[1]

Software Requirements Specification

for

MEAL TRACKER

Version 3.0 approved

Prepared by <4 Man Team>

<Nanyang Technological University>

<25/03/2021>

Team Members:
Chua Ming Hui
Ng Pek Han
Neo Rui Xuan Berlynn
Tang Yuting
Wang Binli

Table of Contents

Table of Contents ..... **ii**
1. Introduction **1**
1.1. Purpose 1
1.2. Document Conventions 1
Bolded phrases are used to highlight the main points of the section. 1
1.3. Intended Audience and Reading Suggestions 1
1.4. Product Scope 1
1.5. References 1
2. Overall Description **2**
2.1. Product Perspective 2
2.2. Product Functions 2
2.3. User Classes and Characteristics 3
2.4. Operating Environment 3
2.5. Design and Implementation Constraints 3
2.6. User Documentation 3
2.7. Assumptions and Dependencies 3
3. External Interface Requirements **4**
3.1. User Interfaces 4
3.2. Hardware Interfaces 5
3.3. Software Interfaces 5
3.4. Communications Interfaces 5
4. System Features **5**
4.1. Record Food Consumption 5
4.2. Healthy Food Recommendation 7
4.3. Track Health Statistics 8
4.4. Account 10
5. Other Nonfunctional Requirements **12**
5.1. Performance Requirements 12
5.2. Security and Privacy Requirements 12
5.3. Software Quality Attributes 13
5.4. Other Requirements 13
6. Appendix A: Data Dictionary and Glossary **14**
7. Appendix B: Analysis Models **15**
7.1. Use Case Model 15
7.2. Use Case Description 18
7.3. Class Diagram 32
7.4. Sequence Diagram 36
7.5. Dialog Map 49
7.6. ER Diagram 50
8. Appendix C: System Design **51**
8.1. Architecture Design 51
8.2. Design Issue 52
8.3. Design Patterns 52
9. Appendix D: UI Mockups **54**
10. Appendix E: To Be Determined List **63**

1. Introduction

1.1. Purpose

The purpose of this document is to provide details about our application, which aims to facilitate healthy eating by providing users with a way to record calories and nutrients of each meal and based on that to maintain or reach their goal weight. This document covers the functional and non-functional requirements, data dictionary, initial use case model and UI mock-ups.

1.2. Document Conventions

Bolded phrases are used to highlight the main points of the section.

1.3. Intended Audience and Reading Suggestions

The intended audience are developers, project managers, marketing staff, users, testers and documentation writers. The rest of this SRS contains functional and non-functional requirements, use case diagrams and UI mockups.
It is recommended that readers follow this sequence:

1. Functional requirements (Section 4. System Features)
2. Non-functional requirements (Section 5)
3. Use Case Models (Appendix B)
4. UI Mockups (Appendix C)

1.4. Product Scope

Our application aims to encourage healthy eating habits amongst Singaporeans by providing them with a convenient way to track their calorie intake and get nutritional information. Additionally, they can obtain information about the nutrients they lack and what foods contain such nutrients. They can also obtain an overview of their weekly calorie intake to view their progress and review their meals from previous days.

This is in line with the Smart Nation movement to change the ways we live and is not just for people who seek to diet, but also those who want to maintain their weight like athletes. This application will greatly benefit such users because it reduces the hassle of trying to obtain information from different sites, as it is all compiled into one simple application.

1.5. References

This SRS refers to CZ2006 lecture notes, API documentations.

2. Overall Description

2.1. Product Perspective

The product is a new, self-contained product.

2.2. Product Functions

The major functions the product must perform are summarized as follows:

- Record Food Consumption
  - Store and retrieve meal records
  - Retrieve calorie and nutrition information of food
- Healthy Food Recommendation
  - Calculate suggested daily intake of all nutrients
  - Calculate weekly average daily intake of nutrients
  - Report all nutrients 40% higher or lower than suggested level
  - Retrieve top 10 nutrients with smallest proportion of actual intake in suggested intake
  - Display top 5 foods rich in nutrients searched by user
  - Display top 10 foods rich in nutrients that user lacks
- Track Health Statistics
  - Calculate suggested daily calorie intake
  - Calculate daily calorie intake based on the day's meal records
  - Calculate remaining calorie quota
  - Calculate weekly average calorie intake
  - Display health statistical summary report
  - Display meal records on particular day queried by user
  - Notify user whether they met the daily calorie intake
  - Store health information and update if user edits it
- Account
  - Store account information
  - Register user
  - Validate user account availability
  - Store new account information if changed
  - Require user to log in before using our application
  - Allow user to change password

The major functions the product must let the user perform are as follows:

- Record Food Consumption
  - Add meal records
  - Edit meal records
  - Query meal records
- Healthy Food Recommendation
  - Query suggested daily intake of all types of nutrients
  - Query average daily intake of nutrients
  - Query top 10 nutrients with smallest proportion of average daily intake compared to suggested intake
  - Enter nutrient's name and query top 5 foods rich in that nutrient
- Track Health Statistics

  - Query health statistical summary report
  - Edit health information
- Account
  - Register or log in with email address, Facebook or Google
  - Request for password change

Detailed information of the functions can be found in section 3.

2.3. User Classes and Characteristics

Our target users include people of all ages who want to eat healthily.

Users	Characteristics	Frequency of use
People on diet	People who are looking to watch and cut down calories	High
Athletes and models	People who must stay fit and have a healthy diet	High
Health enthusiasts and people with illnesses	People who want to stay fit and have a healthy diet	Medium
General population	People who want to eat healthily	Low to Medium

2.4. Operating Environment

The software will operate on Android devices with Android version above 11.0 (inclusive). It will require internet connection and location to operate properly.

2.5. Design and Implementation Constraints

Some constraints include the following:

1. The food database API may not contain information about certain types of food.
2. The user interface is designed for a fixed screen size and users who have devices with other screen sizes may not have the optimal experience.

2.6. User Documentation

The UI Mockups are provided in Appendix C.

2.7. Assumptions and Dependencies

The assumptions are as follows:

- The user is using an Android device.
- The user has an email address.
- If the user links their account with social media, their social media will contain data of their email address which can be taken to be used in our application for further verification purposes.

The dependencies are as follows:

- The accuracy of information on food recognition and food nutritional information depends on the data retrieved from the Facebook API, Google API, food recognition API and food database API respectively.

3. External Interface Requirements

3.1. User Interfaces

The user interface is designed according to Schneiderman's Eight Golden Rules of Interface Design.

3.1.1. Strive for consistency

- Consistent usage of side bar to navigate to different functionalities of the application
- Consistent usage of grey and white theme in all pages
- Consistent use of error message boxes

3.1.2. Cater to universal usability

- Usage of icons in many user interfaces which require user input, such as camera button in 'My Calories' page, edit icon in 'Edit My Calories' page and price tag in 'Food Recommendations' page

3.1.3. Offer informative feedback

- In the side bar menu, the name of the page will be bold accordingly to which page the user is on
- All buttons change from black to white colour when clicked

3.1.4. Design dialog to yield closure

- When a user adds a new food item, a message showing '<food name> is successfully added' is displayed

3.1.5. Offer simple error handling

- Error messages are shown when users enter information in the wrong format (such as for dates and number of calories)

3.1.6. Permit easy reversal of actions

- Option of retaking photo is provided
- A 'back to top' option is provided in most pages
- A 'cancel' option is provided in most pages to allow the user to go back to previous page

3.1.7. Support internal locus of control

- The application ensures that users control all interface changes

3.1.8. Reduce short-term memory load

- All user inputs are shown on screen once they have entered their inputs, including text fields and food consumed

3.2. Hardware Interfaces

The application is designed to run on an Android device.

3.3. Software Interfaces

1. Android 11.0 (Operating System)
2. Eclipse Version 4.12
3. Firebase (Database)

3.4. Communications Interfaces

This product requires the following communication functions:
1. E-mail
2. Internet access (HTTP protocol) to allow data to be transferred between system and various APIs (Food database API, Food recognition API)

4. System Features

4.1. Record Food Consumption

4.1.1 Description and Priority

This feature allows the user to record the types and weights of foods during the meal and returns the meal records when queried.

Priority: High

4.1.2 Stimulus/Response Sequences

- Log in to the application main page
- Go to ‘My Calories Page’
- Click on either ‘upload or take a picture’ or ‘input meal details’

4.1.3 Functional Requirements

REQ-1. The system must output the calorie and nutrition information of a food specified by the name during the query.

REQ-1.1. The system must allow the user to add custom foods to the system database.

REQ-1.1.1. The system must take the input including the name of food in string and its nutrient information in non-negative float numbers.

REQ-1.1.2. The system must ensure the name of the food is unique.

REQ-1.2. The system must allow the user to input text string as the name of one food for query.

REQ-1.2.1. The system must ensure the string's length is less than 50 characters.

REQ-1.3. The system must allow the user to input an image for query.

REQ-1.3.1. The system must ensure the image is in .png or .jpg format.

REQ-1.3.2. The system must return the foods' names in the image with help of food recognition API

REQ-1.3.3. The system must return the foods' nutritional information with help of food database API.

REQ-1.3.4. The system must support a single query with at least 5 different food in an image.

REQ-2. The system must allow the user to add meal records.

REQ-2.1. The system must prevent any meal record without any food.

REQ-2.2. The system must ensure weights of foods are positive float number in the unit of gram(g).

REQ-2.3. The system must output the calorie and nutrition information of the meal in a table format after a meal record is created.

REQ-2.3.1. The system must output the calorie and nutrition information for each food consumed in a table.

REQ-2.3.2. The system must output total calorie and amounts of each type of essential nutrients consumed for that meal.

REQ-3. The system must return all meal records within the time range specified by the query.

REQ-3.1. The system must process the time range from the start date to end date in the format as "dd/mm/yyyy-dd/mm/yyyy".

REQ-3.2. The system must allow user to query meal records for up to 3 months ago.

REQ-4. The system must allow the user to edit any meal records.

REQ-4.1. The system must support to add foods to a meal record.

REQ-4.2. The system must support to remove foods to a meal record.

REQ-4.3. The system must support to change the weight of a food.

4.2. Healthy Food Recommendation

4.2.1 Description and Priority

This feature aims to analyse the nutrients' intake in order to give recommendations on healthy foods based on the user's meal records.

For each query on healthy food, 10 types of foods will be returned.

Priority: High

4.2.2 Stimulus/Response Sequences

- Log in to main application page
- Go to 'Food Recommendations' page
- Click on 'Food with nutrients you need'

4.2.3 Functional Requirements

REQ-1. The system must calculate suggested daily intake of all types of nutrients based on user's health information when queried.

REQ-2. The system must calculate the average daily intake of nutrients from the past meal records in the past 1 week when queried.

REQ-3. The system shall report all types of nutrients that are consumed 40% lower than the suggested level as lacked nutrients.

REQ-4. The system shall report all types of nutrients that are consumed 40% higher than the suggested level.

REQ-5. The system shall find Top 10 nutrients with smallest proportion of actual average daily intake in suggested average daily intake when queried.

REQ-6. The system must output Top 5 foods rich in the nutrient by using the food database API, when the nutrient's name is provided as input during the query.

REQ-7. The system must output 10 foods that are rich in lacked nutrients.

REQ-7.1. The system must guarantee the foods are not duplicated.

REQ-7.2. The system must display each food with the name and the needed weight.

REQ-7.2.1. The system must process the weight in gram(g).

4.3. Track Health Statistics

4.2.1 Description and Priority

This feature aims to help the user to record the user's health information and calorie intake to help user maintain or reach their target weight. It also visualizes the calorie intake statistics.

Priority: Very High

4.2.2 Stimulus/Response Sequences

- Log in to the application main page

- Go to ‘My Calories Page’

4.2.3 Functional Requirements

REQ-1. The system must calculate suggested daily calorie intake based on user’s health information and target weight.

REQ-1.1. The system must allow user to edit desired daily calorie intake.

REQ-2. The system must calculate the daily calorie intake, based on meal records on the day queried.

REQ-3. The system must calculate remaining calorie quota today.

REQ-4. The system must calculate average calorie intake within the week.

REQ-5. The system must display the health statistical summary report when the user queries.

REQ-5.1. The system must display suggested daily calorie intake.

REQ-5.2. The system must display historical calorie intake data in bar chart within a week.

REQ-5.2.1. The system must highlight the days or weeks in bar chart when the actual calorie intake exceeds the suggested by 40 percent.

REQ-5.2.2. The system must highlight the days or weeks in bar chart when the actual calorie intake is below the suggested by 40 percent.

REQ-5.3. The system must display remaining calorie quota today.

REQ-5.4. The system must display average calorie intake within the week.

REQ-6. The system must display all meal records on a day when the user queries.

REQ-7. The system shall notify the user whether the user met the daily calorie intake at the 23:59 daily.

REQ-8. The system must allow the user to edit the health information.

REQ-9. The system must have a database to contain user's health information.

REQ-10. The system must update the changed health information on the server after the user set up or edit the health information.

4.4. Account

4.2.1 Description and Priority

This feature aims to make the user own an account to hold important data, including health information, meal records, etc.

Priority: Very high

4.2.2 Stimulus/Response Sequences

- Click on 'Create Account'
- Choose either 'Create Account', 'Create Account via Facebook' or 'Create Account via Google'

4.2.3 Functional Requirements

REQ-1. The system must have a database to hold the account information, including the username, address, real name, and password.

REQ-2. The system must be able to register the user.

REQ-2.1. The system must require the user to input the username, the real name, the email address, as well as password and confirmed password, when the user registers with the email address.

REQ-2.1.1. The system must guarantee the input username and real name are within 20 characters.

REQ-2.1.2. The system must guarantee the email addresses contains "@" and ".".

REQ-2.1.3. The system must guarantee the email addresses and usernames are unique.

REQ-2.1.4. The system must verify the registration email address after the user submits the account information.

REQ-2.1.5. The system must ensure the password has at least 8 characters.

REQ-2.2. The system must allow user to register via Google account or Facebook account instead of via email address directly.

REQ-23.2.1 The system must synchronize user information, such as username and email address, through Google or Facebook account and store user information in the database.

REQ-2.3. The system must add the newly registered account to the database.

REQ-2.4. The system must guarantee that user fills the health information, including weight, goal weight, height, age, gender and daily activity level after the account is created.

REQ-3. The system must require the user to log in before using this application.

REQ-3.1. The system must support log-in with email address or username.

REQ-3.2. The system shall direct the user to registration page if the account does not exist.

REQ-3.3. The system must verify the correctness of the password.

REQ-3.4. The system must support log-in with Facebook account and Google account.

REQ-4. The system must allow the user to change the password.

REQ-4.1. The system must require the user to provide the username or email address.

REQ-4.2. The system must send an email with automatically generated verification code to the registration email address.

REQ-4.2.1. The system must ensure the verification code is a alphanumeric strings with 8 characters.

REQ-4.3. The system shall update the password to the new password input by the user after checking the verification code input by the user is correct.

REQ-4.4. The system shall remember the account log-in information.

5. Other Nonfunctional Requirements

5.1. Performance Requirements

REQ-1. The system must recognize the types of foods from the input image within 1.5s.

REQ-2. Food Database API should return the calorie and nutritional information within 1.5s.

REQ-3. The database must return the past meal records when queried within 1.5s.

REQ-4. The system must guarantee to display the page within 1.5s during the interface jump.

REQ-5. The system must ensure that when a user login or signs up via Google account or Facebook account, the redirection to the external interface can be done within 3s.

REQ-6. The system must ensure that the certification email will be sent to the correct email address within 5s when a user signs in or change password via email.

5.2. Security and Privacy Requirements

REQ-1. The system must encrypt the user account information, health information and meal records.

REQ-2. The database must be encrypted at rest.

REQ-3. The system must obtain the user’s permission before connecting mobile data or Wi-Fi network.

REQ-4. The system must obtain the user's permission before accessing user's camera and album.

REQ-5. The system must ensure that no data from database, especially for user information like password, email address, health information and meal records, will be transmitted to unauthenticated sources.

REQ-6. The system must ensure that the application will never disclose user information to users that are not authorized to see.

REQ-7. The system must only allow logged-in users to use the application.

5.3. Software Quality Attributes

REQ-1. The system must display the user's guide when the user launches the app at the first time.

REQ-2. The system must allow user to reverse an action easily.

REQ-3. The system must specify input requirements for user inputs such as password and food weights clearly.

REQ-4. The system must display appropriate messages that guide the user when exceptions occur. For example, the system shall ask the user to add custom foods to database when there is no result returned after query.

REQ-5. The system must allow the user to turn off notifications from the app.

REQ-6. The system must guarantee the food database API contains at least 5000 types of foods.

5.4. Other Requirements

REQ-1. Database Requirements

- The database must have enough storage space to store 1000 users' past meal and calorie records for at least 3 months.
- The system must store user's account information, health information, suggested calorie intake, and meal records to the database.

REQ-2. Legal Requirements

- The application must not contain any information which can be sensitive to any individual other than the health information provided by the user.
- The images and icons used shall have a legal copyright status.

6. Appendix A: Data Dictionary and Glossary

Term	**Definition**
App	The app refers to the application and is an abbreviated word form of application.
System	The system refers to the application.
User	A user is an individual using the application.
Email	Email simply refers to email address.
Healthy food recommendation	Food recommendation based on the amount of nutrients the user lacks
Calorie	A calorie is a unit of measurement describing how much energy your body could get from eating or drinking it.
Nutrients	The essential nutrients for well-being. Specifically speaking, this term refers to fat, cholesterol, sodium, potassium, sugar, dietary fibre, protein, calcium, vitamin C, iron, cobalamin, and magnesium.
Meal	Contains one or more Food. Example includes Chicken Rice, which includes roasted chicken meat and rice as Food
Food	The smallest unit making up a meal for which its nutritional value has been determined. Example includes roasted chicken meat, rice, egg
Food database API	The database that given the input of foods' names, outputs the nutrient information.
Remaining calorie quota	The number of calories left which user can consume within that day without exceeding the suggested calorie intake.
Social media account	Specifically refers to Facebook and Google accounts.
Social media account API	API provides by Facebook and Google to connect with the user's social media account.
Health information	Consists of weight, goal weight, height, age, gender and daily activity level.

7. Appendix B: Analysis Models

7.1. Use Case Model

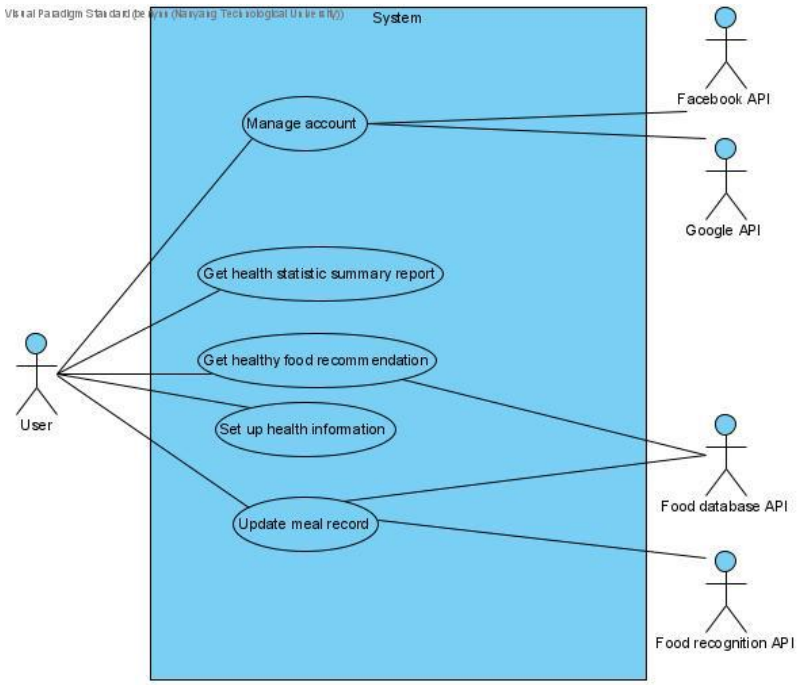

Figure 7.1.1: Main Use Case Diagram

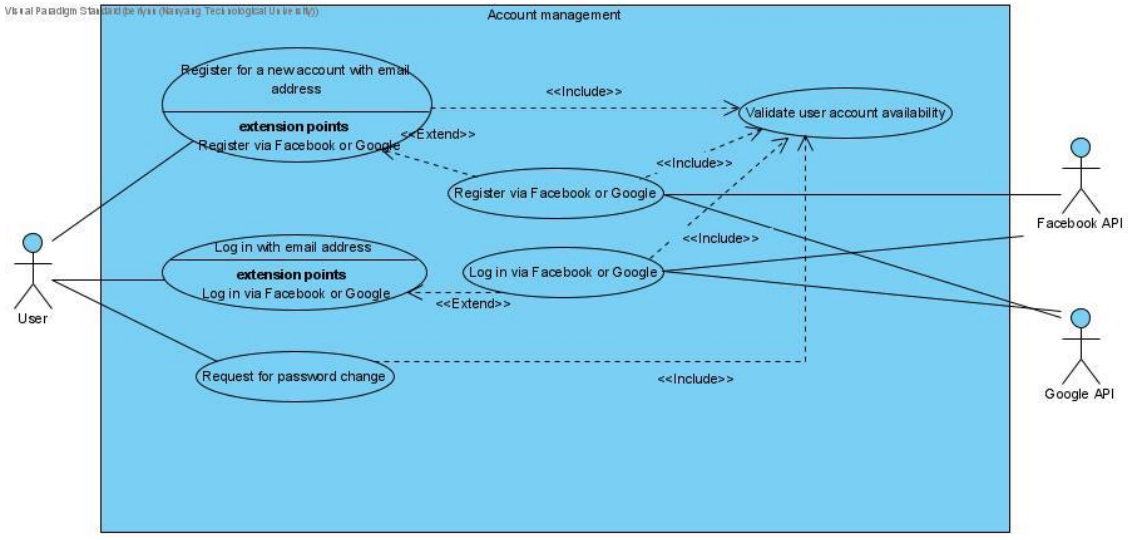

Figure 7.1.2: Use Case Diagram for Account Management

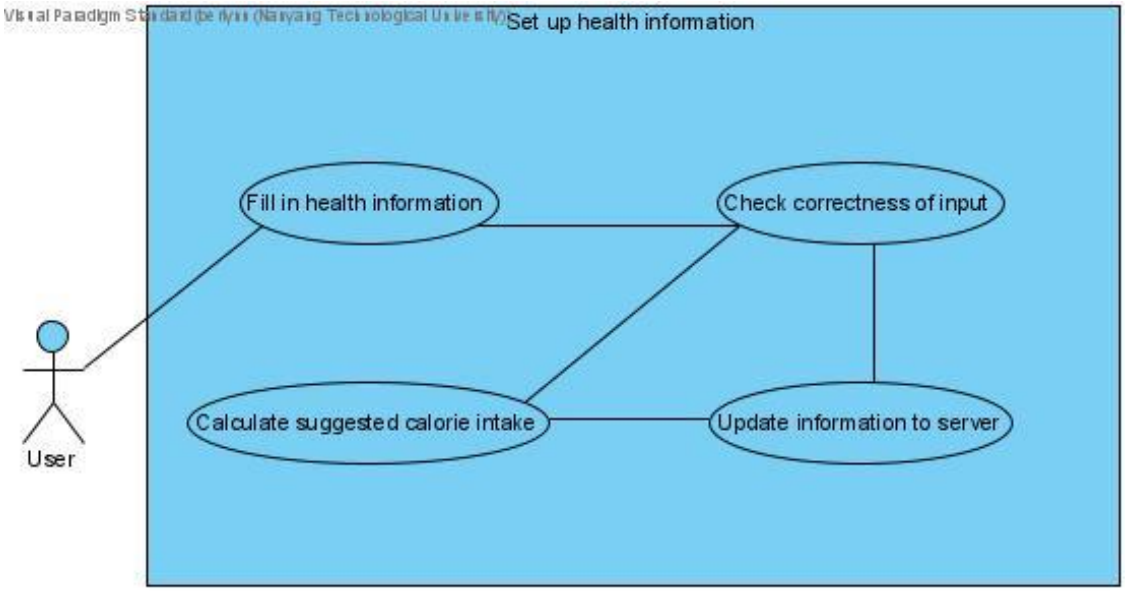

Figure 7.1.3: Use Case Diagram for Setting Up Health Information

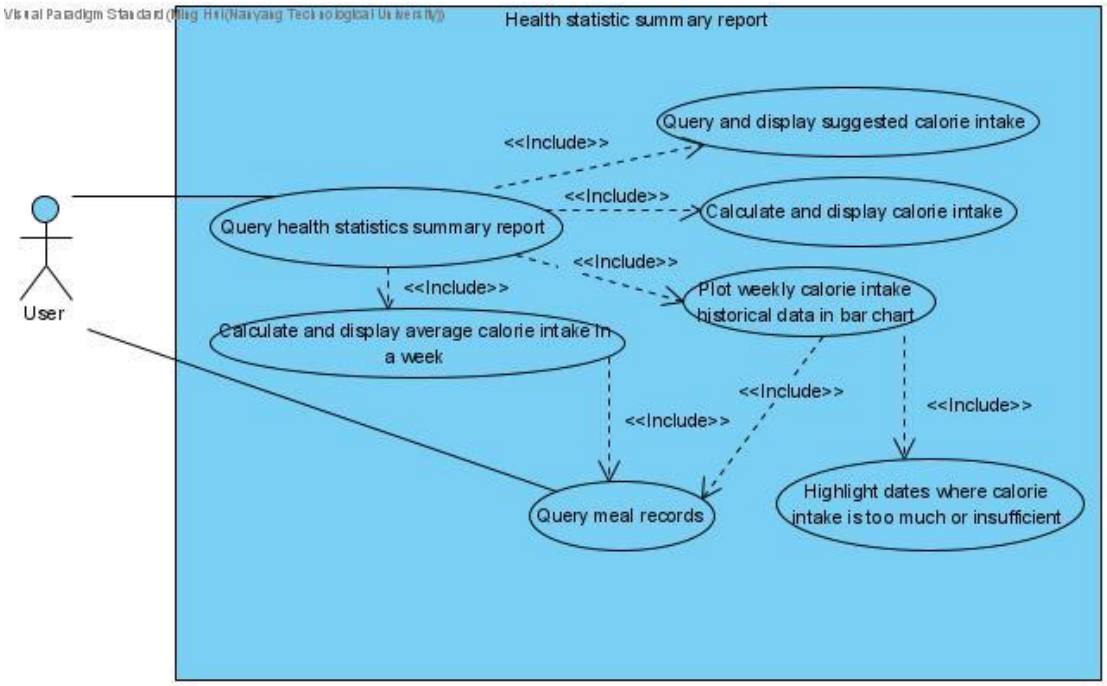

Figure 7.1.4: Use Case Diagram for Health Statistic Summary Report

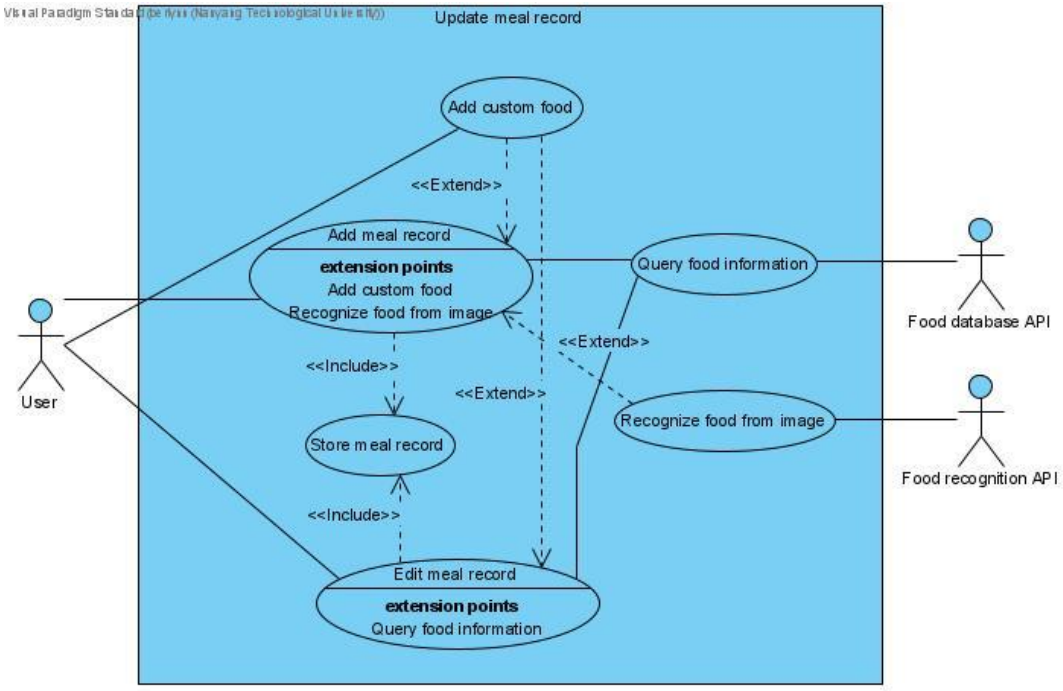

Figure 7.1.5: Use Case Diagram for Updating Meal Record

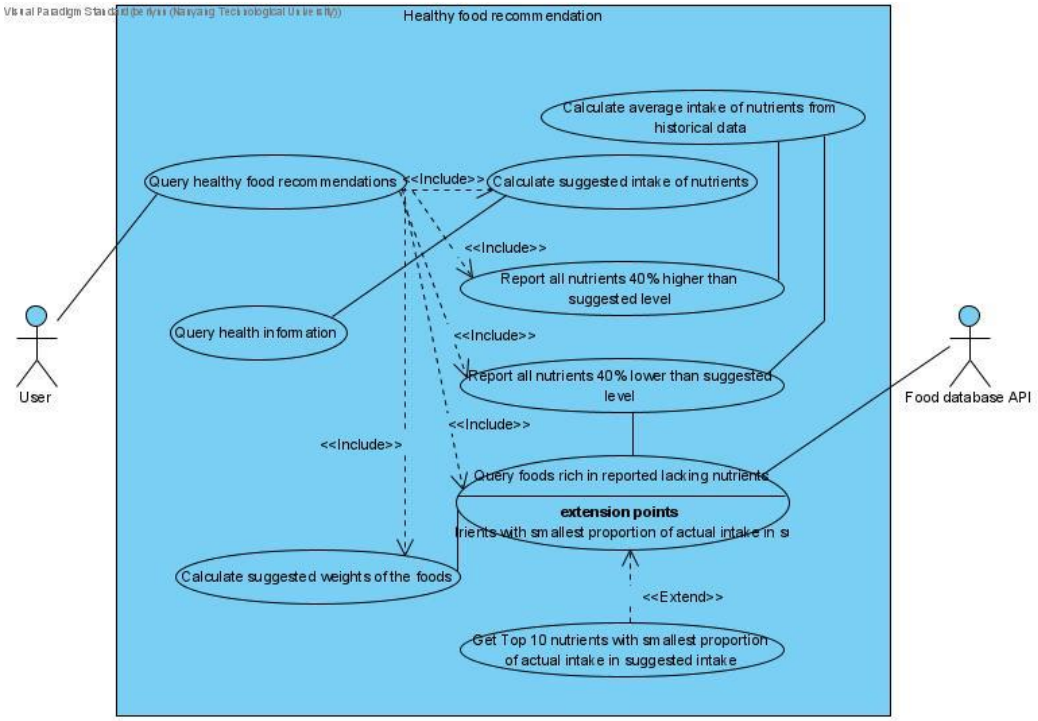

Figure 7.1.6: Use Case Diagram for Healthy Food Recommendation

7.2. Use Case Description

7.2.1. Account Management

Use Case ID:	1		
Use Case Name:	Register a new account with email address.		
Created By:	Neo Rui Xuan Berlynn	Last Updated By:	TANG YUTING
Date Created:	02/02/2021	Date Last Updated:	19/02/2021

Actor:	User
Description:	This use case allows users to register a new account with email address.
Preconditions:	1. User device must have Internet access. 2. The user is not logged in.
Postconditions:	1. The user account is in the database. 2. The user is redirected to the page where they need to fill the health information.

	3. The successful registration message is printed.
Priority:	Very high
Frequency of Use:	1-2 times per lifetime
Flow of Events:	1. User clicks “register an account”. 2. User enters a valid username, first name, last name, email address, password and confirm password fields. 3. System verifies the account by checking there are no duplications email address and the username in the database. 4. System sends the verification email to the email. 5. If User enters correct verification code, System adds the newly registered account to the database. 6. System prints successful registration message. 7. The system redirects the user to fill the health information.
Alternative Flows:	-
Exceptions:	**User email address is already connected to an existing account, the fields are invalid, or password and confirmed password are different.** 1. Print the corresponding error messages.
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Use Case ID:	2		
Use Case Name:	Register account via Facebook or Google		
Created By:	Neo Rui Xuan Berlynn	Last Updated By:	TANG YUTING
Date Created:	02/02/2021	Date Last Updated:	18/02/2021

Actor:	User
Description:	User registers for a new user account via his Facebook or Google account.
Preconditions:	1. User device must be connected to Wi-Fi or mobile data.
Postconditions:	1. The user’s information is in the database. 2. The successful registration message is printed.
Priority:	Very high
Frequency of Use:	1-2 times per lifetime
Flow of Events:	1. User clicks the “Create account via Facebook” or “Create account via Google” button. 2. User is validated by Google/Facebook API. 3. System extracts user’s data from his Facebook or Google account.

	4. System validates user account availability with the database. 5. System registers the user into the database. 6. System displays the successful registration message. 7. System redirects the user to set up health information.
Alternative Flows:	-
Exceptions:	**User email address is already connected to an existing account, the fields are invalid, password and confirmed password are different.** 1. Print the corresponding error messages.
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Use Case ID:	3		
Use Case Name:	Log in with email address		
Created By:	Neo Rui Xuan Berlynn	Last Updated By:	TANG YUTING
Date Created:	03/02/2021	Date Last Updated:	04/02/2021

Actor:	User
Description:	User logs in to his account with email address.
Preconditions:	1. User device must have Internet access. 2. User is not logged in.
Postconditions:	1. User logs in to his account.
Priority:	Very high
Frequency of Use:	3-4 times per day
Flow of Events:	1. User clicks “log in”. 2. User enters his username or email address and password in the appropriate fields. 3. User clicks the “Log in” button. 4. System checks that the user account exists in the database and that the password entered is correct. 5. User logs in to his account on our application.
Alternative Flows:	-
Exceptions:	**User enters username or email address that does not exist in database.** 1. System displays error message “Username does not exist. Please try again.” or “Email address does not exist. Please try again.” respectively.

	User enters incorrect password when matched with user's password stored in the database. 1. System displays error message "Incorrect password. Please try again."
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Use Case ID:	4		
Use Case Name:	Log in via Facebook or Google		
Created By:	Neo Rui Xuan Berlynn	Last Updated By:	TANG YUTING
Date Created:	03/02/2021	Date Last Updated:	18/02/2021

Actor:	User
Description:	User logs in to his account via Facebook or Google.
Preconditions:	1. User device must have Internet access.
Postconditions:	1. User logs in to his account.
Priority:	Very high
Frequency of Use:	3-5 times a day
Flow of Events:	1. User clicks the "Log in via Facebook" or "Log in via Google" button. 2. System calls Google/Facebook API to validate the user's account. 3. If the validation is correct, System logs in User.
Alternative Flows:	-
Exceptions:	**The account is not in the system database.** 1. Print error message "this account has not been registered." 2. Direct the user to register with Facebook/Google account. **The account fails to be validated.** 1. Print error message "Log in fails".
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Use Case ID:	5		
Use Case Name:	Request for password change		
Created By:	Neo Rui Xuan Berlynn	Last Updated By:	Neo Rui Xuan Berlynn

Date Created:	02/02/2021	Date Last Updated:	02/02/2021

Actor:	User
Description:	User requests for password change if they forget their password or for privacy reasons.
Preconditions:	1. User device must be connected to WiFi or mobile data.
Postconditions:	1. System updates user's password and stores it in the database. 2. User can log in with new password.
Priority:	High
Frequency of Use:	1-5 times per year
Flow of Events:	1. User clicks on "Forgot/Change Password?" button. 2. User enters the username or email address linked to their account and clicks the "Request" button. 3. System validates user account availability with the database. 4. System sends a verification code to the email address linked with the user account. 5. User enters the verification code and clicks the "Submit" button. 6. System validates the verification code. 7. If the validation is passed, User enters the new password and confirms new password and clicks the "Submit" button. 8. System checks if the new password is the same as what is entered in the confirm new password field and if it is the same as the old password. 9. System saves the new password in the database.
Alternative Flows:	**User links Google/Facebook account. (Step 1)** Step 2-6 is replaced by "System calls API to validate User account."
Exceptions:	**The username or email address entered by the user is not found in the database.** 1. System displays an error message "Username is not found. Please try again." or "Email address is not found. Please try again." respectively. **The verification code entered is invalid.** 1. System displays an error message "Invalid verification code." **System detects that the new password and confirm new password fields are different.** 1. System displays an error message "New Password and Confirm New Password are not the same. Please try again."

	System detects that the new password is the same as the old password. 1. System displays an error message "New Password is same as your previous password. Please try again."
Includes:	-
Special Requirements:	-
Assumptions:	Database can refer to System, Facebook or Google's account database.
Notes and Issues:	-

7.2.2. Setting Up Heath Information

Use Case ID:	6		
Use Case Name:	Set up health information		
Created By:	Wang Binli	Last Updated By:	TANG YUTING
Date Created:	02/02/2021	Date Last Updated:	04/02/2021

Actor:	User
Description:	User sets up the health information, including weight, height, age, gender, daily activity level and goal weight.
Preconditions:	1. User health information must not be in the database. 2. User device must have Internet access.
Postconditions:	1. User health information is added and stored in the system database. 2. System updates the suggested daily calorie intake to the server.
Priority:	Very high
Frequency of Use:	1-2 times per lifetime
Flow of Events:	1. User enters health information including gender, age, height, weight, goal weight and activity level. 2. User clicks the "Submit" button. 3. System checks the correctness of the input. 4. System calculates new suggested calorie intake value. 5. System stores user health information and new suggested calorie intake value into the database.
Alternative Flows:	-
Exceptions:	**The health information is incorrect.** 1. Highlight the box to inform the user to correct input. **The health information provided by user is not complete.**

	System displays an error message “Please fill all blanks in order to complete your health information.”.
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Use Case ID:	7		
Use Case Name:	Edit account health information.		
Created By:	Wang Binli	Last Updated By:	TANG YUTING
Date Created:	02/02/2021	Date Last Updated:	04/2/2021

Actor:	User
Description:	User edits account health information.
Preconditions:	1. User account with complete health information must already be in the database. 2. User must be already logged in. 3. User device must have Internet access.
Postconditions:	1. User health information will be updated in the database. 2. The suggested daily calorie intake must be changed correspondingly.
Priority:	High
Frequency of Use:	1-4 times per month
Flow of Events:	1. User enters the “Account and Setting” interface. 2. User clicks blanks and modifies the health information that needed to be updated. 3. Users clicks the “Submit button”. 4. System checks the correctness of the input. 5. System calculates new suggested daily calorie intake value. 6. System updates the user health information and new suggested calorie intake value in the database.
Alternative Flows:	-
Exceptions:	**The health information is incorrect.** 1. Highlight the box to inform the user to correct input. **The health information provided by user is not complete.** System displays an error message “Please fill all blanks in order to complete your health information.”.
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

7.2.3. Health Statistics Summary Report

Use Case ID:	8		
Use Case Name:	Query health statistics summary report.		
Created By:	Wang Binli	Last Updated By:	TANG YUTING
Date Created:	02/02/2021	Date Last Updated:	04/02/2021
Actor:	User		
Description:	User can query the health statistics summary report consists of suggested daily calorie intake, calorie quota remaining and weekly calorie historical intake data in bar chat.		
Preconditions:	1. User health information must already be in the database. 2. User must already login. 3. User device must have Internet access.		
Postconditions:	1. The health statistics summary report displayed.		
Priority:	High		
Frequency of Use:	1-4 times each day		
Flow of Events:	1. User enters the "My calorie" interface to query the health statistics report. 2. System queries meals' records within a week (including today). 3. System queries and displays the suggested daily calorie intake. 4. System calculates and displays the calorie quota remaining. 5. System plots the weekly calorie historical intake data in bar chat with days, when the actual intake is too high or too low, highlighted. 6. System calculates and displays average calorie intake within a week.		
Alternative Flows:	-		
Exceptions:	**When the user does not have any meal record** 1. Raise error and ask the user to add at least 1 meal record.		
Includes:	-		
Special Requirements:	-		
Assumptions:	1. The suggested intake is purely determined by the user's health information. 2. When system highlights dates, 40% is the threshold to determine if the calorie intake is lacking or excessive compared with suggested.		
Notes and Issues:	The order for System displays calorie quota remaining, System displays weekly calorie historical intake data in bar chat and		

	System displays average calorie intake within a week can be switched.

7.2.4. Update Meal Record

Use Case ID:	9		
Use Case Name:	Add meal record with text input		
Created By:	Chua Ming Hui	Last Updated By:	TANG YUTING
Date Created:	4/2/21	Date Last Updated:	4/2/21

Actor:	User, food database API
Description:	This allows user to record their meals in the system by input text to calculate calories and nutrition information in the future.
Preconditions:	1. User is logged in. 2. System has Internet access.
Postconditions:	1. System displays the calorie and nutrition information of each food using data in Food database API 2. System displays the total calorie and total amount of each nutrition for that meal 3. System stores meal record in database
Priority:	High
Frequency of Use:	Almost Everyday
Flow of Events:	1. User clicks “input meal details”. 2. User adds names and weights of foods. 3. System queries food database API for foods’ nutrient information. 4. System adds the meal record to the server. 5. User can delete the food record. 6. User confirms the change. 7. System adds the meal record to the database. 8. System displays the information of the added meal record.
Alternative Flows:	**The food is not found in the food database. (An additional step is changed between Step 3 and 4)** 1. Ask the user to add custom foods for all not found foods, as specified in User Case 11.
Exceptions:	**The weight of food is not positive integer.** Print error message “Weight of food is incorrect.”
Includes:	Add custom food to user’s database
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Use Case ID:	10		
Use Case Name:	Add meal record with image input		
Created By:	Chua Ming Hui	Last Updated By:	Ng Pek Han
Date Created:	4/2/21	Date Last Updated:	4/2/21

Actor:	User, food database API, food recognition API
Description:	This allows user to record their meals in the system by giving an image to calculate calories and nutrition information in the future.
Preconditions:	1. User is logged in. 2. System has Internet access.
Postconditions:	1. System displays the calorie and nutrition information of each food using data in Food database API 2. System displays the total calorie and total amount of each nutrition for that meal 3. System stores meal record in database
Priority:	High
Frequency of Use:	Almost Everyday
Flow of Events:	1. User clicks “upload or take a picture”. 2. System recognizes and displays foods from the image by using food recognition API. 3. User inputs the weights of foods. 4. System queries food database API for foods’ nutrient information. 5. User continues to add foods into the meal record as demonstrated in Use Case 9.
Alternative Flows:	**No foods are recognized by AI in Step 2** 1. Message showing ‘Unable to detect food item’ is displayed 2. Go to Step 5
Exceptions:	-
Includes:	text-input use case (use case 9)
Special Requirements:	1. Food Database API should recognise all the food in the meal within 1.5s. 2. Food Database API should return the calorie and nutritional information in 1.5s.
Assumptions:	-
Notes and Issues:	-

Use Case ID:	11		
Use Case Name:	Add custom food to user’s database		
Created By:	Chua Ming Hui	Last Updated By:	Ng Pek Han
Date Created:	4/2/21	Date Last Updated:	4/2/21

Actor:	User

Description:	This allows user to add custom food that are not originally in Food Database API
Preconditions:	1. User is logged in. 2. Device has Internet access.
Postconditions:	1. Message showing '<Food Name> successfully added' is shown.
Priority:	High
Frequency of Use:	3-4 times a month
Flow of Events:	1. User goes to 'My Calories' tab in app and clicks 'Add custom food'. 2. User enters food name, calories (in kcal), types of nutrients and amount of each nutrient (in g/mg) per serving, and User presses 'submit'. 3. If the input is valid, new food name and information stored in the database. 4. A table showing the entered information is shown.
Alternative Flows:	-
Exceptions:	**Incomplete information of food submitted, or information submitted is in the wrong format (e.g., calories not in integer).** 1. Raise error and ask user to complete the information of food correctly.
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Use Case ID:	12		
Use Case Name:	Query past meals		
Created By:	Chua Ming Hui	Last Updated By:	Ng Pek Han
Date Created:	4/2/21	Date Last Updated:	4/2/21

Actor:	User
Description:	This allows the user to view past meal records up to 3 months ago.
Preconditions:	1. User is logged in. 2. Device has Internet access.
Postconditions:	The meal records are displayed.
Priority:	High
Frequency of Use:	2-3 times a week
Flow of Events:	1. User goes to 'My Meals' tab in app. 2. User inputs the start and end date in "dd/mm/yyyy-dd/mm/yyyy" format.

	3. System queries the meal record. 4. System displays corresponding meal records in a list.
Alternative Flows:	**User has not recorded any meal** 'No meals recorded yet' message will be shown.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Use Case ID:	13		
Use Case Name:	Edit meal record		
Created By:	Chua Ming Hui	Last Updated By:	Ng Pek Han
Date Created:	4/2/21	Date Last Updated:	4/2/21

Actor:	User, Food database API
Description:	This allows the user to add food, remove food or change the weight of food they consume in a meal that has been recorded.
Preconditions:	1. User is logged in. 2. Device has Internet access.
Postconditions:	1. System displays the calorie and nutrition information of each food using data in Food database API 2. System displays the total calorie and total amount of each nutrition for that meal 3. System stores meal record in database
Priority:	Medium
Frequency of Use:	2-3 times a week
Flow of Events:	1. User go to 'My Meal's tab in app. 2. User inputs date of meal in "dd/mm/yyyy - dd/mm/yyyy" format. 3. System displays a list of meals within the timeframe. 4. User selects a meal to edit as demonstrated in the sub-frame in Use Case 9. 5. System removed the old meal record.
Alternative Flows:	-
Exceptions:	**The edited record contains no food.** 1. Notify the user and System does not confirm the edit. **Some food may not be identifiable by the food database API.** 1. Redirect the user to add custom foods.
Includes:	Add custom food to user's database
Special Requirements:	-

Assumptions:	-
Notes and Issues:	-

7.2.5. Healthy Food Recommendation

Use Case ID:	14		
Use Case Name:	Recommend foods rich in lacking nutrients		
Created By:	TANG YUTING	Last Updated By:	TANG YUTING
Date Created:	3/2/21	Date Last Updated:	4/2/21

Actor:	User, food database API
Description:	The user can query 10 recommended healthy foods that supply them with the nutrients they lack based on their past meal records.
Preconditions:	1. The application has Internet access.
Postconditions:	1. 10 recommended healthy foods are displayed.
Priority:	High
Frequency of Use:	About 4-7 times per week
Flow of Events:	1. The user clicks “view food recommendation” button. 2. The system calculates the suggested intake of all nutrients based on user’s health information such as weight, height, gender, age, etc. 3. System calculates the actual intake of nutrients. 4. The system reports all nutrients that are more than 40% higher or lower than the suggested intake after comparing between the actual intake and the suggested intake within one week. 5. The system queries the external food database API for foods rich in nutrients that are more than 40% lower than the suggested intake. 6. The system calculates the suggested intake of the recommended foods. 7. The system returns the 10 food recommendations to the user.
Alternative Flows:	
Exceptions:	**When the user does not have any meal record** 1. Raise error and ask the user to add at least 1 meal record.
Includes:	-
Special Requirements:	-
Assumptions:	1. The suggested intake is purely determined by the user’s health information.

	2. 40% is the threshold to determine if a nutrient is lacking or excessive. 3. The user has at least 10 types of foods not excessively taken, which guarantees that the recommendation will not return excessively taken foods.
Notes and Issues:	-

7.3. Class Diagram

Key boundary classes:	**Key control classes:**
UI	AccountManager
RegisterAccountUI	HealthInfoManager
LoginUI	MealRecordManager
VerificationUI	Recommender
StatisticsUI	
MealRecordMainUI	
InfoSetupUI	
MealRecordUI	
MainUI	
FoodCustomUI	
FoodRecognitionUI	
FoodRecommendationUI	
FoodRecordUI	
PasswordChangeUI	
PastMealRecordUI	

*Images of class diagrams have been uploaded to Tortoise SVN lab3 folder

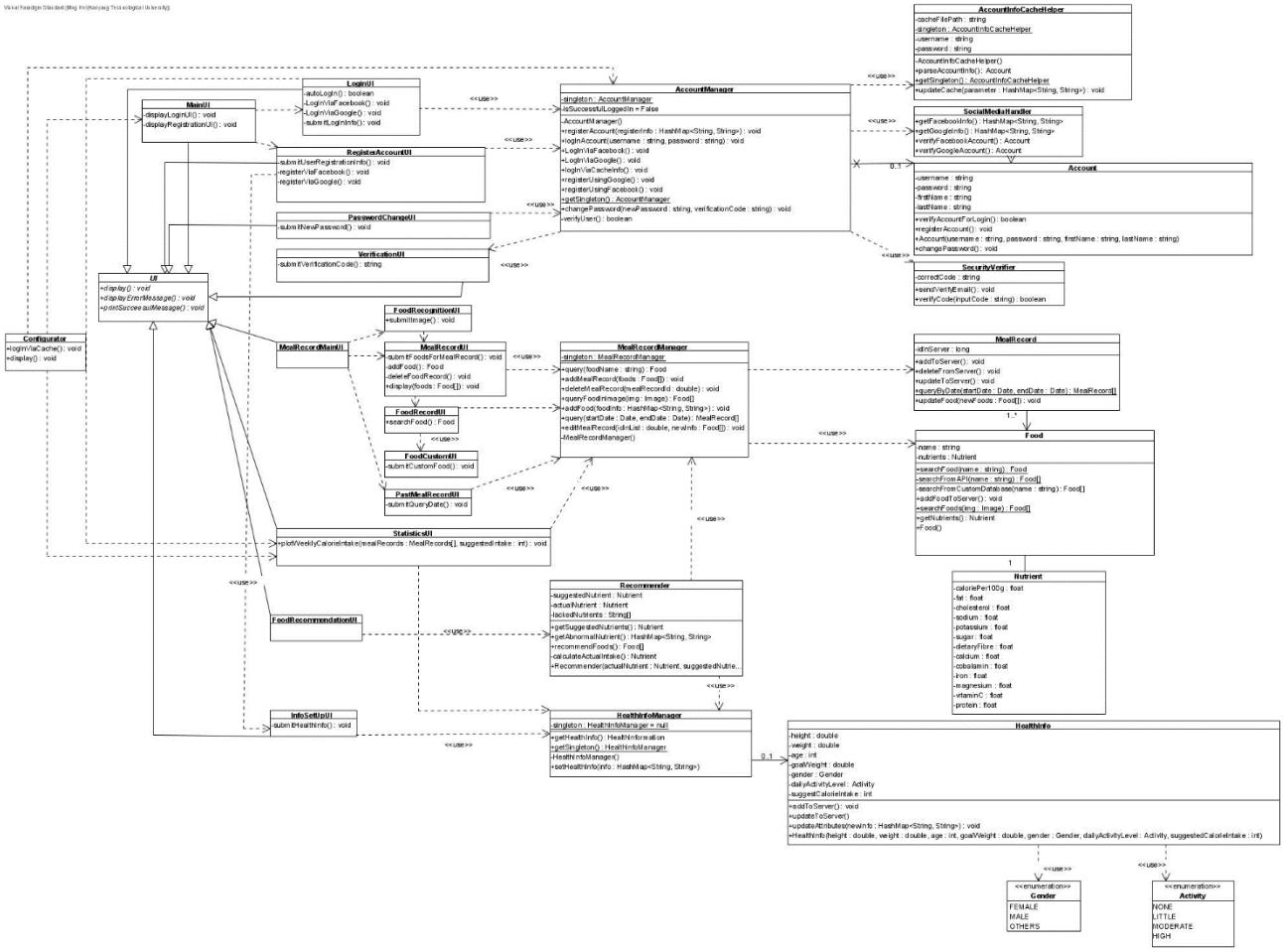

Figure 7.3.1: Main Class Diagram

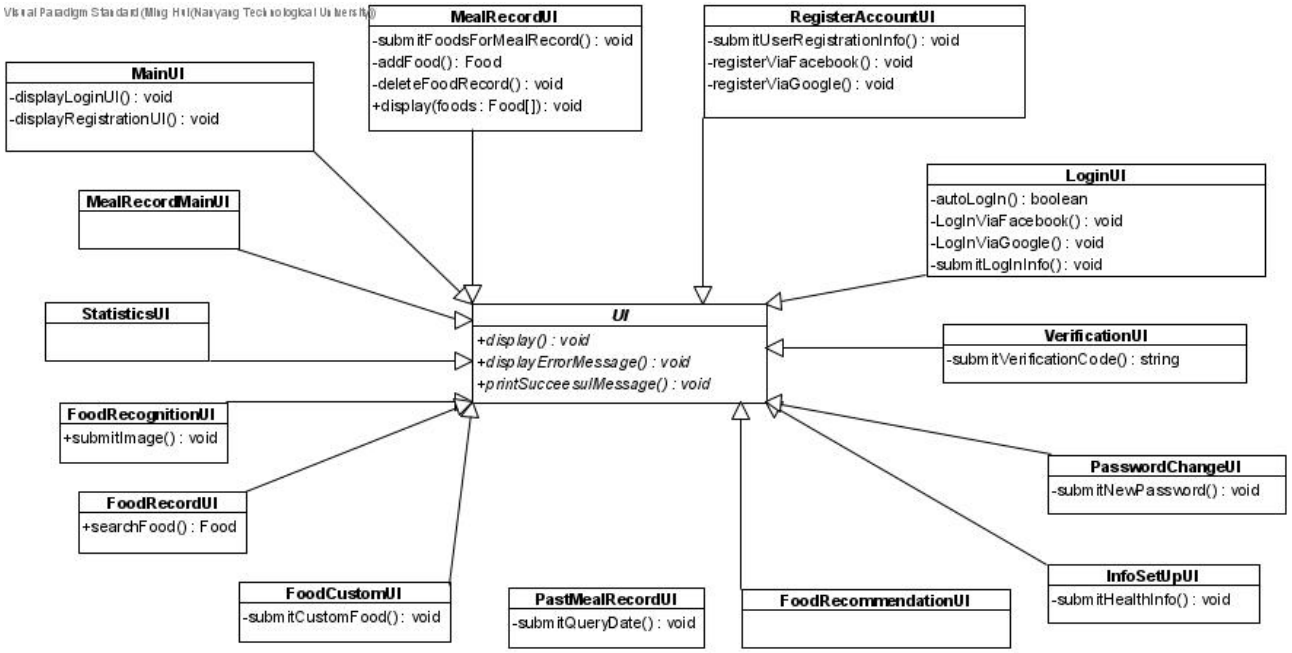

Figure 7.3.2: Interface Class Diagram

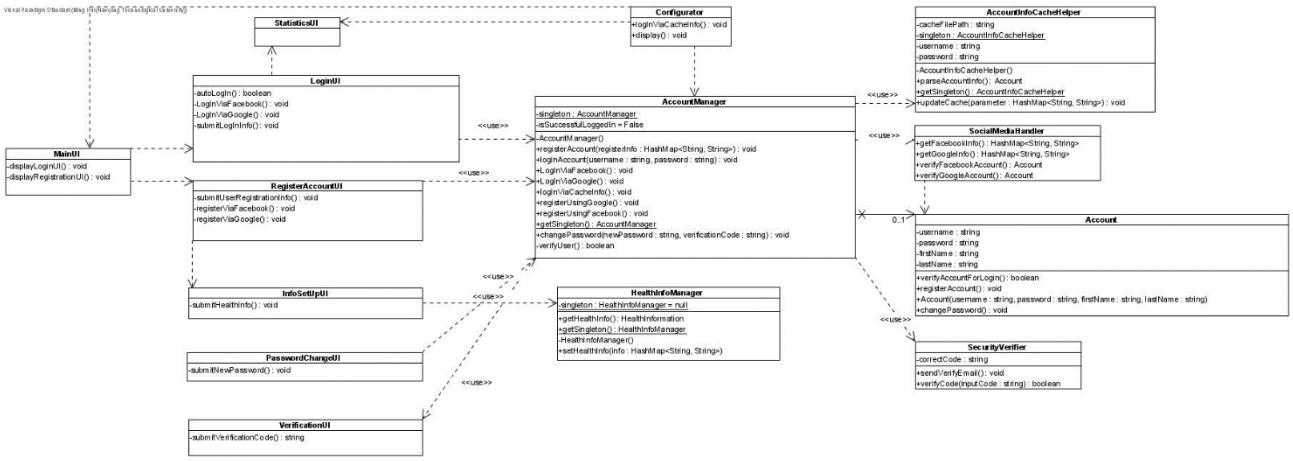

Figure 7.3.3: Account Class Diagram

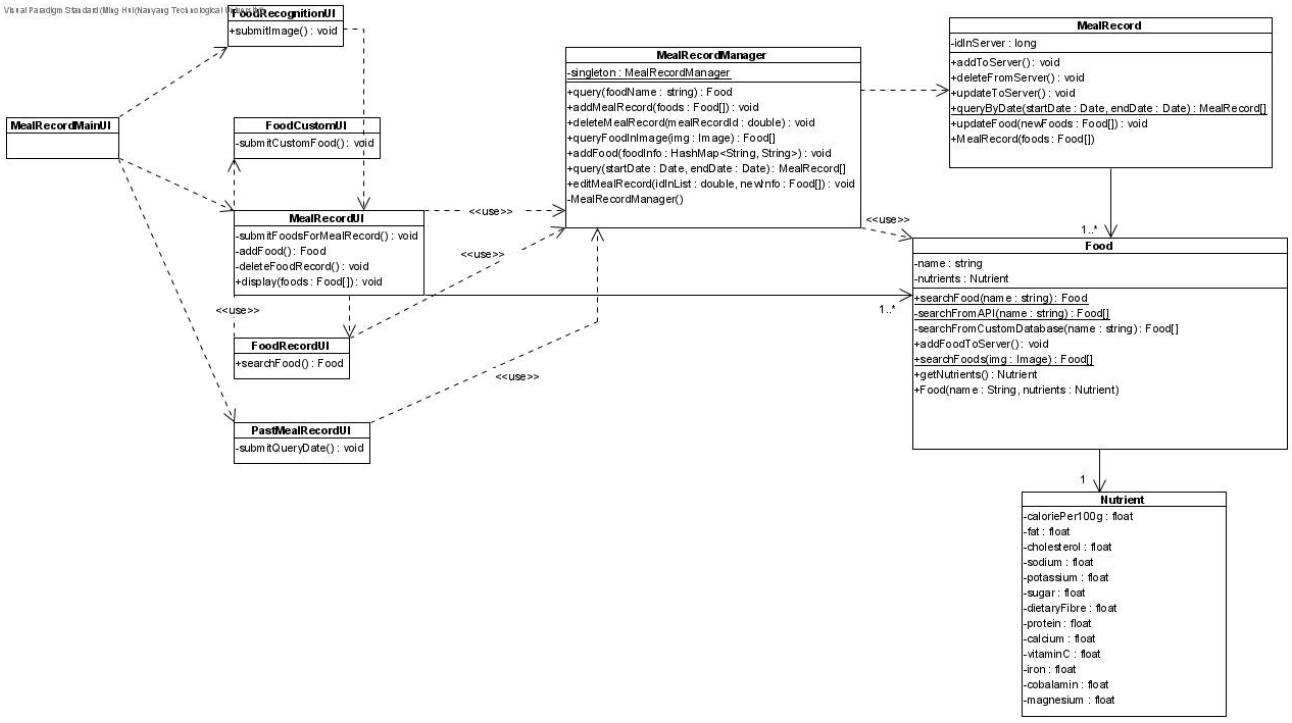

Figure 7.3.4: Meal Record Subsystem Class Diagram

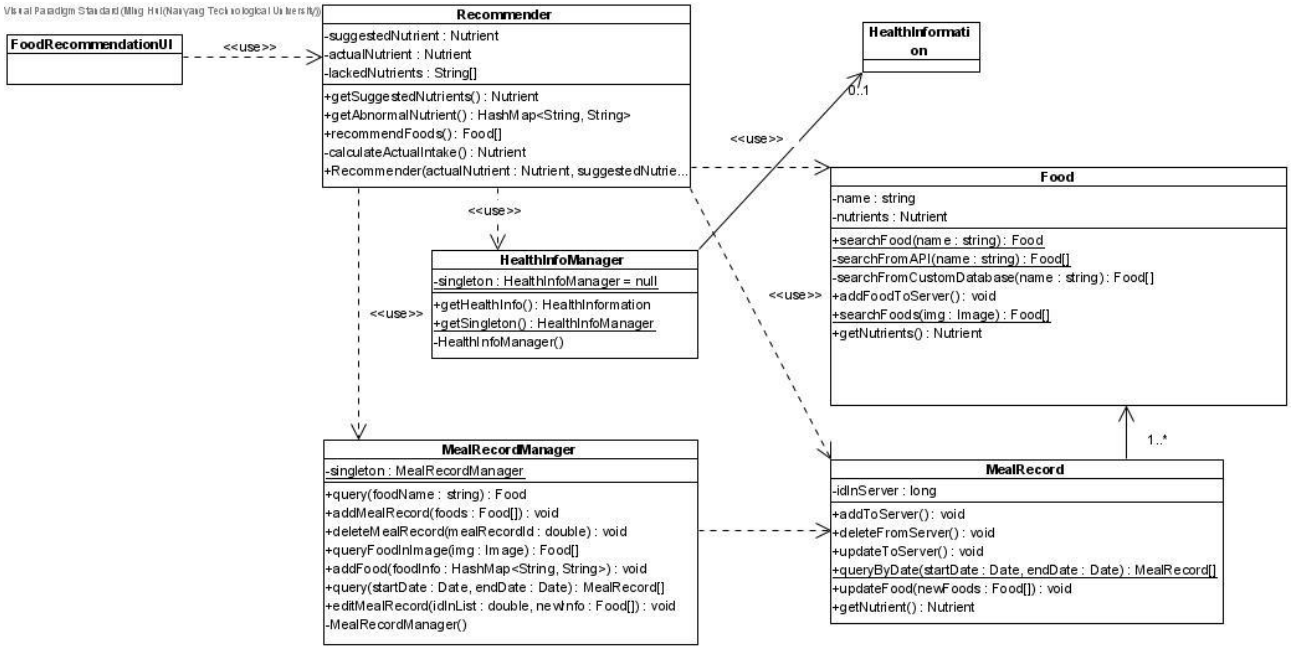

Figure 7.3.5: Food Recommendation Class Diagram

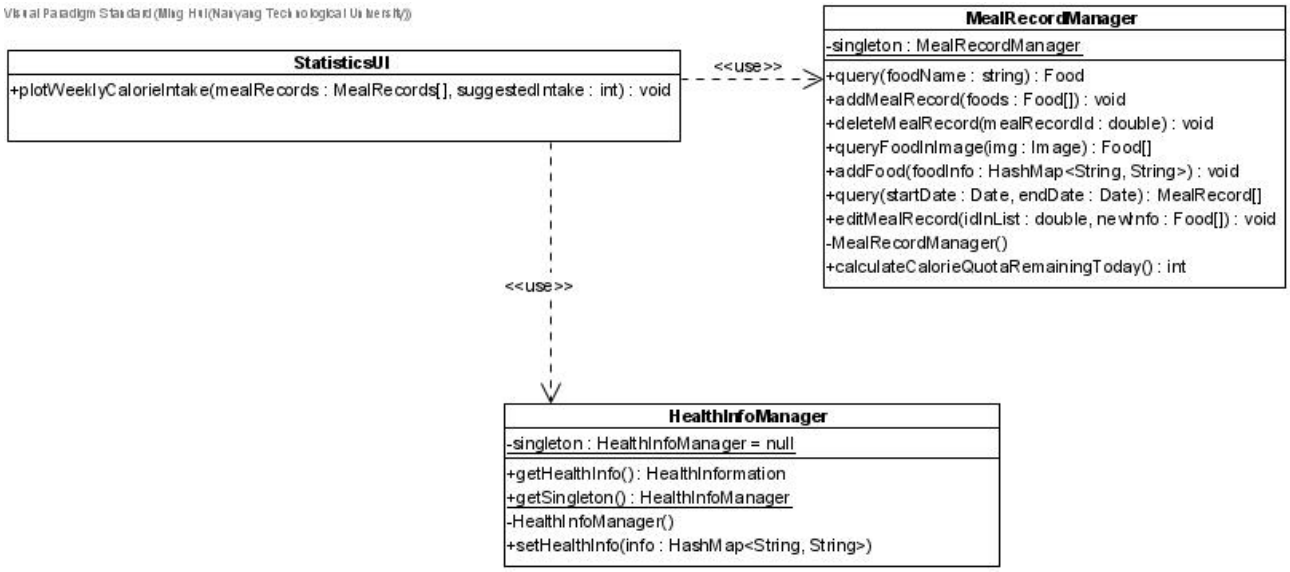

Figure 7.3.6: Health Statistics Class Diagram

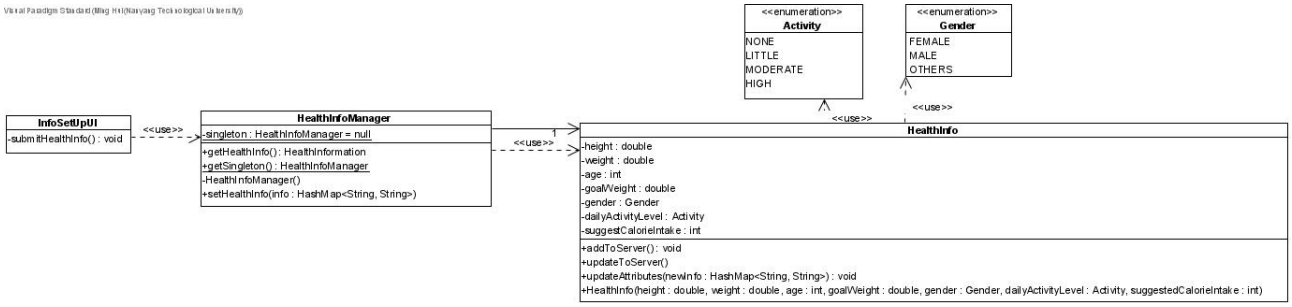

Figure 7.3.7: Health Information Subsystem Class Diagram

7.3.1. Start-up Class

Configurator is the start-up class, which will try logging in the user based on account information stored in local cache. If the user is logged in successfully, configurator launch the statistics interface. If the information stored is invalid, such as wrong password or empty information, the configurator will launch the main menu where the user needs to choose either log-in or register.

7.4. Sequence Diagram

*Images of sequence diagrams have been uploaded to Tortoise SVN lab3 folder

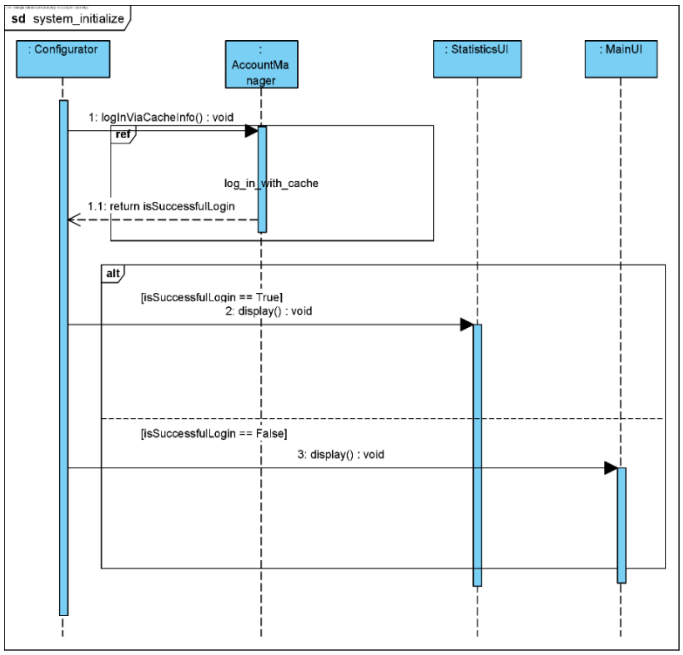

Figure 7.4.1: Sequence diagram of system initialization

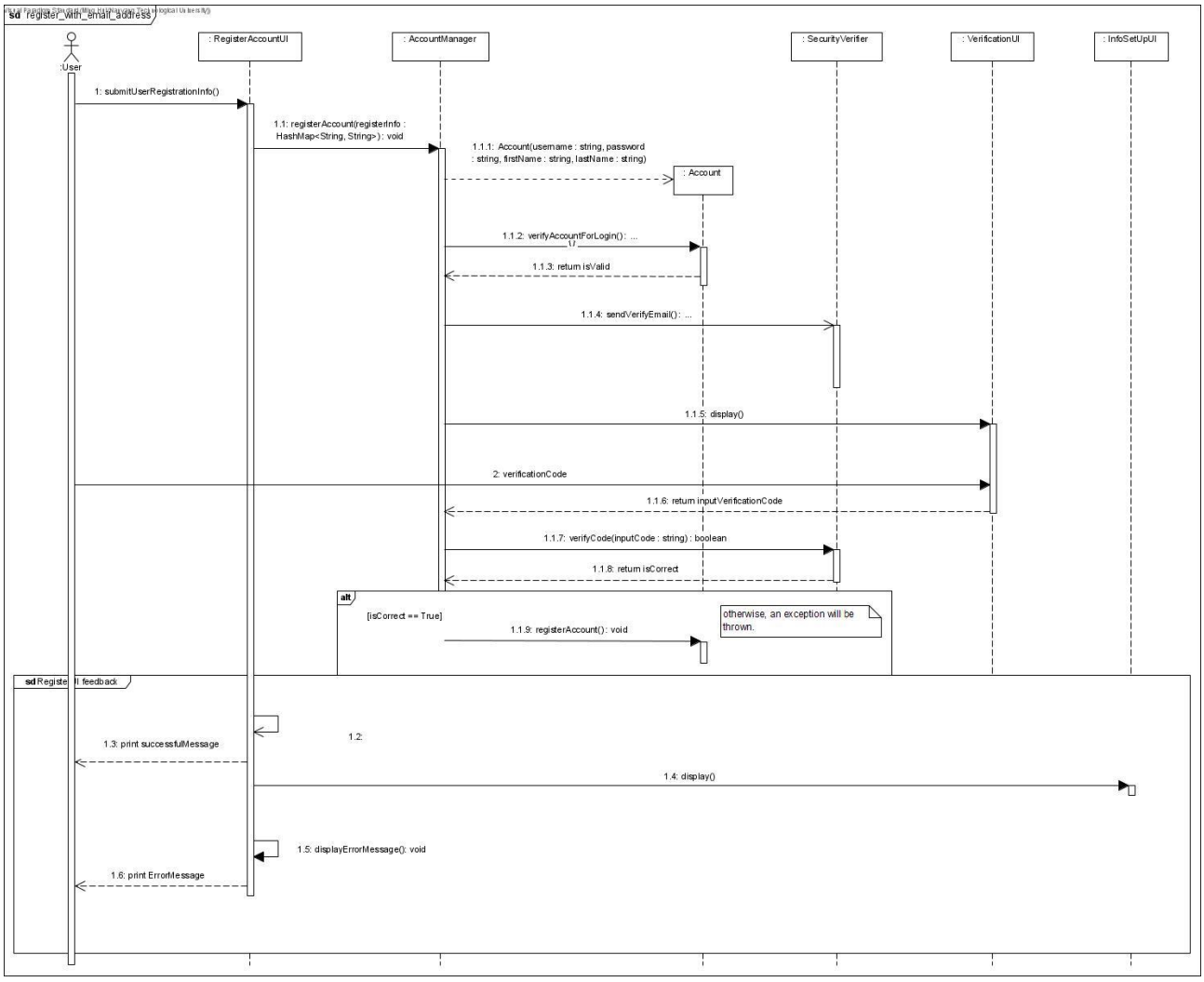

Figure 7.4.2: Sequence Diagram – Register with Email Address

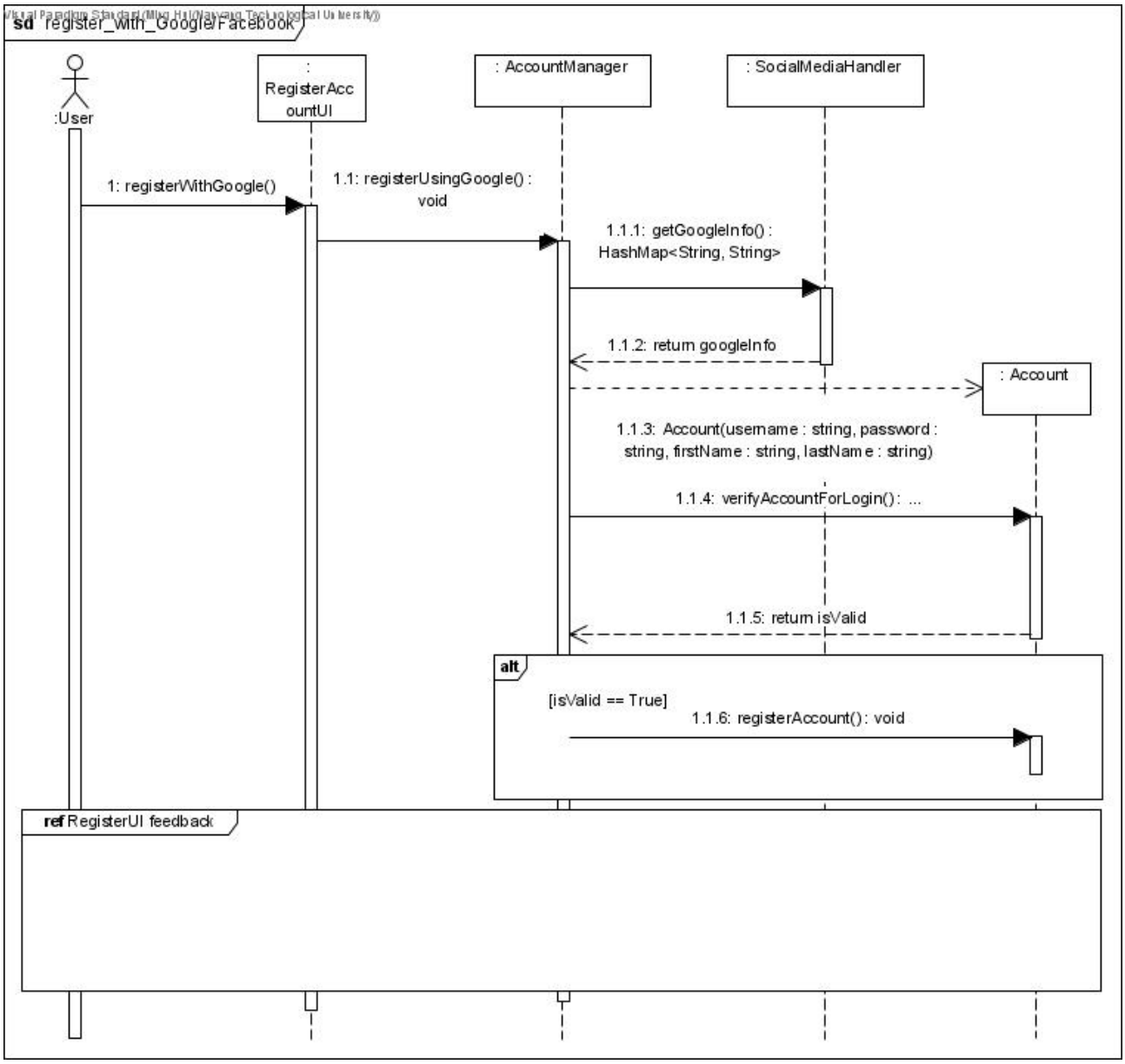

Figure 7.4.3: Sequence Diagram – Register with Social Media

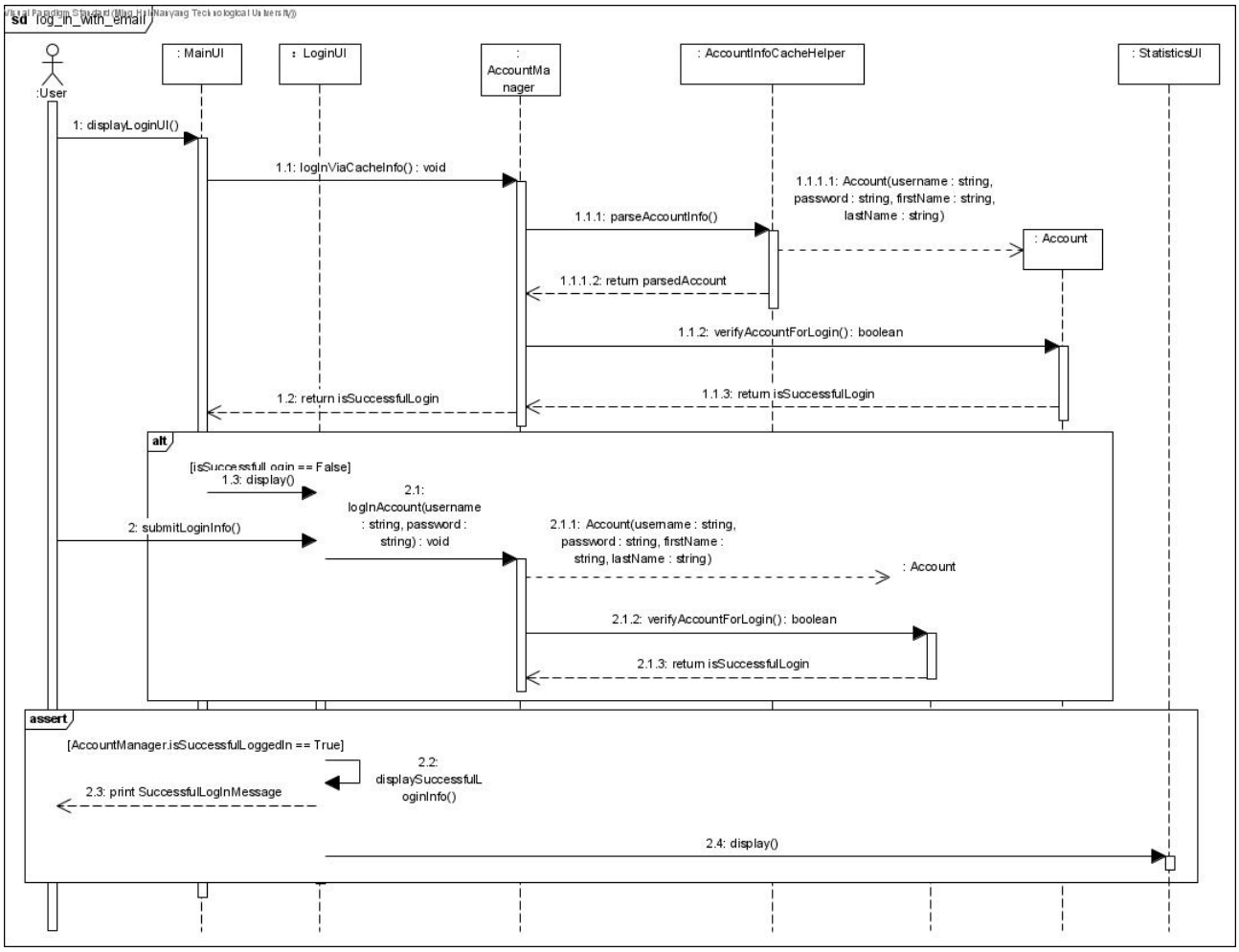

Figure 7.4.4: Sequence Diagram – Log in with Email Address

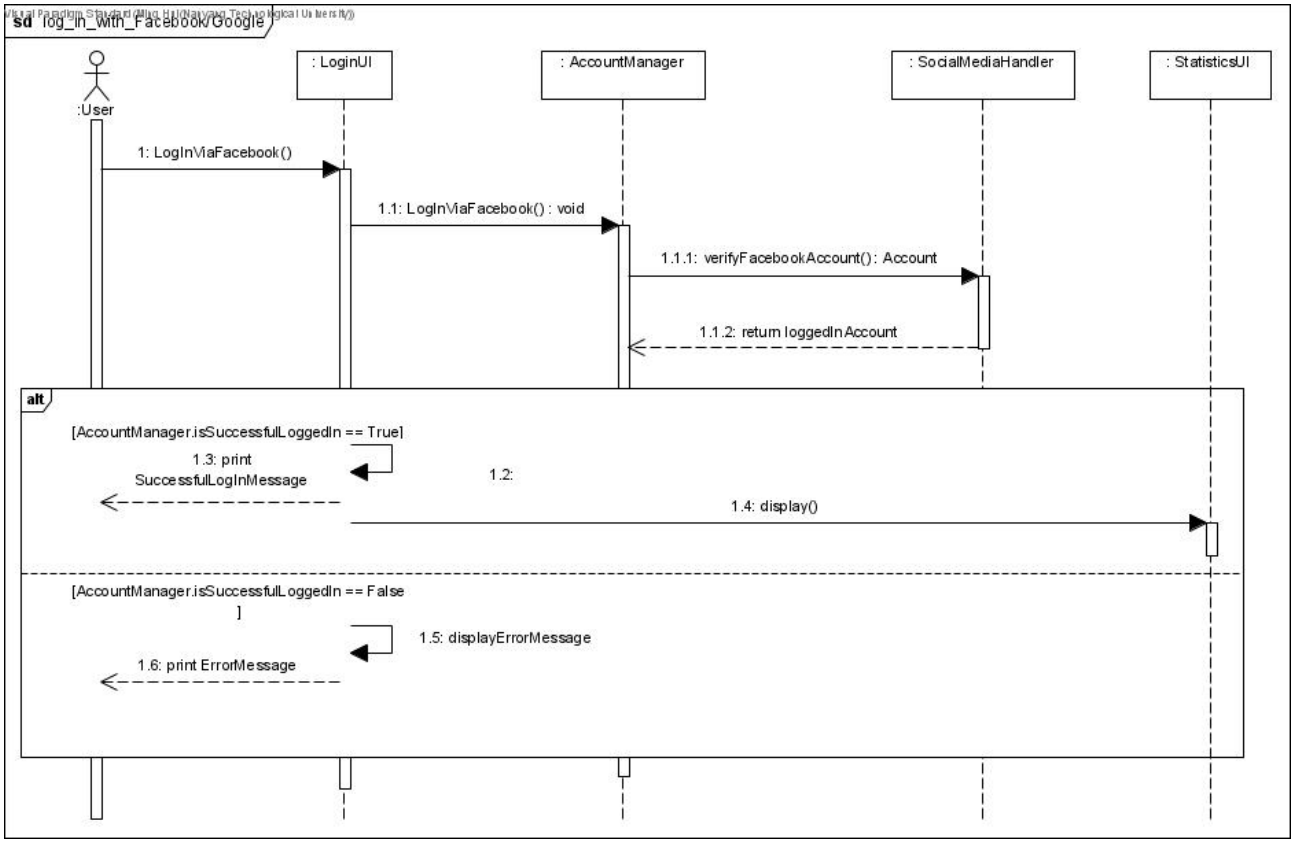

Figure 7.4.5: Sequence Diagram – Log in with Social Media

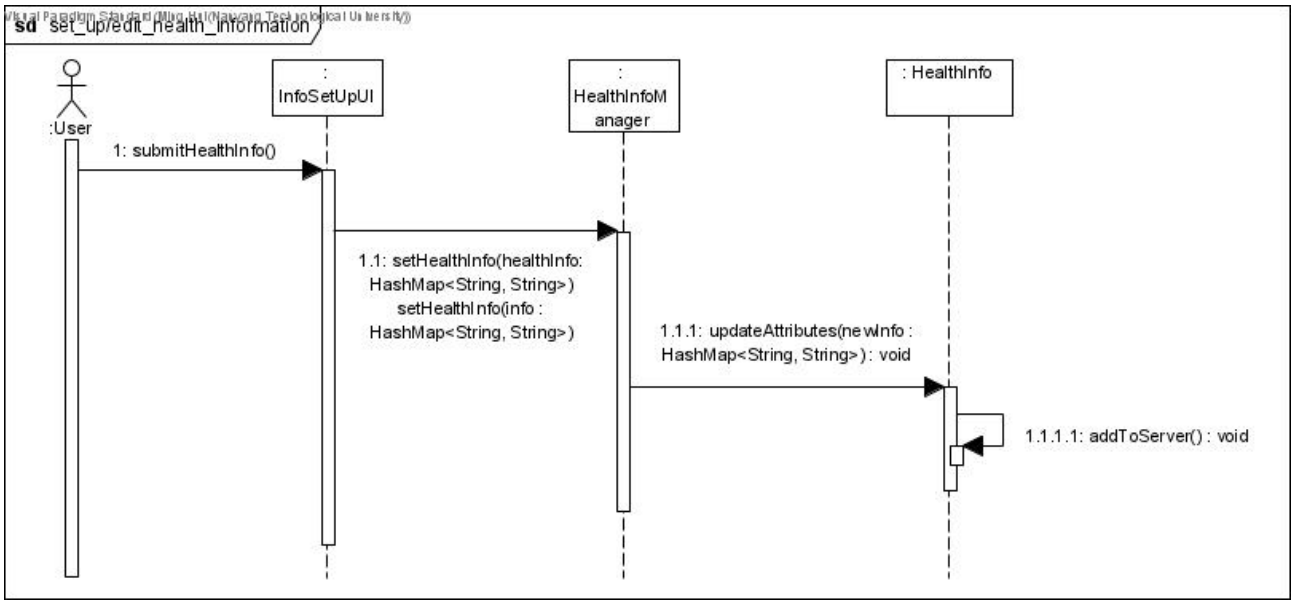

Figure 7.4.6: Sequence Diagram – Set up/Edit Health Information

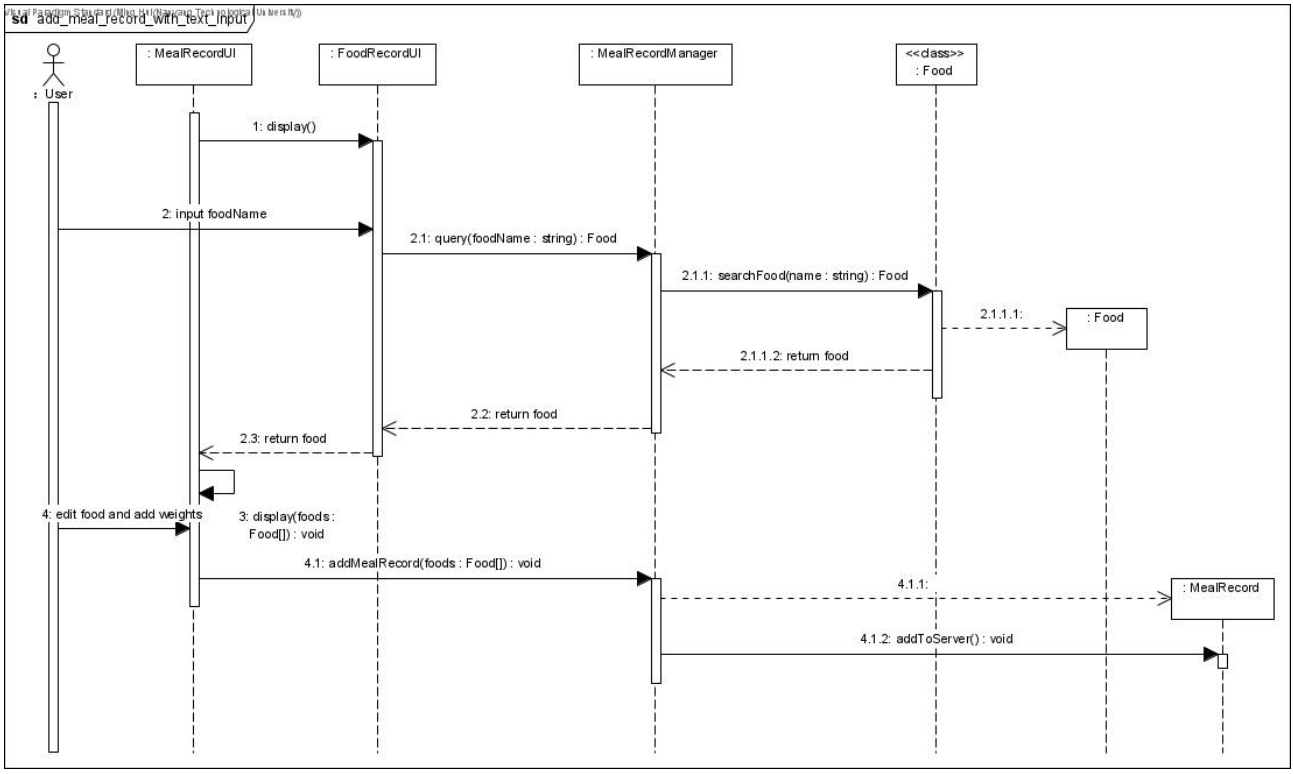

Figure 7.4.7: Sequence Diagram – Add Meal Record with Text Input

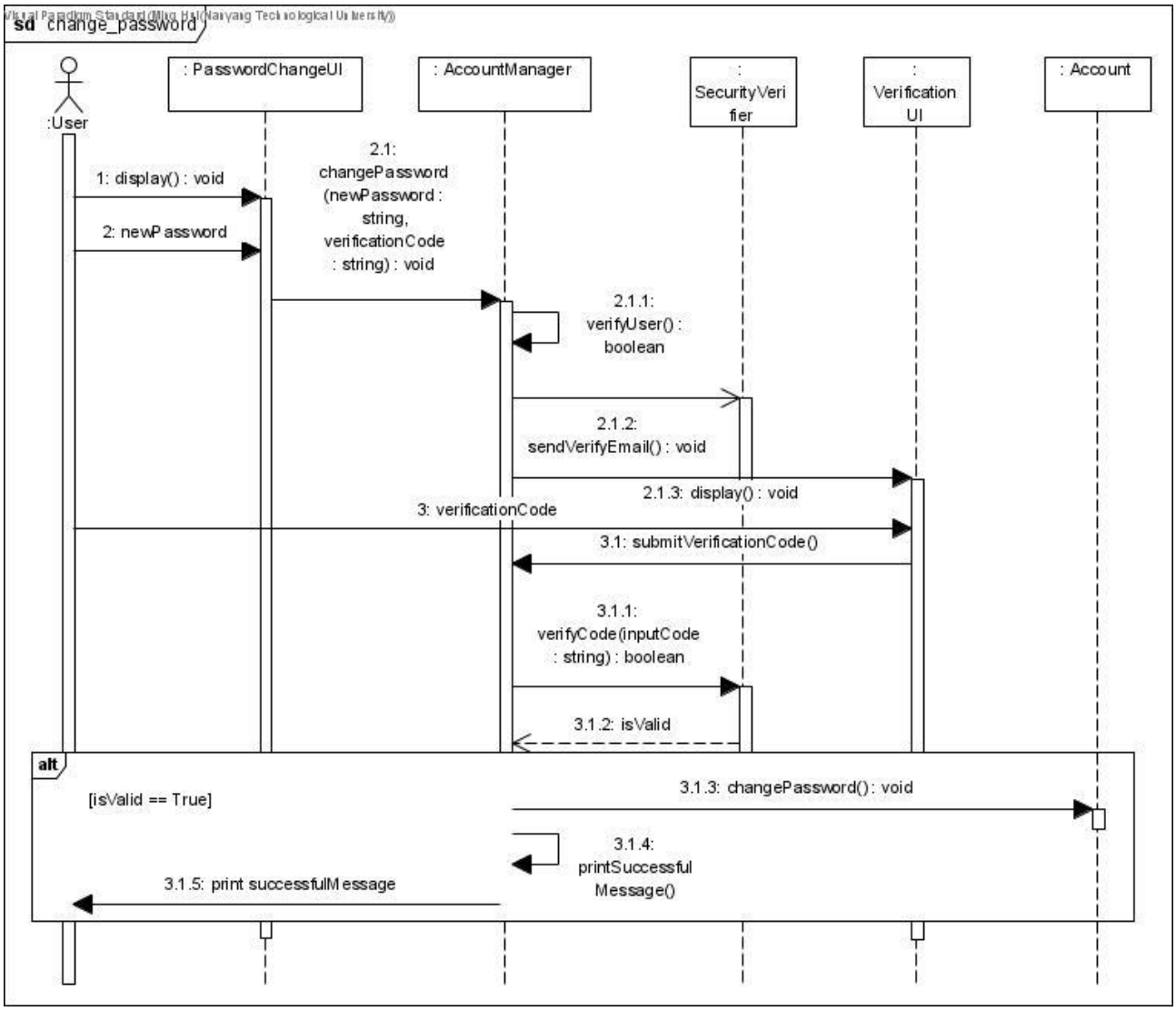

Figure 7.4.8: Sequence Diagram – Change Password

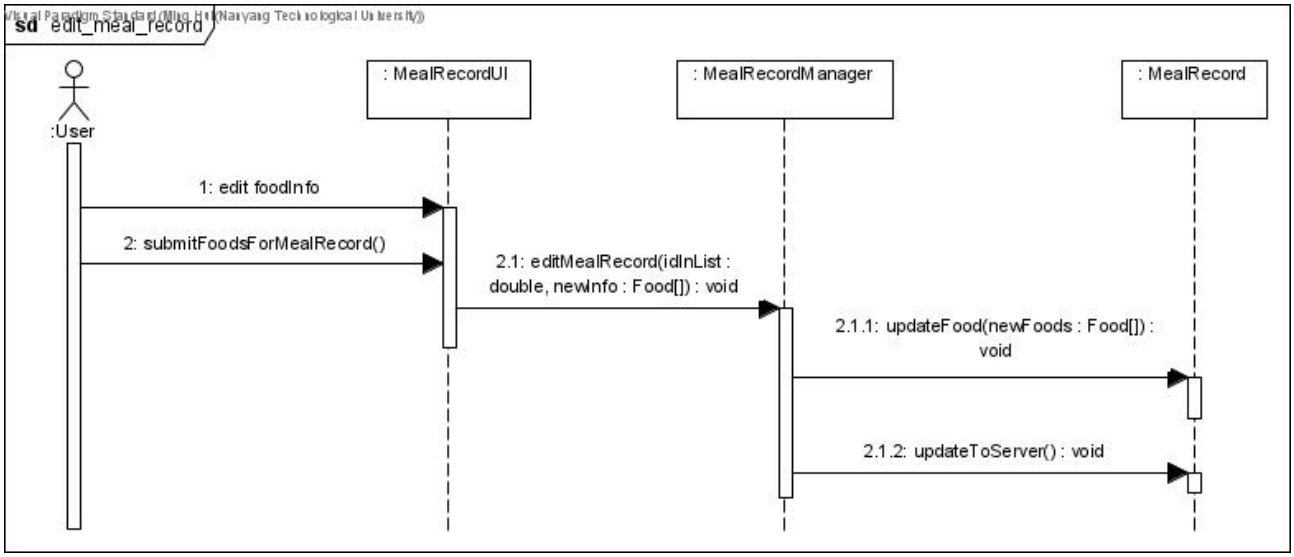

Figure 7.4.9: Sequence Diagram – Edit Meal Record

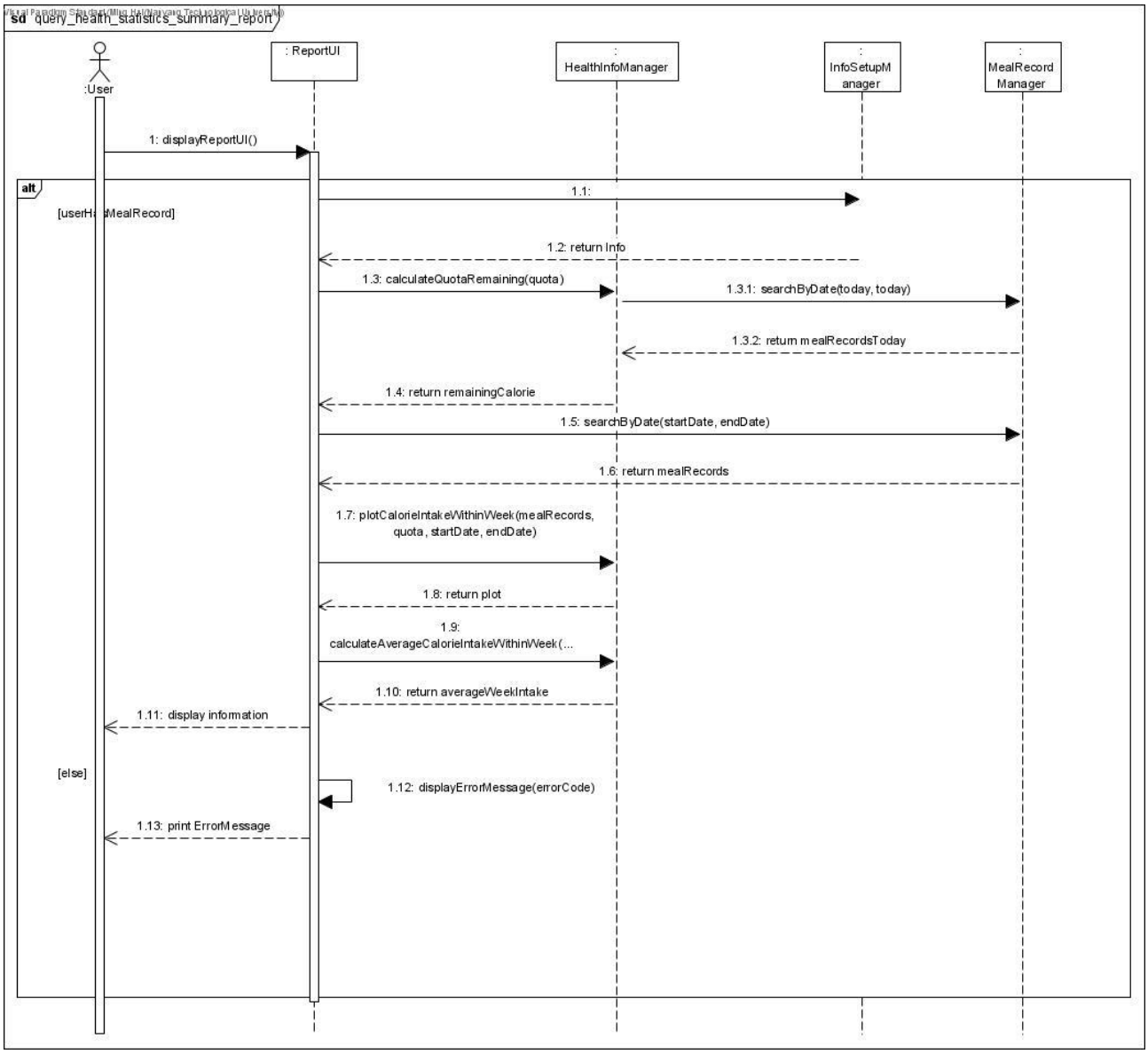

Figure 7.4.10: Sequence Diagram – Query Health Statistics Summary Report

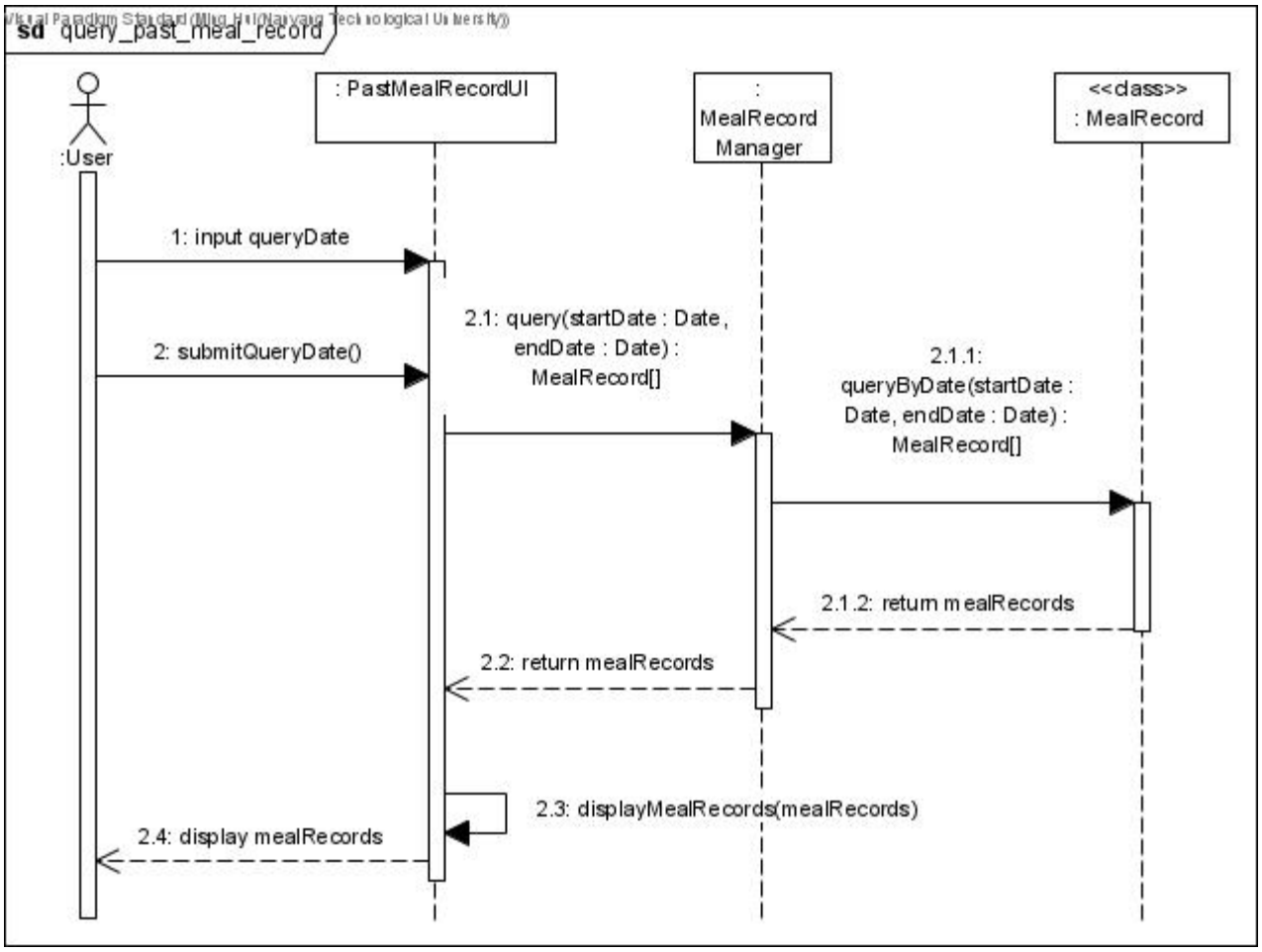

Figure 7.4.11: Sequence Diagram – Query Past Meal Record

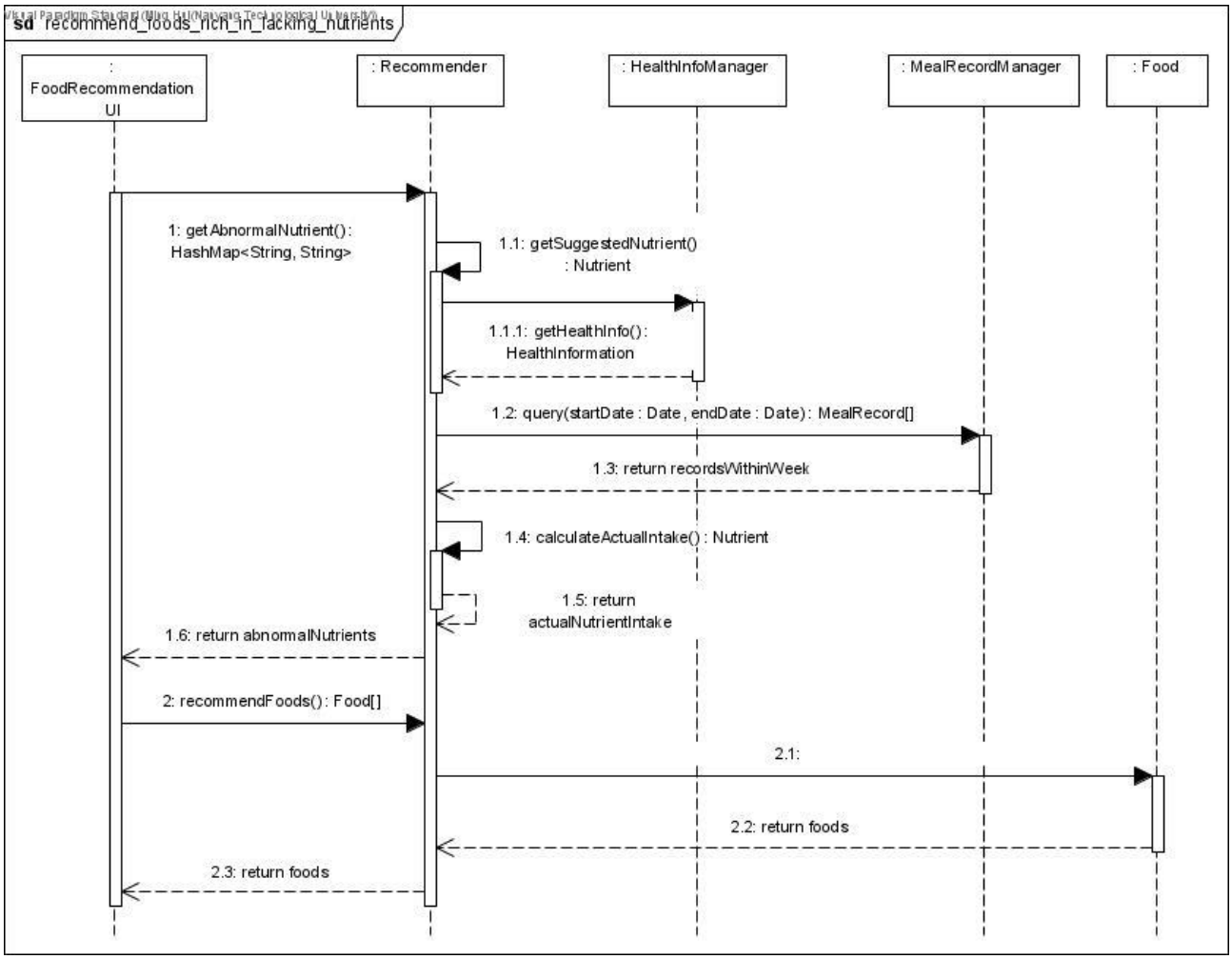

Figure 7.4.12: Sequence Diagram – Recommend foods rich in lacking nutrients

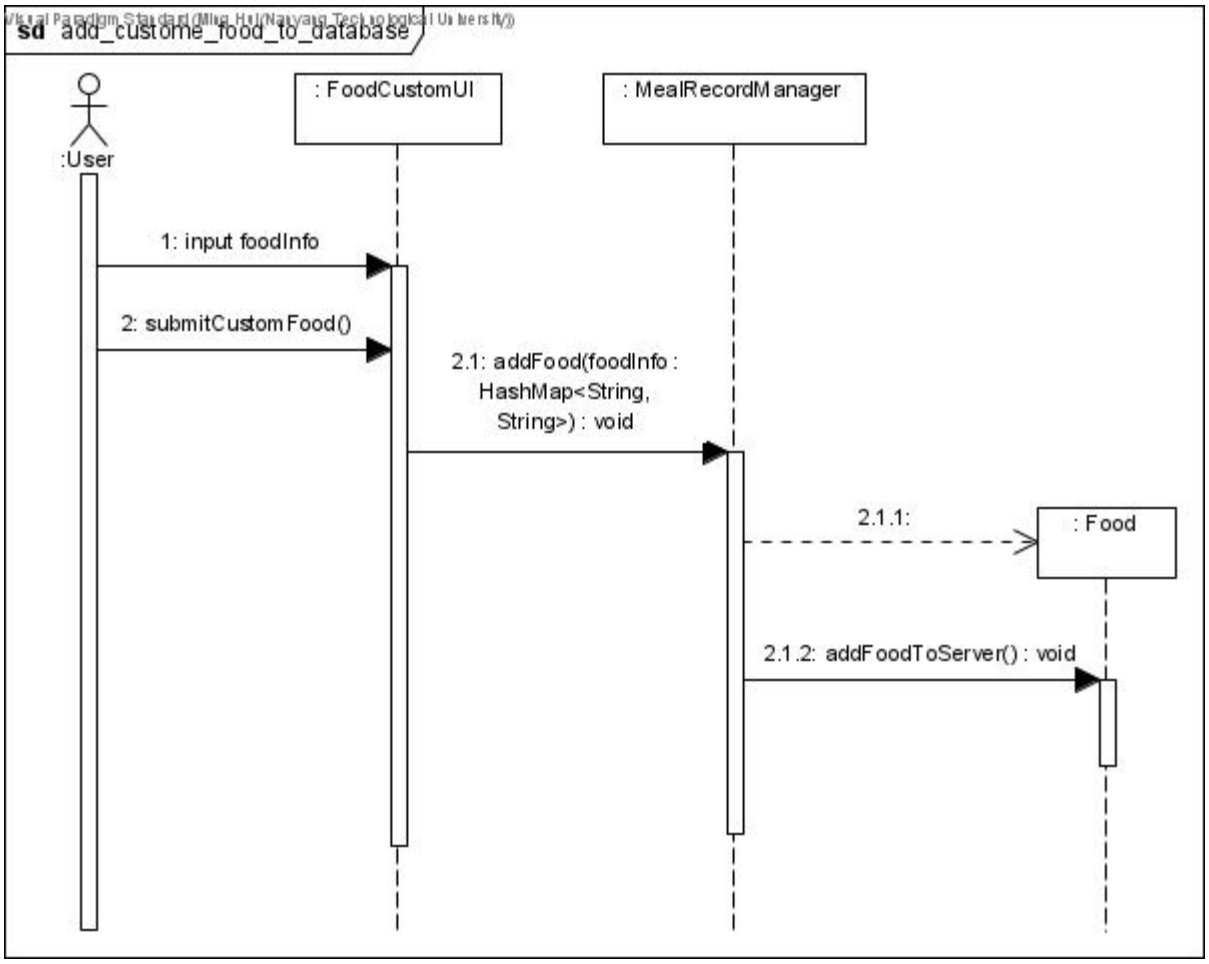

Figure 7.4.13: Sequence Diagram – Add custom food to database

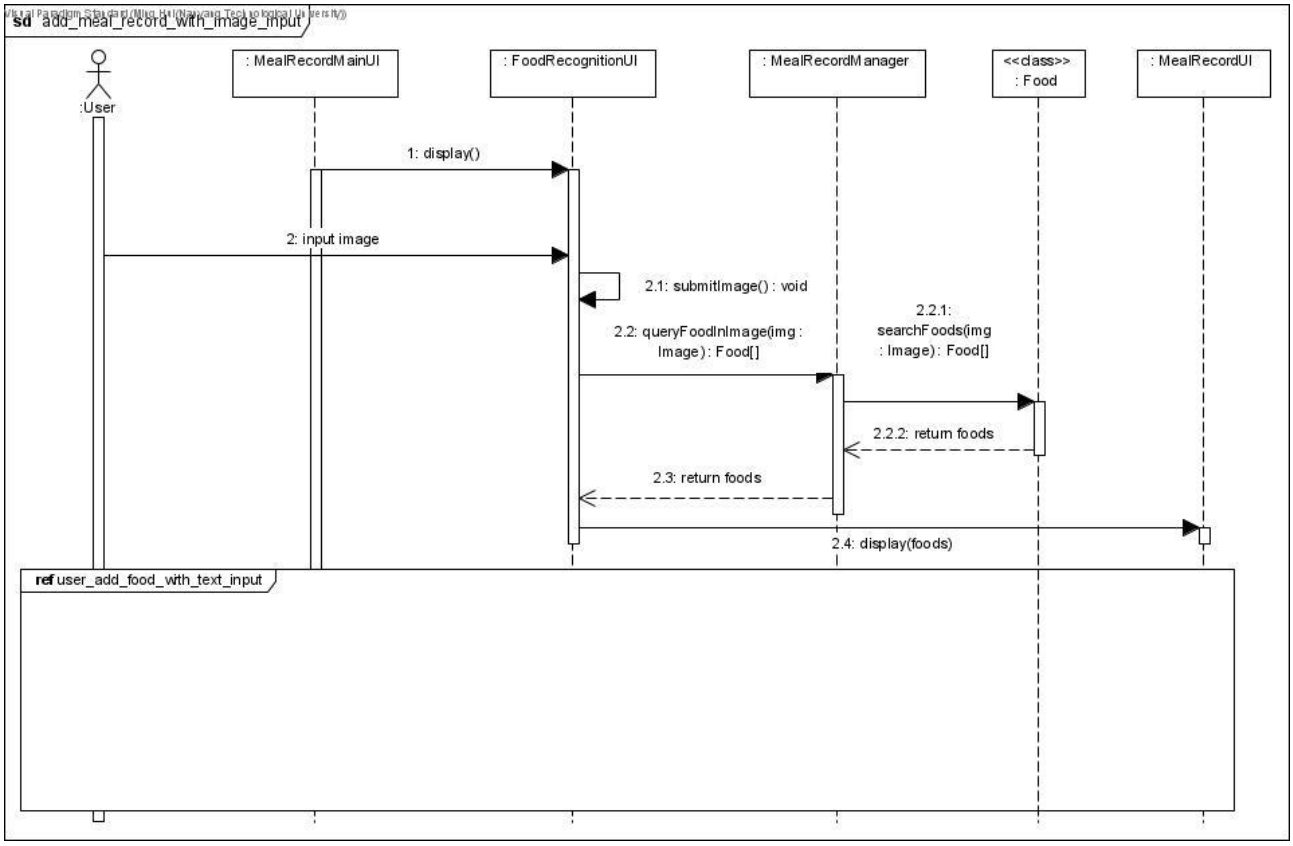

Figure 7.4.14: Sequence Diagram – Add meal record with image input

7.5. Dialog Map

*Image of dialog map has been uploaded to Tortoise SVN lab3 folder

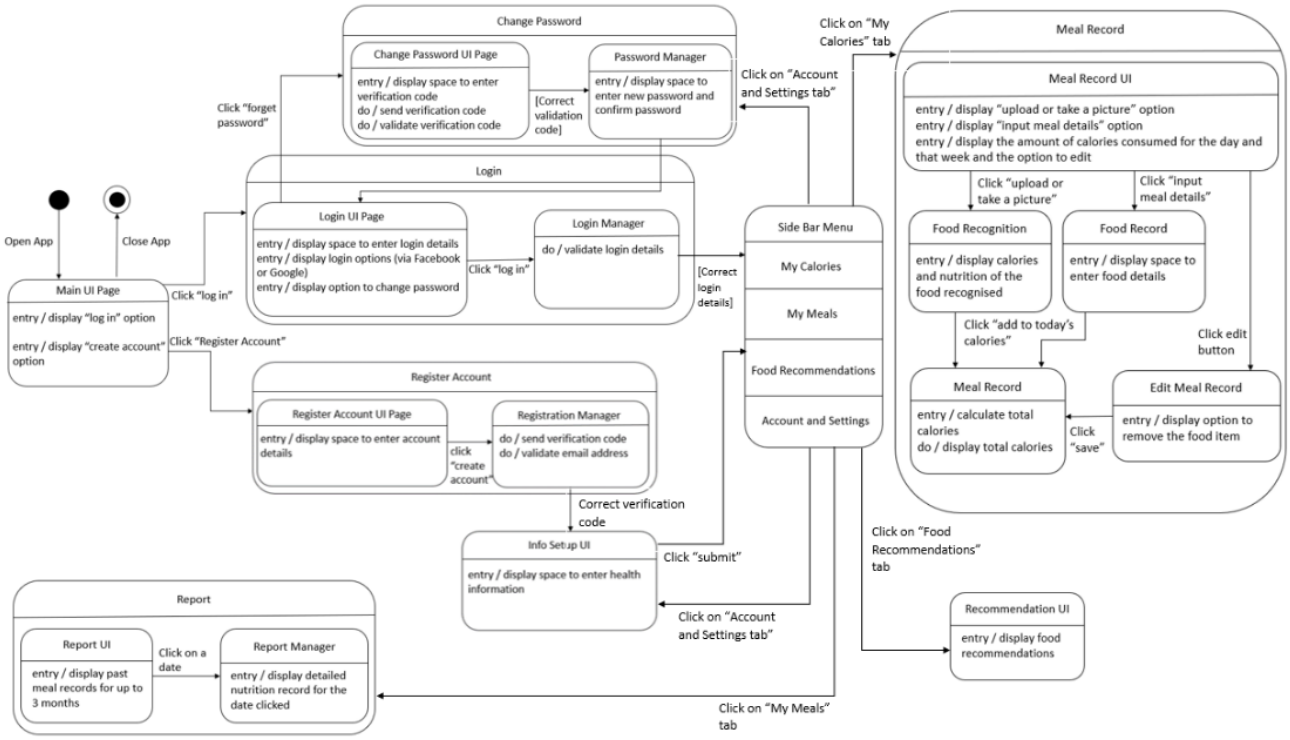

Figure 7.5.1: Dialog Map

7.6. ER Diagram

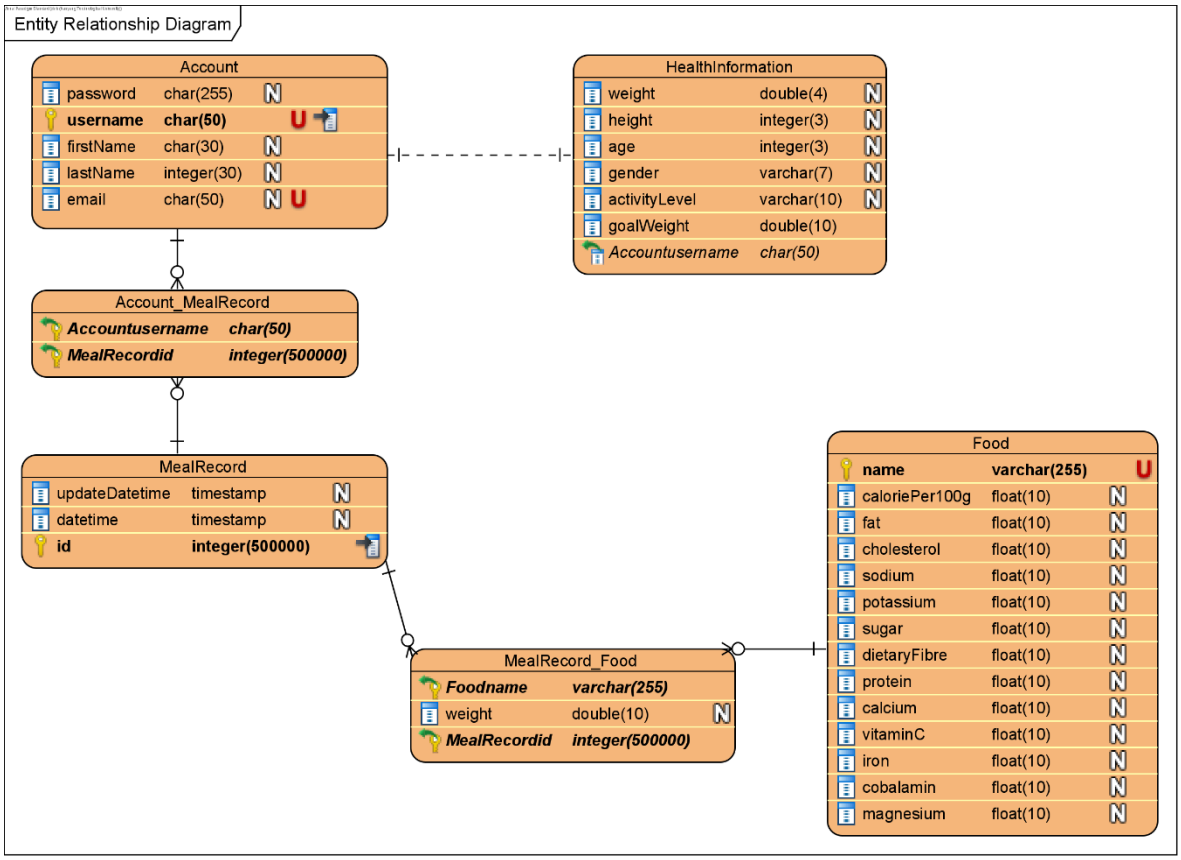

Figure 7.6.1: ER Diagram

*Images of ER diagram has been uploaded to Tortoise SVN lab3 folder

8. Appendix C: System Design

8.1. Architecture Design

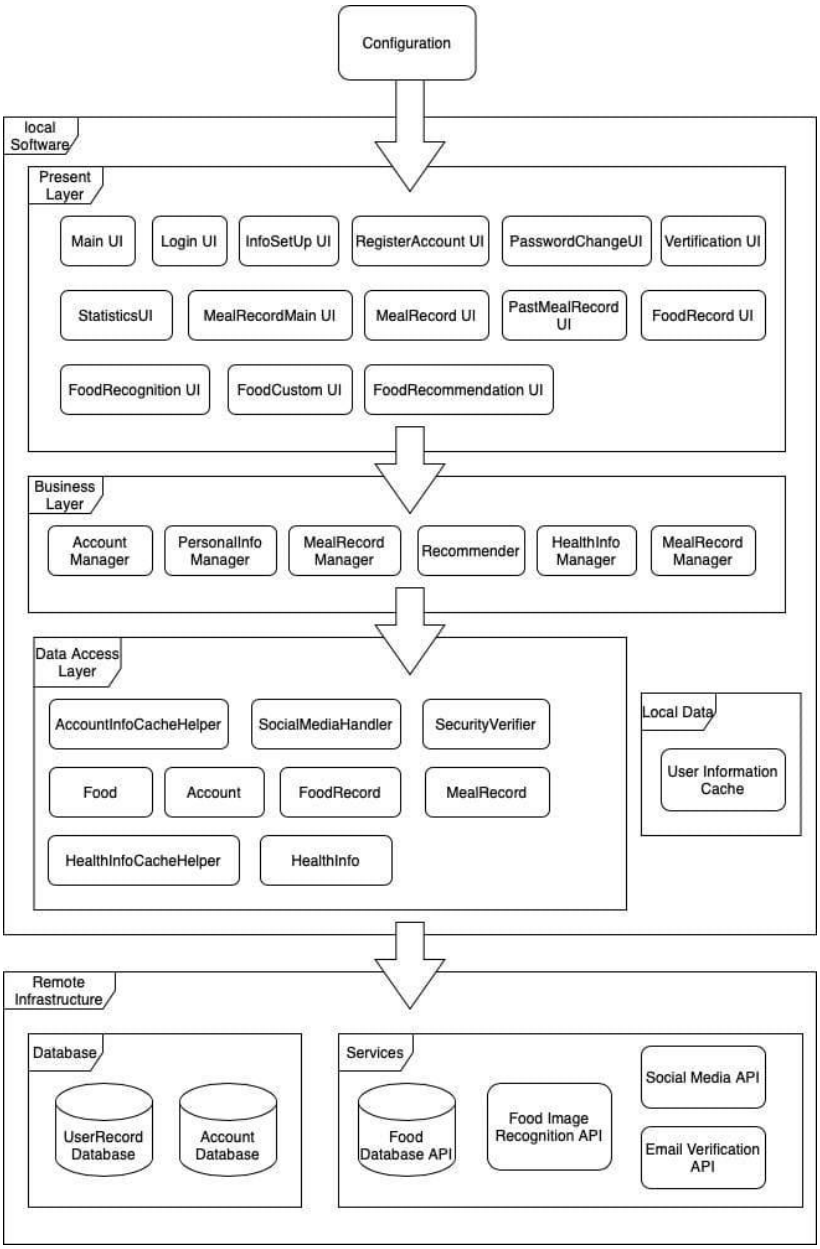

Figure 8.1.1. System Architecture diagram

*Image of system architecture diagram has been uploaded to Tortoise SVN lab3 folder

The architecture of the system is N-tier architecture.

- **Local Software,** which is on mobile phone.

  - **Present Layer** is user interfaces to interact with the user and translate tasks by calling classes on business layer.
  - **Business Layer** is to implement the app logic.
  - **Data Access Layer** is to store and manipulate the data by interacting with remote infrastructure, and it passes the data to business layer for calculation.
- **Remote Infrastructure**, which is deployed on the server or connected to API.
  - **Database**
  - **Service** is mostly 3rd party API.

8.2. Design Issue

8.2.1. Identifying and storing persistent data

The persistent data consists of the Account, Health information, Food and Meal Record data because although these data may be updated, they are still relatively permanent.
Persistent data can be stored using Disk Persistence in Firestore, which caches a copy of the Firestore data the app is actively using.

8.2.2. Providing access control

To protect private data of users, the system provides Discretionary access control (DAC). Database such as Firestore is protected, and only the owner of the database can view the complete data. Users' access is limited by the policy.

- **Authentication**
  - The user is authenticated via password and email. To be more specific, the user requires to log-in by providing correct username and password to use the application. When the user registers and changes the password, the verification code sent to the registered email is needed for authentication purpose.
- **Authorization**
  - When the local device queries data from remote database, the database will only return the records of that user. In other words, database will not return other users' records to the user.
- **Access**
  - Once authenticated and authorized, the user can access the data that is related to them from the database. For example, the meal records created by themselves, their health information, etc.
- **Manage**
  - The management of access control system is implemented on remote infrastructure side.

8.3. Design Patterns

8.3.1. Creational Patterns

8.3.1.1. Factory Method

This pattern is used when one has a set of classes but is not sure which one you will need to instantiate until runtime. It defines an interface to create different objects, without knowing what

sort of objects it needs to create or how it is created at first, and lets subclasses decide which class to instantiate.
This is implemented in UI as users will be able to click on different fragments in a navigation bar such that when one is selected, the appropriate UI is displayed. This helps to decouple class selection and object creation from the client, leading to greater flexibility in object creation.

8.3.1.2. Singleton

This pattern restricts the instantiation of a class to a single instance, and exactly one object is required to organize actions across the system.
For example, MealRecordManager is a Singleton and only has one global instance for all related classes and operations. This ensures the encapsulation of data from users.

8.3.2. Structural Patterns

8.3.2.1. Data Access Object (DAO) Pattern

This pattern provides an interface to database or persistent data, separating the persistence layer from the application layer. DAOs are in data access layer. DAO performs CRUD (Create, Retrieve, Update, Delete) operations and allows for data access from the persistence storage and manipulation of data in the persistence storage. This helps decouple the persistent storage implementation from the rest of the application.
Account sub-system is an example,

- **Create** During registration, an Account instance will be created. It will post the information including username, password and real name to User database.
- **Retrieve** When User logs in, an Account instance will be created. Its correctness will be checked by the server in a secure manner. The user's name can also be retrieved from the database as well.
- **Update** After User changes the password, Account object will submit the post to the database.
- **Delete** Deletion is not applicable for account, but User can delete meal records. MealRecord (DAO) will submit the deletion activity to the database.

9. Appendix D: UI Mockups

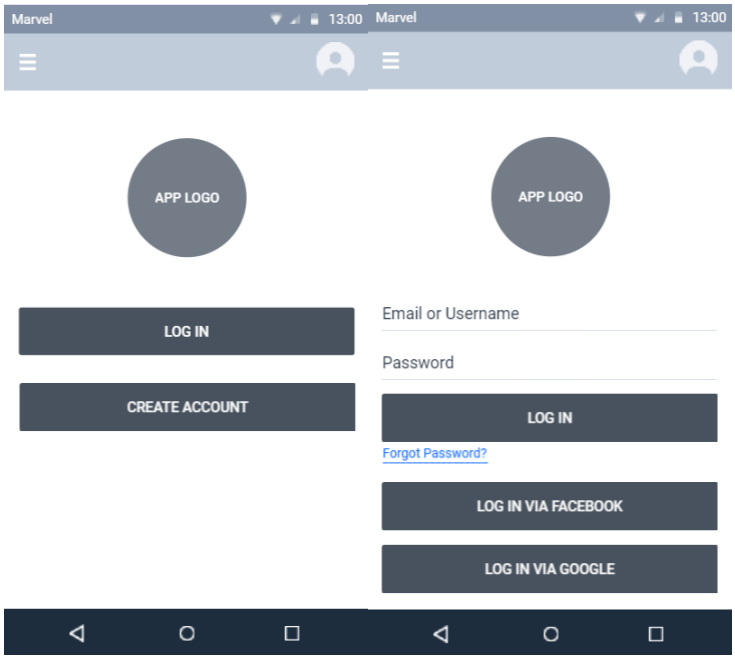

Figure 1: Login

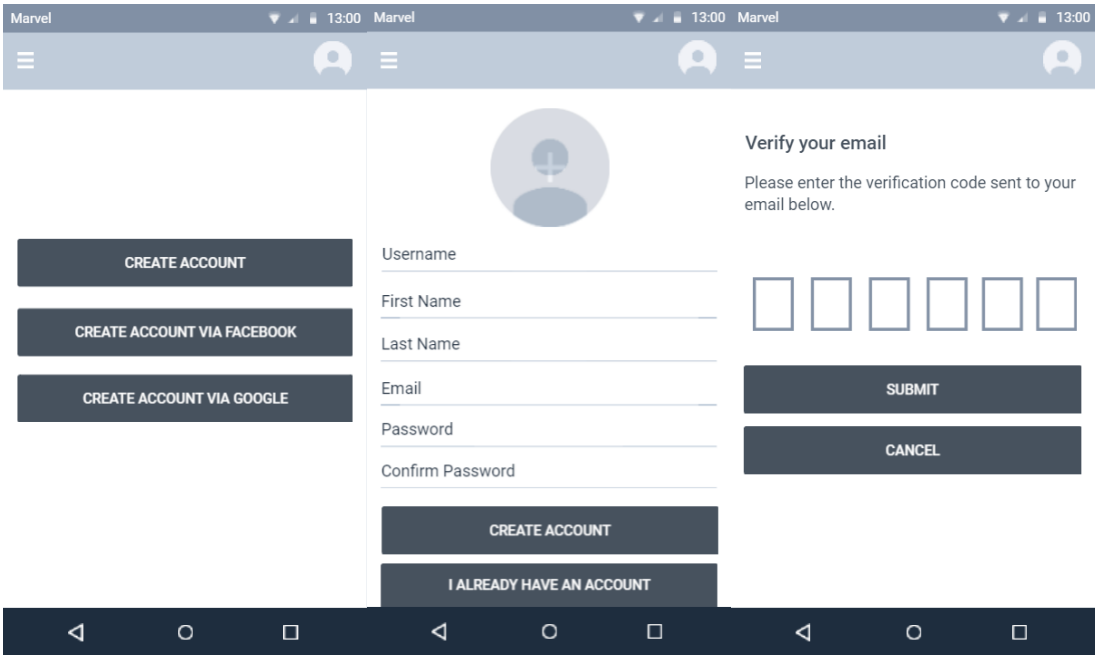

Figure 2: Account creation and verification of email

Figure 1 and 2 show the user interfaces for login and account creation. The interface is minimalistic and designed to be similar to those used in real life applications for users to easily traverse through the account login and creation process.

For account creation, the system sends a verification code to the email address entered by the user in the appropriate field. Once the user enters the correct code and the user's email address is verified, the account is created.

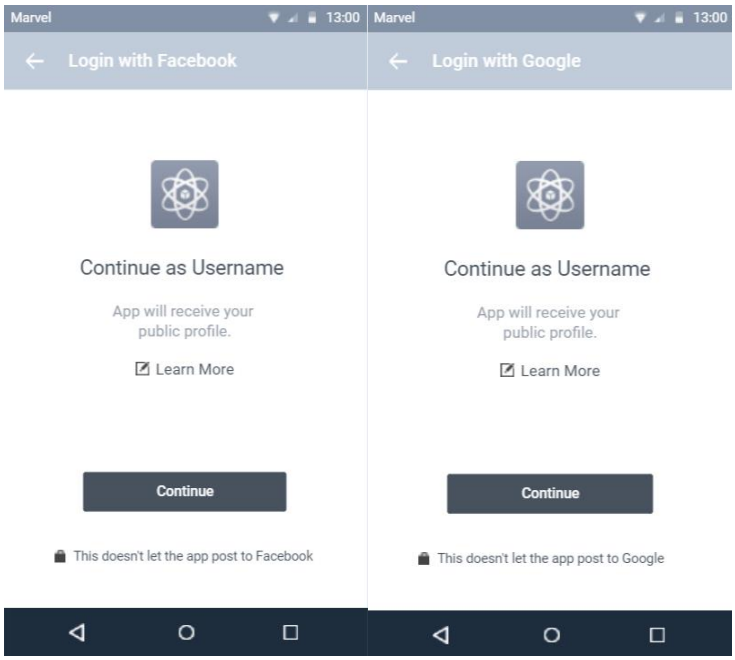

Figure 3: Request to grant permission to user's Facebook or Google account

Figure 3 shows the user interface of the application request user to grant it permission to his Facebook or Google account. After this, the user will be able to create and log in to his account via Facebook or Google.

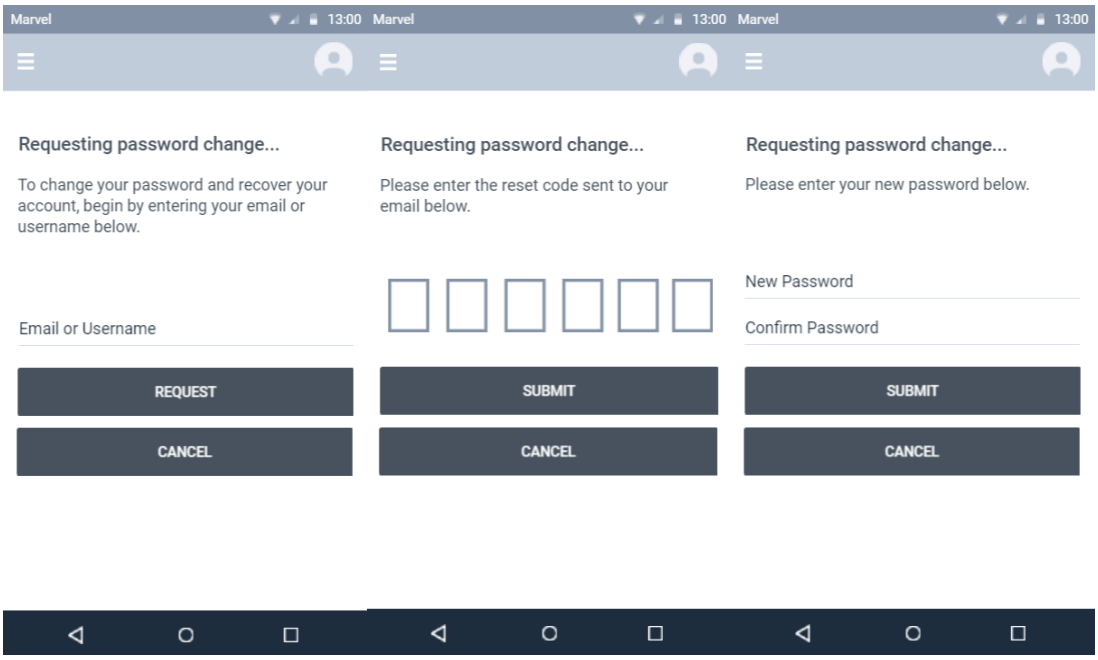

Figure 4: Request Password Change

Figure 4 shows the user interface for requesting password change. After the user submits his email address or username, the system sends a verification code to the email address linked to the user's account in the appropriate field. Once the user enters the correct code and the user's email address is verified, the user is able to enter the new password. Once verified to be new passwords and that both fields are matching, the user's password is updated in the database

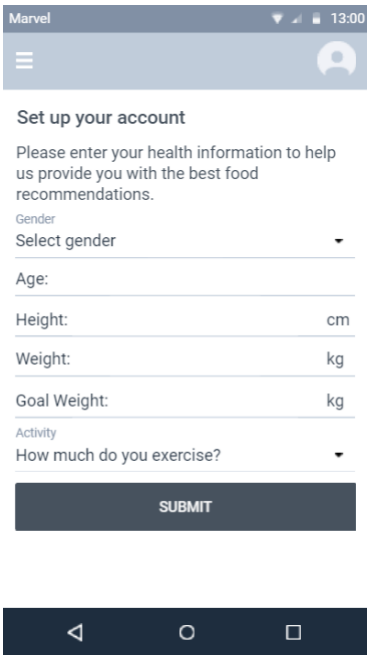

Figure 5: Input health information

Figure 5 shows the user interface for set up the account with health information. This is required for the app to provide food recommendations based on the user's goal weight and calculate calories quota remaining for the day.

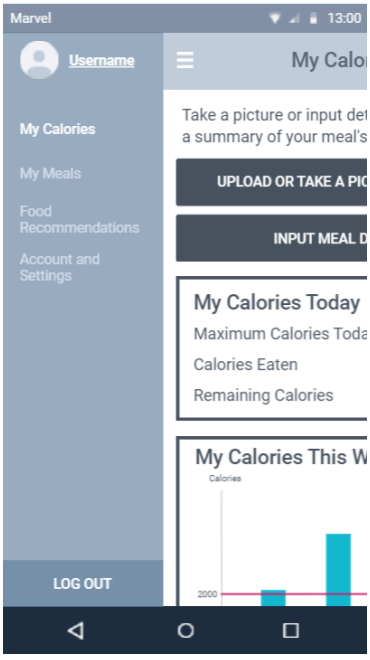

Figure 6: Side bar menu

Figure 6 shows the user interface of the side bar menu, which contains four pages that a user can traverse to: My Calories, My Meals, Food Recommendations and Account and Settings. Additionally, the user will be able to log out of their account from here.

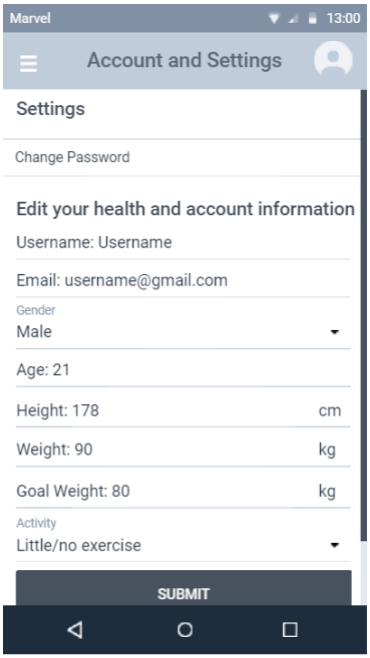

Figure 7: Edit health information or change password

Figure 7 shows the user interface for editing the account information and an option to change password. This is to ensure greater security in the app by allowing user to change passwords regularly. The user will also have the flexibility to edit the height, weight, goal weight and activity since these may change overtime. The user also has the freedom to edit the username and email address.

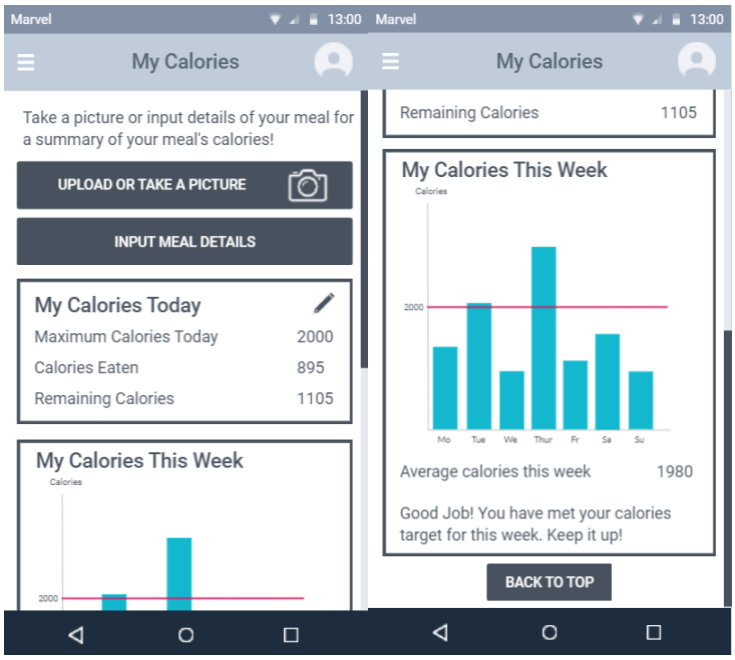

Figure 8: My Calories Page

Figure 8 shows the user interface for My Calories page. It allows users to input details of the meal via uploading or taking a picture or manual input. The user can edit the calories consumed today. Besides that, it also displays the health statistics summary report.

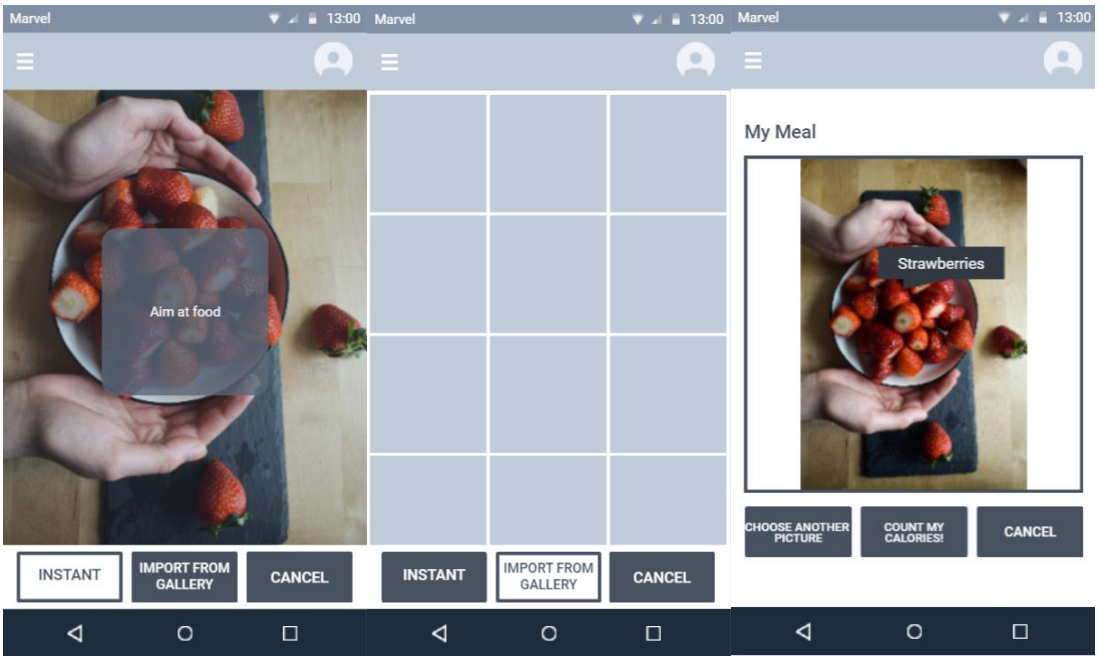

Figure 9: Upload food image and Food recognition

Figure 9 shows the user interface for taking picture of their meal or upload an image from their gallery in order to add meal records and query food information. In the last page, the food is detected.

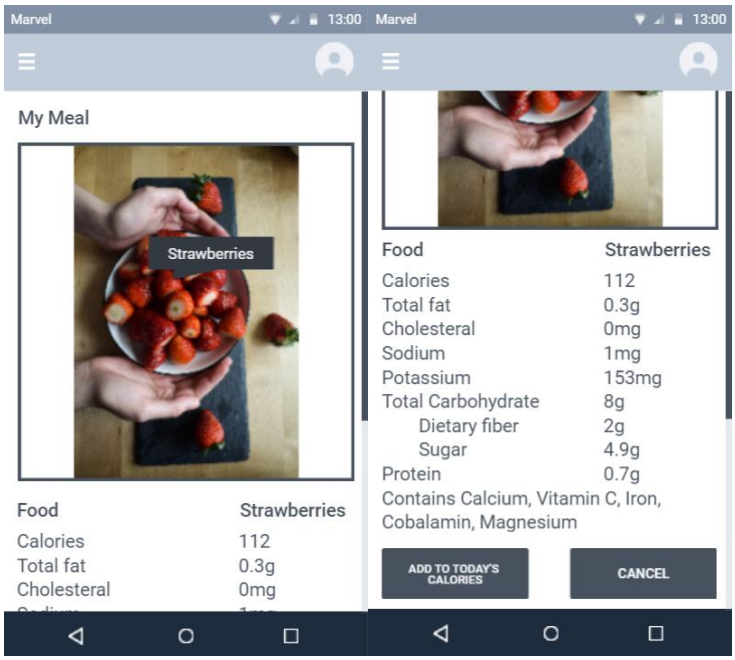

Figure 10: Meal nutrition information provided after image recognition

Figure 10 shows the user interface for the meal nutrition information provided after image recognition. The user can add the meal calories to the day's calories.

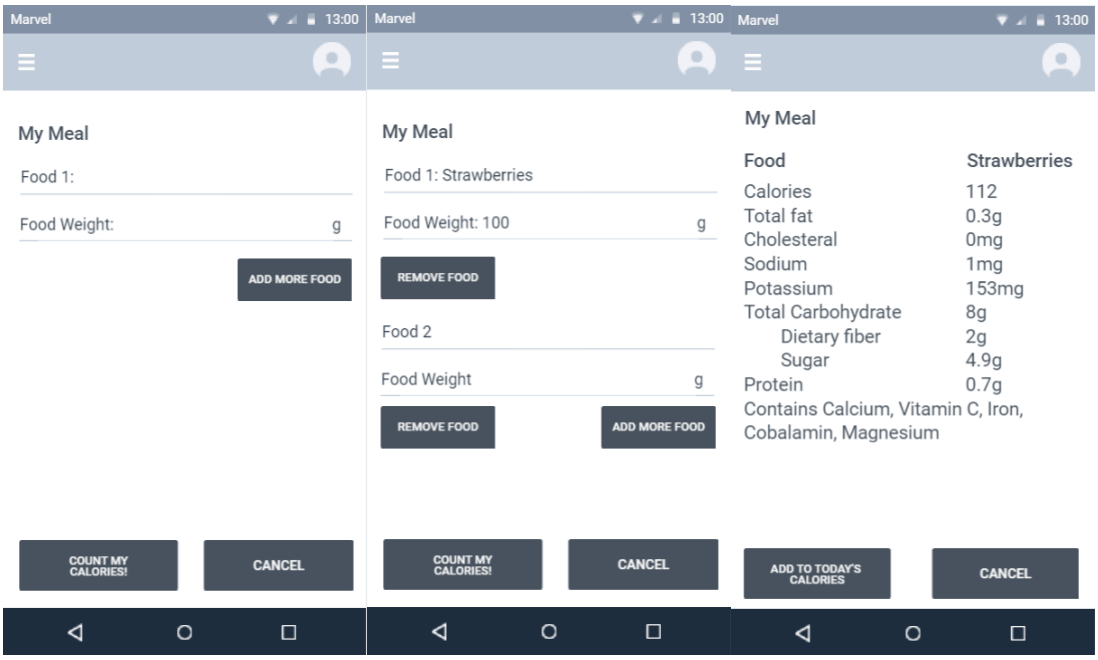

Figure 11: Manual input of meal details and meal nutrition information provided

Figure 11 shows the user interface for the manual input of meal details. The user can add more foods or remove foods by pressing the appropriate buttons as desired. In addition, similar to the picture taking or upload method as seen in Figure 10, it produces the meal nutrition information after the user clicks “Count my calories!”

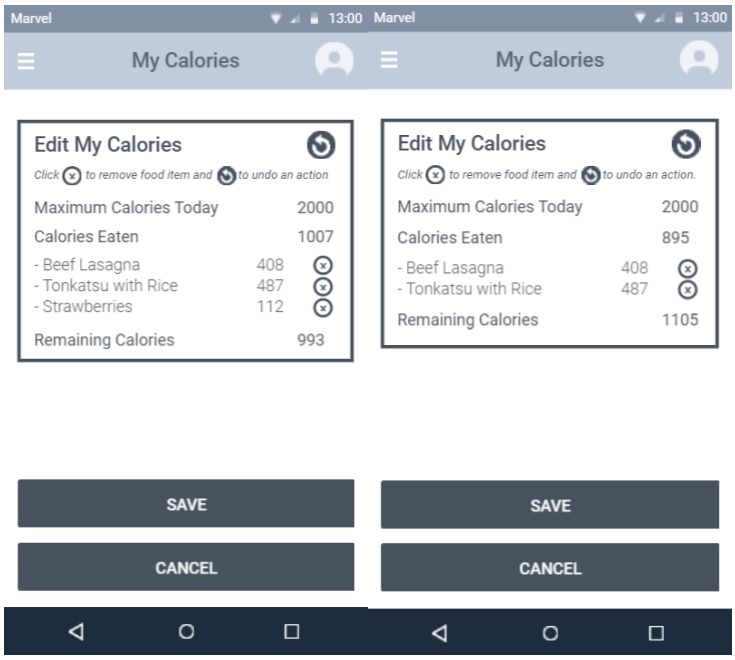

Figure 12: Edit meal calories

Figure 12 shows the user interface of editing meal calories. The user can remove a food that he accidentally added by clicking on the X button, or the undo button to undo that action. The calories consumed and calories quota remaining will change correspondingly.

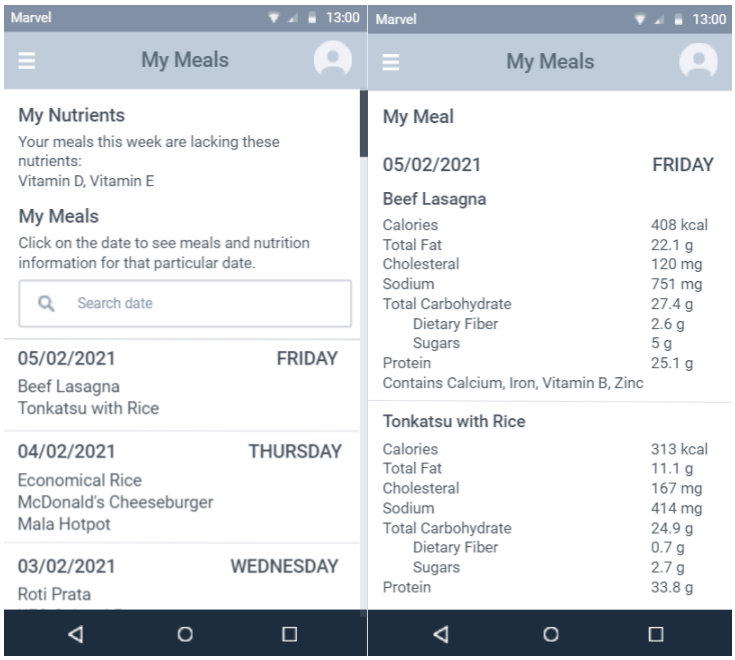

Figure 13: Review past meal details

Figure 13 shows the user interface of reviewing past meal details. The user can either scroll through the dates or search for a particular date to view his meal details on that day. After clicking on the date, he is able to view the full nutrition information of all of his meals on that day. The user can view previous meals from up to 3 months ago.

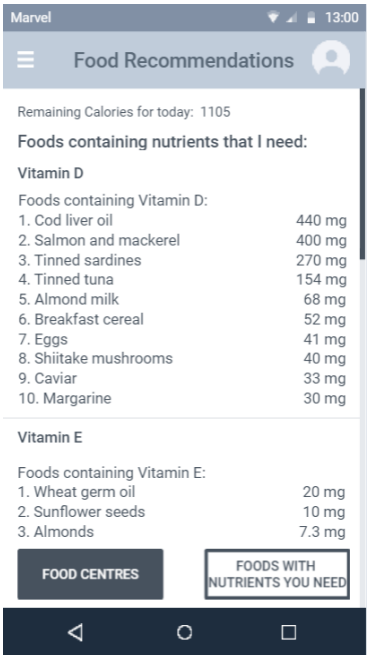

Figure 14: Food type recommendations based on the nutrients the user lacks

Figure 14 shows the user interface of food type recommendations based on the nutrients the user lacks. For each nutrient lacked, the top 10 foods containing such nutrients will be listed for the user to incorporate in their meal as much as possible.

10. Appendix E: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>

Source: http://www.frontiernet.net/~kwiegers/process_assets/srs_template.doc