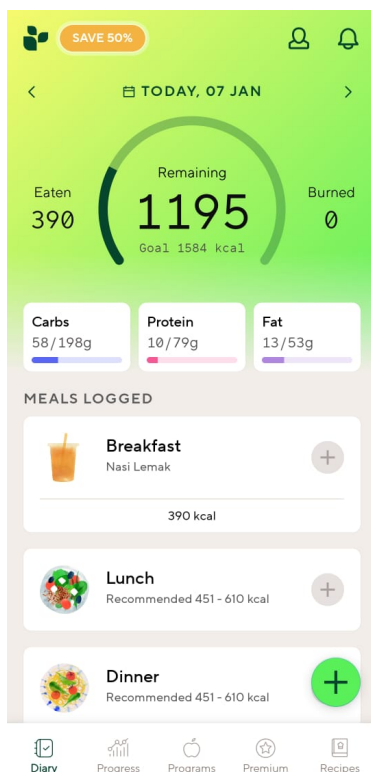


Figure 2.7.3 Nutrient Tracking in Lifesum

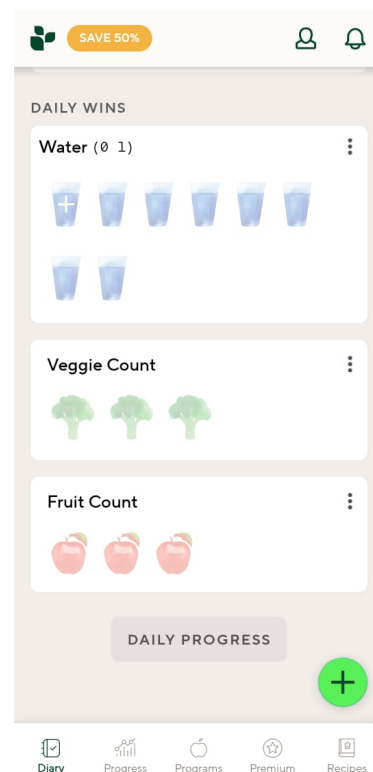


Figure 2.7.4 Water, Veggie, and Fruit Count Tracking in Lifesum

2.8 Comparison of the Previous Works

MyFitnessPal, YAZIO, HealthifyMe, Eatr, Foodvisor, and Lifesum were developed to help users track meals and make healthier choices. All existing applications provide food databases, nutrient analysis tools, and goal-setting functions to support long-term diet improvement. MyFitnessPal and HealthifyMe integrate with wearable devices, allowing activity and health data synchronization, while YAZIO focuses on structured diet programs and fasting trackers. Eatr provides AI-driven recipe recommendation and nutrient calculations, Foodvisor uses photo-based food recognition for automatic nutrient estimation, and Lifesum offers goal-oriented meal plans with progress tracking and health-platform integration.

MyFitnessPal is recognised for its extensive food database and barcode scanner, and HealthifyMe stands out for access to certified nutritionists and personalised coaching, with strong coverage of South Asian cuisines. YAZIO is suitable for users following specific diet plans or fasting schedules.

2.9 Limitation of Previous Works

Several limitations can still be identified in the existing applications that were reviewed. Although MyFitnessPal, YAZIO, HealthifyMe, Eatr, Foodvisor, and Lifesum provide functions for meal logging, calorie tracking, and basic nutrient monitoring, these applications do not yet include several important functions that would be useful for users with specific health conditions.

One major limitation is the absence of a function for inputting health information to refine meal recommendations. Consequently, most recommendations remain general and are not based on health conditions, making them less beneficial for individuals who require meal recommendations based on their health conditions.

Another limitation is the lack of integration with nutrition label extraction to support nutritional values input process. This function may reduce the time for entering the nutritional values and avoid entering wrong values. Without this function, the users need more time to enter all required fields.

Although these applications are suitable for tracking calories and macronutrients, they typically do not provide deeper nutrient analysis or risk alerts based on actual health information. For users who aim to prevent getting the chronic diseases or manage health conditions, this gap limits the effectiveness of the existing applications.

2.10 Recommendation Models

This section discusses the models that are used to provide daily nutrient intake recommendations. Three models which are commonly used for recommendation were reviewed.

The first model is Random Forest Regressor which operates by constructing multiple decision trees during the training process [17]. Each tree is trained on randomly selected subsets of the dataset and features, and the final prediction is produced by averaging the outputs of all trees. This approach reduces overfitting and improves prediction stability, as the combined output of many trees tends to be more accurate and robust than that of a single decision tree [17]. The model can capture complex, non-linear relationships within the data, making it suitable for predicting nutritional recommendations based on multiple features [17].

Previous studies have shown the effectiveness of the Random Forest Regressor in various recommendation tasks. [18] used Random Forest Regressor for crop recommendation, and the model achieved a highest accuracy of 0.99 which performed better than other models such as Decision Tree, Naive Bayes, Support Vector Machine, and Logistic Regression. [19] also used Random Forest Regressor for diet and physical exercise recommendation system. The model predicted the diet and physical exercise for each mealtime category and for vegetarians or non-vegetarians. The highest accuracy for diet recommendations was for non-vegetarians at lunchtime, at 99.03%, while the lowest accuracy was for vegetarians in the morning, at 70.74%. For physical exercise recommendations, the accuracy was stable at 73.65%, indicating relatively stable accuracy for physical exercise recommendations.

The second model is Multilayer Perceptron (MLP) Regressor which is a type of artificial neural network used for regression tasks [20]. It consists of an input layer, one or more hidden layers, and an output layer, with each layer made up of interconnected neurons. During training, the model learns patterns by adjusting its internal weights using the backpropagation algorithm. The MLP can model complex non-linear relationships between inputs and outputs,

making it useful for predicting recommended nutrient intake based on patient characteristics [20].

Several previous studies also demonstrated the effectiveness of the MLP Regressor. [18] also performed research for fertiliser recommendation using Decision Tree, XGBoost, Support Vector Machine, Logistic Regression, Random Forest Regressor, and MLP Regressor. The accuracy achieved by the MLP Regressor was slightly higher than the Random Forest Regressor. [21] developed a movie recommendation system using MLP Regressor. This research compared the MLP Regressor model with Logistic Regression, Random Forest, and Gradient Boosting Decision Tree models and the MLP Regressor model achieved the best performance among them with the lowest mean absolute error of 0.52 and lowest root mean squared error of 0.67.

The third model is Linear Regression which is used to model and analyse the relationship between a dependent variable and one or more independent variables by fitting a straight line to the observed data [22]. It works by finding the line of best fit that minimises the difference between the actual data points and the predicted values, allowing outcomes estimation and prediction based on input variables [22]. It assumes that the relationship between the variables is linear, meaning that changes in the independent variable lead to proportional changes in the dependent variable [22].

Previous studies have demonstrated that Linear Regression performs effectively when predicting numerical data. In [23], a crop recommendation model was developed to assist farmers in selecting suitable crops by considering factors such as soil and geographical location. The results showed that Linear Regression achieved one of the highest accuracies at 88.26%, compared to Neural Network (89.88%) and K-Nearest Neighbour (88%). Similarly, [24] aimed to develop an obesity prediction model. Two machine learning models were evaluated, and Linear Regression obtained a higher accuracy of 89.26% compared to K-Nearest Neighbour.

The previous studies showed that the Linear Regression, Random Forest Regressor, and MLP Regressor performed well on different types of datasets and situations. Therefore, these models are selected to be trained and compared in this project to determine the most suitable model for daily nutrient intake recommendation.

2.11 Proposed Solutions

Table 2.11.1 shows the functions of the previous works compared with the proposed application.

Table 2.11.1 Comparison of the Previous Works with Proposed Application

Functions	MyFitnessPal	YAZIO	HealthifyMe	Eatr	Foodvisor	LifeSum	Proposed Application
User registration and authentication	√	√	√	√	√	√	√
Manual meal logging	√	√	√	√	√	√	√
Nutrient details	√	√	√	√	√	√	√
Meal searching	√	√	√	√	√	√	√
Custom meals	√	√	√	√	√	√	√
Reminder	√	√	√	√	√	√	√
Community Forum	√	X	X	X	X	X	√
Nutrition label extraction	X	X	X	X	X	X	√
Health information input	X	X	X	X	X	X	√
Meal recommendation	X	X	X	X	X	X	√

The improvement for the limitations of the previous works addressed will be implemented in the proposed application. Many applications can count calories and track nutrients, but these applications lack of the functions of offering guidance that fits the actual health condition. With this project, meal recommendation would be shaped around the health conditions which would be factored in.

Furthermore, alerts would be built in to keep users on track. This is useful especially when a limit for nutrient such as calories, protein, carbohydrates, and fats is about to be crossed or missing for the day, as the proposed application could send a quick reminder. Compared to previous works, which focus mainly on tracking meal consumption, the proposed application would take a more proactive role, which is like having a nutrition coach on hand.

Chapter 3

System Methodology

The methodology for this project was carried out through several key phases, beginning with requirement identification and dataset selection, followed by data pre-processing to prepare the inputs for model development. The Linear Regression, Random Forest Regressor, and Multilayer Perceptron Regressor models were then trained and evaluated using standard performance metrics to generate accurate nutrient recommendations. After the best-performing model was selected, it was deployed using FastAPI and integrated into the proposed application, where functionalities such as meal logging, nutrient tracking, and personalised meal recommendations were implemented. The entire development process was conducted iteratively using the Agile methodology to ensure continuous refinement and reliable performance.

3.1 Design Specifications

3.1.1 Methodologies and General Work Procedures

Agile Methodology will be used for this project to develop the proposed application. Agile methodology is a method to handle project development by separating the work into smaller parts. It focuses on working together and making constant improvement in a repeating cycle [25]. There are 6 phases in Agile Methodology which are planning, design, development, testing, deployment, and review as shown in Figure 3.1.1.1.

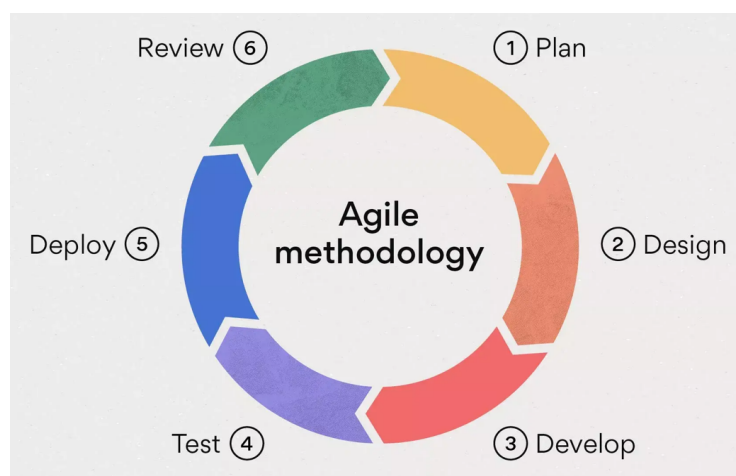


Figure 3.1.1.1 Agile Methodology Phases [25]

Planning

In the planning stage of this project, the key functions are identified, including user registration and authentication, meal logging, nutrient tracking, and the recommendation model. This phase also outlines additional tools such as progress tracking of nutrients consumption. Activities such as defining health-related data inputs and planning the integration of nutritional databases will be carried out during this stage to make sure that the proposed application meets both user health needs and application objectives.

Design

The design phase involves creating the layout for the front-end architecture, which serves as the user interface, including the meal logging dashboard, nutrient breakdown display, and personalised meal recommendations view. The back-end architecture is designed to manage user accounts, store historical meal data, and process nutrient calculations. The database is structured to store information such as meals items, nutrient values, health conditions, and model-generated recommendations.

Development

In the development stage, the core functions of the proposed are implemented. This includes coding the user registration and authentication functions, building the meal logging and nutrient tracking modules, and integrating the recommendation model to recommend suitable meals based on health conditions. Progress tracking functionality is developed to allow users to view historical trends and goal achievements. The proposed application is developed incrementally, with additional enhancements such as customised meal input for later iterations.

Testing

During the testing stage, it is ensured that users can register, log in, record meals, and receive accurate nutrient breakdowns. The recommendation model is tested to verify that it generates suitable meal recommendation according to the health condition of the users. The accuracy of nutrient calculations and the integrity of the stored data is also examined. Testing is conducted on the user interface and system performance to ensure compliance with functional requirements and reliability standards.

Deployment

In the deployment phase, the proposed application is released to allow users to log meals, track nutrients, and receive personalised meal recommendations in real time. Users can also review their progress history and adjust goals accordingly. Nutritional analytics are made available to help users identify patterns and improve dietary habits.

Review

Testing is conducted concurrently with development to ensure that the proposed application meets the defined requirements and functions correctly. Each function is evaluated incrementally as it is implemented, allowing early detection and correction of errors. The proposed application is further assessed through self-review and testing to verify usability, accuracy, and overall performance. Insights gained from these evaluations are applied in subsequent development cycles to improve the proposed application. For example, adjustments to the meal search functionality or enhancements to the recommendation model may be made based on observed performance and testing results. Continuous testing and review support iterative improvement and contribute to a high-quality and robust final solution.

Following the Agile methodology, the project returns to the planning phase for the next sprint after the review stage [25]. Through continuous iterations, the proposed application evolves to remain responsive to changing user needs and dietary trends.

3.2 System Design Diagram

3.2.1 System Architecture Diagram

Figure 3.2.1.1 illustrates the overall structure of the proposed application and the interactions of the components.

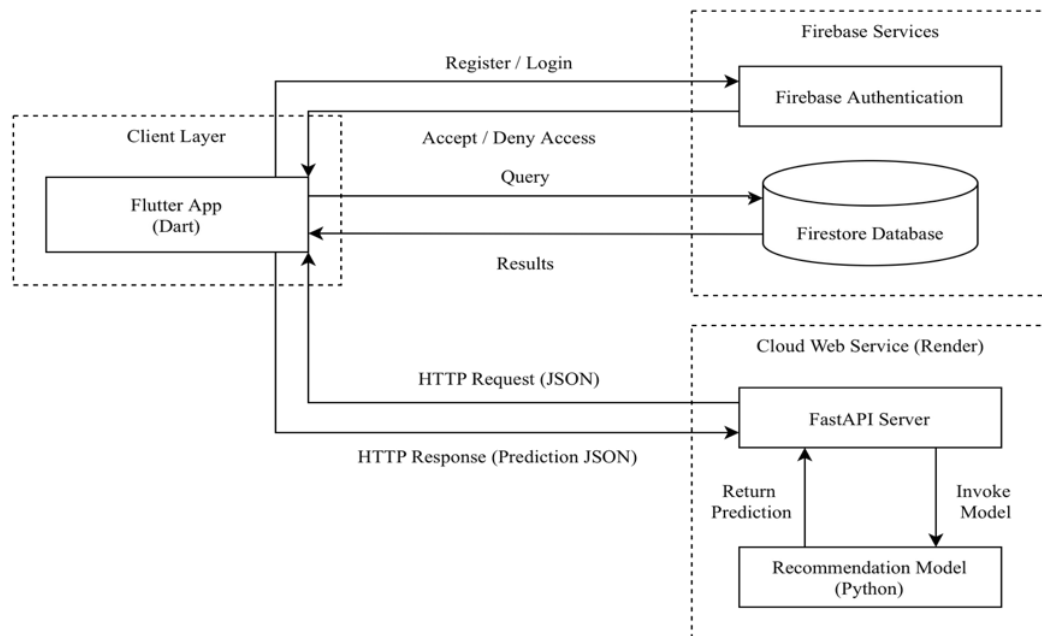


Figure 3.2.1.1 System Architecture Diagram

The Client Layer is represented by the mobile application, which is used as the main interface for user interaction with the mobile application. Actions such as registration, login, and viewing personalised meal recommendations are performed by users through this layer.

User authentication and data management are handled by the Firebase Services component. Specifically, user registration and login processes are managed by Firebase Authentication, while user-related data, including personal information and meal records, is stored in the Firestore Database. Queries are performed and results are retrieved through direct communication between the mobile application and Firebase.

The FastAPI server and the recommendation model developed in Python are hosted by the Cloud Web Service (Render). When a meal recommendation is required by the mobile application, an HTTP request in JSON format is sent to the FastAPI server. The request is processed by the server, the model is invoked, and prediction results are returned in JSON format to the mobile application.

3.2.3 Use Case Description

The use case description is elaborated based on the use case diagram in Figure 3.2.2.1. Table 3.2.3.1 shows one of the examples for Use Case Description for Register Account.

Register Account

Table 3.2.3.1 Use Case Description for Register Account

Use Case Name: Register Account	ID: 1	Importance Level: High
Primary Actors: New User, New Admin	Use Case Type: Detail, Essential	
Stakeholders and Interests: New User / New Admin – wants to register an account to access the proposed application.		
Brief Description: This use case describes how a new user or admin registers an account to access the proposed application.		
Trigger: The user or admin opens the proposed application for the first time after installation. Type: External		
Relationships: Association: New User, New Admin		
Normal Flow of Events: 1. The new user or admin installs and opens the application. 2. The application displays the registration page. 3. The new user or admin selects their role. 4. If the new user or admin wants to sign up manually, the S-1: Sign Up Manually is performed. 5. If the new user or admin wants to sign up with Google, the S-2: Sign Up Using Google is performed. 6. The application saves the registration information into the user database. 7. The new user or admin is now registered in the application.		
SubFlows: S-1: Sign Up Manually 1. The new user or admin enters an email address and password. 2. The new user or admin enters their password again for confirmation. 3. The application checks whether the fields are empty. 4. The application verifies that both passwords match. 5. The application checks whether the entered email already exists in the user database.		

6. The application checks whether the format of the entered email is correct. S-2: Sign Up Using Google 1. The new user or admin selects an email account to sign up. 2. The application checks whether the selected email already exists in the user database.
Alternate/Exception Flows: S-1: Sign Up Manually 3a. One or more fields are empty. The application displays an error message and prompts the user or admin to fill in all the fields. 4a. The passwords do not match. The application displays an error message and prompts the user or admin to enter the passwords again. 5a. The entered email already exists in the user database. The application displays an error message and requests a different email address. 6a. The entered email format is incorrect. The application displays an error message and prompts the user or admin to enter the email again. S-2: Sign Up Using Google 2a. The selected email already exists in the user database. The application displays an error message and requests a different email address.

The rest of the use case descriptions are provided in Appendix A, Table A.1.1 to A.1.18, including the primary actors, relationships, and interactions involved.

3.2.4 Activity Diagram

The activity diagram is elaborated based on the flows described in the use case description in Section 3.2.3. Figure 3.2.4.1 shows one of the examples for Activity Diagram for Register Account.

Register Account

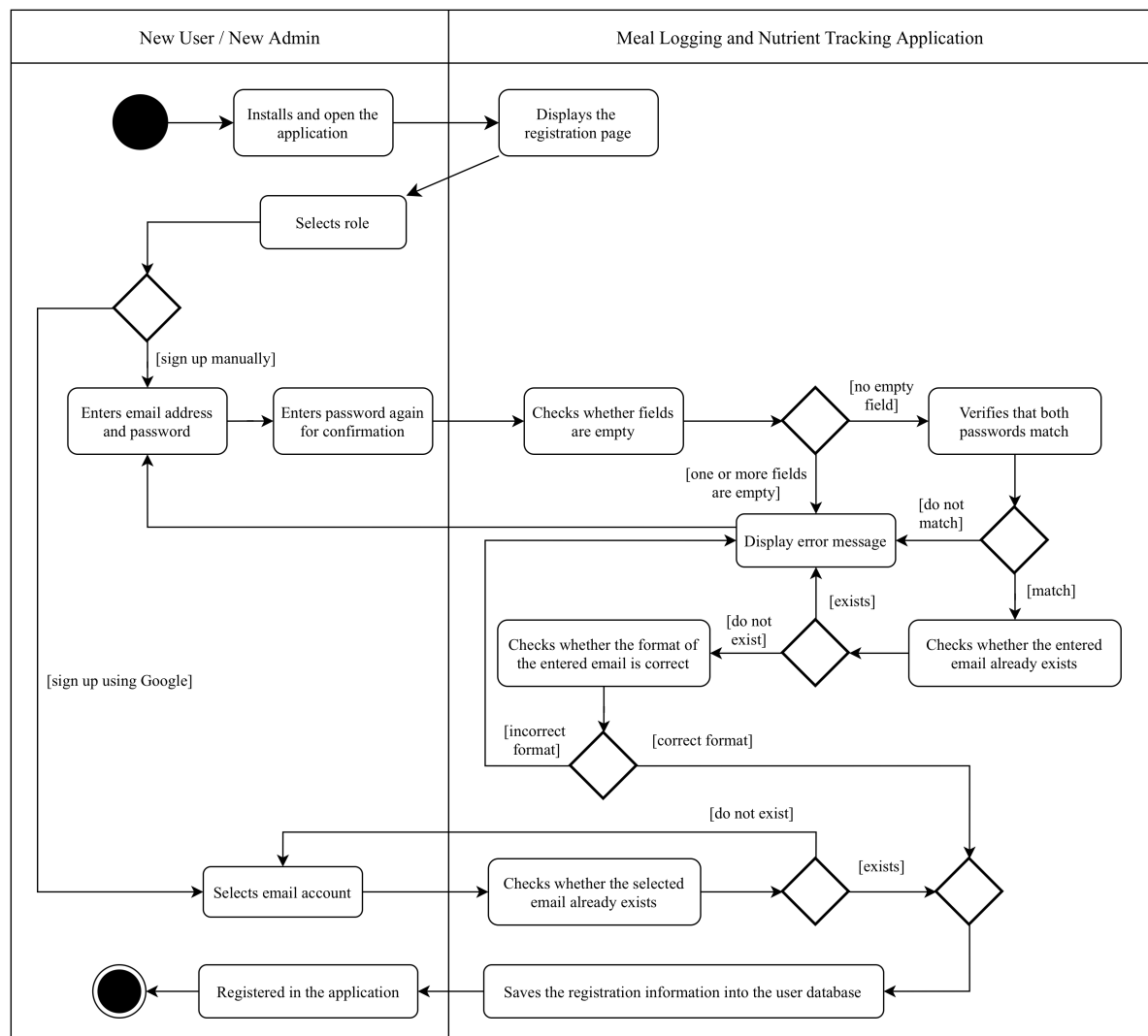


Figure 3.2.4.1 Activity Diagram for Register Account

The rest of the activity diagrams are provided in Appendix A, Figure A.2.1 to A.2.18, describing the sequence of user and application activities involved in the process.

3.3 Development of the Recommendation Models for Daily Nutrient Intake

3.3.1 Data Collection

The dataset used to train the models, named Personalized Medical Diet Recommendations Dataset and uploaded by Ziya, was downloaded from Kaggle for model development [26]. This dataset was chosen for this project as it was also used by other research as well for recommendation purposes [27]. The dataset contained 5000 rows and 30 attributes in total, out of which 14 attributes were selected for model training, as shown in Table 3.3.1.1. The attributes used included patient demographics, health indicators, and previous dietary intake. Specifically, patient age, height, weight, BMI, systolic and diastolic blood pressure, cholesterol level, blood sugar level, as well as protein, carbohydrate, and fat intake from the previous day were used as the training features. The model targets were the recommended protein, carbohydrate, and fat intake values.

Table 3.3.1.1 Attributes Used for Model Training

Attribute Name	Description	Attribute Type
Age	Age of the patient	Numerical
Height_cm	Height of the patient in cm	Numerical
Weight_kg	Weight of the patient in kg	Numerical
BMI	BMI of the patient	Numerical
Blood_Pressure_Systolic	Blood pressure systolic of the patient	Numerical
Blood_Pressure_Diastolic	Blood pressure diastolic of the patient	Numerical
Cholesterol_Level	Cholesterol level of the patient	Numerical
Blood_Sugar_Level	Blood sugar level of the patient	Numerical
Protein_Intake	Total protein intake of the previous day	Numerical
Carbohydrate_Intake	Total carbohydrate intake of the previous day	Numerical
Fat_Intake	Total fat intake of the previous day	Numerical
Recommended_Protein	Recommended protein intake	Numerical
Recommended_Carbs	Recommended carbohydrate intake	Numerical
Recommended_Fats	Recommended fats intake	Numerical

3.3.2 Dataset Preparation

The dataset was first examined, and it was found that no missing values were present. The data was then divided into X and y, where X represented the input features and y represented the target variable used for model training and prediction. After this separation, the dataset of 5,000 rows was further partitioned into training and testing sets by 80% and 20%. This distribution was selected based on the mathematical justification provided by [28], which demonstrates that an 80/20 split is an optimal configuration. A total of 4000 rows were allocated for training set, while the remaining 1000 rows were reserved for testing set, allowing the models to be trained on the majority of the data and evaluated on unseen samples.

3.3.3 Models Training

Random Forest Regressor

Random Forest Regressor model was trained using the training set. GridSearchCV was applied to tune the hyperparameters within a MultiOutputRegressor framework as shown in Figure 3.3.3.1. Following a coarse-to-fine search strategy [29], an initial broad search was conducted to identify the high-performance region for key parameters. This iterative approach ensures the identification of a local optimum while maintaining computational efficiency. A 3-fold cross-validation was utilized to evaluate combinations of tree depth and minimum split samples, with the final selection determined by the lowest Mean Absolute Error (MAE) to ensure maximum predictive accuracy.

```
# Base RF model wrapped for multi output
rf = MultiOutputRegressor(RandomForestRegressor(random_state=42))

# Grid search params
param_grid = {
    "estimator__n_estimators": [400, 500, 600],
    "estimator__max_depth": [3, 4, 5],
    "estimator__min_samples_split": [2, 3, 4],
}

# GridSearchCV
grid_search = GridSearchCV(
    estimator=rf,
    param_grid=param_grid,
    scoring="neg_mean_absolute_error",
    cv=3,
    verbose=2,
    n_jobs=-1
)

# Train with grid search
grid_search.fit(X_train, y_train)

print("Best parameters:", grid_search.best_params_)
```

Figure 3.3.3.1 Random Forest Regressor Model Training

Multilayer Perceptron Regressor

A Multilayer Perceptron (MLP) Regressor was also developed and trained to generate nutritional recommendations as shown in Figure 3.3.3.2. According to [30], the model was constructed using three types of layers, where the input layer received all selected features from the training dataset. Two hidden layers were included, consisting of 128 and 64 neurons respectively, and both layers used the ReLU activation function to introduce non-linearity and allow the model to learn complex relationships in the data. The output layer contained three neurons, corresponding to the predicted recommended protein, carbohydrate, and fat intake.

Before training, the model was compiled using the Adam optimiser, which provided efficient gradient-based learning, and the loss function was set to Mean Squared Error (MSE) because the task involved continuous numerical prediction. The model was trained for 50 epochs with 20% of the training data reserved for validation, allowing the monitoring of training and validation performance during the learning process [28]. According to [31], 50 epochs is commonly used for datasets between 1,000 and 10,000 rows. This setup enabled the neural network to iteratively adjust its internal weights to minimise prediction error and improve recommendation accuracy.

```
# Build a small neural network
model = keras.Sequential([
    layers.Input(shape=(X_train.shape[1],)),
    layers.Dense(128, activation='relu'),
    layers.Dense(64, activation='relu'),
    layers.Dense(3) # 3 outputs: Protein, Carbs, Fats
])

model.compile(optimizer='adam', loss='mse', metrics=['mae'])

# Train
history = model.fit(X_train, y_train, validation_split=0.2, epochs=50, verbose=0)
```

Figure 3.3.3.2 MLP Model Training

Linear Regression

Linear Regression model was trained using the training set without the need of fine tuning using GridSearchCV as the algorithm calculated the best fit linear line directly as shown in Figure 3.3.3.3. This simplified the training process and saved a lot of training time.

```
# Linear Regression
lr = MultiOutputRegressor(LinearRegression())
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)
```

Figure 3.3.3.3 Linear Regression Model Training

3.4 Meal Database

The meal database used in this project were obtained from the Malaysian Food Composition Database (MyFCD) [32]. A total of 810 meals were extracted, consisting of commonly consumed Malaysian foods across various categories such as rice dishes, noodles, snacks, beverages, and local specialties.

MyFCD was selected because it is an official and nationally recognised database that provides standardised and scientifically validated nutritional information for Malaysian foods. The database ensures that nutrient values are presented in a consistent and reliable format suitable for computational modelling and meal recommendation application. The extracted nutrient values were used as the primary reference for meal logging. Figure 3.4.1 shows an example of the meal information available in the MyFCD.

 MyFCD Malaysian Food Composition Database				
HOME ABOUT US PUBLICATIONS LINKS DIRECTORY USER MANUAL SIGN IN				
SWAMP CABBAGE / WATER CONVULVULUS (KANGKUNG) ; IPOMOEA AQUATICA/I. REPTANS Food Group : Vegetables and vegetable products Return to Search Results Print				
Nutrient	Unit	1 Value per 100g	2 2 stalks (L, 40 cm) [22.6 g]	1 1 cup (chopped) [77.8 g]
Proximates				
Water	g	90.9	20.54	70.72
Energy	Kcal	29	7	23
Protein	g	3.1	0.7	2.41
Fat	g	0.2	0.05	0.16
Carbohydrate	g	3.6	0.81	2.8
Fibre	g	1	0.2	0.8
Ash	g	1.2	0.3	0.9
Minerals				
Calcium, Ca	mg	88	20	68
Iron, Fe	mg	5.2	1.18	4.05
Phosphorus, P	mg	9	2	7
Potassium, K	mg	78	18	61
Sodium, Na	mg	65	15	51
Vitamins				
Vitamin C	mg	48.5	11	37.7
Thiamin (B1)	mg	0.1	0.02	0.08
Riboflavin (B2)	mg	0.55	0.12	0.43
Niacin (B3)	mg	0.6	0.1	0.5
Retinol	µg	n	n	n

Figure 3.4.1 Malaysian Food Composition Database

3.5 Timeline

3.5.1 FYP1 Timeline

Figure 3.5.1.1 shows the timeline for FYP1.

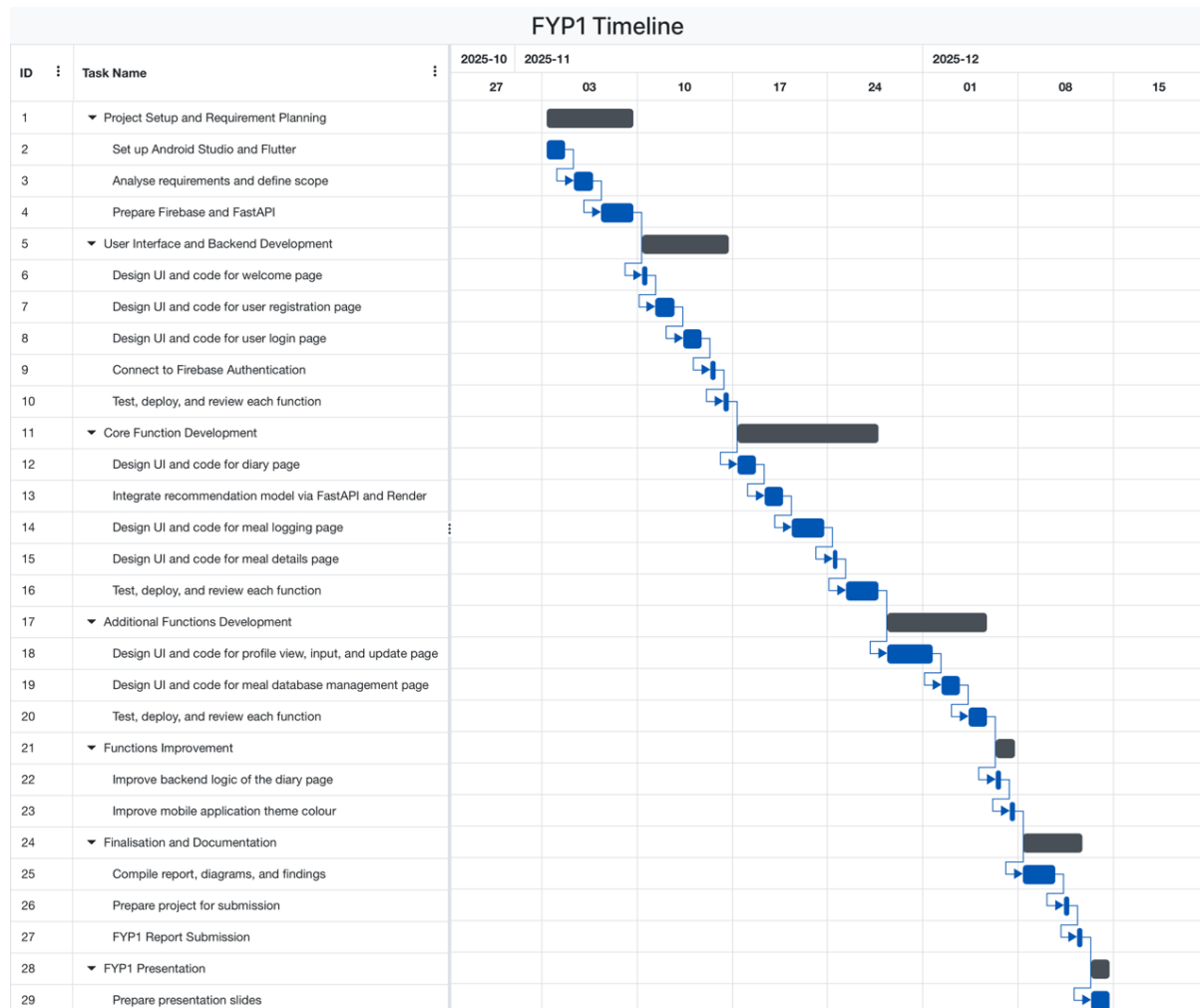


Figure 3.5.1.1 FYP1 Timeline

3.5.2 FYP2 Timeline

Figure 3.5.2.1 shows the timeline for FYP2.

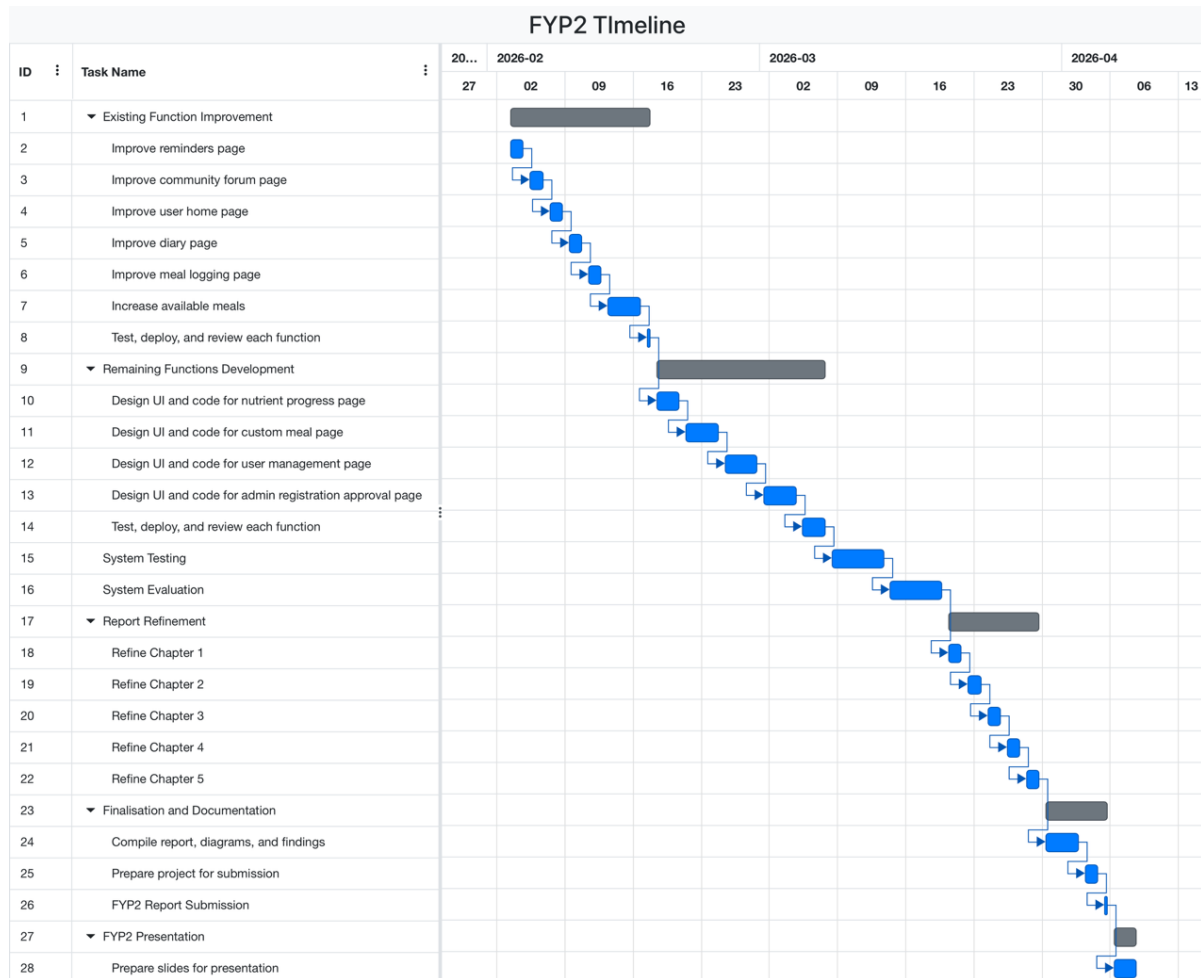


Figure 3.5.2.1 FYP2 Timeline

Chapter 4

System Design

4.1 System Block Diagram

Figure 4.1.1 shows the system block diagram of the proposed application which illustrates the interaction of the main components.

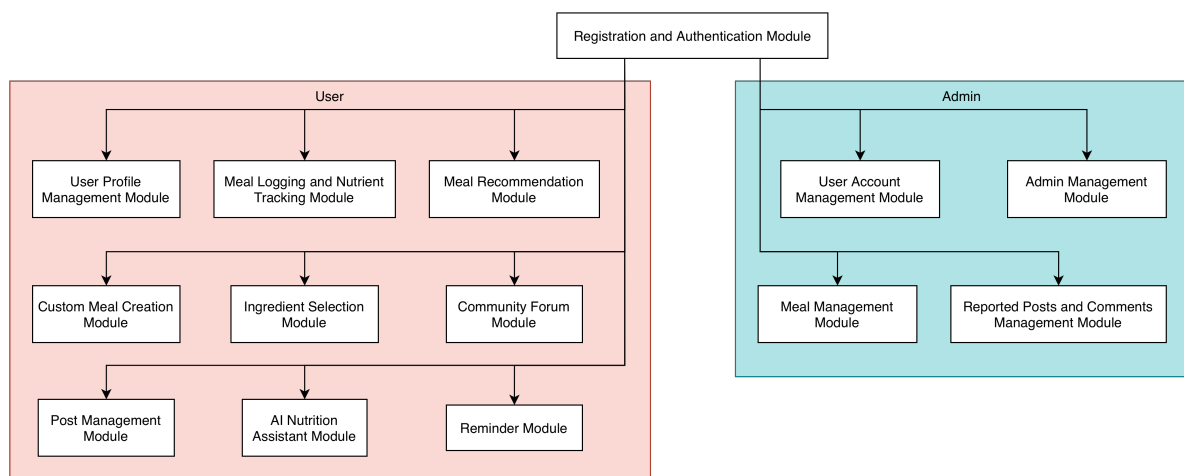


Figure 4.1.1 System Block Diagram

4.2 System Components Specifications

This section describes the major modules of the proposed application. Each module represents a core part of the proposed application and is responsible for handling specific functionalities. The proposed application is designed using a modular architecture to ensure maintainability, scalability, and clear separation of responsibilities.

Registration and Authentication Module

The registration and authentication module is responsible for managing registration, login, password recovery, and account access control. Account credentials are validated, password requirements are enforced, and third-party authentication is supported. Access is restricted based on account status, including active, banned, pending, or declined status. Users and admins are navigated to their respective home page based on their roles.

User Profile Management Module

The user profile management module is designed to manage personal and health information. User input is validated before submission. Unit conversions and health metric calculations are performed. All validated information is stored in the database and can be updated when required.

Meal Logging and Nutrient Tracking Module

The meal logging and nutrient tracking module is used to log meals, track water intake and monitor nutrient consumption. Progress indicators, balance values, and recommendation targets are updated based on user inputs. Special conditions, such as first-time usage and missing data scenarios, are handled appropriately. Navigation between different dates is supported while restricting access to future dates.

Meal Recommendation Module

The meal recommendation module is used to generate personalised meal recommendations based on recommended nutrient intake values, user profile information, and nutrient intake from previous records. The recommendations aimed to support users in meeting their nutritional requirements.

Custom Meal Creation Module

The custom meal creation module enables meals to be created using Quick Meal or Full Recipe modes. Nutritional values, serving options, and ingredient lists are validated before submission.

Ingredient Selection Module

The ingredient selection module allows ingredients to be searched and selected. Serving options and quantities are specified, and total weight is calculated. Selected ingredients are added to the ingredient list, and total grams are updated accordingly.

Community Forum Module

The community forum module facilitates user interaction through post creation, commenting, liking, saving, reporting, searching, and sorting. Post ownership is verified before edit or delete actions are permitted. Content updates are reflected in the respective forum lists.

Post Management Module

The post management module handles the creation, editing, and deletion of posts and comments. Input validation is performed before content is submitted. Reported content is forwarded to the reported posts and comments management module for admin review.

AI Nutrition Assistant Module

The AI nutrition assistant module provides meal and nutrient recommendations. Data is retrieved from the database and processed by available Google AI models. Each Google AI model limit is monitored, and alternative models are used when necessary. Error messages are displayed if all models are unavailable. Chat history can be cleared upon user confirmation.

Reminder Management Module

The reminder management module allows users to schedule, enable, disable, and delete reminders. Notifications are generated based on user settings and predefined meal period logic. Reminder configurations are stored and managed within the proposed application.

User Account Management Module

The user account management module enables admin control over user accounts. User accounts can be searched, sorted, banned, or unbanned. Banning timelines and admin actions are recorded and displayed. Account status updates are reflected in real time.

Admin Management Module

The admin management module is used to manage admin registrations and approvals. Pending applications can be approved or declined. Admin records can be searched and sorted. Registration status updates are reflected immediately in the proposed application.

Meal Management Module

The meal management module is utilised by admin to manage meal records. Meals can be added, edited, deleted, searched, filtered, and sorted. Nutritional values and serving options are validated before being stored in the database.

Reported Posts and Comments Management Module

The reported posts and comments management module is responsible for handling reported posts and comments. Reports can be ignored or associated content can be removed. A report history is maintained for monitoring purposes.

4.3 Database Design

A NoSQL document-based database provided by Firebase named Cloud Firestore is implemented as the database for the proposed application. The database structure is organised into collections and documents to store and manage data efficiently. Table 4.3.1 shows one of the examples for Database Design for the diary Collection.

Diary Collection

Document ID: diaryId

Table 4.3.1 Database Design for the diary Collection

Field / Subcollection Name	Data Type	Description	Example Value
date	String	The date of the diary	"Sunday, 04 Jan 2026"
includeSnacks	Boolean	Indicates whether the snack is included for the specific date	false
userId	String	The user account of this diary	"dreadfZlgCeEH9ouT6UMF9aKg222"

The rest of the database designs are provided in Appendix A, Table A.3.1 to A.3.20. Each table shows the field or subcollection name, data type, description, and example value to explain how the data is structured and stored in the database.

Chapter 5

System Implementation

5.1 Hardware Setup

A laptop and an android smartphone are used as the hardware in this project. The laptop is used for developing the proposed application, training a model to recommend nutrients, and performing testing on the proposed application. An android smartphone is used for final testing on real smartphone. Table 5.1.1 shows the laptop specifications and Table 5.1.2 shows the smartphone specifications.

Table 5.1.1 Specifications of Laptop

Description	Specifications
Model	MacBook Air (M1, 2020)
Processor	Apple M1 chip
Operating System	macOS Sequoia 15.5
Graphic	Apple-designed 7-core GPU
Memory	8 GB unified memory
Storage	256 GB SSD

Table 5.1.2 Specifications of Smartphone

Description	Specifications
Model	Samsung Galaxy M21
Processor	Exynos 9611
Operating System	One UI 4.1 Core
Memory	4 GB
Storage	64 GB
Main Camera	48MP 2160p

5.2 Software Setup

Android Studio

Android Studio was used as the main tool to develop the proposed application using Flutter and the Dart programming language. It provides useful features such as code suggestions, error checking, and a built-in debugger that make the development process easier and faster. The integrated terminal also helps in running Flutter commands directly inside the IDE. The Android Studio emulator was used to test and run the proposed application on virtual Android phones, so the proposed application could be checked without needing a real device.

Firebase

Firebase was implemented for the proposed application as the backend service. Firebase Authentication handled the login and registration process for both user and admin accounts, ensuring that only authorised users could access the proposed application.

GitHub

GitHub was used to store the files needed for the recommendation model. The model was saved as a .pkl file and uploaded to the GitHub repository together with the main.py script, where the model is loaded and executed using FastAPI. A requirements.txt file was also included to specify all the Python libraries required to run the model. These files were uploaded to GitHub so that the Render web service could access them and call the model.

Render

The FastAPI backend was hosted on Render to provide a deployment solution without the need for manual server management. During deployment, the main.py script and the recommendation model were fetched from the GitHub repository by Render. Requests from the proposed application were sent to the server over the internet and were processed by the FastAPI service using the recommendation model. Predictions were then returned to the proposed application in real-time. By using Render, the backend was kept online continuously, ensuring reliable access for the proposed application.

Google Colab

Google Colab was used to train the recommendation model in this project. The dataset was uploaded to the Colab environment, and all preprocessing, feature selection, and model training steps were performed within the notebook. The platform was chosen because it provides free access to computational resources such as GPUs, which accelerated the training time. All model parameters, including hyperparameter tuning and iterative training, were executed on Colab, and the results were subsequently saved for further evaluation and integration with the backend API.

5.3 Setting and Configuration

Android Studio Configuration

Android Studio was configured by installing the Flutter SDK and setting up the required environment variables. The Dart and Flutter plugins were also installed for mobile application development. In addition, an Android emulator was created and configured to test the proposed application without requiring a physical device. Figure 5.3.1 shows the Android Studio configured with Flutter project and emulator.

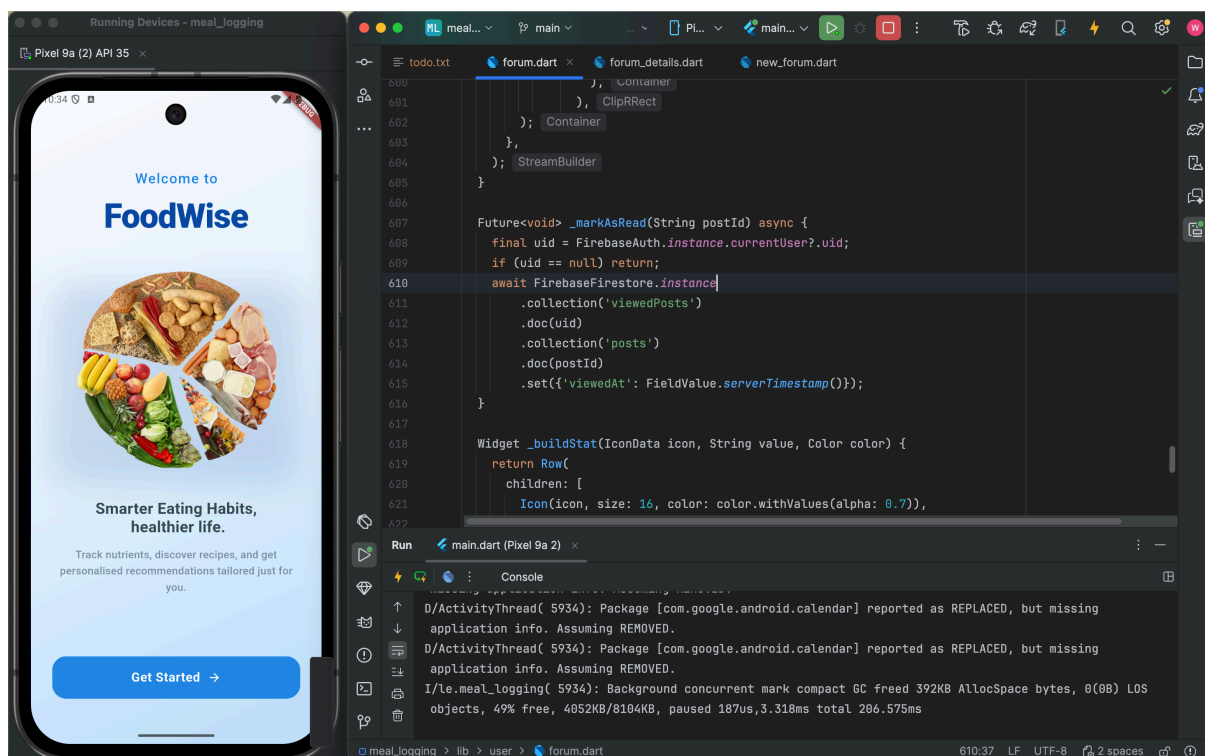


Figure 5.3.1 Android Studio Configured with Flutter Project and Emulator

Application Permissions Configuration

The required permissions were configured in the AndroidManifest.xml file to ensure the proposed application could function properly. These permissions include internet access, camera usage, notification permission, and alarm-related permissions required for the reminder function.

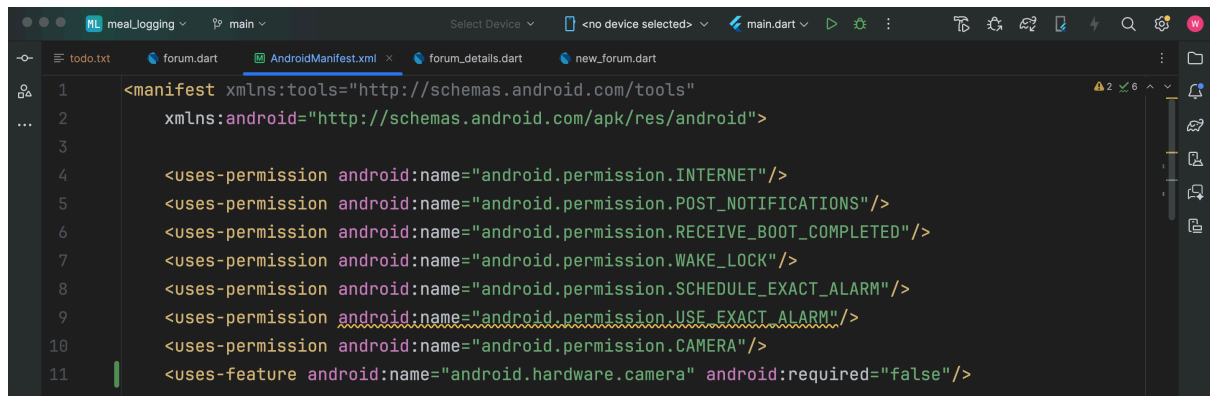


Figure 5.3.2 Application Permissions Configured in AndroidManifest.xml

Firebase Configuration

Firebase was configured by creating a new Firebase project and connecting it to the proposed application. Firebase Authentication was enabled to support the login and registration functions. The file needed for configuration was intalled and added to the project to allow communication between the application and Firebase services.

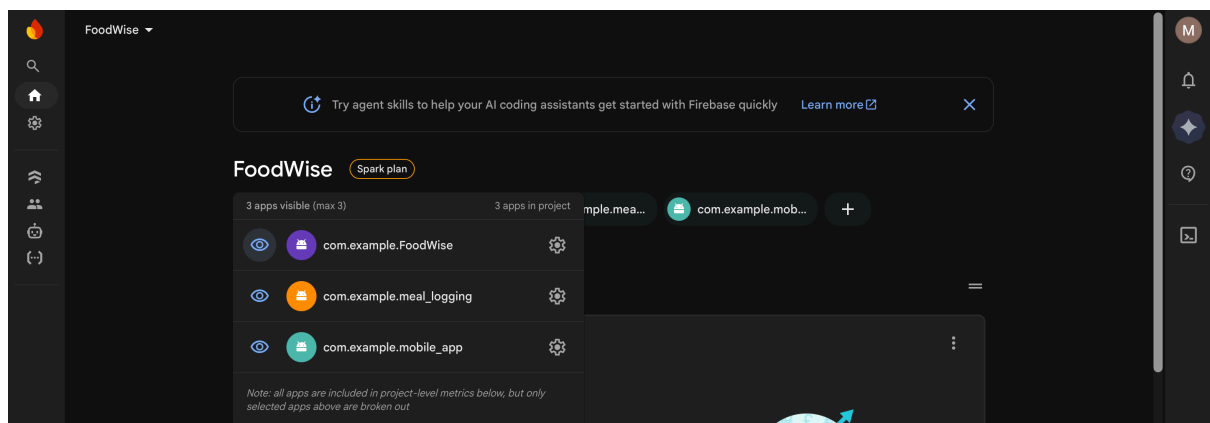


Figure 5.3.3 Firebase Project Configured for the Proposed Application

Backend Configuration

GitHub and Render were configured to deploy the FastAPI backend. The trained recommendation model and requirements.txt were uploaded to GitHub. The repository was then connected to Render, and the API link provided by Render was added to the proposed application to enable real-time communication.

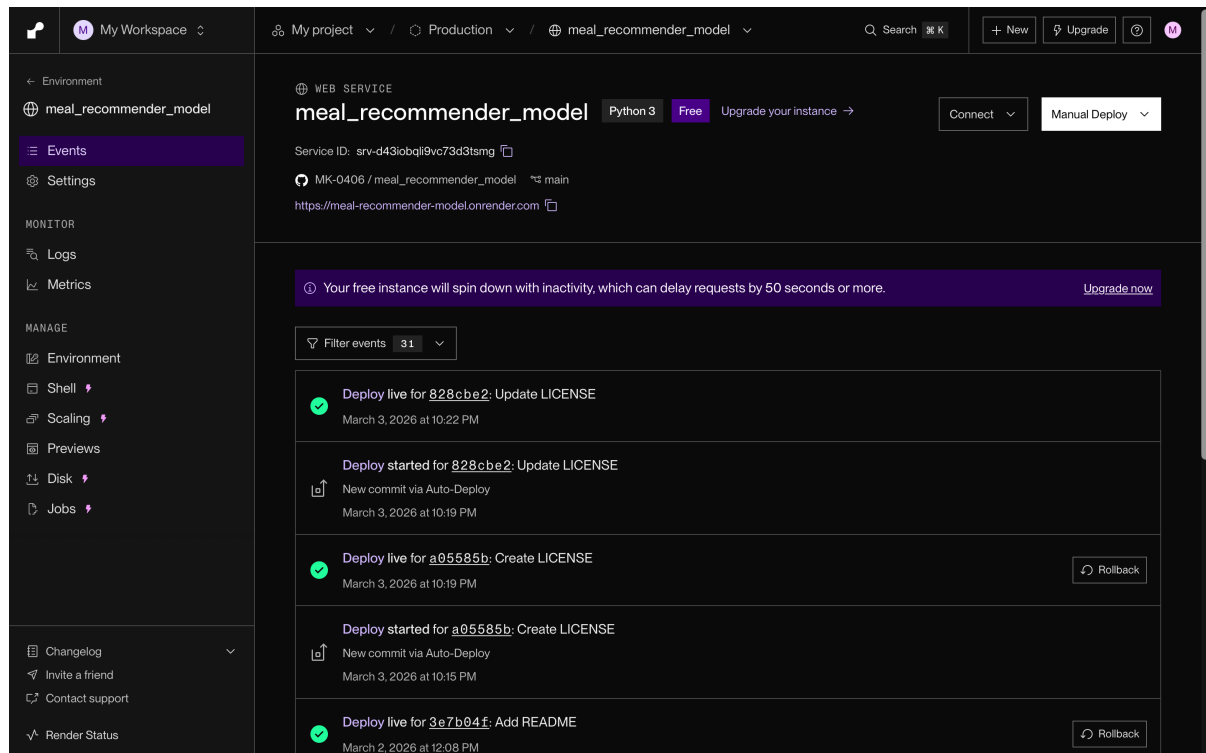


Figure 5.3.4 Backend API Successfully Deployed on Render

5.4 System Operation

Welcome Page

The welcome page is displayed when the user opens the proposed application as shown in Figure 5.4.1. The title “Welcome to FoodWise” is shown at the top, and an image containing various types of food is presented to introduce the theme of the app. A “Get Started” button is provided so that the next screen, which is the login page, can be accessed.

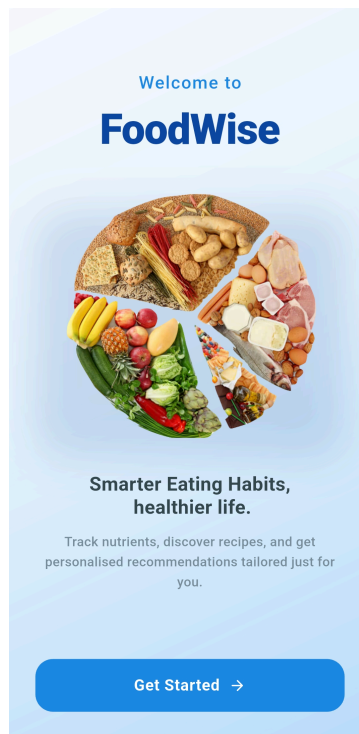


Figure 5.4.1 Welcome Page

Registration and Login Pages

The registration and login pages are used to create and access accounts in the proposed application. Two types of accounts can be registered which are user and admin as shown in Figure 5.4.2. After a user account is created, the user is allowed to log in immediately by entering the registered email address and password or selecting a Google account as shown in Figure 5.4.3 and Figure 5.4.4. For admin registration, the same information is required. However, admin access is not granted immediately. The admin account must first be approved by existing admin before login is allowed.

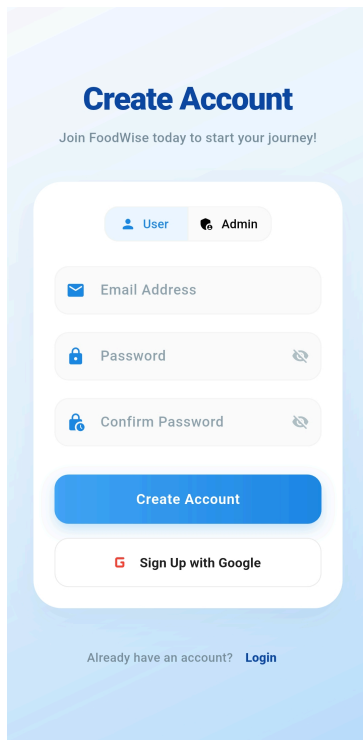


Figure 5.4.2 User and Admin
Registration Page

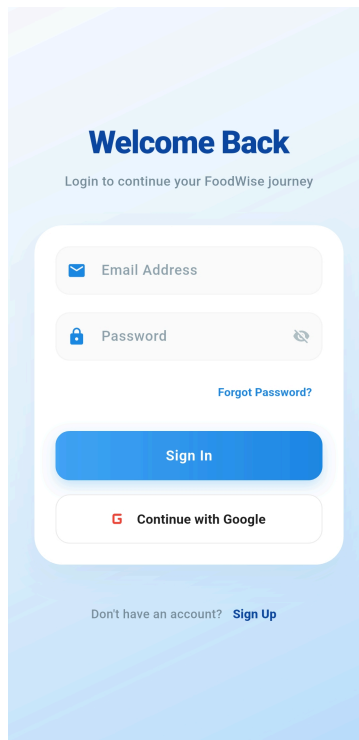


Figure 5.4.3 Login Page

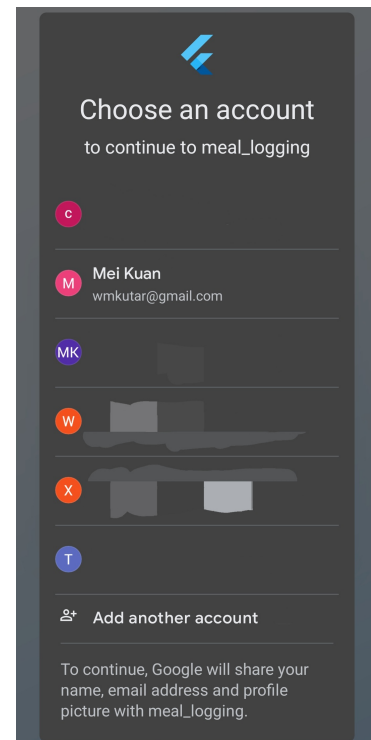


Figure 5.4.4 Email Account
Selection

If the user or admin has forgot their password, they may tap on the “Forgot Password” to enter their email address and the password reset link will be sent to their email as shown in Figure 5.4.5 and Figure 5.4.6. After the user or admin taps the reset link, a page is displayed allowing the user or admin to enter their new password and the password is updated when the “Save” button is tapped as shown in Figure 5.4.7.

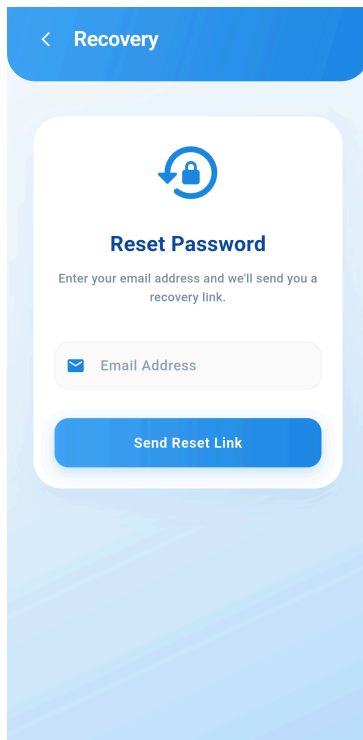


Figure 5.4.5 Password
Recovery Page

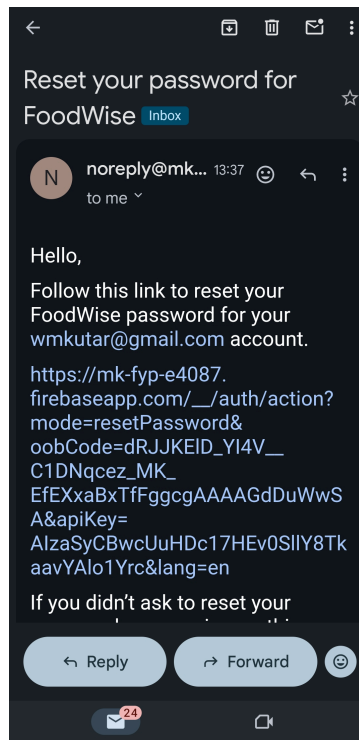


Figure 5.4.6 Password
Reset Email

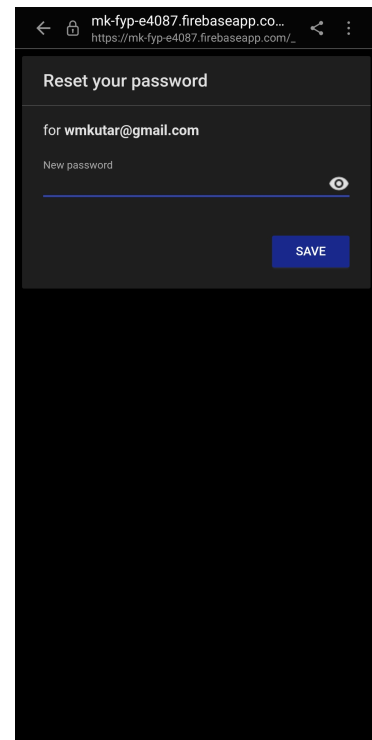


Figure 5.4.7 Password
Reset Page

If the email address format is wrong or registered, the proposed application will prompt the user to enter the email address again as shown in Figure 5.4.8 and 5.4.9. The minimum password length is set to six so the proposed application would prompt the user to enter the password again if the password entered length is less than six as shown in Figure 5.4.10. The entered password and password confirmation should have the same content else the proposed application will prompt the user to re-enter the password confirmation.

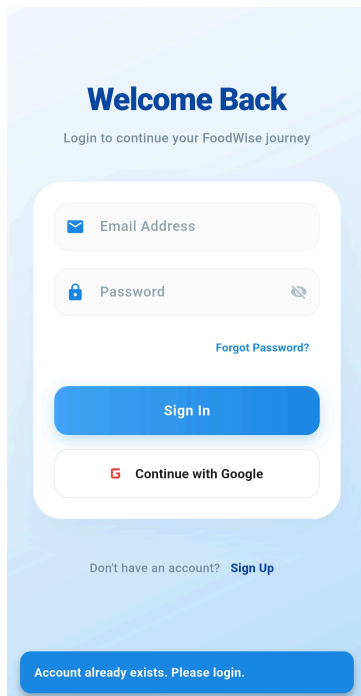


Figure 5.4.8 Email is Registered in Registration Page

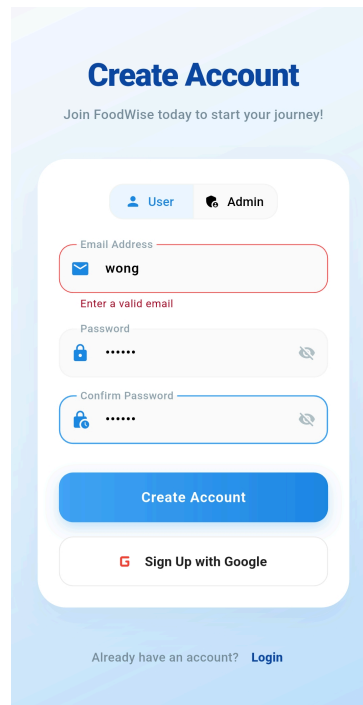


Figure 5.4.9 Incorrect Format Email in Registration Page

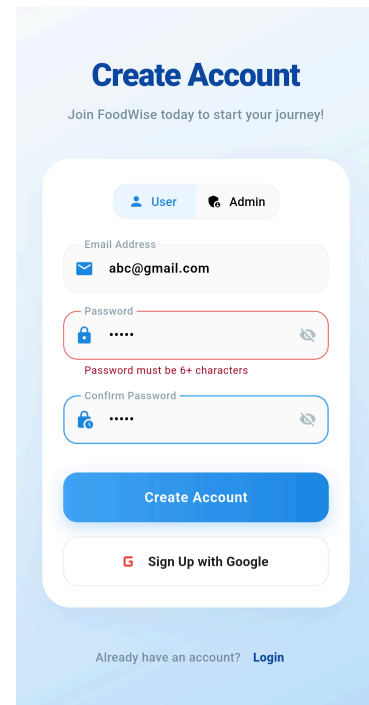


Figure 5.4.10 Password Length Error in Registration Page

User Home Page

The home page is displayed to the user after successful login. The health conditions of the user are identified and displayed to the user based on their BMI, blood pressure, blood sugar, and cholesterol values as shown in Figure 5.4.11. The range value of different categories, current value of the user, and a link for more information are displayed to the user when any of the health conditions is tapped as shown in Figure 5.4.12. The user will be navigated to a website which shows extra information together with the range values showed if the link is tapped [33], [34], [35], [36]. Other than that, the information of each nutrient is displayed for better understanding of the nutrients as shown in Figure 5.4.13.

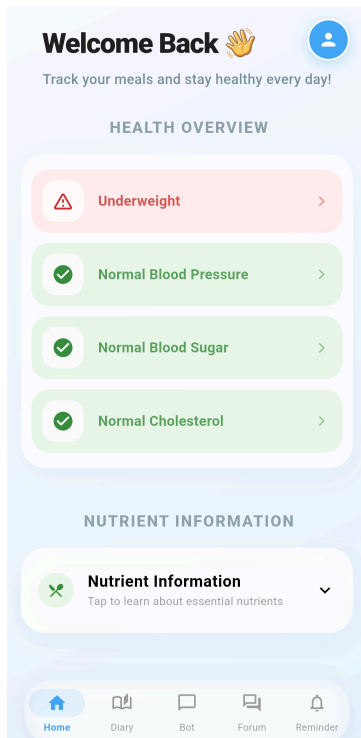


Figure 5.4.11 Health Conditions in User Home Page

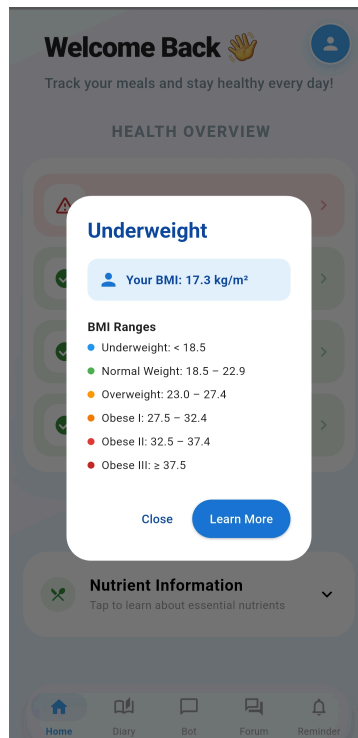


Figure 5.4.12 Health Conditions Information in User Home Page

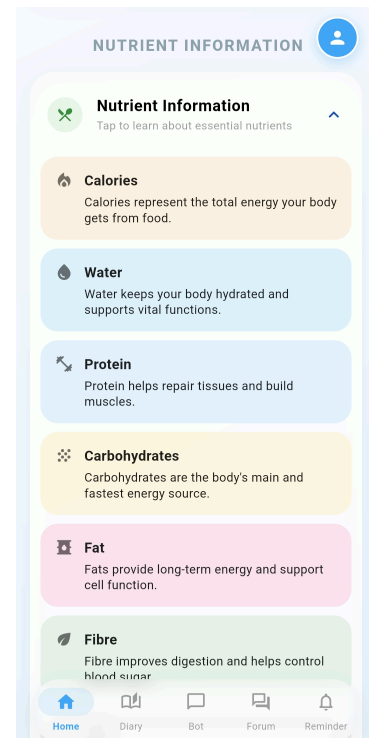


Figure 5.4.13 Nutrient Information in User Home Page

Diary Page

The diary page is used to display the daily recommended nutrient intake as shown in Figure 5.4.14. A progress bar is shown to indicate the total nutrient intake for the selected day compared to the recommended values. A date selector is provided for the user to view past or future records as shown in Figure 5.4.15. A water and four meal periods which are breakfast, lunch, dinner, and snack are presented on the page as shown in Figure 5.4.16. The daily recommended water intake for male is 3700 ml and female is 2700 ml [37].

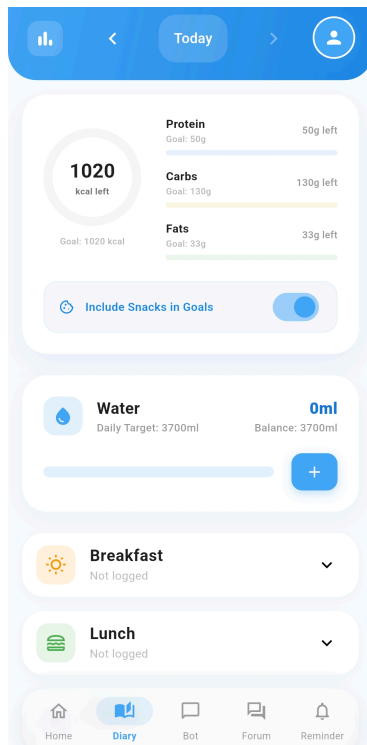
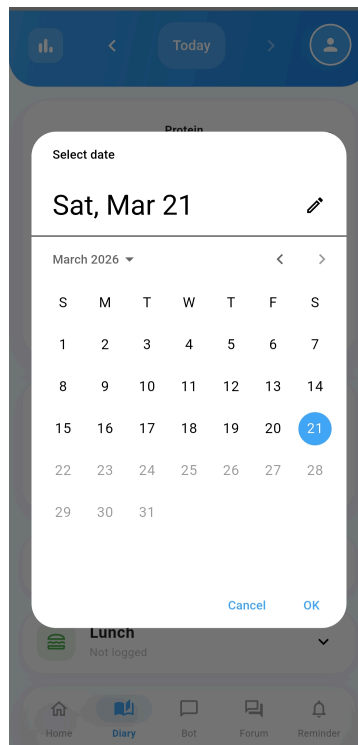


Figure 5.4.14 Progress Bars in Diary Page



Diary Page

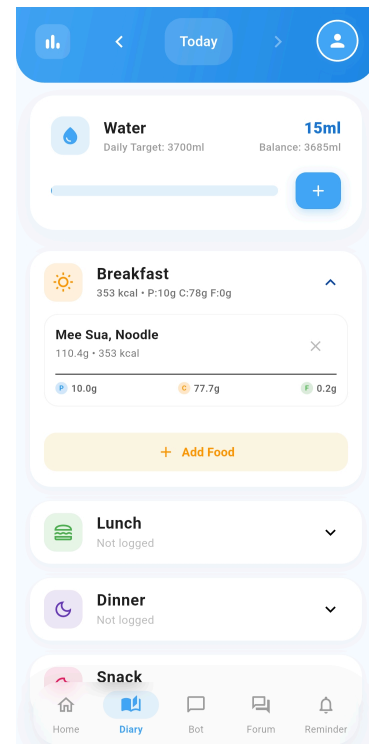


Figure 5.4.16 Meal Periods and Water in Diary Page

When any meal period is tapped, the list of meals logged for that period is displayed. The meal details can be viewed by tapping on a specific meal. The include snack for recommendation checkbox is displayed for user to choose whether snack need to be included for recommendation. By default, the snack will be included for the recommendation, and the weightage of the breakfast, lunch, dinner, and snack will be 20%, 31%, 39%, and 10% [38]. If the snack is excluded, then the weightage of the breakfast, lunch, and dinner will be 22%, 33%, and 45% [38].

Users are allowed to add new meals or delete existing ones, and the progress bars are updated automatically after any change. If the user logs water intake, a dialog box is displayed to enter or select the water intake amount as shown in Figure 5.4.17. The unit conversion of water from gram to millilitre (ml) is 1:1 [39]. If the user is using the proposed application for the first time, the proposed application will ask whether to use default recommendation or log the yesterday meals as shown in Figure 5.4.18. The default recommendation value for protein [40], carbohydrates [41], and fats [42] is within the minimum acceptable range. If the user has not logged any of the meal period, the proposed application will ask whether to log the meal now or later as shown in Figure 5.4.19.

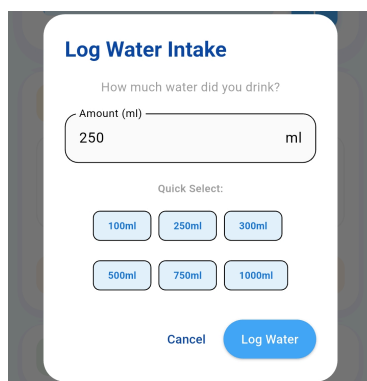


Figure 5.4.17 Water Intake
Logging in Diary Page

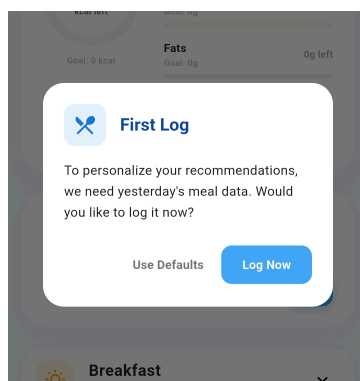


Figure 5.4.18 First Time
Logging in Diary Page

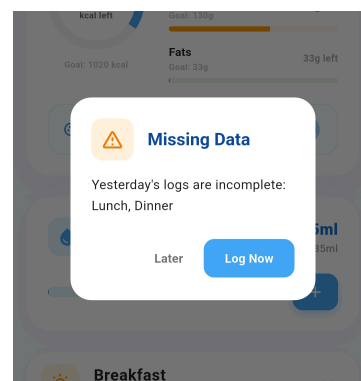


Figure 5.4.19 Missing
Yesterday Meals in Diary Page

Meal Details Page

The meal details page is used to display complete information about a selected meal. The meal name, category, food group, and nutrition highlights such as calories, protein, carbohydrates, and fat are shown at the top of the page together with a like button as shown in Figure 5.4.20. Detailed nutritional information per 100 g is also displayed on this page as shown in Figure 5.4.21.

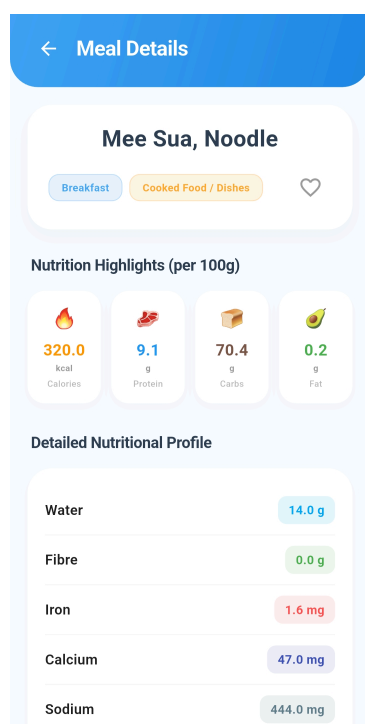


Figure 5.4.20 Upper View of the Meal Details
Page

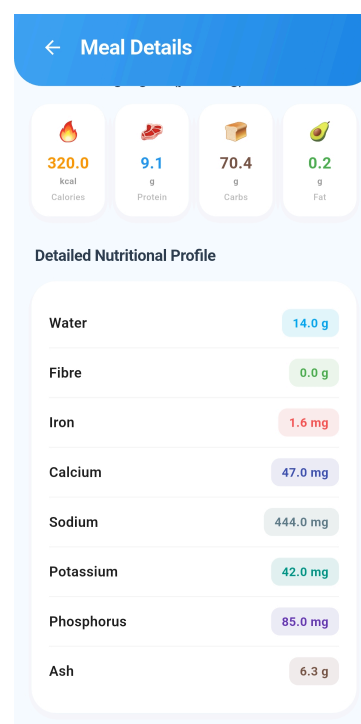


Figure 5.4.21 Lower View of the Meal Details
Page

Meal Logging Page

The meal logging page is used to help users find and log meals. A search bar is provided so that keywords can be entered to look for a specific or ideal meal. Based on the recommended nutrient intake for the selected meal period, a list of five random recommended meals is displayed in the “Recommended” tab as shown in Figure 5.4.22. The favourite meal and custom meal lists are also displayed in the “Favourite” and “Custom” tabs as shown in Figure 5.4.23 and Figure 5.4.24. A reload button is provided so that both lists can be refreshed at any time.

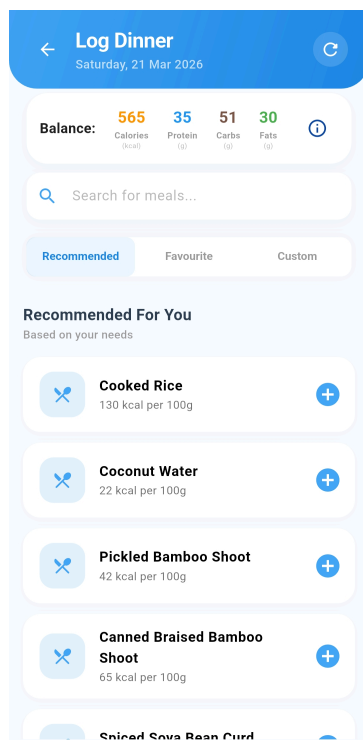


Figure 5.4.22 Recommended Meals in Meal Logging Page

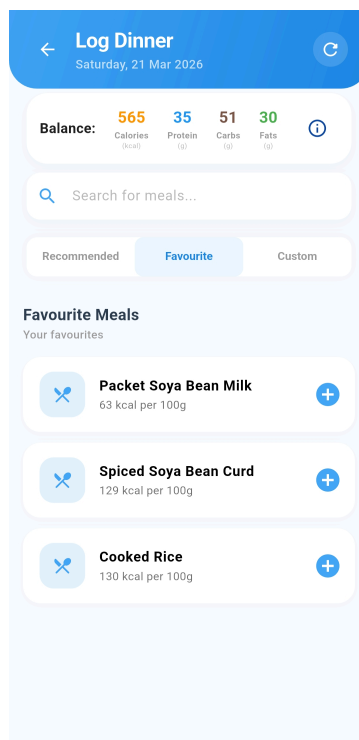


Figure 5.4.23 Favourite Meals in Meal Logging Page

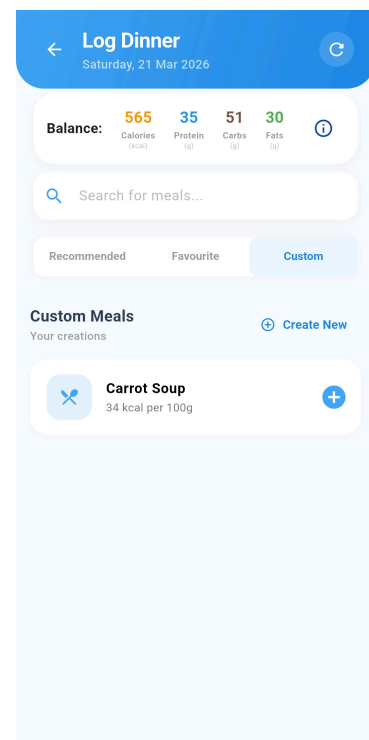


Figure 5.4.24 Custom Meals in Meal Logging Page

Any meal can be tapped to view its details, or the “+” button can be selected to log the meal. When a meal is being logged, the serving size is required to be entered by the user as shown in Figure 5.4.25. If the total nutrient of the meal exceeds the recommended nutrient, the proposed application will alert the user, but the user may continue to log the meal as shown in Figure 5.4.26. Once the serving size is confirmed, the meal is added to the selected meal period. The nutrient balance calculation details dialog box is also displayed when the user taps the info button as shown in Figure 5.4.27.

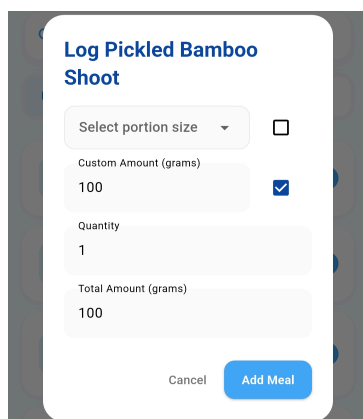


Figure 5.4.25 Serving Size Input in Meal Logging Page

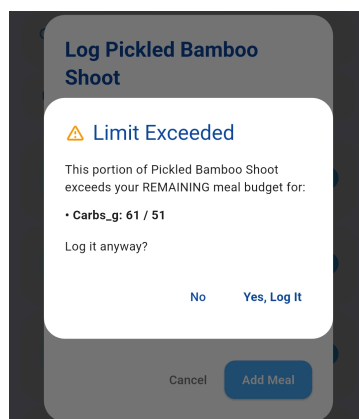


Figure 5.4.26 Exceeds Limit in Meal Logging Page

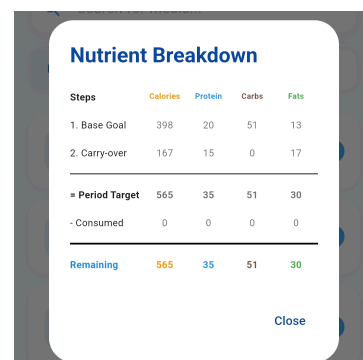


Figure 5.4.27 Nutrient Breakdown in Meal Logging Page

Custom Meal Page

The custom meal page allows users to create their own meals or recipes. If the users want to create meal manually, they may enter all required fields in the “Quick Meal” tab as shown in Figure 5.4.28 and Figure 5.4.29. The users may also snap the nutrition label or select a picture of nutrition label to extract the nutrient values, so they do not need to key in the nutrient values manually as shown in Figure 5.4.30.

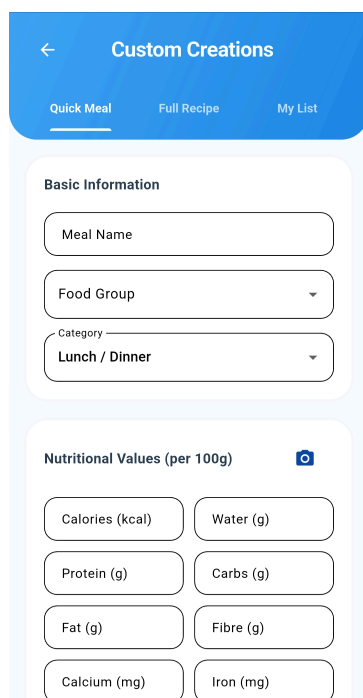


Figure 5.4.28 Quick Meal Tab in Custom Meal Page

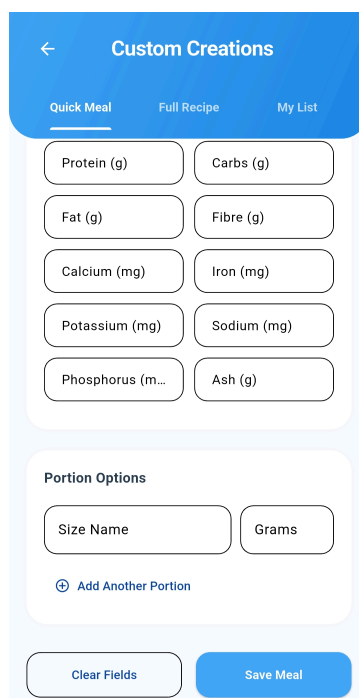


Figure 5.4.29 Quick Meal Tab in Custom Meal Page

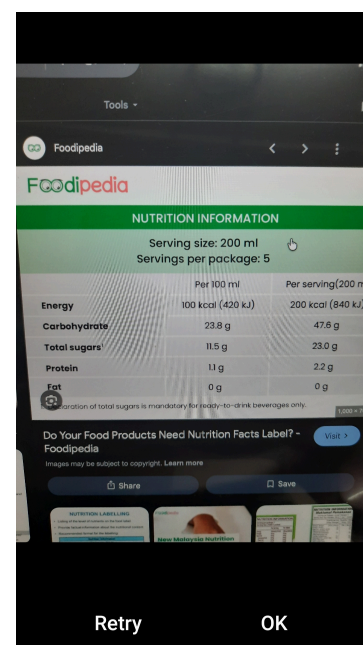


Figure 5.4.30 Nutrition Label Extraction in Custom Meal Page

In “Full Recipe” tab, the users may create meals using recipe. The users may enter the name, food group, food category, and serving size of the meal as shown in Figure 5.4.31. The users may tap the “+” button to add ingredients of the meal and the ingredient selection page will be displayed with a list of ingredients as shown in Figure 5.4.32. After the users select an ingredient, a dialog box is displayed requesting the total weight of the ingredient selected as shown in Figure 5.4.33.

Figure 5.4.31 “Full Recipe”
Tab in Custom Meal Page

Figure 5.4.32 Ingredient
Selection Page

Figure 5.4.33 Ingredient
Weight Input in Ingredient
Selection Page

In “My List” tab, the users may view the custom meals they created as shown in Figure 5.4.34. If the user wants to edit a meal, they may tap the edit button, and the proposed application will display the corresponding tab for the user to edit it. If the user wants to delete a meal, they may tap the delete button, and the proposed application will display a confirmation dialog box before deletion as shown in Figure 5.4.35.

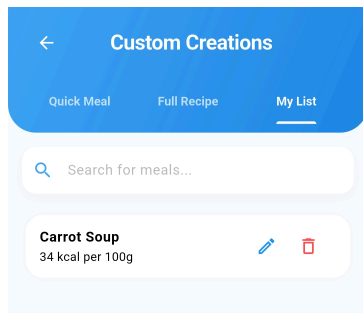


Figure 5.4.34 “My List” Tab in Custom Meal Page

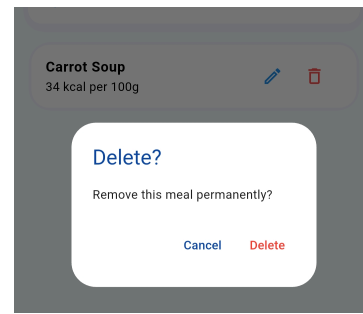


Figure 5.4.35 Delete Confirmation in Custom Meal Page

Nutrition Trends Page

The nutrition trends page allows users to track and monitor their nutrient intakes for the recent 7 days. Figure 5.4.36 shows the trend chart and average calories intake compared to recommended values for 7 days.

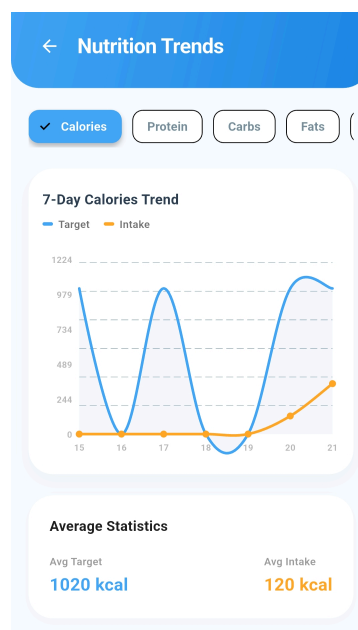


Figure 5.4.36 Nutrition Trends Page

AI Nutrition Assistant Page

The AI nutrition assistant page allows users to chat with AI bot about meal or nutrition. Users may ask about topics that is not provided in the proposed application. A Google AI API key is created and implemented, allowing the proposed application to send the question that

the user asked and receive a reply from the available Google AI model. The reply is based on the personal and health information or the meal intake records retrieved from the database. Figure 5.4.37 shows an example of chats between AI bot and user. If the user wants to delete chat history, they may tap the button on the header.

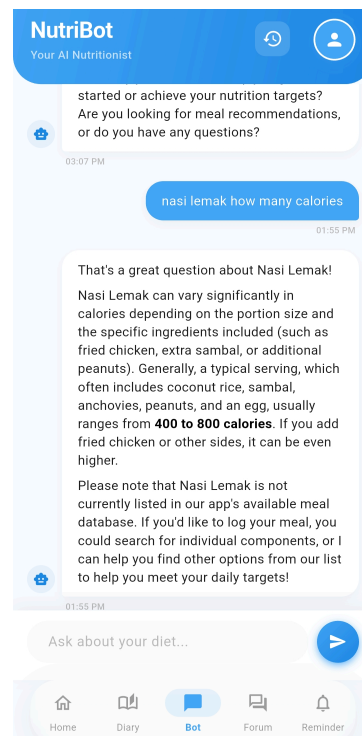


Figure 5.4.37 AI Nutrition Assistant Page

Community Forum Page

The community forum page allows users to share their experiences of maintaining good health among themselves. The users may scroll the page to view forum posts up to five posts in “Trending”, “Latest”, “Saved”, and “My Posts” tabs as shown in Figure 5.4.38. The unread posts are highlighted so that the user can easily identify new forum posts that have not yet been viewed. If the user wants to view more forum posts, they may tap on the “See more” text button then, five more forum posts will be displayed to the user until there is no more forum to show, the text “No more posts” will be displayed at the bottom as shown in Figure 5.4.39 and 5.4.40. Each post details such as post title, content, like, and comment counts are displayed for the user to have an overview of the forum.

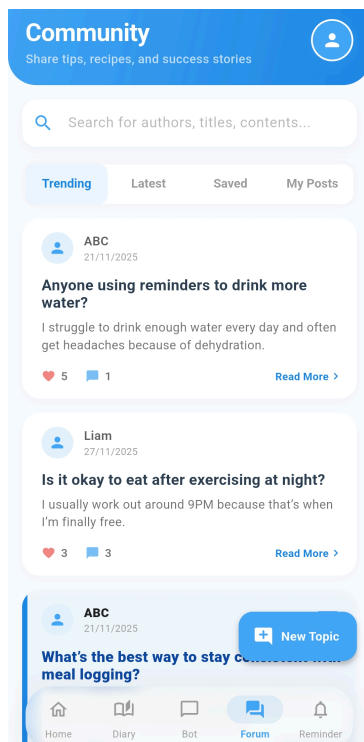


Figure 5.4.38 List of Forum Posts in Community Forum Page

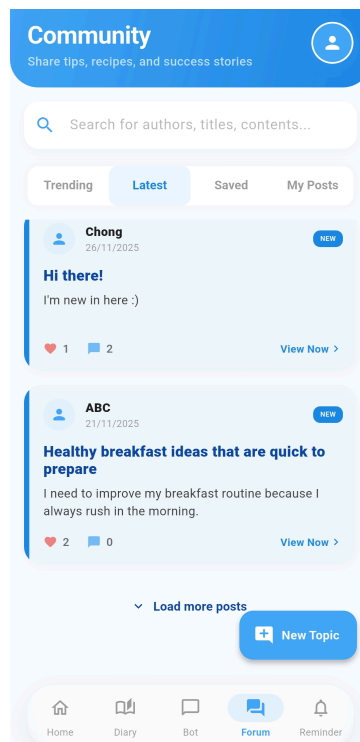


Figure 5.4.39 “Load more posts” Button in Community Forum Page

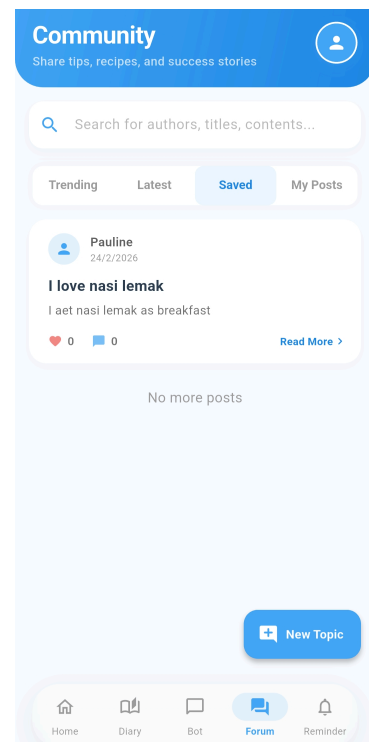


Figure 5.4.40 No More Posts in Community Forum Page

If the user is interested with the forum content, they may tap on the forum post, and the user will be navigated to the forum details page. The forum post details such as forum title, content, author, time posted, and list of comments are displayed as shown in Figure 5.4.41. If the user wants to add a comment or reply for the forum post, they may tap on the input field on the bottom of the screen and enter their comment. The send button beside the comment input field is used to send the comment and the comment sent will be updated to the comments list.

The user may sort the comment list by selecting one of the sorting options. If the post or comment content is helpful, the user may tap on the like button to help other users find out the most helpful forum. If the user wants to save the post, they may tap the save button. If the post or comment content is not suitable or inappropriate, the user may report it by submitting a report as shown in Figure 5.4.42. If the user wants to edit their post or comment, they may tap the edit button and change the contents as shown in Figure 5.4.43.

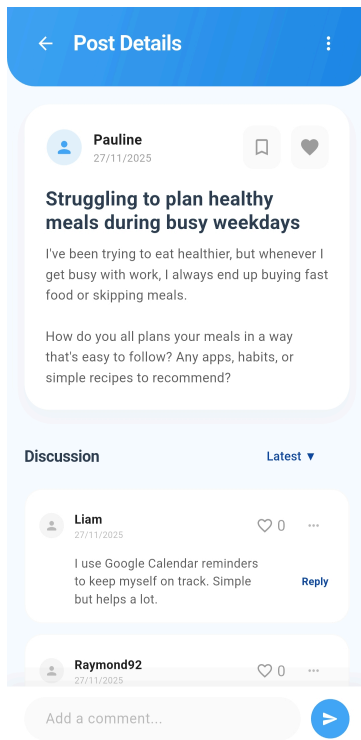


Figure 5.4.41 Post Details Page

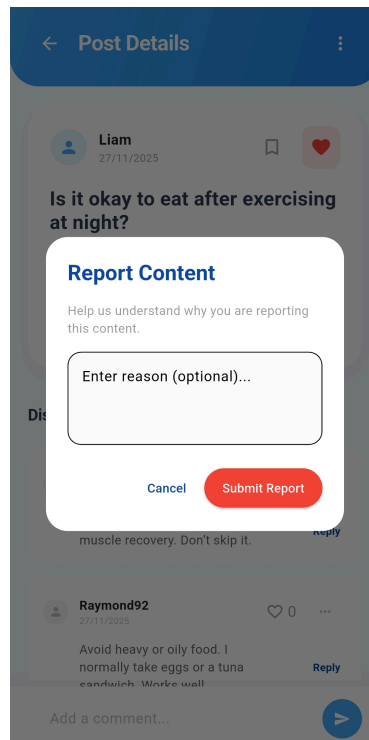


Figure 5.4.42 Report Post or Comment in Post Details Page

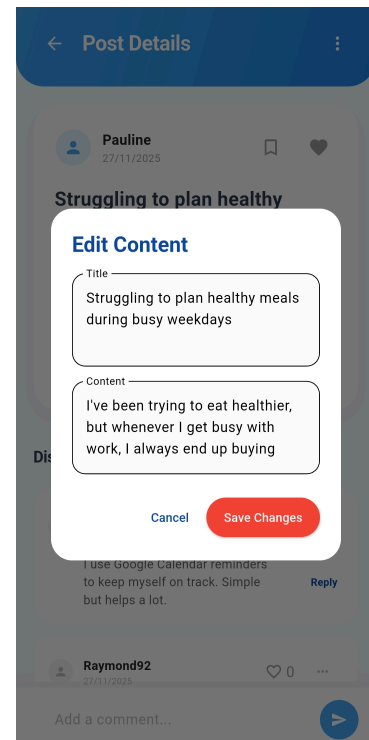


Figure 5.4.43 Edit Post or Comment in Post Details Page

On the other hand, the user may tap on the “New Topic” button to add a new forum post. The title and content of the new forum are needed to be entered by the user as shown in Figure 5.4.44. After the new forum details are entered, the user may tap on the “Post” button to upload the new forum post. If any of the details is empty, the user will be prompted by the proposed application to enter the empty field as shown in Figure 5.4.45.

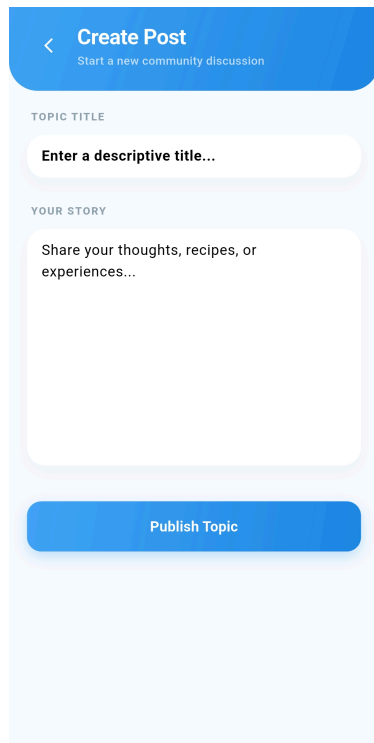


Figure 5.4.44 New Post Page

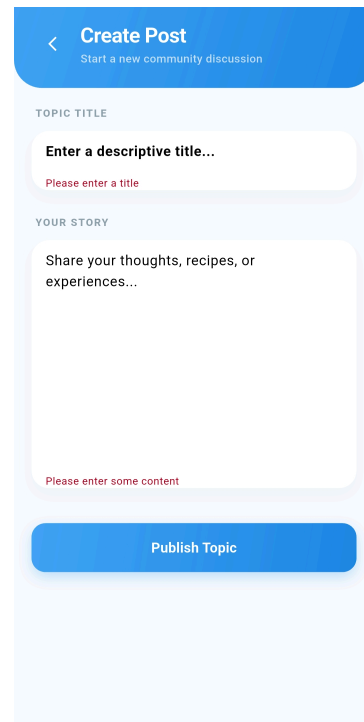


Figure 5.4.45 Empty Fields in New Post Page

Reminders Page

The reminders page is used to remind the user to have their meals on time as shown in Figure 5.4.46. The users may create new reminder by entering the reminder name and pick the reminder time in the dialog box after tapping the “+” button as shown in Figure 5.4.47. The time of each meal period can be set by tapping on the reminder card and a floating window will be displayed. The hour and minute can be set by tapping on the time picker or enter them manually as shown in Figure 5.4.48. Then, the Ante Meridiem (AM) and Post Meridiem (PM) which mean before noon and after noon are required to be selected by the user.

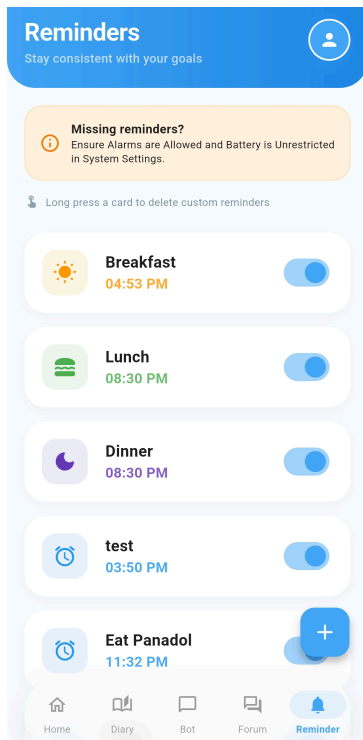


Figure 5.4.46 Reminders Page

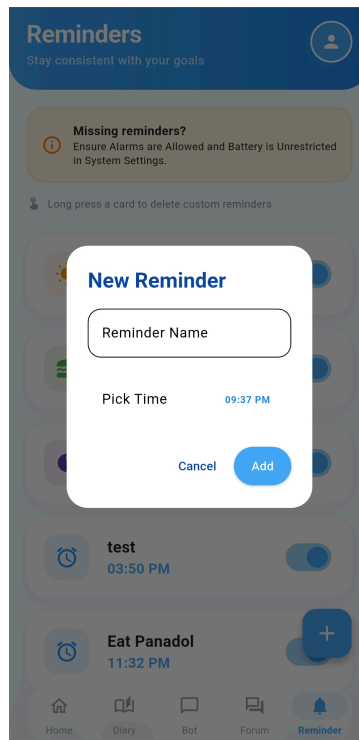


Figure 5.4.47 New
Reminder in Reminders Page

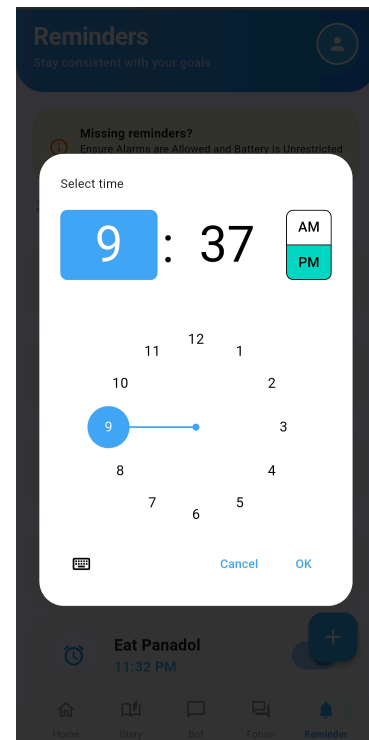


Figure 5.4.48 Time Picker in
Reminders Page

Profile Page

The profile page is used to display the personal and health information of the user as shown in Figure 5.4.49 and 5.4.50. The registered email address and the account type, which is user or admin, are shown at the top of the page. Personal information is also displayed, including name, gender, age, height, weight, and BMI. In addition, health metrics such as blood pressure, blood sugar, and cholesterol are presented on the page. Options are provided for the profile to be updated or for the user to log out from the proposed application.

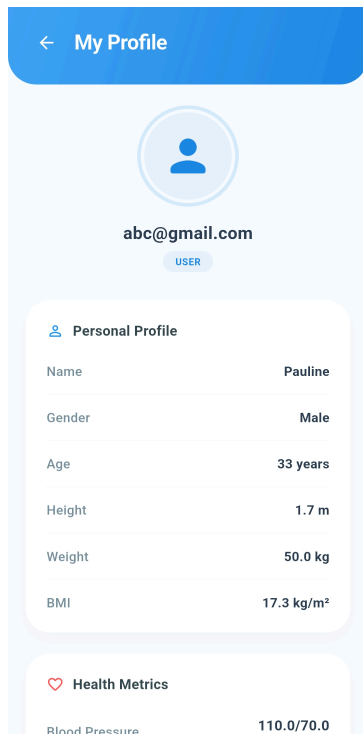


Figure 5.4.49 Upper View of Profile Page

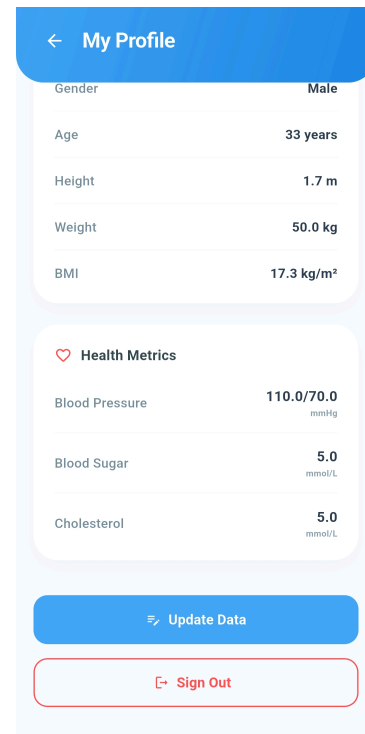


Figure 5.4.50 Lower View of Profile Page

Input and Update Profile Page

The input and update profile page is used to collect and modify the information of the user. Input fields are provided for personal information and health information as shown in Figure 5.4.51 and 5.4.52. For height, weight, cholesterol level, and blood sugar level, unit selections are also provided so that the correct measurement units can be chosen.

After the required information is entered, a save option is provided so that the updated details can be stored in the database. The data format and fields are validated, and the user will be prompted by the proposed application to input again if there are any incorrect data format or empty fields as shown in Figure 5.4.53. This page ensures that the personal and health information of the user can be maintained accurately.

Figure 5.4.51 Upper View of Input and Update Profile Page

Figure 5.4.52 Lower View of Input and Update Profile Page

Figure 5.4.53 Empty Field and Incorrect Data Format in Input and Update Profile Page

Admin Home Page

The home page is displayed to the admin after successful login and approval from the existing admin. The number of users and admins registered, total meals available in the database, the number of pending admin approval for new admin registrations, and total unresolved reports are displayed as shown in Figure 5.4.54. The admin will be navigated to the login page when the log out button is tapped.

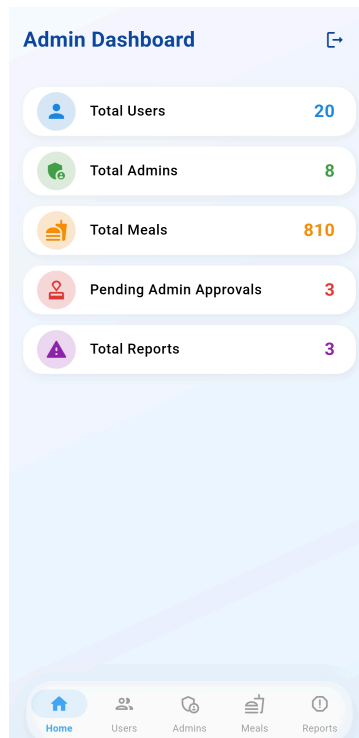


Figure 5.4.54 Admin Home Page

User Management Page

The user management page is used to manage user accounts. A list of active user accounts is displayed in the “Active” tab, while the list of banned user accounts is displayed in the “Banned” tab, as shown in Figure 5.4.55 and Figure 5.4.56. Sorting options are also provided to allow the user account list to be sorted based on alphabetical order or time, as shown in Figure 5.4.57. The admin may enter a keyword, such as a username or email address, to search for a specific user account.

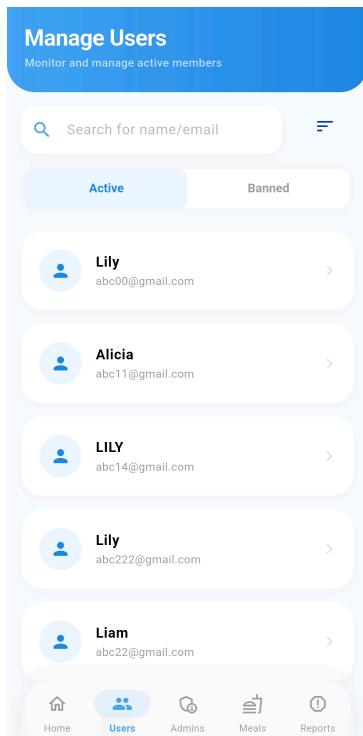


Figure 5.4.55 Active User List in User Management Page

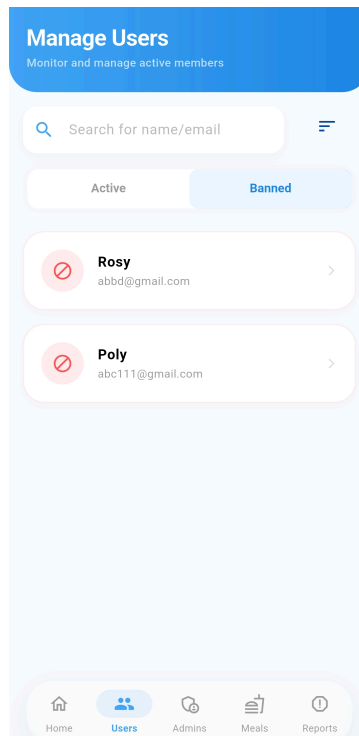


Figure 5.4.56 Banned User List in User Management Page

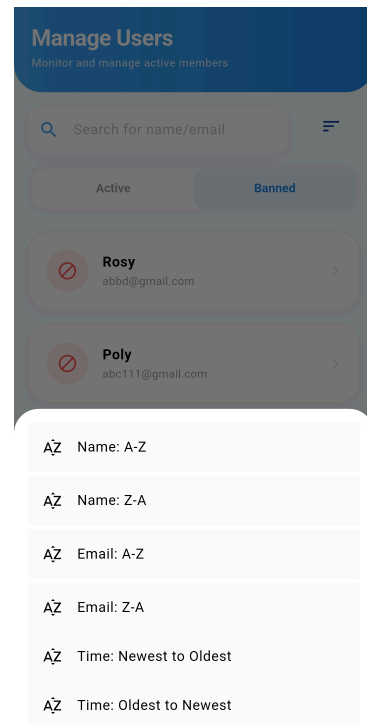


Figure 5.4.57 Sorting Options in User Management Page

If the admin wants to view the user account details, a dialog box consisting of the user account details will be displayed when the admin taps the user account card, as shown in Figure 5.4.58 and Figure 5.4.59. The admin may also view the banning timeline, as shown in Figure 5.4.60.

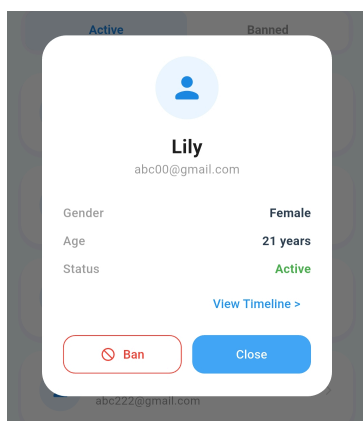


Figure 5.4.58 Active User Details in User Management Page

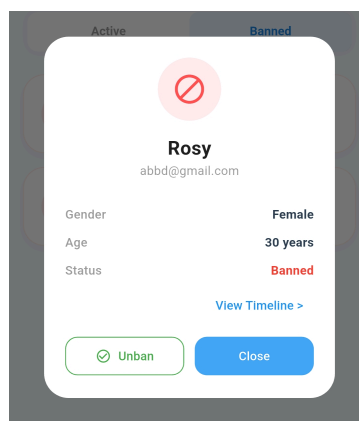


Figure 5.4.59 Banned User Details in User Management Page

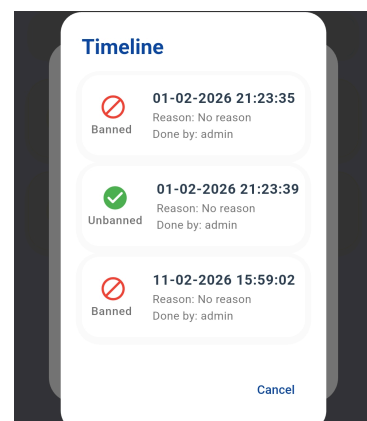


Figure 5.4.60 Banning Timeline in User Management Page

Admin Management Page

The admin management page is used to manage admin registrations. A list of pending admin registrations is displayed in the “Pending” tab, while the lists of active and declined admins are displayed in the “Active” and “Declined” tabs, as shown in Figure 5.4.61. Sorting options are also provided to allow the admin list to be sorted based on alphabetical order or time, as shown in Figure 5.4.62. If the admin wants to view the admin registration details, a dialog box consisting of the details will be displayed when the admin taps the admin card, as shown in Figure 5.4.63. The admin may also enter a keyword to search for a specific admin.

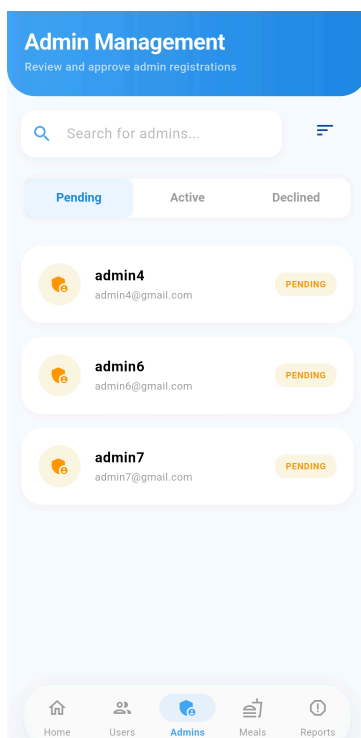


Figure 5.4.61 Admin List in Admin Management Page

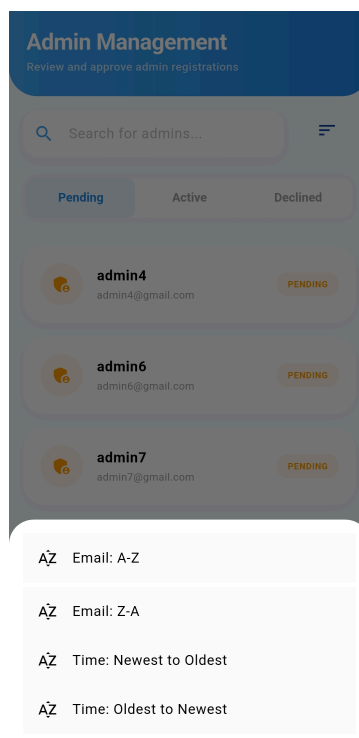


Figure 5.4.62 Sorting Options in Admin Management Page

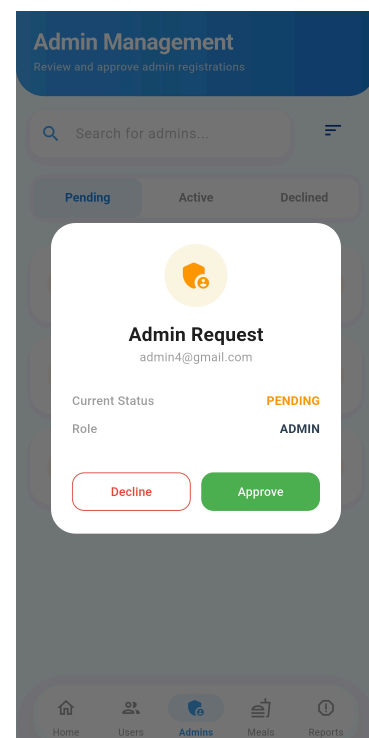


Figure 5.4.63 Admin Registration Details in Admin Management Page

Meal Management Page

The meal management page is used to manage the meal in the database. A list of meals is displayed together with the “Add New Meal” button as shown in Figure 5.4.64. The meal name and total calories in the meal per 100 g are displayed for each meal in the list. The admin may enter a keyword to search for a specific admin or filter the list based on the food group and category. When any of the meal is tapped, a floating window containing the meal details, edit, delete, and close button is displayed as shown in Figure 5.4.65. If the edit button is tapped by the admin, a floating window will be displayed with the existing details of the meal for editing as shown in Figure 5.4.66. If the delete button is tapped, the corresponding meal will be removed from the meal list.

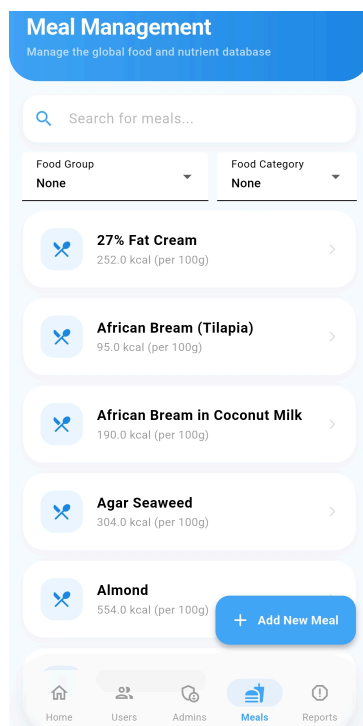


Figure 5.4.64 Meal Management Page

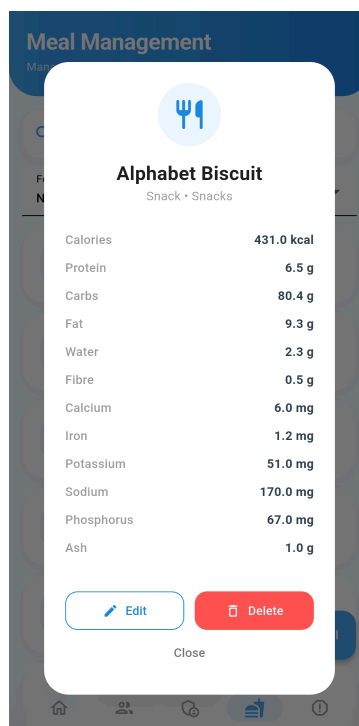


Figure 5.4.65 Meal Details in Meal Management Page

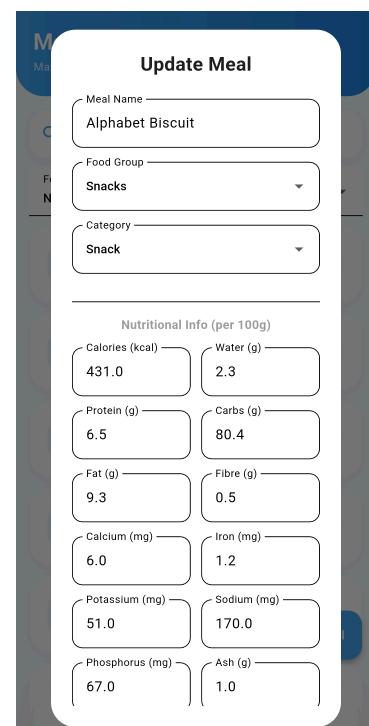


Figure 5.4.66 Edit Meal in Meal Management Page

When the “Add New Meal” button is tapped, a floating window is displayed with all the fields needed for a meal which are meal name, food group, food category, nutrients of the meal, and the serving options available for the meal as shown in Figure 5.4.67 and 5.4.68. The nutrients needed for each meal are water, calories, protein, fats, carbohydrates, fibre, ash, calcium, iron, phosphorus, potassium, and sodium per 100 g. When the “Add Serving Option”

is tapped, the serving name and grams input fields are displayed for the admin to input the values. The admin will be prompted if there is any empty field or incorrect data format as shown in Figure 5.4.69.

Figure 5.4.67 Upper View of Add Meal Window in Meal Management Page

Figure 5.4.68 Lower View of Add Meal Window in Meal Management Page

Figure 5.4.69 Empty Fields and Incorrect Data Format in Meal Management Page

Reported Posts and Comments Management Page

The reported posts and comments management page is used to manage reported posts and comments. A list of post or comment reports, which includes the name of the author, title, content, and reason for reporting, is displayed as shown in Figure 5.4.70. The report history page is displayed when the admin taps the history button, as shown in Figure 5.4.71.

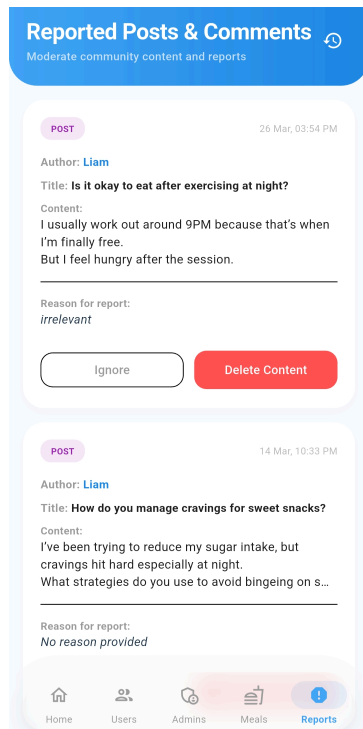


Figure 5.4.70 Reported Posts and Comments Management Page

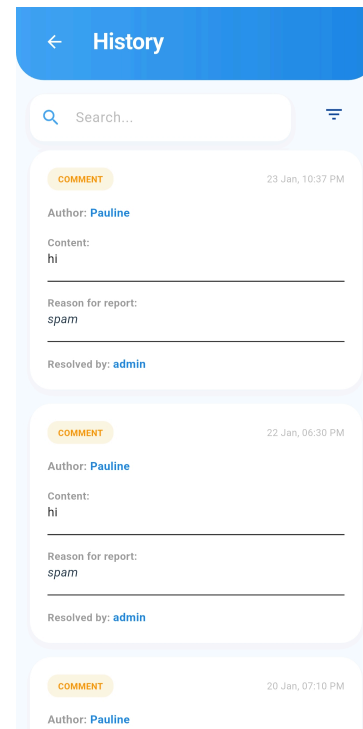


Figure 5.4.71 Report History Page

5.5 Recommendation Models for Daily Nutrients Intake

Random Forest Regressor

Figure 5.5.1 shows the best parameters obtained from performing grid search and Figure 5.5.2 shows the results obtained from the best Random Forest Regressor model. A high overall R^2 score of approximately 0.935 was achieved by the model which indicates that the model can explain about 93.5% of the variance in the outputs. This means the predictions for protein, carbohydrates, and fats closely match the actual values in the dataset. The mean absolute error (MAE) obtained by the model remained relatively low across all nutrients, with errors around 7.6 g for protein, 25.5 g for carbohydrates, and 7.6 g for fats. The good performance of the model was proved by the RMSE values which penalise larger errors more heavily and shows that the model produces stable and consistent predictions without extreme deviations.

```
Fitting 3 folds for each of 27 candidates, totalling 81 fits
Best parameters: {'estimator__max_depth': 4, 'estimator__min_samples_split': 2, 'estimator__n_estimators': 500}
```

Figure 5.5.1 Best Model of Random Forest Regressor Model

Overall R² Score: 0.9347455992625374
MAE: [7.55928602 25.49510552 7.64219402]
RMSE: [8.78915178 29.46030577 8.81525038]

Figure 5.5.2 Random Forest Regressor Results

Multilayer Perceptron Regressor

Figure 5.5.3 shows the results obtained by the multilayer perceptron regressor model. A high overall R² score of approximately 0.932 was achieved by the model which means the model can explain around 93.2% of the variance in the target nutrients. Strong predictive capability was achieved by the model although the R² is slightly lower than the Random Forest Regressor model. The MAE values of the model were around 7.8 g for protein, 25.9 g for carbohydrates, and 7.7 g for fats, showing that the mean absolute error remained small and acceptable. Based on the RMSE values with a similar pattern, the MLP model achieved stable predictions but tends to produce slightly higher variability compared to the Random Forest Regressor model.

Overall R² Score: 0.9322229083361112
MAE: [7.7962134 25.90069801 7.65757321]
RMSE: [9.16570216 29.96222764 8.86215173]

Figure 5.5.3 Multilayer Perceptron Regressor Results

Linear Regression

Figure 5.5.4 shows the results obtained by the linear regression model. An overall R² score of approximately 0.936 was achieved by the model which means the model can explain around 93.6% of the variance in the target nutrients. Strong predictive capability was achieved by the model as its R² value is the highest compared to the Random Forest Regressor and MLP model. The MAE values of the model were around 7.5 g for protein, 25.3 g for carbohydrates, and 7.6 g for fats, showing that the mean absolute error remained small and acceptable. Based on the RMSE values with a similar pattern, the linear regression model achieved the most stable predictions among the models.

R²: 0.935975468240058
MAE: [7.45347818 25.34531344 7.57323703]
RMSE: [8.65519436 29.28612711 8.70676842]

Figure 5.5.4 Linear Regression Results

Model Selection

Linear Regression model was selected as the model achieved the highest R^2 value compared to Random Forest Regressor and MLP models, indicating that it explained more variance in the dataset and produced more reliable predictions. The MAE of Linear Regression model was lower than Random Forest Regressor and MLP models, showing that the predicted daily recommended nutrient intake was closer to the actual daily recommended nutrient intake. On the other hand, the RMSE of Linear Regression model was also the lowest among the models, meaning it had the smallest overall difference between predicted and actual values.

Linear Regression model performed better because it focused only on the major trends in the data. Unlike Random Forest and MLP, which tried to memorize every small detail and random error, the Linear Regression model remained stable. Because it is a simpler model, it was less likely to overfit, leading to a more reliable result when testing it on new, unseen data. Since the personal and health values do not follow linear patterns which means they are mostly independent to each other, Random Forest Regressor captures these complex interactions without heavy tuning. Unlike MLP, which required specific tuning of learning rates, hidden layers, and epochs. Linear Regression model trained faster and was easier to optimise in this project where the computational resources or time are limited.

5.6 Implementation Issues and Challenges

Several issues and challenges were encountered during the implementation of the proposed application. One of the main difficulties involved integrating the recommendation model, which was developed in Python and deployed using FastAPI, with the Flutter mobile application. Ensuring smooth communication between the backend API and the mobile frontend required careful handling of API endpoints, response formats, and error cases. Another challenge was handling the dataset during model development. Although the dataset contained no missing values, selecting appropriate features and tuning hyperparameters demanded multiple iterations, especially for the MLP model, which was sensitive to scaling and prone to unstable training if the data was not standardised properly.

Model deployment also presented issues, particularly in hosting the FastAPI server using Render. Delays in server response time and occasional timeout errors occurred when the

proposed application attempted to request predictions, requiring adjustments to server settings. In addition, integrating Firebase Authentication and Firestore introduced difficulties in managing user roles, data structure consistency, and real-time updates. Ensuring that user data, such as personal health information, was stored accurately and retrieved efficiently required additional testing and debugging.

Finally, challenges were faced in the mobile development environment itself. Setting up Android Studio with the required Flutter and Dart dependencies took time, and the emulator sometimes failed to connect to external APIs or the local network, slowing down testing. Debugging UI behaviour and ensuring smooth navigation across pages also required significant effort. These challenges collectively shaped the development process and improved the robustness of the final application.

5.7 Concluding Remark

In conclusion, this chapter described the implementation and operation of the proposed application, including the system setup, configuration, and key functionalities. The proposed application was successfully developed and integrated with the required components, allowing features such as meal logging, nutrient tracking, and recommendation generation to function properly. Although some challenges were encountered during implementation, they were resolved to ensure the proposed application runs smoothly. Overall, the implementation demonstrates that the proposed application is workable and ready for evaluation.

Chapter 6

System Evaluation and Discussion

6.1 System Testing and Performance Metrics

6.1.1 System Testing

Functional testing was used to test the proposed application. Function testing is a type of software testing that validates the system against the functional requirements or specifications [43]. Its primary purpose is to test each function of the software application by providing appropriate input and verifying the output against the expected results. It ignores the internal source code and focuses on checking user interfaces, APIs, database interactions, and client/server communications to ensure they operate as intended.

6.1.2 System Performance Definition

The system performance was defined based on key targets that the recommendation model and proposed application must achieve. These targets served as benchmarks to evaluate whether the proposed application operated efficiently and met the expected quality levels. The overall performance targets include:

Prediction Accuracy

The recommendation model was expected to achieve a high level of accuracy when predicting nutritional values. Accuracy was evaluated using metrics such as mean absolute error (MAE), root mean squared error (RMSE), and R-squared (R^2):

- MAE measures the average difference between predicted and actual recommended nutrient values [44].
- RMSE highlights larger errors by giving more weight to predictions that are far from the actual values [45].
- R^2 indicates the proportion of variance in the actual recommended nutrient values explained by the model [46].

The performance targets were set as follows:

- **R²:** 0.8, indicating a very good regression model performance as stated by [47].
- **MAE and RMSE:** The targets were set at 26 g for protein, 50 g for carbohydrate, and 18 g for fats. These values represent approximately 20% of the average recommended intakes in the dataset, ensuring that prediction errors remained within acceptable bounds for nutrient recommendation [48] [49].

Response Time

The proposed application should generate recommendations within a short processing time, ensuring a smooth and responsive user experience.

Reliability

The proposed application must consistently produce valid outputs and handle errors during data processing and prediction.

6.1.3 Verification Plan

The verification plan was prepared to ensure that the proposed application and recommendation model were functioning correctly and meeting the defined requirements. Verification activities were carried out to confirm that each feature behaves as expected under various conditions.

Input Verification

Different types of inputs are tested to ensure they are handled correctly by the proposed application:

- **Valid inputs**

Inputs that follow correct formats are provided, such as correct user information, meal details, and nutrient values.

- **Invalid inputs**

Empty fields, incorrect data types, and out-of-range values are entered to ensure they are rejected appropriately.

- **Boundary inputs**

Extreme but allowable values, such as minimum and maximum limits for age, calories, or nutrient levels, are tested.

Functional Verification

Each main feature of the proposed application was checked to verify that correct behaviour was achieved such as:

- **User authentication**

Registration, login, logout, and incorrect credential handling were tested to ensure proper access control was maintained.

- **Profile management**

The adding, updating, and retrieving of personal and health information were verified.

- **Meal logging**

The functions for logging, viewing, and saving meal records were tested to confirm correct processing and retrieving data from the database.

- **Recommendation generation**

The success transmission of user data to the model and the return of valid recommendations were verified.

- **Error handling**

The display of appropriate error messages during failures such as missing inputs or connection issues was confirmed.

Recommendation Model Results Verification

The predicted nutrient recommendations were compared against expected nutrient patterns obtained from published research [50][51][52][53]. These patterns, which define whether protein, fats, or carbohydrates should be higher or lower for each health condition, were used to verify that the model generated medically consistent outputs.

- **For obesity**

According to [50], higher protein, lower fats, and lower carbohydrates were expected because higher protein may help in weight loss by increasing satiety to reduce appetite and boosting metabolism to burn more calories. On the other hand, low-fat diets may help with weight loss by reducing overall calorie intake, as fat contributes more than half of the calories per gram compared to protein or carbohydrates. Reducing carbohydrates lowers insulin, which decreases fat storage and increases satiety, leading to reduced hunger and overall food intake.

- **For hypertension**

According to [51], higher protein, lower fats, and lower carbohydrates were expected because high-protein diets can help lower blood pressure through mechanisms like amino acids promoting blood vessel dilation. On the other hand, low-carbohydrate diets can help lower blood pressure by reducing insulin levels, which decreases the retention of sodium of the body. Low-fat diets can help lower blood pressure by reducing overall calorie intake, leading to weight loss, which is a key factor in lowering blood pressure.

- **For diabetes**

According to [52], higher protein and fats with lower carbohydrates were expected because high-protein and high-fat diets can help lower blood sugar by slowing the digestion of carbohydrates, which prevents large glucose spikes after a meal. On the other hand, low-carbohydrate diets help lower blood sugar by reducing the primary source of glucose your body uses for energy.

- **For heart disease**

According to [53], lower fats were expected because low-fat diets are good for heart disease by lowering cholesterol, especially low-density lipoprotein (LDL) cholesterol, which reduces the buildup of plaque in arteries and lowers the risk of heart attack and stroke.

6.2 Testing Setup and Result

6.2.1 Recommendation Model Result Testing

The recommendation model was tested by comparing its predicted nutrient values against the expected nutrient output as described in Section 6.1.3. For each health condition, the nutrient intake values of the previous day and the model-generated recommended values were examined to verify whether the model followed the expected “higher” or “lower” pattern as shown in Table 6.2.1.1. The nutrient intake values of the previous day and the corresponding recommended value for obesity, hypertension, diabetes, and heart disease are shown in Figure 6.2.1.1 to 6.2.1.4.

Table 6.2.1.1 Model Result Testing Based on Health Condition Requirements

Health Condition	Expected Output	Actual Output	Result
Obesity	Higher protein Lower fats Lower carbohydrates	108 g → 112 g 71 g → 40 g 474 g → 454 g	Pass
Hypertension	Higher protein Lower fats Lower carbohydrates	108 g → 112 g 71 g → 40 g 474 g → 454 g	Pass
Diabetes	Higher protein Higher fats Lower carbohydrates	97 g → 100 g 31 g → 35 g 468 g → 449 g	Pass
Heart Disease	Lower fats	71 g → 40 g	Pass

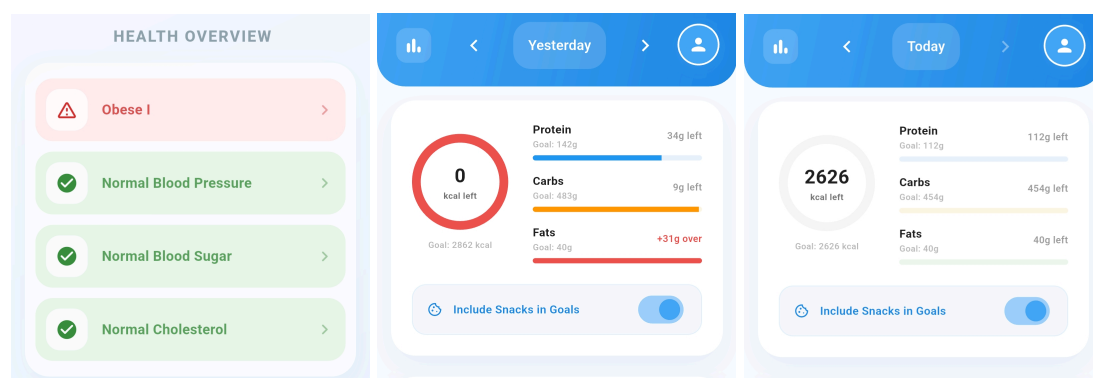


Figure 6.2.1.1 Nutrients Recommendation for Obesity

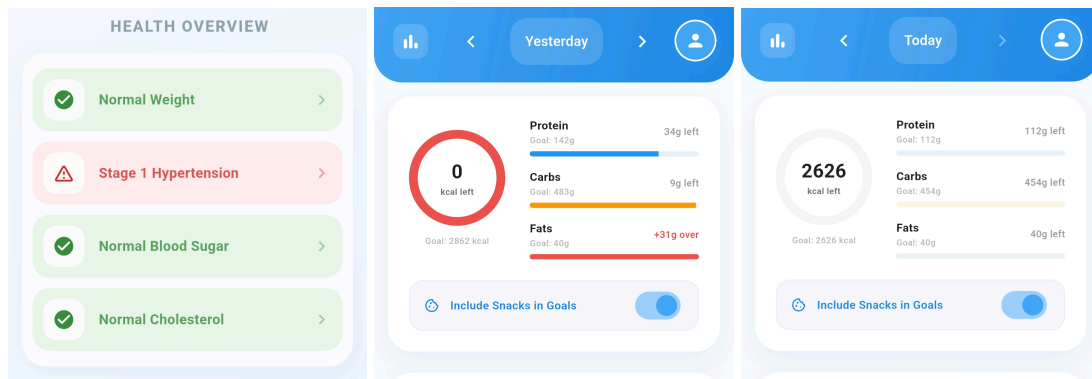


Figure 6.2.1.2 Nutrients Recommendation for Hypertension

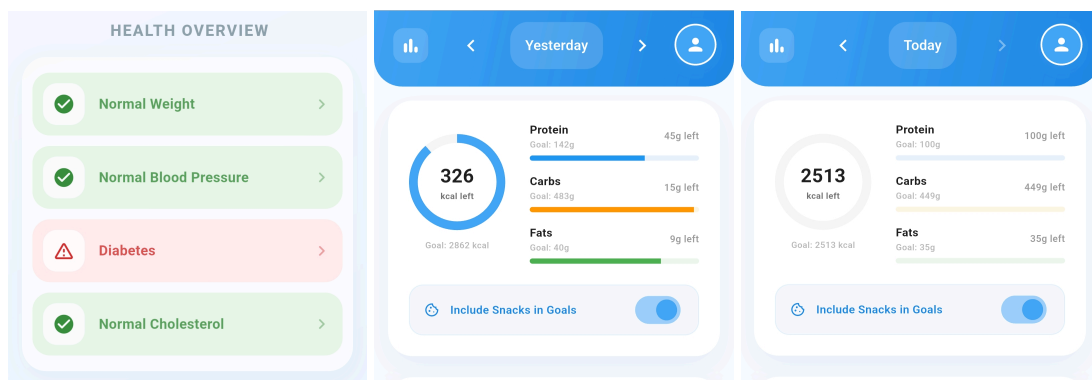


Figure 6.2.1.3 Nutrients Recommendation for Diabetes

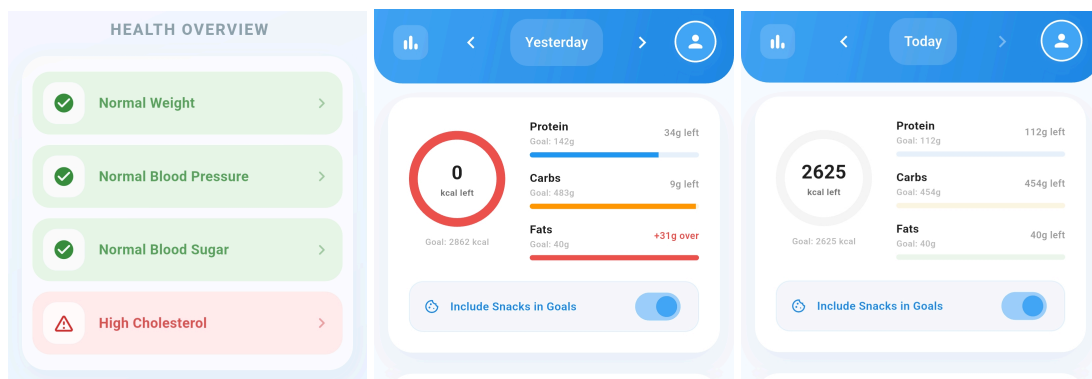


Figure 6.2.1.4 Nutrients Recommendation for Heart Disease

6.2.2 Functional Testing of Proposed Application Modules

This section presents the functional testing conducted on each module of the proposed application. The purpose of this testing was to verify that all functions operate according to the specified requirements and that user interactions produce the expected results. Table 6.2.2.1 shows one of the examples for Functional Test Cases for the Login Module.

Login Module

Table 6.2.2.1 Functional Test Cases for the Login Module

Test Case ID	Test Case Description	Expected Result	Actual Result	Status
TC01	User taps the “Sign In” button when the email address or password fields are empty.	The proposed application displays the error messages “Please enter your email” or “Please enter your password”.	The error messages are displayed.	Pass
TC02	User taps the “Sign In” button when the email address entered is in an invalid format.	The proposed application displays the error message “Enter a valid email”.	The error message is displayed.	Pass
TC03	User taps the “Sign In” button when the email address entered is not registered or has expired.	The proposed application displays the error message “The supplied credential is incorrect, malformed or has expired”.	The error message is displayed.	Pass
TC04	User taps the “Sign In” button when the password entered is less than 6 characters.	The proposed application displays the error message “Password must be 6+ characters”.	The error message is displayed.	Pass
TC05	User taps the “Sign In” button when the password entered is incorrect.	The proposed application displays the error message “The supplied credential is incorrect, malformed or has expired”.	The error message is displayed.	Pass
TC06	User taps the eye icon in the password field.	The proposed application hides or shows the password.	The password is hidden or shown.	Pass
TC07	User taps the “Sign In” button when the entered email address and password	The proposed application displays an appropriate error	The appropriate error message is displayed.	Pass

	are correct, but the account is banned, pending, or declined.	message indicating the account status.		
TC08	User taps the “Sign In” button when the entered email address and password are correct and the account is active.	The proposed application displays the User or Admin Home page. If the user is new user, the proposed application displays the Input and Update Profile page.	The User or Admin Home page is displayed. The Input and Update Profile page is displayed for a new user.	Pass
TC09	User taps the “Forgot Password?” button.	The proposed application displays the Password Recovery page.	The Password Recovery page is displayed.	Pass
TC10	User taps the “Continue with Google” button when selected email address is not registered.	The proposed application displays the error message “Account does not exist”.	The error message is displayed.	Pass
TC11	User taps the “Continue with Google” button when the selected email address is registered.	The proposed application displays the User or Admin Home page.	The User or Admin Home page is displayed.	Pass
TC12	User taps the “Sign up” button.	The proposed application displays the Registration page.	The Registration page is displayed.	Pass

The rest of the functional test cases are provided in Appendix A, Table A.4.1 to A.4.22. Each module was tested using predefined test cases to ensure correctness, reliability, and proper behaviour.

6.3 Project Challenges

During development, there were problems with the Android Emulator not working well with macOS. This caused frequent “System UI” crashes, interrupting debugging. Because of this, testing had to be done on a real device, which was slower and less convenient than using the emulator. On the other hand, it was difficult to find reliable meal dataset. Many available datasets did not include local foods or detailed nutrient information causing the available meals in the proposed application is reduced.

The number of requests could be made by Google AI APIs is limited for free version. This caused the functions of chatbot, and nutrition label extraction are limited as well. As there is no API provided for the meal database used in this project, the meals data were entered manually which took a lot of time and effort. With a limited project timeline, it was challenging to balance data entry and proposed application development. Priorities had to be adjusted to make sure the main functions were completed first.

6.4 Objectives Evaluation

This project is evaluated based on the objectives stated in Section 1.2:

- i. To develop a mobile application that can record daily meals of the users and track their nutrient intake.**

User registration and authentication were implemented successfully via manual input and Google account selection to ensure a secure and personalised experience which each user was allowed to create and verify their account. Once logged in, users can record meal details successfully by searching for available meals or by adding custom entries. The proposed application analysed the meal records to generate a breakdown of key nutritional values. Nutrient progress tracking function is included successfully, enabling users to monitor their dietary habits and measure their progress toward health goals over time. Additional functions such as custom meal creation, personalised nutritional advice using Google AI, community forum, and reminders were included successfully for the users. On the other hand, management of user accounts, admin registration, meals, and reported posts and comments are included successfully for the admins.

ii. To integrate a model to recommend suitable meals to users based on their health conditions.

The health conditions such as hypertension, diabetes, high cholesterol, or obesity were considered in the meal recommendation function to ensure that recommendations are both safe and beneficial. The model was trained successfully by analysing the health condition and nutrient intake history of the users for generating personalised meal options designed to encourage healthier eating habits. This function helped users make informed dietary decisions that align with their nutritional requirements and health conditions.

6.5 Concluding Remark

In conclusion, this chapter evaluated the performance and functionality of the proposed application through testing and analysis. The results showed that the proposed application can operate correctly and meet the intended objectives, including meal logging, nutrient tracking, and personalised meal recommendations. Despite several challenges encountered during development, appropriate solutions were implemented to ensure the reliability of the proposed application. Overall, the findings indicate that the proposed application is effective and capable of supporting users in monitoring their nutritional intake and improving their dietary habits.

Chapter 7

Conclusion and Recommendation

7.1 Conclusion

In conclusion, this project successfully developed a mobile application for meal logging and nutrient tracking with personalised meal recommendations. The proposed application enables users to record their daily meals, monitor nutrient intake, and receive recommendations based on their health conditions.

All the objectives of this project were achieved, including the development of core features such as user authentication, meal logging, nutrient tracking, and recommendation generation. The proposed application was implemented using Flutter, Firebase, and FastAPI, and the selected model which is Linear Regression demonstrated good performance in generating nutrient recommendations. Through testing and evaluation, the proposed application was found to function correctly and effectively support users in managing their dietary habits.

Overall, the proposed application provided a useful solution for individuals who want to monitor their nutrition and improve their eating habits. It also addresses the limitations of existing applications by incorporating health conditions into the recommendation process, making the system more personalised and practical for real-world use.

7.2 Recommendation

For future improvements, several enhancements can be considered to further improve the functionality and usability of the proposed application. First, additional features such as nutrient progress tracking over longer periods and more detailed visualisation can be included to provide better insights to users.

Second, the proposed application can be enhanced by supporting automatic meal recognition using image processing, allowing users to log meals more conveniently without manual input. Integration with wearable devices or fitness trackers can also be implemented to provide a more comprehensive view of health data of the user.

Lastly, expanding the database to include more local and international food items will improve the accuracy and usability of the proposed application. The recommendation model can also be further improved by using larger and more diverse datasets to increase prediction accuracy.

REFERENCES

- [1] “Spring - The Best Meal Kits & Recipe Boxes UAE,” *Springfeeling.com*, 2024. <https://springfeeling.com/blog/the-importance-of-regular-meals>
- [2] F. J. Pendergast, K. M. Livingstone, A. Worsley, and S. A. McNaughton, “Correlates of meal skipping in young adults: a systematic review,” *International Journal of Behavioral Nutrition and Physical Activity*, vol. 13, Dec. 2016, doi: <https://doi.org/10.1186/s12966-016-0451-1>.
- [3] C. Kanazawa, Y. Shimba, S. Toyonaga, F. Nakamura, and T. Hosaka, “Effects of skipping breakfast, lunch or dinner on subsequent postprandial blood glucose levels among healthy young adults,” *Nutrition & Metabolism*, vol. 22, Jul. 2025, doi: <https://doi.org/10.1186/s12986-025-00975-4>.
- [4] Centre for Health Protection, “Centre for Health Protection, Department of Health - The Food Pyramid – A Guide to a Balanced Diet,” *Centre for Health Protection*, Jan. 15, 2024. <https://www.chp.gov.hk/en/static/90017.html>
- [5] J. Fletcher, “6 essential nutrients: Sources and why you need them,” *www.medicalnewstoday.com*, Aug. 22, 2019. <https://www.medicalnewstoday.com/articles/326132>
- [6] H. Tang and M. Yang, “Nutritional Assessment in Patients with Chronic Diseases: Tools, Challenges, and Future Directions,” *Nutrients*, vol. 15, no. 22, pp. 4794–4794, Nov. 2023, doi: <https://doi.org/10.3390/nu15224794>.
- [7] J. S. Bland, “Age as a Modifiable Risk Factor for Chronic Disease,” *Integrative Medicine: A Clinician’s Journal*, vol. 17, no. 4, pp. 16–19, Aug. 2018, Available: <https://pmc.ncbi.nlm.nih.gov/articles/PMC6469457/>
- [8] A. M. H. Chen, J. A. Draime, S. Berman, J. Gardner, Z. Krauss, and J. Martinez, “Food as medicine? Exploring the impact of providing healthy foods on adherence and clinical and economic outcomes,” *Exploratory Research in Clinical and Social Pharmacy*, vol. 5, p. 100129, Mar. 2022, doi: <https://doi.org/10.1016/j.rcsop.2022.100129>.
- [9] M. Verma, R. Hontecillas, N. Tubau-Juni, V. Abedi, and J. Bassaganya-Riera, “Challenges in Personalized Nutrition and Health,” *Frontiers in Nutrition*, vol. 5, Nov. 2018, doi: <https://doi.org/10.3389/fnut.2018.00117>.

- [10] A. Firoozzare, F. Boccia, N. Yousefian, S. Ghazanfari, and S. Pakook, “Understanding the role of awareness and trust in consumer purchase decisions for healthy food and products,” *Food quality and preference*, vol. 121, Jul. 2024, doi: <https://doi.org/10.1016/j.foodqual.2024.105275>.
- [11] “Calorie Tracker & BMR Calculator to Reach Your Goals | MyFitnessPal,” *www.myfitnesspal.com*. <https://www.myfitnesspal.com/en>
- [12] “Healthy Weight Loss & Eating: Lose Weight Fast with YAZIO,” *YAZIO*. <https://www.yazio.com/en>
- [13] “HealthifyMe - Track Nutrition, Plan Your Diet, Get Fitter, Workout from Home,” *HealthifyMe*. <https://www.healthifyme.com/my/>
- [14] “EATR - Your Culinary Companion,” *EATR - Your Culinary Companion*. <https://eatr.com>
- [15] “Foodvisor App | Calorie Counter - Eat Healthy & Lose Weight,” *foodvisor.io*. <https://www.foodvisor.io/en/>
- [16] “Lifesum Health App – Get Healthy & Lose Weight - Lifesum,” *lifesum.com*. <https://lifesum.com>
- [17] “Random Forest Regression in Python,” *GeeksforGeeks*, Nov. 13, 2025. <https://www.geeksforgeeks.org/machine-learning/random-forest-regression-in-python/>
- [18] N. R. Ferdous, N. S. Rahman, N. S. Mia, and N. H. Rahman, “A machine learning-based model for crop recommendation using Agro-climatic and soil nutrient parameters (Agrismart),” *International Journal of Science and Research Archive*, vol. 15, no. 3, pp. 1080–1089, Jun. 2025, doi: <https://doi.org/10.30574/ijrsra.2025.15.3.1713>.
- [19] Muhammad Ilham Hafizha and Z. K. A. Baizal, “Diet and Physical Exercise Recommendation System Using a Combination of K-Means and Random Forest,” *Ind. Journal on Computing*, vol. 9, no. 2, Jun. 2024, doi: <https://doi.org/10.34818/indojc.2024.9.2.959>.
- [20] T. Subramanya, D. Harutyunyan, and R. Riggio, “Machine learning-driven service function chain placement and scaling in MEC-enabled 5G networks,” *Computer Networks*, vol. 166, p. 106980, Nov. 2019, doi: <https://doi.org/10.1016/j.comnet.2019.106980>.

- [21] J. Guo , “Improving Movie Recommendations Using Multilayer Perceptron Model and Sparse Feature Integration,” *The Frontiers of Society, Science and Technology*, vol. 6, no. 9, pp. 35–40, 2024, doi: <https://doi.org/10.25236/fsst.2024.060906>.
- [22] IBM, “Linear Regression,” *Ibm.com*. <https://www.ibm.com/think/topics/linear-regression>
- [23] P. A, S. Chakraborty, A. Kumar, and O. R. Pooniwala, “Intelligent Crop Recommendation System using Machine Learning,” *IEEE Xplore*, Apr. 01, 2021. <https://ieeexplore.ieee.org/abstract/document/9418375>
- [24] V. B, S. R, and L. E, “Forecasting Obesity and Customized Nutrition Planning using KNN and Linear Regression Models for Higher Accuracy,” *2025 8th International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 884–889, Jul. 2025, doi: <https://doi.org/10.1109/iccmc65190.2025.11140798>.
- [25] S. Laoyan, “What is agile methodology? (A beginner’s guide),” *Asana*, Feb. 20, 2025. <https://asana.com/resources/agile-methodology>
- [26] Ziya, “Personalized Medical Diet Recommendations Dataset,” *Kaggle.com*, 2025. <https://www.kaggle.com/datasets/ziya07/personalized-medical-diet-recommendations-dataset/data> (accessed Dec. 04, 2025).
- [27] S. Mousavirad, R. AlGhamdi, G. G. Tejani, S. K. Sharma, and S. J. Mousavirad, “Federated learning-based multimodal approach for early detection and personalized care in cardiac disease,” *Frontiers in Physiology*, vol. 16, Apr. 2025, doi: <https://doi.org/10.3389/fphys.2025.1563185>.
- [28] A. Gholamy, V. Kreinovich, and O. Kosheleva, “Why 70/30 or 80/20 Relation between Training and Testing Sets: a Pedagogical Explanation.” Available: <https://www.cs.utep.edu/vladik/2018/tr18-09.pdf>
- [29] J. Bergstra and Y. Bengio, “Random Search for Hyper-Parameter Optimization,” *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012, Available: <https://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf>
- [30] GeeksforGeeks, “Layers in Artificial Neural Networks (ANN),” *GeeksforGeeks*, Dec. 17, 2025. <https://www.geeksforgeeks.org/deep-learning/layers-in-artificial-neural-networks-ann/>
- [31] “How many epochs are enough to train your model,” *Nebius.com*, Jul. 02, 2025. <https://nebius.com/blog/posts/factors-influencing-epoch-count#dataset-size-and-quality>

- [32] “WELCOME TO MALAYSIAN FOOD COMPOSITION DATABASE (MYFCD),” *myfcd.moh.gov.my*. <https://myfcd.moh.gov.my>
- [33] Ministry of Health Malaysia, “Clinical Practice Guidelines Management of Obesity 2nd Edition,” 2023. Available: [https://www.moh.gov.my/moh/resources/Penerbitan/CPG/Endocrine/CPG_Management_of_Obesity_\(Second_Edition\)_2023.pdf](https://www.moh.gov.my/moh/resources/Penerbitan/CPG/Endocrine/CPG_Management_of_Obesity_(Second_Edition)_2023.pdf)
- [34] American Heart Association, “Understanding Blood Pressure Readings,” *American Heart Association*, May 17, 2024. <https://www.heart.org/en/health-topics/high-blood-pressure/understanding-blood-pressure-readings>
- [35] Cleveland Clinic, “Blood Glucose (Sugar) Test,” *Cleveland Clinic*, 2022. <https://my.clevelandclinic.org/health/diagnostics/12363-blood-glucose-test>
- [36] C. Ee, “Cholesterol Levels: By Age, Gender, LDL, HDL & More,” *Homage Malaysia*, Aug. 11, 2022. <https://www.homage.com.my/health/cholesterol-level/>
- [37] “The Importance of Hydration | Pantai Hospitals,” *Pantai.com.my*, Nov. 23, 2022. <https://www.pantai.com.my/health-pulse/hydration>
- [38] K. Murakami *et al.*, “Characterisation of breakfast, lunch, dinner and snacks in the Japanese context: an exploratory cross-sectional analysis,” *Public Health Nutrition*, vol. 25, no. 3, pp. 689–701, Nov. 2020, doi: <https://doi.org/10.1017/s1368980020004310>.
- [39] “Measurement Equivalents,” *Exploratorium*. <https://www.exploratorium.edu/food/measurements>
- [40] American Heart Association, “Protein and Heart Health,” *www.heart.org*. <https://www.heart.org/en/healthy-living/healthy-eating/eat-smart/nutrition-basics/protein-and-heart-health>
- [41] P. Dyson, “Low Carb Diets,” *bda.uk.com*, Oct. 2019. <https://www.bda.uk.com/static/015e7600-28a8-4fc6-8758567f538dfc29/lowcarbdietsfoodfactsheet.pdf>
- [42] “Fats.” Available: https://www.moh.gov.my/images/perkhidmatan-utama/info-kesihatan/pemakanan/3_chat.pdf
- [43] P. Powell and I. Smalley, “What is functional testing?,” *IBM*. <https://www.ibm.com/think/topics/functional-testing>

- [44] “Mean Absolute Error (MAE),” *DagsHub*. <https://dagshub.com/glossary/mean-absolute-error-mae/> (accessed Dec. 04, 2025).
- [45] E. Kosourova, “RMSE Explained: A Guide to Regression Prediction Accuracy,” *Datacamp.com*, Jun. 30, 2025. <https://www.datacamp.com/tutorial/rmse>
- [46] S. Taylor, “R-Squared,” *Corporate Finance Institute*, 2025. <https://corporatefinanceinstitute.com/resources/data-science/r-squared/>
- [47] D. Chicco, M. J. Warrens, and G. Jurman, “The Coefficient of Determination R-squared Is More Informative than SMAPE, MAE, MAPE, MSE and RMSE in Regression Analysis Evaluation,” *PeerJ Computer Science*, vol. 7, no. 5, p. e623, 2021, doi: <https://doi.org/10.7717/peerj-cs.623>.
- [48] M. Taylor, “Mean Absolute Percentage Error,” *www.vexpower.com*, Jan. 19, 2022. <https://www.vexpower.com/brief/mean-absolute-percentage-error>
- [49] A. Choudhary, “All About: Mean Squared Error (MSE), Mean Absolute Error (MAE) and RMSE,” *Medium*, Apr. 19, 2024. <https://alok05.medium.com/performance-metrics-or-loss-function-in-machine-learning-for-regression-7fae992577f0> (accessed Dec. 04, 2025).
- [50] V. R. Johnson *et al.*, “Food as Medicine for Obesity Treatment and Management,” *Clinical Therapeutics*, vol. 44, no. 5, pp. 671–681, May 2022, doi: <https://doi.org/10.1016/j.clinthera.2022.05.001>.
- [51] L. A. Bazzano, T. Green, T. N. Harrison, and K. Reynolds, “Dietary Approaches to Prevent Hypertension,” *Current Hypertension Reports*, vol. 15, no. 6, pp. 694–702, Oct. 2018, doi: <https://doi.org/10.1007/s11906-013-0390-z>.
- [52] E. Koloverou and D. B. Panagiotakos, “Macronutrient Composition and Management of Non-Insulin-Dependent Diabetes Mellitus (NIDDM): A New Paradigm for Individualized Nutritional Therapy in Diabetes Patients,” *The Review of Diabetic Studies*, vol. 13, no. 1, pp. 6–16, 2016, doi: <https://doi.org/10.1900/rds.2016.13.6>.

- [53] Y. Ma *et al.*, “Dietary Macronutrient Intake and Cardiovascular Disease Risk and Mortality: A Systematic Review and Dose-Response Meta-Analysis of Prospective Cohort Studies,” *Nutrients*, vol. 16, no. 1, p. 152, Jan. 2024, doi: <https://doi.org/10.3390/nu16010152>.

APPENDIX A

A.1 Use Case Description

Login Account

Table A.1.1 Use Case Description for Login Account

Use Case Name: Login Account	ID: 2	Importance Level: High
Primary Actor: Existing User, Existing Admin	Use Case Type: Detail, Essential	
Stakeholders and Interests: Existing User / Existing Admin – wants to log in to access the proposed application.		
Brief Description: This use case describes how an existing user or admin logs into the proposed application using their registered credentials to access personal data and functionalities.		
Trigger: The user or admin opens the proposed application and wants to login. Type: External		
Relationships: Association: Existing User, Existing Admin		
Normal Flow of Events: 1. The existing user or admin opens the application and proceeds to login. 2. The application displays the login page. 3. If the existing user or admin wants to login manually, the S-1: Login Manually is performed. 4. If the existing user or admin wants to login using Google, the existing user or admin selects their email account. 5. The application checks whether the login details match a record in the user database. 6. The application grants access to the user or admin account. 7. The existing user or admin successfully logs into the application.		
SubFlows: S-1: Login Manually 1. The existing user or admin enters their email address and password. 2. The application validates the entered email and password.		
Alternate/Exception Flows: S-1: Login Manually 1a. The email or password field is left empty. The application displays an error message prompting the user or admin to fill in both fields.		

1b. The email format is wrong. The application displays an error message prompting the user or admin to fill in the email address again.
2a. The entered email or password is invalid. The application displays an error message indicating invalid credentials and prompts the user or admin to try again.
5a. The login details do not match any record in the user database. The application displays an error message indicating that the account does not exist.

Input Profile

Table A.1.2 Use Case Description for Input Profile

Use Case Name: Input Profile		ID: 3	Importance Level: High
Primary Actor: Existing User		Use Case Type: Detail, Essential	
Stakeholders and Interests: Existing User – wants to enter accurate personal and health details to receive appropriate meal recommendations.			
Brief Description: This use case describes how the user enters personal and health information into the proposed application.			
Trigger: The user logs into the proposed application for the first time. Type: External			
Relationships: Association: Existing User			
Normal Flow of Events: 1. The existing user logs into the proposed application. 2. The application displays the “Input Profile” page. 3. The existing user enters personal and health information. 4. The application validates all inputs for completeness and correct data formats. 5. The application stores all entered information and calculated BMI in the user database. 6. The application displays a confirmation message indicating successful data submission.			
SubFlows: Not applicable.			
Alternate/Exception Flows: 4a. One or more fields are left empty or contain invalid values. The application highlights the missing or incorrect fields and prompts the user to re-enter the information.			

View Profile

Table A.1.3 Use Case Description for View Profile

Use Case Name: View Profile		ID: 4	Importance Level: High
Primary Actor: Existing User		Use Case Type: Detail, Essential	
Stakeholders and Interests: Existing User – wants to view their personal details or health information.			
Brief Description: This use case describes how the user views personal or health information in the proposed application.			
Trigger: The user taps the profile button on the top right corner of the screen. Type: External			
Relationships: Association: Existing User Extend: Update Profile			
Normal Flow of Events: 1. The existing user taps the profile button. 2. The application retrieves and displays personal and health information. 3. If the existing user wants to change their personal or health information, execute the Update Profile use case. 4. If the existing user wants to log out from the account the application logs the existing user out and navigates to the welcome page.			
SubFlows: Not applicable.		Alternate/Exception Flows: Not applicable.	

Update Profile

Table A.1.4 Use Case Description for Update Profile

Use Case Name: Update Profile		ID: 5	Importance Level: Medium
Primary Actor: Existing User		Use Case Type: Detail, Essential	
Stakeholders and Interests: Existing User – wants to modify personal details or health information to keep their profile accurate.			
Brief Description: This use case describes how the user updates personal or health information in the proposed application.			
Trigger: The user taps the “Update Data” button in profile page. Type: External			
Relationships: Association: Existing User			
Normal Flow of Events: 1. The existing user taps the “Update Data” button. 2. The application retrieves and displays the personal and health information of the user. 3. The existing user edits one or more fields. 4. The application validates the updated information for completeness and correctness. 5. The application updates and displays the calculated BMI. 6. The application saves all updated information in the user database. 7. The application displays a confirmation message indicating that the profile has been successfully updated.			
SubFlows: Not applicable.			
Alternate/Exception Flows: 4a. One or more fields are left empty or contain invalid values. The application highlights the missing or incorrect fields and prompts the user to re-enter the information.			

View Daily Nutrient Intake Recommendations

Table A.1.5 Use Case Description for View Daily Nutrient Intake Recommendations

Use Case Name: View Daily Nutrient Intake Recommendations		ID: 6	Importance Level: High
Primary Actor: Existing User		Use Case Type: Detail, Essential	
Stakeholders and Interests: Existing User – wants to view nutrient intake recommendation for the selected date.			
Brief Description: This use case describes how the proposed application displays the daily nutrient intake recommendations of the user for a specific date.			
Trigger: The user navigates to the diary page or selects a different date from the diary interface. Type: External			
Relationships: Association: Existing User			
Normal Flow of Events: 1. The existing user navigates to the diary page or changes the diary date. 2. The application retrieves the recommendations from the database. 3. The application displays the recommendation as daily nutrient recommendation. 4. The existing user reviews the nutrient recommendations.			
SubFlows: Not applicable.			
Alternate/Exception Flows: 2a. No recommendation for the selected date. The application uses the default recommendation values.			

View Diary

Table A.1.6 Use Case Description for View Diary

Use Case Name: View Diary	ID: 7	Importance Level: High
Primary Actor: Existing User	Use Case Type: Detail, Essential	
Stakeholders and Interests: Existing User – wants to view the diary that displays daily logged meals and nutrient information to monitor daily progress.		
Brief Description: This use case describes how an existing user views their daily meal diary in the proposed application. The diary displays all meals logged for the selected date and compares total nutrient intake with the daily recommended values. The user can also switch between different dates to review previous meal logs.		
Trigger: The user selects the “Daily” option from the navigation bar. Type: External		
Relationships: Association: Existing User Include: View Daily Nutrient Intake Recommendations Extend: View Logged Meals, Track Nutrient Progress		
Normal Flow of Events: 1. The existing user selects the “Diary” option from the navigation bar. 2. The application checks whether the existing user has previously logged any meals in the diary. 3. The application checks whether all meals from the previous day have been logged. 4. The application executes View Daily Nutrient Intake Recommendations use case to display the recommended daily nutrient intake. 5. The application displays the water intake progress, nutrient intake progress, and meals logged for each meal period. 6. The existing user reviews the diary contents for the selected date. 7. If the existing user wants to view another date, the S-1: View Another Date is performed. 8. If the existing user wants to exclude snacks for the selected date, the application disables the “Add Food” button for snacks and moves all the logged snacks to the dinner category. 9. If the existing user wants to log their water intake, the S-2: Log Water Intake is performed.		

10. If the existing user wants to view all meals logged for each meal period, execute the View Logged Meals use case. 11. If the existing user wants to track nutrient progress, execute the Track Nutrient Progress use case.
SubFlows: S-1: View Another Date 1. The existing user selects a different date from the calendar or uses the “<” and “>” buttons. 2. The application updates the diary page to displays the meals and nutrient summary for the selected date. S-2: Log Water Intake 1. The existing user enters water amount or selects a predefined amount card. 2. The application validates the entered water amount. 3. The application updates the water intake progress.
Alternate/Exception Flows: 2a. The existing user has not logged any meals before. The application displays an alert message with option to log meals for the previous day or start with the default nutrient recommendations. If the existing user chooses to log previous day meals, the application displays the previous day dairy page. If the existing user chooses to start with the default nutrient recommendations, the application displays the default nutrient recommendations. 3a. At least one meal from the previous day has not been logged. The application displays an alert message with option to log the missing meals now or later. If the existing user chooses to log the missing meals now, the application displays the previous day diary page. If the existing user chooses to log the missing meals later, the application uses the default recommendation values for any nutrient recommendation that is lower than the default value. S-2: Log Water Intake 2a. The entered water amount is invalid. The application displays error message.

View Logged Meals

Table A.1.7 Use Case Description for View Logged Meals

Use Case Name: View Logged Meals	ID: 8	Importance Level: High
Primary Actor: Existing User	Use Case Type: Detail, Essential	
Stakeholders and Interests: Existing User – wants to view logged meals for a specific meal period.		
Brief Description: This use case describes how the application displays the meals logged for a selected meal period on a selected date.		
Trigger: The user selects a meal period from the diary page. Type: External		
Relationships: Association: Existing User Extend: Log Meals		
Normal Flow of Events: 1. The existing user selects a meal period on the diary page. 2. The application retrieves all meals logged for the selected date and meal period from the database. 3. The application displays the list of logged meals with the “Add Food” button. 4. If the existing user selects a meal from the list, the S-1: View Meal Details is performed. 5. If the existing user wants to add a meal, executes the Log Meals use case and the S-2: Retrieve Recommendation is performed. 6. If the existing user wants to delete a meal, the application removes the selected meal record and updates the list of logged meals.		
SubFlows: S-1: View Meal Details 1. The application retrieves and displays the meal details from the meal database. 2. If the existing user taps the favourite button, the application updates the meal in the favourited meal list and database. S-2: Retrieve Recommendation 1. The application retrieves all logged meals, personal information, and health information from the database. 2. The application sends the retrieved data as input to the recommendation model.		

3. The recommendation model processes the input data and predicts the recommended nutrient intake for the next day. 4. The application stores the results into the database.
Alternate/Exception Flows: 2a. No meals have been logged for the selected meal period. The application displays the message “No meals logged yet” together with the “Add Food” button.

View Meal Recommendations

Table A.1.8 Use Case Description for View Meal Recommendations

Use Case Name: View Meal Recommendations	ID: 9	Importance Level: High
Primary Actor: Existing User	Use Case Type: Detail, Essential	
Stakeholders and Interests: Existing User – wants to receive personalised meal recommendation.		
Brief Description: This use case describes how the application displays personalised meal recommendations to the user based on the recommended nutrient intake.		
Trigger: The user opens the “Log Meal” page. Type: External		
Relationships: Association: Existing User		
Normal Flow of Events: 1. The existing user opens the “Log Meal” page. 2. The application retrieves meals which is below the recommended nutrient intake. 3. The application displays a list of recommended meals. 4. The existing user views the list of recommended meals.		
SubFlows: Not applicable.		
Alternate/Exception Flows: 2a. No meals are below the recommended nutrient intake. The application displays a message “No recommendations found”.		

Log Meals

Table A.1.9 Use Case Description for Log Meals

Use Case Name: Log Meals	ID: 10	Importance Level: High
Primary Actor: Existing User	Use Case Type: Detail, Essential	
Stakeholders and Interests: Existing User – wants to log meals to track daily nutrient intake.		
Brief Description: This use case describes how the user logs meals into the proposed application.		
Trigger: The user taps the “Add Food” button from the meal list on the diary or meals page. Type: External		
Relationships: Association: Existing User Include: View Meal Recommendations Extend: Add Custom Meals		
Normal Flow of Events: 1. The existing user taps the “Add Food” button. 2. The application displays the meal logging page. 3. If the existing user wants to reload the meal lists, the application refreshes and displays the updated meal lists. 4. If the existing user wants to search for a meal, the application retrieves and displays meals that match the keyword from the database. 5. If the existing user wants to view nutrient balance information. the application displays the base goal, carry-over, period target, consumed value, and remaining value for each nutrient. 6. If the existing user wants to view recommended meals, executes the View Meal Recommendations use case. 7. If the existing user wants to view favourited meals, the application displays the favourited meal list. 8. If the existing user wants to view custom meals, the S-1: View Custom Meal is performed. 9. If the existing user taps a meal to view its details, the S-2: View Meal Details is performed. 10. The existing user selects a meal from the list to log. 11. The application displays a dialog requesting the portion size and quantity. 12. The existing user enters or selects the portion size and quantity, then confirms the action.		

13. The application validates the portion size and quantity inputs.
14. The application checks whether the nutrient values exceed the recommended nutrient intake.
15. The application saves the logged meal and portion size in the database.
16. The application displays a confirmation message indicating that the meal was successfully logged.

SubFlows:

S-1: View Custom Meal

1. The application displays the custom meals list.
2. If the existing user wants to add a new custom meal,
executes the Add Custom Meals use case.

S-2: View Meal Details

3. The application retrieves and displays the selected meal details from the meal database.
4. If the existing user wants to favourite the meal,
the application updates the meal in the favourited meal list and database.

Alternate/Exception Flows:

- 13a. The portion size field is empty or contains invalid values. The application highlights the incorrect field and prompts the user to re-enter the information.
- 14a. The nutrient values exceed the recommended nutrient intake. The application displays an alert message asking whether the existing user still wants to log the meal.

Add Custom Meals

Table A.1.10 Use Case Description for Add Custom Meals

Use Case Name: Add Custom Meals	ID: 11	Importance Level: Medium
Primary Actor: Existing User	Use Case Type: Detail, Essential	
Stakeholders and Interests: Existing User – wants to add a custom meal that does not exist in the meal database.		
Brief Description: This use case describes how an existing user creates a custom meal.		
Trigger: The existing user taps the “Create New” button on the “Custom” tab. Type: External		
Relationships: Association: Existing User		
Normal Flow of Events: 1. The existing user taps on the “Create New” button on the “Custom” tab. 2. The application displays the custom meal page. 3. If the existing user chooses to add a meal manually, the S-1: Add Meal Manually is performed. 4. If the existing user chooses to add a meal using a recipe, the S-2: Add Meal Using Recipe is performed. 5. If the existing user wants to view their custom meals, the S-3: View Custom Meals is performed.		
SubFlows: S-1: Add Meal Manually 1. The application displays fields for basic information, nutritional values, and portion size. 2. If the existing user chooses to add another portion, the application adds another set of portion information fields. 3. The existing user enters all the required information. 4. If the existing user chooses to obtain nutritional values from a nutrition label by snapping a photo or selecting an image, the application extracts the text from the image and displays the nutritional values. 5. If the existing user clears the fields, the application empties all input fields. 6. The application validates all entered fields. 7. The application stores the meal information into the database. S-2: Add Meal Using Recipe		

1. The application displays fields for basic information, serving size, and ingredient list.
2. The existing user selects ingredients and specifies their serving size.
3. The application validates all fields and the ingredient list.
4. The application stores the meal information in the database.

S-3: View Custom Meals

1. The application displays the custom meals list.
2. If the existing user searches for a custom meal,
the application displays meals that match the keyword.
3. If the existing user deletes a custom meal,
the application removes the meal from the list.
4. If the existing user edits a custom meal,
the application displays the corresponding page to edit the meal information.
5. If the existing user taps a custom meal,
the application displays the meal details.

Alternate/Exception Flows:

S-1: Add Meal Manually

- 4a. The captured or selected image is blurred, or no nutrition label is detected. The application prompts the existing user to capture or select the image again.
- 6a. There are empty fields or invalid values. The application prompts the existing user to enter the field again.

S-2: Add Meal Using Recipe

- 3a. There are empty fields, invalid values, or the ingredient list is empty. The application prompts the existing user to complete the required fields.

Track Nutrient Progress

Table A.1.11 Use Case Description for Track Nutrient Progress

Use Case Name: Track Nutrient Process		ID: 12	Importance Level: High
Primary Actor: Existing User		Use Case Type: Detail, Essential	
Stakeholders and Interests: Existing User – wants to monitor nutrient intake and track progress over time.			
Brief Description: This use case describes how the user views their nutrient progress based on previously logged meals.			
Trigger: The user taps the chart button in the “Diary” page. Type: External			
Relationships: Association: Existing User			
Normal Flow of Events: 1. The existing user taps the chart button on the “Diary” page. 2. The application retrieves daily nutrients intake and recommendation values for the most recent 7 days. 3. The application displays the nutrient intake and recommendation values in a trend chart for each nutrient. 4. The application calculates and displays the average intake and average recommendation values. 5. The existing user views the trend chart and average values for each nutrient.			
SubFlows: Not applicable.		Alternate/Exception Flows: Not applicable.	

Get Personalised Nutrition Advice

Table A.1.12 Use Case Description for Get Personalised Nutrition Advice

Use Case Name: Get Personalised Nutrition Advice		ID: 13	Importance Level: Medium
Primary Actor: Existing User		Use Case Type: Detail, Essential	
Stakeholders and Interests: Existing User – wants to obtain personalised nutrition advice generated by an AI-based chatbot.			
Brief Description: This use case describes how the existing user gets the personalised nutrition advice through the AI chatbot.			
Trigger: The user selects the “Bot” option. Type: External			
Relationships: Association: Existing User			
Normal Flow of Events: 1. The existing user selects the “Bot” option. 2. The application retrieves the previous chat history from the database. 3. The existing user enters a nutrition-related query. 4. The application sends the query and relevant user data to the AI service. 5. The AI service generates a personalised nutrition response. 6. The application displays the response to the user. 7. The application stores chat interaction in the database. 8. If the existing user chooses to clear the chat history, the application clears all the chat history.			
SubFlows: Not applicable.		Alternate/Exception Flows: Not applicable.	

View Community Forum

Table A.1.13 Use Case Description for View Community Forum

Use Case Name: View Community Forum	ID: 14	Importance Level: Medium
Primary Actor: Existing User		Use Case Type: Detail, Essential
Stakeholders and Interests: Existing User – wants to view community forum to gain more knowledge.		
Brief Description: This use case describes how the existing user views the community forum. The existing user can search for forum posts, filter posts by category, view post details, like or save posts, comment posts, and create new forum.		
Trigger: The user selects the “Forum” option.		Type: External
Relationships: Association: Existing User		
Normal Flow of Events: 1. The existing user selects the “Forum” option. 2. The application displays the community forum page. 3. If the existing user enters a keyword in the search bar, the application retrieves the forum posts that match the keyword from the database. 4. If the existing user selects the “Trending” tab, the application retrieves forum posts sorted from most likes to least likes. 5. If the existing user selects the “Latest” tab, the application retrieves forum posts sorted from latest to oldest. 6. If the existing user selects the “Saved” tab, the application retrieves forum posts saved by the existing user. 7. If the existing user selects the “My Posts” tab, the application retrieves the forum posts created by the existing user. 8. The application displays a list of forum posts along with the “+” button. 9. If the existing user taps a forum post, the S-1: View Forum Post is performed. 10. If the existing user chooses to create a new forum post, the S-2: Add New Forum Post is performed.		
SubFlows: S-1: View Forum Post 1. The application displays the post information, comment list, and comment input field. 2. If the existing user taps the “Report Post” option and submits a report,		

the application stores the report information in the database. 3. If the existing user taps the “Delete Post” option and confirms the action, the application removes the post from the forum post lists. 4. If the existing user taps the “Edit Post” option and updates the information, the application validates and updates the post in the database. 5. If the existing user enters a comment or reply and sends it, the application updates the comment list and stores the comment in the database. 6. If the existing user taps the like or save button on the post, the application updates the like or save status and stores it in the database. 7. If the existing user selects a sorting option for comments, the application sorts comments based on the sorting option. 8. If the existing user taps “Report Comment” and submits a report, the application stores the report information in the database. 9. If the existing user taps “Delete Comment” and confirms the action, the application removes the comment from the comment list. 10. If the existing user taps “Edit Comment” and updates the comment, the application validates and updates the comment in the database. 11. If the existing user taps the like button on a comment, the application updates the like status and stores it in the database. S-2: Add New Forum Post 1. The existing user taps the “+” button. 2. The existing user enters the forum post title and content. 3. The application validates the post information. 4. The application saves the new forum post in the database.	Alternate/Exception Flows: S-1: View Forum Post 4a. The edited post contains empty fields. The application prompts the existing user to enter the required information again. 10a. The edited comment field is empty. The application prompts the existing user to enter the comment again. S-2: Add New Forum Post 3a. The title or content field is empty. The application displays an error message prompting the existing user to complete the required fields.
---	--

Receive Reminders

Table A.1.14 Use Case Description for Receive Reminders

Use Case Name: Receive Reminders		ID: 15	Importance Level: High
Primary Actor: Existing User		Use Case Type: Detail, Essential	
Stakeholders and Interests: Existing User – wants to receive reminders for meals to ensure meals are logged on time.			
Brief Description: This use case describes how the existing user manages meal reminders and receives notification alerts at the scheduled times.			
Trigger: The user selects the “Reminder” option. Type: External			
Relationships: Association: Existing User			
Normal Flow of Events: 1. The existing user selects the “Reminder” option. 2. The application displays the Reminder page. 3. If the existing user taps a reminder card and edits the reminder time, the application updates the reminder information in the database. 4. If the existing user taps the “+” button, the S-1: Add New Reminder is performed. 5. If the existing user toggles the reminder switch on or off, the application saves the reminder status in the database. 6. If the existing user long-presses a reminder card and confirms the deletion, the application removes the reminder from the database. 7. The application generates and sends a notification when the scheduled time is reached. 8. The existing user receives the reminder notification.			
SubFlows: S-1: Add New Reminder 1. The application displays the new reminder dialog box. 2. The existing user enters the reminder name and selects a time. 3. The application saves the reminders details in the database.			
Alternate/Exception Flows: Not applicable.			

Manage User Accounts

Table A.1.15 Use Case Description for Manage User Accounts

Use Case Name: Manage User Accounts		ID: 16	Importance Level: High
Primary Actor: Existing Admin		Use Case Type: Detail, Essential	
Stakeholders and Interests: Existing Admin – wants to efficiently manage user accounts by banning or unbanning users.			
Brief Description: This use case describes how the admin manages user accounts within the proposed application.			
Trigger: The admin selects the “Users” option to manage user accounts. Type: External			
Relationships: Association: Existing Admin			
Normal Flow of Events: 1. The existing admin selects the “Users” option. 2. The application displays the user management page. 3. If the existing admin enters a keyword in the search bar, the application retrieves user accounts that match the keyword from the database. 4. If the existing admin selects a sorting option, the application sorts the user list based on the selected option. 5. If the existing admin taps the “Active” or “Banned” tab, the application retrieves active or banned user accounts from the database. 6. The application displays a list of user accounts. 7. If the existing admin taps a user card, the S-1: View User Details is performed.			
SubFlows: S-1: View User Details 1. The application displays a user information dialog box with “Ban” or “Unban” button. 2. If the existing admin taps the “Ban” or “Unban” button, the application updates the status in the database and refreshes the user list. 3. If the existing admin taps the “View Timeline” button, the application displays the banning timeline of the user account.			
Alternate/Exception Flows: Not applicable.			

Manage Admin Registration

Table A.1.16 Use Case Description for Manage Admin Registration

Use Case Name: Manage Admin Registration		ID: 17	Importance Level: High
Primary Actor: Existing Admin		Use Case Type: Detail, Essential	
Stakeholders and Interests: Existing Admin – wants to approve or decline pending admin account requests.			
Brief Description: This use case describes how an existing admin manages new admin registration requests.			
Trigger: The existing admin selects the option to manage admin account registration requests. Type: External			
Relationships: Association: Existing Admin			
Normal Flow of Events: 1. The existing admin selects the “Admin” option. 2. The application displays the admin registration management page. 3. If the existing admin enters a keyword in the search bar, the application retrieves admin accounts that match the keyword from the database. 4. If the existing admin selects a sorting option, the application sorts the admin list based on the selected option. 5. If the existing admin taps the “Pending”, “Active”, or “Declined” tab, the application retrieves the corresponding admin accounts from the database. 6. The application displays a list of admin registrations. 7. If the existing admin taps a registration card, the S-1: View Admin Registration Details is performed.			
SubFlows: S-1: View Admin Registration Details 1. The application displays a registration information dialog box. 2. If the admin registration status is “Pending”, the S-2: Pending Registration Status is performed. S-1: Pending Registration Status 1. The application displays “Approve” and “Decline” buttons. 2. If the existing admin taps the “Approve” or “Decline” button, the application updates the status in the database and refreshes the admin list.			
Alternate/Exception Flows: Not applicable.			

Manage Meals

Table A.1.17 Use Case Description for Manage Meals

Use Case Name: Manage Meals	ID: 18	Importance Level: Medium
Primary Actor: Existing Admin	Use Case Type: Detail, Essential	
Stakeholders and Interests: Existing Admin – wants to maintain accurate and up-to-date meal information so that users can log meals and receive recommendations correctly.		
Brief Description: This use case describes how the admin manages the meal database in the proposed application.		
Trigger: The admin wants to manage meal database. Type: External		
Relationships: Association: Existing Admin		
Normal Flow of Events: 1. The existing admin selects the “Meals” option. 2. The application displays a list of all meals currently stored in the database. 3. If the existing admin enters a keyword in the search bar, the application retrieves meals that match the keyword from the database. 4. If the existing admin selects a filter option for food group or category, the application updates the filtered meal list. 5. If the existing admin taps the “Add New Meal” button, the S-1: Add New Meal is performed. 6. If the existing admin taps a meal card, the S-2: View Meal Details is performed.		
SubFlows: S-1: Add New Meal 1. The application displays the “Add New Meal” dialog box. 2. The existing admin enters the meal details. 3. The application validates all required fields. 4. The application stores the new meal information in the database. S-2: View Meal Details 1. The application displays the meal information dialog box with “Edit” and “Delete” buttons. 2. If the existing admin taps the “Edit” button, the S-3: Update Meal Details is performed.		

3. If the existing admin taps the “Delete” button,
the S-4: Delete Meal Record is performed.

S-3: Update Meal Details

1. The application displays “Update Meal” dialog box.
2. The existing admin edits meal details.
3. The application validates all required fields.
4. The application updates the meal in the database.

S-4: Delete Meal Record

1. The existing admin confirms the deletion.
2. The application removes the meal from the database and updates the meal list.

Alternate/Exception Flows:

S-1: Add New Meal

- 3a. The entered meal details are empty or contain invalid values. The application prompts the existing admin to re-enter the required information.

S-3: Update Meal Details

- 3a. The updated meal details are empty or contain invalid values. The application prompts the existing admin to re-enter the required information.

Manage Reported Posts and Comments

Table A.1.18 Use Case Description for Manage Reported Posts and Comments

Use Case Name: Manage Reported Posts and Comments		ID: 19	Importance Level: Medium
Primary Actor: Existing Admin		Use Case Type: Detail, Essential	
Stakeholders and Interests: Existing Admin – wants to review and handle reported forum content to ensure that inappropriate or harmful posts are properly moderated.			
Brief Description: This use case describes how an existing admin reviews user reports on forum posts or comments and decides whether to remove the reported content or ignore the report.			
Trigger: The existing admin selects the “Reports” option. Type: External			
Relationships: Association: Existing Admin			
Normal Flow of Events: 1. The existing admin selects the “Reports” option. 2. The application retrieves and displays a list of submitted reports. 3. If the existing admin taps the “Delete Content” button, the application removes the reported forum post or comment from the database and removes the report from the report list. 4. If the existing admin taps the “Ignore” button, the application removes the report from the report list. 5. If the existing admin taps the history button, the S-1: View History is performed.			
SubFlows: S-1: View History 1. The application displays the report history list. 2. If the existing admin enters a keyword in the search bar, the application retrieves reports that match the keyword from the database. 3. If the existing admin selects a filter option, the application updates the filtered report list.			
Alternate/Exception Flows: Not applicable.			

A.2 Activity Diagram

Login Account

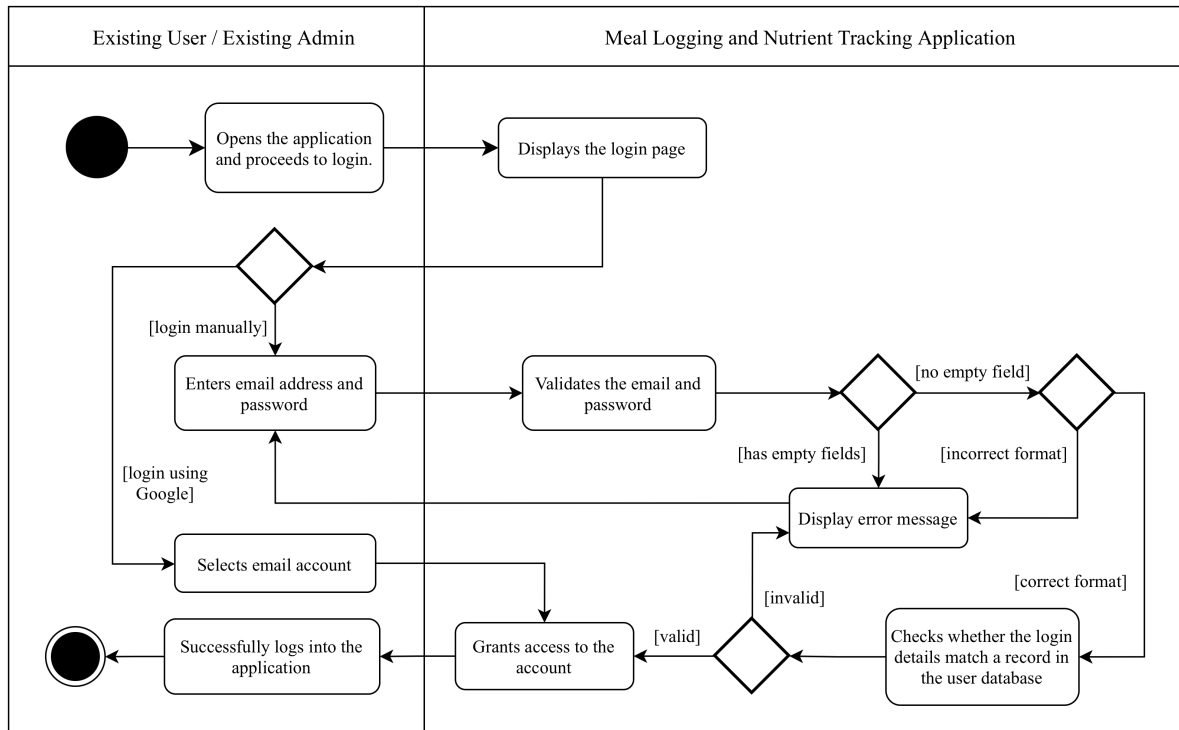


Figure A.2.1 Activity Diagram for Login Account

Input Profile

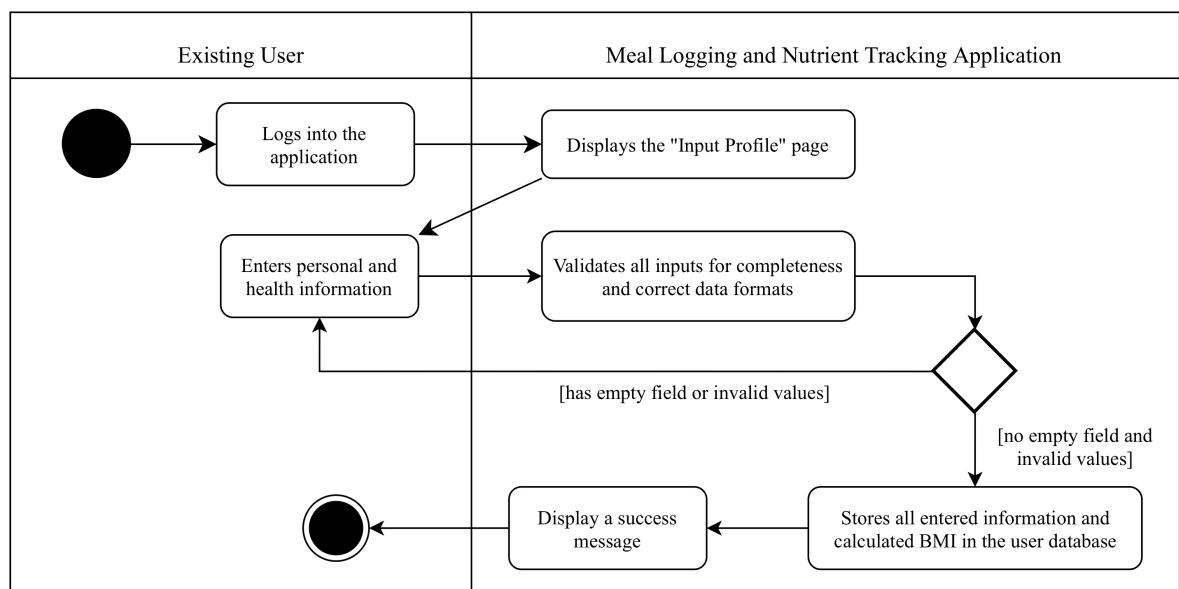


Figure A.2.2 Activity Diagram for Input Profile

View Profile

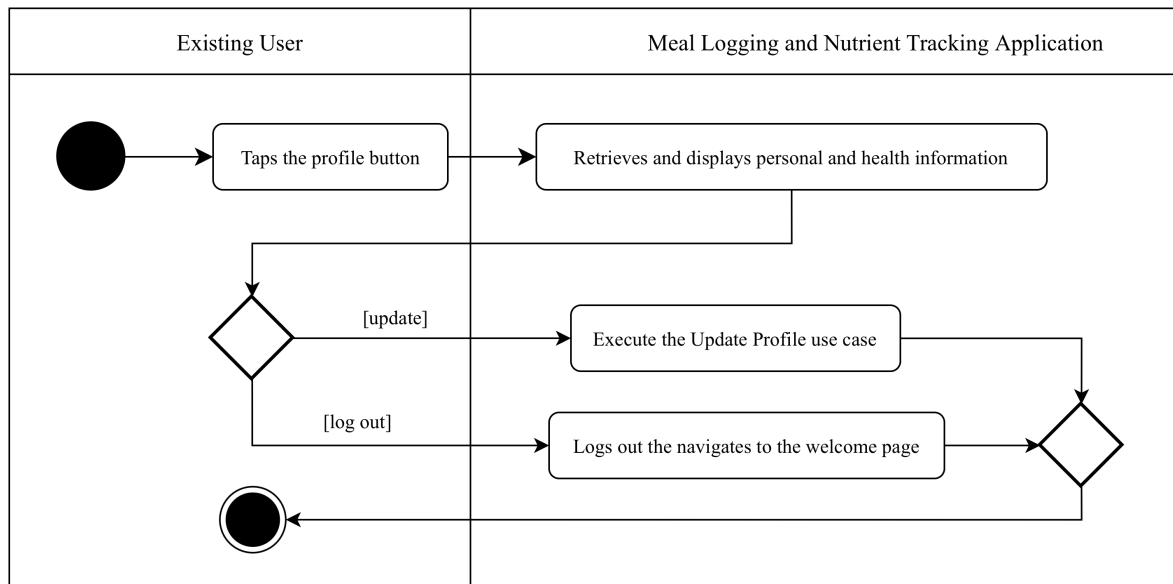


Figure A.2.3 Activity Diagram for View Profile

Update Profile

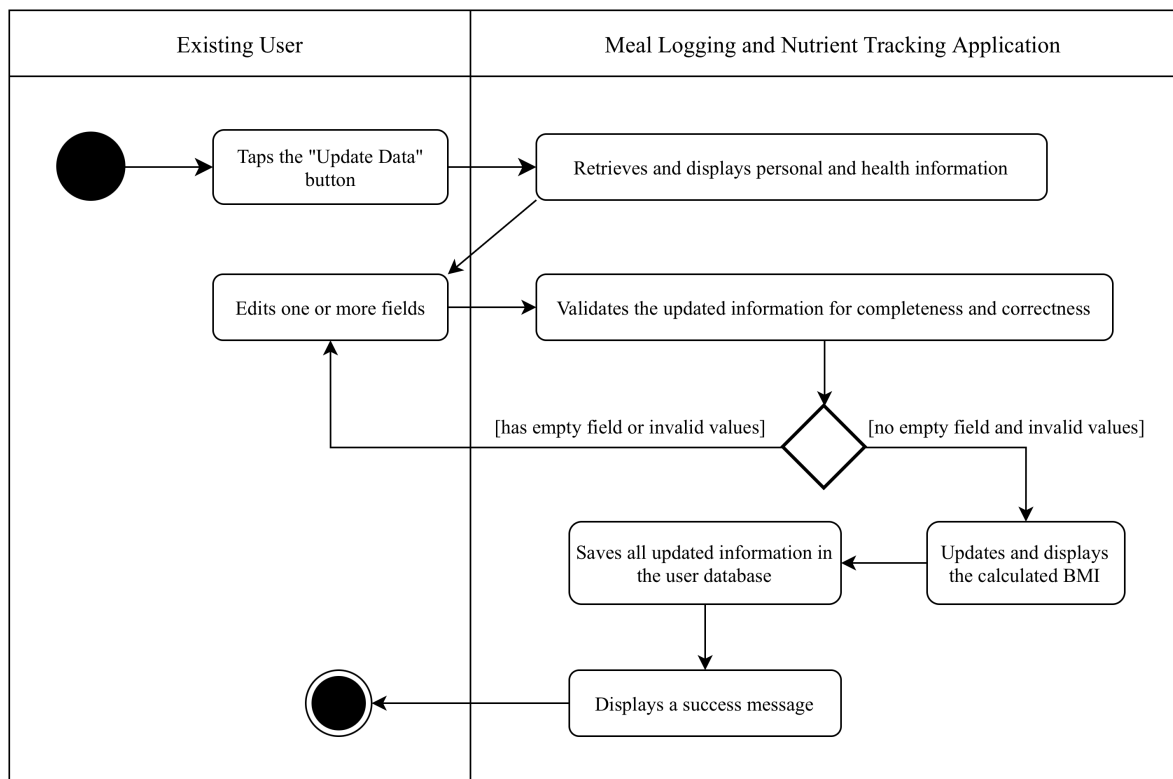


Figure A.2.4 Activity Diagram for Update Profile

View Daily Nutrient Intake Recommendations

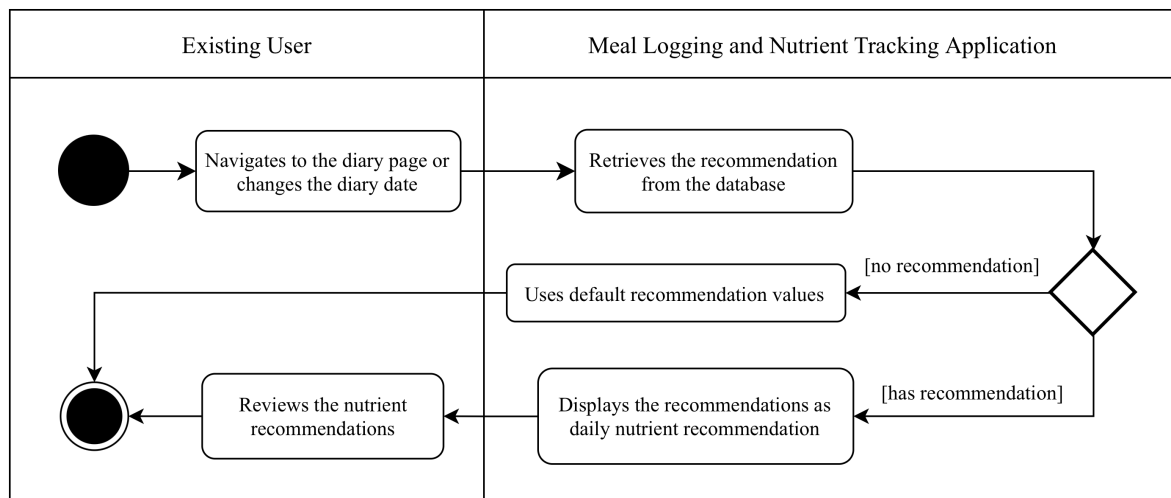


Figure A.2.5 Activity Diagram for View Daily Nutrient Intake Recommendations

View Diary

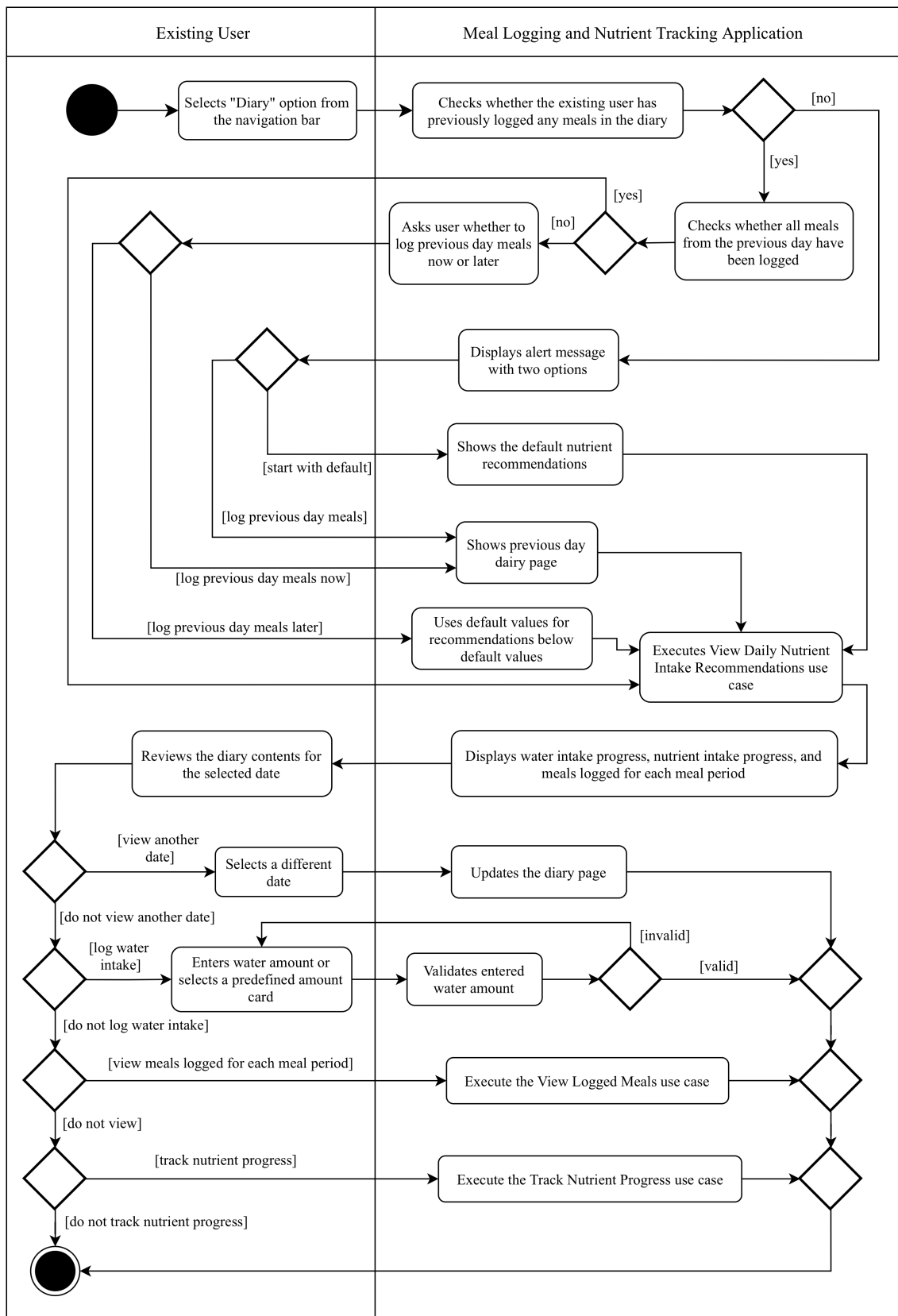


Figure A.2.6 Activity Diagram for View Diary

View Logged Meals

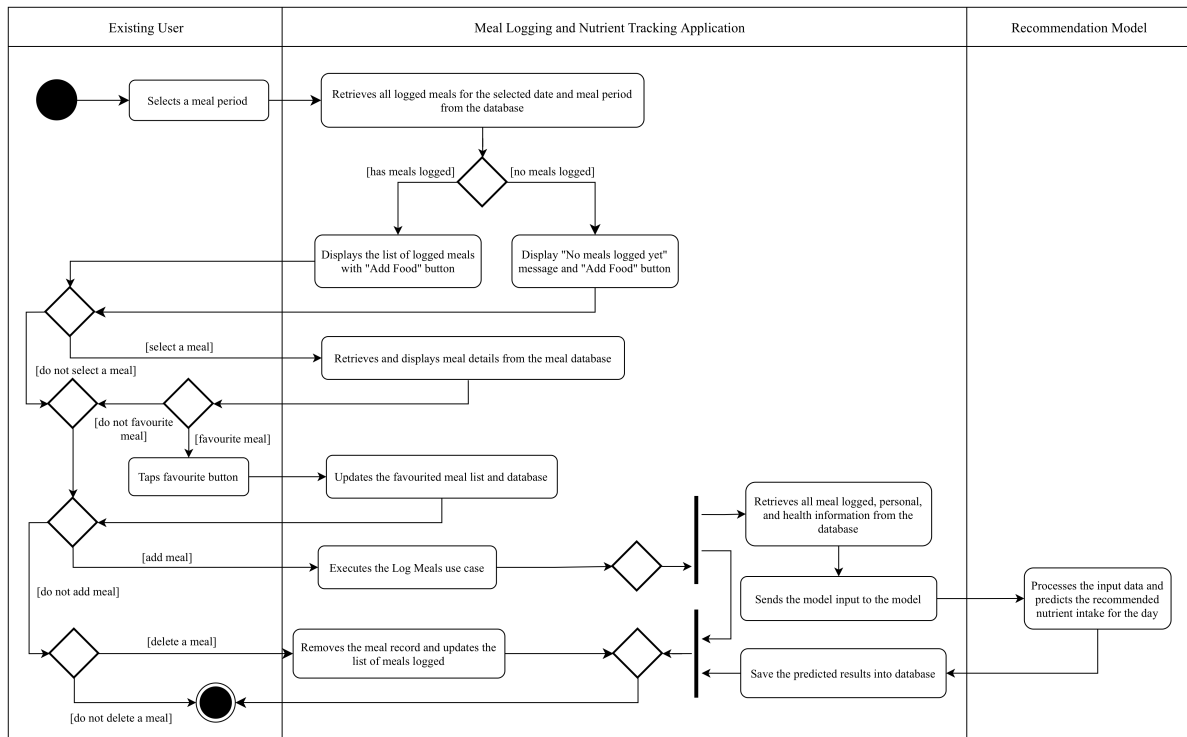


Figure A.2.7 Activity Diagram for View Logged Meals

View Meal Recommendations

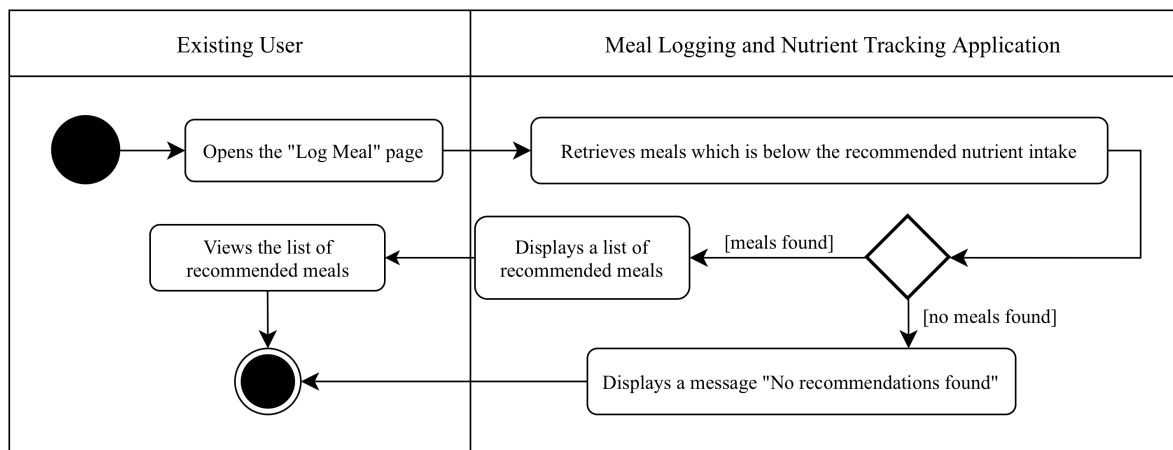


Figure A.2.8 Activity Diagram for View Meal Recommendations

Log Meals

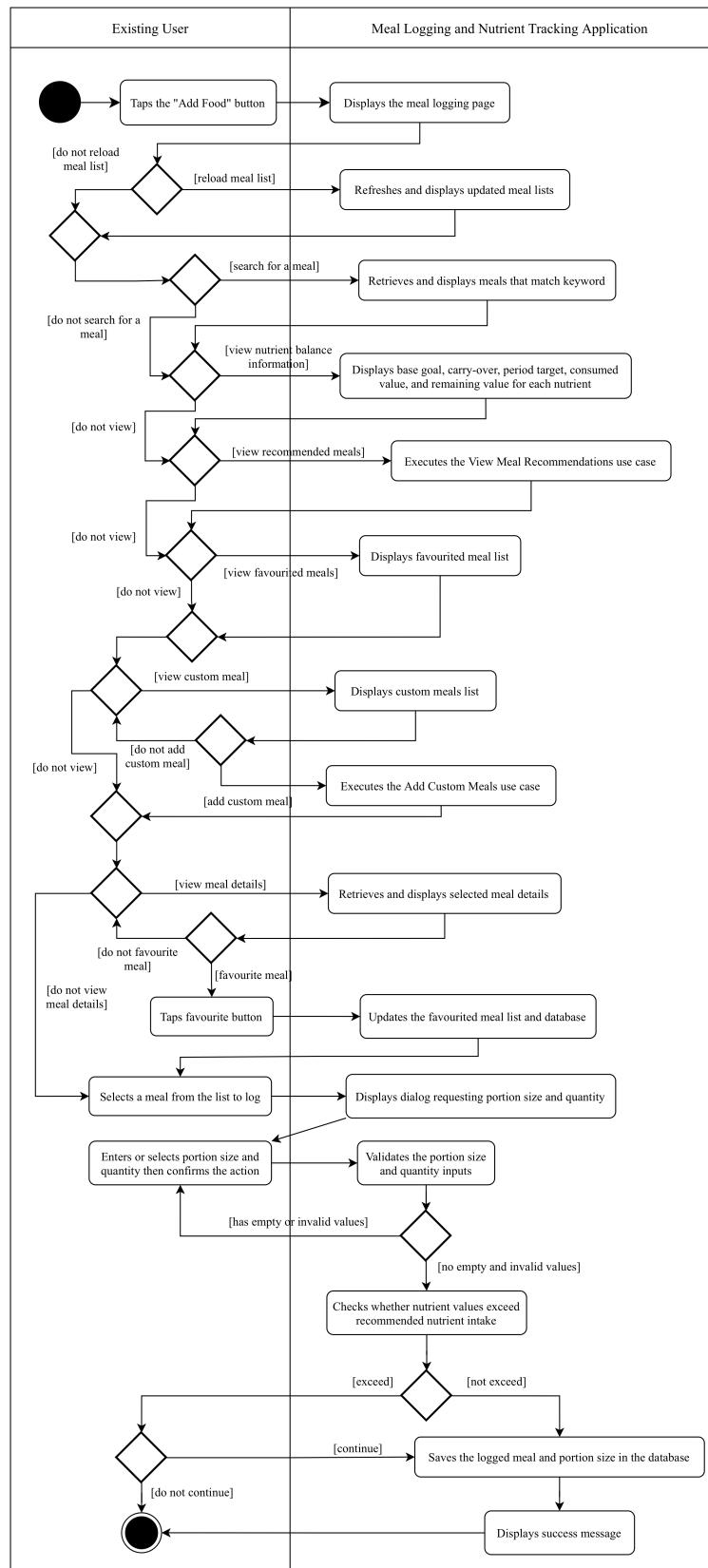


Figure A.2.9 Activity Diagram for Log Meals

Add Custom Meals

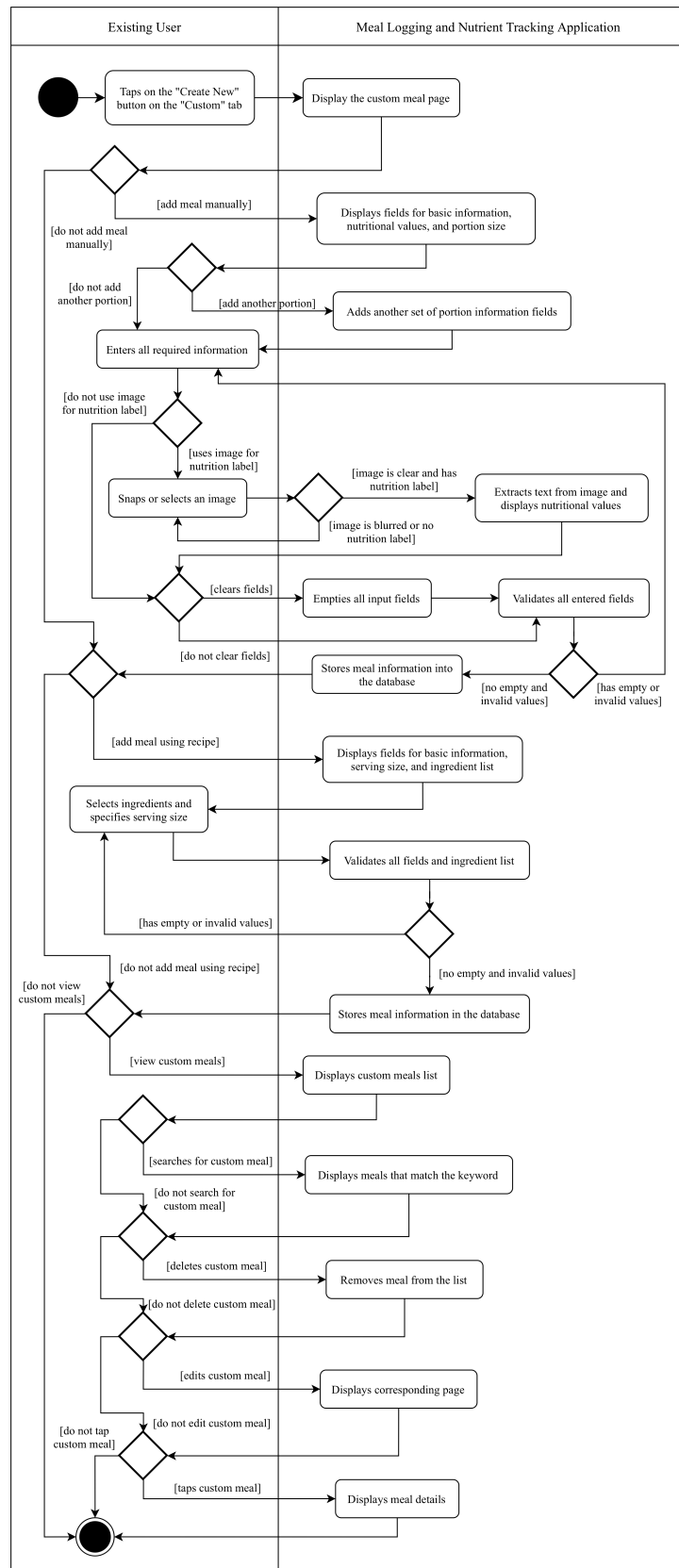


Figure A.2.10 Activity Diagram for Add Custom Meals

Track Nutrient Progress

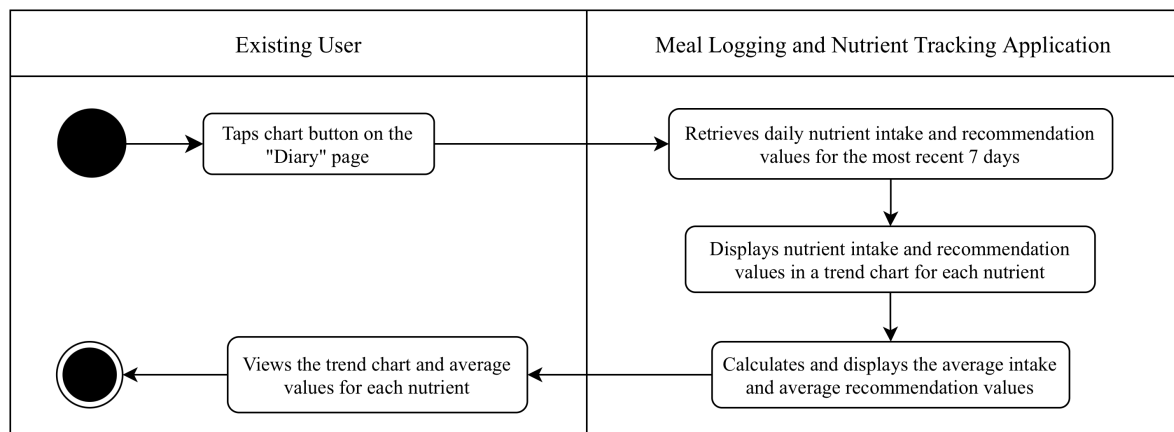


Figure A.2.11 Activity Diagram for Track Nutrient Progress

Get Personalised Nutrition Advice

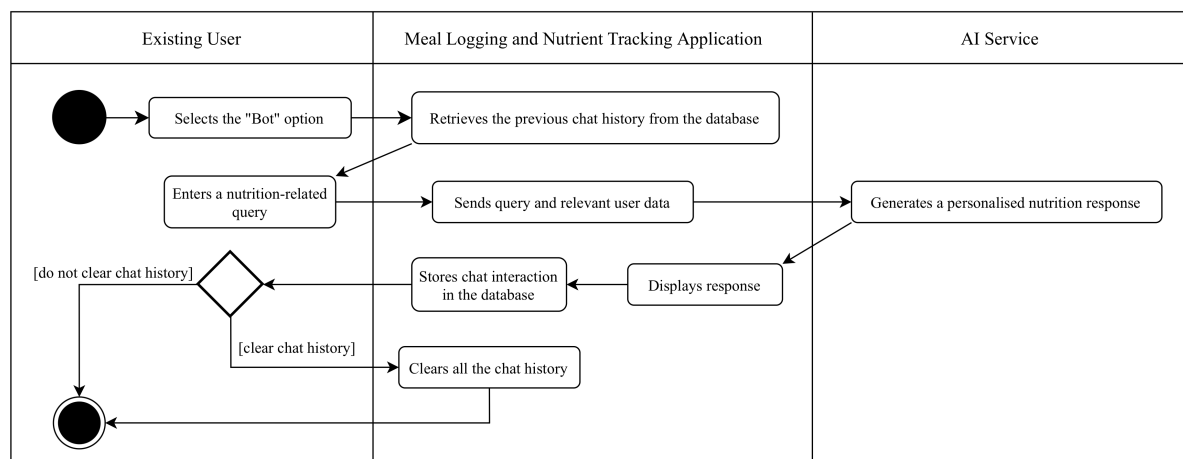


Figure A.2.12 Activity Diagram for Get Personalised Nutrition Advice

View Community Forum

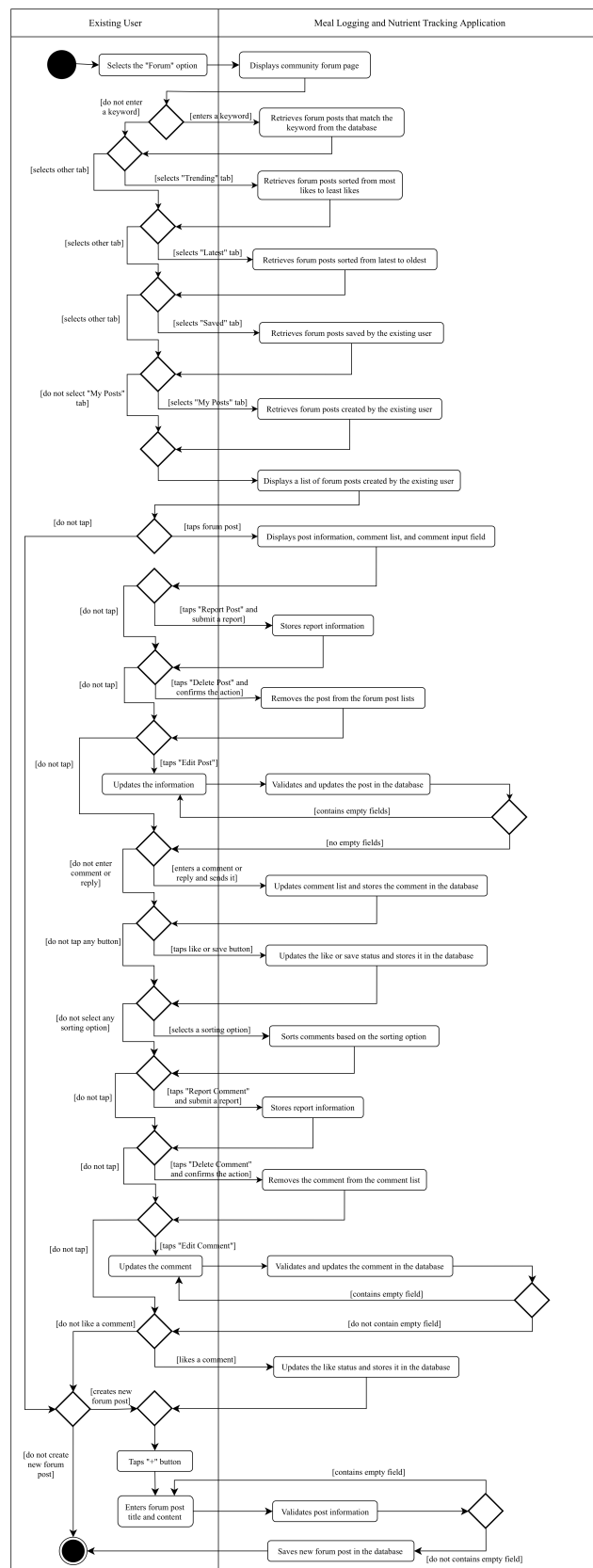


Figure A.2.13 Activity Diagram for View Community Forum

Receive Reminders

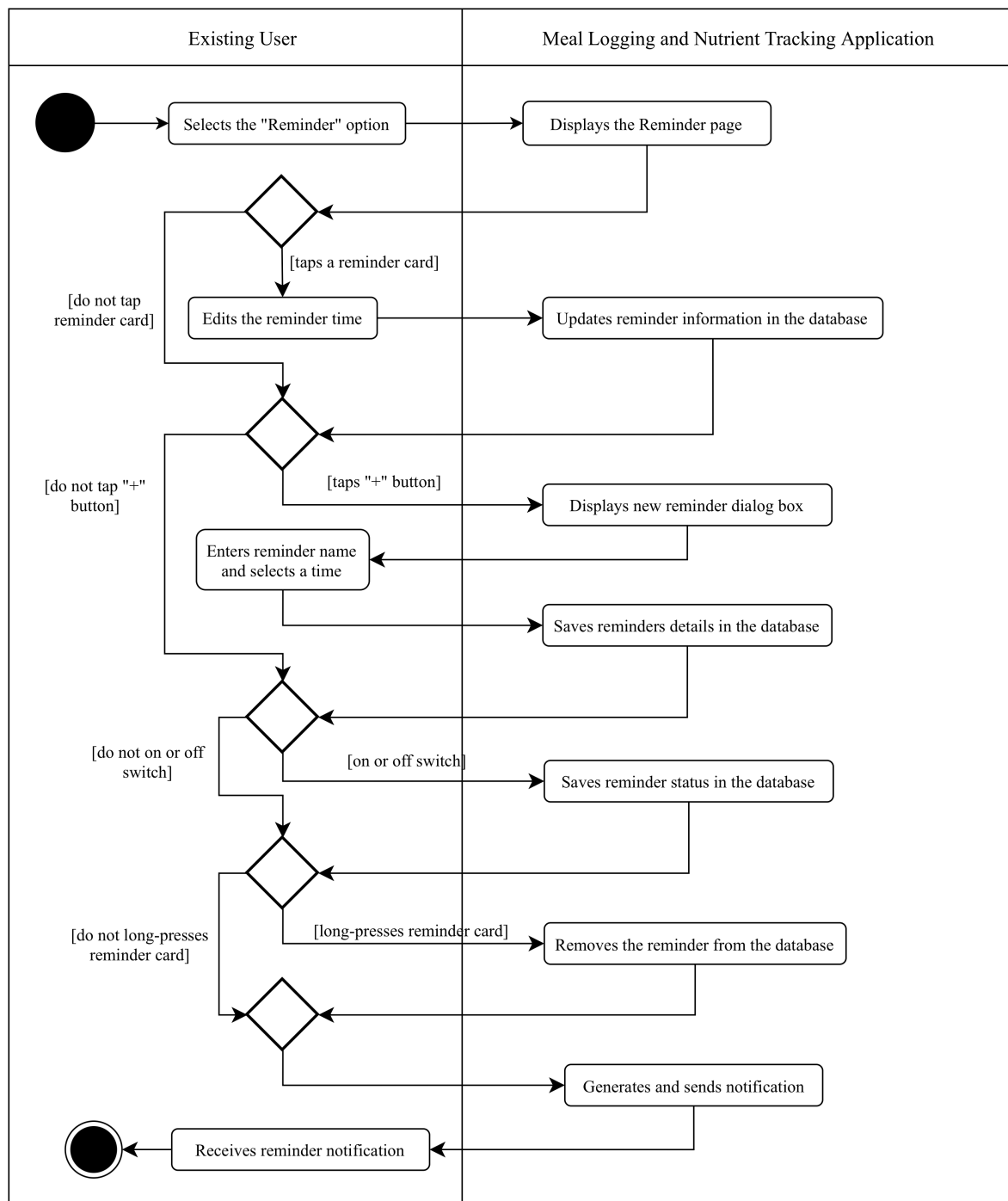


Figure A.2.14 Activity Diagram for Receive Reminders

Manage User Accounts

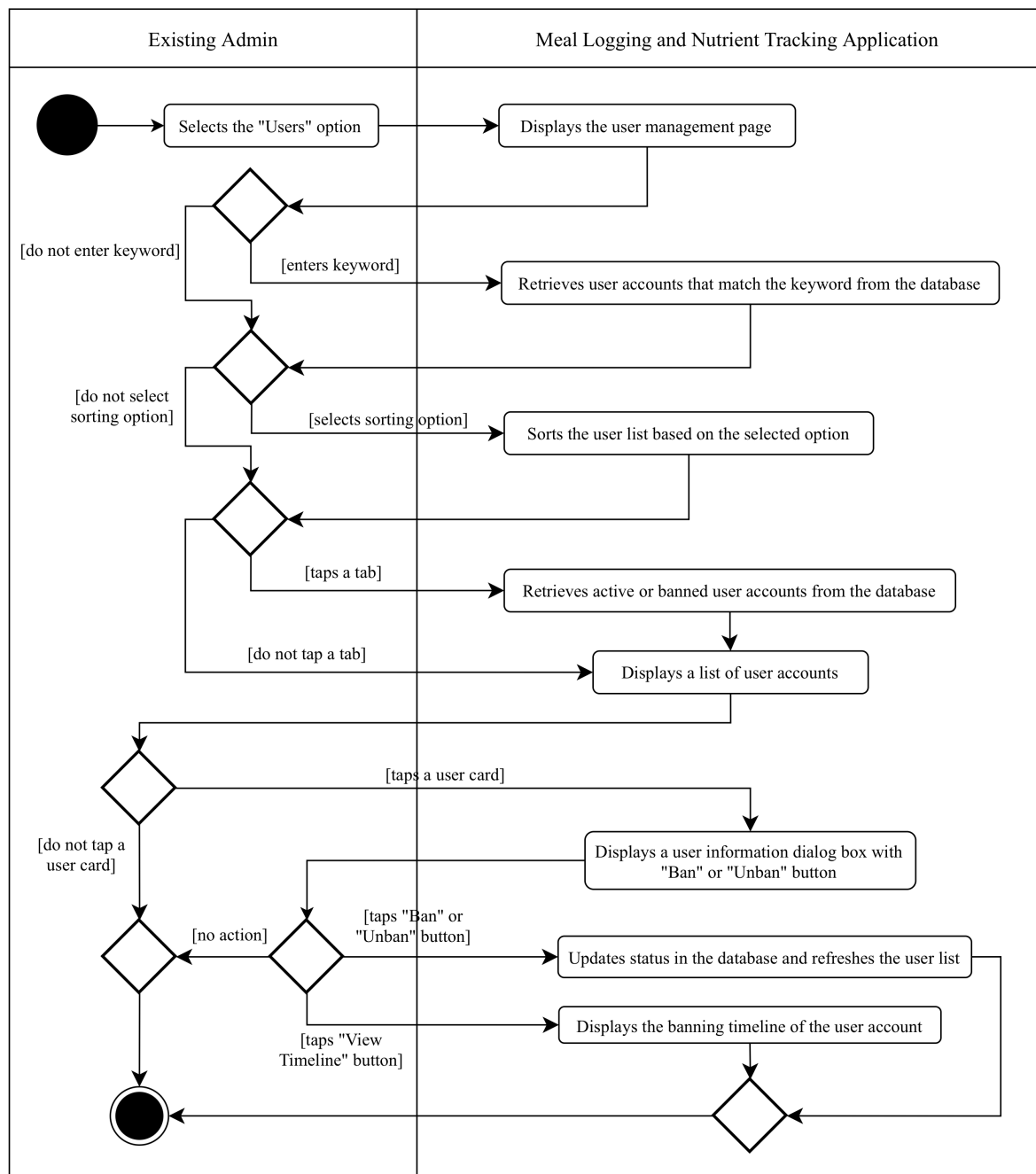


Figure A.2.15 Activity Diagram for Manage User Accounts

Manage Admin Registration

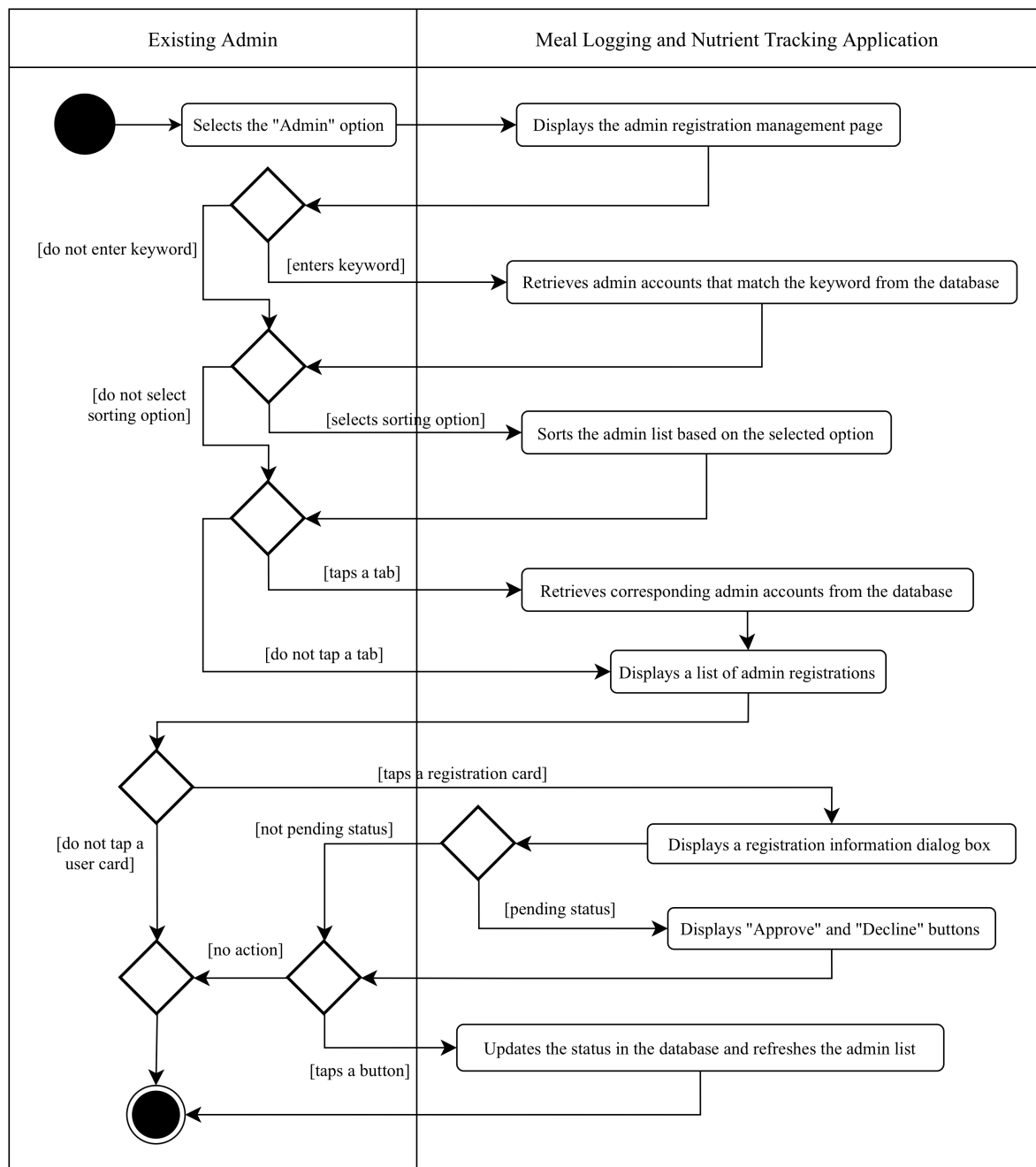


Figure A.2.16 Activity Diagram for Manage Admin Registration

Manage Meals

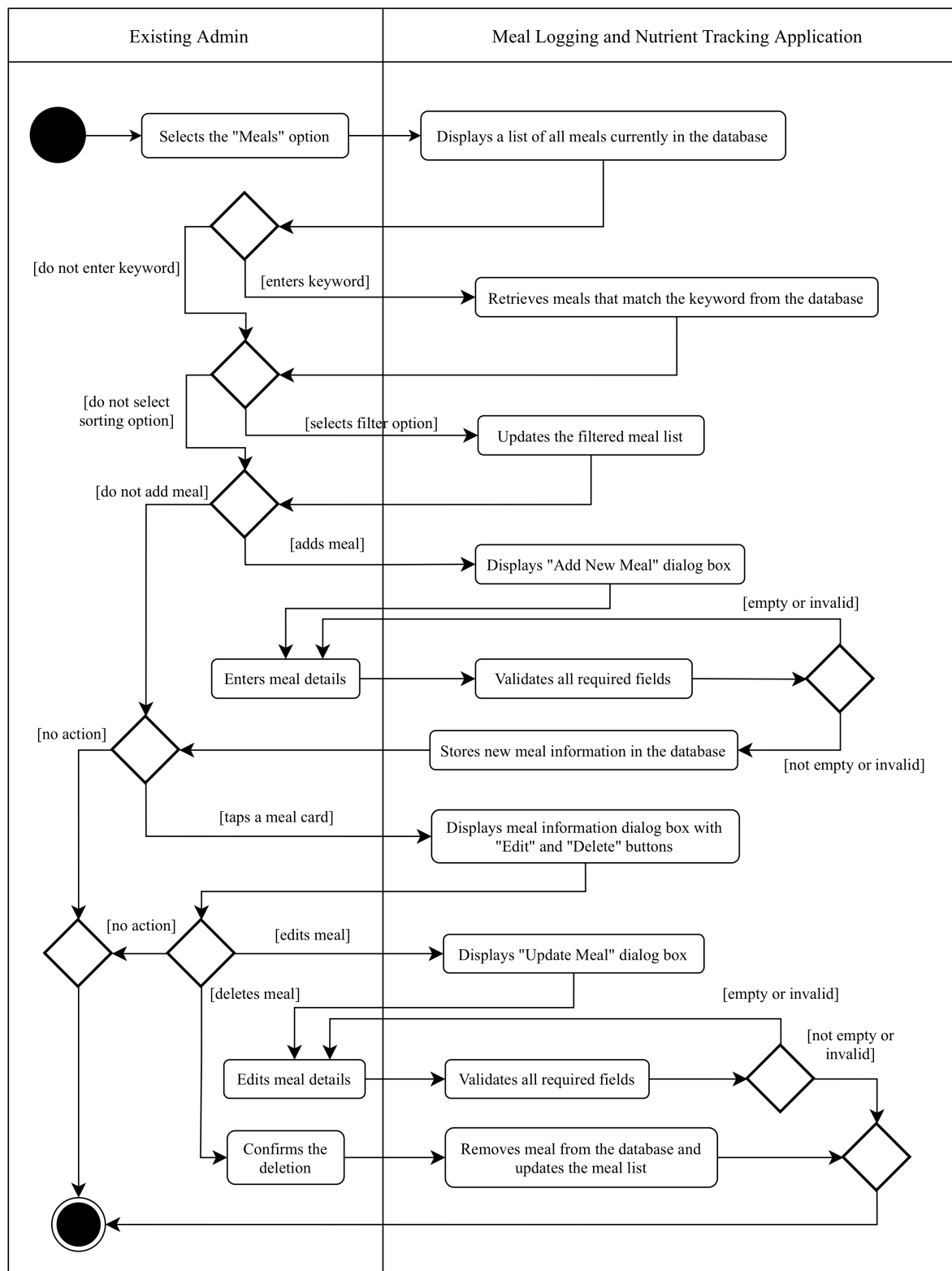


Figure A.2.17 Activity Diagram for Manage Meals

Manage Reported Posts and Comments

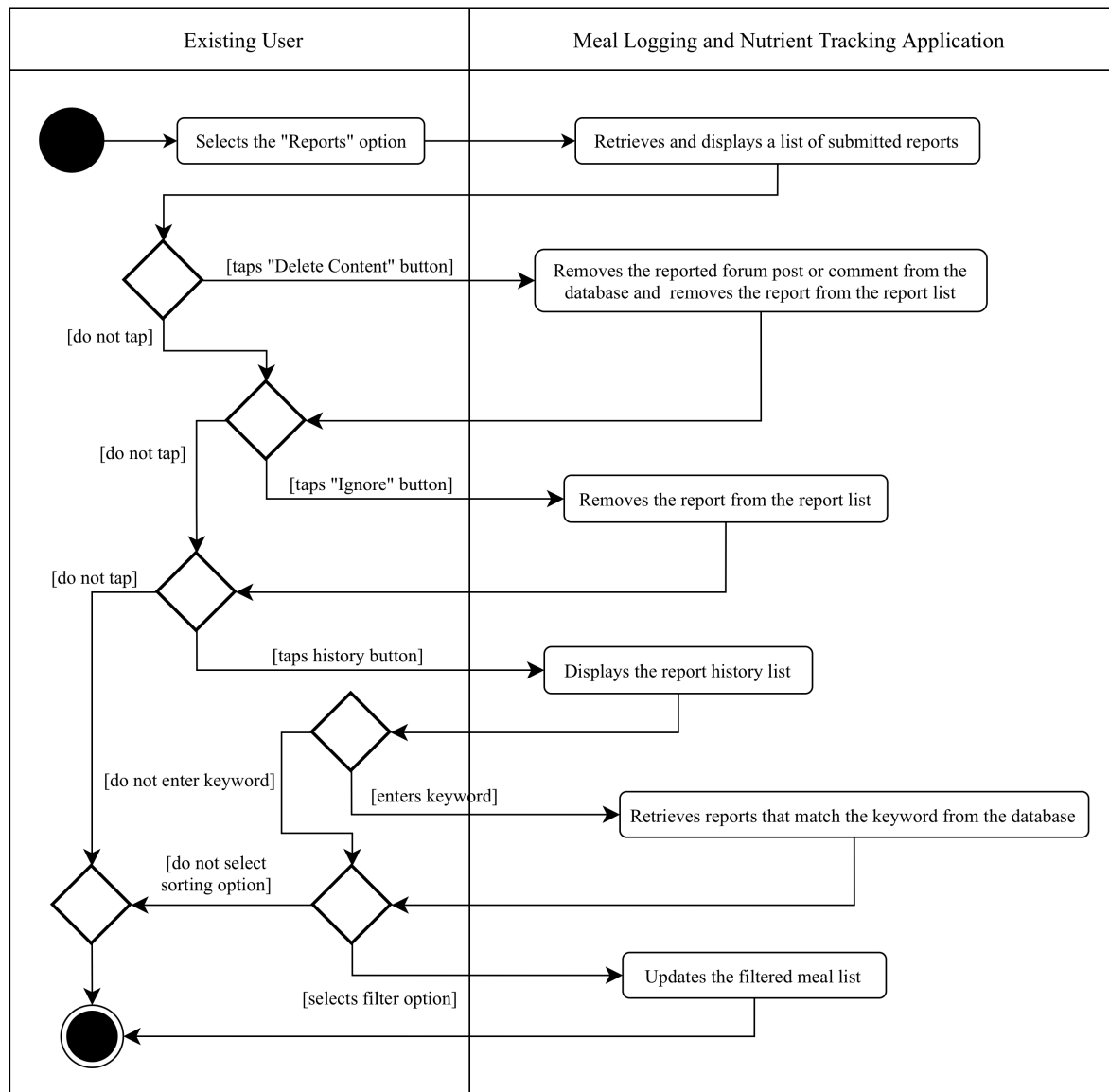


Figure A.2.18 Activity Diagram for Manage Reported Posts and Comments

A.3 Database Design

Users Collection

Document ID: userId

Table A.3.1 Database Design for the users Collection

Field / Subcollection Name	Data Type	Description	Example Value
createdAt	Timestamp	The date and time when the user account was created	January 30, 2026 at 6:04:11 PM UTC+8
email	String	The email address of the user account	"admin6@gmail.com"
registrationStatus	String	Indicates whether the account status	"pending"
role	String	The role assigned to the user in the system	"admin"
updatedAt	Timestamp	The latest date and time the user account was updated	January 30, 2026 at 6:04:11 PM UTC+8
admin	String	The ID of the admin who approved or rejected the admin account	"EaROMvO8aGR20ltPptlHgYzGd1u1"

Chats Collection

Document ID: userId

Table A.3.2 Database Design for the chats Collection

Field / Subcollection Name	Data Type	Description	Example Value
chat	-	Subcollection that stores chat messages for each user	-

Chat Subcollection

Document ID: chatId

Table A.3.3 Database Design for the chat Subcollection

Field / Subcollection Name	Data Type	Description	Example Value
role	String	The model or user who the chat item belongs to	"model"
text	String	Content of chat item	"Hello there! How can I assist you today with your meal logging or nutrition? I'm here to help you make healthy choices and track your progress."
time	Timestamp	Time of the chat item being created	March 16, 2026 at 1:00:29 PM UTC+8

Meal Logs Collection

Document ID: logId

Table A.3.4 Database Design for the mealLogs Collection

Field / Subcollection Name	Data Type	Description	Example Value
date	String	The date of the log	"Friday, 24 Oct 2025"
mealID	String	The ID of the logged meal	"eUNSdcE8Ey2pGGCgZwSI"
mealName	String	The name of the logged meal	"Peanut Butter"
mealType	String	The meal period when the meal was logged	"Breakfast"
servingSize	double	The serving size of the meal logged	100
uid	String	The ID of the user	"FcYBNyY49xRPdoeJhBS70CyJFTo2"

Custom Meal Collection

Document ID: userId

Table A.3.5 Database Design for the customMeal Collection

Field / Subcollection Name	Data Type	Description	Example Value
meals	-	Subcollection that stores custom meals for each user	-

Meals Subcollection

Document ID: mealId

Table A.3.6 Database Design for the meals Subcollection

Field / Subcollection Name	Data Type	Description	Example Value
ash	Double	The ash amount in the meal	0.7
calcium	Double	The calcium amount in the meal	69
calorie	Double	The calorie amount in the meal	43
carb	Double	The carbohydrates amount in the meal	9
createdAt	Timestamp	The date and time when the meal was created	January 14, 2026 at 3:28:24 PM UTC+8
deleted	Boolean	Indicates whether the meal is deleted	False
fat	Double	The fat amount in the meal	0.1
fibre	Double	The fibre amount in the meal	0.3
foodCategory	String	The food category of the meal	"Lunch / Dinner"
foodGroup	String	The food group of the meal	"Vegetables"
Ingredients	Array	The ingredients of the meal	-
iron	Double	The iron amount of the meal	1.4
name	String	The name of the meal	"Carrot Soup"

phosphorus	Double	The phosphorus amount of the meal	50
potassium	Double	The potassium amount of the meal	369
protein	Double	The protein amount of the meal	1.4
servings	Array	The servings of the meal	0: {grams: "282.8", name: "plate"}
sodium	Double	The sodium amount of the meal	26
type	String	The type of the meal which is recipe or meal	"recipe"
updatedAt	Timestamp	The latest date and time the meal was updated	January 29, 2026 at 4:50:19 PM UTC+8
water	Double	The water amount of the meal	88.5

Meals Collection

Document ID: mealId

Table A.3.7 Database Design for the meals Collection

Field / Subcollection Name	Data Type	Description	Example Value
ash	Double	The ash amount in the meal	0.7
calcium	Double	The calcium amount in the meal	69
calorie	Double	The calorie amount in the meal	43
carb	Double	The carbohydrates amount in the meal	9
deleted	Boolean	Indicates whether the meal is deleted	False
fat	Double	The fat amount in the meal	0.1
fibre	Double	The fibre amount in the meal	0.3
foodCategory	String	The food category of the meal	"Not Applicable"

foodGroup	String	The food group of the meal	"Vegetables"
iron	Double	The iron amount of the meal	1.4
name	String	The name of the meal	"Beetroot"
phosphorus	Double	The phosphorus amount of the meal	50
potassium	Double	The potassium amount of the meal	369
protein	Double	The protein amount of the meal	1.4
servings	Array	The servings of the meal	0: {grams: "141.4", name: "Cup (Diced)"} 1: {grams: "471.5", name: "Whole, Medium"}
sodium	Double	The sodium amount of the meal	26
updatedAt	Timestamp	The latest date and time the meal was updated	November 24, 2025 at 10:10:45 PM UTC+8
water	Double	The water amount of the meal	88.5

Posts Collection

Document ID: postId

Table A.3.8 Database Design for the posts Collection

Field / Subcollection Name	Data Type	Description	Example Value
commentCount	Int64	The comment count of the forum post	1
content	String	The content of the forum post	"I struggle to drink enough water every day and often get headaches because of dehydration. Do hydration reminders actually help, or do they become annoying after a while?"
createdAt	Timestamp	The date and time when the	November 21, 2025 at 2:20:55 PM UTC+8

		forum post was created	
deleted	Boolean	Indicates whether the forum post is deleted	False
likeCount	Int64	The likes count of the forum psot	5
title	String	The title of the forum post	"Anyone using reminders to drink more water?"
userId	String	The ID of the author	"FaBcEKrtceaSGMuLvBJBRW1NNRU2"
username	String	The name of the author	"Tony"
comments	-	Subcollection that stores comments for each forum post	-
likes	-	Subcollection that stores the user who liked the forum post	-

Comments Subcollection

Document ID: commentId

Table A.3.9 Database Design for the comments Subcollection

Field / Subcollection Name	Data Type	Description	Example Value
authorId	String	The ID of the author	"dreadfZlgCeEH9ouT6UMF9aKg222"
authorName	String	The name of the author	"Pauline"
createdAt	Timestamp	The date and time when the comment was created	February 4, 2026 at 9:49:17 PM UTC+8
deleted	Boolean	Indicates whether the comment is deleted	true
parentId	String	The ID of the comment this reply replying to	null
text	String	The content of the comment	"hi"

Likes Subcollection

Document ID: userId

Table A.3.10 Database Design for the likes Subcollection

Field / Subcollection Name	Data Type	Description	Example Value
createdAt	Timestamp	The date and time when the like are done	February 10, 2026 at 5:56:08 PM UTC+8
liked	Boolean	The like status of the user	true

Reminders Collection

Document ID: userId

Table A.3.11 Database Design for the reminders Collection

Field / Subcollection Name	Data Type	Description	Example Value
remindersList	Array	The reminder list of the user	{0: {hour: 16, isEnabled: true, minute: 53, name: "Breakfast"}}

Recommendations Collection

Document ID: userId

Table A.3.12 Database Design for the recommendations Collection

Field / Subcollection Name	Data Type	Description	Example Value
dates	-	Subcollection that stores the recommendation for each date	-

Dates Subcollection

Document ID: dateId

Table A.3.13 Database Design for the dates Subcollection

Field / Subcollection Name	Data Type	Description	Example Value
Breakfast	Map	The recommendation values for breakfast	{Calorie: 204, Carbs_g: 26, Fats_g: 6.67, Protein_g: 10}
Calories	Int64	The recommended calorie intake for the day	1020
Carbs_g	Int64	The recommended carbohydrates intake for the day	130
Dinner	Map	The recommendation values for dinner	{Calories: 918, Carbs_g: 117, Fats_g: 30, Protein_g: 45}
Fats_g	Double	The recommended fats intake for the day	33.33
Lunch	Map	The recommendation values for lunch	{Calories: 520.2, Carbs_g: 66.3, Fats_g: 17, Protein_g: 25.5}
Protein_g	Int64	The recommended protein intake for the day	50
Snack	Map	The recommendation values for snack	{Calories: 1020, Carbs_g: 130, Fats_g: 33.33, Protein_g: 50}

Reports Collection

Document ID: reportId

Table A.3.14 Database Design for the reports Collection

Field / Subcollection Name	Data Type	Description	Example Value
commentId	String	The ID of the reported comment	"vEC30LjDAc7XyCFUjFFK"
postId	String	The ID of the reported post	"EJ4zOBKmMYsr7EEDoE8U"
reason	String	The reason of reporting the post or comment	"spam"
reporterId	String	The ID of the reporter	"dreadfZlgCeEH9ouT6UMF9aKg222"
resolvedAt	Timestamp	The date and time when the report was resolved	January 22, 2026 at 6:31:09 PM UTC+8
resolvedById	String	The ID of the admin who resolved the report	"grY8x0gpZyUbFRvOdUL2I2kOxjf2"
resolvedByName	String	The name of the admin who resolved the report	"admin"
status	String	The status of the report	"deleted"
timestamp	Timestamp	The date and time when the report was created	January 22, 2026 at 6:30:43 PM UTC+8
type	String	The type of the reported item	"comment"

Saved Posts Collection

Document ID: userId

Table A.3.15 Database Design for the savedPosts Collection

Field / Subcollection Name	Data Type	Description	Example Value
posts	-	Subcollection that stores the saved posts for each user	-

Posts Subcollection

Document ID: postId

Table A.3.16 Database Design for the posts Subcollection

Field / Subcollection Name	Data Type	Description	Example Value
saved	Boolean	Indicates whether the post is saved	true
savedAt	Timestamp	The date and time when the post is saved	January 27, 2026 at 6:00:12 PM UTC+8

Users Information Collection

Document ID: userId

Table A.3.17 Database Design for the usersInfo Collection

Field / Subcollection Name	Data Type	Description	Example Value
age	Int64	The age of the user	33
ban	Boolean	The ban status of the user account	false
bloodPressureDiastolic	Double	The blood pressure diastolic of the user	70
bloodPressureSystolic	Double	The blood pressure systolic of the user	110
bloodSugar_mmolL	Double	The blood sugar of the user	5
bmi	Double	The BMI of the user	17.3
cholesterol_mmolL	Double	The cholesterol of the user	5
createdAt	Timestamp	The date and time when the user created the information	December 15, 2025 at 3:12:20 PM UTC+8
gender	String	The gender of the user	"Male"
height_m	Double	The height of the user	1.7
name	String	The name of the user	"Pauline"
updatedAt	Timestamp	The date and time when the information is updated	February 24, 2026 at 1:56:12 PM UTC+8
weight_kg	Double	The weight of the user	50

Viewed Posts Collection

Document ID: userId

Table A.3.18 Database Design for the viewedPosts Collection

Field / Subcollection Name	Data Type	Description	Example Value
posts	-	Subcollection that stores the viewed posts for each user	-

Posts Subcollection

Document ID: postId

Table A.3.19 Database Design for the posts Subcollection

Field / Subcollection Name	Data Type	Description	Example Value
viewedAt	Timestamp	The date and time of the viewing	March 14, 2026 at 10:32:42 PM UTC+8

Water Logs Collection

Document ID: waterLogsId

Table A.3.20 Database Design for the waterLogs Collection

Field / Subcollection Name	Data Type	Description	Example Value
amountMl	Int64	The amount of water intake	300
date	String	The date and time when the water intake was logged	"Monday, 02 Feb 2026"
timestamp	Timestamp	The date and time when the water intake was logged	February 2, 2026 at 10:32:46 PM UTC+8
uid	String	The ID of the user	"dreadfZlgCeEH9ouT6UMF9aKg222"

A.4 Functional Testing of Proposed Application Modules

Registration Module

Table A.4.1 Functional Test Cases for the Registration Module

Test Case ID	Test Case Description	Expected Result	Actual Result	Status
TC01	User taps the “Create Account” button when the email address, password, or confirm password fields are empty.	The proposed application displays the error messages “Please enter your email”, “Please enter your password”, or “Please confirm your password”.	The error messages are displayed.	Pass
TC02	User taps the “Create Account” button when the email address entered is in an invalid format.	The proposed application displays the error message “Enter a valid email”.	The error message is displayed.	Pass
TC03	User taps the “Create Account” button when the email address entered is already registered.	The proposed application displays the error message “The email address is already in use by another account”.	The error message is displayed.	Pass
TC04	User taps the “Create Account” button when the password entered is less than 6 characters.	The proposed application displays the error message “Password must be 6+ characters”.	The error message is displayed.	Pass
TC05	User taps the “Create Account” button when the password and confirm password do not match.	The proposed application displays the error message “Passwords do not match”.	The error message is displayed.	Pass
TC06	User taps the eye icons in the password fields.	The proposed application hides or shows the passwords.	The passwords are hidden or shown.	Pass
TC07	User taps the “Create Account” button when the email address, password, and confirm	The proposed application creates an account based on the selected role. If the role is user, the proposed application displays the Input and Update Profile page. If the role is	The user or admin account is created, and the Input and Update Profile or	Pass

	password entered are valid.	admin, the proposed application displays the Login page.	Login page is displayed.	
TC08	User taps the “Sign Up with Google” button when the selected email address is already registered.	The proposed application displays the error message “Account already exists. Please login.”	The error message is displayed.	Pass
TC09	User taps the “Sign Up with Google” button when the selected email address is not registered.	The proposed application creates an account based on the selected role. If the role is user, the proposed application displays the Input and Update Profile page. If the role is admin, the proposed application displays the Login page.	The user or admin account is created, and Input and Update Profile or Login page is displayed.	Pass
TC10	User taps the “Login” button.	The proposed application displays the Login page.	The Login page is displayed.	Pass

Password Recovery Module

Table A.4.2 Functional Test Cases for the Password Recovery Module

Test Case ID	Test Case Description	Expected Result	Actual Result	Status
TC01	User taps the “Send Reset Link” button when the email address field is empty.	The proposed application displays the error message “Please enter your email”.	The error message is displayed.	Pass
TC02	User taps the “Send Reset Link” button when the email address entered is in an invalid format.	The proposed application displays the error message “Enter a valid email”.	The error message is displayed.	Pass
TC03	User taps the “Send Reset Link” button when the email address entered is valid.	The proposed application sends a password reset email to the user.	An email with a password reset link is sent to the user.	Pass
TC04	User enters a new password and taps the “Save” button.	The proposed application updates the password to the new password entered.	The password is updated successfully.	Pass

User Home Module

Table A.4.3 Functional Test Cases for the User Home Module

Test Case ID	Test Case Description	Expected Result	Actual Result	Status
TC01	User taps the “Home” button in the navigation bar.	The proposed application displays the User Home page.	The User Home page is displayed.	Pass
TC02	User taps any of the health condition cards.	The proposed application displays a dialog box containing the current health condition, values, and ranges with the “Learn More” and “Close” buttons.	The dialog box is displayed.	Pass
TC03	User taps the “Learn More” button in the dialog box.	The proposed application navigates to a website that provides more information about the selected health condition.	The website is navigated successfully.	Pass
TC04	User taps the “Close” button in the dialog box.	The proposed application closes the dialog box.	The dialog box is closed.	Pass
TC05	User taps the “Nutrient Information” card.	The proposed application shows or hides the list of nutrient information.	The list of nutrient information is shown or hidden.	Pass

Diary Module

Table A.4.4 Functional Test Cases for the Diary Module

Test Case ID	Test Case Description	Expected Result	Actual Result	Status
TC01	User taps the “Diary” button in the navigation bar.	The proposed application displays the Diary page.	The Diary page is displayed.	Pass
TC02	The meal of the previous day is not logged, and the user is a first-time user.	The proposed application displays the “First Log” dialog box containing the “Use Defaults” and “Log Now” buttons.	The “First Log” dialog box is displayed.	Pass

TC03	User taps the “Use Defaults” button in the “First Log” dialog box.	The proposed application updates the recommendation values to the default values.	The recommendation values are updated to the default values.	Pass
TC04	The meal of the previous day is not logged, and the user is not a first-time user.	The proposed application displays the “Missing Data” dialog box containing the “Later” and “Log Now” buttons.	The “Missing Data” dialog box is displayed.	Pass
TC05	User taps the “Log Now” button in the “First Log” or “Missing Data” dialog box.	The proposed application updates the recommendation values to the minimum values.	The recommendation values are updated to the minimum values.	Pass
TC06	User taps the “Log Now” button in the “First Log” or “Missing Data” dialog box.	The proposed application displays the Diary page of the previous day without any dialog box.	The Diary page of the previous day is displayed without any dialog box.	Pass
TC07	User taps the bar chart icon.	The proposed application displays the Nutrition Trends page.	The Nutrition Trends page is displayed.	Pass
TC08	User taps the “<” button in the header.	The proposed application displays the Diary page of the previous day.	The Diary page of the previous day is displayed.	Pass
TC09	User taps the “>” button in the header.	The proposed application displays the Diary page of the next day, excluding future dates.	The Diary page of the next day excluding future dates is displayed.	Pass
TC10	User taps the date button and selects a date.	The proposed application displays a calendar with future days disabled, then displays the diary page for the selected date.	The calendar is displayed and the diary page for the selected date is displayed.	Pass
TC11	User taps the “Include Snacks in Goals” switch.	The proposed application enables the “Add Food” button in the “Snack” card when switch is on and disables the “Add Food” button in the “Snack” card then moves logged	The “Add Food” button is enabled or disabled based on the switch status and the logged meals are moved to “Dinner” when the switch is off.	Pass

		snack meals to “Dinner” when the switch is off.		
TC12	User taps the add button in the “Water” card.	The proposed application displays the “Log Water Intake” dialog box containing the amount field, amount cards, “Cancel” button, and “Log Water” button.	The “Log Water Intake” dialog box is displayed.	Pass
TC13	User taps the “Log Water” button when the entered amount is invalid.	The proposed application displays the error message “Please enter a valid amount”.	The error message is displayed.	Pass
TC14	User selects any of the amount cards.	The proposed application updates the amount field with the selected card value.	The amount field value is updated.	Pass
TC15	User taps the “Log Water” button when the entered or selected amount is valid.	The proposed application updates the progress bar, intake amount, and balance amount in the “Water” card.	The progress bar, intake amount, and balance amount are updated.	Pass
TC16	User taps any of the meal period cards.	The proposed application displays the logged meals list and the “Add Food” button.	The logged meals list and the “Add Food” button are displayed.	Pass
TC17	User taps any of the meal cards in the logged meals list.	The proposed application displays the Meal Details page containing the meal information.	The Meal Details page with the correct meal information is displayed.	Pass
TC18	User taps the “Add Food” button in a meal period card.	The proposed application displays the Meal Logging page for the selected meal period.	The Meal Logging page for the selected meal period is displayed	Pass
TC19	User logs a meal in the Meal Logging page or taps the close button on any meal card.	The proposed application updates the nutrient and water intake progress bars and balance amounts then updates the logged meal list for the respective meal period.	The progress bars and balance amounts are updated, and the logged meal list is updated with the added or removed meal.	Pass

Nutrition Trends Module

Table A.4.5 Functional Test Cases for the Nutrition Trends Module

Test Case ID	Test Case Description	Expected Result	Actual Result	Status
TC01	User selects any of the nutrient cards.	The proposed application updates the trend chart and average statistics based on the selected nutrient card.	The trend chart and average statistics are updated based on the selected nutrient card.	Pass
TC02	User taps the trend chart on a specific date.	The proposed application displays the target and intake for that date.	The target and intake for the date are displayed.	Pass
TC03	User taps the back button	The proposed application displays the Diary page.	The Diary page is displayed.	Pass

Meal Logging Module

Table A.4.6 Functional Test Cases for the Meal Logging Module

Test Case ID	Test Case Description	Expected Result	Actual Result	Status
TC01	User enters a keyword into the search bar.	The proposed application displays a list of search results.	A list of search results is displayed.	Pass
TC02	User removes the keyword from the search bar.	The proposed application displays three tabs and their corresponding meal lists.	Three tabs and their corresponding meal lists are displayed.	Pass
TC03	User taps the “Recommended” tab.	The proposed application displays the meal list containing nutrients within the recommended values for the specific meal period.	The meal list containing nutrients within the recommended values for the specific meal period are displayed.	Pass
TC04	User taps the “Favourite” tab.	The proposed application displays a list of meals favoured by the user.	A list of meals favoured by the user is displayed.	Pass

TC05	User taps the “Custom” tab.	The proposed application displays the “Create New” button and a list of meals created by the user.	The “Create New” button and a list of meals created by the user are displayed.	Pass
TC06	User taps the “Create New” button in the “Custom” tab.	The proposed application displays the Custom Meal page.	The Custom Meal page is displayed.	Pass
TC07	User taps any of the tabs.	The proposed application displays the corresponding meal list.	The corresponding meal list is displayed.	Pass
TC08	No meal is favourited, no custom meal is created, or no meal meets the recommend nutrient values.	The proposed application displays a text stating that no meal is found for the respective tab.	A text is displayed.	Pass
TC09	User taps the add button in a meal card.	The proposed application displays the “Log Meal” dialog box containing the portion size dropdown menu, custom amount field, checkboxes, quantity field, total amount field, “Cancel” button, and “Add Meal” button.	The “Log Meal” dialog box is displayed.	Pass
TC10	User enters valid values in all fields, checkboxes, and dropdown menu.	The proposed application calculates and updates the “Total Amount” field based on the portion size and quantity.	The “Total Amount” field is calculated and updated correctly.	Pass
TC11	User taps the “Add Meal” button with all fields valid.	The proposed application updates the balance amounts of the recommended values.	The balance amounts are updated.	Pass
TC12	User taps the “Add Meal” button with any invalid or empty field.	The proposed application displays error messages.	Error messages are displayed	Pass
TC13	User taps the “Cancel” button in the “Log Meal” dialog box.	The proposed application closes the “Log Meal” dialog box.	The “Log Meal” dialog box is closed	Pass
TC14	User taps any meal card.	The proposed application displays the Meal Details page containing the respective meal information.	The Meal Details page is displayed with the correct information.	Pass

TC15	User taps the info button in the balance card.	The proposed application displays a dialog box containing the base goal, carry-over, period target, consumed amount, and remaining amount for each nutrient in the selected meal period.	The dialog box with the correct base goal, carry-over, period target, consumed amount, and remaining amount for each nutrient is displayed.	Pass
TC16	User taps the refresh button.	The proposed application refreshes all the meal lists.	All the meal lists are refreshed.	Pass
TC17	User taps the back button.	The proposed application displays the Diary page.	The Diary page is displayed.	Pass

Custom Meal Module

Table A.4.7 Functional Test Cases for the Custom Meal Module

Test Case ID	Test Case Description	Expected Result	Actual Result	Status
TC01	User taps the “Quick Meal” tab.	The proposed application displays a page with fields for basic information, nutritional values, and portion information, along with camera, “Clear Fields” and “Save Meal” buttons.	A page with fields for basic information, nutritional values, and portion information, including camera, “Clear Fields” and, “Save Meal” buttons is displayed.	Pass
TC02	User taps the camera button.	The proposed application displays the options “Camera” and “Gallery”.	The options “Camera” and “Gallery” are displayed.	Pass
TC03	User taps the “Camera” option and snaps a photo of a nutrition label.	The proposed application updates each nutrient field with the values extracted from the nutrition label.	The extracted values are updated in the corresponding nutrient fields.	Pass
TC04	User taps the “Gallery” option and selects a photo of a nutrition label.	The proposed application updates each nutrient field with the values extracted from the nutrition label.	The extracted values are updated in the corresponding nutrient fields.	Pass
TC05	User taps the “Save Meal” button when value	The proposed application displays error messages.	The error messages are displayed.	Pass

	entered for numeric fields are invalid.			
TC06	User taps the “Save Meal” button when nutritional values entered are less than 0 or the portion amount entered is less than or equal to 0.	The proposed application displays error messages.	The error messages are displayed.	Pass
TC07	User taps the “Add Another Portion” button.	The proposed application adds another set of portion information fields.	Another set of portion information fields is added.	Pass
TC08	User taps the “Full Recipe” tab.	The proposed application displays the name field, group field, category field, serving size name field, ingredient list with add button, “Clear Fields” button and “Save Recipe” button.	The fields, ingredient list, and buttons are displayed.	Pass
TC09	User taps the add button in the ingredient list.	The proposed application displays the Ingredient Selecting page.	The Ingredient Selecting is displayed.	Pass
TC10	User taps the remove button of any ingredient in the ingredient list.	The proposed application removes the ingredient from the ingredient list and deducts the ingredient grams from the total grams.	The ingredient is removed and the total grams are deducted accordingly.	Pass
TC11	User taps the “Save Recipe” button when no ingredient is added.	The proposed application displays the error message “Add at least one ingredient”.	The error message is displayed.	Pass
TC12	User taps the “Save Meal” or “Save Recipe” button when there are empty fields.	The proposed application displays error messages.	The error messages are displayed.	Pass
TC13	User taps the “Save Meal” or “Save Recipe” button when all fields are valid.	The proposed application updates the meal lists in the “My List” tab of the Custom	The meal lists are updated successfully with the created meal.	Pass

		Meal page and the “Custom” tab of the Meal Logging page.		
TC14	User taps the “Clear Fields” button in the “Quick Meal” or “Full Recipe” tabs.	The proposed application clears all fields in the selected tab.	All fields in the selected tab are cleared.	Pass
TC15	User taps the “My List” tab.	The proposed application displays a search bar and a list of meals created by the user.	A search bar and a list of meals created by the user is displayed.	Pass
TC16	User enters a keyword into the search bar.	The proposed application displays the meals that match the keyword.	The matching meals are displayed.	Pass
TC17	User taps any meal card in the meal list.	The proposed application displays the Meal Details page containing the selected meal information.	The Meal Details page containing the correct information is displayed.	Pass
TC18	User taps the edit button on any meal card in the meal list.	The proposed application displays the “Quick Meal” or “Full Recipe” tab for editing based on the meal type.	The appropriate tab is displayed based on the meal type.	Pass
TC19	User taps the delete button on any meal card in the meal list.	The proposed application removes the meal from the meal lists in the “My List” tab of the Custom Meal page and “Custom” tab of the Meal Logging page.	The meal is removed from the meal lists.	Pass
TC20	User taps the back button.	The proposed application displays the Meal Logging page.	The Meal Logging page is displayed.	Pass

Ingredient Selecting Module

Table A.4.8 Functional Test Cases for the Ingredient Selecting Module

Test Case ID	Test Case Description	Expected Result	Actual Result	Status
TC01	User enters a keyword into the search bar.	The proposed application displays the ingredients that match the keyword.	The matching ingredients are displayed.	Pass
TC02	User taps any ingredient card in the ingredient list.	The proposed application displays a dialog box containing the serving option, quantity, and total weight fields with “Cancel” and “Add” buttons.	A dialog box containing the fields and buttons is displayed.	Pass
TC03	User selects a serving option and enters a quantity.	The proposed application updates the total weight based on selected the serving option and quantity.	The total weight is updated with the correct value.	Pass
TC04	User taps the “Add” button when the total weight field is empty or invalid.	The proposed application displays error messages.	The error messages are displayed.	Pass
TC05	User taps the “Add” button when the total weight field is valid.	The proposed application updates the ingredient list and total grams in the “Full Recipe” tab of the Custom Meal page.	The ingredient list and total grams are updated successfully.	Pass
TC06	User taps the “Cancel” button in the dialog box.	The proposed application closes the dialog box.	The dialog box is closed.	Pass
TC07	User taps the back button.	The proposed application displays the “Full Recipe” tab of the Custom Meal page.	The “Full Recipe” tab of the Custom Meal page is displayed.	Pass

Meal Details Module

Table A.4.9 Functional Test Cases for the Meal Details Module

Test Case ID	Test Case Description	Expected Result	Actual Result	Status
TC01	User taps the favourite button.	The proposed application updates the favourite status and the “Favourite” tab meal list on the Meal Logging page.	The favourite status and the “Favourite” tab meal list are updated.	Pass
TC02	User taps the back button.	The proposed application displays the Meal Logging page.	The Meal Logging page is displayed.	Pass

AI Nutrition Assistant Module

Table A.4.10 Functional Test Cases for the AI Nutrition Assistant Module

Test Case ID	Test Case Description	Expected Result	Actual Result	Status
TC01	User taps the “Bot” button in the navigation bar.	The proposed application displays the AI Nutrition Assistant page with chat history, an input field, and a send button.	The AI Nutrition Assistant page is displayed.	Pass
TC02	User asks for a meal recommendation.	The AI assistant provides suitable meal and nutrient recommendations retrieved from the database.	suitable meal and nutrient recommendations are provided.	Pass
TC03	User sends requests that exceed one of the AI model limits.	The proposed application switches to another available AI model.	Another AI model is used successfully.	Pass
TC04	User sends requests when all AI model limits are reached.	The AI assistant displays the error message “All models are busy. Please try again later.”	The error message is displayed by the AI assistant.	Pass
TC05	User taps the clear button in the header and confirms by tapping the “Clear” button in the dialog box.	The proposed application displays a confirmation dialog box and clears the chat history upon confirmation.	The confirmation dialog box is displayed, and the chat history is cleared after confirmation.	Pass

Community Forum Module

Table A.4.11 Functional Test Cases for the Community Forum Module

Test Case ID	Test Case Description	Expected Result	Actual Result	Status
TC01	User taps the “Forum” button in the navigation bar.	The proposed application displays the Community Forum page with a search bar, tabs, and a “New Topic” button.	The Community Forum page is displayed.	Pass
TC02	User taps the “Trending” tab.	The proposed application displays forum posts sorted from highest to lowest number of likes.	The forum post list is displayed in descending order of likes.	Pass
TC03	User taps the “Latest” tab.	The proposed application displays forum posts sorted from latest to oldest.	The forum post list is displayed in chronological order	Pass
TC04	User taps the “Saved” tab.	The proposed application displays the list of saved forum posts.	The saved forum post list is displayed.	Pass
TC05	User taps the “My Posts” tab.	The proposed application displays the forum posts created by the user.	The forum post list created by the user is displayed.	Pass
TC06	User enters a keyword into the search bar.	The proposed application displays forum posts that match the keyword within the selected tab.	The matching forum post list is displayed in the selected tab.	Pass
TC07	There are more than five forum posts available.	The proposed application displays the “Load more posts” button.	The “Load more posts” button is displayed.	Pass
TC08	User taps the “Load more posts” button.	The proposed application displays five additional forum post cards.	Five additional forum post cards are displayed.	Pass
TC09	There are no more forum posts to be displayed.	The proposed application displays the message “No more posts” below the forum post list.	The message “No more posts” is displayed below the forum post list.	Pass
TC10	User taps any forum post card.	The proposed application displays the Post Details page.	The Post Details page is displayed.	Pass
TC11	User taps the “New Topic” button.	The proposed application displays the Post Creation page.	The Post Creation page is displayed.	Pass

TC12	There are new or unread forum posts.	The proposed application highlights the post cards with a “NEW” label.	The post cards are highlighted with a “NEW” label.	Pass
------	--------------------------------------	--	--	------

Post Details Module

Table A.4.12 Functional Test Cases for the Post Details Module

Test Case ID	Test Case Description	Expected Result	Actual Result	Status
TC01	User taps the save button.	The proposed application updates the saved status and the forum post list in the “Saved” tab of the Community Forum page.	The saved status and the forum post list are updated.	Pass
TC02	User taps the like button.	The proposed application updates the like count and the forum post list in the “Trending” tab of the Community Forum page.	The like count and the forum post list are updated.	Pass
TC03	User selects the “Latest” or “Most Likes” filter.	The proposed application updates the comment list according to the selected sorting order.	The comment list is updated according to the selected sorting order.	Pass
TC03	User enters a comment in the comment input field and taps the send button.	The proposed application updates the comment list and comment count with the new comment.	The comment list and comment count are updated.	Pass
TC04	User taps the like button on a comment card.	The proposed application updates the comment like counts, like status, and comment list.	The comment like counts, like status, and comment list are updated.	Pass
TC05	User taps the three-dots button on the post header.	The proposed application displays the “Report Post” button if the post is not created by the user. If the post is created by the user, the “Edit Post” and “Delete Post” buttons are displayed.	The appropriate buttons are displayed based on the post ownership.	Pass

TC06	User taps the three-dots button on a comment card.	The proposed application displays the “Report Comment” button if the comment is not created by the user. If the comment is created by the user, the “Delete Comment” and “Edit Comment” buttons are displayed.	The appropriate buttons are displayed based on the comment ownership.	Pass
TC07	User taps the “Report Post” or “Report Comment” button.	The proposed application displays the “Report Content” dialog box with a reason input field, “Cancel” button and “Submit Report” button.	The “Report Content” dialog box is displayed.	Pass
TC08	User taps the “Submit Report” button in the “Report Content” dialog box.	The proposed application adds the report to the report list on the Report page for admin review.	The report is added to the report list.	Pass
TC09	User taps the “Delete Post” or “Delete Comment” button.	The proposed application displays the “Delete Content” dialog box with “Cancel” and “Delete” buttons.	The “Delete Content” dialog box is displayed	Pass
TC10	User taps the “Delete” button in the “Delete Content” dialog box.	The proposed application removes the post or comment from the forum post lists or comment list and updates the comment count.	The post or comment is removed, and the comment count is updated.	Pass
TC11	User taps the “Edit Post” or “Edit Comment” button.	The proposed application displays the “Edit Content” dialog box containing title and content fields or a comment field with “Cancel” and “Save Changes” buttons.	The “Edit Content” dialog box is displayed.	Pass
TC12	User taps the “Save Changes” when the title, content, or comment fields are empty.	The proposed application displays error messages.	The error messages are displayed.	Pass
TC13	User taps the “Save Changes” button when the title, content, or comment fields are valid.	The proposed application updates the post title and content or the comment content.	The post or the comment content is updated successfully.	Pass

TC14	User taps the “Cancel” button in the “Report Content”, “Delete Content”, or “Edit Content” dialog box.	The proposed application closes the corresponding dialog box.	The corresponding dialog box is closed.	Pass
TC15	User taps the reply button on a comment card and submits a reply.	The proposed application updates the reply list under the corresponding comment and updates the comment count.	The reply list and comment count are updated.	Pass
TC16	User taps the back button.	The proposed application displays the Community Forum page.	The Community Forum page is displayed.	Pass

Post Creation Module

Table A.4.13 Functional Test Cases for the Post Creation Module

Test Case ID	Test Case Description	Expected Result	Actual Result	Status
TC01	User taps the “Publish Topic” button when the title or content fields are empty.	The proposed application displays error messages.	The error messages are displayed.	Pass
TC02	User taps the “Publish Topic” button when the title and content fields are valid.	The proposed application updates the forum post lists in all relevant tabs except the “Saved” tab.	The forum post lists are updated in all relevant tabs except the “Saved” tab.	Pass
TC03	User taps the back button.	The proposed application displays the Community Forum page.	The Community Forum page is displayed.	Pass

Reminders Module

Table A.4.14 Functional Test Cases for the Reminders Module

Test Case ID	Test Case Description	Expected Result	Actual Result	Status
TC01	User taps the “Reminder” button in the navigation bar.	The proposed application displays the reminder list and an add button.	The reminder list and the add button are displayed.	Pass
TC02	User taps a reminder card.	The proposed application displays the time selection dialog box.	The time selection dialog box is displayed.	Pass
TC03	User taps the add button.	The proposed application displays the name field and the time selection dialog box.	The name field and the time selection dialog box are displayed.	Pass
TC04	User toggles the switch on a reminder card.	The proposed application enables or disables the notification for the selected reminder based on the switch status.	The notification is enabled or disabled accordingly.	Pass
TC05	One hour before a meal period reminder when the meal has not been logged.	The proposed application sends a notification containing the nutrient and meal recommendations even if the reminder switch is turned off.	The notification containing nutrient and meal recommendations is sent.	Pass
TC06	User long- presses a reminder card excluding the three meal period reminder cards.	The proposed application displays a confirmation dialog box for reminder deletion. If confirmed, the reminder card is removed, and the notification is no longer sent.	The confirmation dialog box is displayed. The reminder card is removed, and the notification is no longer sent upon confirmation.	Pass

Profile Module

Table A.4.15 Functional Test Cases for the Profile Module

Test Case ID	Test Case Description	Expected Result	Actual Result	Status
TC01	User taps the profile button at the top right corner.	The proposed application displays the personal and health information with “Update Data” and “Sign Out” buttons.	The correct information and buttons are displayed.	Pass
TC02	User taps the “Update Data” button.	The proposed application displays the Input and Update Profile page.	The Input and Update Profile page is displayed.	Pass
TC03	User taps the “Sign Out” button.	The proposed application displays the Login page.	The Login page is displayed.	Pass
TC04	User taps the back button.	The proposed application returns to the previous page.	The previous page is displayed.	Pass

Input and Update Profile Module

Table A.4.16 Functional Test Cases for the Input and Update Profile Module

Test Case ID	Test Case Description	Expected Result	Actual Result	Status
TC01	User taps the “Save Profile Data” button when there is any empty field.	The proposed application display error messages.	The error messages are displayed.	Pass
TC02	User taps the “Save Profile Data” button when a numeric field is invalid.	The proposed application display error messages.	The error messages are displayed.	Pass
TC03	User changes the unit of height, weight, cholesterol level, or blood sugar level.	The proposed application recalculates and updates the field value based on the selected unit.	The field value is recalculated and updated based on the selected unit.	Pass
TC04	User changes the height or weight value.	The proposed application recalculates and updates the BMI field.	The BMI field is recalculated and updated.	Pass

TC05	User taps the “Save Profile Data” button when all fields are valid.	The proposed application saves the updated information to the database.	The information is saved to the database.	Pass
TC06	User taps the back button.	The proposed application displays the Profile page.	The Profile page is displayed.	Pass

Admin Home Module

Table A.4.17 Functional Test Cases for the Admin Home Module

Test Case ID	Test Case Description	Expected Result	Actual Result	Status
TC01	Admin taps the “Home” button in the navigation bar.	The proposed application displays the Admin Home page with the total number of users, admins, meals, pending admin approvals, and reports.	The Admin Home page is displayed with the correct total values.	Pass
TC02	Admin taps the logout button in the top right corner.	The proposed application displays the Login page.	The Login page is displayed.	Pass

User Management Module

Table A.4.18 Functional Test Cases for the User Management Module

Test Case ID	Test Case Description	Expected Result	Actual Result	Status
TC01	Admin taps the “Users” button in the navigation bar.	The proposed application displays the User Management page with a search bar, tabs, and a user list.	The User Management page is displayed.	Pass
TC02	Admin taps the “Active” tab.	The proposed application displays a list of active user accounts.	A list of active user accounts is displayed.	Pass
TC03	Admin taps the “Banned” tab.	The proposed application displays a list of banned user accounts.	A list of banned user accounts is displayed.	Pass
TC04	Admin enters a keyword in the search bar.	The proposed application displays a list of user accounts that match the keyword.	A list of matching user accounts is displayed.	Pass

TC05	Admin taps the sort button and selects a sorting option.	The proposed application displays a bottom sheet containing a list of sorting options and sorts the user list based on the selected option.	The bottom sheet is displayed, and the user list is sorted according to the selected option.	Pass
TC06	Admin taps a user card.	The proposed application displays a dialog box containing user information, a “View Timeline” button, and a “Close” button. If the account is active, a “Ban” button is displayed. Otherwise, an “Unban” button is displayed.	A dialog box containing the correct user information and appropriate buttons is displayed.	Pass
TC07	Admin taps the “View Timeline” button.	The proposed application displays the banning timeline with the banning status, time, reason, and the admin who performed the action.	The banning timeline is displayed.	Pass
TC08	Admin taps the “Ban” or “Unban” button.	The proposed application updates the banning status and refreshes the user list in the corresponding tabs.	The banning status and the user list are updated.	Pass
TC09	Admin taps the “Close” button.	The proposed application closes the dialog box.	The dialog box is displayed.	Pass

Admin Management Module

Table A.4.19 Functional Test Cases for the Admin Management Module

Test Case ID	Test Case Description	Expected Result	Actual Result	Status
TC01	Admin taps the “Admin” button in the navigation bar.	The proposed application displays the Admin Management page with a search bar, tabs, and a list of admin cards.	The Admin Management page is displayed.	Pass
TC02	Admin taps the “Pending” tab.	The proposed application displays a list of pending admin approvals.	A list of pending admin approvals is displayed.	Pass
TC03	Admin taps the “Active” tab.	The proposed application displays a list of active admins.	A list of active admins is displayed.	Pass
TC04	Admin taps the “Declined” tab.	The proposed application displays a list of declined admin approvals.	A list of declined admin approvals is displayed.	Pass

TC05	Admin enters a keyword in the search bar.	The proposed application displays a list of admins that match the keyword.	A list of matching admin cards is displayed.	Pass
TC06	Admin taps the sort button and selects a sorting option.	The proposed application displays a bottom sheet containing a list of sorting options and sorts the admin list based on the selected option.	The bottom sheet is displayed, and the admin list is sorted according to the selected option.	Pass
TC07	Admin taps an admin card.	The proposed application displays a dialog box containing the admin information. If the approval status is pending, the “Decline” and “Approve” buttons are displayed. Otherwise, only the “Close” button is displayed.	The dialog box with the appropriate buttons is displayed.	Pass
TC08	Admin taps the “Decline” or “Approve” button.	The proposed application updates the registration status and refreshes the approval lists.	The registration status and approval lists are updated.	Pass
TC09	Admin taps the “Close” button.	The proposed application closes the dialog box.	The dialog box is closed.	Pass

Meal Management Module

Table A.4.20 Functional Test Cases for the Meal Management Module

Test Case ID	Test Case Description	Expected Result	Actual Result	Status
TC01	Admin taps the “Meals” button in the navigation bar.	The proposed application displays the Meal Management page with a search bar, filters, a meal list, and a “Add New Meal” button.	The Meal Management page is displayed.	Pass
TC02	Admin enters a keyword in the search bar or selects an option in the “Food Group” and “Food Category” filters.	The proposed application displays the matching meal list.	The matching meal list is displayed.	Pass

TC03	Admin taps a meal card.	The proposed application displays a dialog box containing the meal information with “Edit”, “Delete”, and “Close” buttons.	A dialog box with the meal information and buttons is displayed.	Pass
TC04	Admin taps the “Delete” button.	The proposed application removes the meal from the meal list.	The meal is removed from the meal list.	Pass
TC05	Admin taps the “Close” button.	The proposed application closes the dialog box.	The dialog box is closed.	Pass
TC06	Admin taps the “Edit” button.	The proposed application displays a dialog box containing the meal information fields with camera, “Add Serving Option”, “Cancel”, and “Update” buttons.	The edit dialog box is displayed.	Pass
TC07	Admin taps the “Add New Meal” button.	The proposed application displays a dialog box containing the meal information fields with camera, “Add Serving Option”, “Cancel”, and “Add” buttons.	The add meal dialog box is displayed.	Pass
TC08	User taps the camera button.	The proposed application displays the options “Camera” and “Gallery”.	The options “Camera” and “Gallery” are displayed.	Pass
TC09	User taps the “Camera” option and snaps a photo of a nutrition label.	The proposed application updates each nutrient field with the values extracted from the nutrition label.	The extracted values are updated in the corresponding nutrient fields.	Pass
TC10	User taps the “Gallery” option and selects a photo of a nutrition label.	The proposed application updates each nutrient field with the values extracted from the nutrition label.	The extracted values are updated in the corresponding nutrient fields.	Pass
TC11	Admin taps the “Add Serving Option” button.	The proposed application adds a new set of serving option fields.	A new set of serving option fields is added.	Pass
TC12	Admin taps the “Cancel” button.	The proposed application closes the dialog box.	The dialog box is closed.	Pass
TC13	Admin taps the “Add” or “Update” button when any required field is empty.	The proposed application displays error messages.	The error messages are displayed.	Pass

TC14	Admin taps the “Add” or “Update” button when any numeric field is invalid.	The proposed application displays error messages.	The error messages are displayed.	Pass
TC15	Admin taps the “Add” or “Update” button when all fields are valid.	The proposed application adds the new meal to the meal list or updates the existing meal information.	The meal is added to the meal list, or the meal information is updated successfully.	Pass

Reported Posts and Comments Management Module

Table A.4.21 Functional Test Cases for the Reported Posts and Comments Management Module

Test Case ID	Test Case Description	Expected Result	Actual Result	Status
TC01	Admin taps the “Reports” button in the navigation bar.	The proposed application displays the Reported Posts and Comments Management page with report cards containing the report information, an “Ignore” button, and a “Delete Content” button.	The Reported Posts and Comments page with the report cards is displayed.	Pass
TC02	Admin taps the “Ignore” button.	The proposed application removes the report card from the report list.	The report card is removed from the report list.	Pass
TC03	Admin taps the “Delete Content” button.	The proposed application removes the reported post or comment and removes the report from the report list.	The reported post or comment and report are removed.	Pass
TC04	Admin taps the history button.	The proposed application displays the Report History page with a list of report history cards containing the report information.	The Report History page is displayed.	Pass

Report History Module

Table A.4.22 Functional Test Cases for the Report History Module

Test Case ID	Test Case Description	Expected Result	Actual Result	Status
TC01	Admin enters a keyword into the search bar.	The proposed application updates the matching report list.	The matching report list is updated	Pass
TC02	Admin taps the filter button.	The proposed application displays a dropdown list with the options of “All”, “Posts”, and “Comments”.	The dropdown list is displayed.	Pass
TC03	Admin selects “All”, “Posts”, or “Comments” in the dropdown list.	The proposed application filters the report list based on the selection.	The report list is filtered correctly.	Pass

APPENDIX B

Poster



Meal Logging and Nutrient Tracking Application

INTRODUCTION

This project developed a meal logging and nutrient tracking application that helps users monitor their daily nutrient intake and recommends meals using recommended nutrient values based on their health conditions.

OBJECTIVE

- To develop a mobile application that can record daily meals of the users and track their nutrient intake.
- To integrate a model to recommend suitable meals to users based on their health conditions.

PROPOSED METHOD

- System architecture:**
Mobile application as client, with Firebase (Storage & Authentication) and Render (FastAPI + GitHub) as server.
- Nutrient recommendation model:**
Linear Regression

MAIN FUNCTIONS

- Users:**
Meal logging, user authentication, profile management, nutrient recommendations, custom meal creation, personalised nutritional advice, reminders, and community forum.
- Admins:**
Management of user accounts, admin registration, meals, and reported posts and comments.

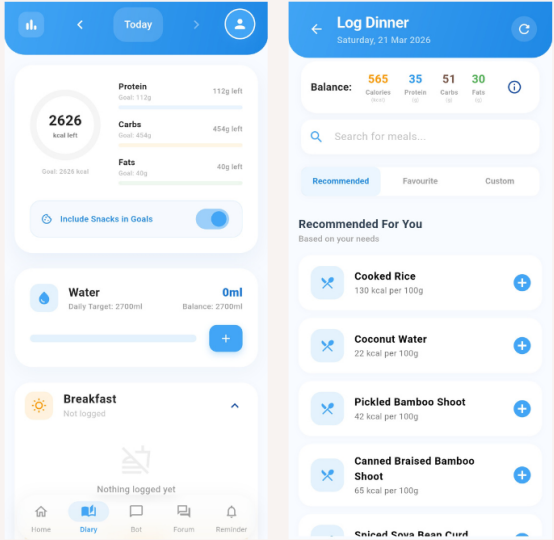
WHY THE PROPOSED APPLICATION IS BETTER?

- Includes personal health information for more accurate recommendations.
- Recommends daily nutrient intake and suitable meals based on health conditions.

CONCLUSION

The meal logging and nutrient tracking application (FoodWise) uses users' personal and health information together with their previous day's nutrient intake to generate nutrient intake recommendations and display suitable meals.

EXAMPLE OF RESULTS





Developer: Wong Mei Kuan Supervisor: Ts. Dr. Lim Seng Poh

