

---

# Software Requirements Specification

for

## EatWellthy

Version 1.0 approved

Prepared by

LIU XIAOTAO,  
LOW JO YI, NICOLE,  
MAHI PANDEY  
MEHTA RISHIKA,  
ZHANG YICHI  
ZHAO QIXIAN

TEAM 31, SDDA, NANYANG TECHNOLOGICAL UNIVERSITY

<2024-11-07>

<b>Revision History .....</b>	<b>iv</b>
<b>1. Introduction.....</b>	<b>1</b>
Purpose.....	1
Document Conventions.....	1
Intended Audience and Reading Suggestions.....	1
Product Scope .....	2
References.....	2
<b>2. Overall Description .....</b>	<b>3</b>
Product Perspective.....	3
Product Functions .....	4
User Classes and Characteristics .....	5
Operating Environment.....	7
Design and Implementation Constraints.....	8
User Documentation .....	8
Assumptions and Dependencies .....	8
<b>3. External Interface Requirements .....</b>	<b>9</b>
User Interfaces .....	9
Hardware Interfaces .....	24
Software Interfaces .....	25
Communications Interfaces .....	26
<b>4. System Features .....</b>	<b>26</b>
Account Registration .....	27
Account Login .....	30
Google OAuth.....	32
Reset Forgotten Password.....	33
Update Password.....	35
Delete User Account.....	36
Update Basic Profile Information .....	38
Manage Dietary Preferences.....	39
Using the Nutritional Information Finder.....	41
Generate AI-Powered Dietary Suggestions .....	43
Log Your Daily Meals .....	45
Edit Logged Meal .....	47
Delete Logged Meal.....	50
View and Search Meal History .....	51
Add Customised Food Option .....	53
Generate User Report.....	55
Navigate to Grocery Store Pages .....	56
Add Event in Calendar.....	58
Update Event in Calendar .....	59
Add Event to Google Calendar.....	61
Sync Diet Plan with Calendar.....	63
Display FAQ Page .....	66
Search for Near-by Grocery Stores.....	67
Ask Nutritional Questions .....	69
4.25 Query Personalized Nutritional Information via Chatbot.....	71
<b>5. Other Nonfunctional Requirements.....</b>	<b>74</b>
Performance Requirements.....	74
Safety Requirements .....	74
Security Requirements.....	74
Software Quality Attributes .....	75
Business Rules .....	76
<b>6. Other Requirements .....</b>	<b>77</b>
<b>7. API Testing.....</b>	<b>79</b>
<b>8. Appendix A: Data Dictionary .....</b>	<b>82</b>
<b>9. Appendix B: Analysis Models .....</b>	<b>83</b>

<b>Appendix C: To Be Determined List.....</b>	<b>88</b>
---	-----------

## Revision History

Name	Date	Reason For Changes	Version
Mahi	25 October 2024	Initial Template Upload	1.0
Mahi	25 October 2024	The initial write-ups for the Introduction and Overall Description were completed, providing a comprehensive overview of the EatWellthy's goals and context.	1.1
Rishika	26 October 2024	An initial write-up for Other Nonfunctional Requirements was prepared, covering aspects such as system performance, security, and scalability.	1.2
Nicole	26 October 2024	The first draft of the System Features section was written, detailing the core functionalities and expected behavior of the system.	1.3
Mahi	26 October 2024	The External Interface Requirements were initially documented, specifying the interactions between the system and external entities like hardware, software, and other systems.	1.4
Rishika	28 October 2024	API Testing summary was created to outline the process for testing individual API endpoints to ensure correct functionality.	1.5
Mahi	5 November 2024	The White Box and Black Box Testing Test Cases were initially written, focusing on testing system functions based on the requirements without looking into internal workings.	1.6
Nicole	6 November 2024	The User Interface write-up was completed, providing detailed descriptions of the system's interface design and user experience.	1.7
Nicole, Mahi, Rishika	10 November 2024	Finalization of the entire SRS and other requirements.	1.8

# 1. Introduction

## Purpose

The purpose of the Software Requirement Specification (SRS) is to serve as a documentation of a website application, EatWellthy. This document will serve as a guide for the development, design, testing and deployment of EatWellthy. It will ensure alignment with stakeholder expectations and requirements. EatWellthy aims to assist users in making nutritious food choices by offering convenient meal tracker and personalised meal recommendations.

## Document Conventions

This SRS document follows standard documentation conventions to ensure clarity and consistency. Text in bold will be used for emphasis while text in italics will be used for newly introduced or key terms.

**Software Requirement Specification Standard:** IEEE 830-1998. Priorities of higher-level requirements are inherited by detailed level requirements unless explicitly stated otherwise.

**Font:** Times New Roman

**Heading1:** Size 18, Bold

**Heading2:** Size 14, Bold

**Heading3:** Size 12, Bold

**Heading4:** Size 11, Bold

**Content:** Size 12

**Spacing in content:** 1 line spaced

## Intended Audience and Reading Suggestions

This SRS document is designed to cater to the needs of various stakeholders, such as developers, project managers, marketing staff, testers, documentation writers and users of EatWellthy.

While we encourage all readers to review the document in whole to have a complete understanding of EatWellthy, we have identified the key sections different stakeholders should focus on:

- **Developers:** Focus on the overall system architecture and detailed requirement specifications to gain a comprehensive understanding of the technical implementation.
- **Project Managers:** Review the entire document, paying particular attention to the project objectives, product scope and high-level requirements to align project activities with desired outcomes.
- **Marketing Staff:** Focus on the product scope and user interface sections to ensure their marketing strategies align with the product's capabilities.
- **Testers:** Focus on functional, non-functional requirements and exception handling to develop thorough test plans.

- **Documentation writers:** Reference the entire document to create user manuals and help guides, emphasizing on user documentation to ensure comprehensive guides.
- **Users:** Encouraged to review the overview and user interaction sections to become familiar with the system's features and user interface.

## Product Scope

EatWellthy serves as an innovative web application dedicated to streamlining the process of diet and nutrition tracking. By integrating advanced nutritional resources and artificial intelligence, EatWellthy aims to empower users with tools to make informed dietary choices and efficiently meet their health goals. EatWellthy's primary goals include simplifying nutrition management, personalizing dietary recommendations, and integrating meal plans seamlessly into users' daily routines. By focusing on 4 core functionalities—Nutrition Tracking, Grocery Store Locator, Diet Suggestions, and an Interactive Nutrition Helper Chatbot—EatWellthy positions itself as a comprehensive solution for diet management.

For Nutrition Tracking, EatWellthy leverages the Nutritionix API to provide users access to an extensive food database, enabling precise tracking of nutritional content. Users can log meals, specifying food items and quantities, or create custom food entries if needed. This feature also includes an integrated calendar, allowing users to plan their meals, import them into Google Calendar, and monitor their dietary progress over time. By providing a comprehensive view of daily intake, EatWellthy helps users make informed dietary decisions that align with their health goals.

Powered by artificial intelligence, EatWellthy's diet suggestion feature creates adaptive meal plans tailored to individual goals and preferences. The AI considers factors such as gender, age, height, weight, dietary restrictions, allergies, and activity levels, generating recommendations that support users' unique wellness objectives. Whether users aim to maintain weight, improve fitness, or address specific health concerns, this functionality offers a highly personalized approach to meal planning.

Grocery Store Locator helps users find nearby supermarkets and access online websites of local grocery stores, making healthy food purchase easier and more efficient. By providing location-based search capabilities, users can make informed choices that align with both their health goals, reinforcing EatWellthy's commitment to convenient and accessible nutrition management.

Welloh, EatWellthy's interactive chatbot, serves as a real-time assistant for all nutrition-related inquiries. Users can ask Welloh about specific foods, seek diet suggestions, and receive nutritional guidance, benefiting from instant, personalized support. This feature enhances user engagement by offering an accessible, conversational approach to diet management, making the journey towards healthier eating habits more interactive and enjoyable.

## References

1. Nutritionix API Documentation  
V2.0 API Home. (n.d.). <https://developer.nutritionix.com/docs/v2>

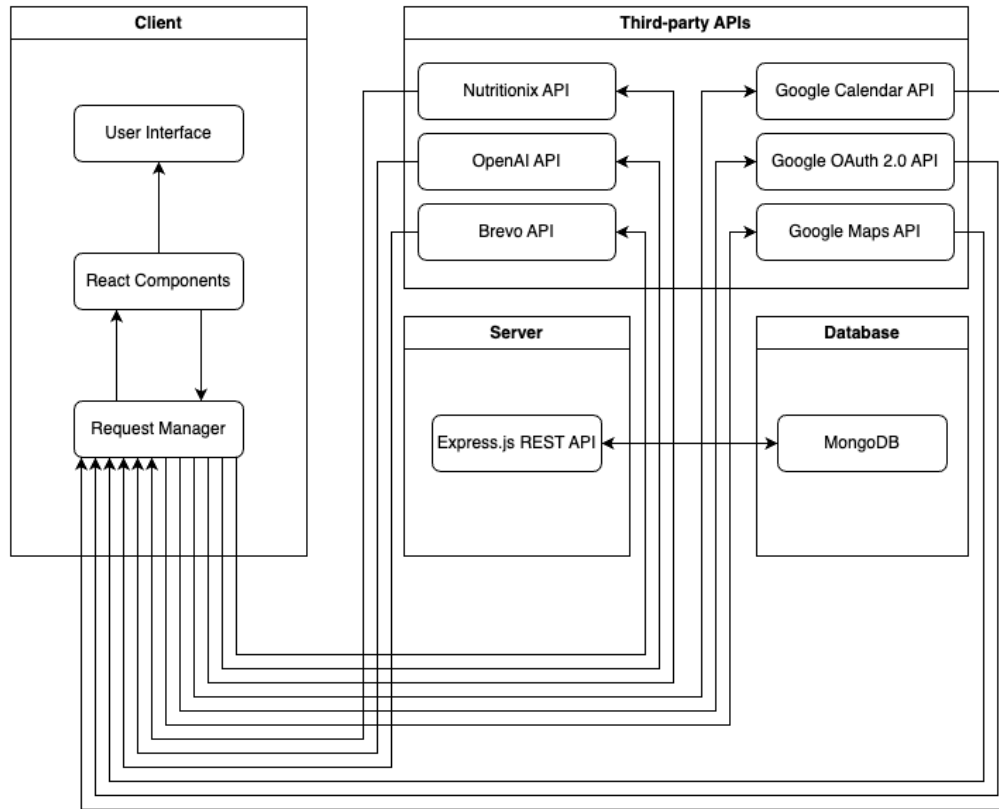
2. BREVO API Documentation  
*Brevo API*. (n.d.). Brevo API.  
<https://developers.brevo.com/>
3. Google API Library  
*API Library – APIs & Services – Google Cloud console*. (n.d.).  
<https://console.cloud.google.com/apis/library>
4. OpenAI API Documentation  
OpenAI Platform. (n.d.).  
<https://platform.openai.com/docs/api-reference/introduction>
5. IEEE Good Software Engineering Standards  
Ieeinnovate. (2019, December 9). *IEEE Software Engineering Standards / Innovate*.  
Innovate.  
<https://innovate.ieee.org/ieee-software-engineering-standards/>
6. *Good practices for educational software engineering projects*. (n.d.). IEEE Conference  
Publication | IEEE Xplore.
7. <https://ieeexplore.ieee.org/document/4222631>
8. Nanyang Technological University, Singapore. (n.d.). SC2006 Software Engineering.  
Lecture Notes.

## 2. Overall Description

### Product Perspective

The EatWellthy Web Application is an innovative, self-contained product created to serve as a comprehensive dietary and nutritional management tool. It is not part of an existing product family nor does it replace any existing systems. Instead, EatWellthy is crafted to integrate seamlessly with a range of features including meal planning, nutritional tracking, and diet suggestions to provide a unified nutrition management experience.

A simplified system architecture diagram is shown below for a brief overview of the operation of EatWellthy.



## Product Functions

EatWellthy features consists of 11 categories – Account Management, Profile, Dashboard, Tracker, Analysis, Grocery, Calendar, FAQs, Location, Welloh and Database Management. This section provides a specific summary of the system features contained in the mobile application.

### 2.2.1 Account Management

1. New User Registration
2. User Login via Email
3. User Login via Google Account
4. Reset Forgotten Password
5. Update Password
6. Delete User Account

### 2.2.2 Profile

1. Update Profile Information
2. Manage Dietary Preferences

### 2.2.3 Dashboard

1. Using the Nutritional Information Finder
2. Generate AI-Powered Dietary Suggestions



**2.2.4 Dashboard**

1. Log Your Daily Meals
2. Edit a Logged Meal
3. Delete a Logged Meal
4. View and Search Meal History
5. Add Customized Food Option

**2.2.5 Analysis**

1. Generate User Report

**2.2.6 Grocery**

1. Navigate to Grocery Store Pages

**2.2.7 Calendar**

1. Add Event in Calendar
2. Update Event in Calendar
3. Add Event to Google Calendar
4. Sync Diet Plan with Calendar

**2.2.8 FAQs**

1. Display FAQ Page

**2.2.9 Location**

1. Search for Near-by Grocery Stores

**2.2.10 WellOh**

1. Ask Nutritional Questions
2. Query Personalized Nutritional Information via Chatbot

**2.2.11 Database Management**

1. Update Food Database
2. Manage User Data
3. Backup and Restore

**User Classes and Characteristics**

The anticipated user classes include:

**2.3.1 Casual Users**

Aspect	Description
Frequency of use	Occasional
Subset of Product Functions Used	All

Technical Expertise	<ul style="list-style-type: none"> <li>- Owns a computer or laptop</li> <li>- Familiar with using browser-based application</li> </ul>
Age	All
Characteristics	Individuals who are looking to gain general dietary insights or for temporary diet tracking and are less motivated by stringent health goals

### 2.3.2. Health Enthusiasts

Aspect	Description
Frequency of use	Daily
Subset of Product Functions Used	Nutrition tracking, diet suggestions
Technical Expertise	<ul style="list-style-type: none"> <li>- Owns a computer or laptop</li> <li>- Familiar with using browser-based application</li> </ul>
Age	All
Characteristics	Individuals who are highly motivated and disciplined about their health goals

### 2.3.3 Busy Professionals

Aspect	Description
Frequency of use	Weekly
Subset of Product Functions Used	Calendar, diet suggestions
Technical Expertise	<ul style="list-style-type: none"> <li>- Owns a computer or laptop</li> <li>- Familiar with using browser-based application</li> </ul>
Age	All
Characteristics	Individuals who seek efficiency and effectiveness in tools they use and prefer automated features that reduce the need for manual input

### 2.3.4 Weight Management Users

Aspect	Description
Frequency of use	Daily
Subset of Product Functions Used	Nutrition tracking, diet suggestions
Technical Expertise	<ul style="list-style-type: none"> <li>- Owns a computer or laptop</li> <li>- Familiar with using browser-based application</li> </ul>
Age	All
Characteristics	Individuals who are highly goal-oriented, often with specific targets in terms of weight and seeks structure and guidance in their eating habits to achieve desired weight management outcomes

## Operating Environment

The development environment of EatWellthy web application will be covered under this section.

### 2.4.1 User Device Permissions

EatWellthy requires an active internet connection to function, as all application features depend on HTTP requests between the front-end and back-end servers.

### 2.4.2 Google Permissions

EatWellthy will only request for Google profile data when users choose to authenticate via Google OAuth.

### 2.4.3 Development Environment

Development Environment	Description
Front-end Development:  React.js	React.js is a popular JavaScript library developed by Facebook (now known as Meta) for building dynamic and responsive user interfaces, especially single-page applications (SPAs). It utilizes a component-based architecture, allowing developers to create reusable UI components that manage their own state.  React's Virtual DOM efficiently updates and renders components, enhancing performance and user experience. With its declarative syntax, React simplifies the process of designing interactive UIs by letting developers describe how the interface should look based on the current state. Additionally, React integrates seamlessly with other libraries and frameworks, making it a versatile choice for modern web development.
Back-end Development:  Express.js	Express.js is a minimalist and flexible web application framework for Node.js, designed to streamline the process of building robust server-side applications and APIs. It offers a simple yet powerful set of features, including middleware support for handling HTTP requests, routing for defining application endpoints, and template engine integration for dynamic content rendering.  Express.js's unopinionated nature allows developers to structure their applications as they see fit, providing the flexibility to incorporate various plugins and extensions. Its lightweight footprint and high performance make Express.js a favored choice for developing scalable web services and real-time applications.
Database:	MongoDB is a leading NoSQL, document-oriented database known for its scalability, flexibility, and high performance. It stores data in JSON-like BSON documents, allowing for dynamic and hierarchical data structures without the constraints of fixed schemas. This flexibility enables rapid development and easy adaptation to changing application requirements.

MongoDB	MongoDB supports horizontal scaling through sharding, ensuring it can handle large volumes of data and high traffic loads efficiently. Additionally, it offers robust indexing and querying capabilities, replication for high availability, and seamless integration with various programming languages and frameworks, making it ideal for modern, data-driven applications.
---------	--

## Design and Implementation Constraints

The development of EatWellthy will be subjected to several constraints that will guide the design and implementation of the web application:

- Use of a specific tech stack, React.js for front-end and Node.js for back-end
- Adherence to data protection regulations like GDPR while handling user data

## User Documentation

EatWellthy will include a set of comprehensive set of documentation to ensure clarity and ease of use for developers. The user documentation components will include:

### 2.6.1 README File

A README file for the main repository gives an overview of EatWellthy as well as a step-by-step approach, starting from cloning the repository to running the application locally, ensuring anyone can start contributing without hassle. We have also included a section on database connection setup and reinstalling modules to cover potential troubleshooting needs.

### 2.6.2 Code comments

Clear code comments enhance collaboration by making the purpose of each section easily understandable. They are used to describe each function's purpose, access level, and any notable logic to help future developers understand the flow quickly and make modifications with confidence.

## Assumptions and Dependencies

### 2.7.1 Assumptions:

- Continuous availability of third-party services for calendar and nutrition data.
- Steady internet connection for end-users to access the web application.

### 2.7.2 Dependencies:

#### 2.1.1.1 Front-end Libraries

Library	Purpose
React.js	To create dynamic single page application
react-big-calendar	To provide a calendar component
react-datepicker	To provide a customizable date picker component
react-markdown	To convert Markdown text into HTML
react-bootstrap	To provide Bootstrap-styled React components
react-redux	To connect React components with the Redux store
react-thunk	To enable asynchronous actions and side effects in Redux applications

#### 2.1.1.2 Back-end Libraries

Library	Purpose
Express.js	To create scalable web applications in Node.js
passport-google-oauth20	To enable user authentication via Google OAuth 2.0
bcrypt	To provide hashing functions
cookie-session	To manage and store user session data using cookies

#### 2.1.1.3 External Services

EatWellthy depends on third-party APIs like:

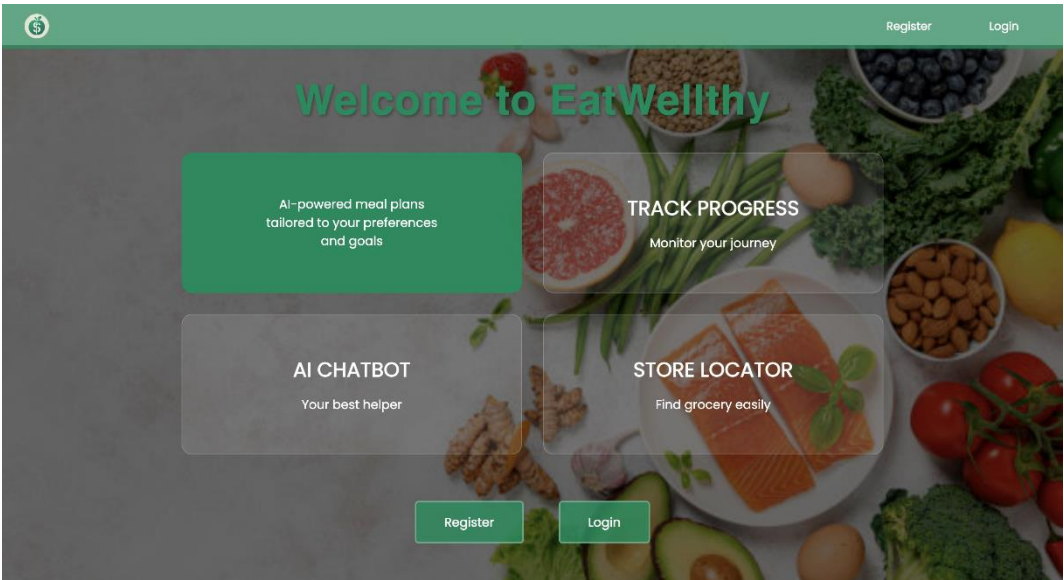
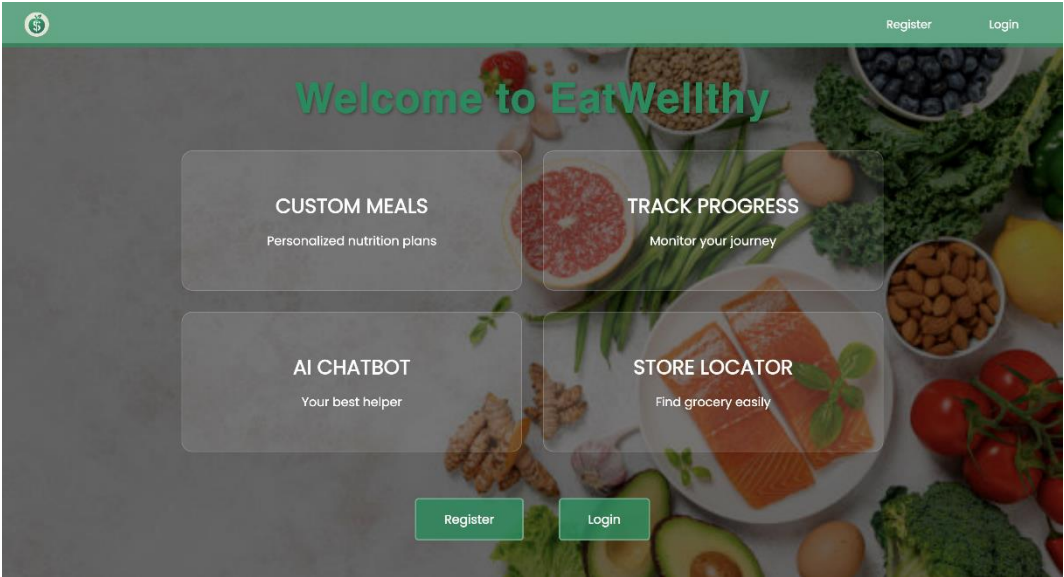
- Google OAuth
- Google Maps
- Google Calendar
- Nutritionix
- OpenAI
- Brevo

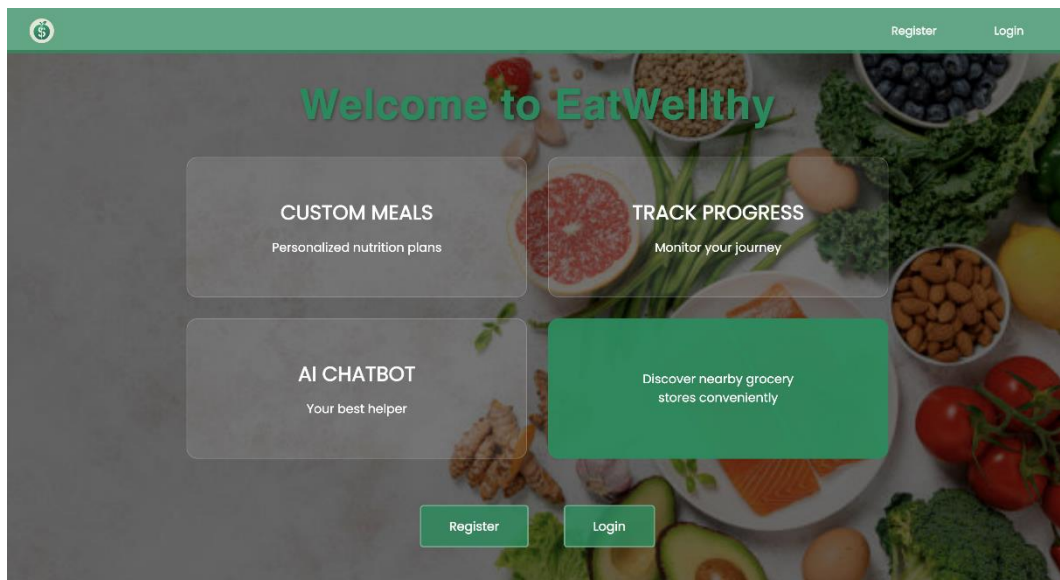
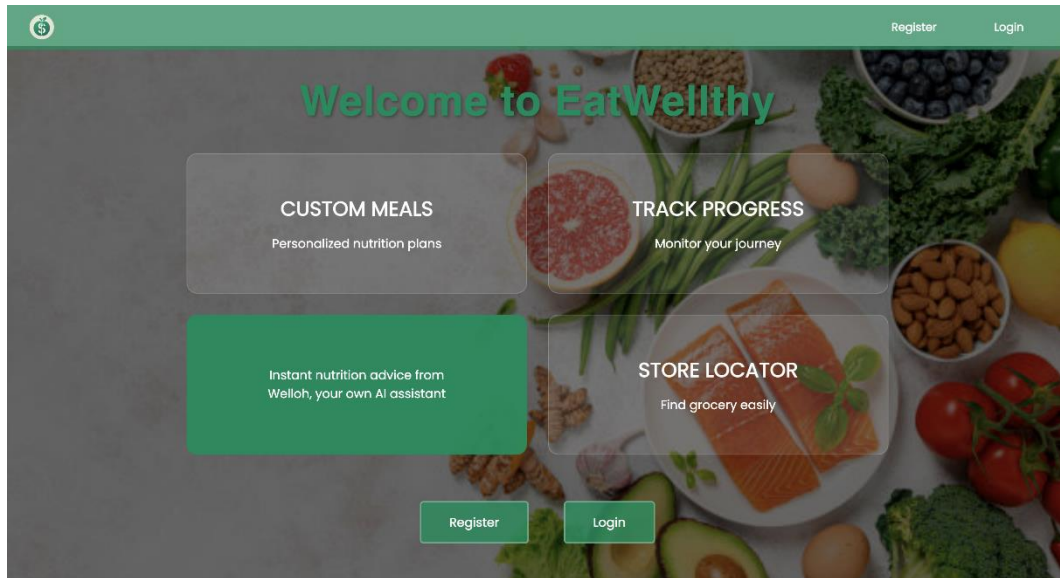
## 3. External Interface Requirements

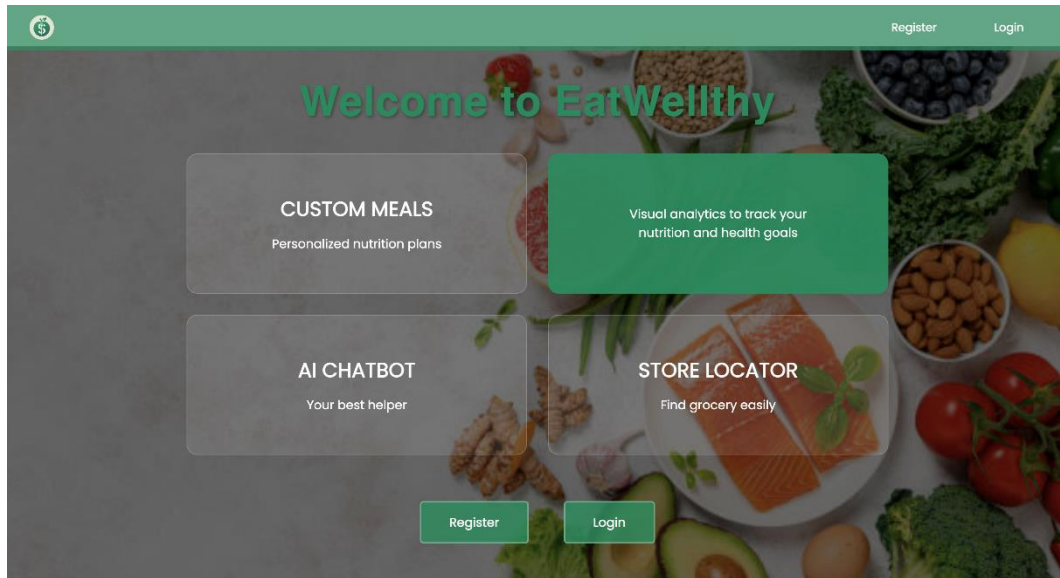
### User Interfaces

This section will showcase all GUI of EatWellthy's web application.

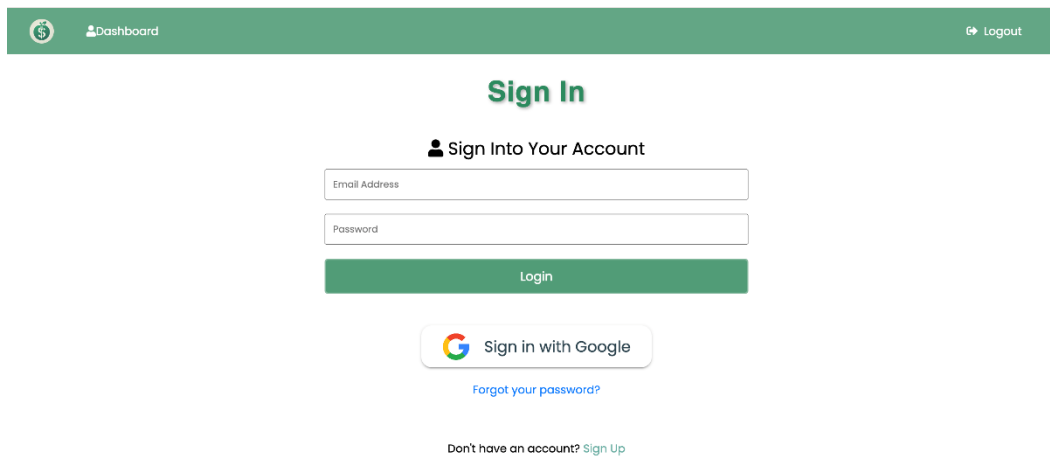
#### 3.1.1 Landing Page





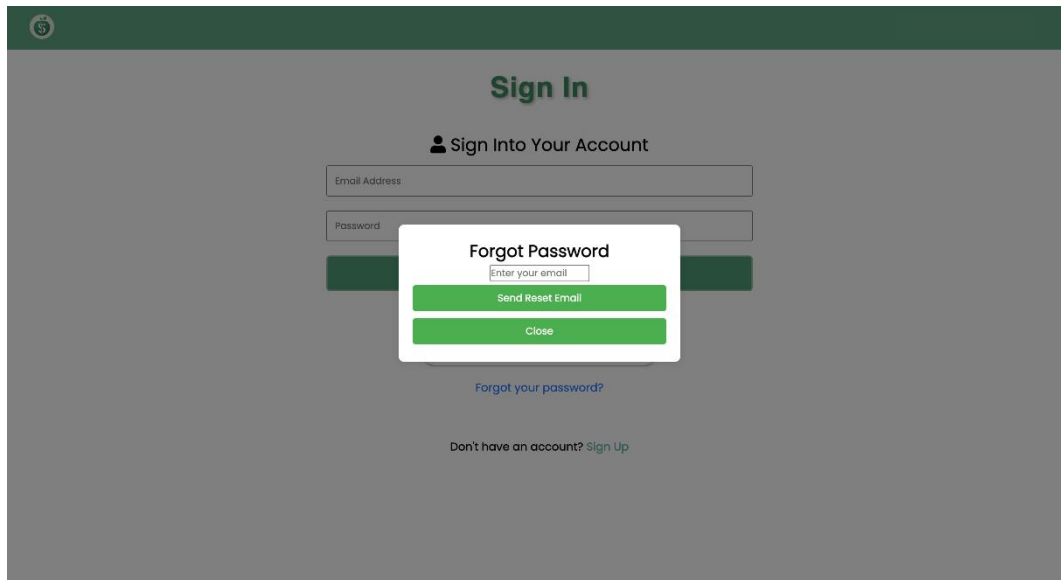


### 3.3.2 Login Page



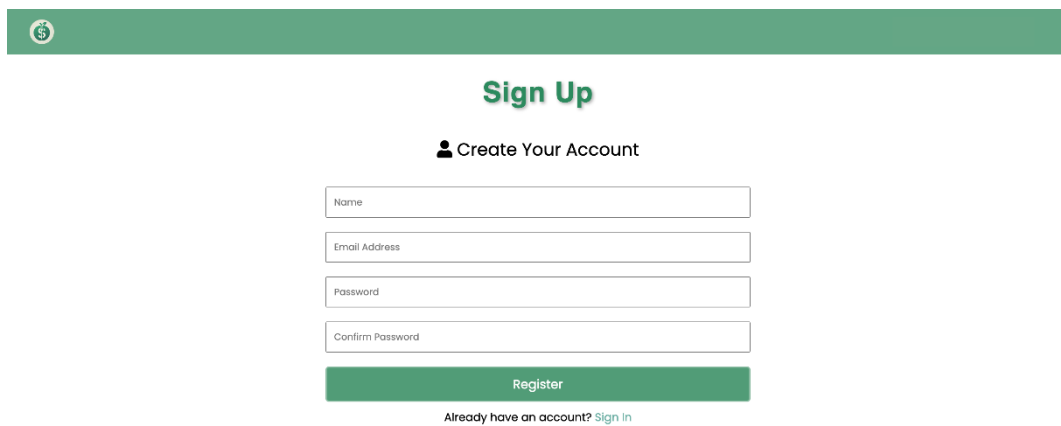
### 3.3.3 Forgot Password Page





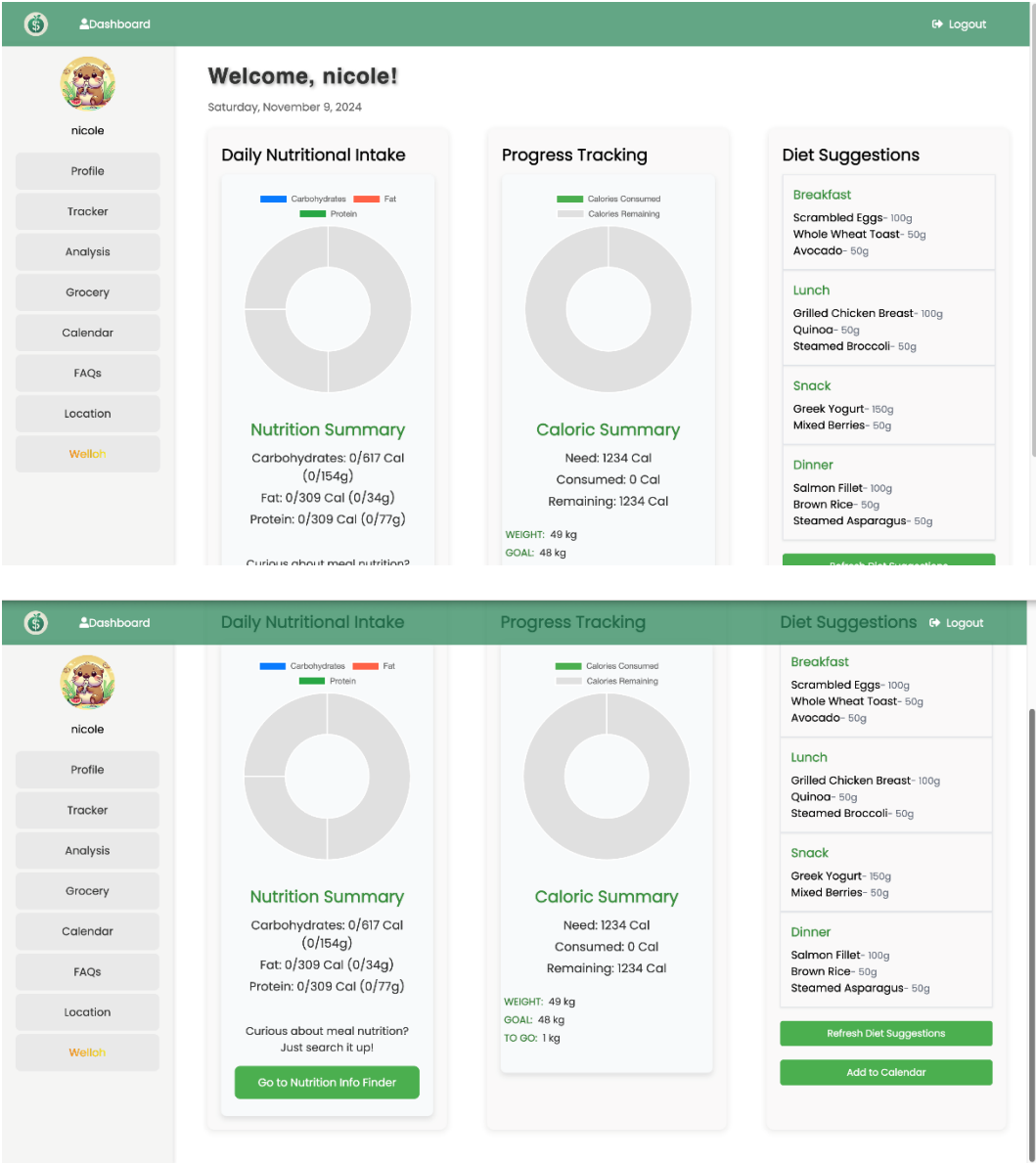
The screenshot shows a web application interface for signing in. At the top, there is a dark green header with a circular logo on the left. The main content area has a light gray background. The title "Sign In" is centered in a bold, dark green font. Below it, the subtitle "Sign Into Your Account" is centered, accompanied by a user icon. There are two input fields: "Email Address" and "Password". A green button is partially visible below the password field. A white modal box titled "Forgot Password" is open in the center, containing an "Enter your email" input field, a green "Send Reset Email" button, and a green "Close" button. Below the modal, there is a link "Forgot your password?" and a text "Don't have an account? Sign Up".

### 3.3.4 Registration Page





The screenshot shows a web application interface for signing up. At the top, there is a green header with a circular logo on the left. The main content area has a light gray background. The title "Sign Up" is centered in a bold, dark green font. Below it, the subtitle "Create Your Account" is centered, accompanied by a user icon. There are four input fields: "Name", "Email Address", "Password", and "Confirm Password". A green button labeled "Register" is positioned below the input fields. At the bottom, there is a link "Already have an account? Sign In".

### 3.3.4 Dashboard Page



3.3.5 Profile Page

 Dashboard

  
nicole

Profile

Tracker

Analysis

Grocery

Calendar

FAQs

Location

Wellch


## Profile Settings


### Basic Information

Name nicole	Gender Female	Age 20
Height (cm) 152	Current Weight (kg) 49	Target Weight (kg) 48
Daily Budget (\$GD) 40	Activity Level Sedentary (little or no exerci	Diet Plan Weight Loss Plan

Dietary Preferences

Allergies (comma separated)

 Dashboard

  
nicole

Profile

Tracker

Analysis

Grocery


Calendar









FAQs

Location

Wellch

## Choose Your Profile Icon

Current Icon: 

 Bear	 Capybara	 Cat	 Dog
 Otter	 Panda	 Rabbit	 Tiger

Update Profile

## Change Password

New Password

nicole

- Profile
- Tracker
- Analysis
- Grocery
- Calendar
- FAQs
- Location
- Welloh

Update Profile

### Change Password

New Password

Confirm Password

Update Password

### Delete Account

This action cannot be undone. Please be certain.

Delete My Account

### 3.3.6 Nutrition Tracker Page

nicole

- Profile
- Tracker
- Analysis
- Grocery
- Calendar
- FAQs
- Location
- Welloh

## Nutrition Tracker

Recent meals
Tracker
History Search
Customised Food

### Today's Meals

Meal Type	Food	Amount	Energy(kcal)	Fat(g)	Sugar(g)	Fiber(g)	Protein(g)	Sodium(mg)	Vitamin C(mg)	Calcium(mg)	Iron(mg)
snacks	apple	1	95	0	19	4	0	2	8	11	0
Daily Totals:			95	0	19	4	0	2	8	11	0

### Yesterday's Meals

Meal Type	Food	Amount	Energy(kcal)	Fat(g)	Sugar(g)	Fiber(g)	Protein(g)	Sodium(mg)	Vitamin C(mg)	Calcium(mg)	Iron(mg)
No meals recorded											

### Two Days Ago Meals

Meal Type	Food	Amount	Energy(kcal)	Fat(g)	Sugar(g)	Fiber(g)	Protein(g)	Sodium(mg)	Vitamin C(mg)	Calcium(mg)	Iron(mg)
No meals recorded											

Dashboard

nicole

Profile

Tracker

Analysis

Grocery

Calendar

FAQs

Location

Wellch

Nutrition Tracker

Recent mealsTrackerHistory SearchCustomised Food

Time: dd/mm/yyyy, ---

Meal Type: -- Select a meal type -

Food Taken: Search or type food

Portion Size: Enter portion size

Add

What you took:

Meal Type	Food	Amount	Energy(kcal)	Fat(g)	Sugar(g)	Fiber(g)	Protein(g)	Sodium(mg)	Vitamin C(mg)	Calcium(mg)	Iron(mg)
Total Nutrition:			0	0	0	0	0	0	0	0	0

Dashboard

nicole

Profile

Tracker

Analysis

Grocery

Calendar

FAQs

Location

Wellch

Nutrition Tracker

Recent mealsTrackerHistory SearchCustomised Food

Search Date: 09/11/2024

Search

What you took:

Meal Type	Food	Amount	Energy(kcal)	Fat(g)	Sugar(g)	Fiber(g)	Protein(g)	Sodium(mg)	Vitamin C(mg)	Calcium(mg)	Iron(mg)
snacks	apple	1	94.64	0.31	18.91	4.37	0.47	1.82	8.372	10.92	0.2184
Total Nutrition:			94.64	0.31	18.91	4.37	0.47	1.82	8.372	10.92	0.2184

Dashboard

Logout

nicole

Profile

Tracker

Analysis

Grocery

Calendar

FAQs

Location

Welloh

### Nutrition Tracker

Recent mealsTrackerHistory SearchCustomised Food

#### My Custom Foods

Name	Energy(kcal)	Fat(g)	Sugar(g)	Fiber(g)	Protein(g)	Sodium(mg)	Vitamin C(mg)	Calcium(mg)	Iron(mg)	Actions
otah	180	12	1	1	10	300	0	15	1	<div>Edit</div> <div>Delete</div>

Add New Custom Food

### 3.3.7 Nutrition Analysis Page

Dashboard

Logout

nicole

Profile

Tracker

Analysis

Grocery

Calendar

FAQs

Location

Welloh

### Your Nutrition Analysis Report

Height: 152cm

Weight: 49kg

Activity: Sedentary (little or no exercise)

Daily Calories

1542

calories/day

Maintenance Plan

BMI

21.2

Normal weight

Maintain a balanced diet with regular exercise.

BMR

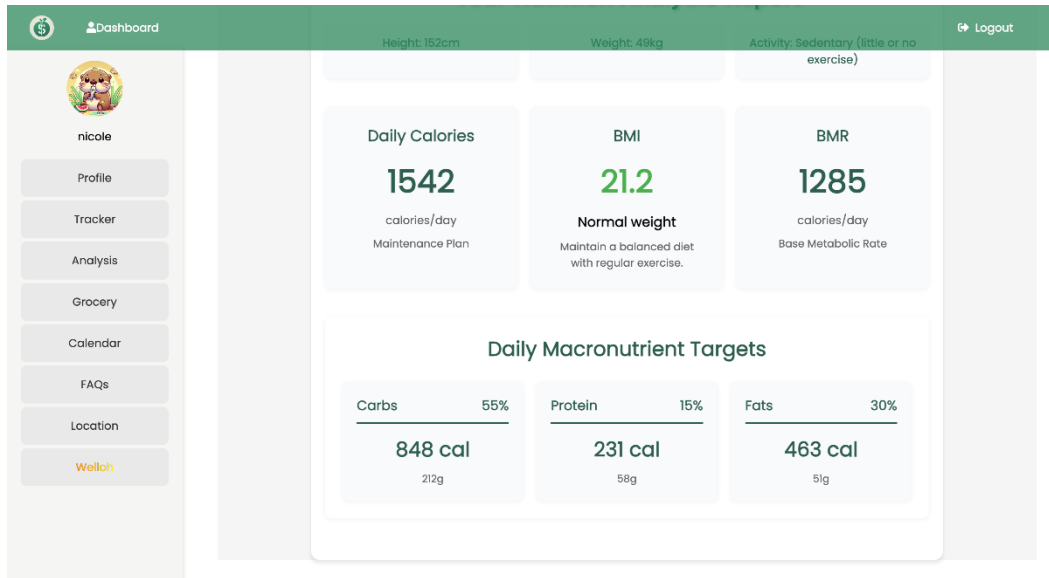
1285

calories/day

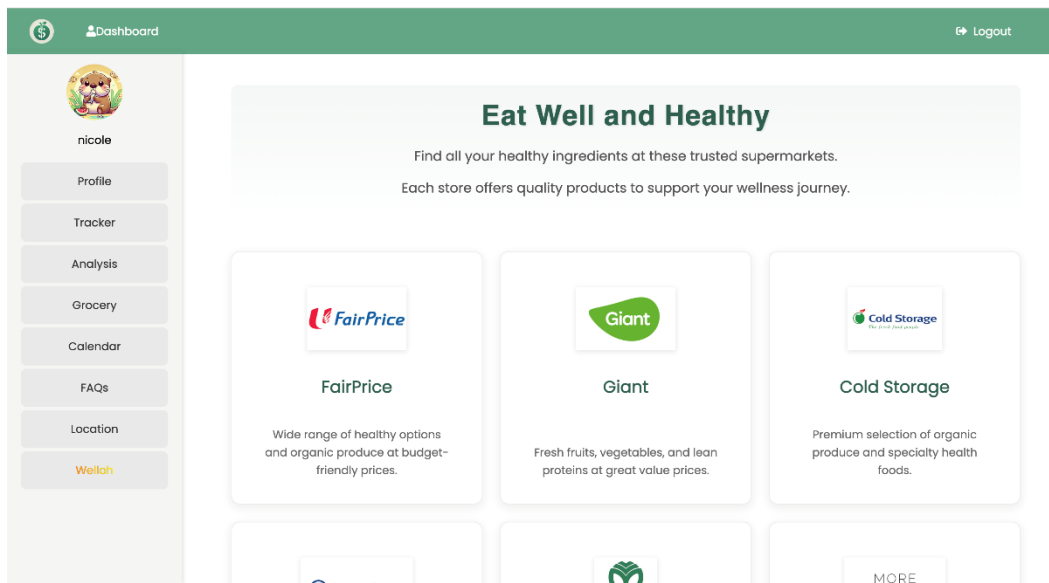
Base Metabolic Rate

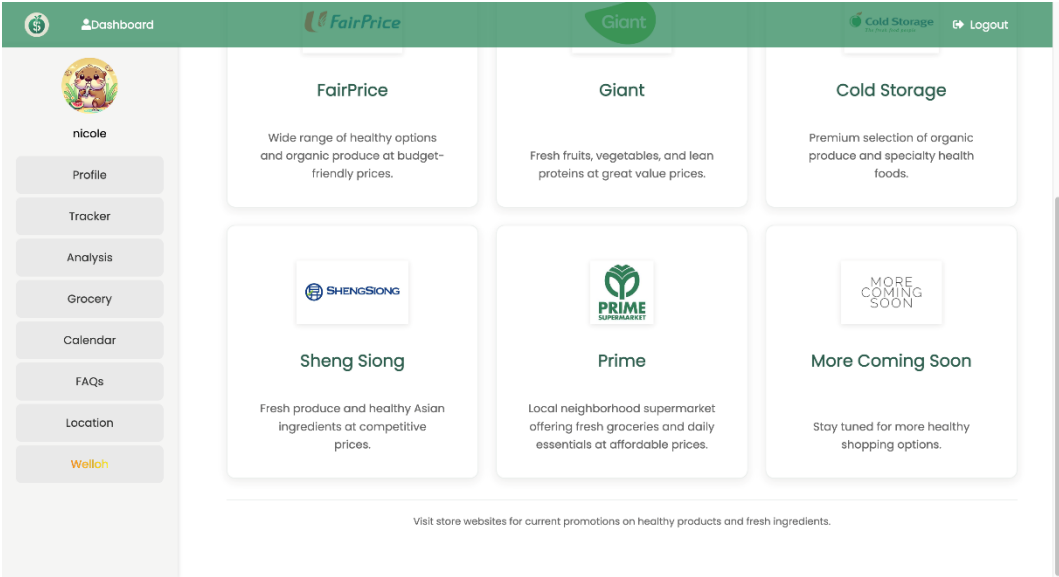
#### Daily Macronutrient Targets

Carbs	55%	Protein	15%	Fats	30%
848 cal		231 cal		463 cal	

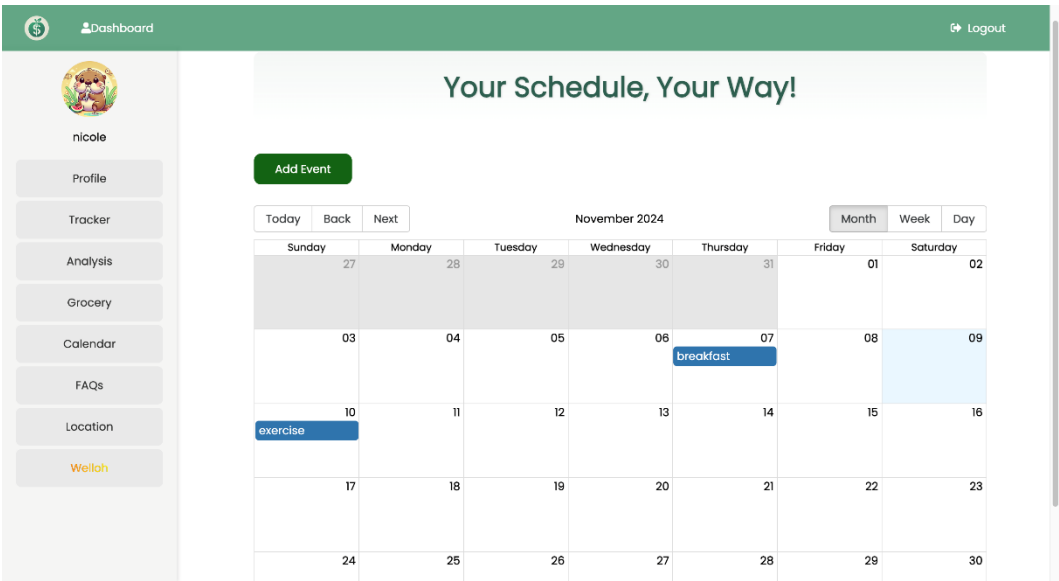


### 3.3.8 Grocery Page

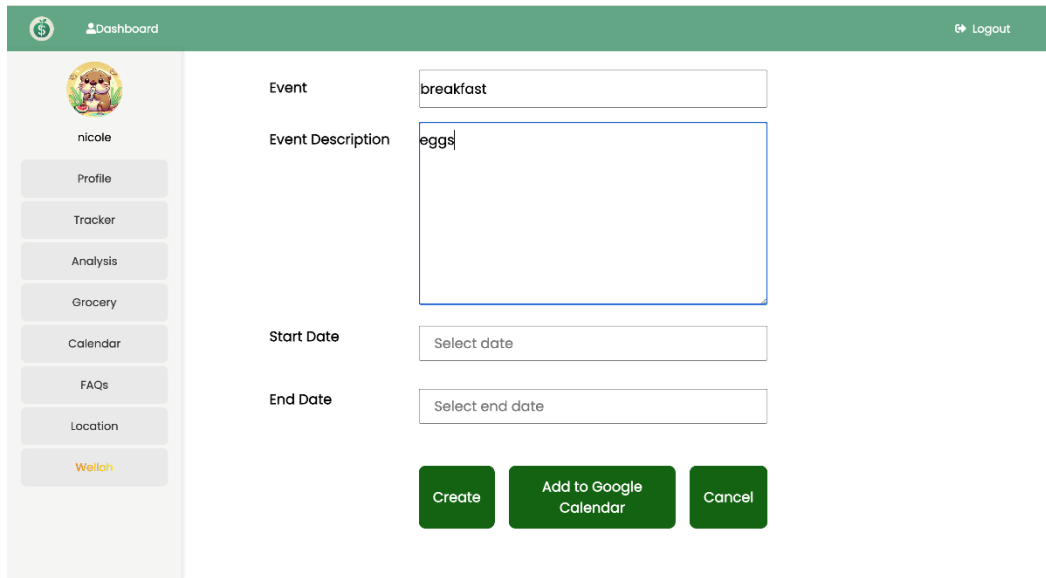




3.3.9 Calendar Page





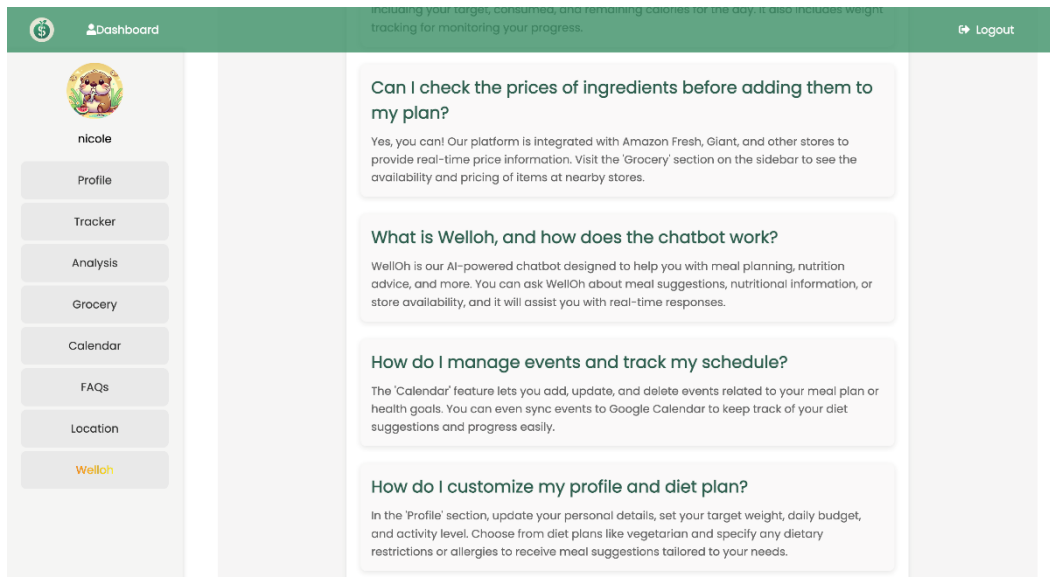
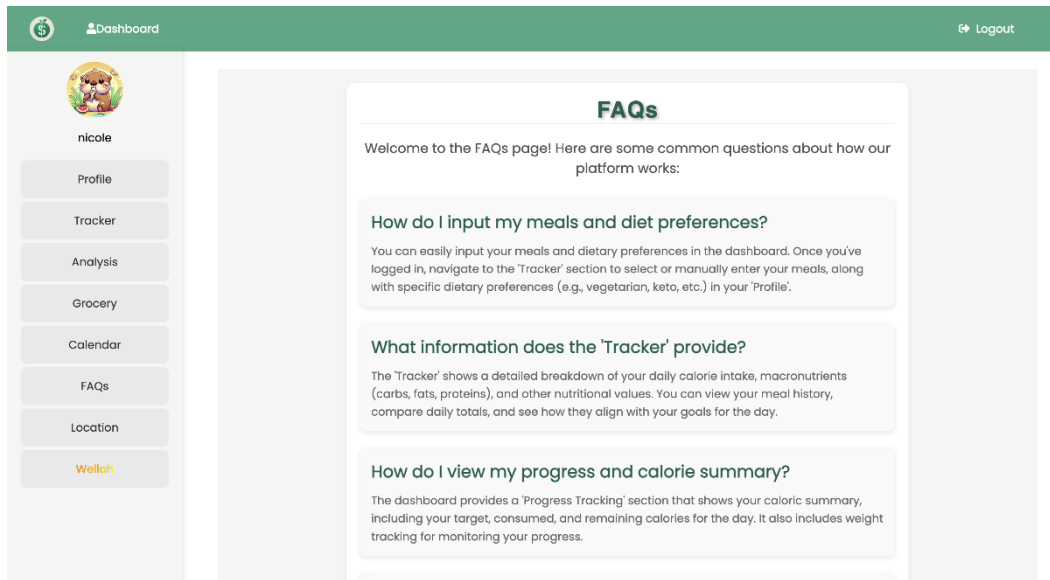


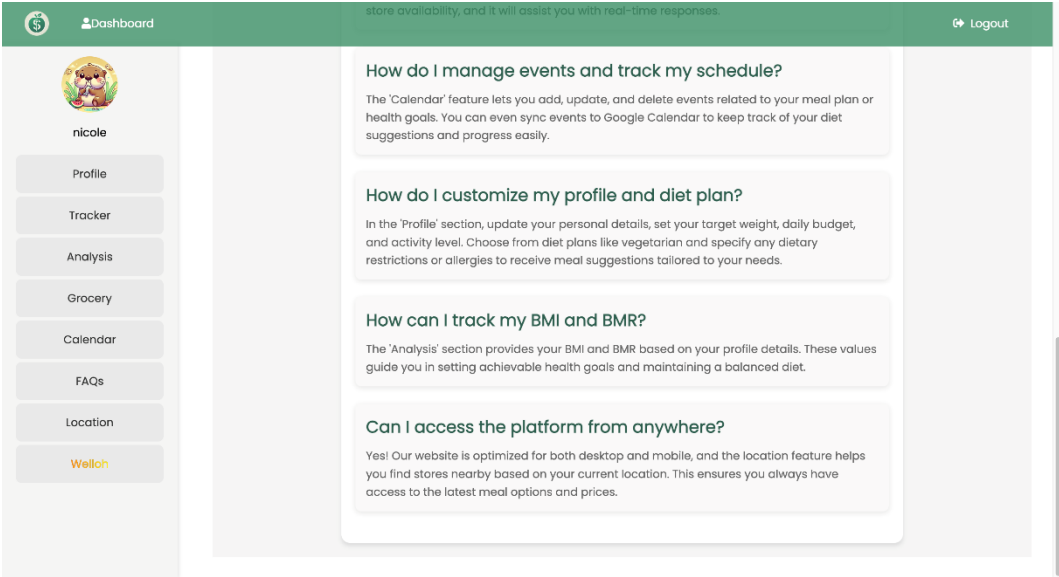
The screenshot displays the 'Event' creation page in the EatWellthy application. The interface features a green header bar with a 'Logout' link. A left sidebar contains a user profile for 'nicole' and a menu with options: Profile, Tracker, Analysis, Grocery, Calendar, FAQs, Location, and Wellth. The main content area is titled 'Event' and contains the following fields:

- Event:** A text input field containing the value 'breakfast'.
- Event Description:** A large text area containing the value 'eggs'.
- Start Date:** A date selection field with the placeholder text 'Select date'.
- End Date:** A date selection field with the placeholder text 'Select end date'.

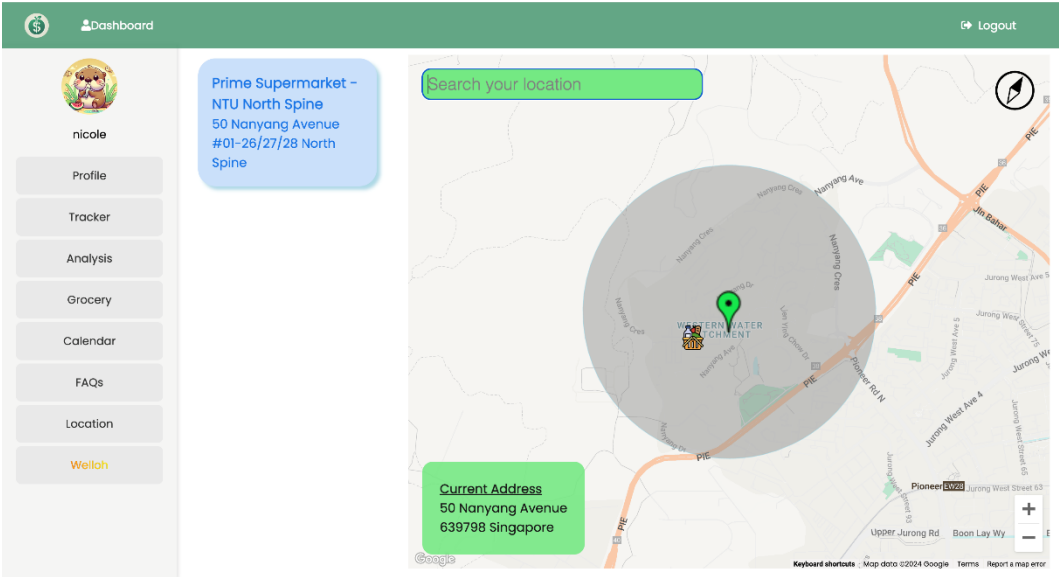
At the bottom of the form, there are three green buttons: 'Create', 'Add to Google Calendar', and 'Cancel'.

### 3.3.10 FAQs Page

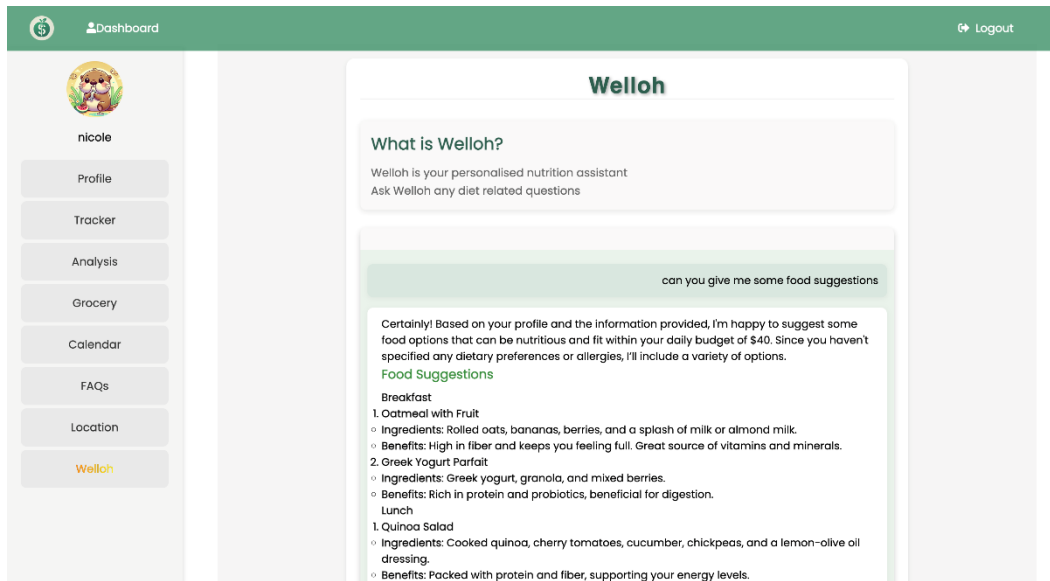




3.3.11 Location



### 3.3.12 Welloh



## Hardware Interfaces

This section describes the hardware interfaces required for the EatWellthy website to perform effectively and reliably. These interfaces cover the devices through which users access the platform, the data and control interactions that facilitate system operations, and the communication protocols that ensure smooth connectivity and performance.

Requirements	Description
Device Types	<ul style="list-style-type: none"> <li>Desktop &amp; Laptop Computers that are compatible with standard web browsers (e.g., Chrome, Firefox, Safari) on Windows, macOS and Linux systems</li> <li>Tablets &amp; Mobile Devices with browsers on Android and iOS systems, ensuring a responsive interface for smaller screens</li> </ul>
Data and Control Interactions	<ul style="list-style-type: none"> <li>The user's device interacts with the software through keyboards, touchscreens and a mouse for input and displays for output</li> </ul>
Communication Protocols	<ul style="list-style-type: none"> <li>The platform utilizes HTTPS to ensure secure communication over the web. API interactions, websocket connections, and other services will rely on standard web ports, with port 80 used for HTTP traffic and port 443 for HTTPS traffic.</li> </ul>
Network Connection	<ul style="list-style-type: none"> <li>The platform requires an active internet connection, facilitated through a Wireless Network Interface Card (WNIC) or a cellular modem chip. This ensures connectivity for secure data transfer and communication with the system.</li> </ul>
Interaction	<ul style="list-style-type: none"> <li>User interaction with the platform is supported through standard input devices, including a functional touchpad or mouse, enabling efficient navigation and control of the website's features.</li> </ul>

## Software Interfaces

### 3.3.1 Operating Systems

Any OS capable of running a modern web browser (Windows, macOS, Linux, iOS, Android).

### 3.3.2 Web Browsers

Latest versions of Chrome, Firefox, Safari, Edge, and others capable of supporting HTML5, CSS3, and JavaScript.

### 3.3.3 Databases

NoSQL database systems, such as MongoDB, accessed through backend services.

### 3.3.4 Front-end Tools

- React.js (version 18.3.1)
- CSS3 for Styling
- Prettier and ESLint (version 8.3.0) for code formatting

### 3.3.5 Back-end Tools

- MongoDB (version 6.9.0)
- Express.js (version 4.21.0)
- Postman (version 11.19) for backend route testing

### 3.3.6 RESTful APIs

Our application communicates between the frontend and backend through RESTful API endpoints, which are directly implemented in the code. While there is no separate documentation for the API, the endpoints are designed to handle specific HTTP methods (GET, POST, PUT, DELETE) with structured request and response bodies. These bodies are defined within the backend code itself using JavaScript/Node.js, ensuring that the API follows a consistent format for sending data from the frontend and returning data from the server. Developers can refer to the code for the expected input and output structures, as all the necessary details are embedded within the API functions and endpoints.

### 3.3.7 Data Sharing

All data exchange between the frontend and backend is conducted in JSON format. Data persistence is managed through a NoSQL database, MongoDB, which stores data in flexible, schema-less collections. The data models are dynamically defined within the backend code, allowing for efficient handling and retrieval of data without a fixed structure. MongoDB's document-oriented storage ensures scalability and adaptability to evolving data requirements.

## Communications Interfaces

For EatWellthy, several communication interfaces are crucial for a smooth and interactive user experience:

- **WebSocket:** WebSocket enables real-time, bi-directional interaction between clients and the server, enhancing the responsiveness of live updates and notifications.
- **RESTful API:** It facilitates communication between clients and server by handling requests and responses in a stateless, cacheable, and uniform interface.
- **Electronic Forms:** Any form data is transmitted over HTTPS to protect the integrity and confidentiality of user-submitted data.
- **Network Protocols:** TCP/IP (Transmission Control Protocol/Internet Protocol) is used implicitly by HTTP/HTTPS and WebSockets to ensure reliable, ordered, and error-checked delivery of a stream of data between the users and the application.
- **HTTP/HTTPS:** Foundational protocols for data communication on the World Wide Web. HTTPS is the secure version of HTTP and encrypts data sent between the client and server, ensuring that all user interactions are secure from any interception.
- **Security and Encryption:** OAuth is used for secure authorization in a simple and standardized way from web, mobile, and desktop applications. JSON Web Tokens (JWT) facilitates token-based authentication in web and mobile applications, ensuring all server requests are authenticated and authorized securely.

## 4. System Features

This section will cover the core system features of EatWellthy.

## Account Registration

### Description and Priority

**Description:** Users must be registered before they proceed to log in to use the app.

**Priority:** High

### Stimulus/Response Sequences

Use Case ID:	U0101		
Use Case Name:	New User Registration using email		
Created By:	Zhao Qixian	Last Updated By:	Mahi Pandey
Date Created:	29/8/2024	Date Last Updated:	09/11/24

Actor:	New User
Description:	It enables new users to create an account within EatWellthy's system database. To create an account, users are required to provide their Name, Email, and Password. New Users will be assigned a unique UserID.
Preconditions:	<ol style="list-style-type: none"> <li>1. Email address must be valid, and user must have access to it.</li> <li>2. Passwords must be 6 characters in length.</li> <li>3. The supplied Email must not be associated with an existing user account.</li> </ol>
Postconditions:	<ol style="list-style-type: none"> <li>1. Upon a successful registration, the system seamlessly executes the following actions:               <ol style="list-style-type: none"> <li>a. Instantly dispatches a 6-digit code to the user's provided Email address, ensuring account verification.</li> <li>b. Asks for the 6-digit code</li> </ol> </li> <li>2. Once Account is verified, redirection to dashboard.</li> <li>3. The user is assigned a unique UserID.</li> <li>4. The user is now equipped to access the app's features, using their newly created Username and Password, or their Email and Password.</li> <li>5. The user's UserID, email, Username and encrypted Password are stored in the database.</li> </ol>
Priority:	High
Frequency of Use:	One-off
Flow of Events:	<ol style="list-style-type: none"> <li>1. The new user opens EatWellthy and selects the "New User Register" option.</li> <li>2. The system presents the user with a registration form, soliciting the following details:               <ul style="list-style-type: none"> <li>• Name</li> <li>• Email</li> <li>• Password</li> </ul> </li> <li>3. The user fills in the necessary information and clicks "Register".</li> </ol>

	<p>4. Upon a successful registration, the system seamlessly executes the following actions:</p> <ol style="list-style-type: none"> <li>Instantly dispatches a 6-digit code to the user's provided Email address, ensuring account verification.</li> <li>Asks for the 6-digit code</li> </ol> <p>5. Once Account is verified, redirection to dashboard.</p> <ul style="list-style-type: none"> <li>If any precondition is not met, the system displays precise error messages, guiding the user to correct their input.</li> <li>If the input is valid, the registration process advances, and the user is directed to email verification.</li> </ul>
Alternative Flows:	<p>1. Validation Errors</p> <ul style="list-style-type: none"> <li>If user inputs don't meet preconditions (e.g., invalid password, duplicate Username, existing Email), the system displays error messages for correction.</li> <li>The user corrects input and resubmits for validation.</li> <li>The process continues when all input is valid.</li> </ul> <p>2. Password Reset</p> <ul style="list-style-type: none"> <li>Users can initiate a password reset if forgotten or facing login issues.</li> <li>The system sends a temporary password to the user's email.</li> <li>User can use the temporary password to log in and then change their password under Profile section.</li> </ul> <p>3. Confirmation Link Resend:</p> <ul style="list-style-type: none"> <li>Users can request a resend of the confirmation code every 60 seconds.</li> <li>The system sends a new confirmation code to their Email.</li> </ul>
Exceptions:	<p>1. Email Not Received:</p> <ol style="list-style-type: none"> <li>If the user does not receive the confirmation code, they may request a resend. If the issue persists, the system advises the user to check spam/junk folders or contact support.</li> </ol>
Includes:	<ol style="list-style-type: none"> <li>Email &amp; Password Verification</li> <li>Display Success/Error Registration</li> </ol>
Special Requirements:	<ol style="list-style-type: none"> <li>The system must ensure that passwords are stored securely using encryption.</li> <li>The system should handle email delivery promptly and securely.</li> </ol>



Assumptions:	<ol style="list-style-type: none"> <li>1. It is assumed that the user has a valid and accessible email address.</li> <li>2. It is assumed that the user understands and complies with the password requirements.</li> </ol>
Notes and Issues:	

## Functional Requirements

### 1. Registration.

1.1. Users must be able to register with an email address.

1.1.1 System must provide three input fields to input information.

1.1.1.1. One of the input fields must be a username.

1.1.1.2. One of the input fields must be an email address.

1.1.1.3. One of the input fields must be a password.

1.1.2. System must ensure that the user fills in all the input fields before allowing registration.

1.1.3. System must verify all input information.

1.1.3.1. System must verify that the username is unique.

1.1.3.2. System must verify that the email meets the requirements.

1.1.3.2.1. System must verify that the email is valid with a character “@”.

1.1.3.2.2. System must verify that the email is unique.

1.1.3.2.3. System must direct the user to the LogIn Page when the input email is linked to a registered account.

1.1.3.3. System must verify that the password meets the requirements.

1.1.3.3.1. The password must be a minimum of 8 characters

1.1.3.3.2. The password shall have at least one uppercase letter

1.1.3.3.3. The password shall have at least one lowercase letter

1.1.3.3.4. The password shall have at least one number

1.1.3.3.5. The password shall have at least one special character (e.g., !, @, #, \$)

1.1.4. System must ask the user to input information again when their previous input failed the verification.

1.1.5 System must send a six digits OTP to verify the user's email address.

1.1.5.1. System must provide one input field for the user to input the OTP received in their email.

1.1.5.2. System must provide a request button for the user to request the OTP again.

1.1.5.3. System must ensure that OTP is valid for a limited period.

1.1.6. System must register an account for the user when the user email is verified.

1.1.6.1 System must assign a unique UserID to the user's account.

1.1.6.2. System must record the information to the user's account.

1.1.7 System must redirect user to Login Page when user's account is created.

## Account Login

### Description and Priority

**Description:** A user account must be verified with an existing database.

**Priority:** High

### Stimulus/Response Sequences

Use Case ID:	U0102		
Use Case Name:	User Login via Email		
Created By:	Mahi Pandey	Last Updated By:	Mahi Pandey
Date Created:	01/09/2024	Date Last Updated:	02/09/2024

Actor:	Registered User
Description:	This use case enables registered users to log into the EatWellthy system using their email and password. Successful login grants access to the app's features.
Preconditions:	<ol style="list-style-type: none"> <li>1. The user must have a valid registered account.</li> <li>2. The user must provide a valid email and password.</li> </ol>
Postconditions:	<ol style="list-style-type: none"> <li>1. The user is successfully logged into the system.</li> <li>2. The user's session is authenticated and maintained</li> <li>3. The user is redirected to the EatWellthy main page</li> </ol>
Priority:	High
Frequency of Use:	Every time user wishes to login, Multiple Times
Flow of Events:	<ol style="list-style-type: none"> <li>1. The user opens the EatWellthy app and selects the Login option.</li> <li>2. The system presents the login form, requesting Email and Password.</li> <li>3. The user enters their Email and Password and clicks "Login"</li> <li>4. The system validates the credentials. <ul style="list-style-type: none"> <li>• If valid, the system logs in the user and redirects them to the main page.</li> <li>• If invalid, the system displays an error message.</li> </ul> </li> </ol>
Alternative Flows:	<ol style="list-style-type: none"> <li>1. Invalid Credentials</li> </ol>

	<ul style="list-style-type: none"> <li>The system informs the user of incorrect Email or Password and prompts re-entry.</li> </ul> <ol style="list-style-type: none"> <li>2. Password Reset           <ul style="list-style-type: none"> <li>If the user forgets their password, they can select “Forgot Password” to initiate a reset process.</li> </ul> </li> </ol>
Exceptions:	<ol style="list-style-type: none"> <li>1. If the system is down or the database is unreachable, an error message is displayed, and the user is asked to try again later.</li> </ol>
Includes:	<ol style="list-style-type: none"> <li>1. Password Verification</li> <li>2. Display Success/Error Login</li> </ol>
Special Requirements:	The system must ensure secure transmission of login credentials.
Assumptions:	It is assumed that the user has registered and verified their email address.
Notes and Issues:	

## Functional Requirements

### 1.2. Login

#### 1.2.1. Users must be able to log in with an email address.

1.2.1.1. System must provide two input fields to input login credentials.

1.2.1.1.1. One of the input fields must be for the email address.

1.2.1.1.2. One of the input fields must be for the password.

1.2.1.2. System must ensure that the user fills in both input fields before allowing login.

1.2.1.3. System must verify the email and password provided by the user.

1.2.1.3.1. System must verify that the email exists in the system.

1.2.1.3.2. System must verify that the password matches the one associated with the provided email.

1.2.1.3.3. System must allow the user three attempts to input the correct password.

1.2.1.3.3.1. System must lock the account for a specified period after three failed login attempts.

1.2.1.3.3.2. System must notify the user to reset password via email when the account is locked.

1.2.1.4. System must redirect the user to the Home Page upon successful login.

1.2.1.5. System must provide a “Forgot Password” option.

1.2.1.5.1. System must provide one input field for the user to input their registered email address.

1.2.1.5.2. System must send a password reset link to the provided email.

1.2.1.5.3. System must verify the password reset request by asking the user to

input a new password that meets the password requirements (as defined in Section 1.1.3.3).

## Google OAuth

### Description and Priority

**Description:** Users have the option to sign in with their Google credentials, which eliminates the need for them to remember additional details like their password.

**Priority:** Medium

### Stimulus/Response Sequences

Use Case ID:	U0103		
Use Case Name:	User Login via Google Account		
Created By:	Mahi Pandey	Last Updated By:	Mahi Pandey
Date Created:	02/09/2024	Date Last Updated:	09/11/2024

Actor:	Registered User
Description:	This use case allows users to log into the EatWellthy system using their Google Account. Upon successful login, users are granted access to the website's features.
Preconditions:	<ol style="list-style-type: none"> <li>1. The user must have a Google account.</li> <li>2. The Google account must be linked to an EatWellthy user profile.</li> </ol>
Postconditions:	<ol style="list-style-type: none"> <li>1. The user is successfully logged into the system.</li> <li>2. The user session is authenticated and maintained.</li> <li>3. The user is redirected to the EatWellthy main page.</li> </ol>
Priority:	Medium
Frequency of Use:	Every time user wishes to login, Multiple Times
Flow of Events:	<ol style="list-style-type: none"> <li>1. The user selects "Login with Google".</li> <li>2. System redirects the user to Google's login page.</li> <li>3. The user enters their Google credentials and authorises access.</li> <li>4. System retrieves the user's Google Account. <ul style="list-style-type: none"> <li>• If valid, the user is logged in and redirected to the main page.</li> <li>• If invalid, an error message is displayed.</li> </ul> </li> </ol>
Alternative Flows:	<ol style="list-style-type: none"> <li>1. Google Authorization Failure</li> </ol>

	<ul style="list-style-type: none"> <li>If the user fails to log into their Google Account, the system displays an error message.</li> <li>The user can retry or choose another login method</li> </ul>
Exceptions:	1. If Google API is unavailable, the user is informed and asked to try again later or use email login method.
Includes:	<ol style="list-style-type: none"> <li>Google Authorization</li> <li>Display Success/Error Login</li> </ol>
Special Requirements:	System must comply with Google's OAuth 2.0 standards.
Assumptions:	The user has a valid and accessible Google account.
Notes and Issues:	

### Functional Requirements

1.3 Users must be able to log in via their Google Account.

1.3.1. System must provide one input field to input a Google email address.

1.3.1.1. System must verify that the Google email is valid and ends with "[@gmail.com](mailto:)."

1.3.2 System must verify that the Google email is linked to an existing account in the system.

1.3.3. System must redirect the user to the Home Page upon successful login via Google.

### Reset Forgotten Password

#### Description and Priority

**Description:** The application can function without this feature

**Priority:** Medium

#### Stimulus/Response Sequences

Use Case ID:	U0104		
Use Case Name:	Reset Forgotten Password		
Created By:	Mahi Pandey	Last Updated By:	Mahi Pandey
Date Created:	02/09/2024	Date Last Updated:	09/11/2024

Actor:	Registered User
--------	-----------------

Description:	This use case enables users who have forgotten their password to reset it using their registered email address.
Preconditions:	<ol style="list-style-type: none"> <li>1. The user must have a registered account with EatWellthy.</li> <li>2. The user must have access to the registered email address.</li> </ol>
Postconditions:	<ol style="list-style-type: none"> <li>1. The user successfully resets their password.</li> <li>2. The user is prompted to log in with the new password.</li> </ol>
Priority:	Medium
Frequency of Use:	Infrequent, as needed
Flow of Events:	<ol style="list-style-type: none"> <li>1. The user selects "Forgot Password" on the login page.</li> <li>2. The system prompts the user to enter their registered email.</li> <li>3. The user enters their email and clicks "Submit".</li> <li>4. The system verifies the email and sends a temporary password.</li> <li>5. The user receives the email and uses the temporary password to log into their account.</li> <li>6. Under Profile, the user can now update their password.</li> </ol>
Alternative Flows:	<ol style="list-style-type: none"> <li>1. Invalid Email <ul style="list-style-type: none"> <li>• If the entered email is not registered, the system informs the user and prompts for re-entry.</li> </ul> </li> </ol>
Exceptions:	If the system cannot send the reset email, the user is informed of a delay and asked to try again later.
Includes:	<ol style="list-style-type: none"> <li>1. Password Reset</li> </ol>
Special Requirements:	The system must ensure secure handling of password reset requests.
Assumptions:	The user can access their registered email account.
Notes and Issues:	Consider implementing a rate limit for password reset requests to prevent abuse.

### Functional Requirements

1.4. System must provide a "Forgot Password" option.

1.4.1. System must provide one input field for the user to input their registered email address.

1.4.2. System must send a temporary password to the provided email.

1.4.3. System must verify the password reset request by asking the user to input the temporary password that meets the password requirements (as defined in Section 1.1.3.3).

## Update Password

### Description and Priority

**Description:** Users should be able to update their passwords which ensures a secure authentication process.

**Priority:** Medium

### Stimulus/Response Sequences

Use Case ID:	U0105		
Use Case Name:	Change Account Password		
Created By:	Mahi Pandey	Last Updated By:	Mahi Pandey
Date Created:	02/09/2024	Date Last Updated:	09/11/2024

Actor:	Registered User
Description:	This use case allows registered users to change their account password while logged into the system.
Preconditions:	<ol style="list-style-type: none"><li>1. The user must be logged into their account.</li><li>2. The user must know their current password.</li></ol>
Postconditions:	<ol style="list-style-type: none"><li>1. The user's password is successfully changed.</li><li>2. The system logs the user out and prompts them to log in again with the new password.</li></ol>
Priority:	Medium
Frequency of Use:	Infrequent, as needed
Flow of Events:	<ol style="list-style-type: none"><li>1. The user navigates to the "Profile" page.</li><li>2. The system presents the option to change password.</li><li>3. The user enters the new password, then confirms the new password.</li><li>4. The system validates the current password and checks the new password against security requirements.</li><li>5. If valid, the system updates the password and logs the user out.</li><li>6. The user is prompted to log in with new password.</li></ol>
Alternative Flows:	<ol style="list-style-type: none"><li>1. Password Strength Failure<ul style="list-style-type: none"><li>• If the new password does not meet security requirements, the system prompts the user to enter a stronger password.</li></ul></li><li>2. Passwords Do Not Match<ul style="list-style-type: none"><li>• If the two passwords do not match each other, an error message is displayed.</li></ul></li></ol>

Exceptions:	If the system is unable to update the password due to technical issues, an error message is displayed, and the user is asked to try again later.
Includes:	1. Verify Original Password
Special Requirements:	The system must ensure secure handling and storage of passwords.
Assumptions:	The user has access to their current password.
Notes and Issues:	Provide feedback on the strength of the password.

### Functional Requirements

1.5. The system shall provide an option to change the password in the user profile section under "Change Password."

1.5.1. The system shall only allow access to the "Update Password" section if the user is currently logged in.

1.5.2. The system shall display input fields for the new password and confirm new password.

1.5.3. The system shall require the new password to meet security criteria, minimum of 6 characters.

1.5.4 The system shall check that the new password and confirm password fields match before allowing submission.

1.5.5 If the new password does not meet security requirements, the system shall display an error message prompting the user to strengthen the password.

1.5.6 If the new password and confirmation password do not match, the system shall display an error message: "Passwords do not match."

### Delete User Account

#### Description and Priority

**Description:** Users should be able to permanently delete their account from the EatWellthy database

**Priority:** Medium

#### Stimulus/Response Sequences

Use Case ID:	U0106		
Use Case Name:	Delete User Account		
Created By:	Mahi Pandey	Last Updated By:	Mahi Pandey
Date Created:	02/09/2024	Date Last Updated:	02/09/2024

Actor:	Logged-in & Registered User
Description:	This use case allows users to permanently delete their EatWellthy account. Deleting an account removes all user data from the system and makes the account irretrievable.



Preconditions:	<ol style="list-style-type: none"> <li>1. The user must be logged into EatWellthy account.</li> <li>2. The user must confirm their intention to permanently delete the account.</li> <li>3. The system must inform user of the consequences of deletion, including the irretrievable loss of data.</li> </ol>
Postconditions:	<ol style="list-style-type: none"> <li>1. The user's account and all associated data are permanently removed from the system.</li> <li>2. The user is logged out and cannot recover the account or data.</li> </ol>
Priority:	Medium
Frequency of Use:	Infrequent, as needed
Flow of Events:	<ol style="list-style-type: none"> <li>1. The user navigates to "Profile" and selects the "Delete Account" option.</li> <li>2. The system prompts the user to confirm their decision to permanently delete the account.</li> <li>3. The user confirms the deletion by providing their password for security verification.</li> <li>4. The system warns the user about the permanent nature of account deletion and the loss of all data.</li> <li>5. The user confirms the deletion.</li> <li>6. The system permanently removes the user's account and all associated data from the database.</li> <li>7. The user is logged out and redirected to the homepage with a confirmation of account deletion.</li> </ol>
Alternative Flows:	<ol style="list-style-type: none"> <li>1. Cancellation of Deletion <ul style="list-style-type: none"> <li>• If the user decides not to delete the account after being prompted for confirmation, they can cancel the process, and no changes are made to the account.</li> </ul> </li> </ol>
Exceptions:	If the system fails to delete the account due to technical issues, an error message is displayed, and the user is advised to try again later or contact support.
Includes:	
Special Requirements:	<ol style="list-style-type: none"> <li>1. The system must ensure that the deletion process is secure and irreversible.</li> <li>2. The system should comply with relevant data protection regulations to avoid legal disputes.</li> </ol>
Assumptions:	It is assumed that the user understands the implications of account deletion, including the permanent loss of access and data.
Notes and Issues:	

### Functional Requirements

- 1.6. Users must be able to delete their profile.
  - 1.6.1. Users must be able to delete their account through the profile management interface.
  - 1.6.2. System must delete all of user's data in the database.

## Update Basic Profile Information

### Description and Priority

**Description:** Users should be able to update their personal information such as age, height, weight etc.

**Priority:** High

### Stimulus/Response Sequences

Use Case ID:	U0201		
Use Case Name:	Update Basic Profile Information		
Created By:	Mahi Pandey	Last Updated By:	Mahi Pandey
Date Created:	02/09/2024	Date Last Updated:	02/09/2024

Actor:	Logged-in & Registered User
Description:	This use case allows users to view and update their profile information within the EatWellthy app, including personal details, dietary preferences and activity history.
Preconditions:	<ol style="list-style-type: none"> <li>The user must be logged into their EatWellthy account.</li> </ol>
Postconditions:	<ol style="list-style-type: none"> <li>The user's profile information is displayed on the screen.</li> <li>Dashboard is personalised based on the given information</li> </ol>
Priority:	High
Frequency of Use:	Occasional
Flow of Events:	<ol style="list-style-type: none"> <li>The user selects "Profile" from the dashboard menu.</li> <li>The user manually enters the following information:               <ol style="list-style-type: none"> <li>Name</li> <li>Gender</li> <li>Age</li> <li>Height (cm)</li> <li>Current Weight (kg)</li> <li>Target Weight (kg)</li> <li>Daily Budget (SGD)</li> <li>Activity Level</li> <li>Dietary Preferences</li> </ol> </li> </ol>

	j. Allergies 3. The user presses the “Update Profile” button and system displays a success message. 4. Data is stored in the database 5. Dashboard is updated accordingly, and data is also uploaded to WellOh.
Alternative Flows:	1. Invalid Input <ul style="list-style-type: none"> <li>a. If any inputs are invalid, an error message is displayed, and the user is prompted to re-enter the field.</li> </ul>
Exceptions:	If the system fails to update the profile due to technical issues, an error message is displayed, and the user is asked to try again later.
Includes:	
Special Requirements:	
Assumptions:	1. It is assumed that the user has valid profile information that they want updated.
Notes and Issues:	

## Functional Requirements

### 2. Profile

2.1. Users must be able to upload and update their personal information.

2.1.1. Users must be able to access this functionality through the dashboard interface.

2.1.2. Users must be able to input and update their profile details.

2.1.2.1. Users must be able to input their age.

2.1.2.2. Users must be able to input their height.

2.1.2.3. Users must be able to input their weight.

2.1.2.4. Users must be able to input their expected daily meal budget on average.

2.1.2.5. Users must be able to upload or update their profile picture.

2.1.2.5.1. System must support image file formats such as JPEG and PNG.

2.1.2.5.2. Users must be able to preview their profile picture before saving.

2.2. Users' profile data must be updated in real-time and reflected across all relevant features.

2.3.1. Users must see any changes made to their profile (e.g., weight, budget) immediately reflected in personalised recommendations, meal plans, or other related features.

2.3.2. Users must have consistent and up-to-date profile data across all instances of the platform.

## Manage Dietary Preferences

### Description and Priority

**Description:** It is of high priority for users to manage their dietary preferences so that the app can generate personalized meal plans for them

**Priority:** High

### Stimulus/Response Sequences

Use Case ID:	U0202		
Use Case Name:	Manage Dietary Preferences		
Created By:	Mahi Pandey	Last Updated By:	Mahi Pandey
Date Created:	02/09/2024	Date Last Updated:	02/09/2024

Actor:	Logged-in & registered user
Description:	This use case allows users to specify or update their dietary preferences within the EatWellthy app, which will be used to tailor meal recommendations.
Preconditions:	<ol style="list-style-type: none"> <li>1. The user must be logged into their EatWellthy account.</li> </ol>
Postconditions:	<ol style="list-style-type: none"> <li>1. The user's dietary preferences are successfully updated in the system.</li> </ol>
Priority:	High
Frequency of Use:	Occasional, as needed
Flow of Events:	<ol style="list-style-type: none"> <li>1. The user navigates to "Dietary Preferences" from their profile.</li> <li>2. The system displays the current dietary preferences.</li> <li>3. The user modifies their dietary preferences and allergies and clicks "Update".</li> <li>4. The system updates the preferences in the database and confirms the update.</li> </ol>
Alternative Flows:	<ol style="list-style-type: none"> <li>1. Invalid Preference <ul style="list-style-type: none"> <li>• If any preference selection is invalid, the system prompts the user to correct it before saving.</li> </ul> </li> </ol>
Exceptions:	If the system is unable to update dietary preferences due to technical issues, an error message is displayed, and the user is asked to try again later.
Includes:	<ol style="list-style-type: none"> <li>1. Change Confirmation</li> </ol>
Special Requirements:	Allergies must be separated by commas.
Assumptions:	<ol style="list-style-type: none"> <li>1. It is assumed that the user has dietary preferences that they wish to manage.</li> </ol>
Notes and Issues:	

### Functional Requirements

2.2.1. Users must be able to add and update their dietary preferences and allergies to ensure the diet suggestions are personalised.

2.2.2.1.1. The system shall restrict access to dietary preferences management to authenticated users only.

2.2.2.1.2. The system shall validate that the dietary preferences and allergies fields are formatted correctly before saving the information.

2.2.3. The system shall consider the user's dietary preferences and allergies when generating AI-powered dietary suggestions, excluding foods that conflict with these settings.

2.2.4. The system shall ensure that dietary preferences and allergy information are stored securely and are accessible only to the authenticated user.

## Using the Nutritional Information Finder

### Description and Priority

**Description:** Not necessary for users to use it unless they want to know the nutritional content of a specific food

**Priority:** Low

### Stimulus/Response Sequences

Use Case ID:	U0301		
Use Case Name:	Using the Nutritional Information Finder		
Created By:	Mehta Rishika	Last Updated By:	Mahi Pandey
Date Created:	29/8/2024	Date Last Updated:	9/11/2024

Actor:	Logged in User
Description:	This use case allows users who are logged into EatWellthy to be able to search for food items and their nutritional and caloric value. This can be done by passing the name of the item through the Nutritionix API da. The system will retrieve the desired information from the food database.
Preconditions:	<ol style="list-style-type: none"> <li>1. The user must be logged in.</li> <li>2. The user must provide valid input.</li> </ol>
Postconditions:	<ol style="list-style-type: none"> <li>1. Upon successful request, the system performs the following actions: <ul style="list-style-type: none"> <li>• The system retrieves the data from the food database</li> <li>• The system displays the caloric intake of the user based on the data it retrieves.</li> <li>• The user is equipped to access the app's features using their Google account credentials.</li> </ul> </li> </ol>
Priority:	Low
Frequency of Use:	As needed

Flow of Events:	<ol style="list-style-type: none"> <li>1. The user will log in to EatWellthy using their credentials.</li> <li>2. In the dashboard, the user navigates to “Nutrition Info Finder”</li> <li>3. On the “Nutrition Info Finder” page, the user enters the food that they wish to look up. <ul style="list-style-type: none"> <li>• If the user enters an invalid input, the system displays the message: "Input is invalid, please enter a correct value."</li> <li>• If the user inputs a valid input, the system fetches the necessary data from the database and gives the nutritional and caloric value of the given meal.</li> </ul> </li> <li>4. The system displays the nutritional and caloric value.</li> <li>5. The user can check the nutritional content of other food items.</li> </ol>
Alternative Flows:	<ol style="list-style-type: none"> <li>1. Valid Input But No Data Found</li> </ol> <p>If the meal data entered is valid but not found in the database, the system displays: "No data available for the entered meal. Please try another meal."</p>
Exceptions:	<ol style="list-style-type: none"> <li>1. Database Unavailable <ul style="list-style-type: none"> <li>• If the food database is down or unreachable, the system displays: "Service is currently unavailable. Please try again later."</li> </ul> </li> </ol>
Includes:	
Special Requirements:	
Assumptions:	<p>It is assumed that User has already registered and possesses an account to access the features of the website.</p> <p>The food database is up-to-date and contains accurate nutritional information.</p>
Notes and Issues:	

### Functional Requirements

#### 3. Dashboard

- 3.1. The system shall provide access to the "Nutrition Information Finder" feature in the dashboard once the user is logged in.
  - 3.1.1. The system shall ensure that only authenticated users can access the Nutrition Information Finder.
  - 3.1.2. The system shall validate that the input field is not left blank before the user submits a search.
  - 3.1.3. The system shall retrieve nutritional information from the Nutritionix API or an internal nutrition database upon receiving a valid food item input.
  - 3.1.4. The system shall fetch the following nutritional details for the food item:
    - Calories
    - Protein content
    - Fat content

- Carbohydrate content
- Serving size
- Additional vitamins and minerals (if available)

3.1.5. If the food item does not exist in the database, the system shall display a message: "No data available for the entered food item."

## Generate AI-Powered Dietary Suggestions

### Description and Priority

**Description:** It is of high priority to users who have a specific nutrition goal in mind and want to follow it strictly.

**Priority:** High

### Stimulus/Response Sequences

Use Case ID:	U0302		
Use Case Name:	Generate AI-Powered Dietary Suggestions		
Created By:	Liu Xiaotao	Last Updated By:	Mahi Pandey
Date Created:	2/9/2024	Date Last Updated:	09/11/2024

Actor:	Current User
Description:	This use case allows users to generate a personalized diet plan tailored to their individual health goals, dietary preferences, and nutritional needs. The system takes into account various user inputs, such as age, gender, weight, height, activity level, and specific dietary restrictions or preferences, to create a customized diet plan.
Preconditions:	<ol style="list-style-type: none"><li>1. The user must be logged into the EatWellthy application with a valid account.</li><li>2. The user must provide relevant health and dietary information within the application.</li><li>3. The system must have access to a comprehensive nutrition database to generate accurate diet plans.</li></ol>
Postconditions:	<ol style="list-style-type: none"><li>1. The user receives a personalized diet plan that aligns with their health goals and dietary preferences.</li><li>2. The generated diet plan is stored in the user's profile for easy access and future reference.</li><li>3. The user can follow the diet plan and track their progress within the EatWellthy application.</li></ol>
Priority:	High
Frequency of Use:	Daily

Flow of Events:	<ol style="list-style-type: none"> <li>1. The user logs into the EatWellthy application and navigates to the "Generate Diet Plan" section in the Dashboard.</li> <li>2. In Profile, the system prompts the user to provide or update the following information: <ul style="list-style-type: none"> <li>• Age, Gender, Weight, Height</li> <li>• Activity level (e.g., sedentary, lightly active, moderately active, very active)</li> <li>• Health goals (e.g., weight loss, muscle gain, maintenance)</li> <li>• Dietary preferences (e.g., vegetarian, vegan, low-carb, gluten-free)</li> <li>• Any food allergies or restrictions</li> </ul> </li> <li>3. The user enters the required information and confirms their input.</li> <li>4. The system validates the user's input for completeness and accuracy. If any required information is missing or incorrect, the system prompts the user to correct it.</li> <li>5. Upon successful validation, the system accesses its nutrition database and calculate the user's daily caloric needs and macronutrient distribution (e.g., protein, fats, carbohydrates).</li> <li>6. The system generates a personalized diet plan, including: <ul style="list-style-type: none"> <li>• Suggested daily calorie intake</li> <li>• Recommended meals and snacks</li> <li>• Portion sizes for each meal</li> <li>• Nutritional breakdown (e.g., calories, macronutrients, vitamins, minerals) for each meal</li> <li>• Meal timing recommendations (e.g., breakfast, lunch, dinner, snacks)</li> </ul> </li> <li>7. The system displays the personalized diet plan to user, allowing them to review and confirm the plan.</li> <li>8. The system stores the diet plan in the user's profile and makes it accessible from the dashboard for daily reference and meal logging.</li> </ol>
Alternative Flows:	<ol style="list-style-type: none"> <li>1. Plan Regeneration <ul style="list-style-type: none"> <li>• If the user is not satisfied with the initial diet plan, they can regenerate the plan with adjusted inputs (e.g., changing the health goal or activity level).</li> <li>• The system generates a new plan based on the updated inputs.</li> </ul> </li> </ol>
Exceptions:	<ol style="list-style-type: none"> <li>1. Database Access Issues: <ul style="list-style-type: none"> <li>• If the system cannot access the nutrition database (e.g., due to a network issue), it displays an error message and suggests the user try again later.</li> <li>• The user is informed that their current data will be saved and can be used to generate the diet plan once the issue is resolved.</li> </ul> </li> <li>2. Inconsistent or Conflicting Inputs:</li> </ol>



	<ul style="list-style-type: none"> <li>If the user's inputs are inconsistent or conflict with each other (e.g., selecting both high-carb and low-carb preferences), the system alerts the user and requests clarification.</li> <li>The user can modify their inputs to resolve the conflict and proceed with generating the diet plan.</li> </ul>
Includes:	
Special Requirements:	User should be able to copy the generated plan and paste it in their personal diet site
Assumptions:	1. The user has updated their "Profile" with the necessary details.
Notes and Issues:	

### Functional Requirements

3.2. The system must generate personalised meal plans using OpenAI API.

3.2.1. The system must generate meal plans based on user preferences, credentials, and dietary needs.

3.2.1.1. The system must consider user-provided information such as age, gender, weight, height, activity level, and any specific dietary preferences (e.g., vegetarian, halal).

3.2.1.2. The system must account for user-specific nutritional goals, health conditions, and any allergies or dietary restrictions.

3.2.1.3. The system must consider user budget constraints when suggesting meal plans, optimising for cost-effectiveness.

## Log Your Daily Meals

### Description and Priority

**Description:** It is of high priority since it allows users to track their daily nutritional content intake.

**Priority:** High

### Stimulus/Response Sequences

Use Case ID:	U0401		
Use Case Name:	Log Daily Meal		
Created By:	Liu Xiaotao	Last Updated By:	Mahi Pandey
Date Created:	1/9/2024	Date Last Updated:	09/11/2024

Actor:	Current User
Description:	This use case enables users to log their daily meals into the EatWellthy system. Users can input details such as meal type (breakfast, lunch, dinner, snacks), food items, portion sizes, and time of consumption. The system will store this information.

Preconditions:	<ol style="list-style-type: none"> <li>1. The user must be logged into the EatWellthy system with a valid account.</li> <li>2. The system must have access to the user's profile to store meal logs.</li> </ol>
Postconditions:	<ul style="list-style-type: none"> <li>• The meal information is successfully stored in the user's profile in the database.</li> <li>• The system updates the user's nutritional summary for the day, reflecting the logged meal.</li> <li>• Users can view, edit, or delete the meal entry at any time.</li> <li>• The system may provide feedback or suggestions based on the nutritional content of the logged meals.</li> </ul>
Priority:	High
Frequency of Use:	Daily
Flow of Events:	<ol style="list-style-type: none"> <li>1. The user accesses the EatWellthy application and navigates to the "Tracker" section.</li> <li>2. The system presents the user with a form to input meal details: <ul style="list-style-type: none"> <li>• Meal type (e.g., breakfast, lunch, dinner, snacks)</li> <li>• Food items consumed</li> <li>• Portion sizes</li> </ul> </li> <li>3. The user fills in the necessary information and clicks "Submit."</li> <li>4. The system validates the input: <ul style="list-style-type: none"> <li>• Ensures that all required fields are filled.</li> <li>• Verifies that the portion sizes are reasonable (e.g., not negative).</li> </ul> </li> <li>5. If the input is valid, the system stores the meal information in the user's profile.</li> <li>6. The system updates the user's daily nutritional summary with the new data.</li> <li>7. The system displays a confirmation message to the user, indicating that the meal has been successfully logged.</li> </ol>
Alternative Flows:	<ol style="list-style-type: none"> <li>1. Validation Errors: <ul style="list-style-type: none"> <li>• If any required fields are missing or contain invalid data, the system displays an error message.</li> <li>• The user corrects the input and resubmits the form.</li> <li>• The process continues once the input is valid.</li> </ul> </li> <li>2. Editing a Meal Log: <ul style="list-style-type: none"> <li>• The user can navigate to their meal history and select a previously logged meal.</li> <li>• The system displays the meal details, allowing the user to make changes.</li> </ul> </li> </ol>

	<ul style="list-style-type: none"> <li>The user submits the changes, and the system updates the meal information in the database and adjusts the nutritional summary accordingly.</li> </ul> <p>3. Deleting a Meal Log:</p> <ul style="list-style-type: none"> <li>The user selects a meal from their meal history that they wish to delete.</li> <li>The system prompts the user to confirm the deletion.</li> <li>Upon confirmation, the system removes the meal from the user's profile and adjusts the nutritional summary</li> </ul>
Exceptions:	<p>Network Failure:</p> <ul style="list-style-type: none"> <li>If there is a network issue, the system will prompt the user to retry or save the meal log locally until the network is restored.</li> </ul> <p>System Unavailability:</p> <ul style="list-style-type: none"> <li>If the system is down for maintenance or encounters an error, the user will receive a notification and will be unable to log meals until the issue is resolved.</li> </ul>
Includes:	Confirm message
Special Requirements:	
Assumptions:	
Notes and Issues:	

## Functional Requirements

### 4. Tracker

4.1. Users must be able to log each meal they consume.

4.1.1. Users must access the meal logging feature from the main dashboard or through a designated meal tracking section within the application.

4.1.2. The system must allow users to input specific details about each meal.

4.1.3. Users must be able to enter various components of each meal, including:

4.1.3.1. Type of meal (e.g., breakfast, lunch, dinner, snack).

4.1.3.2. Food items consumed.

4.1.3.3. Portion sizes of each food item.

4.1.3.4. Time of the meal.

4.1.4. The system must provide an interface with input fields or selection options for each of these details.

## Edit Logged Meal

### Description and Priority

**Description:** Users would not be using this feature unless they have keyed in inaccurate details for their meals

**Priority:** Low

### Stimulus/Response Sequences

Use Case ID:	U0402		
Use Case Name:	Edit Logged Meal		
Created By:	Liu Xiaotao	Last Updated By:	Mahi Pandey
Date Created:	1/9/2024	Date Last Updated:	9/11/2024

Actor:	Current User
Description:	This use case allows users to edit a previously logged meal in the EatWellthy system. Users can modify details such as meal type, food items, portion sizes, and time of consumption. The system updates the user's nutritional summary to reflect the changes.
Preconditions:	<ol style="list-style-type: none"> <li>1. The user must be logged into the EatWellthy system with a valid account.</li> <li>2. The user must have previously logged a meal that they wish to edit.</li> <li>3. The system must have access to the user's meal history.</li> </ol>
Postconditions:	<ol style="list-style-type: none"> <li>1. The updated meal information is successfully stored in the user's profile in the database.</li> <li>2. The system recalculates and updates the user's daily nutritional summary based on the changes.</li> <li>3. The user can view the updated meal entry and the adjusted nutritional information.</li> </ol>
Priority:	Low
Frequency of Use:	As needed
Flow of Events:	<ol style="list-style-type: none"> <li>1. The user accesses the EatWellthy application and navigates to their meal history.</li> <li>2. The system presents a list of previously logged meals, organised by date and time.</li> <li>3. The user selects the meal they wish to edit.</li> <li>4. The system displays the meal details, including: <ul style="list-style-type: none"> <li>• Meal type (e.g., breakfast, lunch, dinner, snacks)</li> <li>• Food items consumed</li> <li>• Portion sizes</li> <li>• Time of consumption</li> </ul> </li> <li>5. The user makes the necessary changes to the meal details.</li> <li>6. The user clicks "Save" to submit the changes.</li> <li>7. The system validates the new input: <ul style="list-style-type: none"> <li>• Ensures all required fields are filled.</li> </ul> </li> </ol>

	<ul style="list-style-type: none"> <li>Verifies that the portion sizes are reasonable (e.g., not negative).</li> </ul> <ol style="list-style-type: none"> <li>If the input is valid, the system updates the meal information in the user's profile.</li> <li>The system recalculates the user's daily nutritional summary to reflect the updated meal data.</li> <li>The system displays a confirmation message, indicating that the meal has been successfully updated.</li> </ol>
Alternative Flows:	<ol style="list-style-type: none"> <li>Validation Errors: <ul style="list-style-type: none"> <li>If any required fields are missing or contain invalid data, the system displays an error message.</li> <li>The user corrects the input and resubmits the form.</li> <li>The process continues once the input is valid.</li> </ul> </li> <li>Cancelling an Edit: <ul style="list-style-type: none"> <li>The user may choose to cancel the edit at any time before saving.</li> <li>The system discards any changes made and returns to the meal history view without altering the original meal log.</li> </ul> </li> </ol>
Exceptions:	<ol style="list-style-type: none"> <li>Network Failure: <ol style="list-style-type: none"> <li>If there is a network issue, the system will prompt the user to retry or save the meal log locally until the network is restored.</li> </ol> </li> <li>System Unavailability: <ul style="list-style-type: none"> <li>If the system is down for maintenance or encounters an error, the user will receive a notification and will be unable to log meals until the issue is resolved.</li> </ul> </li> </ol>
Includes:	
Special Requirements:	
Assumptions:	<ol style="list-style-type: none"> <li>Users have already previously logged a meal that they now wish to edit.</li> </ol>
Notes and Issues:	

### Functional Requirements

4.2. Users must be able to edit logged meals to correct or update meal entries.

4.2.1. Users must access the editing feature from within their meal log.

4.2.2. The system must allow users to modify any part of the meal entry, including:

4.2.2.1. Type of meal (breakfast, lunch, dinner, snack).

4.2.2.2. Food items consumed.

4.2.2.3. Portion sizes of each food item.

4.2.2.4. Time of the meal.

4.2.3 The system must provide an interface for editing meal entries.

4.2.3.1. The interface should display current meal details in editable fields.

4.2.3.2. Users should be able to save changes with a 'Save' button or discard changes with a 'Cancel' button.

## Delete Logged Meal

### Description and Priority

**Description:** Users would not use this feature unless they have logged a meal wrongly

**Priority:** Low

### Stimulus/Response Sequences

Use Case ID:	U0403		
Use Case Name:	Delete Logged Meal		
Created By:	LiuXiaotao	Last Updated By:	Mahi Pandey
Date Created:	1/9/2024	Date Last Updated:	9/11/2024

Actor:	Current User
Description:	This use case allows users to delete a previously logged meal from the EatWellthy system. Once deleted, the meal information is removed from the user's profile, and the system updates the user's daily nutritional summary to reflect the deletion.
Preconditions:	<ol style="list-style-type: none"> <li>The user must be logged into the EatWellthy system with a valid account.</li> <li>The user must have at least one previously logged meal in their history.</li> <li>The system must have access to the user's meal history.</li> </ol>
Postconditions:	<ol style="list-style-type: none"> <li>The selected meal is permanently removed from the user's profile in the database.</li> <li>The system recalculates and updates the user's daily nutritional summary based on the deletion.</li> <li>The user can view their updated meal history and nutritional information.</li> </ol>
Priority:	Low
Frequency of Use:	As needed
Flow of Events:	<ol style="list-style-type: none"> <li>The user accesses the EatWellthy application and navigates to their meal history.</li> <li>The system presents a list of previously logged meals, organized by date and time.</li> <li>The user selects the meal they wish to delete.</li> <li>The system displays the meal details, including: <ul style="list-style-type: none"> <li>Meal type (e.g., breakfast, lunch, dinner, snacks)</li> <li>Food items consumed</li> <li>Portion sizes</li> <li>Time of consumption</li> </ul> </li> <li>The user clicks the "Delete" button to remove the selected meal.</li> </ol>

	<ol style="list-style-type: none"> <li>3. The system validates the request and proceeds to delete the meal from the user's profile.</li> <li>4. The system recalculates the user's daily nutritional summary to reflect the removal of the meal data.</li> <li>5. The user's meal history is updated to exclude the deleted meal.</li> </ol>
Alternative Flows:	NIL
Exceptions:	<ol style="list-style-type: none"> <li>1. Network Failure: <ul style="list-style-type: none"> <li>• If there is a network issue, the system will prompt the user to retry or save the meal log locally until the network is restored.</li> </ul> </li> <li>2. System Unavailability: <ul style="list-style-type: none"> <li>• If the system is down for maintenance or encounters an error, the user will receive a notification and will be unable to log meals until the issue is resolved.</li> </ul> </li> </ol>
Includes:	
Special Requirements:	
Assumptions:	<ol style="list-style-type: none"> <li>1. User has a previously logged meal that they want to know delete</li> <li>2. The user is sure about deleting the current meal.</li> </ol>
Notes and Issues:	

### Functional Requirements

4.3. Users must be able to delete logged meals if they are no longer relevant or were logged in error.

4.3.1. Users must access the deletion option from within their meal log.

4.3.2. The system must require user confirmation before permanently deleting a meal entry to prevent accidental data loss.

## View and Search Meal History

### Description and Priority

**Description:** Important for users to view their meal history so that they know the nutritional content of it

**Priority:** High

### Stimulus/Response Sequences

Use Case ID:	U0404		
Use Case Name:	View and Search Meal History		
Created By:	Liu Xiaotao	Last Updated By:	Mahi Pandey

Date Created:	1/9/2024	Date Last Updated:	09/11/2024
---------------	----------	--------------------	------------

Actor:	Current User
Description:	This use case allows users to view their logged meal history within the EatWellthy system. Users can access details of past meals, including the type of meal, food items consumed, portion sizes, and nutritional information.
Preconditions:	<ol style="list-style-type: none"> <li>1. The user must be logged into the EatWellthy system with a valid account.</li> <li>2. The user must have logged at least one meal in the system.</li> <li>3. The system must have access to the user's meal history data.</li> </ol>
Postconditions:	<ol style="list-style-type: none"> <li>1. The user must be logged into the EatWellthy system with a valid account.</li> <li>2. The user must have logged at least one meal in the system.</li> <li>3. The system must have access to the user's meal history data.</li> </ol>
Priority:	High
Frequency of Use:	As needed
Flow of Events:	<ol style="list-style-type: none"> <li>1. The user accesses the EatWellthy application and navigates to the "History Search" section.</li> <li>2. The user inputs the date for which they want to see the meal history for.</li> <li>3. The system retrieves the user's logged meal data from the database.</li> <li>4. The system presents the meal history in a chronological list, displaying the following for each meal: <ul style="list-style-type: none"> <li>• Date and time of the meal</li> <li>• Meal type (e.g., breakfast, lunch, dinner, snacks)</li> <li>• Summary of food items consumed</li> <li>• Total calories and key nutritional information (e.g., protein, carbs, fats)</li> </ul> </li> <li>5. The user can exit the meal history section and return to the main dashboard or another section of the application.</li> </ol>
Alternative Flows:	<ol style="list-style-type: none"> <li>1. No Logged Meals: <ul style="list-style-type: none"> <li>• If the user has not logged any meals, the system displays a message indicating that no meal history is available.</li> <li>• The user is prompted to log their first meal.</li> </ul> </li> </ol>
Exceptions:	<ol style="list-style-type: none"> <li>1. Data Retrieval Failure:</li> </ol>



	<ul style="list-style-type: none"> <li>If the system fails to retrieve meal history due to a network or server issue, the user is notified with an error message.</li> <li>The system may offer an option to retry or advise the user to check back later.</li> </ul> <p>2. Incomplete Data:</p> <ul style="list-style-type: none"> <li>If any part of the meal history data is incomplete or corrupted, the system displays a warning and attempts to recover or display what is available.</li> <li>The user is informed about incomplete data and is given options to refresh or contact support.</li> </ul>
Includes:	
Special Requirements:	
Assumptions:	1. The user has logged meals for the day they wish to search up.
Notes and Issues:	Avoid the situation that the system loaded the data of another user

### Functional Requirements

4.4 The system shall provide access to the "Meal History" section from the tracker page.

## Add Customised Food Option

### 4. Description and Priority

**Description:** Not necessary for users to use this feature unless they have a certain food that they want to add, and it is not in the database

**Priority:** Medium

### 4. Stimulus/Response Sequences

Use Case ID:	U0405		
Use Case Name:	Add Customised Food Option		
Created By:	Liu Xiaotao	Last Updated By:	Mahi Pandey
Date Created:	1/9/2024	Date Last Updated:	09/11/2024

Actor:	Current User
Description:	This use case allows the user to add a custom food item to their profile. The customized food option can include specific nutritional information, portion size, and any additional dietary notes, which will then be considered in the user's personalized recommendations.
Preconditions:	<ol style="list-style-type: none"> <li>The user must be logged into their account.</li> <li>The user must have navigated to the section where they can manage or add food options.</li> </ol>

Postconditions:	<ol style="list-style-type: none"> <li>1. The custom food option is saved to the user's profile and can be used for future meal planning and recommendations.</li> <li>2. The system updates the user's nutritional profile to include the new customized option.</li> </ol>
Priority:	Medium
Frequency of Use:	Expected to be used infrequently, mainly when users want to add foods that are not in the existing database.
Flow of Events:	<ol style="list-style-type: none"> <li>1. The user selects the option to add a custom food item.</li> <li>2. The system displays a form where the user can input details about the food item (e.g., name, portion size, calories, macronutrients, etc.).</li> <li>3. The user completes the form and submits it.</li> <li>4. The system validates the information and saves the custom food item to the user's profile.</li> <li>5. The system confirms to the user that the custom food option has been successfully added.</li> </ol>
Alternative Flows:	<p><b>Alternative Flow 1: User Cancels Action</b></p> <ul style="list-style-type: none"> <li>• At any point during the flow, the user can cancel the action.</li> <li>• The system will discard any entered information and return the user to the previous screen.</li> </ul> <p><b>Alternative Flow 2: Duplicate Food Option</b></p> <ul style="list-style-type: none"> <li>• If the user tries to add a food item that already exists in the system or their profile, the system prompts the user to update the existing entry instead of adding a duplicate.</li> </ul>
Exceptions:	<ol style="list-style-type: none"> <li>1. Validation Error – If the user enters invalid data (e.g., negative values for calories), the system prompts the user to correct the information.</li> <li>2. Connection Error – If there is a network issue while submitting, the system notifies the user and prompts them to try again later.</li> </ol>
Includes:	1. User Authentication
Special Requirements:	Flexible data entry, secure handling and storage of user data
Assumptions:	<ol style="list-style-type: none"> <li>1. The user is aware of the nutritional information of the custom food option</li> </ol>
Notes and Issues:	

## Functional Requirements

4.5 Users should have the option to add custom food items to their meal log.

4.5.1. The system must allow users to create and save custom food items that are not available in the database.

4.5.1.1. Custom entries should include the name of the food, macronutrient breakdown, and caloric content.

## Generate User Report

### Description and Priority

**Description:** Not mandatory for users to generate their user report unless they have a specific goal/nutritional goal in mind.

**Priority:** Medium

### Stimulus/Response Sequences

Use Case ID:	U0501		
Use Case Name:	Generate User Reports		
Created By:	Zhang Yichi	Last Updated By:	Mahi Pandey
Date Created:	08/30/2024	Date Last Updated:	9/11/2024

Actor:	User
Description:	System generates an analysis report for the user based on their profile information.
Preconditions:	<ol style="list-style-type: none"><li>1. User must be logged in.</li><li>2. The system should have access to all relevant user data.</li><li>3. Report generation functionalities must be implemented.</li></ol>
Postconditions:	<ol style="list-style-type: none"><li>1. The system generates a comprehensive report based on user data.</li><li>2. The report accurately reflects the user's dietary and nutritional intake.</li></ol>
Priority:	Medium
Frequency of Use:	Monthly (Users typically generate reports on a monthly basis.)
Flow of Events:	<ol style="list-style-type: none"><li>1. User logs into the system.</li><li>2. User navigates to the “Analysis” section.</li><li>3. The system retrieves necessary data from the database.</li><li>4. The system generates the report and displays it on the screen or allows download.</li></ol>
Alternative Flows:	<ol style="list-style-type: none"><li>1. If the system fails to generate the report, an error message is displayed to the user.</li><li>2. If no data is available for the selected parameters, the system informs the user.</li></ol>
Exceptions:	<ol style="list-style-type: none"><li>1. Data inconsistencies or missing data may affect the accuracy of the analysis.</li></ol>

Includes:	
Special Requirements:	
Assumptions:	1. The user has provided accurate profile information
Notes and Issues:	

## Functional Requirements

### 5. Analysis

5.1. Users must be able to generate personalized reports that include BMI and nutritional goals.

5.1.1. Users must access the report generation feature through their dashboard or a dedicated reports section within the application.

5.2. The system must calculate the user's BMI based on the latest stored height and weight data.

5.2.1. The system must use the formula  $BMI = \frac{kg}{m^2}$  where weight is in kilograms and height is in meters squared.

5.2.2. The system must ensure that height and weight data are up to date and prompt the user to update if not.

5.3. The report must include detailed information on the user's nutritional goals.

5.3.1. Nutritional goals must include daily calorie intake, macronutrient distribution (proteins, fats, carbohydrates), and micronutrient intake recommendations.

5.3.2. The system must customize nutritional goals based on user-specific data such as age, gender, activity level, and dietary preferences.

5.4. The system must provide a visual representation of the user's progress towards these goals.

5.4.1. Visuals may include graphs, progress bars, or pie charts that display the user's status versus the goals.

5.4.2. The system must update these visuals in real-time as new data is logged or goals are adjusted.

## Navigate to Grocery Store Pages

### 4. Description and Priority

**Description:** Not necessary for users to check out the grocery store page unless they want to know the prices of certain food.

**Priority:** Medium

### Stimulus/Response Sequences

Use Case ID:	U0601		
Use Case Name:	Navigate to Grocery Store Pages		
Created By:	Mahi Pandey	Last Updated By:	Mahi Pandey

Date Created:	09/11/2024	Date Last Updated:	09/11/2024
---------------	------------	--------------------	------------

Actor:	Current User
Description:	This use case allows the user to navigate to various grocery store pages within the <i>EatWellthy</i> application. Users can view available stores and select one to see items, offers, or other relevant information provided by the grocery store.
Preconditions:	<ol style="list-style-type: none"> <li>1. The user must be logged into their account.</li> <li>2. The user must have a stable internet connection to load store information.</li> <li>3. The application must have preconfigured grocery store data, including store names, locations, and any available promotional details.</li> </ol>
Postconditions:	<ol style="list-style-type: none"> <li>1. The user is successfully navigated to the selected grocery store page.</li> </ol>
Priority:	Medium
Frequency of Use:	Expected to be used moderately, particularly when users want to explore store-specific items or promotions.
Flow of Events:	<ol style="list-style-type: none"> <li>1. The user selects the “Grocery Stores” option from the main menu or homepage.</li> <li>2. The system displays a list of available grocery stores, with basic information for each (e.g., name, location, and logo).</li> <li>3. The user selects a specific store to view.</li> <li>4. The system loads the page for the selected grocery store, displaying available items, promotions, and any relevant details.</li> <li>5. The user can browse the items, add items to their list, or navigate back to the main store selection page.</li> </ol>
Alternative Flows:	NIL
Exceptions:	<ol style="list-style-type: none"> <li>1. Connection Error – If there is a network issue while loading the store page, the system notifies the user and prompts them to retry.</li> </ol>
Includes:	
Special Requirements:	
Assumptions:	<ol style="list-style-type: none"> <li>1. The application has up-to-date information on grocery stores, including their products and promotions.</li> <li>2. Users may access this feature mainly for browsing and adding items to their shopping lists.</li> </ol>
Notes and Issues:	

## Functional Requirements

### 6. Grocery

6.1 The system must provide daily price information for the suggested food items.

6.1.1. The system must collect pricing data from at least four FairPrice, Prime, Cold Storage, and Sheng Siong

## Add Event in Calendar

### Description and Priority

**Description:** Users can add their events into the calendar

**Priority:** Medium

### Stimulus/Response Sequences

Use Case ID:	U0701		
Use Case Name:	Add Event in Calendar		
Created By:	Mahi Pandey	Last Updated By:	Mahi Pandey
Date Created:	09/11/2024	Date Last Updated:	09/11/2024

Actor:	Current User
Description:	This use case allows the user to add an event to the in-app calendar. The event could include reminders for meal planning, grocery shopping, or fitness activities that align with the user's dietary and wellness goals.
Preconditions:	<ol style="list-style-type: none"> <li>1. The user must be logged into their account.</li> <li>2. The application must have access to the in-app calendar feature or integrated calendar functionality.</li> </ol>
Postconditions:	<ol style="list-style-type: none"> <li>1. The new event is saved in the user's calendar.</li> <li>2. The system may send notifications or reminders based on the event's scheduled time.</li> </ol>
Priority:	Medium
Frequency of Use:	Expected to be used occasionally, particularly when users plan upcoming activities related to meal prep, grocery shopping, or health goals.
Flow of Events:	<ol style="list-style-type: none"> <li>1. The user selects the "Calendar" option from the side bar.</li> <li>2. The system displays the calendar interface, showing the current month and any existing events.</li> <li>3. The user selects a specific date and time to add a new event.</li> <li>4. The system displays a form where the user can input details such as event title, description, time, and frequency (one-time or recurring).</li> <li>5. The user completes the form and submits it.</li> <li>6. The system saves the event in the calendar and confirms the addition.</li> <li>7. The user can view the new event in the calendar interface.</li> </ol>
Alternative Flows:	<b>Alternative Flow 1:</b> User Cancels Event Addition

	<ul style="list-style-type: none"> <li>At any point, the user can cancel the event addition process, and the system will discard any entered information.</li> </ul> <p><b>Alternative Flow 2: Edit or Delete Event</b></p> <ul style="list-style-type: none"> <li>If the user wishes to change or remove an event, they can select the existing event in the calendar, and the system will provide options to edit or delete it.</li> </ul>
Exceptions:	<ol style="list-style-type: none"> <li>Invalid Date or Time – If the user selects an invalid date or time (e.g., a past date or overlapping event), the system prompts them to choose a valid date and time.</li> <li>Connection Error – If there is a network issue when saving the event, the system notifies the user and prompts them to try again.</li> </ol>
Includes:	
Special Requirements:	
Assumptions:	<ol style="list-style-type: none"> <li>The user will primarily use this feature to add events related to food, fitness, or wellness.</li> <li>The system's calendar functionality is up-to-date and accurately displays event information.</li> </ol>
Notes and Issues:	

## Functional Requirements

### 7. Calendar

7.1. Users must be able to add events to their calendar within the application.

7.1.1. Users must access the calendar feature from the main dashboard or navigation menu.

7.2. The system must allow users to input details for new calendar events.

7.2.1. Users must be able to enter essential event details such as:

7.2.1.1. Event title

7.2.1.2. Date and time of the event

## Update Event in Calendar

### Description and Priority

**Description:** Users can update and edit events in the calendar

**Priority:** Medium

### Stimulus/Response Sequences

Use Case ID:	U0702		
Use Case Name:	Update Event in Calendar		
Created By:	Mahi Pandey	Last Updated By:	Mahi Pandey
Date Created:	09/11/2024	Date Last Updated:	09/11/2024

Actor:	Current User
Description:	This use case allows the user to update an existing event in the in-app calendar. Users may modify event details such as the title, date, time, description, or frequency to ensure their calendar reflects their latest plans and commitments.
Preconditions:	<ol style="list-style-type: none"> <li>1. The user must be logged into their account.</li> <li>2. There must be an existing event in the calendar that the user can update.</li> </ol>
Postconditions:	<ol style="list-style-type: none"> <li>1. The event is successfully updated in the calendar with the new information.</li> </ol>
Priority:	Medium
Frequency of Use:	Expected to be used occasionally, particularly when users need to adjust event details due to changes in their schedule.
Flow of Events:	<ol style="list-style-type: none"> <li>1. The user navigates to the calendar section and selects an existing event they wish to update.</li> <li>2. The system displays the event's current details.</li> <li>3. The user selects the "Edit" option and updates the desired details (e.g., title, date, time, description).</li> <li>4. The user submits the updated information.</li> <li>5. The system saves the updated event details and confirms the change.</li> <li>6. The updated event now appears in the calendar with the new details.</li> </ol>
Alternative Flows:	<p><b>Alternative Flow 1: User Cancels Update</b></p> <ul style="list-style-type: none"> <li>• The user can cancel the update process at any time, and the system discards any changes.</li> </ul>
Exceptions:	<ol style="list-style-type: none"> <li>1. Invalid Update Data – If the user enters invalid information (e.g., end time before start time), the system prompts the user to correct the input.</li> <li>2. Connection Error – If a network issue occurs while updating the event, the system notifies the user and prompts them to try again later.</li> </ol>
Includes:	
Special Requirements:	
Assumptions:	<ol style="list-style-type: none"> <li>1. The user has a basic understanding of event management and calendar functions.</li> <li>2. The system accurately tracks and displays updated event information.</li> </ol>
Notes and Issues:	

## Functional Requirements

7.2. The system must allow the user to navigate to the calendar section and select an existing event to update.



7.1.1. The user must be able to find and select the event they wish to edit, and the system should display the event's current details (e.g., title, date, time, description).

7.3. The system must allow the user to modify the event's details, such as the title, date, time, description, and frequency.

7.2.1. The user should be able to edit the selected event's details through an intuitive interface, ensuring the fields are editable as required.

7.3. The system must save the updated event details and display a confirmation message to the user.

7.3.1. After submitting the updates, the system should update the event in the calendar with the new information and notify the user of the successful update.

7.4. The system must validate the updated event details and prompt the user to correct invalid inputs (e.g., conflicting times, missing fields).

7.4.1. If the user enters invalid data (e.g., an end time before the start time), the system should display an error message, highlight the issue, and prompt the user to correct it before proceeding.

## Add Event to Google Calendar

### Description and Priority

**Description:** Users will be able to add events in their Google Calendar.

**Priority:** Medium

### Stimulus/Response Sequences

Use Case ID:	U0703		
Use Case Name:	Add Event to Google Calendar		
Created By:	Liu Xiaotao	Last Updated By:	Mahi Pandey
Date Created:	1/9/2024	Date Last Updated:	9/11/2024

Actor:	Current User
Description:	This use case allows the user to add an event created within the <i>EatWellthy</i> app to their Google Calendar. This feature provides convenience for users who want to manage their events and reminders in a unified calendar system.
Preconditions:	<ol style="list-style-type: none"> <li>1. The user must be logged into their <i>EatWellthy</i> account.</li> <li>2. The user must have a Google account and grant permission for <i>EatWellthy</i> to access and modify their Google Calendar.</li> </ol>
Postconditions:	<ol style="list-style-type: none"> <li>1. The selected event is successfully added to the user's Google Calendar with the specified details (e.g., title, date, time, and any additional notes).</li> <li>2. The system confirms that the event has been synced with Google Calendar.</li> </ol>
Priority:	Medium
Frequency of Use:	Expected to be used occasionally, especially when users prefer to consolidate their events in one calendar.

Flow of Events:	<ol style="list-style-type: none"> <li>1. The user selects an event in the <i>EatWellthy</i> app they wish to add to Google Calendar.</li> <li>2. The system prompts the user to connect their Google account if they have not already done so.</li> <li>3. The user provides authorization for <i>EatWellthy</i> to access their Google Calendar.</li> <li>4. The system displays a confirmation prompt showing the event details to be synced.</li> <li>5. The user confirms the event addition.</li> <li>6. The system adds the event to Google Calendar and displays a success message.</li> </ol>
Alternative Flows:	<p><b>Alternative Flow 1: Authorization Denied</b></p> <ul style="list-style-type: none"> <li>• If the user denies authorization, the system displays a message explaining that Google Calendar integration requires permission and returns the user to the event page.</li> </ul> <p><b>Alternative Flow 2: User Cancels Event Addition</b></p> <ul style="list-style-type: none"> <li>• The user can cancel the process at any time before the final confirmation, and the system discards any unsynced changes.</li> </ul>
Exceptions:	<ol style="list-style-type: none"> <li>1. Authorization Error – If the system encounters an issue during authorization (e.g., invalid credentials), it prompts the user to retry or check their Google account settings.</li> <li>2. Connection Error – If there is a network issue while syncing, the system notifies the user and prompts them to try again later.</li> </ol>
Includes:	<ol style="list-style-type: none"> <li>1. User Authentication</li> <li>2. Google API Integration</li> </ol>
Special Requirements:	The feature should comply with Google's API usage policies and limits.
Assumptions:	<ol style="list-style-type: none"> <li>1. The user has a functional Google account and understands the authorization process.</li> <li>2. The Google Calendar API is available and operational during the event addition process.</li> </ol>
Notes and Issues:	

## Functional Requirements

7.1 The system must ensure the user is logged into their EatWellthy account and authorized to access and modify their Google Calendar.

7.1.1. The user must be prompted to authenticate and grant permission for EatWellthy to access their Google account before adding events to Google Calendar.

7.2 The user must be able to select an event from the EatWellthy app and confirm the details to be synced with Google Calendar.

7.2.1. After selecting the event, the system should display a confirmation prompt with the event details (e.g., title, date, time) for the user to review before finalizing the sync.

7.3. The system must successfully add the selected event to the user's Google Calendar.

7.3.1. Once confirmed, the system should create the event in the user's Google Calendar with the correct details and notify the user of the successful sync.

7.4. The system must handle errors such as authorization issues or network connectivity problems and provide appropriate error messages and retry options.

7.4.1. In case of authorization denial, connection failure, or invalid credentials, the system should notify the user and provide options to retry or troubleshoot the issue.

## Sync Diet Plan with Calendar

### Description and Priority

**Description:** Allows users to add their diet suggestions into the calendar.

**Priority:** Medium

### Stimulus/Response Sequences

Use Case ID:	U0704		
Use Case Name:	Sync Diet Plan with Calendar		
Created By:	Liu Xiaotao	Last Updated By:	Mahi Pandey
Date Created:	1/9/2024	Date Last Updated:	9/11/2024

Actor:	Current User
Description:	This use case allows users to sync their diet plan from EatWellthy with their Google Calendar. By doing this, users can have meal reminders, diet plan events, and nutritional goals automatically appear in their Google Calendar.
Preconditions:	<ol style="list-style-type: none"> <li>1. The user must be logged into the EatWellthy system with a valid account.</li> <li>2. The user must have a diet plan created within the EatWellthy application.</li> <li>3. The user must have an active Google account and must be signed into Google services.</li> <li>4. The user must authorise the EatWellthy app to access and manage their Google Calendar.</li> </ol>
Postconditions:	<ol style="list-style-type: none"> <li>1. The user's diet plan is successfully synced with their Calendar.</li> <li>2. The system automatically creates calendar events for each meal or diet-related activity, based on the user's diet plan.</li> <li>3. Users are prompted to add the diet plan to their Google Calendar as well.</li> <li>4. Users receive notifications on their devices as per the scheduled times in the Google Calendar.</li> </ol>

Priority:	Medium
Frequency of Use:	Frequently (every time the plan is generated)
Flow of Events:	<ol style="list-style-type: none"> <li>1. The user logs into the EatWellthy application and navigates to the "Diet Plan" section in the Dashboard.</li> <li>2. The user selects the option to sync their diet plan with Calendar.</li> <li>3. The diet plan is added to the EatWellthy calendar.</li> <li>4. The system prompts the user to add their diet plan to their Google Calendar.</li> <li>5. The system prompts the user to sign in to their Google account, if they haven't already.</li> <li>6. The system requests the necessary permissions to access and manage the user's Google Calendar.</li> <li>7. The user grants permission, allowing the EatWellthy app to access their Google Calendar.</li> <li>8. The system retrieves the user's diet plan and begins syncing the data with the Google Calendar.</li> <li>9. For each meal or diet-related event in plan, system: <ul style="list-style-type: none"> <li>• Creates a corresponding event in Google Calendar.</li> <li>• Sets the event time and date based on the meal schedule.</li> <li>• Includes details in the event description, such as the meal type, food items, and nutritional goals.</li> </ul> </li> <li>8. The system confirms the successful sync and notifies the user that their diet plan is now available in Google Calendar.</li> <li>9. The user can now view, edit, or receive reminders for their diet plan directly from Google Calendar.</li> </ol>
Alternative Flows:	<ol style="list-style-type: none"> <li>1. User does not wish to Sync to Google Calendar <ul style="list-style-type: none"> <li>• If the user denies the need to add to Google Calendar, the diet plan is stored in the Eatwellthy Calendar.</li> </ul> </li> <li>2. Authorization Failure: <ul style="list-style-type: none"> <li>• If the user denies the authorization request, the system displays a message explaining that syncing cannot proceed without the necessary permissions.</li> <li>• The user is given the option to retry the authorization process.</li> </ul> </li> <li>3. Sync Failure: <ul style="list-style-type: none"> <li>• If the system encounters an issue while syncing (e.g., network issues, Google API errors), the system displays an error message.</li> <li>• The user is given options to retry the sync or to troubleshoot the issue.</li> </ul> </li> <li>4. Partial Sync: <ul style="list-style-type: none"> <li>• If only part of the diet plan syncs successfully (e.g., some events were not created), the system notifies the user of the partial success.</li> <li>• The system provides details on which events were not synced and offers options to retry syncing only those events.</li> </ul> </li> </ol>

Exceptions:	<ol style="list-style-type: none"> <li>1. Network Issues: <ol style="list-style-type: none"> <li>1. If there is a network connectivity problem during the sync, the system alerts the user and pauses the syncing process until the connection is restored.</li> <li>2. The user can manually retry the sync after resolving any network issues.</li> </ol> </li> <li>• Google Calendar Quota Limits: <ol style="list-style-type: none"> <li>1. If the user's Google Calendar has reached its event creation limit, the system notifies the user and cancels the sync process.</li> <li>2. The user is advised to clear up space in their Google Calendar or contact Google support for further assistance.</li> </ol> </li> <li>3. Calendar Event Conflicts: <ul style="list-style-type: none"> <li>• If there are conflicting events in the Google Calendar during the sync (e.g., overlapping times with existing events), the system warns the user.</li> <li>• The user can choose to skip conflicting events or manually resolve the conflicts in Google Calendar.</li> </ul> </li> </ol>
Includes:	<ol style="list-style-type: none"> <li>1. User Authentication</li> <li>2. Google API Integration</li> </ol>
Special Requirements:	The system should access and interact with Google calendar successfully.
Assumptions:	<ol style="list-style-type: none"> <li>1. We assume that the user has a Google account, and the system can access it.</li> </ol>
Notes and Issues:	

## Functional Requirements

7.1 The system must allow users to log in to EatWellthy and authenticate their Google account.

7.1.1. Users must be logged into their EatWellthy account and authorized to access their Google account by granting the necessary permissions to the system for syncing the diet plan.

7.2 The system must sync the user's diet plan with their Google Calendar.

7.2.1. Once the user grants authorization, the system should retrieve the diet plan and automatically create corresponding events for each meal or diet-related activity in Google Calendar.

7.3 The system must create calendar events with accurate meal times, types, and nutritional goals.

7.3.1. For each meal or diet-related activity in the user's plan, the system should create an event in Google Calendar, including event details such as the meal type, food items, and nutritional goals.

7.4 The system must handle errors and provide options to retry syncing in case of failures.

7.4.1. If there are issues such as network connectivity problems, authorization failure, or Google Calendar quota limits, the system should notify the user and provide options to retry the sync or troubleshoot the issue.

## Display FAQ Page

### Description and Priority

**Description:** Users will not be using this page unless they have queries about how the app works

**Priority:** Low

### Stimulus/Response Sequences

Use Case ID:	U0801		
Use Case Name:	Display FAQ Page		
Created By:	Mahi Pandey	Last Updated By:	Mahi Pandey
Date Created:	9/11/2024	Date Last Updated:	9/11/2024

Actor:	Current User
Description:	This use case allows the user to view the Frequently Asked Questions (FAQ) page. The FAQ page provides users with answers to common questions about <i>EatWellthy</i> , helping them navigate the application, understand its features, and troubleshoot basic issues.
Preconditions:	<ol style="list-style-type: none"> <li>1. The user must have access to the <i>EatWellthy</i> application.</li> </ol>
Postconditions:	<ol style="list-style-type: none"> <li>1. The FAQ page is displayed, and the user can view and scroll through a list of commonly asked questions and answers.</li> </ol>
Priority:	Low
Frequency of Use:	Expected to be used occasionally, primarily when users need guidance or have questions about using <i>EatWellthy</i> .
Flow of Events:	<ol style="list-style-type: none"> <li>1. The user selects the “FAQ” option from the side bar.</li> <li>2. The system loads the FAQ page, displaying a list of categorized questions and answers.</li> <li>3. The user scrolls through the list to find answers.</li> <li>4. If needed, the user can navigate back to the previous page or other parts of the application.</li> </ol>
Alternative Flows:	<p><b>Alternative Flow 1:</b> FAQ Page Not Available</p> <ul style="list-style-type: none"> <li>• If the FAQ page fails to load (e.g., due to network issues), the system displays an error message and provides options to retry or contact support.</li> </ul>
Exceptions:	<ol style="list-style-type: none"> <li>1. Network Error – If there is a network issue, the FAQ page may not load. The system notifies the user and offers a retry option.</li> <li>2. Page Not Found – If the FAQ page URL is incorrect or the page is unavailable, the system shows an error message and suggests navigating to the help center.</li> </ol>

Includes:	
Special Requirements:	
Assumptions:	1. The FAQ content is kept up to date with the latest information about <i>EatWellthy</i> .
Notes and Issues:	

### Functional Requirements

- 8.1 The system must provide a visible and accessible "FAQ" option in the sidebar.
- 8.2 The system must display a categorized list of questions and answers on the FAQ page.
- 8.3 The system must provide an option for users to navigate back to the previous page or other sections of the application.

## Search for Near-by Grocery Stores

### Description and Priority

**Description:** Allows users to know what stores are located near them, making it more convenient for them

**Priority:** High

### Stimulus/Response Sequences

Use Case ID:	U0901		
Use Case Name:	Search for Near-by Grocery Stores		
Created By:	Mahi Pandey	Last Updated By:	Mahi Pandey
Date Created:	9/11/2024	Date Last Updated:	9/11/2024

Actor:	Current User
Description:	This use case enables users to search for nearby grocery stores based on their current location or a location they key into the search bar. The feature aims to help users find accessible stores for convenient shopping, displaying relevant store details such as location, hours, and potential offers.
Preconditions:	<ol style="list-style-type: none"> <li>1. The user must be logged into their <i>EatWellthy</i> account.</li> <li>2. The application must have permission to access the user's location.</li> <li>3. The device must have internet connectivity for retrieving store data.</li> </ol>
Postconditions:	<ol style="list-style-type: none"> <li>1. The user is presented with a list of grocery stores within a specified radius of their current location.</li> </ol>
Priority:	High
Frequency of Use:	Expected to be used frequently, especially by users looking for nearby grocery options or planning their shopping trips.

Flow of Events:	<ul style="list-style-type: none"> <li>The user navigates to the search feature and selects the option to find nearby grocery stores.</li> <li>The system requests permission to access the user's current location (if not already granted).</li> <li>The system retrieves the user's location and searches for grocery stores within a specified radius.</li> <li>The system displays a list of nearby grocery stores, including basic information such as store name, address, and distance.</li> <li>The user can select a store from the list to view more details, such as store hours, directions, or available promotions.</li> </ul>
Alternative Flows:	<p><b>Alternative Flow 1: User Denies Location Access</b></p> <ul style="list-style-type: none"> <li>If the user denies access to their location, the system displays a message explaining that location access is required to use this feature and returns the user to the main search page.</li> </ul> <p><b>Alternative Flow 2: Search by Zip Code or Address</b></p> <ul style="list-style-type: none"> <li>If the user does not want to share their current location, they can manually enter a zip code or address to search for nearby stores.</li> <li>The system retrieves stores based on the provided location and displays results as in the primary flow.</li> </ul>
Exceptions:	<ol style="list-style-type: none"> <li>Location Error – If the system cannot retrieve the user's location, it notifies the user and prompts them to check device settings or enter a location manually.</li> <li>No Stores Found – If there are no stores within the search radius, the system displays a message suggesting the user expand the radius or try another location.</li> </ol>
Includes:	<ol style="list-style-type: none"> <li>Google Maps API Integration</li> </ol>
Special Requirements:	
Assumptions:	<ol style="list-style-type: none"> <li>Users may rely on this feature to quickly locate grocery stores near them.</li> </ol>
Notes and Issues:	

## Functional Requirements

9.1. The system must suggest the nearest supermarket branches where the user can buy the suggested food items.

9.1.1. The system must calculate the distance between the user's location and the nearest supermarket branches of FairPrice, Prime, Cold Storage, Sheng Siong and Giant.



## Ask Nutritional Questions

### Description and Priority

**Description:** It is important for users who want to meet their dietary needs and goals.

**Priority:** High

### Stimulus/Response Sequences

Use Case ID:	U1001		
Use Case Name:	Ask Nutritional Questions		
Created By:	Liu Xiaotao	Last Updated By:	Mahi Pandey
Date Created:	02/09/2024	Date Last Updated:	09/11/2024

Actor:	Current User
Description:	This use case allows users to interact with the Welloh ChatBot to ask various nutritional questions. The Welloh ChatBot is powered by GPT-based technology, fine-tuned to provide accurate, relevant, and personalised nutritional information. Users can inquire about dietary advice, nutritional content of foods, meal suggestions, and other related topics. The chatbot is designed to deliver responses quickly and efficiently.
Preconditions:	<ol style="list-style-type: none"> <li>1. The user must be logged into the EatWellthy application.</li> <li>2. The Welloh ChatBot must be connected to the internet and have access to the latest nutritional databases and GPT-based model.</li> <li>3. The ChatBot's underlying GPT model must be optimised for performance, ensuring quick and relevant responses.</li> </ol>
Postconditions:	<ol style="list-style-type: none"> <li>1. The user receives an accurate and relevant response to their nutritional question</li> <li>2. The system logs the interaction for future reference and potential improvement of the ChatBot's responses.</li> </ol>
Priority:	High
Frequency of Use:	As needed
Flow of Events:	<ol style="list-style-type: none"> <li>1. The user opens the EatWellthy application and navigates to the "Welloh ChatBot" feature.</li> <li>2. The user types a nutritional question into the ChatBot interface. Examples of questions include: <ul style="list-style-type: none"> <li>• "What are the health benefits of eating avocados?"</li> <li>• "How many calories are in a medium-sized apple?"</li> <li>• "Can you suggest a high-protein breakfast?"</li> </ul> </li> <li>3. The system receives the user's input and processes the question through the Welloh ChatBot's GPT-based model.</li> </ol>

	<ol style="list-style-type: none"> <li>4. The ChatBot analyzes the question, retrieving relevant information from its nutritional database and applying natural language processing to generate a clear and concise response.</li> <li>5. The system displays the response to the user within 2 seconds of receiving the question. The response may include: <ul style="list-style-type: none"> <li>• Nutritional information (e.g., calorie content, macronutrient breakdown)</li> <li>• Health benefits and potential risks of certain foods</li> <li>• Personalized meal suggestions based on the user's profile and dietary preferences</li> </ul> </li> <li>6. The user reviews the response and can either ask follow-up questions, save the response, or incorporate suggestions into their meal plan.</li> <li>7. The system logs the interaction for future reference, potentially improving future responses through machine learning.</li> </ol>
Alternative Flows:	<ol style="list-style-type: none"> <li>1. Unavailable Information: <ul style="list-style-type: none"> <li>• If the ChatBot cannot find relevant data for the user's query, it informs the user and offers to search for related information or suggest alternative questions.</li> <li>• The user can either modify their question or accept the ChatBot's suggestions.</li> </ul> </li> <li>2. Technical Issues: <ul style="list-style-type: none"> <li>• If the ChatBot experiences technical issues (e.g., connectivity problems, server downtime), it informs the user and suggests trying again later.</li> <li>• The system may offer offline alternatives, such as browsing the FAQ section or accessing preloaded nutritional guides.</li> </ul> </li> </ol>
Exceptions:	<ol style="list-style-type: none"> <li>1. Excessive Response Time: <ol style="list-style-type: none"> <li>1. If the ChatBot takes longer than 2 seconds to generate a response, the system displays a message indicating that the response is delayed.</li> <li>2. The user is given the option to wait or rephrase the question for a potentially faster response.</li> </ol> </li> </ol>
Includes:	
Special Requirements:	
Assumptions:	<ol style="list-style-type: none"> <li>1. It assumed the system can have a stable GPT service.</li> <li>2. The user has a valid nutritional question.</li> </ol>
Notes and Issues:	May need some prompt techniques for an accurate response.

## Functional Requirements

10.1 The Welloh chatbot must provide real-time responses to user queries related to diet and nutrition.

10.1.1 The chatbot must use the ChatGPT API to generate responses that are accurate, informative, and relevant to the user's needs.

10.1.2 The chatbot must provide personalised responses by considering the user's profile information, such as age, weight, height, activity level, dietary preferences, and health conditions.

## 4.25 Query Personalized Nutritional Information via Chatbot

### Description and Priority

**Description:** Not necessary for users to use this feature unless they have queries about the nutritional content of certain foods

**Priority:** Medium

### Stimulus/Response Sequences

Use Case ID:	U1002		
Use Case Name:	Query Nutritional Information via Chatbot		
Created By:	Liu Xiaotao	Last Updated By:	Mahi Pandey
Date Created:	2/9/2024	Date Last Updated:	9/11/2024

Actor:	Current User
Description:	This use case allows users to query specific nutritional information via the Welloh ChatBot. The ChatBot, powered by a GPT-based model optimised for enhanced performance, provides detailed nutritional data on various foods, ingredients, and meals. Users can inquire about calorie counts, macronutrient breakdowns, vitamin content, and more, receiving tailored responses based on their dietary preferences and needs.
Preconditions:	<ol style="list-style-type: none"> <li>1. The user must be logged into the EatWellthy application.</li> <li>2. The Welloh ChatBot must be operational, with access to the latest nutritional databases and the GPT-based model.</li> <li>3. The ChatBot's system must be up-to-date, ensuring that the nutritional information provided is accurate and reliable.</li> </ol>
Postconditions:	<ol style="list-style-type: none"> <li>1. The user receives precise nutritional information based on their query.</li> <li>2. The system logs the interaction for future reference, helping to improve the accuracy and relevance of future responses.</li> <li>3. The user has the option to save the queried information or add related items to their meal plan.</li> </ol>
Priority:	Medium
Frequency of Use:	As needed

Flow of Events:	<ol style="list-style-type: none"> <li>1. The user opens the EatWellthy application and navigates to the "Welloh ChatBot" feature.</li> <li>2. The user types a specific query regarding nutritional information into the ChatBot interface. Examples of queries include: <ul style="list-style-type: none"> <li>• "How can I incorporate more carbohydrates in my current plan?"</li> <li>• "Is it good for me to eat beef?"</li> </ul> </li> <li>3. The system receives the user's query and processes it through the Welloh ChatBot's GPT-based model.</li> <li>4. The ChatBot analyzes the query, retrieves relevant nutritional data from its database, and generates a response.</li> <li>5. The system displays the nutritional information to the user within 2 seconds of receiving the query. The response may include: <ul style="list-style-type: none"> <li>• Calorie content</li> <li>• Macronutrient breakdown (proteins, fats, carbohydrates)</li> <li>• Micronutrient details (vitamins, minerals)</li> <li>• Additional health-related information or advice</li> </ul> </li> <li>6. The user reviews the information and can choose to ask follow-up questions for more details.</li> <li>7. The system logs the interaction for future analysis and improvement of the ChatBot's responses.</li> </ol>
Alternative Flows:	<ol style="list-style-type: none"> <li>1. Unavailable Information: <ul style="list-style-type: none"> <li>• If the ChatBot cannot find specific nutritional data for the user's query, it informs the user and offers alternatives (e.g., similar foods or items).</li> <li>• The user can choose to refine their query or select from the suggested alternatives.</li> </ul> </li> </ol>
Exceptions:	<ol style="list-style-type: none"> <li>1. Excessive Response Time: <ul style="list-style-type: none"> <li>• If the ChatBot takes longer than 2 seconds to generate a response, the system displays a message indicating that the response is delayed.</li> <li>• The user is given the option to wait or rephrase the question for a potentially faster response.</li> </ul> </li> </ol>
Includes:	
Special Requirements:	
Assumptions:	<ol style="list-style-type: none"> <li>1. It assumed the system can have a stable GPT service.</li> <li>2. It is assumed that the user has updated their profile information accurately.</li> <li>3. It is assumed user has a valid question</li> </ol>
Notes and Issues:	May need some prompt techniques for an accurate response.

## Functional Requirements

10.2. The system must display detailed nutritional information in response to user queries.

10.2.1. Nutritional responses must include calorie content, macronutrient breakdown, and micronutrient details.

10.2.2. If requested, the system should provide additional health-related advice or information.

## 5. Other Nonfunctional Requirements

### Performance Requirements

**Objective:** These performance standards are set to ensure the application delivers a fast and responsive experience, meeting user expectations and playing a crucial role in enhancing user retention and satisfaction.

- i. The landing page should load within 3 seconds, as measured by tools such as Google PageSpeed Insights, to provide a fast and seamless user experience, minimizing the risk of high bounce rates.
- ii. All pages, unless explicitly exempted, must load within 3 seconds to ensure a consistent and smooth user experience throughout the application.
- iii. API responses, especially for retrieving meal data, must occur within 5 seconds, even during peak traffic, to ensure timely and efficient meal planning.
- iv. Input validation for email and password on the registration page should be completed within 1 second, delivering immediate feedback to the user for a responsive and interactive experience.
- v. All calculations must be based on a reliable algorithm and process within 5 seconds, including daily caloric needs calculations within 3 seconds and nutrition calculations from the Nutrition Database.
- vi. All user actions, such as logging in and generating diet plans, must be processed and returned within 5 seconds, with standard database queries (e.g., fetching user profile) returning results within 2 seconds, and all UI components responding within 1 second.
- vii. User authentication should be completed within 3 seconds of submitting credentials, with OAuth-based logins (e.g., Google) completed within 2.5 seconds.
- viii. The page load time for the dashboard after login should not exceed 3 seconds for 95% of users, and transitions between UI states should not exceed 2 seconds.

### Safety Requirements

**Objective:** These safety protocols are essential to safeguard users from potential harm and to ensure the product complies with the ethical standards for providing health-related recommendations.

- i. The application must adhere to the Health Sciences Authority (HSA) regulations relevant to digital health applications, ensuring the safety of users.
- ii. Dietary advice within the app will comply with the Health Promotion Board's guidelines to ensure alignment with Singapore's health standards.
- iii. Any health-related recommendations within the app must be accompanied by a disclaimer, and users must acknowledge their understanding of the associated risks.

### Security Requirements

**Objective:** The objective is to ensure robust user privacy, data protection, and compliance with regulations like PDPA and GDPR. By implementing secure authentication, data encryption, and strict access controls, the system aims to maintain user trust and safeguard sensitive information while ensuring transparency and accountability.

- i. The system must implement secure user authentication via email and password, encrypted with at least SHA-256 encryption, or through verified Google account integration.
- ii. Strong password policies must be enforced, requiring at least 8 characters with a combination of uppercase, lowercase, numbers, and special characters.
- iii. All data exchanged between the client and server must be encrypted using TLS 1.2 or higher to prevent unauthorized access during transmission.
- iv. Sensitive data stored in the system's database must be encrypted to ensure security even in the event of a database breach.
- v. Administrative actions, such as data deletion and role assignment, must be logged, with access to these logs restricted to authorized personnel only.
- vi. All user activities that involve sensitive operations must be logged, including timestamps and user IDs, with logs retained for at least 3 months.
- vii. The system must comply with relevant data protection regulations, such as GDPR or PDPA, ensuring user data is handled according to legal requirements.
- viii. The system must provide a clear privacy policy explaining how user data is collected, stored, and protected, with explicit user consent required for data processing.
- ix. The system must undergo regular security audits and penetration testing at least twice a year to identify and address any vulnerabilities.
- x. An incident response plan must be in place, outlining actions to take in the event of a data breach, including user notification, data recovery, and mitigation strategies.
- xi. Administrative access to user data shall be logged and audited on a monthly basis to prevent unauthorized access and maintain user privacy.
- xii. User passwords are stored using industry-standard hashing and salting techniques, as per OWASP recommendations.
- xiii. Authentication via OAuth 2.0 will not retain passwords or email addresses within the system, ensuring adherence to the Personal Data Protection Act (PDPA) of Singapore.
- xiv. A complete removal of user data from the database is executed within 48 hours after a user has confirmed account deletion, in line with PDPA (Right to be forgotten).
- xv. User accounts can only be deleted by the account owner or an administrator, with the process being audited and records kept for 5 years.

## Software Quality Attributes

**Objective:** These attributes have been chosen to ensure the product is reliable, user-friendly, and maintainable, which will contribute to the overall satisfaction and retention of the users.

### 1. Availability:

- a. The system should be available 99.9% of the time, excluding scheduled maintenance windows, to ensure consistent access for users.
- b. Auto-scaling mechanisms must dynamically allocate resources to maintain an average page load time of under 3 seconds, even as user numbers increase.

### 2. Reliability:

- a. The application should have an error rate of less than 0.1% to ensure reliable performance.
- b. The system should respond within 2 seconds for 95% of requests.

- c. The system should reach a mean time between failures over 7 days.
- 3. **Usability:**
  - a. The design must provide full information for users with mobile and desktop devices.
  - b. The system must achieve a System Usability Scale (SUS) score of 80 or higher in user testing.
  - c. The colour scheme must pass a minimum contrast ratio of 4.5:1 for text and background colours to support readability.
  - d. The user interface must allow a new user to complete core tasks (e.g., logging food, viewing diet suggestions) within 3 minutes without external assistance.
  - e. The dashboard interface must allow users to customize the display (e.g., choosing which statistics to view) within 3 clicks or taps.
  - f. All forms must include inline validation, providing real-time feedback within 0.5 seconds after the user completes a field.
- 4. **Maintainability:**
  - a. The codebase must have consistent naming conventions, modular design, and thorough documentation to ensure that the system is easy to maintain and extend.
  - b. The system must support zero-downtime deployments, allowing updates to be rolled out without disrupting user access or functionality.
  - c. Regular maintenance windows must be scheduled, and users must be notified in advance. These windows should be used for tasks like database optimization, security patching, and performance tuning.

## Business Rules

**Objective:** These business rules ensure secure, role-based access and enforce strict data privacy controls, fostering compliance with privacy regulations and enhancing both user trust and the overall security of the system.

1. **Role-Based Access Control:** Users are assigned specific roles (e.g., regular user, administrator) that define the level of access and functionality available to them within the system.
2. **Sensitive Data Access Restrictions:** Access to sensitive user information is granted based on user roles. Regular users can only view their own data, while administrators have access to all user data, but only for support and system maintenance purposes.
3. **Permission Granularity:** Each user role comes with a defined set of permissions, limiting the actions users can take within the application, such as viewing, editing, or deleting data.
4. **Audit Trails for Sensitive Actions:** Any actions performed on sensitive user data, especially those by administrators, must be logged and auditable to ensure accountability and security.
5. **Compliance with Privacy Regulations:** Business rules must ensure that user data is handled in accordance with relevant privacy laws (e.g., GDPR, PDPA), restricting access and actions based on user roles to enhance compliance and user trust.



## 6. Other Requirements

**Objective:** These additional requirements ensure the application is not only legally compliant but also scalable, secure, and maintainable. By adhering to international standards for privacy, accessibility, and reusability, the system will meet both user and business needs while supporting future growth and technological evolution.

### i. Database Requirements:

- The database must support horizontal scaling to handle an increase in concurrent user activity, with the capability to scale seamlessly to handle 50,000 concurrent users.
- The database must be designed for fault tolerance, with backup and recovery systems in place to prevent data loss in case of system failure.

### ii. Legal and Compliance Requirements:

- The system must ensure full compliance with local and international privacy laws, including Singapore's PDPA, Europe's GDPR, and other applicable data protection regulations, ensuring that data is processed and stored securely.
- The application must include user consent mechanisms for collecting and processing personal data, with explicit opt-in options for both local and international users.
- The system must support the ability to erase or anonymize user data upon request, in compliance with the "right to be forgotten" clause under GDPR.

### iii. Internationalization and Localization Requirements:

- The application must be designed to support multiple languages and regional formats, allowing easy localization of content and adapting date/time, currency, and numeric formats based on user preferences or location.
- The system must handle time zone differences effectively, with features such as user-specific time zone preferences for event scheduling and notifications.

### iv. Reusability and Modular Architecture:

- The system must be designed with modular components, enabling the reuse of key features, such as user authentication and payment processing, across different applications within the ecosystem.
- A component library or microservices architecture should be considered to allow for easy upgrades and integration of future features with minimal disruption to existing functionality.

### v. Scalability Requirements:

- The system must be scalable to support a projected 100% user growth annually, ensuring system performance remains optimal even as the user base grows.
- The infrastructure must include auto-scaling to manage traffic surges during peak usage times (e.g., holidays or sales events), maintaining system performance with no significant downtime.

### vi. Supportability and Maintenance:

- The codebase must follow industry best practices for maintainability, including clean, readable code, and comprehensive comments, ensuring long-term support and updates.
- The system must support seamless patching and versioning of software components, ensuring that any updates do not affect current users' data or functionality.

**vii. Security Requirements:**

- The system must employ multi-factor authentication (MFA) for users with administrative access, ensuring secure access to sensitive data and system settings.
- The system must include robust protection against common cyber threats such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).
- All sensitive user data must be stored in encrypted formats, both at rest and in transit, and access to this data must be limited to authorized personnel based on role-based access control.

## 7. API Testing

The smooth and efficient operation of our web application depends significantly on the functionality and stability of our **RESTful API**, which serves as the backbone of our backend infrastructure. As the central point of communication between client and server, this API handles requests and manages responses that enable the user experience to remain responsive and error-free. Given its importance, it is essential that we rigorously test each API endpoint, ensuring that it meets functional requirements and operates without interruption. Through thorough testing, we aim to maintain high reliability and enhance user satisfaction with a robust backend framework.

To achieve this, we've developed a comprehensive testing strategy that combines both unit testing and integration testing. These two approaches allow us to assess each endpoint from multiple perspectives, achieving both precision and holistic validation. At the unit level, testing is applied to individual API endpoints, helping us validate each component in isolation. This step ensures that every endpoint, whether it's responsible for handling user authentication, data retrieval, or processing transactions, functions exactly as intended. Unit testing helps us catch specific errors within individual API endpoints early, ensuring they perform correctly in a controlled environment before they interact with the broader system.

Following unit testing, we implement integration testing to evaluate how these endpoints interact within a larger context. Integration testing verifies that endpoints work effectively together and behave as expected in real-world scenarios, simulating specific use cases that users are likely to encounter. This approach ensures that API endpoints can handle complex operations smoothly, such as simultaneous data requests or multi-step transactions, and allows us to assess the reliability and functionality of the API in scenarios that require multiple endpoints to coordinate seamlessly.

Our testing strategy relies heavily on **Postman's API testing tools**, which offer a versatile and user-friendly environment for developing, automating, and executing tests. Using Postman, we simulate various API calls and observe how the backend handles different scenarios, ranging from standard data requests to edge cases and error conditions. Postman enables us to quickly identify performance bottlenecks, pinpoint failures, and validate that responses meet predefined expectations, even under high loads. The platform's automated testing capabilities allow us to create repeatable test suites, which we can run at any stage of development, ensuring that every endpoint remains stable and efficient as the application evolves.

By combining the precision of unit testing with the broader insights gained from integration testing, our testing approach offers multiple layers of validation, helping to ensure that the API not only functions correctly in isolated instances but also performs reliably within complex user interactions. Through these meticulous testing methods, we are able to deliver a web application that provides a consistently smooth and dependable experience for users, supported by an API infrastructure that has been tested and optimized for excellence at every level.

### 1. Account Management

- **Register User**  
Endpoint: POST /users  
Payload: { "name": "Test User", "email": "[testuser@example.com](mailto:testuser@example.com)", "password": "testPassword123" }  
Expected Response: Confirmation message and token.
- **Login User**  
Endpoint: POST /users/auth  
Payload: { "email": "[testuser@example.com](mailto:testuser@example.com)", "password": "testPassword123" }  
Expected Response: Authentication token if credentials are valid.
- **Forgot Password**  
Endpoint: POST /users/forgot-password  
Payload: { "email": "[testuser@example.com](mailto:testuser@example.com)" }  
Expected Response: Success message if the email exists; error if the email is not found.
- **Verify Code**  
Endpoint: POST /users/verify-code  
Payload: { "email": "[testuser@example.com](mailto:testuser@example.com)", "code": "123456" } (replace with actual code)  
Expected Response: Success message if the code is correct.

## 2. Profile Management

- **Create or Update Profile**  
Endpoint: POST /api/profile  
Payload: { "age": 25, "gender": "female", "height": 165, "weight": 65, "targetWeight": 60, "dailyBudget": 2000, "dietaryPreferences": "vegetarian", "allergies": ["peanuts", "gluten"], "activityLevel": "active", "dietPlan": "weight loss", "profileIcon": "bear" }  
Expected Response: Profile created or updated confirmation message.
- **Retrieve Profile**  
Endpoint: GET /api/profile/me  
Headers: Authorization: Bearer <token>  
Expected Response: Profile details for the authenticated user.

## 3. Meal Tracking

- **Log Meal**  
Endpoint: POST /log\_meal  
Payload: { "owner": "user\_id", "meal\_type": "lunch", "food\_taken": "salad", "portion": 200, "time": "2023-10-01T12:00:00Z" }  
Expected Response: Confirmation of meal logged with meal details.
- **Update Meal**  
Endpoint: PUT /meal\_update/:id  
Payload: { "owner": "user\_id", "meal\_type": "dinner", "food\_taken": "grilled chicken", "portion": 150, "time": "2023-10-01T19:00:00Z" }  
Expected Response: Confirmation of meal update with updated details.
- **Delete Meal**  
Endpoint: DELETE /meal\_delete/:id  
Expected Response: Success message indicating meal deletion.

## 4. Nutrition Data Management

- **Query Food Nutrition**  
Endpoint: POST /query\_food  
Payload: { "owner": "user\_id" }  
Expected Response: List of foods and nutritional data for the user.
- **Add Custom Food Item**  
Endpoint: POST /add  
Payload: { "name": "Broccoli", "owner": "user\_id", "energy": 50, "fat": 0.5, "sugar": 2.5, "fiber": 3.0, "protein": 3.5, "sodium": 20, "vitamin\_c": 80, "calcium": 47, "iron": 1.3 }  
Expected Response: Success message and the new food item details.

## 5. Calendar

- **Create Google Calendar Event**  
Endpoint: POST /eventRoute  
Payload: { "title": "Lunch Meeting", "start": "2023-10-01T12:00:00Z", "end": "2023-10-01T13:00:00Z", "describe": "Lunch with team" }  
Expected Response: Confirmation of event creation with Google Calendar event ID if integrated.

## 6. Location

- **Search Nearby Grocery Stores**  
Endpoint: POST /location  
Payload: { "lat": "1.3521", "lng": "103.8198" }  
Expected Response: List of nearby grocery stores filtered by criteria.

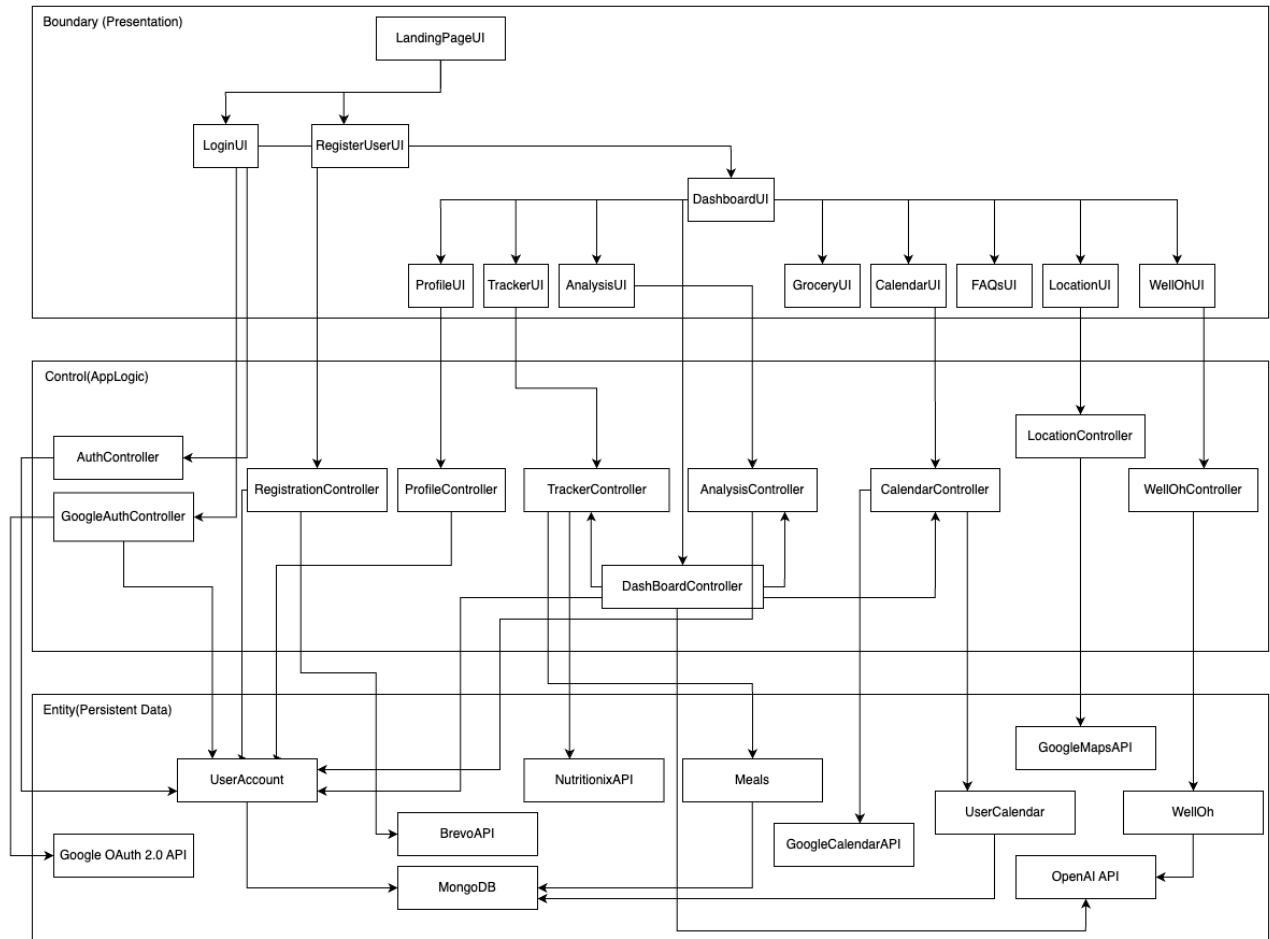
## 7. Welloh ChatBot

- **Ask Dietary Questions**  
Endpoint: POST /wellohAI/chat  
Payload: { "userMessage": "What are the nutritional benefits of kale?" }  
Expected Response: AI-generated response with nutritional information.

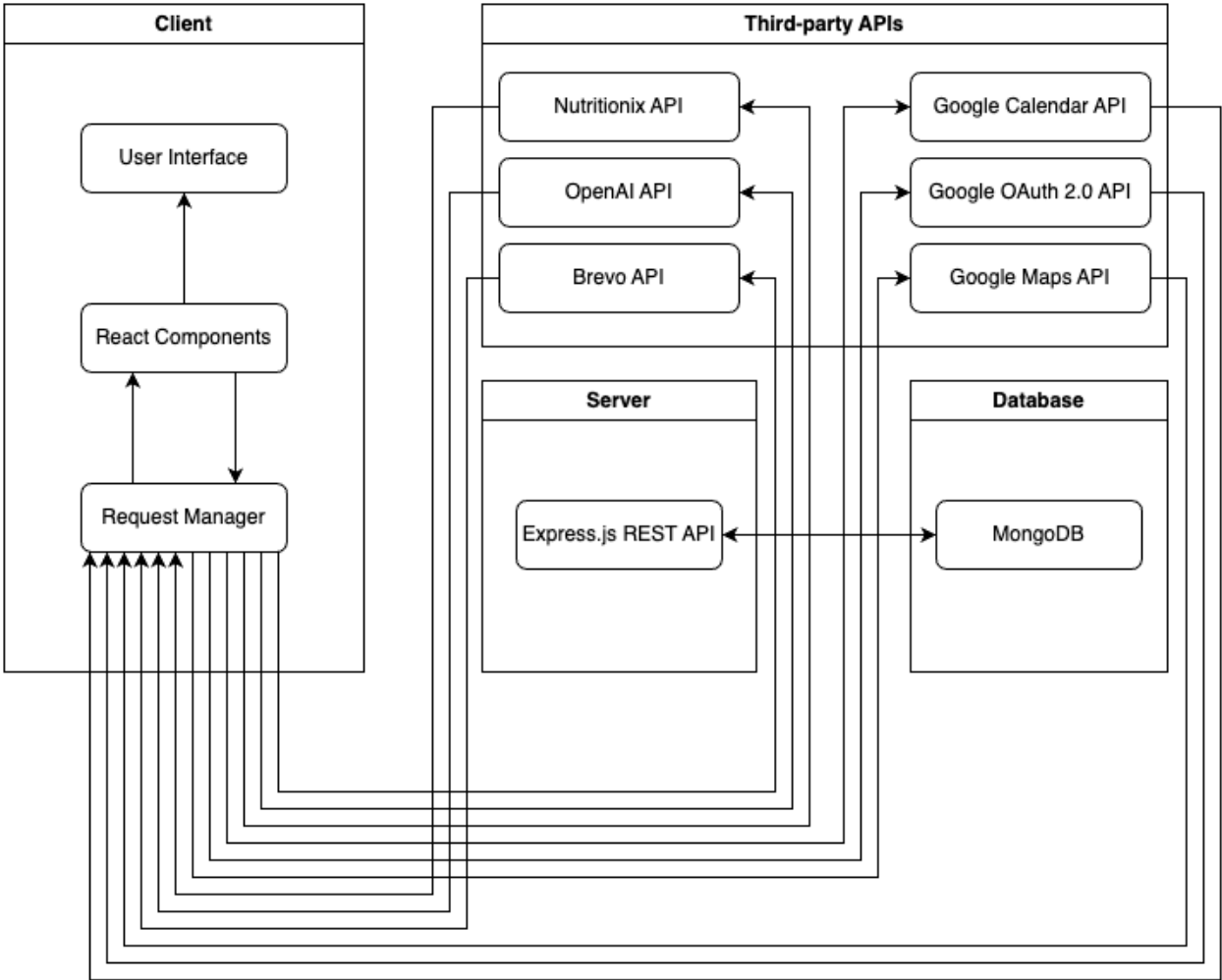
## 8. Appendix A: Data Dictionary

Terms	Definition
EatWellthy	The name of the web application that helps users record and track their diet, manage nutrition, and receive personalized consumption and purchase suggestions based on their input and dietary preferences.
User	The authorized end user who has a registered account and is using services provided by the EatWellthy web application.
UserID	Unique integer identifier for each user within the system (primary key in database).
Username	The name chosen by the user for logging into EatWellthy and display.
Email	The email address associated with the user's account, used for account recovery and notifications.
Password	An encrypted string of characters used by the user to access their account.
Profile	Collection of user-specific data including age, weight, height, budget, allergies and dietary preferences, linked to UserID.
User Registration	Creation of a UserID and Profile containing email, password, username. The information of the user will be stored in the database.
User Login	Web accessibility to users who have already registered can login through username or email and password, or through Google Account linking.
Dashboard	A centralized interface where users can view summaries of their nutritional intake, progress tracking, diet suggestions, and other personalized insights.
Custom Food Entry	A feature that allows users to manually input food items, including their nutritional information, into the user's database only.
Dietary Preferences	User-selected options like halal, vegetarian, and allergies, which the system filters when providing diet suggestions and tracking nutrition.
Nutritional values	Includes information such as calories, proteins, fats, carbohydrates, vitamins, and minerals, relevant to the food items entered by the user or predefined in the Nutrition Database.
Nutrition Database	The centralized database that stores Nutritional Values for all food items, both predefined and user-added entries.
Progress Tracker	A tool that visualizes the user's progress over time, including weight changes, adherence to nutritional goals, and other health metrics.
Diet Suggestions	Personalized recommendations provided by the application based on the user's dietary preferences, nutritional goals, and health conditions.
Calendar	A feature within EatWellthy that allows users to schedule and organize their meals.
Google Calendar Integration	A feature allowing users to sync their diet suggestions and meal plans with Google Calendar.
Welloh	An AI-driven chat interface powered by the ChatGPT API, trained in healthy diet-related data, that allows users to ask dietary questions and receive personalized responses.
Store Locator	A feature that uses Google Maps services to determine the user's location and suggests nearby supermarkets.

## 9. Appendix B: Analysis Models

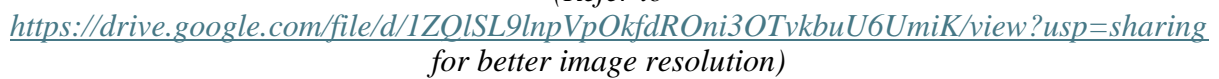


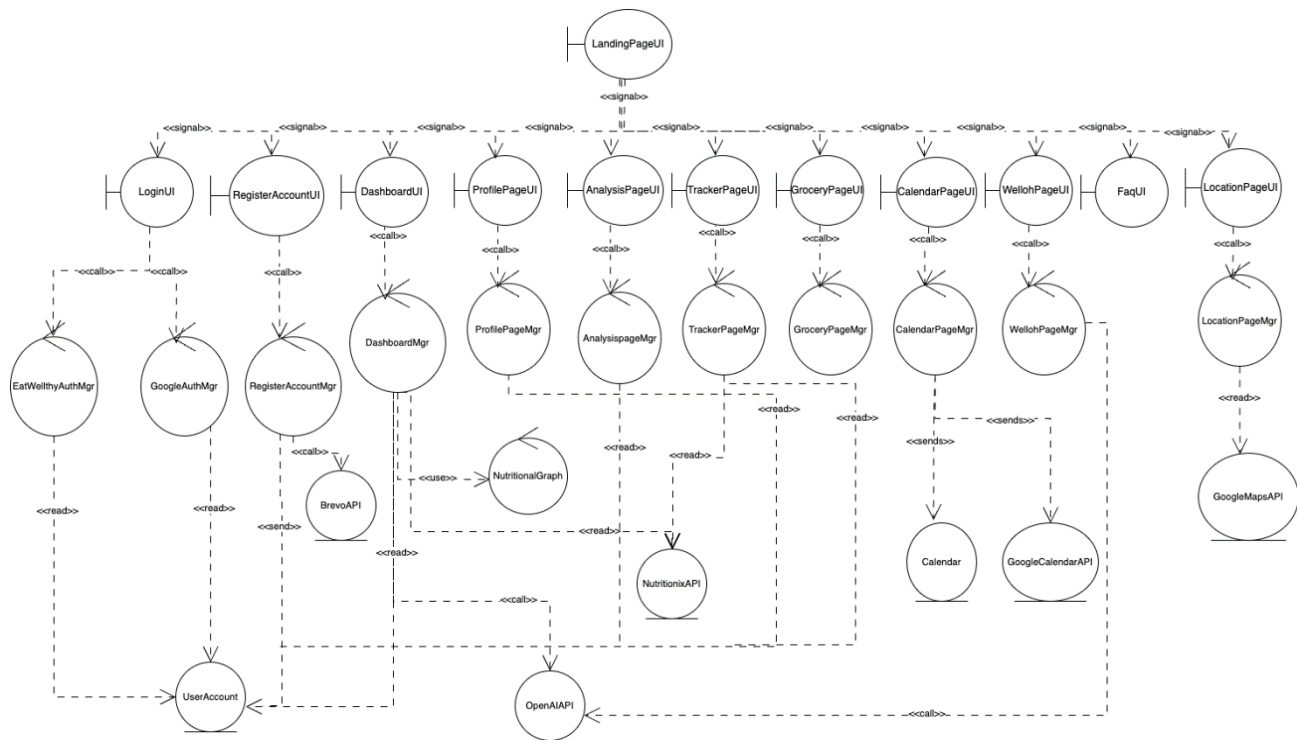
*System Architecture Diagram*



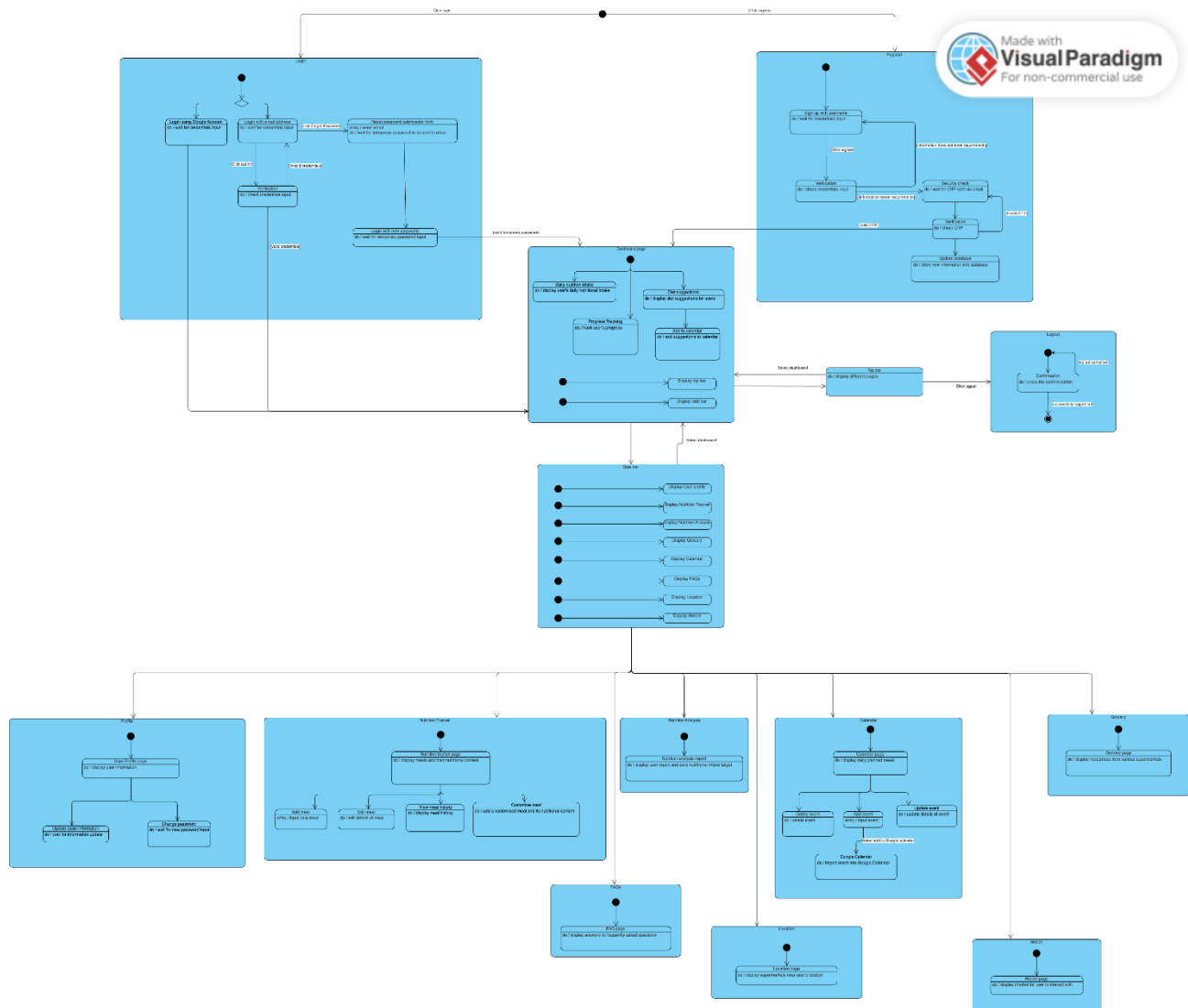
Simplified Architecture Diagram



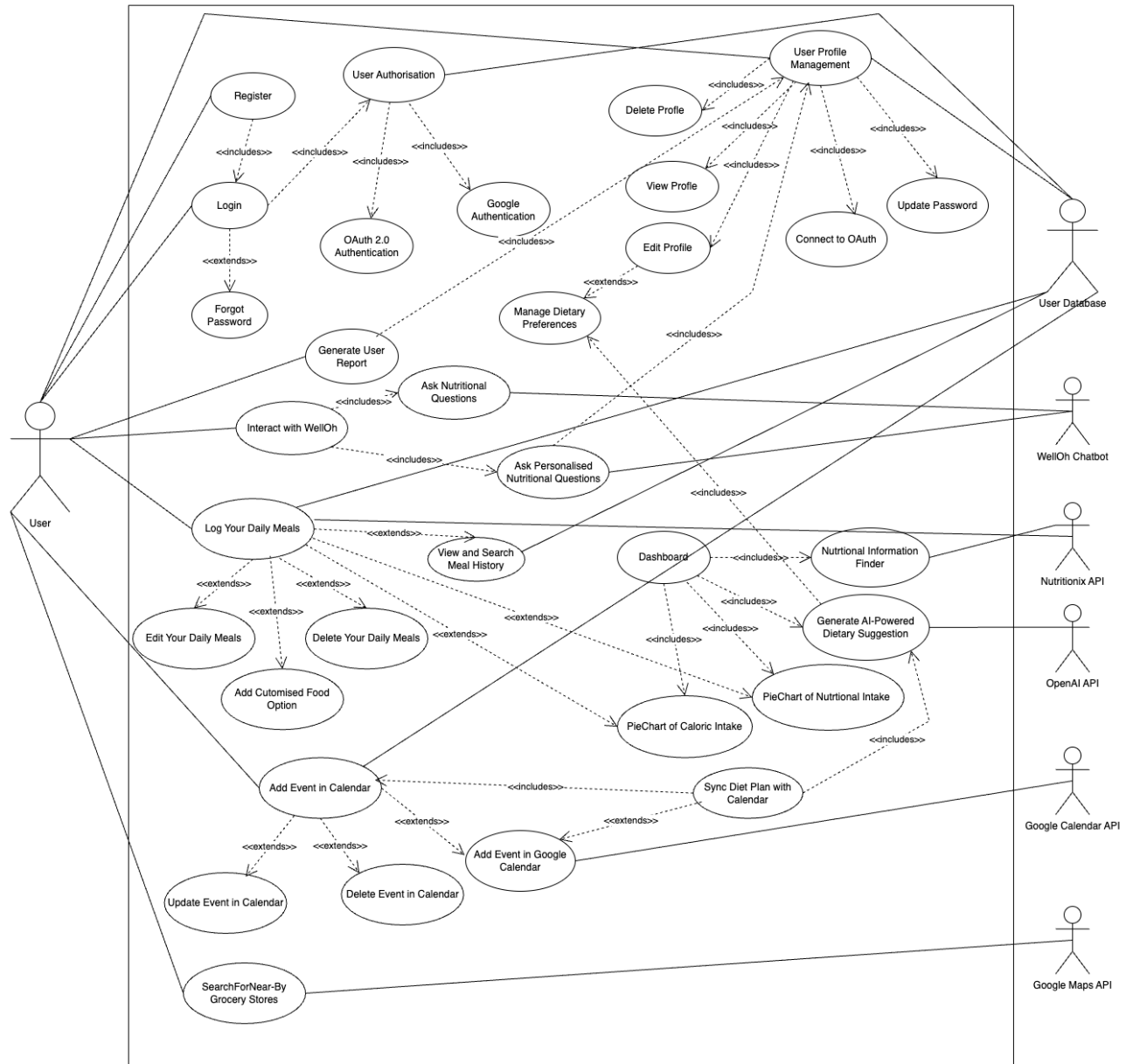




Key Boundary Class Diagram



Dialog Map



Use Case Diagram

## Appendix C: To Be Determined List

There are no TBD items for this version of SRS.

Source: [http://www.frontiernet.net/~kwiegers/process\\_assets/srs\\_template.doc](http://www.frontiernet.net/~kwiegers/process_assets/srs_template.doc)