Software Requirements Specification

for

VI Fitness

Version 1.0 approved

Prepared by

LIM JING JIE,

NG YUEN HERNG,

QIAN JIANHENG OSCAR,

TAN WEI YIN,

JIANG JINYI,

PRAKRITIPONG PHUVAJAKRT

Team VI, Team 04, SCS4, Nanyang Technological University

2023-11-05

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Ng Yuen Herng	24 Oct 2023	Initial write up for Introduction and Overall Description	1.0
Jiang Jinyi	25 Oct 2023	Initial write up for Other Nonfunctional Requirements	1.1
Tan Wei Yin	27 Oct 2023	Initial write up for System Features	1.2
Prakritipong Phuvajakrt	27 Oct 2023	Initial write up for External Interface Requirements	1.2
Jiang Jinyi	1 Nov 2023	Update System Features	1.2
Qian JianHeng Oscar	1 Nov 2023	API Unit Testing	1.2
Lim Jing Jie	3 Nov 2023	Update System Features	1.3
Qian JianHeng Oscar	3 Nov 2023	API Integrated Testing	1.3
Tan Wei Yin	10 Nov 2023	Update System Features	1.4
Lim Jing Jie	11 Nov 2023	Initial write up for Black Box Testing Test Cases	1.5
Lim Jing Jie	11 Nov 2023	Initial write up for 3.1 User Interfaces	1.6
Lim Jing Jie	12 Nov 2023	Initial write up for Software Engineering Practices	1.7
Lim Jing Jie	12 Nov 2023	Write up for 7.2 White Box Testing	1.8
Lim Jing Jie	13 Nov 2023	Completed 3.1 User Interface	1.9

1. Introduction

1.1. Purpose

The purpose of this Software Requirement Specification (SRS) is to serve as a documentation of a web application, *VIFitness*. To facilitate the development process among the relevant stakeholders, this SRS document describes the requirements specifications for *VIFitness* in detail, including but not limited to web application features, functional and non-functional requirements, interface design and limitations.

1.2. Document Conventions

This Software Requirements Specification (SRS) document follows standard typographical conventions to ensure clarity and consistency. Text in bold represents GUI elements, user inputs, or other emphasis. Text in italics denotes newly introduced terms, important notes, or areas requiring special attention. Source code, API endpoints, and database queries are presented in monospace font. Priorities for high-level requirements are inherited by detailed requirements unless explicitly stated otherwise.

Software Requirement Specification Standard: IEEE 830-1998. Priorities of higher level requirements are inherited by detailed level requirements.

Font:	Times New Roman
Heading1:	Size 18, Bold
Heading2:	Size 14, Bold
Heading3:	Size 12, Bold
Heading4:	Size 11, Bold
Content:	Size 11
Spacing in content:	1 line spaced

Further conventions on special terms used throughout this document are described in Appendix A: Data Dictionary.

1.3. Intended Audience and Reading Suggestions

This Software Specification Requirement (SRS) document is designed to cater to the needs of various stakeholders, including the development team, project managers, marketing team, testers, documentation writers, and end-users of *VIFitness*.

While it is advisable for all stakeholders to go through the entire document to gain a holistic understanding of the project, we have identified key sections that are particularly relevant to each group:

Developers: Should focus on the overall system architecture and detailed requirement specifications to understand the technical implementation.

Project Managers: Can review the entire document, paying special attention to the product scope, objectives, and high-level requirements to ensure alignment with project goals.

Marketing Staff: Should concentrate on the product scope and user interface sections to align marketing strategies with product capabilities.

Users: Are encouraged to read the overview and user interaction sections to familiarize themselves with the system's functionalities and user interface.

Testers: Should focus on the detailed requirements and exception handling to devise comprehensive test plans.

Documentation Writers: Can utilize the entire document to create user manuals and help documentation.

External Developers: Can review the overall system architecture and detailed documentation for *VIFitness*' API endpoints

1.4. Product Scope

In response to the rapid evolution of the digital era and Singapore's commitment to becoming a tech-driven Smart Nation, our team introduces *VIFitness*, a holistic web application designed to help users monitor their physical activity, diet, and overall health. The primary goal of *VIFitness* is to empower users with in-depth insights into their fitness and nutritional habits, enabling them to make informed decisions toward their health and fitness objectives. *VIFitness* focuses on three core functionalities, calorie tracking, workout planning and *Strava* Integration.

For calorie Tracking, *VIFitness* integrates the *Nutritionix* API, offering users access to an extensive food database for precise calorie tracking and nutritional information. Users can log and monitor their daily meals, specifying food items and quantities. The application calculates and displays the total calorie intake, empowering users to make informed dietary choices.

For workout Planning: *VIFitness* enables users to plan their workout sessions effectively. Users can create personalized workout plans, specifying exercise types and dates. A calendar view simplifies scheduling and visualizing workout routines. Leveraging the *Nutritionix* API integration, the application dynamically calculates and displays calories burned for each exercise based on user-specific parameters, such as age, weight, and gender.

For *Strava* Integration, *VIFitness* seamlessly integrates with the *Strava* API, allowing users to import their activities from *Strava*. The application calculates and displays the calories burned for each activity, providing users with a holistic view of their daily caloric intake and expenditure.

On top of its core functionalities, *VIFitness* also opens several RESTful API endpoints for third-party developers. This aims to create a healthy community of developers who are able to capitalise on the resources that *VIFitness* provides to develop third-party applications which can bring further utility to end users.

VIFitness's overall objective is to provide users with a comprehensive view of their daily caloric intake and expenditure, aiding them in leading more informed and healthier lifestyles.

1.5. References

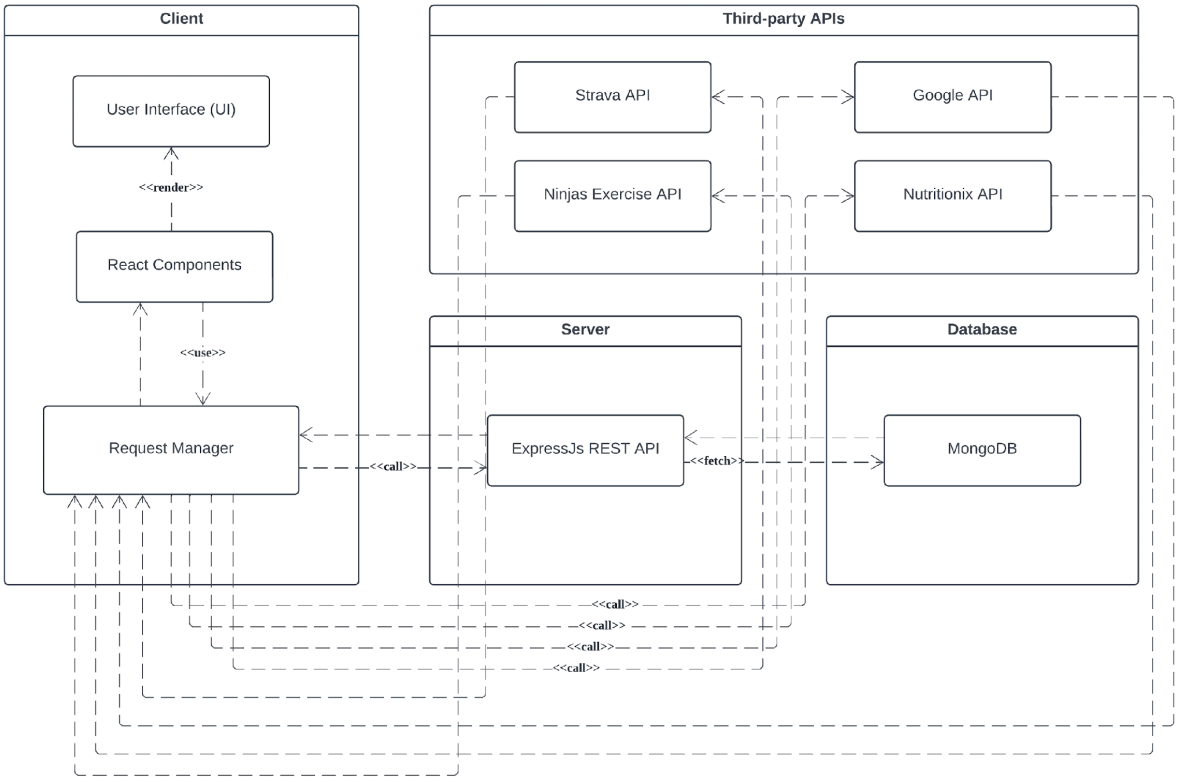
I. *IEEE 830-1998 - IEEE Recommended Practice for Software Requirements Specifications.* IEEE Standards Association. (n.d.). Retrieved 9 November, 2023 from https://standards.ieee.org/ieee/830/1222/
II. *Nutritionix v2.0 API Endpoints documentation*. Nutritionix. (2016). Retrieved 9 November, 2023 from https://developer.nutritionix.com/docs/v2
III. *Strava API v3 API and SDK Reference Developer Documentation.* Strava. (2023). Retrieved 9 November, 2023 from https://developers.strava.com/docs/reference/
IV. *Ninjas Exercise API documentation for developers.* Ninjas API. (2023). Retrieved 9 November, 2023 from https://api-ninjas.com/api/exercises
V. *React Router v6.18.0 Documentation*. Remix Software, Inc. (n.d.). Retrieved 9 November, 2023 from https://reactrouter.com/en/main
VI. *MongoDB documentation. MongoDB, Inc (2023).* Retrieved 9 November, 2023 from https://www.mongodb.com/docs/.

2. Overall Description

2.1. Product Perspective

The *VIFitness* and Nutrition Tracking Web Application is a new, self-contained product designed to serve as a comprehensive health management tool. It is not a follow-on member of an existing product family, nor a replacement for any existing systems. However, it is designed to integrate seamlessly with various health monitoring devices and applications to provide a unified health tracking experience. The product will function both as a standalone system and as a component within a broader ecosystem of health and fitness applications.

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



2.2. Product Functions

VIFitness's features consists of 5 major categories – User Account Management, Fitness Planning, Nutrition Tracking, Strava Integration and Integrated Progress Overview. This section provides a high-level summary of the system features contained in the mobile application.

2.2.1. User Account Management

i. User Registration
ii. User Login
iii. User Profile Settings
iv. Retrieve Lost Password

2.2.2. Fitness Planning

i. Query for Workouts
ii. Add Workout to Calendar
iii. Manage Workouts
iv. Display Calories Burned

2.2.3. Nutrition Tracking

i. Query for Food Nutrition
ii. Add Meals to Tracker
iii. DisplayOverall Nutritional Values
iv. Sort Meals by Date

2.2.4. Strava Integration

i. Login with Strava
ii. Connect Account to Strava
iii. Fetch Strava Activities
iv. Display Calories burned per Strava Activity

2.2.5. Integrated Progress Overview

i. Overall Dashboard view

2.3. 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	- Owns a computer/laptop - Familiar with using browser-based application
Age	All
Characteristics	Individuals seeking to track their daily health routines, likely to use the application occasionally.

2.3.2. Fitness Enthusiasts

Aspect	**Description**
Frequency of Use	Frequency of Workouts
Subset of Product Functions Used	Workout Planner, Strava Integration
Technical Expertise	- Owns a computer/laptop - Familiar with using browser-based application
Age	All
Characteristics	Individuals who are actively working towards fitness goals.

2.3.3. Nutrition-Conscious Users

Aspect	Description
Frequency of Use	Frequency of Meals
Subset of Product Functions Used	Nutrition Tracking
Technical Expertise	- Owns a computer/laptop - Familiar with using browser-based application
Age	All
Characteristics	Individuals interested in monitoring and managing their dietary habits and macros intake.

2.3.4. Health Professionals

Aspect	Description
Frequency of Use	Frequency of Monitoring their clients
Subset of Product Functions Used	All
Technical Expertise	- Owns a computer/laptop - Familiar with using browser-based application
Age	All
Characteristics	Dietitians or fitness coaches monitor the progress of their clients, requiring in-depth analytical features.

2.4. Operating Environment

The production and development environment of *VIFitness* web application will be covered under this section.

2.4.1. User Device Permissions

VIFitness requires access to internet connection to operate as all functionalities of the application are based on HTTP requests between frontend and backend server.

2.4.2. Strava Permissions

VIFitness will request for the following permissions if the user wants to fetch activities from their Strava account:

i. read
ii. activity:read
iii. activity:read_all

2.4.3. Google Permissions

VIFitness will only request for google profile data if they wish to authenticate via Google OAuth.

2.4.4. Production Environment

VIFitness's backend server is hosted with *Cyclic Software*'s cloud hosting service while *VIFitness*'s frontend interface is hosted with *Vercel*'s cloud hosting service. *VIFitness* is compatible with most web-based environments such as Chrome, Firefox, Safari, and Edge. It will be platform-independent but optimised for desktop devices.

2.4.5. Development Environment

Development Environment	Description
Front-end Development: React.js (*Version 18.2.0*)	React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript library for building user interfaces based on components. It is maintained by Meta (formerly Facebook) and a community of individual developers and companies. React can be used to develop single-page, mobile, or server-rendered applications with frameworks like Next.js. Because React is only concerned with the user interface and rendering components to the DOM, React applications often rely on libraries for routing and other client-side functionality.
Back-end Development: Express.js (*Version 4.18.2*)	Express.js is a small framework that works on top of Node.js web server functionality to simplify its APIs and add helpful new features. It makes it easier to organize your application's functionality with middleware and routing. It adds helpful utilities to Node.js HTTP objects and facilitates the rendering of dynamic HTTP objects.
Database: MongoDB (*Version 7.0*)	MongoDB, also known as Mongo, is an open-source document database used in many modern web applications. It is classified as a NoSQL database because it does not rely on a traditional table-based relational database structure. Instead, Mongo uses JSON-like documents with dynamic schemas. This means that, unlike relational databases, MongoDB does not require a predefined schema before you add data to a database. You can alter the schema at

	any time and as often as is necessary without having to set up a new database with an updated schema.

2.5. Design and Implementation Constraints

Constraints include:

- Adherence to GDPR and other relevant data protection regulations for user data.
- Compatibility with widely-used health tracking devices and third-party applications.
- Use of a specific tech stack (e.g., React for front-end, Node.js for back-end) as determined by the development team.
- Limited initial budget, necessitating prioritization of core features.

2.6. User Documentation

User documentation components will include:

2.6.1. User Manual (For Users)

The *User Manual* is a comprehensive guide for end-users to use VIFitness' core features. It can be accessed at https://sc-2006-vi-fitness.vercel.app/docs/guides/introduction/

2.6.2. VI Fitness API V1 Reference (For Developers)

The *VI Fitness API V1 Reference* is a documentation for API endpoints VIFitness opens to third-party developers. The aim is to provide third-party developers a comprehensive guide to using our API endpoints for development. It can be accessed at https://sc-2006-vi-fitness.vercel.app/docs/developerapi/getting-started/

2.7. Assumptions and Dependencies

Assumptions:

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

Dependencies:

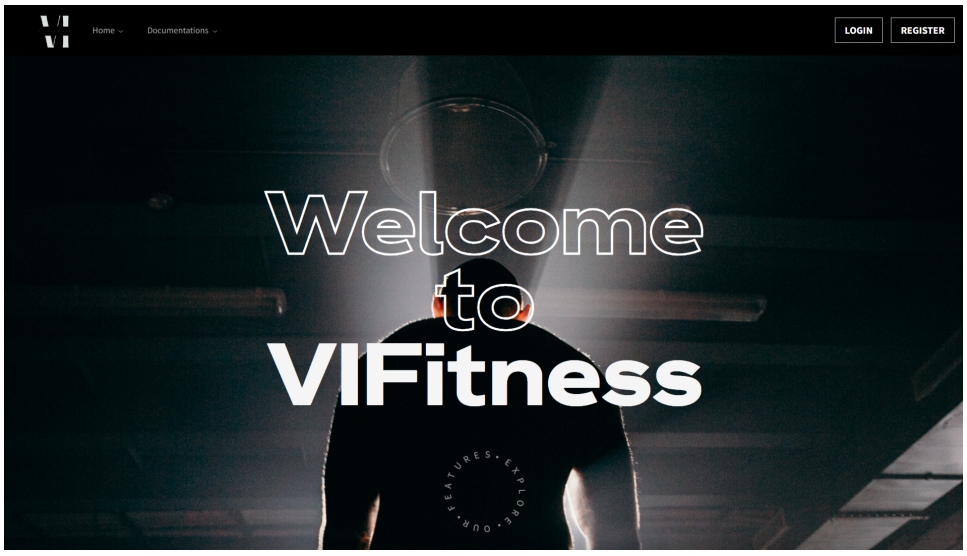
- Reliance on third-party APIs for syncing with health devices and external applications.
- The application's performance is contingent on the chosen cloud hosting service's reliability and uptime.

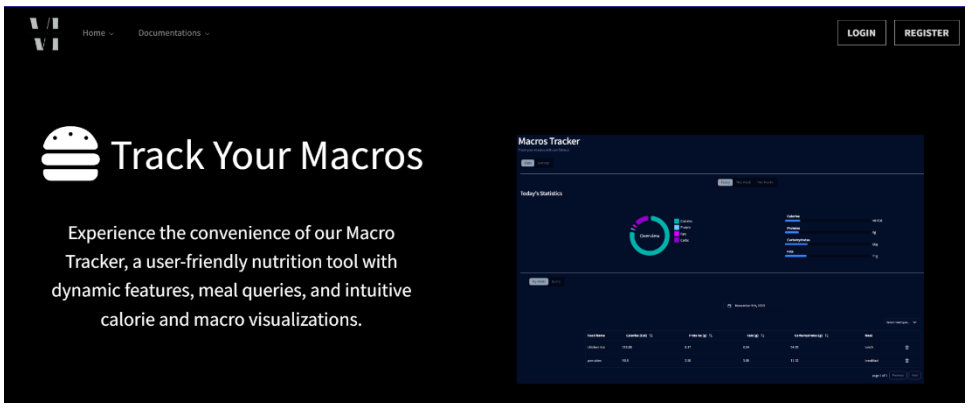
3. External Interface Requirements

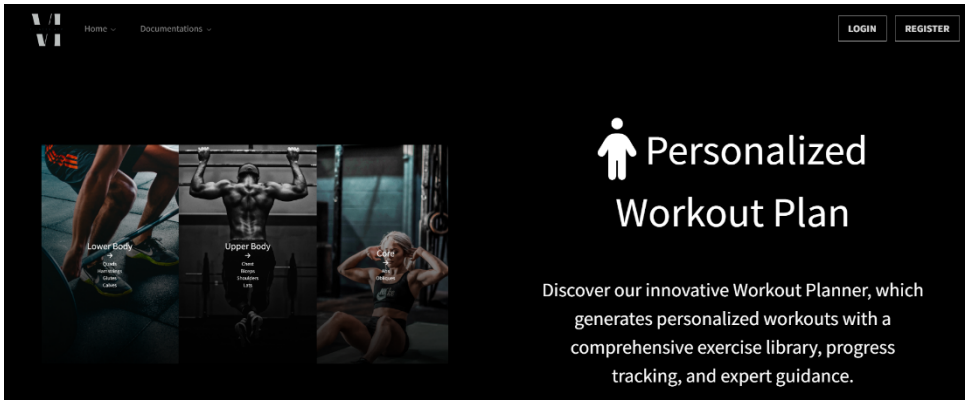
3.1. User Interfaces

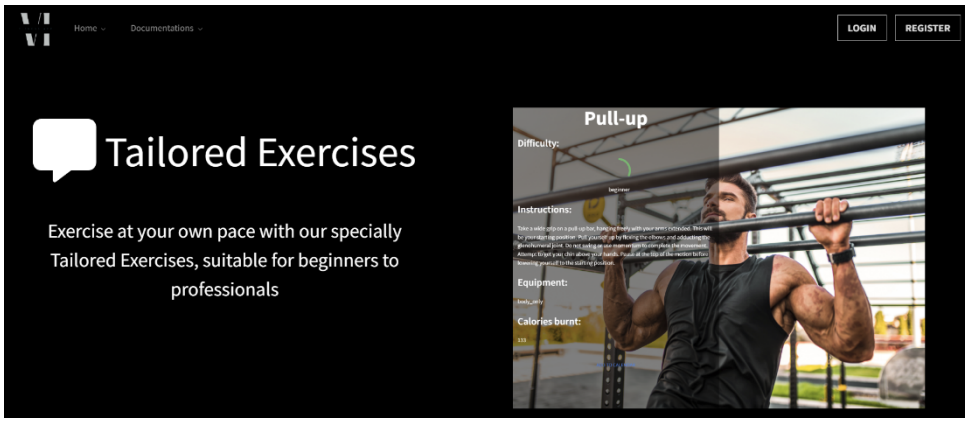
This section will showcase all GUI of VIFitness' web application.

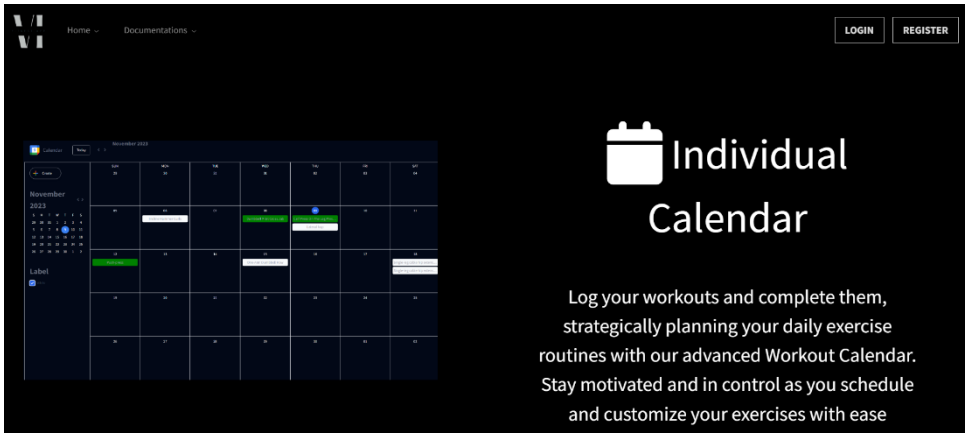
3.1.1. Landing Page

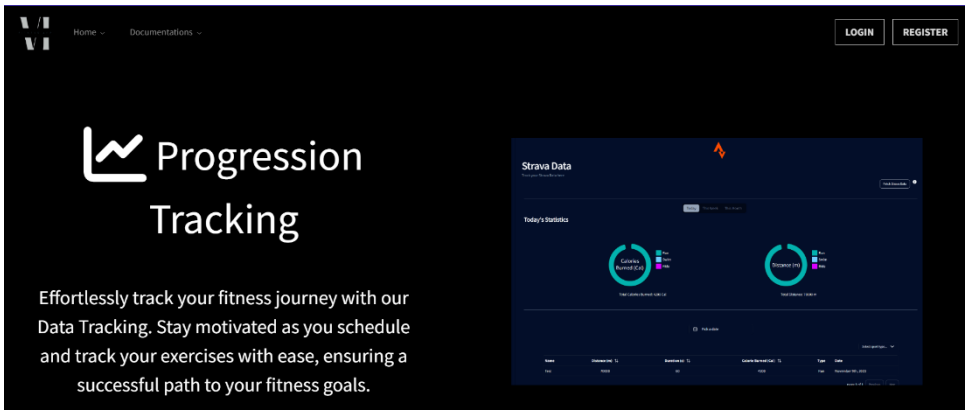




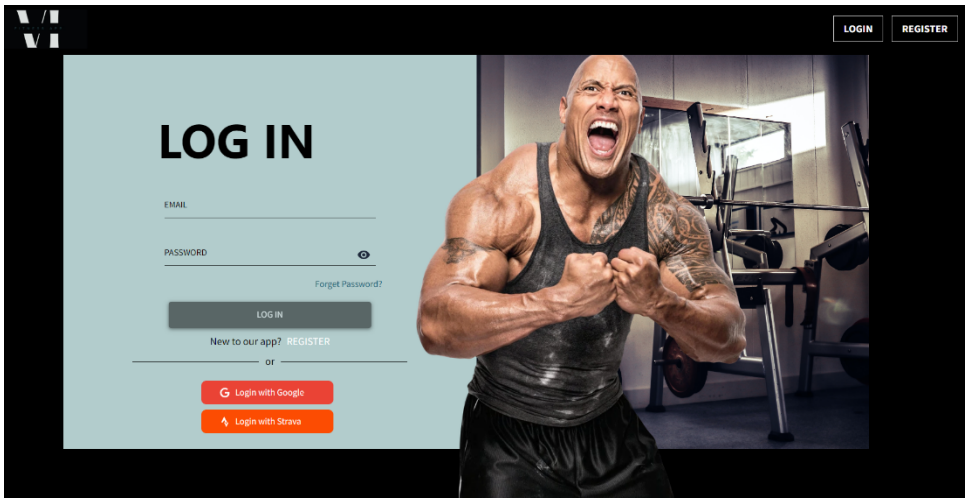

Home
Documentations
LOGIN
REGISTER
Lower Body
Upper Body
Core
Personalized Workout Plan
Discover our innovative Workout Planner, which generates personalized workouts with a comprehensive exercise library, progress tracking, and expert guidance.


Home
Documentations
LOGIN
REGISTER
Tailored Exercises
Exercise at your own pace with our specially Tailored Exercises, suitable for beginners to professionals
Pull-up
Difficulty:
Instructions:
Equipment:
Calories burnt:

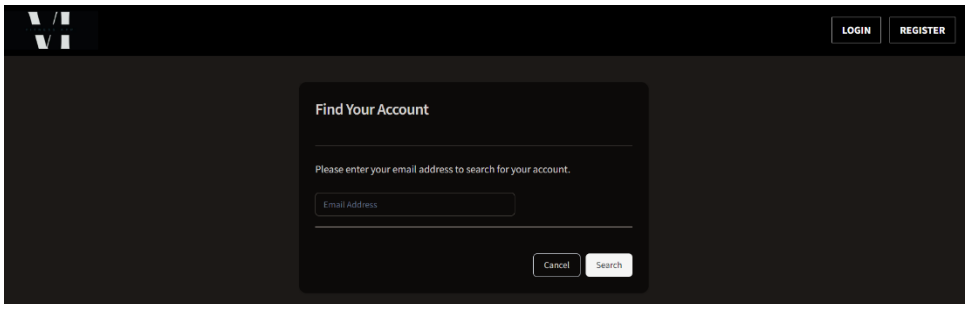

Home
Documentations
LOGIN
REGISTER
Individual Calendar
Log your workouts and complete them, strategically planning your daily exercise routines with our advanced Workout Calendar. Stay motivated and in control as you schedule and customize your exercises with ease


Home
Documentations
LOGIN
REGISTER
Progression Tracking
Effortlessly track your fitness journey with our Data Tracking. Stay motivated as you schedule and track your exercises with ease, ensuring a successful path to your fitness goals.
Strava Data
Today's Statistics

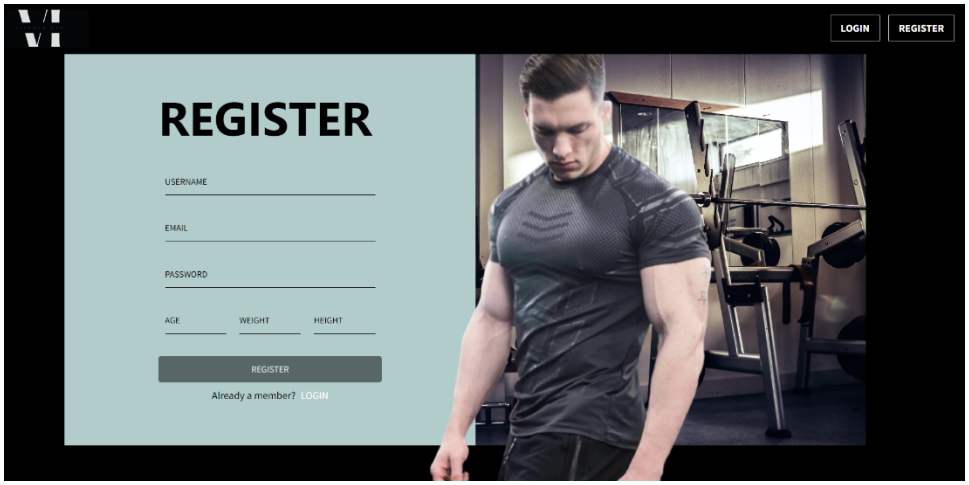
3.1.2. Login Page



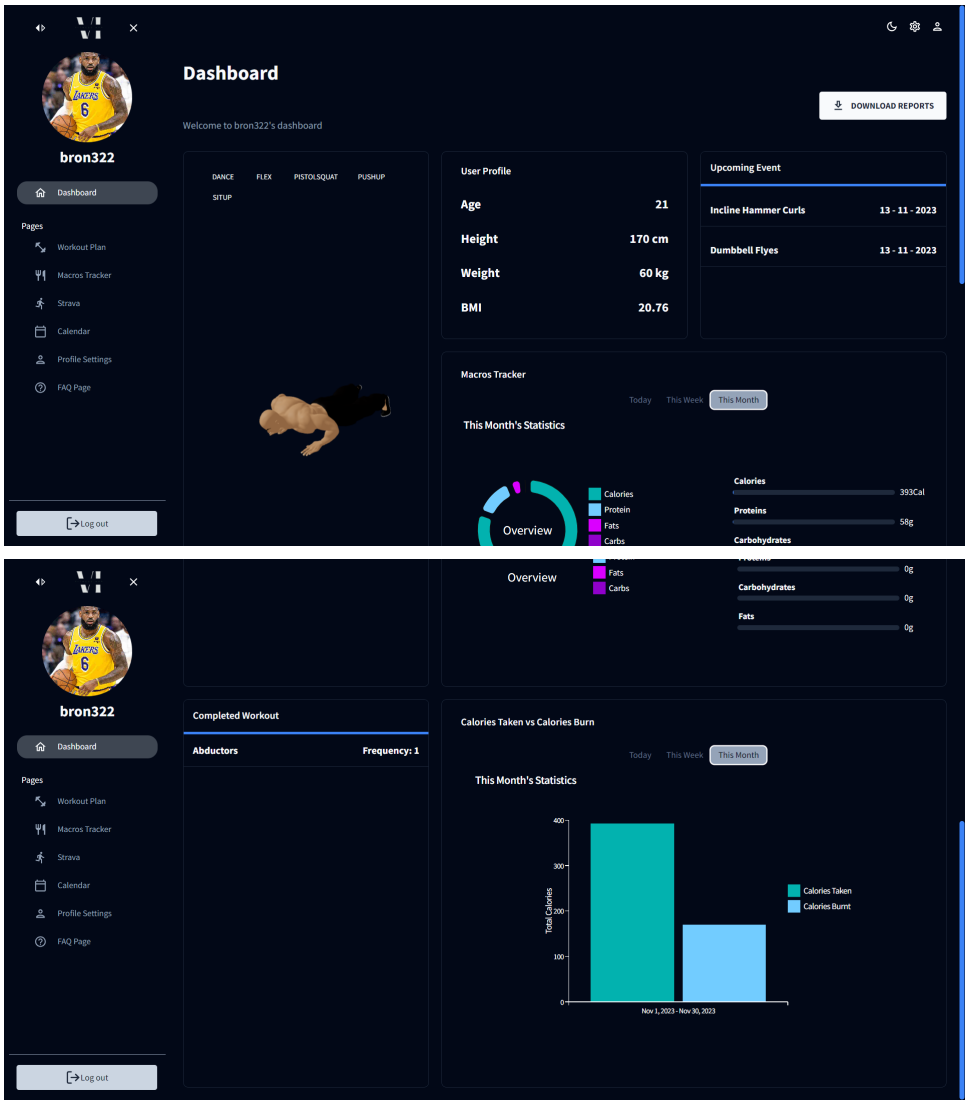
3.1.3. Forgot Password Page



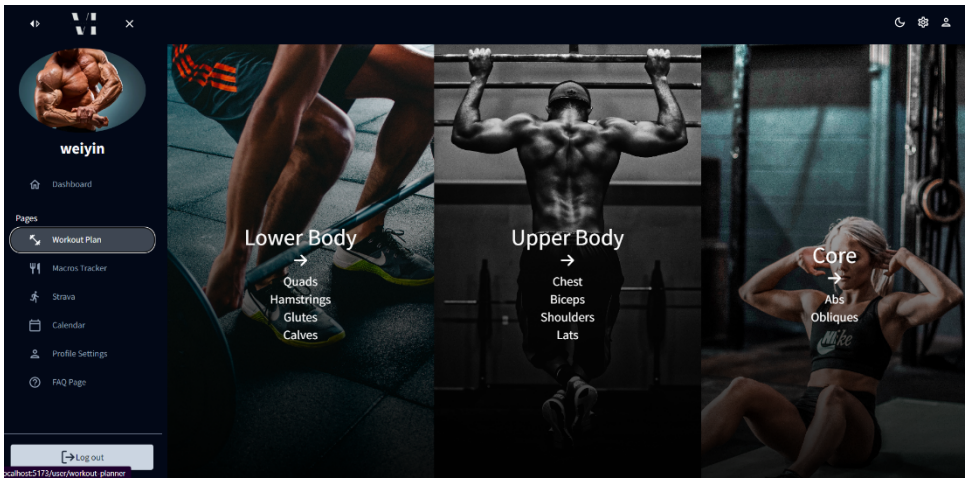
3.1.4. Registration Page



3.1.5. Dashboard Page

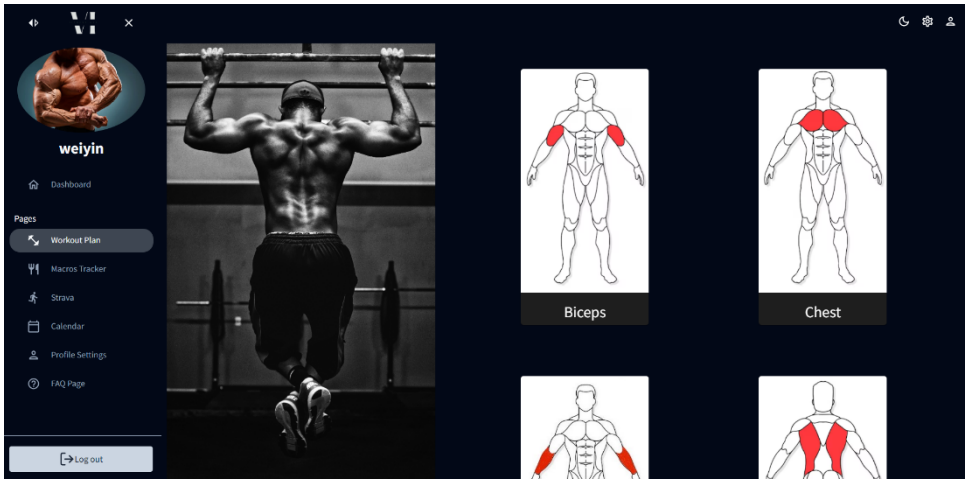


3.1.6. Workout Plan Page

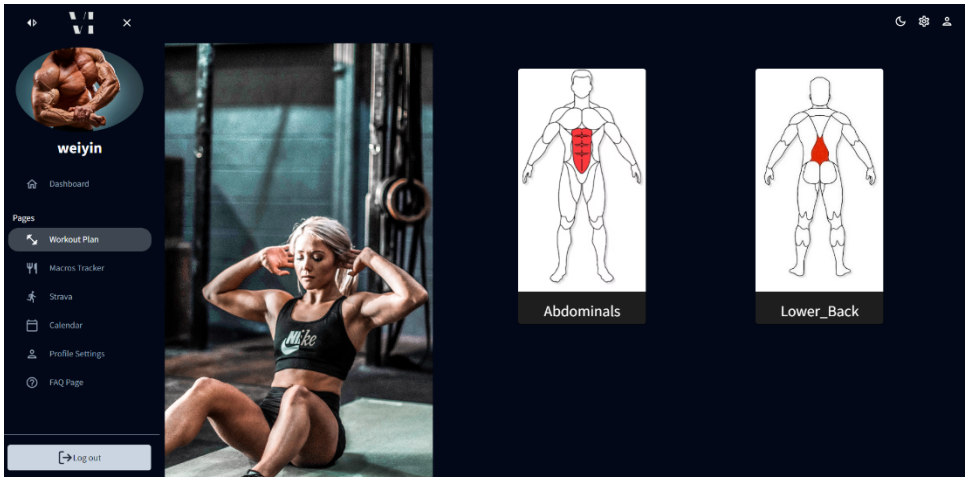


3.1.6.1. Workout Plan Lower Page

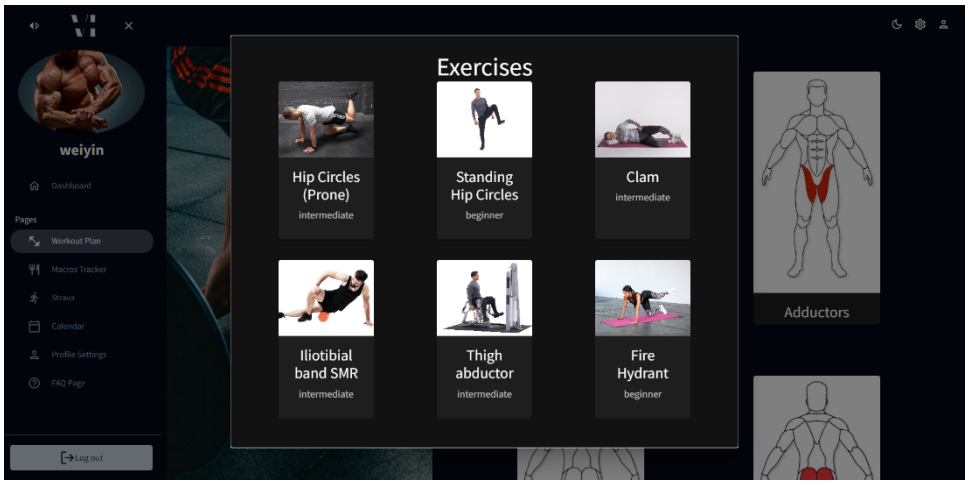
3.1.6.2. Workout Plan Upper Page



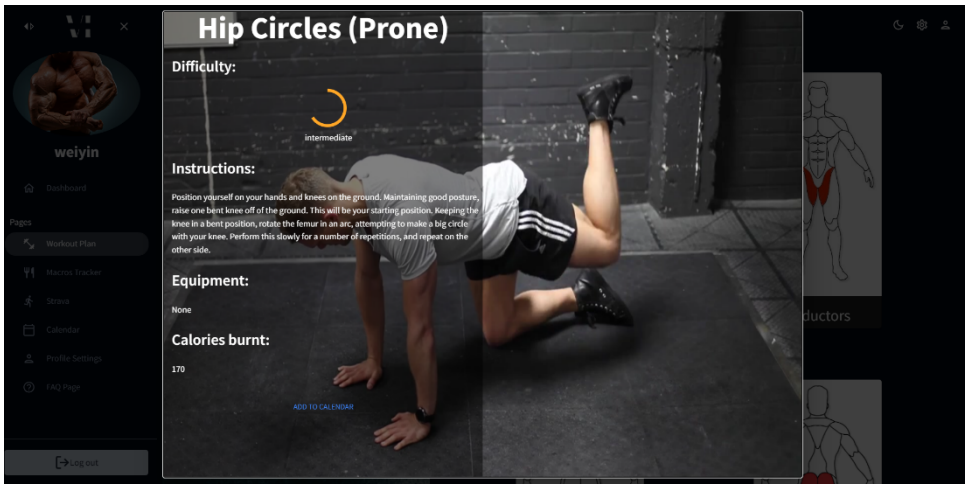
3.1.6.3. Workout Plan Core Page



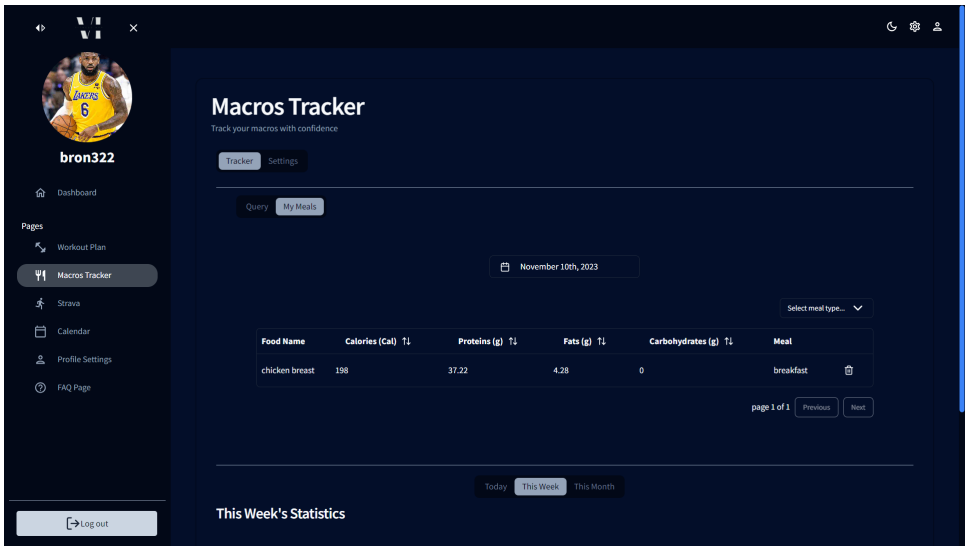
3.1.7. Workout Plan Exercise Card Page

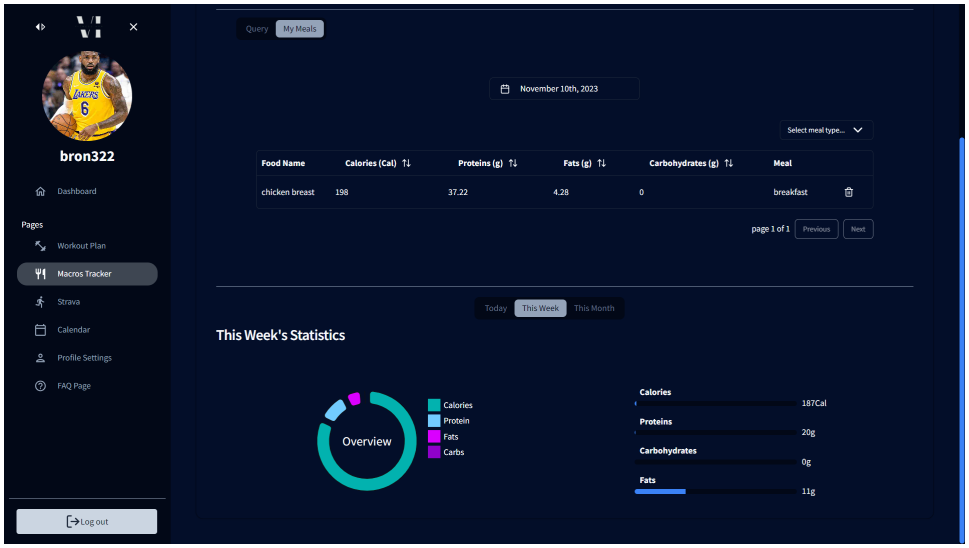


3.1.8. Workout Plan Detailed Exercise Card Page

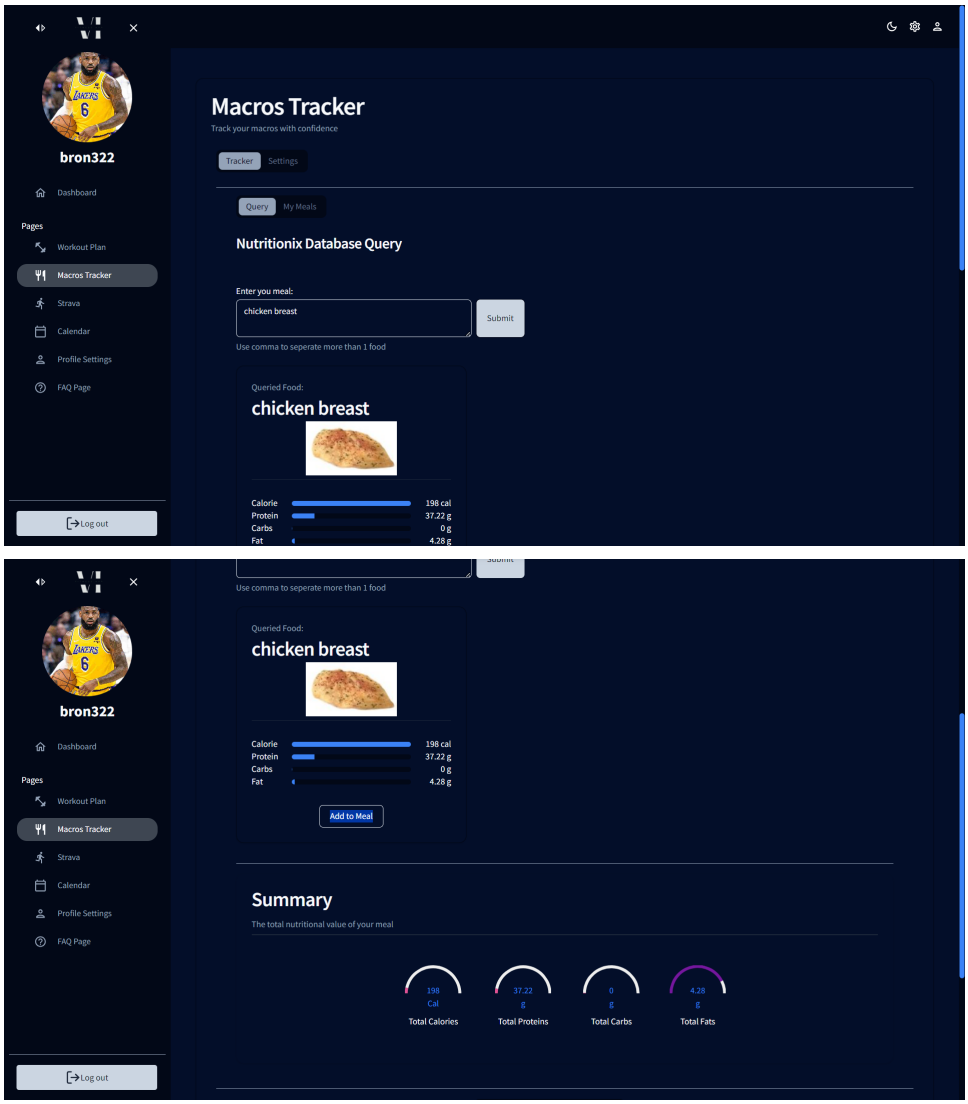


3.1.9. Macro Tracker Statistics Page

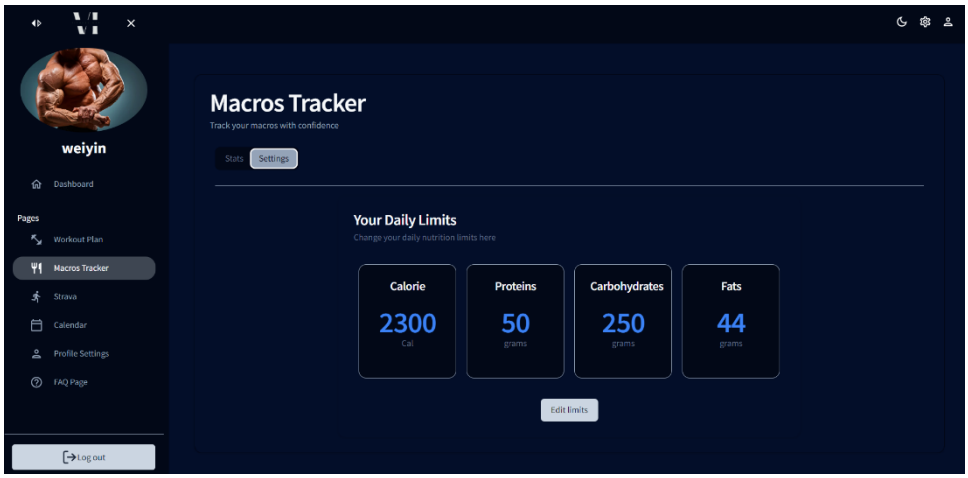



bron322
Dashboard
Pages
Workout Plan
Macros Tracker
Strava
Calendar
Profile Settings
FAQ Page
Log out
Query
My Meals
November 10th, 2023
Select meal type...
Food Name
Calories (Cal)
Proteins (g)
Fats (g)
Carbohydrates (g)
Meal
chicken breast
198
37.22
4.28
0
breakfast
page 1 of 1
Previous
Next
Today
This Week
This Month
This Week's Statistics
Overview
Calories
Protein
Fats
Carbs
Calories
187Cal
Proteins
20g
Carbohydrates
0g
Fats
11g

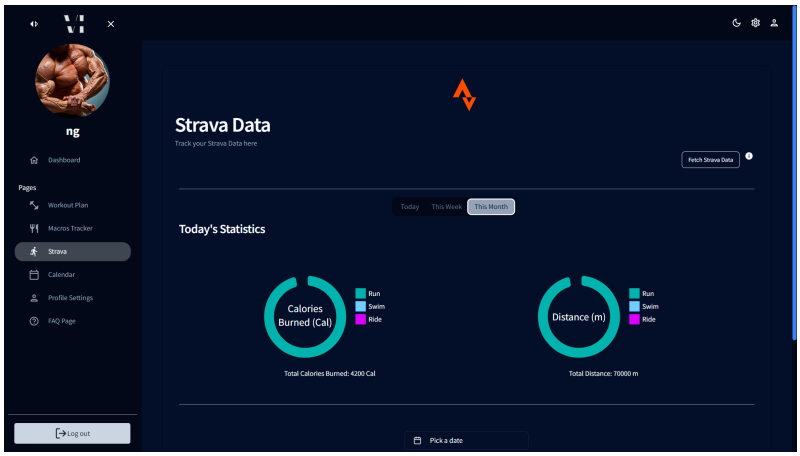
3.1.10. Macro Tracker Query Page

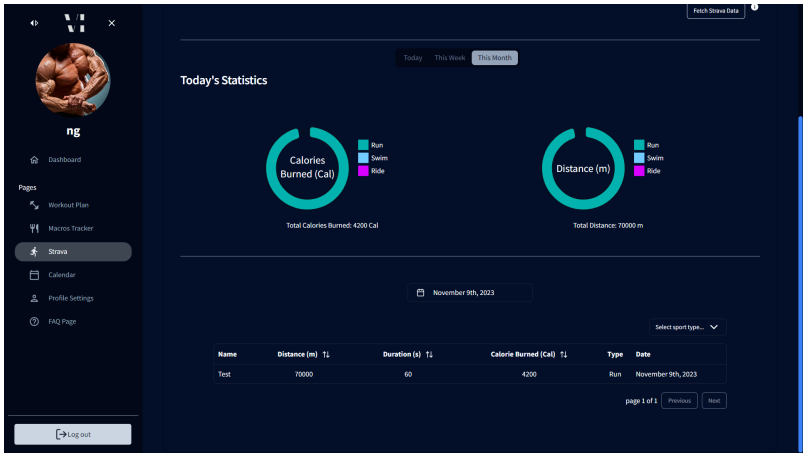


3.1.11. Macro Tracker Limit Settings Page

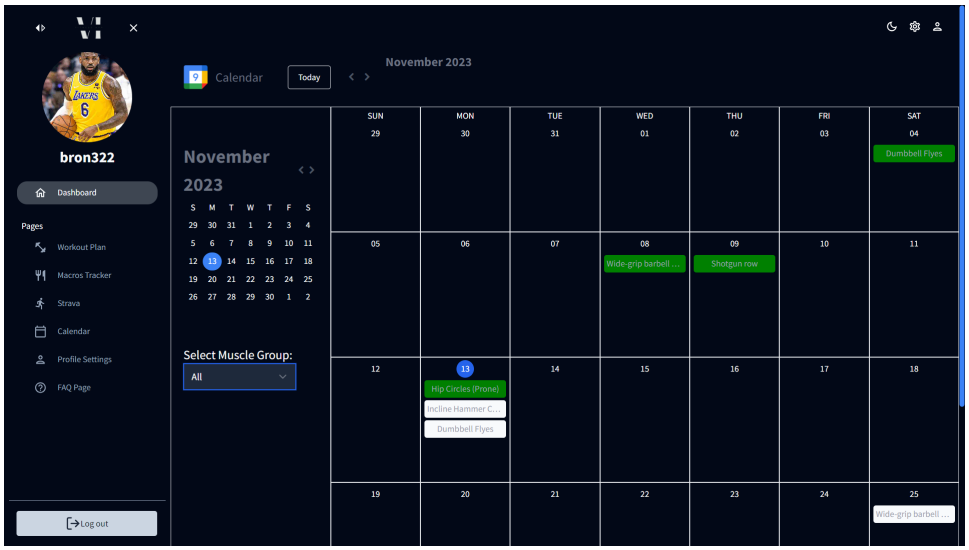


3.1.12. Strava Page

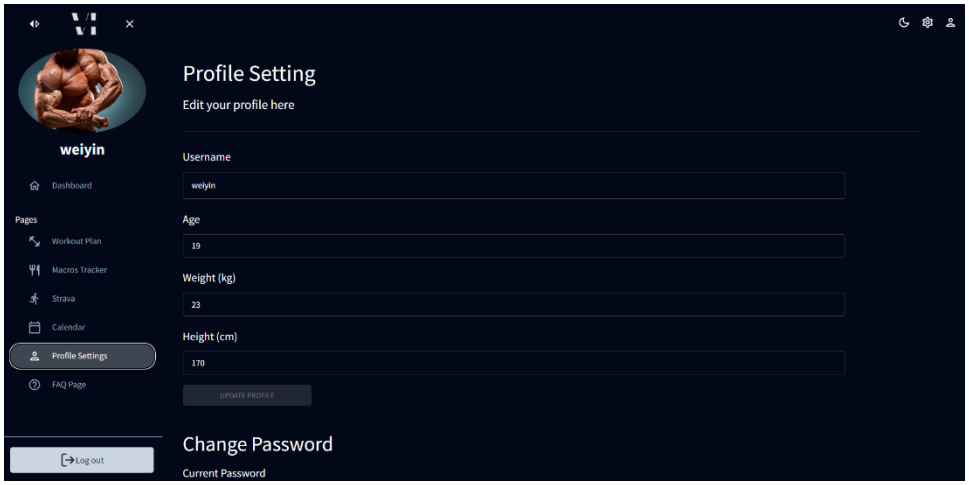


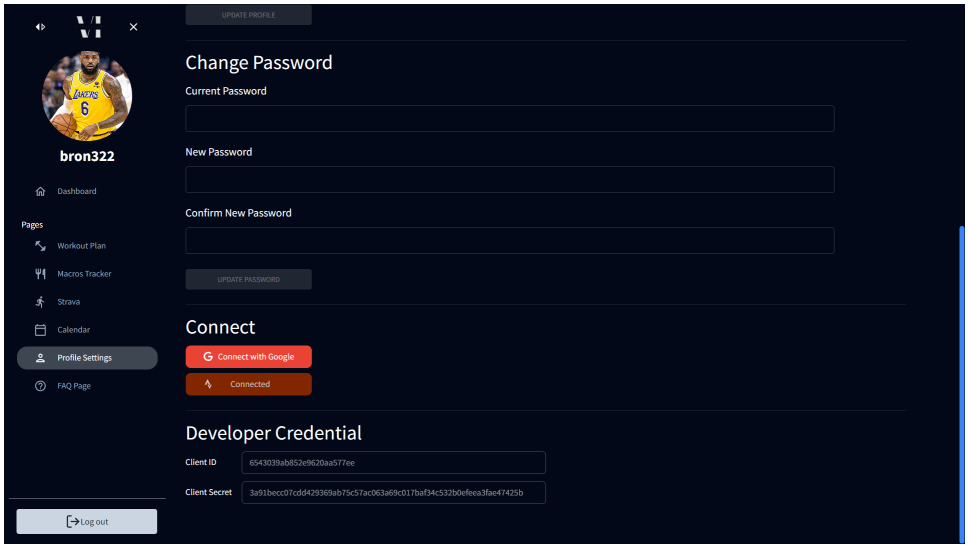


3.1.13. Calendar Page



3.1.14. Profile Settings Page





3.2. Hardware Interfaces

This section describes the hardware interfaces required for VI Fitness website to perform effectively and reliably.

Requirements	**Description**
Operating System	Devices with Windows 11 or Android 14 or iOS 17 or macOS 14 Sonoma onwards
Network Connection	Wireless Network Interface Card (WNIC) or a modem chip with cellular modem
Interaction	A functional touchpad or mouse to navigate through the website

3.3. Software Interfaces

OS: Windows 11

Tools: React JS, Tailwind CSS, Node JS, MongoDB and Express JS

Third-party libraries:

- MaterialUI
  - Library for front-end design components such as Date Picker, Data Grids, icons, etc.
- Nivo
  - Library to create data charts for data visualization
- Radix-UI
  - Library for front-end design components such as Popover, Alert-dialog, Progree, etc.
- React-OAuth
  - Request Data: Google account username and password
  - Return Data: Confirm user account authentication

  - Allows user authentication during log in and registration

- Tanstack
  - Library to create tables and data grids for data visualization

- Dayjs
  - Used to manipulate date in javascript for Calendar component

APIs:
- Nutriotionix API
  - Fetch nutritional information of meals and exercise calories burned to be displayed on Macros Tracker Page and Exercise Card component
- Strava API
  - Get access to Strava activities, allows users to register account using their Strava accounts and authenticate user accounts
- Ninjas API
  - Fetch names, instructions and equipment of exercises to display on Exercise Card components
- Google API
  - Allow users to register account using their Gmail accounts and authenticate user accounts for log in
- Breva API
  - Allows us to send email OTP to users when registering an account

3.4. Communications Interfaces

A stable internet connection is required for data to be fetched from API and user database since communications via HTTP GET, PATCH and DELETE are required. Otherwise, data from APIs and user databases will not be fetched causing the website to not be fully functional.

4. System Features

This section covers the core system features of VIFitness.

4.1. Account Registration

4.1.1. Description and Priority

New users can register an account. Upon registration, a record of the user's email address, username and password is stored in the database. Any subsequent updates made by the user, such as adding workouts to the calendar, adding meals, or updating particulars, will be using this registered account as reference.

Overall Priority	**Description**

High	A user account must be registered before the user proceeds to log in to use the app.

4.1.2. Stimulus/Response Sequences

Use Case ID:	001		
Use Case Name:	Create Account		
Created By:	Jiang Jinyi	Last Updated By:	
Date Created:	04/09/2023	Date Last Updated:	

Actor:	New User
Description:	It enables new users to create an account within our fitness app's system database. To create an account, users are required to provide their Name, Email, Phone Number, and Password.
Preconditions:	1. The selected Username must be unique and not already in use by another user. 2. Passwords must adhere to the following requirements: ● Must contain at least one special character (e.g., @, #, !). ● Must include letters in mixed cases (both uppercase and lowercase). ● Must be a minimum of 8 characters in length. 3. The provided Phone Number must not be linked to an existing user account in the system. 4. The email entered must not be associated with an existing user account.
Postconditions:	1. Upon a successful registration, the system seamlessly executes the following actions: ● Instantly dispatches a confirmation link to the user's provided Email address, ensuring account verification. ● Redirection to the main landing page of the fitness app occurs seamlessly. ● The user is now equipped to access the app's features, using their newly created Username and Password. 2. The user is now equipped to access the app's features, using their newly created Username and Password.
Priority:	High
Frequency of Use:	Low App User is only required to register for an account once.
Flow of Events:	1. The new user opens the fitness website and selects the "Register" option.

	2. The system presents the user with a registration form, soliciting the following details: ● Username ● Email ● Password ● Age ● Weight ● Height 3. The user fills in the necessary information and submits the registration form. 4. The system conducts immediate validation on the user's input, verifying compliance with the preconditions mentioned earlier. ● If any precondition is not met, the system displays precise error messages, guiding the user to correct their input. ● If the input is valid, the registration process advances.
Alternative Flows:	1. Validation Errors ● If user inputs don't meet preconditions (e.g., invalid password, duplicate Username, existing Email), the system displays error messages for correction. ● The user corrects input and resubmits for validation. ● Process continues when all input is valid. 2. Password Reset ● User can initiate a password reset if forgotten or facing login issues. ● The system sends a reset link to the user's Email. ● User clicks the link to set a new password. ● User can log in with new credentials. 3. Confirmation Link Resend: ● Users can request a resend of the confirmation link if not received or accidentally deleted. ● The system sends a new confirmation link to their Email. ● User clicks the link to confirm Email and activate the account.
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.1.3. Functional Requirements

1. User must be able to register for an account with the system.

1.1. System provides six input fields for user to input information.
    1.1.1. One of the input fields must be username.
    1.1.2. One of the input fields must be email address.
    1.1.3. One of the input fields must be password.
    1.1.4. One of the input fields must be age.
    1.1.5. One of the input fields must be height.
    1.1.6. One of the input fields must be weight.
1.2. System must ensure that user fills in all the input field before allowing registration.
1.3. System must verify all input information.
    1.3.1. System must verify that username is not taken.
    1.3.2. System must verify that email address is not taken.
    1.3.3. System must verify that password meets the requirements.
        1.3.3.1. Password must be more than 6 characters
        1.3.3.2. Password must contain at least 1 Uppercase letter
        1.3.3.3. Password must contain at least 1 Lowercase letter
        1.3.3.4. Password must contain at least 1 Special character (!@#$%^&*?)
        1.3.3.5. Password must contain at least 1 digit
    1.3.4. System must provide error message when registration is unsuccessful
1.4. System must send verification code to user's email address
    1.4.1. System must provide one input field for user to input verification code
    1.4.2. System must verify that the code input by user is correct
    1.4.3. System must provide a request code button for user to request the code again.
1.5. System must create an account for user once verification is completed
1.6. System must redirect user to the login page upon successful registration.

4.2. Account Login (Manual login)

4.2.1. Description and Priority

Existing users can login to their account. Users are required to input their email address and password as verification during the login process. Users must login prior to accessing all the features provided by the website.

Overall Priority	**Description**
High	A user account must be verified with the existing database before the app launches.

4.2.2. Stimulus/Response Sequences

Use Case ID:	002		
Use Case Name:	Login		
Created By:	Lim Jing Jie	Last Updated By:	
Date Created:	4/9/2023	Date Last Updated:	

Actor:	Existing app user
Description:	After successfully registered, app user can login with correct credentials stored in our user database.
Preconditions:	1. App User is connected to Internet. 2. App User has completed registration and verified the account. 3. The user database must be up and online
Postconditions:	Upon successful login, the user is redirected to their personalized dashboard, showing their fitness data and avatar. OR The user has been notified of reasons login failed.
Priority:	High
Frequency of Use:	
Flow of Events:	1. Upon arriving at the web app's homepage, user clicks on the "Login" button. 2. The web app presents a login form, user has to enter username/email address and password. 3. User clicks on "Login" button after entering their credentials. 4. OAuth validates the user's credential with the user database. 5. If the credentials are correct, the user is granted access to their account.
Alternative Flows:	<u>Scenario 1: If the user inputs an incorrect username or password</u> 1. When the user clicks on "LOGIN", the system will prompt the message "Wrong username/password. Please try again." 2. The user is prompted to re-enter their password. <u>Scenario 2: If the user clicks on "Forgot Password".</u> 1. The user enters username/email address and click on "Reset Password" button. 2. The system validates the user's input to ensure it matches with an existing account in the user database. 3. Once validated, the system generates a unique password reset link to the user email. 4. The user clicks on the link and is prompted to enter a new password.

	5. The user can now log into their account after the system updates the user's password in the database.
Exceptions:	Scenario 1: User has 5 consecutive failed login attempts. 1. The account will be temporarily locked for security reasons. 2. The system will prompt the user to come back after 30 minutes of cool down period. 3. LOGIN button is disabled for the specific user. Scenario2: User email address has not been verified. 1. The system checks whether the user's email address has been verified. 2. The email has not been verified. 3. The system will prompt a message "User does not exist". 4. The system will direct the user back to registration page.
Includes:	UserAuthentication
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.2.3. Functional Requirements

2. User must be able to login to their account with the system.
    2.1. System must provide two input fields for user to input information
        2.1.1. One of the input fields must be email address.
        2.1.2. One of the input fields must be password.
    2.2. System must ensure that user fills in all the input fields before allowing login.
    2.3. System must verify all input information
        2.3.1. System must verify that email address exist in the database
        2.3.2. System must verify that password matches the password of the user stored in the database.
        2.3.3. System must provide error message when login is unsuccessful
    2.4. System must redirect user to the home page upon successful login.

4.3. Google OAuth

4.3.1. Description and Priority

The process for a user to authenticate and login by using Google OAuth or create a new user if user does not exist in the database.

Overall Priority	**Description**

Medium	A user is allowed to sign with their Google credentials, this can eliminate the need for him/her to create and remember addiotional login credentials.

4.3.2. Stimulus/Response Sequences

Use Case ID:	003		
Use Case Name:	Google Authentication (Login)		
Created By:	Tan Wei Yin	Last Updated By:	
Date Created:	28/10/2023	Date Last Updated:	

Actor:	New User
Description:	This use case outlines the process for a new user to authenticate and log in using their Google account.
Preconditions:	1. The user must have a Google account.
Postconditions:	1. The user is successfully logged into the webpage using their Google account.
Priority:	High
Frequency of Use:	
Flow of Events:	1. A new user accesses the login page. 2. The user selects the option to "Log in with Google." 3. The system redirects the user to the Google authentication page. 4. The user logs in to their Google account using their Google credentials. 5. The user grants necessary permissions for the application to access their Google data (if applicable). 6. The system verifies Google authentication and retrieves relevant user information. 7. The user is logged into the application using their Google account.
Alternative Flows:	If the user cancels the Google authentication process at any point, they are returned to the login page without logging in.
Exceptions:	1. If the user provides incorrect Google login credentials, the Google authentication process fails. 2. If the user does not grant necessary permissions during the Google authentication, the application may not be able to access specific Google data.
Includes:	
Special Requirements:	
Assumptions:	

Notes and Issues:	

4.3.3. Functional Requirements

3. User must be able to login or register their Google account with the system.
    3.1. System must provide Login with Google button for user to select.
    3.2. System redirects the user to the Google authentication page once the user selects their google account.
    3.3. System must verify the email exists in the database.
        3.3.1. System must provide error message when login is unsuccessful
        3.3.2. System must create a new user if the account does not exist in the database.
    3.4. System must redirect user to the home page upon successful login or register.

4.4. Strava OAuth

4.4.1. Description and Priority

The process for a user to authenticate and login by using Strava OAuth or create a new user if user does not exist in the database.

Overall Priority	**Description**
Low to Medium	A user is allowed with their Strava account, enabling the import of running distance to allow the app to calculate the calories burnt.

4.4.2. Stimulus/Response Sequences

Use Case ID:	004		
Use Case Name:	Strava Authentication (Login)		
Created By:	Tan Wei Yin	Last Updated By:	
Date Created:	28/10/2023	Date Last Updated:	

Actor:	New User
Description:	This use case outlines the process for a new user to authenticate and log in using their Strava account.
Preconditions:	1. The user must have a Strava account.
Postconditions:	1. The user is successfully logged into the webpage using their Strava account.
Priority:	Low to Medium
Frequency of Use:	

Flow of Events:	1. A new user accesses the login page. 2. The user selects the option to "Log in with Strava." 3. The system redirects the user to the Strava authentication page. 4. The user logs in to their Strava account using their Strava credentials. 5. The user grants necessary permissions for the application to access their Strava data (if applicable). 6. The system verifies Strava authentication and retrieves relevant user information. 7. The user is logged into the application using their Strava account.
Alternative Flows:	If the user cancels the Strava authentication process at any point, they are returned to the login page without logging in.
Exceptions:	N.A.
Includes:	N.A.
Special Requirements:	N.A.
Assumptions:	N.A.
Notes and Issues:	N.A.

4.4.3. Functional Requirements

4. User must be able to login or register their Strava account with the system.
    4.1. System must provide Login with Strava button for user to select.
    4.2. System redirects the user to the Strava authentication page once the user selects their Strava account.
    4.3. System must verify the email exists in the database.
        4.3.1. System must provide error message when login is unsuccessful
        4.3.2. System must create a new user if the account does not exist in the database.
    4.4. System must redirect user to the home page upon successful login or register.

4.5. Forgot Password

4.5.1. Description and Priority

When the user forgets the user's password, the system will provide the forget password field for the user to change the password.

Overall Priority	**Description**
Medium	The application is still functioning without this feature.

4.5.2. Stimulus/Response Sequences

Use Case ID:	005		
Use Case Name:	Forgot Password		
Created By:	Tan Wei Yin	Last Updated By:	Tan Wei Yin
Date Created:	4/9/2023	Date Last Updated:	31/10/2023

Actor:	Existing User
Description:	This use case outlines the steps for an existing user to reset their password if they have forgotten it.
Preconditions:	1. The user must be an existing user of the system. 2. The user must have forgotten their password.
Postconditions:	1. The user's password is successfully reset.
Priority:	Medium
Frequency of Use:	Occasional
Flow of Events:	1. User clicks on the “Forget Password?” link on the login page. 2. User enters their email address and click “Submit” 3. The system checks if the email address exists in the database. 4. The system generates a unique password reset token and sends an email to the user with a link containing the token. 5. The user clicks on the link in the email to reset their password. 6. The user is redirected to reset password page and enter the new password. 7. The system validates the new password and updates the user’s password in the database. 8. The user is redirected to log in page.
Alternative Flows:	1. If the user enters an invalid email address during the initiation of the password reset process, the system should display an error message and prompt the user to enter a valid email. 2. If the user encounters any issues during the password reset process, they should be guided with appropriate error messages.
Exceptions:	1. If the user's email address is not found in the system, the system should display an error message suggesting that the email is not associated with any account.
Includes:	
Special Requirements:	1. The password reset token should be unique and have a short expiration time (e.g., 1 hour). 2. The password reset link in the email should be HTTPS to prevent eavesdropping.

	3. The new password should be strong and meet the application's password policy. 4. The user should be informed if their password reset request is successful or unsuccessful.
Assumptions:	
Notes and Issues:	

4.5.3. Functional Requirements

5. If the user has forgotten their password, the system must allow the user to reset their password.
    5.1. The system redirects the user to the forgot password page.
    5.2. The user will be required to enter their email.
    5.3. System must verify the email exists in database.
        5.3.1. System will prompt the error page if the user enters invalid email.
    5.4. Email must be sent out to the user with the email address given by the user.
    5.5. The email must be sent out a link which allows the user to reset their password.
    5.6. System redirects the user to reset password page when the user clicks the link.
    5.7. The new password must fulfill the requirements in 1.3.3
        5.7.1. If the password requirements are not fulfilled, then the system will prompt error messages to the user.
    5.8. System redirected the user to login page.

4.6. Add Workout to Calendar

4.6.1. Description and Priority

Users can choose a specific body part (lower, upper, core), and upon selection, the system will direct them to a page displaying the muscle group associated with the selected body part and provide a list of exercises from that muscle group using an API.

Overall Priority	**Description**
High	Its high priority is justified by its central role in the user experience, allowing the user to manage personalized workout plan and explore more on different types of workouts.

4.6.2. Stimulus/Response Sequences

Use Case ID:	006		
Use Case Name:	Add workout to calendar		
Created By:	Lim Jing Jie	Last Updated By:	Tan Wei Yin
Date Created:	4/9/2023	Date Last Updated:	31/10/2023

Actor:	Existing User, API
Description:	This use case outlines the process of generating a workout plan for a user. The user can choose a specific body part (lower, upper, core), and upon selection, the system will direct them to a page displaying the muscle group associated with the selected body part and provide a list of exercises from that muscle group using an API.
Preconditions:	1. The API is up and online. 2. The user is connected to the Internet. 3. The user has an account in the user database. 4. The user has logged into his or her account.
Postconditions:	1. The web page directs the user to a page that displays information about the selected muscle group and provides a list of exercises associated with that muscle group.
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The user navigates to the “Workout Plan" feature. 2. The user is presented with options to select a body part (lower, upper, core). 3. The user selects a specific body part. 4. The system links to the page shows different types of muscle groups. 5. The system makes an API request to fetch a list of exercises related to the selected muscle group. 6. Upon selection of an exercise, a detailed description pops out, including exercise name, difficulty, instructions, equipment required, and calories burnt. 7. The user clicks “add to calendar” and chooses a date to schedule this exercise.
Alternative Flows:	
Exceptions:	
Includes:	*View Available Exercises*
Special Requirements:	
Assumptions:	1. The user is familiar with selecting body parts and following workout plans. 2. The user has the necessary equipment and knowledge to perform the exercises provided in the workout plan.
Notes and Issues:	

4.6.3. Functional Requirements

6. The system must allow user to select a workout.
    6.1. System must provide a button for users to add exercise to their calendar.
    6.2. System must prompt a field for users to select the date.
        6.2.1. System displays a small calendar for the user to choose a date.
        6.2.2. System prompt a notification once the workout is successfully added.

4.7. View Available Exercises

4.7.1. Description and Priority

This feature allows users to browse and explore a list of exercises that they can potentially include in their workout routines.

Overall Priority	**Description**
Medium	This feature provides users with a variety of exercises categorized by muscle groups.

4.7.2. Stimulus/Response Sequences

Use Case ID:	007		
Use Case Name:	View Available Exercises		
Created By:	Prakritipong Phuvajakrt	Last Updated By:	
Date Created:	09/11/2023	Date Last Updated:	

Actor:	Existing user
Description:	The user can view available exercises under different categories of body parts and muscle groups.
Preconditions:	1. The API is up and online. 2. The user is connected to the Internet. 3. The user has an account in the user database. 4. The user has logged into his or her account.
Postconditions:	1. The user can choose whether to view exercise details such as instructions, equipment involved and calories burnt. 2. The user can decide whether to add the exercise into the calendar.
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The user navigates to the “Workout Plan" button. 2. The user is presented with options to select a body part (lower, upper, core). 3. The user selects a specific body part.

	4. The system links to the page shows different types of muscle groups. 5. The system makes an API request to fetch a list of exercises related to the selected muscle group. 6. The web page displays the list of exercises, including exercise names and descriptions.
Alternative Flows:	
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	1. The user can identify the relevant body parts and muscle groups to view the desired exercises
Notes and Issues:	

4.7.3. Functional Requirements

7. The system must allow user to view available workouts.
    7.1. System provides three body parts which are lower body, upper body and core.
    7.2. System provides various muscle groups according to the body parts selected by the user.
    7.3. System provides various exercises based on the muscle groups selected.
    7.4. System provides five types of information about the exercises selected.
        7.4.1. One of the information is difficulty level
        7.4.2. One of the information is instructions of the exercise
        7.4.3. One of the information is equipment required.
        7.4.4. One of the information is the calories burnt.
        7.4.5. System provides a button for users to add exercise to their calendar.

4.8. Request Nutritional Value of a meal

4.8.1. Description and Priority

Users have the ability to request nutritional information for a meal through an API request. Additionally, users can save meal details in the database to monitor their nutritional progress.

Overall Priority	**Description**
High	Its high priority is justified by its core functionality of showing the user's nutritional intake.

4.8.2. Stimulus/Response Sequences

Use Case ID:	008		
Use Case Name:	Request Nutritional Value of a meal (Track Macros)		
Created By:	Oscar	Last Updated By:	
Date Created:	04/09/2023	Date Last Updated:	

Actor:	Existing User, Food Database / API
Description:	User can query the nutritionix API for nutritional value of food they eat.
Preconditions:	1. The API is up and online. 2. The user is connected to the Internet. 3. The user has an account in the user database. 4. The user has logged into his or her account.
Postconditions:	The webpage displays the information requested by the user
Priority:	High
Frequency of Use:	Frequent
Flow of Events:	1. User login successfully 2. User click on ‘Macro tracker and planner’ view 3. User click on ‘request meal information’ button 4. User enter meal information 5. User click on ‘request’ button 6. Webpage fetch data from API 7. Webpage displays information of the meal
Alternative Flows:	AF1: User entered incorrect meal information 1. User delete the previous input 2. User press ‘request’ button 3. Webpage fetch data from API 4. Webpage displays information of the meal
Exceptions:	Ex1: API endpoint is down or not responding 1. API returns with error 2. Webpage displays error message “Currently this feature is not available”
Includes:	Query nutritional information of a meal, Get Food Data, Save Food Data, Filter Food Data, Pie Chart
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.8.3. Functional Requirements

8. The system must allow the user to use the query function to query a food or meal.
    8.1. The query function must allow the user to input the food or meal.
        8.1.1. The user is able to enter more than one food.
    8.2. The system displays the following nutritional information of the queried meal:
        8.2.1. Calorie
        8.2.2. Protein
        8.2.3. Carbs (Carbohydrates)
        8.2.4. Fat

4.9. Add Meal to Macros Planner

4.9.1. Description and Priority

Users are able to add or plan the meals in the planner. Nutritional information of a meal will be obtained from an API endpoint.

Overall Priority	**Description**
High	Its high priority is justified by its core functionality of monitoring the user's nutritional intake, promoting a holistic approach to both fitness and healthy eating habits.

4.9.2. Stimulus/Response Sequences

Use Case ID:	009		
Use Case Name:	Add Meal to Macros Planner (Plan Meal)		
Created By:	Oscar	Last Updated By:	
Date Created:	04/09/2023	Date Last Updated:	

Actor:	Existing User, Food Database / API
Description:	Users are able to add or plan the meals in the planner. Nutritional information of a meal will be obtained from a API endpoint
Preconditions:	1. The API is up and online. 2. The user is connected to the Internet. 3. The user has an account in the user database. 4. The user has logged into his or her account.
Postconditions:	A meal is added to the planner
Priority:	High
Frequency of Use:	Frequent

Flow of Events:	1. User login successfully 2. User click on “Macro Planner” view 3. User click on “Insert a meal” button 4. User enter meal ingredients/ information 5. User click on “submit” button 6. Webpage fetch data from API 7. Webpage displays information of the meal 8. User enter date and type of meal e.g. breakfast/lunch/dinner 9. User click on “add to planner” button 10. Webpage displays pop up confirmation message
Alternative Flows:	AF1: User entered incorrect meal information 1. Webpage displays pop up confirmation message 2. User selects “cancel” button 3. User continue editing meal information
Exceptions:	Scenario 1: API endpoint is down or not responding 1. API returns with error 2. Webpage displays error message “Currently this feature is not available”
Includes:	*Query nutritional information of a meal, Get Food Data, Save Food Data, Filter Food Data, Plan Meals,* Pie Chart
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.9.3. Functional Requirements

9. The system allows user to add and categorize meals as breakfast, lunch, or dinner to their daily planner.
    9.1. The system provides a button for user to add their meal to the meal plan.
        9.1.1. The system prompt a field for user to select the meal types:
            9.1.1.1. Breakfast
            9.1.1.2. Lunch
            9.1.1.3. Dinner
    9.2. The system records the meal and its nutritional values based on the meal type selected in the meal plan.
        9.2.1. The system allows user to filter and sort the following information:
            9.2.1.1. Calories
            9.2.1.2. Proteins
            9.2.1.3. Fats
            9.2.1.4. Carbohydrates
            9.2.1.5. Meal type (breakfast, lunch or dinner)

4.10. Pie Chart Macros

4.10.1. Description and Priority

This use case involves the presentation of nutritional information selected by the user in the form of a pie chart. Users can view the nutritional breakdown for today, the current week, or the current month.

Overall Priority	**Description**
High	Its high priority is justified by its core functionality in providing users with a visually intuitive representation of their nutritional data.

4.10.2. Stimulus/Response Sequences

Use Case ID:	010		
Use Case Name:	Pie Chart Macros		
Created By:	Tan Wei Yin	Last Updated By:	
Date Created:		Date Last Updated:	

Actor:	Existing User, Food Database / API
Description:	This use case involves the presentation of nutritional information selected by the user in the form of a pie chart. Users can view the nutritional breakdown for today, the current week, or the current month.
Preconditions:	1. The user is logged into the application. 2. The user has previously selected specific nutritional data for tracking.
Postconditions:	The application displays a pie chart representing the nutritional values for the selected timeframe (today, week, or month).
Priority:	High
Frequency of Use:	Frequent
Flow of Events:	1. User logs into the application. 2. User navigates to the "Macros Planner" section. 3. User selects the desired timeframe (today, week, month). 4. The application fetches nutritional data from the user database based on the selected timeframe. 5. The application generates a pie chart illustrating the nutritional distribution. 6. The pie chart is displayed to the user.
Alternative Flows:	
Exceptions:	
Includes:	

Special Requirements:	1. The application should support real-time data updates. 2. The pie chart should be interactive, allowing users to hover over segments for detailed information.
Assumptions:	1. The user has selected specific nutritional items for tracking. 2. The Food Database/API contains accurate and up-to-date nutritional information.
Notes and Issues:	

4.10.3. Functional Requirements

10. The system must display the pie chart in real-time, reflecting the nutritional breakdown of meals input by the user.
    - 10.1. The system must integrate with the user database to retrieve accurate and up-to-date nutritional information of the meal input by the user.
    - 10.2. The system must filter and calculate the nutritional information of meal by day, week and month.
    - 10.3. The system must allow users to choose different timeframes for the pie chart display.
        - 10.3.1. Today
        - 10.3.2. Current week
        - 10.3.3. Current month
    - 10.4. The pie chart must be interactive, allowing users to hover over segments to view detailed nutritional information.

4.11. Set Daily Macro Limits

This feature enables users to customize their daily macronutrient intake goals based on their specific dietary preferences.

Overall Priority	**Description**
High	This feature is prioritized because users must be able to tailor their nutritional targets according to their unique needs.

4.11.1. Description and Priority

4.11.2. Stimulus/Response Sequences

Use Case ID:	011		
Use Case Name:	Set Daily Macro Limits		
Created By:	Prakritipong Phuvajakrt	Last Updated By:	Tan Wei Yin
Date Created:	09/11/23	Date Last Updated:	10/11/23

Actor:	User
Description:	This use case allows users to set their daily macro limits, including calorie, protein, carbohydrate, and fat limits, within the Macro Planner page.
Preconditions:	1. The user is logged into the application. 2. The user has navigated to the Macro Planner page.
Postconditions:	The user's specified daily macro limits are saved and applied to the Macro Planner.
Priority:	Low to medium
Frequency of Use:	Occasionally
Flow of Events:	1. User navigates to the Macro Planner page. 2. Within the Macro Planner, the user selects the option to "Set Daily Macro Limits." 3. The system presents the user with fields to input their desired daily limits for calories, protein, carbohydrates, and fats. 4. The user enters the desired values for each macro nutrient. 5. The user confirms their selections. 6. The system saves the user's specified daily macro limits.
Alternative Flows:	
Exceptions:	
Includes:	
Special Requirements:	The system should provide default values for macro limits.
Assumptions:	
Notes and Issues:	

4.11.3. Functional Requirements

11. The system allows user to set their daily macro limits.
    - 11.1. The system must set a default daily macro limit for user.
    - 11.2. The system must allow user to edit their limit.
        - 11.2.1. The system provides a button for user to edit limits.
        - 11.2.2. The system prompts user a field to edit the following macro limits:
            - 11.2.2.1. Calorie
            - 11.2.2.2. Protein
            - 11.2.2.3. Carbohydrate
            - 11.2.2.4. Fat
        - 11.2.3. The system must validate the input field is in numeric value.
    - 11.3. The system must update and display the limit input by user.

4.12. Manage Calendar

4.12.1. Description and Priority

This use case enables users to manage their fitness-related events via an intuitive calendar interface, allowing them to edit, delete, and mark events as completed.

Overall Priority	**Description**
Medium to High	Calendar management is integral for user scheduling and adherence to fitness plans.

4.12.2. Stimulus/Response Sequences

Use Case ID:	012		
Use Case Name:	Manage Calendar		
Created By:	Tan Wei Yin	Last Updated By:	Tan Wei Yin
Date Created:	26/10/2023	Date Last Updated:	11/11/2023

Actor:	Existing User
Description:	This use case involves an existing user managing their calendar within the website. The user can delete events and view their calendar.
Preconditions:	1. User must log in to the webpage
Postconditions:	1. Deleted events are removed. 2. Completed events are marked and the label color turns green.
Priority:	Medium-High – Calendar management is integral for user scheduling and adherence to fitness plans.
Frequency of Use:	Daily – Users are expected to interact with the calendar regularly to organize and track their fitness and dietary routines.
Flow of Events:	1. User navigates to the calendar interface. 2. User clicks the specific event to make changes. 3. The system prompts user a form with the following details: a. Delete button b. Event title c. Date d. Description input field e. Mark as Completed button f. Save button 4. To mark an event as completed:

	a. User select the button “Mark As Completed”. b. The event will change to green to indicate completion. 5. To delete an event: a. User selects the delete button to delete an event. 6. To add a description: a. User chooses the event and input the details under the description field. b. User must click the save button to save the description entered. 7. To change an event title: a. User choose the event and change the title. b. User must click the save button to save the changes.
Alternative Flows:	AF1: User views event details 1. User clicks on an event. 2. The system displays all details associated with the event.
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.12.3. Functional Requirements

12. User must be able to manage the calendar by editing, deleting or marking events as completed.
    - 12.1. The system must display the calendar interface with the current month.
    - 12.2. The system must display the added event with the label color in white.
    - 12.3. The system allows user to make changes by clicking on the events.
        - 12.3.1. The system prompts user a form with the following details:
            - 12.3.1.1. One of the buttons must be delete button
            - 12.3.1.2. One of the input field must be event title
            - 12.3.1.3. One of the input field must be description
            - 12.3.1.4. One of the button must be mark as completed
            - 12.3.1.5. One of the button must be save
        - 12.3.2. The system must record and save the changes made by the user.
        - 12.3.3. The system must prompt a notification once the changes have been made.
        - 12.3.4. The label color of an event must turn green once the mark as completed button is clicked.
    - 12.4. The system allows user to view the next or previous month by clicking the chevron.
    - 12.5. The system allows user to return today’s date by clicking the today button.

4.13. User Data Management (User Settings)

4.13.1. Description and Priority

The application displays the settings page for users to edit their personal information such as username, age, height and weight.

Overall Priority	**Description**
High	Users must be able to control their interactions with the application. This user settings section can foster a sense of ownership.

4.13.2. Stimulus/Response Sequences

Use Case ID:	013		
Use Case Name:	User Data Management (User Settings)		
Created By:	Tan Wei Yin	Last Updated By:	
Date Created:	28/10/2023	Date Last Updated:	

Actor:	Existing User
Description:	The application displays the settings page for users to edit their personal information such as username, age, height and weight.
Preconditions:	1. The user must be logged into their account. 2. The user must have the necessary permissions to modify their data and settings.
Postconditions:	1. User information is updated based on their input. 2. Confirmation is provided to signal successful updates.
Priority:	High
Frequency of Use:	Occasional – Users update their settings as needed.
Flow of Events:	1. The user navigates to the "Profile Settings" area. 2. The user edits fields such as username, age, weight, and height. 3. The user submits the changes. 4. The system validates the new information and updates the profile accordingly. 5. A confirmation message is displayed to notify the user of the successful update.
Alternative Flows:	If changes are not confirmed by the user, the system discards any modifications and retains the previous data.
Exceptions:	If data entered is invalid, an error message is shown, and the user is prompted to provide correct information.
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.13.3. Functional Requirements

13. User must be able to edit their personal information such as username, age, height and weight.
    - 13.1. The system prompts user the profile setting form in the profile settings page.
    - 13.2. The system must provide personal information given during user registration.
    - 13.3. The system provides four input fields for user to edit their personal information.
        - 13.3.1. One of the input fields must be username.
        - 13.3.2. One of the input fields must be age.
        - 13.3.3. One of the input fields must be weight
        - 13.3.4. One of the input fields must be height
    - 13.4. The system must verify all input information.
    - 13.5. The system must prompt a confirmation message upon submission.

4.14. Change Password

4.14.1. Description and Priority

Users are able to update their passwords seamlessly within the user settings, contributing to a robust authentication process. By allowing users to change their passwords, the application enhances user control over account security, ensuring that individuals can regularly update and strengthen their access credentials.

Overall Priority	**Description**
High	This use case allows users to update their passwords, ensuring a secure and reliable authentication process.

4.14.2. Stimulus/Response Sequences

Use Case ID:	014		
Use Case Name:	Change Password (User Settings)		
Created By:	Tan Wei Yin	Last Updated By:	
Date Created:	28/10/2023	Date Last Updated:	

Actor:	Existing User
Description:	This use case enables users to update their passwords seamlessly within the user settings, contributing to a robust authentication process. By allowing users to change their passwords, the application enhances user control over account security, ensuring that individuals can regularly update and strengthen their access credentials.
Preconditions:	1. The user must be logged into their account.

	2. The user must have the necessary permissions to modify their data and settings.
Postconditions:	1. The user's password is successfully changed. 2. The user receives a confirmation message. 3. The user logs in with the new password.
Priority:	High
Frequency of Use:	Occasional – Users update their passwords as needed.
Flow of Events:	1. The user logs in with their current password and navigates to the "Profile Settings" section. 2. The user enters the current password and new password twice. 3. The user confirms the change. 4. The system validates the new password. 5. System validates and updates the password, then confirms the update to the user. 6. The system redirects user to the login page 7. The user must log in with the new password.
Alternative Flows:	1. Password change is canceled, and user is directed back to the main settings menu
Exceptions:	1. If the user provides an incorrect current password, the system displays an error message and prompts the user to try again. 2. If the user provides the new password same as current password, the system displays an error message
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.14.3. Functional Requirements

14. User must be able to change their password.
    - 14.1. The system prompts user the change password form in the user settings page.
    - 14.2. The system provides three input fields for user to change their password.
        - 14.2.1. One of the input fields must be current password.
        - 14.2.2. One of the input fields must be new password.
        - 14.2.3. One of the input fields must be confirm new password.
    - 14.3. The system must validate the following credentials:
        - 14.3.1. The system must verify the input current password matches the account's current password.
            - 14.3.1.1. If the current password entered does not match, then the system will prompt error messages to the user.
        - 14.3.2. The new password must fulfill the requirements in 1.3.3

14.3.2.1. If the password requirements are not fulfilled, then the system will prompt error messages to the user.

14.3.3. The confirm new password must match the new password entered.

14.3.3.1. If the password does not match, then the system will prompt error messages to the user.

14.4. The system must allow users to update their passwords by clicking the update password button after entering all the input field required.

14.5. System redirects the user to login page once the password has changed successfully.

4.15. Change Profile Picture

4.15.1. Description and Priority

This use case empowers users to personalize their profiles by updating their profile pictures, enhancing the overall user experience.

Overall Priority	**Description**
Medium	This use case empowers users to personalize their profiles by updating their profile pictures, enhancing the overall user experience.

4.15.2. Stimulus/Response Sequences

Use Case ID:	015		
Use Case Name:	Change Profile Picture		
Created By:	Tan Wei Yin	Last Updated By:	
Date Created:	10/11/2023	Date Last Updated:	

Actor:	Existing User
Description:	This use case outlines the steps involved when an existing user changes their profile picture.
Preconditions:	1. The user must be logged into the application. 2. The user should have navigated to the "User Settings" section.
Postconditions:	The user's profile picture is successfully updated with the new image.
Priority:	Medium
Frequency of Use:	Frequent
Flow of Events:	1. The user hover through their profile picture. 2. Within the section, the user selects the option to "Choose File."

	3. The system prompts the user to upload a new image file from their device. 4. The user selects the desired image file. 5. The system allows user to preview their profile picture. 6. The system validates the selected file to ensure it meets the acceptable image format and size criteria. 7. If the validation is successful, the system updates the user's profile picture with the new image. 8. The system displays a confirmation message to the user indicating that the profile picture has been successfully changed.
Alternative Flows:	If the user cancels the process at any step, the system returns them to the "User Settings" without updating the profile picture.
Exceptions:	If the selected file does not meet the acceptable criteria (format, size, etc.), the system displays an error message, and the user is prompted to select a different file.
Includes:	
Special Requirements:	The system should support common image formats (e.g., JPEG, PNG). There should be a limit on the file size for the profile picture.
Assumptions:	
Notes and Issues:	

4.15.3. Functional Requirements

15. User must be able to change their profile picture.
    - 15.1. The system must provides the hover effect at the profile picture section.
        - 15.1.1. The system must allow user to choose file or image from their local storage.
        - 15.1.2. The system must validate the file selected has meet the following criteria:
            - 15.1.2.1. File format must be jpeg, jpg or png.
            - 15.1.2.2. File size must be less than 30kB.
        - 15.1.3. If the criteria are not fulfilled, then the system will prompt error messages to the user.
        - 15.1.4. The system must allow user to preview before updating the profile picture
        - 15.1.5. The system must provide a submit button for user to update their profile picture.
    - 15.2. The system must prompt a confirmation message upon submission.

4.16. Connect to OAuth (User Settings)

4.16.1. Description and Priority

Users can connect their account by using their existing credentials from OAuth providers.

Overall Priority	**Description**
Medium to High	This feature provides better user experience if the users connect their VIFitness accounts with external platform and services such as Google and Strava.

4.16.2. Stimulus/Response Sequences

Use Case ID:	016		
Use Case Name:	Connect to OAuth (User Settings)		
Created By:	Tan Wei Yin	Last Updated By:	
Date Created:	28/10/2023	Date Last Updated:	

Actor:	Existing User
Description:	This use case describes the process an existing user goes through to connect their account to an OAuth provider, such as Google, within the user settings section of the application.
Preconditions:	
Postconditions:	1. The user's account is successfully connected to the OAuth provider. 2. The system retrieves relevant data from the user's OAuth account. 3. The user receives a confirmation message.
Priority:	Medium to High
Frequency of Use:	Occasionally
Flow of Events:	1. User logs in to their account. 2. User navigates to the "User Settings" section. 3. In the settings menu, the user selects the option to "Connect to OAuth." 4. The user is redirected to the OAuth provider's authentication page. 5. User logs in to their OAuth account using their OAuth credentials (e.g., Google credentials). 6. The user grants necessary permissions for the application to access their OAuth data. 7. The system retrieves relevant data from the user's OAuth account (e.g., contacts, calendar events). 8. The system validates the data retrieved, ensuring it adheres to any defined rules or constraints. 9. The user receives a confirmation message that their account is now connected to OAuth, and the data has been imported..
Alternative Flows:	

Exceptions:	1. If the user provides incorrect OAuth login credentials, the authentication process fails. 2. If the user does not grant necessary permissions during the OAuth authentication, the system cannot retrieve data.
Includes:	
Special Requirements:	
Assumptions:	1. The user is already registered and logged in. 2. The user has an account.
Notes and Issues:	

4.16.3. Functional Requirements

16. The system must allow user to connect their account by using their existing credentials from OAuth providers.
    - 16.1. The system must provide a button for user to connect their account to OAuth.
    - 16.2. The system redirects user to the authentication page once the user selects their account.
    - 16.3. The system must retrieve user's personal data from OAuth.

4.17. Sidebar

4.17.1. Description and Priority

This sidebar serves as the primary navigation hub for users. This sidebar ensures seamless access to critical features like workout plan and macros tracker.

Overall Priority	**Description**
High	Sidebar holds a high priority in VIFitness because it features essential sections like Workout Plan, Calendar, Macros Tracker, Strava and Profile Settings.

4.17.2. Stimulus/Response Sequences

Use Case ID:	017		
Use Case Name:	Sidebar		
Created By:	Tan Wei Yin	Last Updated By:	
Date Created:	05/11/2023	Date Last Updated:	

Actor:	User
Description:	User can navigate to various pages or features within the application using the sidebar menu
Preconditions:	1. User must have an account.
Postconditions:	1. User is able to access the selected page or feature through the sidebar.
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The application displays the main dashboard or landing page. 2. User sees a sidebar menu on the screen, listing various pages or features. 3. User clicks on one of the items in the sidebar. 4. The application navigates to the selected page or feature.
Alternative Flows:	
Exceptions:	
Includes:	
Special Requirements:	1. The sidebar should be consistently available throughout the application. 2. The sidebar menu should be user-friendly and easy to understand.
Assumptions:	
Notes and Issues:	

4.17.3. Functional Requirements

17. User must be able to access the selected page or feature through the sidebar.
    - 17.1. The sidebar must provide the following items for user to navigate:
        - 17.1.1. Dashboard
        - 17.1.2. Workout Plan
        - 17.1.3. Macros Tracker
        - 17.1.4. Strava
        - 17.1.5. Calendar
        - 17.1.6. Profile Settings
        - 17.1.7. FAQ Page
    - 17.2. The system must hover the items at the sidebar according to the page selected.
    - 17.3. The system must allow user to collapse the sidebar by clicking the cross icon.
    - 17.4. The system must allow user to switch the sidebar to left and right.
    - 17.5. The system must display the user's username and profile picture at the sidebar.

4.18. Topbar

4.18.1. Description and Priority

User can efficiently navigate to page or features within the application using the topbar, which offers quick access to essential functionalities.

Overall Priority	**Description**
High	Topbar holds a high priority in VIFitness because it features essential sections like theme settings and profile settings.

4.18.2. Stimulus/Response Sequences

Use Case ID:	018		
Use Case Name:	Topbar		
Created By:	Tan Wei Yin	Last Updated By:	
Date Created:	05/11/2023	Date Last Updated:	

Actor:	User
Description:	User can efficiently navigate to page or features within the application using the topbar, which offers quick access to essential functionalities.
Preconditions:	1. User must have a valid account.
Postconditions:	1. User can access the selected page or feature through the topbar
Priority:	High
Frequency of Use:	High
Flow of Events:	1. User logs into the application. 2. The topbar is displayed at the top of the screen, providing quick access to essential features. 3. User interacts with the topbar icons to navigate to different pages or access specific features.
Alternative Flows:	
Exceptions:	
Includes:	Select Theme
Special Requirements:	1. The topbar's appearance and functionality should remain consistent across different devices and screen sizes. 2. User interactions with the topbar icons should be responsive and provide visual feedback.

Assumptions:	
Notes and Issues:	

4.18.3. Functional Requirements

18. User must be able to access the feature through the topbar.
    18.1. The topbar must provide the following icons for user to access:
        18.1.1. Toggle to switch between dark and light mode
        18.1.2. Setting icon which link to the profile setting page

4.19. Select Theme

4.19.1. Description and Priority

This feature enables users to choose their preferred theme which can add a layer of personalization as well as enhancing the visual appeal of the application.

Overall Priority	**Description**
Low to Medium	A theme selection feature is implemented in VIFitness. This can be valuable for user customization hence having a moderate priority level.

4.19.2. Stimulus/Response Sequences

Use Case ID:	019		
Use Case Name:	Select theme		
Created By:	Tan Wei Yin	Last Updated By:	
Date Created:	05/11/2023	Date Last Updated:	

Actor:	User
Description:	The user can change the theme mode, such as switching between light and dark modes, for their website or application interface.
Preconditions:	1. The user must have an active account in the application.
Postconditions:	The user successfully changes the theme mode for the website or application interface.
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The user logs into their account. 2. The application's interface is displayed. 3. The user navigates to the settings or preferences section.

	4. Within the settings, the user finds the "Theme" or "Appearance" option. 5. The user selects the desired theme mode (e.g., light or dark mode). 6. The application's interface immediately reflects the chosen theme mode.
Alternative Flows:	
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.19.3. Functional Requirements

19. User must be able to switch the theme to light or dark mode.
    19.1. The system must set the default theme as dark mode.
    19.2. The system must change the color according to the theme selected.

4.20. Strava Integration

4.20.1. Description and Priority

This use case involves the Strava integration feature, allowing users to import their Strava activities from their Strava account. Calories burned for each activity are automatically calculated upon import. This will allow users to have a more comprehensive overview of their caloric expenditure together with workouts.

Overall Priority	**Description**
High	Its high priority is justified by its core functionality in providing users with a comprehensive overview of their caloric expenditure together with workouts planned in workout planner.

4.20.2. Stimulus/Response Sequences

Use Case ID:	020		
Use Case Name:	Strava Integration		
Created By:	Qian Jianheng Oscar	Last Updated By:	Qian Jianheng Oscar
Date Created:	11/11/2023	Date Last Updated:	11/11/2023

Actor:	User
Description:	This use case involves the Strava integration feature, allowing users to import their Strava activities from their Strava account. Calories burned for each activity are automatically calculated upon import. This will allow users to have a more comprehensive overview of their caloric expenditure together with workouts.
Preconditions:	1. The user is logged into the application. 2. The user has a Strava account and activities posted.
Postconditions:	The application displays a list of Strava activities the user imported and displays a statistical overview of calories burned for each activity.
Priority:	High
Frequency of Use:	Frequent
Flow of Events:	1. User logs into the application. 2. User navigate to Strava page. 3. User will be prompted to connect to Strava if they are not already connected. a. If not connected, a button will be displayed which redirects user to profile settings to connect to Strava b. User connects to Strava 4. User will be prompted to import their Strava activities if they have not imported any activities before a. If first time importing Strava activities, a button will be displayed promptly for user to import their Strava activities. 5. Imported activities will be displayed in table form 6. Statistical overview of calorie burned of each activity will be displayed in the statistic section
Alternative Flows:	AF-1. User can filter the total calorie burned by Today, This Week or This Month
Exceptions:	EX-1: User will not be able to proceed if they are unwilling to connect to Strava. EX-2: User will not be able to proceed if no Strava activities are found in their Strava account.
Includes:	
Special Requirements:	1. The application should support real-time data updates.
Assumptions:	1. Strava API server is working normally
Notes and Issues:	

4.20.3. Functional Requirements

20. The system must display the statistics of calories burned in real-time, comparing the total calories burnt by the user for each Strava activity.

20.1. The system must integrate with the user database to retrieve accurate and up-to-date total calories taken and total calories burnt by the user.
20.2. The system must filter and calculate the calories burnt by day, week and month.
20.3. The system must allow users to choose different timeframes for the statistics display.
20.3.1. Today
20.3.2. Current week
20.3.3. Current month
20.4. The statistic display must be interactive, allowing users to hover over segments to view detailed calories.
21. The system must prompt the user to connect to Strava if they are not already connected via a button which redirects the user to profile settings page
21.1. If not connected, the user must not be able to proceed further to access the other UI of Strava page
22. The system must prompt the user to import their Strava activities if it their first time importing
22.1. If the user does not have any activities in their Strava account, the user must not be able to access the other UIs of Strava page
22.2. If the user does not click on the import activities button, the user must not be able to access the other UIs of Strava page
23. The system must only import the activities in the last 3 days
24. Imported activities must be displayed in table form for users to browse

4.21. Compare Calories with Bar Chart

4.21.1. Description and Priority

This use case involves the presentation of the total calories taken and total calories burnt by the user in the form of a bar chart. Users can view the total calories for today, the current week, or the current month.

Overall Priority	**Description**
High	Its high priority is justified by its core functionality in providing users with a visually intuitive comparison between the total calories taken and total calories burnt by the user.

4.21.2. Stimulus/Response Sequences

Use Case ID:	021		
Use Case Name:	Compare Calories with Bar Chart		
Created By:	Tan Wei Yin	Last Updated By:	
Date Created:	10/11/2023	Date Last Updated:	

Actor:	User

Description:	This use case involves the presentation of the total calories taken and total calories burnt by the user in the form of a bar chart. Users can view the total calories for today, the current week, or the current month.
Preconditions:	3. The user is logged into the application. 4. The user has previously selected specific nutritional data for tracking.
Postconditions:	The application displays a bar chart representing the comparison between the total calories taken and total calories burnt for the selected timeframe (today, week, or month).
Priority:	High
Frequency of Use:	Frequent
Flow of Events:	7. User logs into the application. 8. User selects the desired timeframe (today, week, month). 9. The application calculates the calories taken and calories burnt from the user database based on the selected timeframe. 10. The application generates a bar chart illustrating the total calories taken and the total calories burnt within the timeframe selected. 11. The bar chart is displayed to the user.
Alternative Flows:	
Exceptions:	
Includes:	
Special Requirements:	2. The application should support real-time data updates. 3. The bar chart should be interactive, allowing users to hover over segments for detailed information.
Assumptions:	
Notes and Issues:	

4.21.3. Functional Requirements

25. The system must display the bar chart in real-time, comparing the total calories taken and total calories burnt by the user.
    - 25.1. The system must integrate with the user database to retrieve accurate and up-to-date total calories taken and total calories burnt by the user.
    - 25.2. The system must filter and calculate the calories taken and calories burnt by day, week and month.
    - 25.3. The system must allow users to choose different timeframes for the bar chart display.
        - 25.3.1. Today
        - 25.3.2. Current week
        - 25.3.3. Current month
    - 25.4. The bar chart must be interactive, allowing users to hover over segments to view detailed calories.

4.22. View Dashboard

4.22.1. Description and Priority

Users can effortlessly access and review their personalized data and information through the application's dashboard.

Overall Priority	**Description**
High	The designation of a high priority for the dashboard underscores its central role in the application's functionality and user satisfaction. The dashboard serves as the central hub where users effortlessly access and review their personalized data and information.

4.22.2. Stimulus/Response Sequences

Use Case ID:	022		
Use Case Name:	View Dashboard		
Created By:	Tan Wei Yin	Last Updated By:	
Date Created:	10/11/2023	Date Last Updated:	

Actor:	User
Description:	Users can effortlessly access and review their personalized data and information through the application's dashboard.
Preconditions:	The user must possess an active and valid account within the application.
Postconditions:	The user has successfully viewed their data and particulars within the dashboard interface.
Priority:	High
Frequency of Use:	Frequent
Flow of Events:	1. The user securely logs into their account. 2. The application seamlessly presents the main dashboard as their profile page. 3. The user's personal data, including account information, relevant statistics, and other pertinent content, is clearly and comprehensively displayed within the dashboard.
Alternative Flows:	
Exceptions:	
Includes:	

Special Requirements:	1. The dashboard must be easily accessible and navigable, ensuring a user-friendly experience. 2. Data presentation within the dashboard should be organized, clear, and aesthetically pleasing. 3. Regular updates are necessary to maintain the accuracy and currency of the data displayed.
Assumptions:	
Notes and Issues:	

4.22.3. Functional Requirements

26. The system must display personalized data for the user, including personal information, fitness statistics, and their workout plan.
    - 26.1. The dashboard should include interactive bar and pie charts, allowing users to visually analyze data.
    - 26.2. The system allows user to download their dashboard as pdf.
    - 26.3. The system displays the upcoming event and completed workout for user reference.
    - 26.4. The system displays interactive avatar with different animations.

5. Other Nonfunctional Requirements

5.1. Performance Requirements

i. The landing page must load within 3 seconds as measured by tools like Google PageSpeed Insights, to ensure a swift user experience and minimize bounce rates.
ii. All pages, except where specific exceptions are granted, should load within 3 seconds to maintain a consistent user experience across the application.
iii. API queries, particularly for retrieving meal data, should respond within 5 seconds even during peak usage to facilitate efficient meal planning.
iv. Input validation for email and password on the registration page should occur within 1 second to provide immediate feedback to the user

Rationale: These performance benchmarks are established to ensure that the application meets the user expectations for a fast and responsive web service, which is critical for user retention and satisfaction.

5.2. Safety Requirements

i. The application shall comply with the Health Sciences Authority (HSA) regulations relevant to digital health applications to ensure user safety.

ii. In-built mechanisms should prevent over-exertion recommendations, aligning with the Sports Singapore safety guidelines.
iii. The app's dietary guidance will adhere to the Health Promotion Board's Guidelines to ensure it meets Singapore's health standards.
iv. Any feature related to health advice must have a disclaimer and require user confirmation to understand the potential risks.

Rationale: These safety measures are required to protect the users from harm and to ensure the product meets the ethical obligations of providing health-related recommendations.

5.3. Security Requirements

i. Administrative access to user data shall be logged and audited on a monthly basis to prevent unauthorized access and maintain user privacy.
ii. User passwords are stored using industry-standard hashing and salting techniques, as per OWASP (Open Web Application Security Project) recommendations.
iii. 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.
iv. 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).
v. User accounts can only be deleted by the account owner or an administrator with the process being audited and records kept for 5 years.

Rationale: These security measures are crucial for maintaining user trust, privacy, and to comply with legal regulations surrounding data protection and privacy.

5.4. Software Quality Attributes

i. Availability: The system should be available 99.9% of the time, excluding scheduled maintenance windows, to ensure consistent access for users.
ii. Reliability: The application should have an error rate of less than 0.1% to ensure reliable performance.
iii. Usability: The application supports both light and dark themes, emphasizing intuitive design for immediate usability while adhering to standard UX principles.
iv. Maintainability: The code should be modular and well-documented to facilitate easy updates and maintenance.

Rationale: 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.

5.5. Business Rules

i. User Role Definition: Users are assigned roles with specific permissions (e.g., regular users, administrators), with each role determining access to different levels of functionality within the application.

ii. Data Access Control: Access to sensitive user data is strictly role-based, with regular users only having access to their data, and administrators to all user data for support and maintenance purposes only.

Rationale: These business rules controls ensure secure, role-specific functionality and compliance with privacy regulations, enhancing user trust and operational efficiency.

6. Other Requirements

i. Database Requirements: The application's database must support transactional integrity and be capable of handling 10,000 concurrent transactions.
ii. Legal Requirements: The application must comply with Singapore's PDPA for local users, and be prepared to align with GDPR for European users, among other international data protection laws.
iii. Reuse Objectives: Where possible, the application should be designed to allow for the reuse of components or services across the project.

Rationale: These additional requirements are specified to ensure the application is robust, legally compliant, and aligns with the broader goals of scalability and reusability within the organization's product ecosystem.

7. Testing

7.1. Black-Box Testing

7.1.1. Login

Test Case ID	001			**Test Case Priority**		High
Test Case Description	This test case validates the functionality of login feature in VIFitness web application.					
Prerequisite	1. Access to the VIFitness web application. 2. Valid user credentials (username/email and password) for testing the login functionality			**Postrequisite**		1. The user clicks on the "LOGIN" button.
Test Execution Steps						
Step No.	**Action**	**Input**	**Expected Output**	**Actual Output**	**Test Result**	**Test Comment**
1.	Open the VIFitness	N/A	Landing Page	Landing Page	Pass	[Lim Jing Jie 10/11/2023 1:59

	web application in a web browser.					PM] : Launch Successful
2.	Navigate to the login page by clicking on the “Login” button.	N/A	Navigate to Login Page	Navigate to Login Page	Pass	[Lim Jing Jie 10/11/2023 1:59 PM] : Launch Successful
3.	Enter valid login credentials (username/email and password).	Email: lebron13145@gmail.com Password: 123456	Navigate to Dashboard page	Navigate to Dashboard page	Pass	[Lim Jing Jie 10/11/2023 2:00 PM] : Log In Successful
Test Case Status		Success				

Test Case ID		002		**Test Case Priority**		High
Test Case Description		This test case validates the functionality of login feature in VIFitness web application when key in invalid credentials.				
Prerequisite		1. Access to the VIFitness web application. 2. Invalid user credentials (username/email and password) for testing the login functionality.		**Postrequisite**		1. The user clicks on the “LOGIN” button.
Test Execution Steps						
Step No.	**Action**	**Input**	**Expected Output**	**Actual Output**	**Test Result**	**Test Comment**
1.	Open the VIFitness web application in a web browser.	N/A	Landing Page	Landing Page	Pass	[Lim Jing Jie 10/11/2023 1:59 PM] : Launch Successful
2.	Navigate to the login page by clicking on the “Login” button.	N/A	Navigate to Login Page	Navigate to Login Page	Pass	[Lim Jing Jie 10/11/2023 1:59 PM] : Launch Successful
3.	Enter valid login credentials	Email: lebron13145@gmail.com	Error message	Error message	Pass	[Lim Jing Jie 10/11/2023

	(username/email and password).	Password: 222333		"Invalid credentials" is seen on top center of the page.		2:00 PM] : Login unsuccessful. "Invalid credentials" message is prompt.
Test Case Status		Success				

7.1.2. Register

Test Case ID		003		**Test Case Priority**		High
Test Case Description		This test case verifies the functionality of the user registration process in the VIFitness web application.				
Prerequisite		1. Access to the VIFitness web application. 2. A compatible web browser. 3. Stable internet connection. 4. The user must register with a Gmail email address.		**Postrequisite**		1. User receives a confirmation email.
Test Execution Steps						
Step No.	**Action**	**Input**	**Expected Output**	**Actual Output**	**Test Result**	**Test Comment**
1.	Open the VIFitness web application.	N/A	Landing Page	Landing page	Pass	[Lim Jing Jie 10/11/2023 2:13 PM] : Launch Successful
2.	Navigate to Registration page	Press Register button	Registration page	Registration page	Pass	[Lim Jing Jie 10/11/2023 2:13 PM] : Navigate to Registration Page Successful
3.	Enter valid information in the registration form fields including:	Valid Email Address, Valid FirstName, Valid LastName, Valid	Email verification pop out	Email verification pop out	Pass	[Lim Jing Jie 10/11/2023 2:13 PM] : Email verification pop out successful

	Username, Email, Password, Age, Weight and Height.	Password, Valid Confirm Password				
Test Case Status		Success				

Test Case ID	004		**Test Case Priority**	Medium	
Test Case Description	This test case verifies the behavior of the registration process when users attempt to register with invalid input data.				
Prerequisite	1. Access to the VIFitness web application. 2. A compatible web browser. 3. Stable internet connection. 4. The user registers with an invalid email address.		**Postrequisite**	N/A	
Test Execution Steps					
Step No.	**Action**	**Input**	**Expected Output**	**Actual Output**	**Test Result**	**Test Comment**
1.	Open the VIFitness web application.		Landing Page	Landing Page	Pass	[Lim Jing Jie 10/11/2022 2:24 PM] : Launch Successful
2.	Navigate to Registration page	Press Register button	Registration page	Registration page	Pass	[Lim Jing Jie 10/11/2022 2:24 PM] : Navigate to Registration Page Successful
3.	Enter valid information in the registration form fields including, Username,	Valid Email Address, Valid FirstName, Valid LastName, Valid	Email verification pop out	Email verification pop out	Pass	[Lim Jing Jie 10/11/2022 2:13 PM] : Email verification pop out successful

	Email, Password, Age, Weight and Height.	Password, Valid Confirm Password				
Test Case Status		Success				

7.1.3. Google OAuth

Test Case ID	005	**Test Case Priority**	Medium
Test Case Description	This test case ensures that the Google OAuth functionality functions correctly, allowing users to securely authenticate and log in using their Google accounts.		
Prerequisite	1. Active internet connection 2. A valid Google account for testing.	**Postrequisite**	1. The user is successfully authenticated and logged into VIFitness using Google OAuth.

Test Execution Steps						
Step No.	**Action**	**Input**	**Expected Output**	**Actual Output**	**Test Result**	**Test Comment**
1.	Open the VIFitness web application.		Landing Page	Landing Page	Pass	[Lim Jing Jie 10/11/2023 3:31 PM] : Launch Successful
2.	Navigate to Login page	Press Login button	Login page	Login page	Pass	[Lim Jing Jie 10/11/2023 3:32 PM] : Navigate to Login Page Successful
3.	Navigate to Google pop up page.	Press Login with Google button	Google pop up Page	Google pop up Page	Pass	[Lim Jing Jie 10/11/2023 3:33 PM] : Google pop up Successful
4.	Navigate	Valid	Dashboard	Dashboard	Pass	[Lim Jing Jie

	to Dashboard page	Gmail and Gmail password	page	page		10/11/2023 3:34 PM] : Login Successful
Test Case Status		Success				

7.1.4. Strava OAuth

Test Case ID	006	**Test Case Priority**	Medium
Test Case Description	This test case verifies the functionality of Strava OAuth integration in our application.		
Prerequisite	1. Access to VIFitness web application. 2. A Strava account for testing. 3. Stable internet connection.	**Postrequisite**	1. The user's strava account is successfully linked to the application.

Test Execution Steps						
Step No.	**Action**	**Input**	**Expected Output**	**Actual Output**	**Test Result**	**Test Comment**
1.	Open the VIFitness web application.		Landing Page	Landing Page	Pass	[Lim Jing Jie 10/11/2023 3:31 PM] : Launch Successful
2.	Navigate to Login page	Press Login button	Login page	Login page	Pass	[Lim Jing Jie 10/11/2023 3:32 PM] : Navigate to Login Page Successful
3.	Navigate to Strava official page.	Press Login with Strava button	Strava official Page	Strava official Page	Pass	[Lim Jing Jie 10/11/2023 3:33 PM] : Strava official Page Successful
4.	Navigate to Dashboard page	Login Strava account successfully	Dashboard page	Dashboard page	Pass	[Lim Jing Jie 10/11/2023 3:34 PM] : Login Successful

Test Case Status	Success

7.1.5. Forgot Password

Test Case ID	007	**Test Case Priority**	Medium
Test Case Description	This test case verifies the functionality of the “Forgot Password” feature in Login Page.		
Prerequisite	1. Access to VIFitness web application. 2. User forgets his or her password.	**Postrequisite**	1. User’s password is successfully reset, and they can log in using the new password.

Test Execution Steps						
Step No.	**Action**	**Input**	**Expected Output**	**Actual Output**	**Test Result**	**Test Comment**
1.	Open the VIFitness web application.		Landing Page	Landing Page	Pass	[Lim Jing Jie 10/11/2023 4:27 PM] : Launch Successful
2.	Navigate to Login page	Press Login button	Login page	Login page	Pass	[Lim Jing Jie 10/11/2023 4:28 PM] : Navigate to Login Page Successful
3.	Navigate to Forgot Password page.	Press “Forget Password?” button	Forgot Password Page	Forgot Password Page	Pass	[Lim Jing Jie 10/11/2023 4:29 PM] : Navigate to Forgot Password Page Successful
4.	Navigate to Reset Link page	Valid existing email address and press	Reset Link page	Reset Link page	Pass	[Lim Jing Jie 10/11/2023 4:30 PM] : Navigate to Reset Link

		search button				Page Successful
5.	Change to the new password	Click the reset link from email. Enter new password Confirm new password	Redirect to Login Page	Redirect to Login Page	Pass	[Lim Jing Jie 10/11/2023 4:30 PM] : Navigate to Login Page Successful
6.	Enter email and new password	Email: lebron13145@gmail.com Password: newpassword	Navigate to Dashboard page	Navigate to Dashboard page	Pass	[Lim Jing Jie 10/11/2023 4:31 PM] : Log In Successful
Test Case Status		Success				

7.1.6. Query Meal Nutrition

Test Case ID	008	**Test Case Priority**	High
Test Case Description	This test case evaluates the functionality of the “Query Meal Nutrition” feature in the app. This feature allows users to retrieve detailed nutritional information for a specific meal entered.		
Prerequisite	1. User is logged into the VIFitness app. 2. The app is connected to the Nutritionix API.	**Postrequisite**	1. The app displays accurate and detailed nutritional information for the queried meal.

Test Execution Steps

Step No.	**Action**	**Input**	**Expected Output**	**Actual Output**	**Test Result**	**Test Comment**
1.	Open the VIFitness web application.		Landing Page	Landing Page	Pass	[Lim Jing Jie 10/11/2023 9:17 PM] : Launch Successful
2.	Navigate to Dashboard page	Press Login button, Login with existing credentials	Dashboard page	Dashboard page	Pass	[Lim Jing Jie 10/11/2023 9:18 PM] : Login Successful
3.	Navigate to Macro Tracker page	Press Macro Tracker button on the side bar	Macro Tracker page	Macro Tracker page	Pass	[Lim Jing Jie 10/11/2023 9:19 PM] : Navigate to Macro Tracker Page Successful
4.	Navigate to	Press on	Nutritionix	Nutritionix	Pass	[Lim Jing Jie

	Nutritionix Database query section.	Query button	Database query section.	Database query section.		10/11/2023 9:20 PM] : Nutritionix Database query section loaded Successful
5.	Initiate query	Input egg Press Submit button	Queried Food section and Summary of the food nutrition: Total Calories, Total Protein, Total Carbs, Total Fats Note: From Nutritionix API, egg's Total Calories: 71.5 Cal Total Protein: 6.28g Total Carbs: 0.36g Total Fats: 4.76g	Queried Food section and Summary of the food nutrition: Total Calories, Total Protein, Total Carbs, Total Fats showing the exact data from Nutritionix API.	Pass	[Lim Jing Jie 10/11/2023 9:21 PM] : Queried food information shows up successfully
Test Case Status		Success				

7.1.7. Record Meal Nutrition

Test Case ID	009			**Test Case Priority**		High
Test Case Description	This test case ensures the users can record the nutrition details of a meal.					
Prerequisite	User is logged into the VIFitness app. The app is connected to the Nutritionix API.			**Postrequisite**		The recorded meal nutrition details are saved and associated with the correct user's profile. The recorded meal is reflected in the user's nutritional summary.
Test Execution Steps						
Step No.	**Action**	**Input**	**Expected Output**	**Actual Output**	**Test Result**	**Test Comment**
1.	Open the VIFitness web application.		Landing Page	Landing Page	Pass	[Lim Jing Jie 10/11/2023 9:29 PM] : Launch Successful
2.	Navigate to Dashboard page	Press Login button, Login with existing credentials	Dashboard page	Dashboard page	Pass	[Lim Jing Jie 10/11/2023 9:30 PM] : Login Successful
3.	Navigate to Macro Tracker page	Press Macro Tracker button on the side bar	Macro Tracker page	Macro Tracker page	Pass	[Lim Jing Jie 10/11/2023 9:31 PM] : Navigate to Macro Tracker Page Successful
4.	Navigate to Nutritionix Database	Press on Query button	Nutritionix Database query section.	Nutritionix Database query section.	Pass	[Lim Jing Jie 10/11/2023 9:32 PM] : Nutritionix

	query section					Database query section loaded Successful
5.	Initiate query	Input chicken breast Press Submit button	Queried Food section and Summary of the food nutrition: Total Calories, Total Protein, Total Carbs, Total Fats Note: From Nutritionix API, chicken breast's Total Calories: 198 Cal Total Protein: 37.22g Total Carbs: 0g Total Fats: 4.28g	Queried Food section and Summary of the food nutrition: Total Calories, Total Protein, Total Carbs, Total Fats showing the exact data from Nutritionix API	Pass	[Lim Jing Jie 10/11/2023 9:33 PM] : Queried food information shows up successfully
6.	Add the food item	Press on Add to Meal button	Add to Today's Meal pop up	Add to Today's Meal pop up	Pass	[Lim Jing Jie 10/11/2023 9:34 PM] : Add to Today's Meal

						pop up Successful
7.	Record nutrition details	Select Breakfast Press Confirm button	Meal added message prompted	Meal added message prompted	Pass	[Lim Jing Jie 10/11/2023 9:34 PM] : Meal added message prompt Successful
8.	Verify saved meal in My Meals	Press on My Meals button	The saved chicken breast data is reflected in a table format.	The saved chicken breast data is reflected in a table format.	Pass	[Lim Jing Jie 10/11/2023 9:34 PM] : Data table is correct
Test Case Status		Success				

7.1.8. Mark Exercise as Completed

Test Case ID	010			**Test Case Priority**		High
Test Case Description	This test case validates the functionality of marking an exercise as completed in the app. This feature will allow users to indicate that they have finished a specific exercise.					
Prerequisite	User is logged into the account. User has existing workout plan.			**Postrequisite**		The exercise completion status is reflected on the Calendar. Relevant statistics of the completed exercise is reflected in the Dashboard page.
Test Execution Steps						
Step No.	**Action**	**Input**	**Expected Output**	**Actual Output**	**Test Result**	**Test Comment**
1.	Open the VIFitness web application.		Landing Page	Landing Page	Pass	[Lim Jing Jie 10/11/2023 9:46 PM] : Launch Successful
2.	Navigate to Dashboard page	Press Login button, Login with existing credentials	Dashboard page	Dashboard page	Pass	[Lim Jing Jie 10/11/2023 9:47 PM] : Login Successful
3.	Navigate to Calendar page	Press Calendar button on the side bar	Calendar page	Calendar page	Pass	[Lim Jing Jie 10/11/2023 9:48 PM] : Navigate to Calendar Page Successful
4.	Select the exercise that I have completed	Click on the exercise “Clam” that I have	Exercise event modal pop up	Exercise event modal pop up	Pass	[Lim Jing Jie 10/11/2023 9:32 PM] : Exercise

		completed on 10/11/2023				event modal pop up successful
5.	Mark the exercise completed	Click on the "Mark as completed" button	The message "Exercise is marked as completed" pop up. The window is reloaded and the exercise becomes green to indicate completed.	The message "Exercise is marked as completed" pop up. The window is reloaded and the exercise becomes green to indicate completed.	Pass	[Lim Jing Jie 10/11/2023 9:33 PM] : Window is reloaded and the exercise became green successfully
6.	Check the metrics based on completed exercises	Click on Dashboard on the side bar.	The exercise name Clam and date 10-11-2023 is shown on Dashboard under Completed Workout. The calories burnt from the exercise is successfully reflected on the bar chart	The exercise name Clam and date 10-11-2023 is shown on Dashboard under Completed Workout. The calories burnt from the exercise is successfully reflected on the bar chart	Pass	[Lim Jing Jie 10/11/2023 9:34 PM] : Exercise metrics reflected successfully
Test Case Status		Success				

7.1.9. Delete Exercise from Calendar

Test Case ID	011			**Test Case Priority**		High
Test Case Description	This test case validates the functionality of deleting an exercise entry from the user's calendar. The purpose is to ensure that the users can successfully remove exercises, and the changes are reflected accurately in the app.					
Prerequisite	Users are logged into the app. Users have existing exercises added to their calendar.			**Postrequisite**		Selected exercise is removed from the user's calendar. Calendar view is updated to reflect the deletion. The exercise data is removed from the backend database.
Test Execution Steps						
Step No.	**Action**	**Input**	**Expected Output**	**Actual Output**	**Test Result**	**Test Comment**
1.	Open the VIFitness web application.		Landing Page	Landing Page	Pass	Lim Jing Jie 10/11/2023 10:56 PM] : Launch Successful
2.	Navigate to Dashboard page	Press Login button, Login with existing credentials	Dashboard page	Dashboard page	Pass	[Lim Jing Jie 10/11/2023 10:57 PM] : Login Successful
3.	Navigate to Calendar page	Press Calendar button on the side bar	Calendar page	Calendar page	Pass	[Lim Jing Jie 10/11/2023 10:58 PM] :

						Navigate to Calendar Page Successful
4.	Select the exercise that I have completed	Click on the exercise "Clam" that I have completed on 10/11/2023	Exercise event modal pop up	Exercise event modal pop up	Pass	[Lim Jing Jie 10/11/2023 10:59 PM] : Exercise event modal pop up successful
5.	Delete the exercise	Click on the Trash bin button in the top right corner.	The message "Exercise is deleted" pops up. The window is reloaded, and the exercise entry is no longer visible on the selected date.	The message "Exercise is deleted" pops up. The window is reloaded, and the exercise entry is no longer visible on the selected date.	Pass	[Lim Jing Jie 10/11/2023 11:00 PM] : Window is reloaded, and the exercise is removed successfully
6.	Check the dashboard	Click on Dashboard on the side bar.	The exercise name Clam and date 10-11-2023 is no longer shown on Dashboard under Completed Workout. The calories burnt from the exercise is no longer reflected on	The exercise name Clam and date 10-11-2023 is no longer shown on Dashboard under Completed Workout. The calories burnt from the exercise is no longer	Pass	[Lim Jing Jie 10/11/2023 11:01 PM] : Exercise removal reflected successfully

			the bar chart.	reflected on the bar chart.		
Test Case Status		Success				

7.1.10. Change Password

Test Case ID		012		**Test Case Priority**		Medium
Test Case Description		This test case ensures that users can successfully change their account password through application.				
Prerequisite		Users must be logged into the application.		**Postrequisite**		The user's password is updated successfully. The user is able to log in using the new password.
Test Execution Steps						
Step No.	**Action**	**Input**	**Expected Output**	**Actual Output**	**Test Result**	**Test Comment**
1.	Open the VIFitness web application.		Landing Page	Landing Page	Pass	[Lim Jing Jie 10/11/2023 11:01 PM] : Launch Successful
2.	Navigate to Dashboardpage	Login with existing credentials	Dashboard page	Dashboard page	Pass	[Lim Jing Jie 10/11/2023 11:02 PM] : Navigate to Login Page Successful
3.	Navigate to Profile Settings page.	Press Profile Settings button on sidebar	Profile Settings Page	Profile Settings Page	Pass	[Lim Jing Jie 10/11/2023 11:03 PM] : Navigate to Profile Settings Page Successful
4.	Change password.	Enter current password.	Confirmation pop up	Confirmation pop up	Pass	[Lim Jing Jie 10/11/2023 11:04 PM] : Confirmation

		Enter new password Confirm new password Press update password button				pop up successful
5.	Confirm the password change	Press update	Redirect to Landing Page	Redirect to Landing Page	Pass	[Lim Jing Jie 10/11/2023 11:05 PM] : Redirect to Landing Page successful
6.	Navigate to the login page by clicking on the “Login” button.	N/A	Navigate to Login Page	Navigate to Login Page	Pass	[Lim Jing Jie 10/11/2023 11:06 PM] : Launch Successful
7.	Enter the username and old password.	Email: lebron13145@gmail.com Password: oldpassword	Message “Invalid credentials” pops up	Message “Invalid credentials” pops up	Pass	[Lim Jing Jie 10/11/2023 11:07 PM] : Error message pops up
8.	Enter the username and new password	Email: lebron13145@gmail.com Password: newpassword	Navigate to Dashboard page	Navigate to Dashboard page	Pass	[Lim Jing Jie 10/11/2023 11:07 PM] : Log In Successful
Test Case Status		Success				

7.1.11. User Settings

Test Case ID	013			**Test Case Priority**		High
Test Case Description	This test case verifies the functionality and correctness of the “Profile Settings” feature in the application. The section allows users to customize their username, password, age, height and weight.					
Prerequisite	Access to the VIFitness web application. A registered user account with valid credentials.			**Postrequisite**		User settings are successfully updated and saved in the database. No errors occur after modifying user settings.
Test Execution Steps						
Step No.	**Action**	**Input**	**Expected Output**	**Actual Output**	**Test Result**	**Test Comment**
1.	Open the VIFitness web application.		Landing Page	Landing Page	Pass	[Lim Jing Jie 10/11/2023 4:41 PM] : Launch Successful
2.	Navigate to Dashboardp age	Login with existing credentials	Dashboard page	Dashboard page	Pass	[Lim Jing Jie 10/11/2023 4:42 PM] : Navigate to Login Page Successful
3.	Navigate to Profile Settings page.	Press Profile Settings button on sidebar	Profile Settings Page	Profile Settings Page	Pass	[Lim Jing Jie 10/11/2023 4:43 PM] : Navigate to Profile Settings Page Successful
4/	Change username, age, weight and height.	Press username display button, Type in new username	Username, Age, Weight and Height updated.	Username, Age, Weight and Height updated.	Pass	[Lim Jing Jie 10/11/2023 4:44 PM] : Profile information updated Successful and reflected in MongoDB Compass.

		Press Age, Weight and Height display button respectively, Toggle to the new age, weight and height				
Test Case Status						

7.1.12. Select Theme

Test Case ID	014			**Test Case Priority**		Low to Medium
Test Case Description	This test case verifies the Select Theme toggle button functions correctly.					
Prerequisite	User is logged into the VIFitness web app. The web app is fully loaded.			**Postrequisite**		The selected light and dark mode is applied consistently across all app pages.
Test Execution Steps						
Step No.	**Action**	**Input**	**Expected Output**	**Actual Output**	**Test Result**	**Test Comment**
1.	Open the VIFitness web application.		Landing Page	Landing Page	Pass	[Lim Jing Jie 10/11/2023 5:18 PM] : Launch Successful
2.	Navigate to Dashboardpage	Press Login button, Login with existing credentials	Dashboard page	Dashboard page	Pass	[Lim Jing Jie 10/11/2023 5:19 PM] : Login Successful
3.	Locate Select Theme Toggle button		The button is seen at the right-hand side of top bar	The button is seen at the right-hand side of top bar	Pass	[Lim Jing Jie 10/11/2023 5:20 PM] : Toggle button is seen.
4.	Toggle Theme switch	Press on the switch	Visual appearance of the app changes to Light/Dark mode accordingly	Visual appearance of the app changes to Light/Dark mode accordingly	Pass	[Lim Jing Jie 10/11/2023 5:20 PM] : Theme change Successful
5.	Toggle Theme switch for	Press on the switch	App reverts to the initial visual state	App reverts to the initial visual state	Pass	[Lim Jing Jie 10/11/2023 5:20 PM] : Theme

	Theme Reversion					reverted Successful
6.	Persistency across pages	Navigate to different pages within the app	Selected theme persists consistently	Selected theme persists consistently	Pass	[Lim Jing Jie 10/11/2023 5:20 PM] : Theme persisted Successful
Test Case Status		Success				

7.1.13. Query Exercise

Test Case ID	015		**Test Case Priority**		Medium
Test Case Description	This test case verifies the functionality of the “Query Exercise” feature in the VIFitness web app.				
Prerequisite	The user is logged into the VIFitness web app. The app is connected to API Ninjas.		**Postrequisite**		The user receives accurate and relevant exercise results based on the provided query.
Test Execution Steps					
Step No.	**Action**	**Input**	**Expected Output**	**Actual Output**	**Test Result**	**Test Comment**
1.	Open the VIFitness web application.		Landing Page	Landing Page	Pass	[Lim Jing Jie 10/11/2023 8:25 PM] : Launch Successful
2.	Navigate to Dashboardpage	Press Login button, Login with existing credentials	Dashboard page	Dashboard page	Pass	[Lim Jing Jie 10/11/2023 8:26 PM] : Login Successful
3.	Navigate to Workout Planner page	Press Workout Plan button on the side bar	Workout Plan page	Workout Plan page	Pass	[Lim Jing Jie 10/11/2023 8:27 PM] : Navigate to Workout Plan Page Successful
4.	Navigate to Workout-Lower Planner page	Press on the Lower Body button	Workout-Lower Plan page	Workout-Lower Plan page	Pass	[Lim Jing Jie 10/11/2023 8:28 PM] : Navigate to Workout-Lower Plan Page Successful
5.	Navigate to Exercise Card	Press on the wanted muscle	Exercise card consisting	Exercise card consisting	Pass	[Lim Jing Jie 10/11/2023 8:28 PM] : Navigate

		card: Abductors	of 6 exercises	of 6 exercises		to Exercise card consisting of 6 exercises Successful
6.	Navigate to Detailed Exercise Card	Press on the wanted exercise card: Hip Circles(Prone)	A single exercise card showing difficulty, instruction, equipment and calories burnt from the exercise.	A single exercise card showing difficulty, instruction, equipment and calories burnt from the exercise.	Pass	[Lim Jing Jie 10/11/2023 8:28 PM] : Navigate to Exercise card Successful
7.	Navigate to Workout-Upper Planner page	Go back to workout-planner page Press on the Upper Body button	Workout-Upper Plan page	Workout-Upper Plan page	Pass	[Lim Jing Jie 10/11/2023 8:29 PM] : Navigate to Workout-Upper Plan Page Successful
8.	Navigate to Exercise Card	Press on the wanted muscle card: Biceps	Exercise card consisting of 6 exercises	Exercise card consisting of 6 exercises	Pass	[Lim Jing Jie 10/11/2023 8:30 PM] : Navigate to Exercise card consisting of 6 exercises Successful
9.	Navigate to Detailed Exercise Card	Press on the wanted exercise card: Incline Hammer Curls	A single exercise card showing difficulty, instruction, equipment and calories burnt from the exercise.	A single exercise card showing difficulty, instruction, equipment and calories burnt from the exercise.	Pass	[Lim Jing Jie 10/11/2023 8:31 PM] : Navigate to Exercise card Successful
10.	Navigate to Workout-Lower Planner page	Go back to workout-planner page	Workout-Core Plan page	Workout-Core Plan page	Pass	[Lim Jing Jie 10/11/2023 8:32 PM] : Navigate to Workout-Core

		Press on the Core button				Plan Page Successful
11.	Navigate to Exercise Card	Press on the wanted muscle card: Abdominals	Exercise card consisting of 6 exercises	Exercise card consisting of 6 exercises	Pass	[Lim Jing Jie 10/11/2023 8:33 PM] : Navigate to Exercise card consisting of 6 exercises Successful
12.	Navigate to Detailed Exercise Card	Press on the wanted exercise card: Landmine Twist	A single exercise card showing difficulty, instruction, equipment and calories burnt from the exercise.	A single exercise card showing difficulty, instruction, equipment and calories burnt from the exercise.	Pass	[Lim Jing Jie 10/11/2023 8:34 PM] : Navigate to Exercise card Successful
Test Case Status		Success				

7.1.14. Changing Daily Macro Limits

Test Case ID	016		**Test Case Priority**		Medium
Test Case Description	This test case verifies the functionality of allowing users to change their daily macro limits in the app.				
Prerequisite	User has a registered account on the VIFitness web app. User must be logged in to the app.		**Postrequisite**		Changes made to the limits should be saved successfully and correctly in the database. Updated macro limits are reflected in the user's profile.
Test Execution Steps					
Step No.	**Action**	**Input**	**Expected Output**	**Actual Output**	**Test Result**	**Test Comment**
1.	Open the VIFitness web application.		Landing Page	Landing Page	Pass	[Lim Jing Jie 10/11/2023 8:47 PM] : Launch Successful
2.	Navigate to Dashboard page	Press Login button, Login with existing credentials	Dashboard page	Dashboard page	Pass	[Lim Jing Jie 10/11/2023 8:48 PM] : Login Successful
3.	Navigate to Macro Tracker page	Press Macro Tracker button on the side bar	Macro Tracker page	Macro Tracker page	Pass	[Lim Jing Jie 10/11/2023 8:49 PM] : Navigate to Macro Tracker Page Successful
4.	Navigate to Macro Tracker settings page.	Press on Settings button	Macro Tracker settings page	Macro Tracker settings page	Pass	[Lim Jing Jie 10/11/2023 8:50 PM] : Macro Tracker settings Page Successful

5.	Navigate to Settings pop up page	Press Edit Limits button	Settings pop up page	Settings pop up page	Pass	[Lim Jing Jie 10/11/2023 8:51 PM] : Settings pop up Successful
6.	Adjust Macro Limits	Input the values for desired carbohydrates, protein, fat and calories Press Save Changes button.	Updated macro limits are displayed accurately.	Updated macro limits are displayed accurately.	Pass	[Lim Jing Jie 10/11/2023 8:52 PM] : Adjust Macro Limits Successful
Test Case Status		Success				

7.1.15. View Daily Macro Statistics

Test Case ID	017			**Test Case Priority**		Medium
Test Case Description	This test case verifies that users can accurately view their daily macro statistics in the VIFitness web app.					
Prerequisite	A registered user account on VIFitness. The user has set specific daily macro limits. Otherwise, the user is set to use the default daily macro limits.			**Postrequisite**		User is presented with an accurate and updated display of their daily macro limits
Test Execution Steps						
Step No.	**Action**	**Input**	**Expected Output**	**Actual Output**	**Test Result**	**Test Comment**
1.	Open the VIFitness web application.		Landing Page	Landing Page		
2.	Navigate to Dashboardp age	Press Login button, Login with existing credentials	Dashboard page	Dashboard page		
3.	Navigate to Macro Tracker page	Press Macro Tracker button on the side bar	Macro Tracker page	Macro Tracker page		
4.	Interact with the circular pie chart in the “Today’s Statistics” section.		When hover, components and weightage like (Calories 8.21) will be shown.	When hover, components and weightage like (Calories 8.21) will be shown.		
5.	Interact with the circular	Press This week button	When hover,	When hover,		

	pie chart in the “This Week’s Statistics” section.		components and weightage like (Calories 100.56) will be shown. Note: Data must be the accumulation of this week’s food components	components and weightage like (Calories 100.56) is shown.		
6.	Interact with the circular pie chart in the “This Month’s Statistics” section.	Press This Month button	When hover, components and weightage like (Calories 380.21) will be shown. Note: Data must be the accumulation of this month’s food components.	When hover, components and weightage like (Calories 380.21) is shown.		
Test Case Status						

7.1.16. Add Workout to Calendar

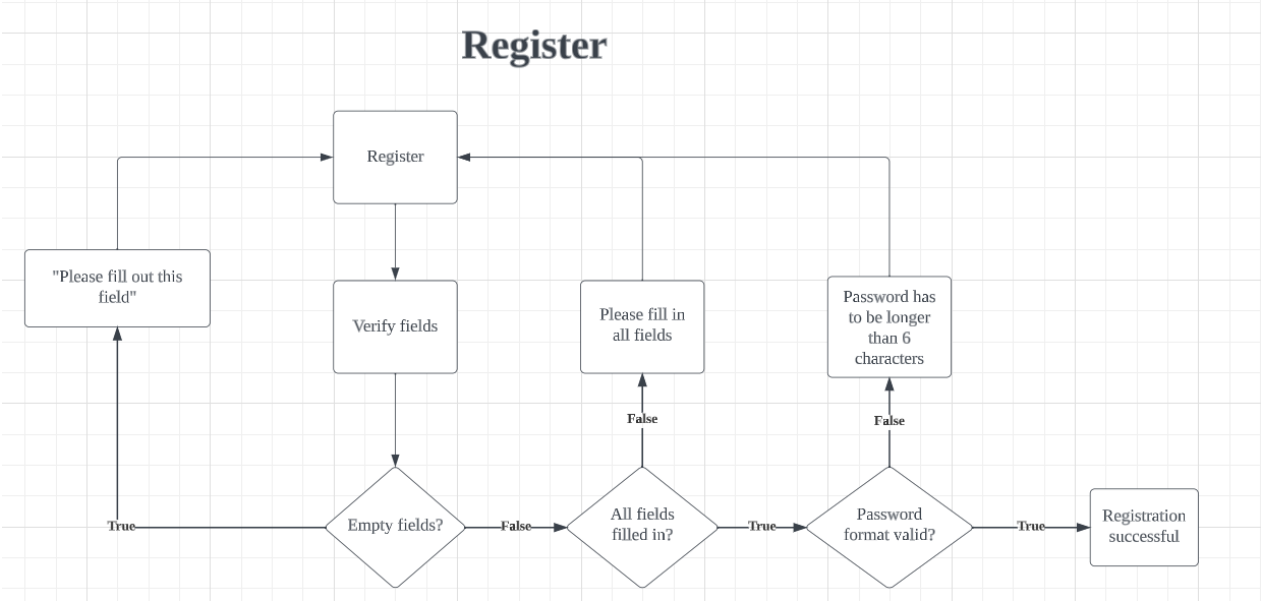
Test Case ID	016	**Test Case Priority**	High
Test Case Description	This test case outlines the process of adding an exercise to the calendar.		

Prerequisite		- User has knowledge on the body parts and muscle groups for each exercise. - User is online		**Postrequisite**		
Test Execution Steps						
Step No.	**Action**	**Input**	**Expected Output**	**Actual Output**	**Test Result**	**Test Comment**
1.	Navigate to 'Workout Plan'	NA	Workout Planner Page	Workout Planner Page	Pass	[Prakritipong Phuvajakrt 12/11/2023 12:32 am] Navigate to Workout Planner Page Successful
2.	Select 'Lower Body' or 'Upper Body' or 'Core'	Lower Body	Lower Body Exercises Page	Lower Body Exercises Page	Pass	[Prakritipong Phuvajakrt 12/11/2023 12:32 am] Navigate to Lower Body Exercises Page Successful
3.	Choose Muscle Group	Hamstrings	6 Exercise Cards consisting of hamstring exercises present	6 Exercise Cards consisting of hamstring exercises present	Pass	[Prakritipong Phuvajakrt 12/11/2023 12:32 am] Hamstring exercises generated
4.	Choose exercise	Clean Deadlift	Clean Deadlift details shown e.g. instructions, equipment involved, Calories burnt, Number of reps	Clean Deadlift details shown e.g. instructions, equipment involved, Calories burnt, Number of reps	Pass	[Prakritipong Phuvajakrt 12/11/2023 12:32 am] Details of exercise chosen presented

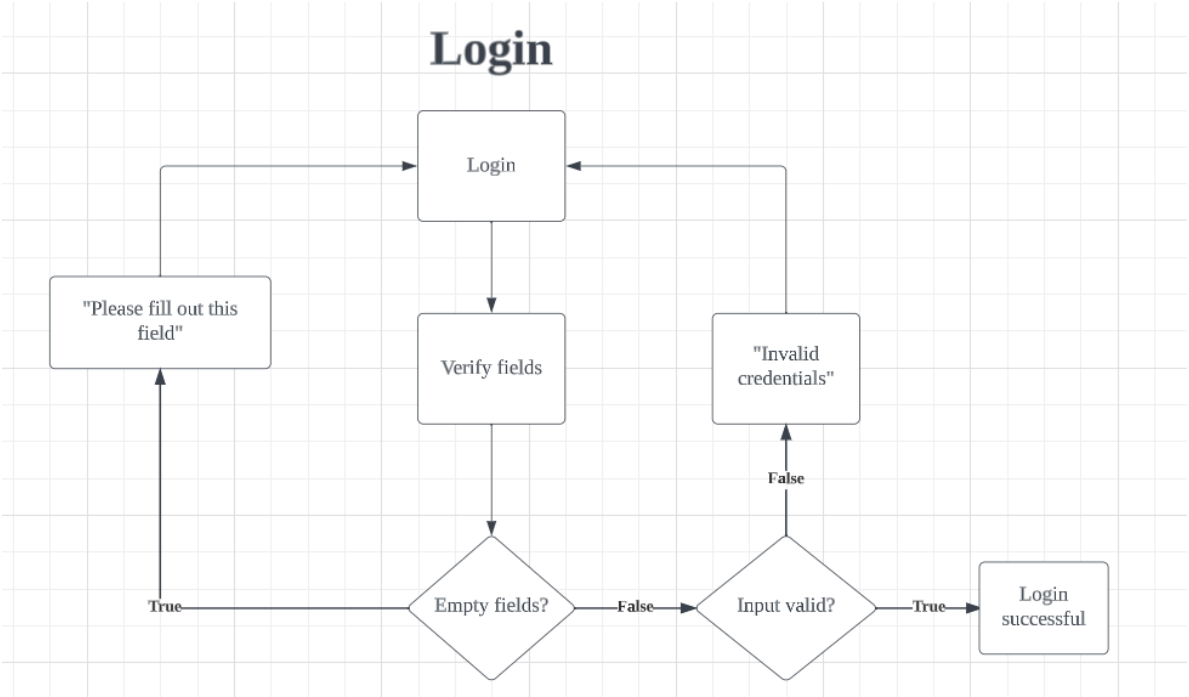
5.	Click 'Add to calendar', choose date and confirm date	20 November 2023	Exercise event reflected on the calendar on 20 November 2023	Exercise event reflected on the calendar on 20 November 2023	Pass	[Prakritipong Phuvajakrt 12/11/2023 12:32 am] Exercise added to calendar successful
Test Case Status		Success				

7.2. White-Box Testing

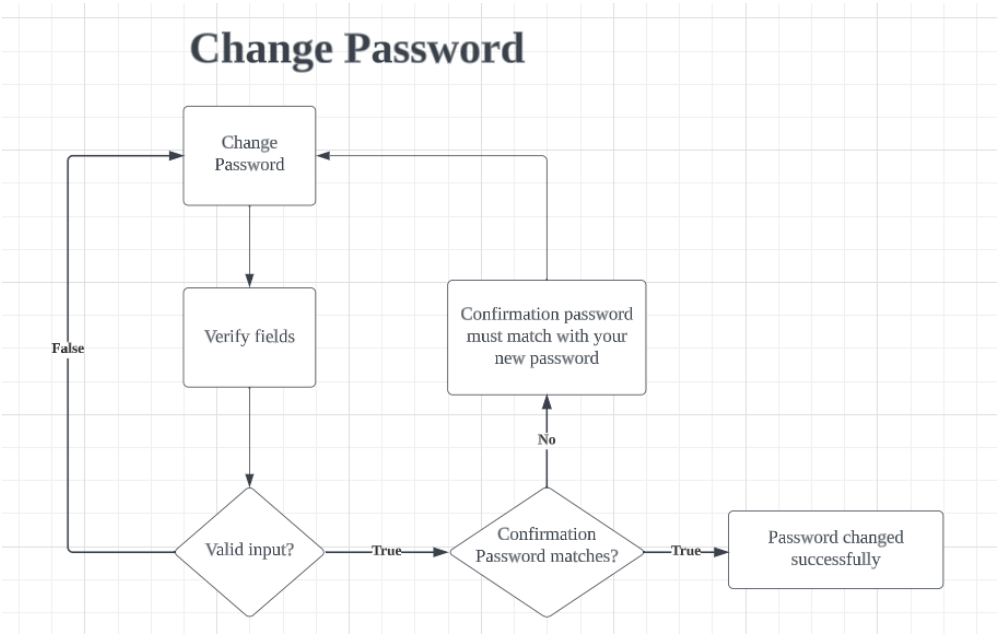
7.2.1. Register



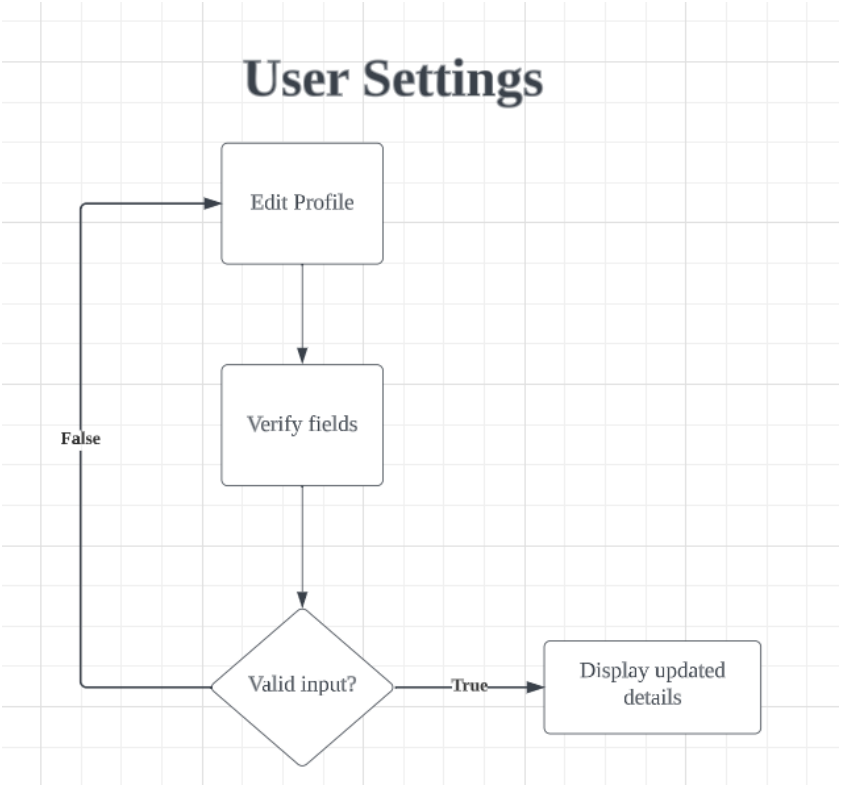
7.2.2. Login



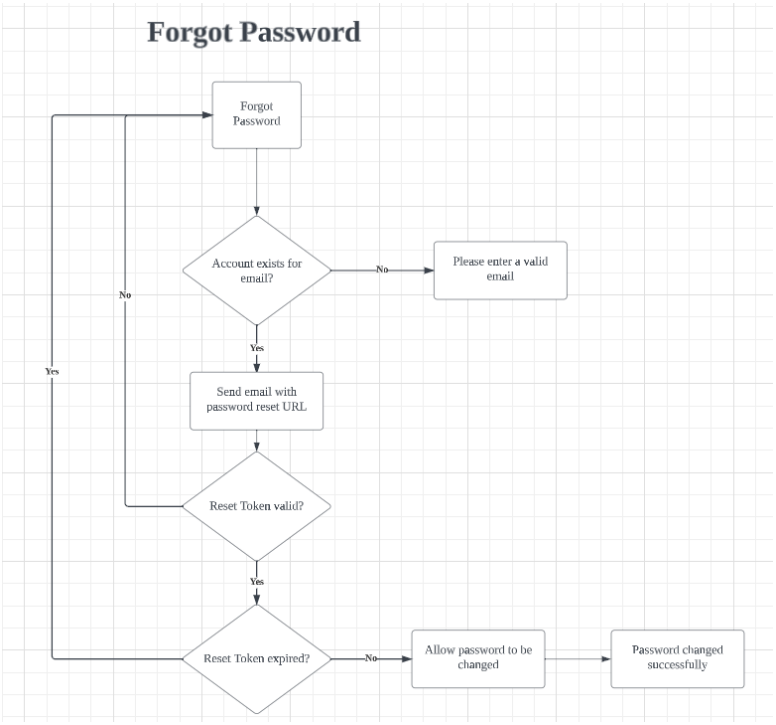
7.2.3. Change Password



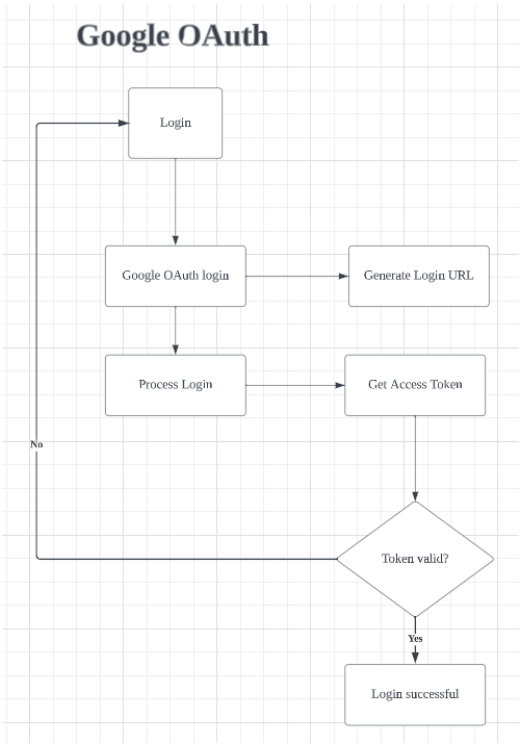
7.2.4. User Settings



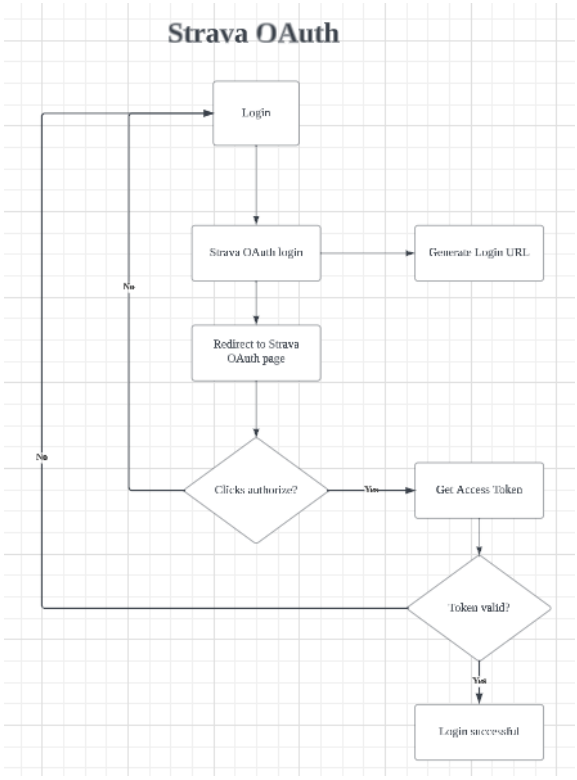
7.2.5. Forgot Password



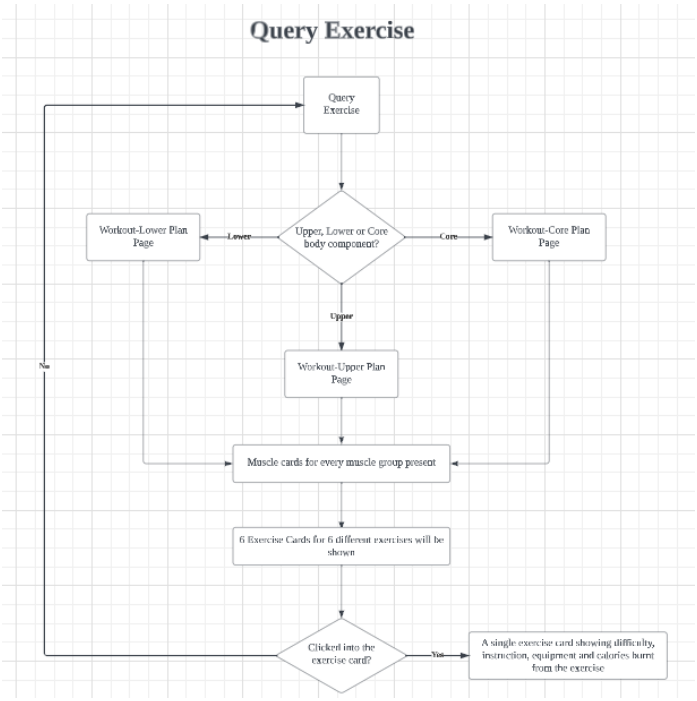
7.2.6. Google OAuth



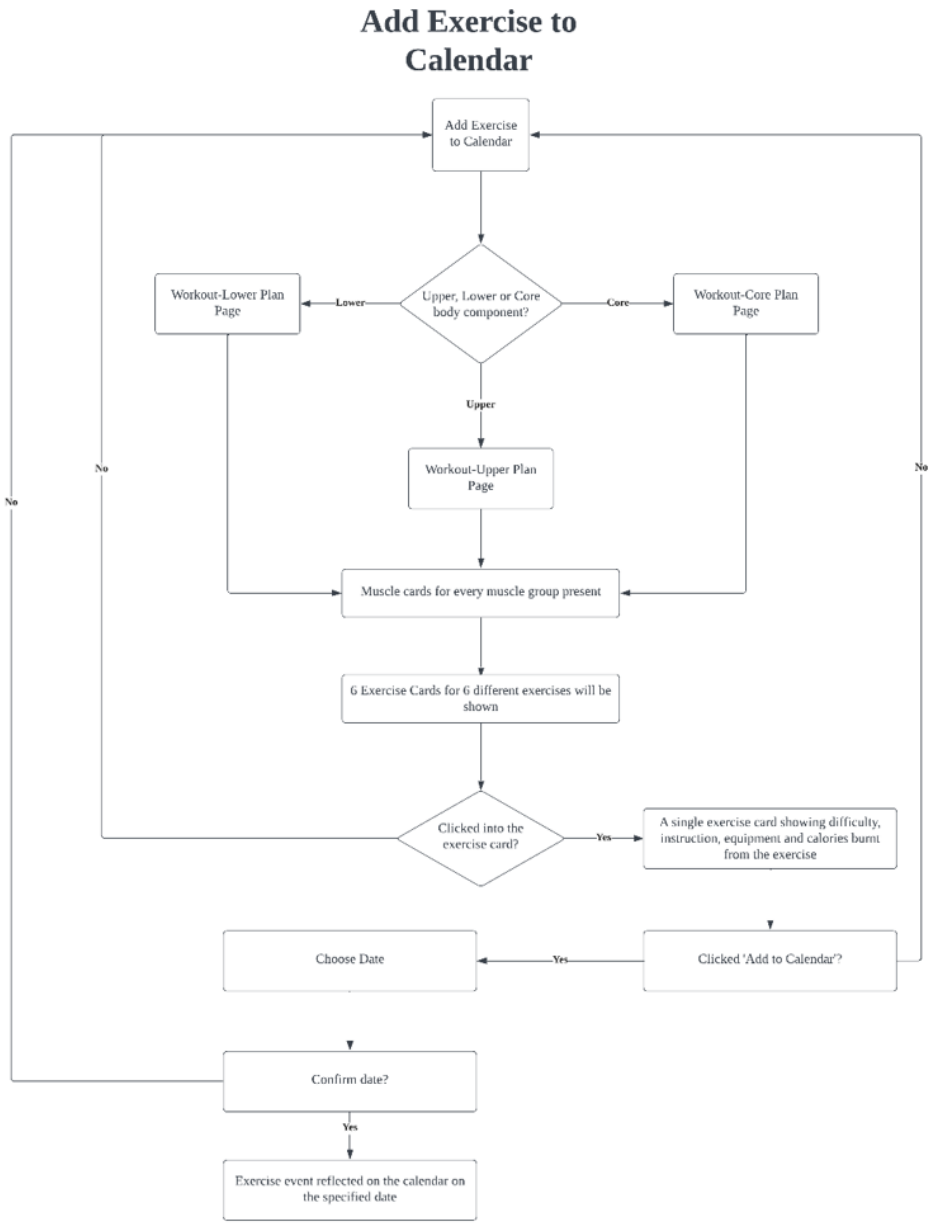
7.2.7. Strava OAuth



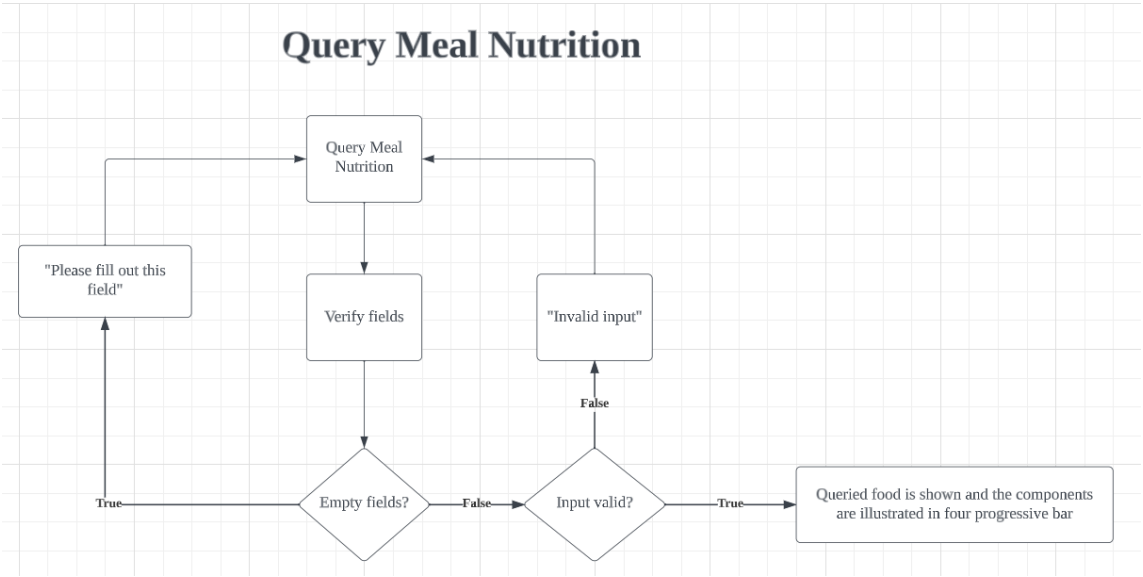
7.2.8. Query Exercise


Query Exercise
Query Exercise
Upper, Lower or Core body component?
Lower
Core
Upper
Workout-Lower Plan Page
Workout-Core Plan Page
Workout-Upper Plan Page
No
Muscle cards for every muscle group present
6 Exercise Cards for 6 different exercises will be shown
Clicked into the exercise card?
Yes
A single exercise card showing difficulty, instruction, equipment and calories burnt from the exercise

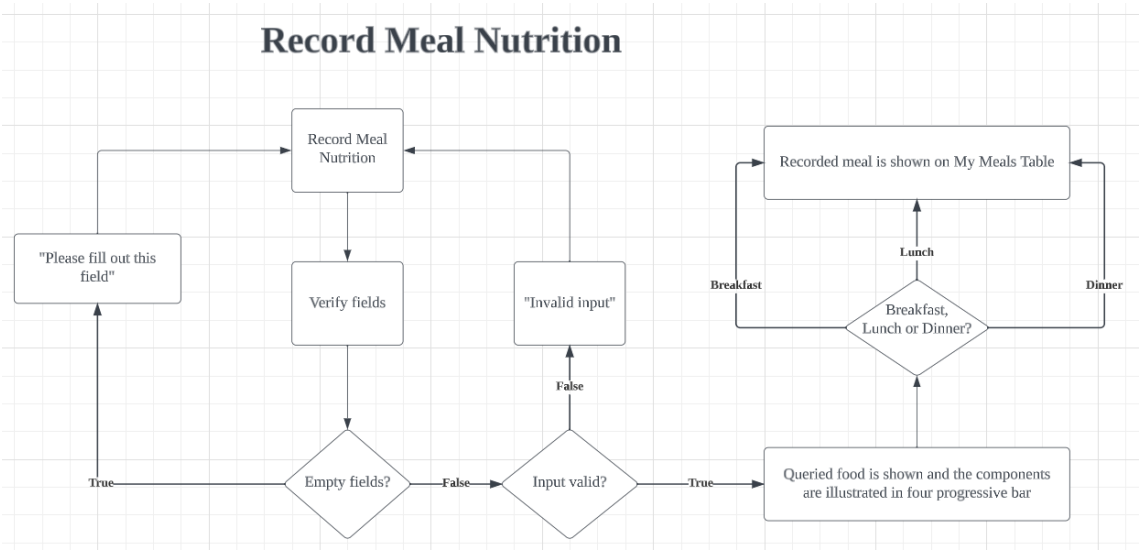
7.2.9. Add Exercise to Calendar



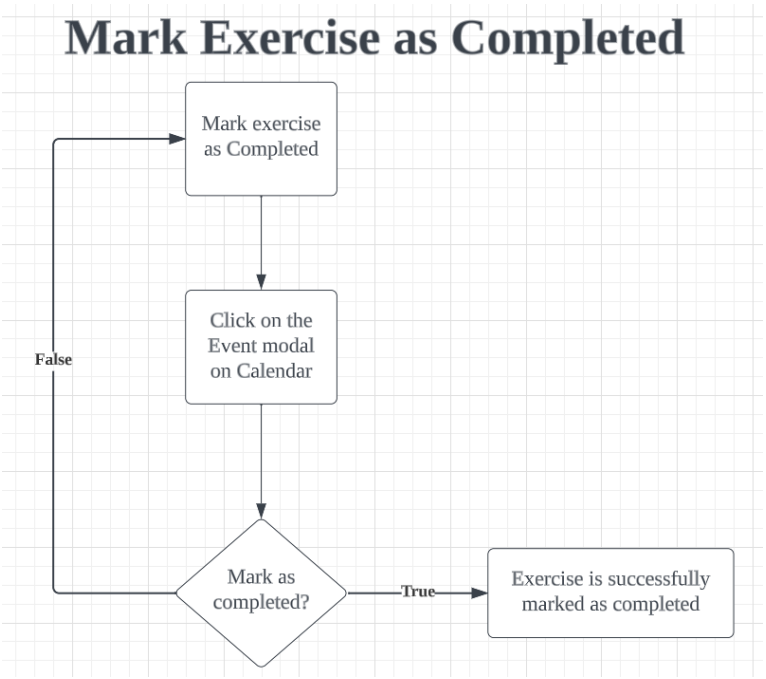
7.2.10. Query Meal Nutrition



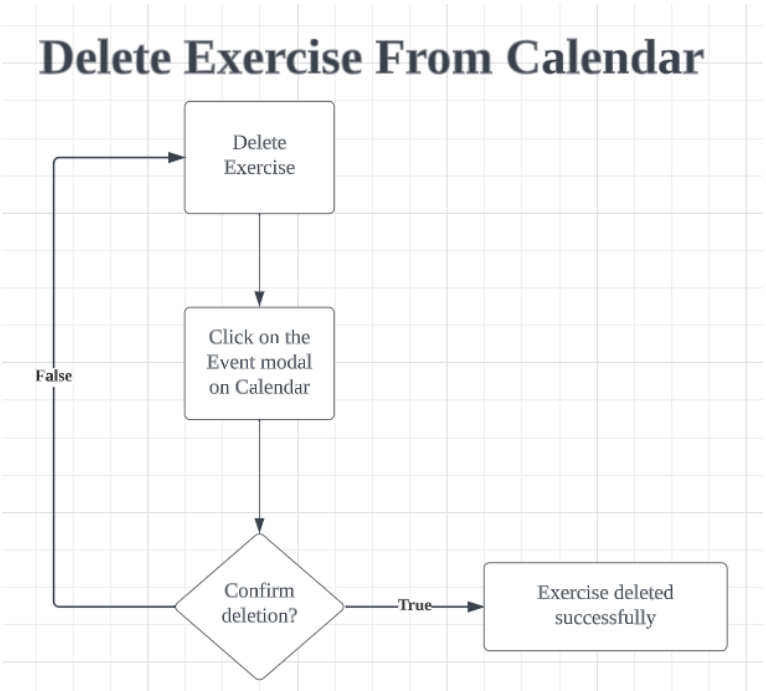
7.2.11. Record Meal Nutrition



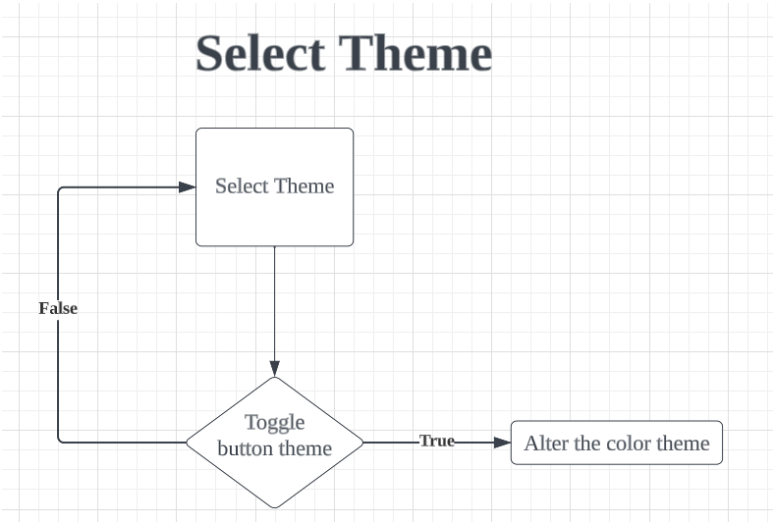
7.2.12. Mark Exercise as Completed



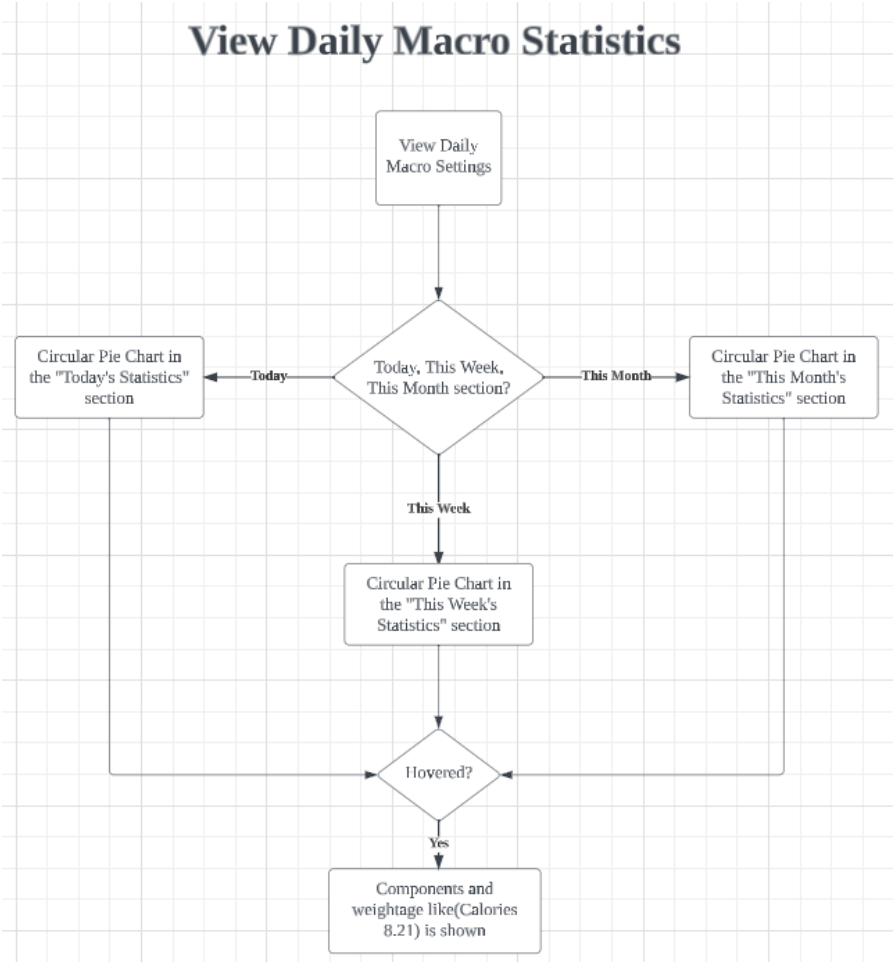
7.2.13. Delete Exercise from Calendar



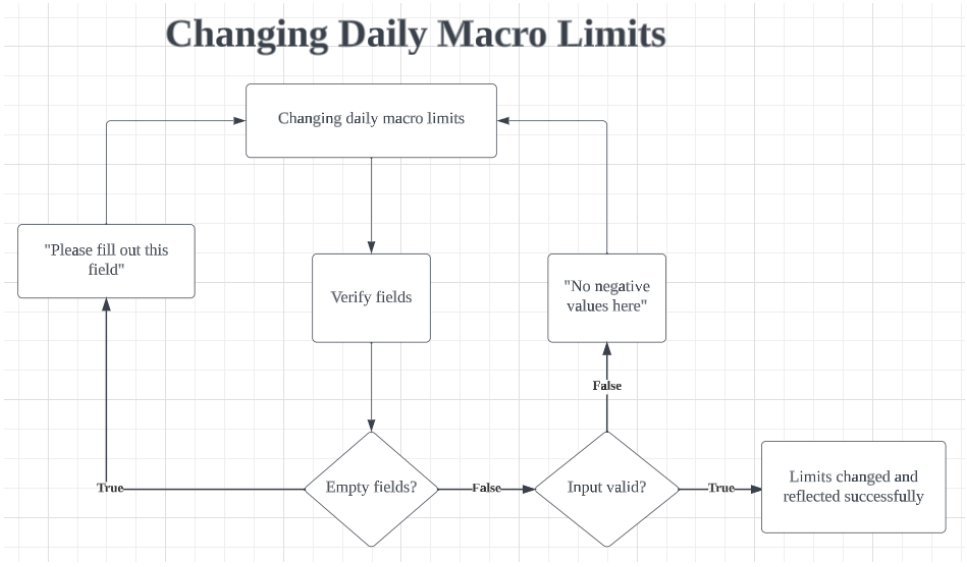
7.2.14. Select Theme



7.2.15. View Daily Macro Statistics


View Daily Macro Statistics
View Daily Macro Settings
Today, This Week, This Month section?
Today
This Month
This Week
Circular Pie Chart in the "Today's Statistics" section
Circular Pie Chart in the "This Week's Statistics" section
Circular Pie Chart in the "This Month's Statistics" section
Hovered?
Yes
Components and weightage like(Calories 8.21) is shown

7.2.16. Changing Daily Macro Limits



7.3. API Testing

The operational smoothness of our web application heavily relies on the efficacy of our RESTful API, which serves as the backbone of our backend. Thorough and comprehensive testing of each API endpoint is paramount to guarantee a seamless and error-free user experience.

In this section, we utilized both unit testing and integrated testing to ensure that our API endpoint is robust. Unit testing ensures that each endpoint is behaving as intended while integrated testing ensures that the API endpoints are performing as expected for specific use cases.

All of our testing for API is done with Postman's API testing tools.

7.3.1. API Unit Testing

7.3.1.1. /api/post

Test Case ID		UT-01		**Date Tested**	01/11/2023	
Endpoint		/api/post				
Endpoint Description		Create a new user inside the database				
Test Parameters						**Result**
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**

1.	POST	nil	{
			username: "testuser1",
			email: "test1@gmail.com",
			password: "qwerty123",
			age: 22,
			weight: 58,
			height: 162
			}
				nil	200
					(OK)
						Pass

2.	POST	nil	{
			username: "",
			email: "",
			password: "",
			age: ,
			weight: ,
			height:
			}
				nil	400
					(Bad request)
						Pass

3.	POST	nil	{
			username: “test1”,
			email: “”,
			password: “”,
			age: 22,
			weight: ,
			height:
			}
				nil	400
					(Bad request)
						Pass

7.3.1.2. /api/post/byGoogle

Test Case ID		UT-02		**Date Tested**	01/11/2023	
Endpoint		/api/post/byGoogle				
Endpoint Description		Create a new user inside the database with google credentials				
Test Parameters						**Result**
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**

1.	POST	nil	{
			username: “testuser1”,
			email: “test1@gmail.com”,
			google_data: { test: test }
			}
				nil	200
					(OK)
						Pass

2.	POST	nil	{
			username: “”,
			email: “”,
			google_data: {}
			}
				nil	400
					(Bad request)
						Pass

3.	POST	nil	{
			username: “testuser1”,
			email: “test1@gmail.com”,
			}
				nil	400
					(Bad request)
						Pass

7.3.1.3. /api/post/byStrava

Test Case ID		UT-03		**Date Tested**	01/11/2023	
Endpoint		/api/post/byStrava				
Endpoint Description		Create a new user inside the database with strava credentials				
Test Parameters						**Result**

No.	Method	Header	Body	Params	Expected Status	Pass/Fail

1.	POST	nil	{
			username: “testuser1”,
			email: “test1@gmail.com”,
			strava_data: { test: test }
			}
				nil	200
					(OK)
						Pass

2.	POST	nil	{
			username: “”,
			email: “”,
			strava_data: {}
			}
				nil	400
					(Bad request)
						Pass

3.	POST	nil	{
			username: “testuser1”,
			email: “test1@gmail.com”,
			}
				nil	400
					(Bad request)
						Pass

7.3.1.4. /api/connectGoogle/:email

Test Case ID	UT-04	Date Tested	01/11/2023
Endpoint	/api/connectGoogle/:email		
Endpoint Description	Update an existing user with new google credentials		

Test Parameters						Result
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**

1.	POST	nil	{
			google_data: { test: test }
			}
				{
				email: “test1@gmail.com”
				}
					200
					(OK)
						Pass

2.	POST	nil	{
			google_data: {}
			}
				{
				email: “test1@gmail.com”
				}
					400
					(Bad request)
						Pass

3.	POST	nil	{
			google_data: { test: test }
			}
				nil	404
					(Not Found)
						Pass

7.3.1.5. /api/connectStrava/:email

Test Case ID	UT-05	Date Tested	01/11/2023
Endpoint	/api/connectStrava/:email		

Endpoint Description	Update an existing user with new strava credentials				
Test Parameters					**Result**
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**
1.	POST	nil	{ strava_data: { test: test } }	{ email: “test1@gmail.com” }	200 (OK)	Pass
2.	POST	nil	{ strava_data: {} }	{ email: “test1@gmail.com” }	400 (Bad request)	Pass
3.	POST	nil	{ strava_data: { test: test } }	nil	404 (Not Found)	Pass

7.3.1.6. /api/updateStravaActivities/:email

Test Case ID	UT-06	**Date Tested**	01/11/2023		
Endpoint	/api/updateStravaActivities/:email				
Endpoint Description	Push new Strava activities into existing user’s data				
Test Parameters					**Result**
No.	**Method**	**Header**	**Body**	**Param**	**Expected Status**	**Pass/Fail**
1.	POST	nil	{ strava_activities: [{ activity1: test}, { activity2: test},] }	{ email: “test1@gmail.com” }	200 (OK)	Pass
2.	POST	nil	{ strava_activities: [] }	{ email: “test1@gmail.com” }	400 (Bad request)	Pass
3.	POST	nil	{ strava_activities: [{ activity1: test}, { activity2: test},] }	nil	404 (Not found)	Pass

7.3.1.7. /api/users

Test Case ID		UT-07		**Date Tested**	01/11/2023	
Endpoint		/api/users				
Endpoint Description		Returns a list of all users in the database				
Test Parameters						**Result**
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**
1.	GET	nil	nil	nil	200 (OK)	Pass

7.3.1.8. /api/user/:username

Test Case ID		UT-08		**Date Tested**	01/11/2023	
Endpoint		/api/user/:username				
Endpoint Description		Return a single user object matched by username				
Test Parameters						**Result**
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**
1.	GET	nil	nil	{ username: “testuser1” }	200 (OK)	Pass
2.	GET	nil	nil	{}	400 (Bad request)	Pass

7.3.1.9. /api/user/getByEmail/:email

Test Case ID		UT-09		**Date Tested**	01/11/2023	
Endpoint		/api/user/getByEmail/:email				
Endpoint Description		Return a single user object matched by email				
Test Parameters						**Result**
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**

1.	GET	nil	nil	{
				email:
				“test1@gmail.com”
				}
					200
					(OK)
						Pass

2.	GET	nil	nil	{}	400 (Bad request)	Pass

7.3.1.10. /api/user/googledata/:gmail

Test Case ID		UT-10		**Date Tested**	01/11/2023	
Endpoint		/api/user/googleData/:gmail				
Endpoint Description		Return a single user object matched by gmail				
Test Parameters						**Result**
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**

1.	GET	nil	nil	{
				gmail:
				“test1@gmail.com”
				}
					200
					(OK)
						Pass

2.	GET	nil	nil	{}	400 (Bad request)	Pass

7.3.1.11. /api/user/stravadata/stravaid

Test Case ID		UT-11		**Date Tested**	01/11/2023	
Endpoint		/api/user/stravadata/:stravaid				
Endpoint Description		Return a single user object matched by strava id				
Test Parameters						**Result**
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**

1.	GET	nil	nil	{
				stravaid: 8562572
				}
					200
					(OK)
						Pass

2.	GET	nil	nil	{}	400 (Bad request)	Pass

7.3.1.12. /api/updateLimits/:email

Test Case ID	UT-12	**Date Tested**	01/11/2023

Endpoint	/api/updateLimits/:email					
Endpoint Description	Updates the macros daily limit of a user matched by email. Returns the updated user object.					
Test Parameters						**Result**
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**

1.	PATCH	nil	{
			calorie: 2500,
			protein: 50,
			carbohydrate: 250,
			fat: 44
			}
				{
				email:
				“test1@gmail.com”
				}
					200
					(OK)
						Pass

2.	PATCH	nil	nil	{
				email:
				“test1@gmail.com”
				}
					400
					(Bad request)
						Pass

3.	PATCH	nil	{
			calorie: 2500,
			protein: 50,
			carbohydrate: 250,
			fat: 44
			}
				nil	404
					(Not found)
						Pass

7.3.1.13. /api/addMeal/:email

Test Case ID	UT-13		**Date Tested**	01/11/2023	
Endpoint	/api/addMeal/:email				
Endpoint Description	Push a new meal into meals array in user matched by email. Returns the updated user object.				
Test Parameters					**Result**
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**

1.	POST	nil	{
			foodName: “chicken”,
			calorie: 100,
			protein: 50,
			carbohydrate: 250,
			fat: 44,
			mealType: “lunch”
			}
				{
				email:
				“test1@gmail.com”
				}
					200
					(OK)
						Pass

2.	POST	nil	nil	{ email: “test1@gmail.com”	400 (Bad request)	Pass

				}		

3.	POST	nil	{
			foodName: “chicken”,
			calorie: 100,
			protein: 50,
			carbohydrate: 250,
			fat: 44,
			mealType: “lunch”
			}
				nil	404
					(Not found)
						Pass

7.3.1.14. /api/deleteMeal/:email

Test Case ID	UT-14		**Date Tested**	01/11/2023	
Endpoint	/api/deleteMeal/:email				
Endpoint Description	Delete a existing meal in the meals array of a user matched by email. Returns the updated user object.				
Test Parameters					**Result**
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**

1.	DELETE	nil	{
			createdAt: Sat Sep 13 275760 00:00:00 GMT+0800 (Coordinated Universal Time)
			}
				{
				email: “test1@gmail.com”
				}
					200
					(OK)
						Pass

2.	DELETE	nil	nil	{
				email: “test1@gmail.com”
				}
					400
					(Bad request)
						Pass

3.	DELETE	nil	{
			createdAt: Sat Sep 13 275760 00:00:00 GMT+0800 (Coordinated Universal Time)
			}
				nil	404
					(Not found)
						Pass

7.3.1.15. /api/addExercise/:email

Test Case ID	UT-15	**Date Tested**	01/11/2023
Endpoint	/api/addExercise/:email		

Endpoint Description	Push a new exercise into workouts array in user matched by email. Returns the updated user object.					
Test Parameters						**Result**
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**

1.	POST	nil	{
			ExerciseData: [
			{ex1: test1},
			{ex2: test2}
			]
			}
				{
				email:
				“test1@gmail.com”
				}
					200
					(OK)
						Pass

2.	POST	nil	{
			ExerciseData: []
			}
				{
				email:
				“test1@gmail.com”
				}
					400
					(Bad request)
						Pass

3.	POST	nil	{
			ExerciseData: [
			{ex1: test1},
			{ex2: test2}
			]
			}
				nil	404
					(Not found)
						Pass

7.3.1.16. /api/updateExercise/:email

Test Case ID	UT-16		**Date Tested**	01/11/2023	
Endpoint	/api/updateExercise/:email				
Endpoint Description	Update isCompleted field in target exercise to “True”. Returns the updated user object.				
Test Parameters					**Result**
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**

1.	PATCH	nil	{
			createdAt: Sat Sep 13 275760 00:00:00 GMT+0800 (Coordinated Universal Time)
			}
				{
				email:
				“test1@gmail.com”
				}
					200
					(OK)
						Pass

2.	PATCH	nil	nil	{
				email:
				“test1@gmail.com”
				}
					400
					(Bad request)
						Pass

3.	PATCH	nil	{
			createdAt: Sat Sep 13 275760 00:00:00 GMT+0800 (Coordinated Universal Time)
			}
				nil	404
					(Not found)
						Pass

7.3.1.17. /api/editExercise/:email

Test Case ID	UT-17		**Date Tested**	01/11/2023	
Endpoint	/api/editExercise/:email				
Endpoint Description	Update name and description field in target exercise. Returns the updated user object.				
Test Parameters					**Result**
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**

1.	PATCH	nil	{
			createdAt: Sat Sep 13 275760 00:00:00 GMT+0800 (Coordinated Universal Time),
			title: “Bench Press”,
			description: “test”
			}
				{
				email: “test1@gmail.com”
				}
					200
					(OK)
						Pass

2.	PATCH	nil	nil	{
				email: “test1@gmail.com”
				}
					400
					(Bad request)
						Pass

3.	PATCH	nil	{
			createdAt: Sat Sep 13 275760 00:00:00 GMT+0800 (Coordinated Universal Time),
			title: “Bench Press”,
			description: “test”
			}
				nil	404
					(Not found)
						Pass

7.3.1.18. /api/deleteExercise/:email

Test Case ID	UT-18	**Date Tested**	01/11/2023

Endpoint	/api/deleteExercise/:email					
Endpoint Description	Delete a existing exercise in the workouts array of a user matched by email. Returns the updated user object.					
Test Parameters						**Result**
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**
1.	DELETE	nil	{ createdAt: Sat Sep 13 275760 00:00:00 GMT+0800 (Coordinated Universal Time) }	{ email: “test1@gmail.com” }	200 (OK)	Pass
2.	DELETE	nil	nil	{ email: “test1@gmail.com” }	400 (Bad request)	Pass
3.	DELETE	nil	{ createdAt: Sat Sep 13 275760 00:00:00 GMT+0800 (Coordinated Universal Time) }	nil	404 (Not found)	Pass

7.3.1.19. /api/updateUserSettings/:email

Test Case ID	UT-19	**Date Tested**	01/11/2023			
Endpoint	/api/updateUserSettings/:email					
Endpoint Description	Update the username, age, weight and height field of target user with new provided values. Returns the updated user object.					
Test Parameters						**Result**
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**
1.	PATCH	nil	{ username: “testuser2”, age: 32, weight: 75, height: 177 }	{ email: “test1@gmail.com” }	200 (OK)	Pass
2.	PATCH	nil	nil	{	400 (Bad request)	Pass

				email: “test1@gmail.com” }		
3.	PATCH	nil	{ username: “testuser2”, age: 32, weight: 75, height: 177 }	nil	404 (Not found)	Pass

7.3.1.20. /api/updateUserPassword/:email

Test Case ID		UT-20		**Date Tested**	01/11/2023	
Endpoint		/api/updateUserPassword/:email				
Endpoint Description		Update the password field of target user with new provided value. Returns the updated user object.				
Test Parameters						**Result**
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**
1.	PATCH	nil	{ password: “321qwerty” }	{ email: “test1@gmail.com” }	200 (OK)	Pass
2.	PATCH	nil	nil	{ email: “test1@gmail.com” }	400 (Bad request)	Pass
3.	PATCH	nil	{ password: “321qwerty” }	nil	404 (Not found)	Pass

7.3.1.21. /api/resetUserPassword/:email

Test Case ID		UT-21		**Date Tested**	01/11/2023	
Endpoint		/api/resetUserPassword/:email				
Endpoint Description		Finds the user matched by email, then create a short-lived Token in the database. Returns the newly created Token object.				
Test Parameters						**Result**
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**

1.	GET	nil	nil	{
				email:
				“test1@gmail.com”
				}
					200
					(OK)
						Pass

2.	GET	nil	nil	nil	404 (Not found)	Pass

7.3.1.22. /api/getToken/:token

Test Case ID		UT-22		**Date Tested**	01/11/2023	
Endpoint		/api/getToken/:token				
Endpoint Description		Returns the Token object matched by token param.				
Test Parameters						**Result**
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**

1.	GET	nil	nil	{
				token:
				“d8asd8as-1j2nnjdjas-8989jd2j-dash8h1dj”
				}
					200
					(OK)
						Pass

2.	GET	nil	nil	nil	404 (Not found)	Pass

7.3.1.23. /api/getUserByOId/:id

Test Case ID		UT-23		**Date Tested**	01/11/2023	
Endpoint		/api/getUserByOId/:id				
Endpoint Description		Returns user matched by Object ID				
Test Parameters						**Result**
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**

1.	GET	nil	nil	{
				id:
				“652caacf9fd7c8d6bae0a39e”
				}
					200
					(OK)
						Pass

2.	GET	nil	nil	nil	404 (Not found)	Pass

7.3.1.24. /api/resetPassword/:id

Test Case ID	UT-24	**Date Tested**	01/11/2023
Endpoint	/api/resetPassword/:id		
Endpoint Description	Find user with ObjectId, then updates password field of user. Delete existing short-lived token in database. Returns updated user object.		

Test Parameters						**Result**
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**
1.	POST	nil	{ newPassword: “qwerty”, token: “njiefn9-sdf9n23-m1indj-amnsdoi" }	{ id: “652caacf9fd7c8d6bae0a39e” }	200 (OK)	Pass
2.	POST	nil	nil	{ id: “652caacf9fd7c8d6bae0a39e” }	400 (Bad request)	Pass
3.	POST	nil	nil	nil	404 (Not found)	Pass

7.3.1.25. /api/updateProfilePic/:email

Test Case ID	UT-25	**Date Tested**	01/11/2023
Endpoint	/api/updateProfilePic/:email		
Endpoint Description	Updates profilePic field of user matched by email. Returns updated user object.		

Test Parameters						**Result**
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**
1.	POST	nil	{ profilepic: “…” }	{ email: “test1@gmail.com” }	200 (OK)	Pass
2.	POST	nil	nil	{ email: “test1@gmail.com” }	400 (Bad request)	Pass

3.	POST	nil	nil	nil	404 (Not found)	Pass

7.3.1.26. /vifitness/initialiseToken

Test Case ID		UT-26		**Date Tested**	01/11/2023	
Endpoint		/vifitness/initialiseToken				
Endpoint Description		Updates client Id and client secret field in user matched by email. Returns updated user object. This endpoint is for external use.				
Test Parameters						**Result**
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**

1.	POST	nil	{
			email: “test1@gmail.com”
			}
				nil	200
					(OK)
						Pass

2.	POST	nil	nil	nil	401 (Bad request)	Pass

7.3.1.27. /vifitness/v1/getUserWorkouts

Test Case ID		UT-27		**Date Tested**	01/11/2023	
Endpoint		/vifitness/v1/getUserWorkouts				
Endpoint Description		Returns all the workout of a user matched by clientID. This endpoint is for external use.				
Test Parameters						**Result**
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**

1.	GET	{
		clientid:
		“652caacf9fd7c8d6bae0a39e”,
		clientsecret:
		“7e2b0eb6be145a932c57d38582ded7be23d1af728fdb779b878db3386464b9ae”
		}
			nil	nil	200
					(OK)
						Pass

2.	GET	nil	nil	nil	401 (Unauthorised)	Pass

7.3.1.28. /vifitness/v1/getUserMeals

Test Case ID		UT-28		**Date Tested**	01/11/2023	
Endpoint		/vifitness/v1/getUserMeals				
Endpoint Description		Returns all the meals of a user matched by clientID. This endpoint is for external use.				
Test Parameters						**Result**
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**
1.	GET	{ clientid: “652caacf9fd7c8d6bae0a39e”, clientsecret: “7e2b0eb6be145a932c57d38582ded7be23d1af728fdb779b878db3386464b9ae” }	nil	nil	200 (OK)	Pass
2.	GET	nil	nil	nil	401 (Unauthorised)	Pass

7.3.1.29. /vifitness/v1/getUserStravaActivities

Test Case ID		UT-29		**Date Tested**	01/11/2023	
Endpoint		/vifitness/v1/getUserStravaActivities				
Endpoint Description		Returns all the strava activities of a user matched by clientID. This endpoint is for external use.				
Test Parameters						**Result**
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**
1.	GET	{ clientid: “652caacf9fd7c8d6bae0a39e”, clientsecret: “7e2b0eb6be145a932c57d38582ded7be23d1af728fdb779b878db3386464b9ae” }	nil	nil	200 (OK)	Pass
2.	GET	nil	nil	nil	401 (Unauthorised)	Pass

7.3.1.30. /vifitness/v1/getUserInfo

Test Case ID		UT-30		**Date Tested**	01/11/2023	
Endpoint		/vifitness/v1/getUserInfo				
Endpoint Description		Returns informations of a user matched by clientID. This endpoint is for external use.				
Test Parameters						**Result**
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**
1.	GET	{ clientid: “652caacf9fd7c8d6bae0a39e”, clientsecret: “7e2b0eb6be145a932c57d38582ded7be23d1af728fdb779b878db3386464b9ae” }	nil	nil	200 (OK)	Pass
2.	GET	nil	nil	nil	401 (Unauthorised)	Pass

7.3.1.31. /vifitness/v1/addMeal

Test Case ID		UT-31		**Date Tested**	01/11/2023	
Endpoint		/vifitness/v1/addMeal				
Endpoint Description		Add a meal to meals array of user matched by client ID. This endpoint is for external use.				
Test Parameters						**Result**
No.	**Method**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**
1.	POST	{ clientid: “652caacf9fd7c8d6bae0a39e”, clientsecret: “7e2b0eb6be145a932c57d38582ded7be23d1af728fdb779b878db3386464b9ae” }	{ foodName: “chicken”, calorie: 100, protein: 50, carbohydrate: 250, fat: 44, mealType: “lunch” }	nil	200 (OK)	Pass

2.	POST	nil	{
			foodName: “chicken”,
			calorie: 100,
			protein: 50,
			carbohydrate: 250,
			fat: 44,
			mealType: “lunch”
			}
				nil	401
					(Unauthorised)
						Pass

3.	POST	{
		clientid: “652caacf9fd7c8d6bae0a39e”,
		clientsecret: “7e2b0eb6be145a932c57d38582ded7be23d1af728fdb779b878db3386464b9ae”
		}
			nil	nil	400
					(Bad request)
						Pass

7.3.2. Integrated API Testing

7.3.2.1. Update Daily Nutrition Limit and Add Meal

Test Case ID		IT-01		**Date Tested**	01/11/2023	
Test Case Description		Activity flow for a user adding a meal				
Test Parameters						**Result**
No.	**Endpoint**	**Method**	**Body**	**Params**		**Pass/Fail**

1.	/api/post	POST	{
			username: “testuser1”,
			email: “test1@gmail.com”,
			password: “qwerty123”,
			age: 22,
			weight: 58,
			height: 162
			}
				nil		Pass

2.	/api/updateLimits/:email	PATCH	{
			calorie: 2500,
			protein: 50,
			carbohydrate: 250,
			fat: 44
			}
				{
				email: “test1@gmail.com”
				}
					Pass

3	/api/addMeal/:email	POST	{
			foodName: “chicken”,
			calorie: 100,
			protein: 50,
			carbohydrate: 250,
			fat: 44,
			mealType: “lunch”
			}
				{
				email: “test1@gmail.com”
				}
					Pass

4	/api/deleteMeal/:email	DELETE	{
			CreatedAt:
			variable.createdAt
			}
				{
				email: “test1@gmail.com”
				}
					Pass

5.	/api/delete/:email	DELETE	nil	{
				email: “test1@gmail.com”
				}
					Pass

7.3.2.2. Add Exercise and related activities

Test Case ID		IT-02		**Date Tested**	01/11/2023
Test Case Description		Activity flow for a user adding a workout and its related user activities.			
Test Parameters					**Result**
No.	**Endpoint**	**Method**	**Body**	**Params**	**Pass/Fail**

1.	/api/post	POST	{
			username: “testuser1”,
			email:
			“test1@gmail.com”,
			password: “qwerty123”,
			age: 22,
			weight: 58,
			height: 162
			}
				nil	Pass

2.	/api/addExercise/:email	POST	{
			exerciseData: [
			{
			name: “Bench Press”,
			isCompleted: false,
				{
				email: “test1@gmail.com”
				}
					Pass

			calories: 87.5, createdAt: new Date() },] }		

3	/api/updateExercise/:email	PATCH	{
			createdAt:
			variable.createdAt
			}
				{
				email: “test1@gmail.com”
				}
					Pass

4.	/api/editExercise/:email	PATCH	{
			createdAt:
			variable.createdAt,
			title: “Plank”,
			description: “test”
			}
				{
				email: “test1@gmail.com”
				}
					Pass

5.	/api/deleteExercise/:email	DELETE	{
			createdAt:
			variable.createdAt
			}
				{
				email:
				“test1@gmail.com”
				}
					Pass

6.	/api/delete/:email	DELETE	nil	{
				email: “test1@gmail.com”
				}
					Pass

7.3.2.3. Account Creation and Profile Settings Related Activities

Test Case ID		IT-03		**Date Tested**	01/11/2023
Test Case Description		Activity flow for account creation and profile settings related activities			
Test Parameters					**Result**
No.	**Endpoint**	**Method**	**Body**	**Params**	**Pass/Fail**

1.	/api/post	POST	{
			username: “testuser1”,
			email:
			“test1@gmail.com”,
			password: “qwerty123”,
			age: 22,
			weight: 58,
			height: 162
			}
				nil	Pass

2.	/api/connectGoogle/:email	POST	{
			google_data: { test: test }
			}
				{
				email: “test1@gmail.com”
				}
					Pass

3	/api/connectStrava/:email	POST	{
			strava_data: { test: test }
			}
				{
				email: “test1@gmail.com”
				}
					Pass

4.	/api/updateUserSettings/:email	PATCH	{
			username: “testuser2”,
			age: 32,
			weight: 75,
			height: 177
			}
				{
				email: “test1@gmail.com”
				}
					Pass

5.	/api/updateUserPassword/:email	PATCH	{
			password: “321qwerty”
			}
				{
				email:
				“test1@gmail.com”
				}
					Pass

6.	/api/delete/:email	DELETE	nil	{
				email: “test1@gmail.com”
				}
					Pass

7.3.2.4. Strava Integration and Import Strava Activities

Test Case ID		IT-04		**Date Tested**	01/11/2023
Test Case Description		Activity flow for strava integrations and import strava activities			
Test Parameters					**Result**
No.	**Endpoint**	**Method**	**Body**	**Params**	**Pass/Fail**

1.	/api/post	POST	{
			username: “testuser1”,
			email:
			“test1@gmail.com”,
			password: “qwerty123”,
			age: 22,
			weight: 58,
			height: 162
			}
				nil	Pass

2	/api/connectStrava/:email	POST	{
			strava_data: { test: test }
			}
				{
				email: “test1@gmail.com”
				}
					Pass

3.	/api/updateStravaActivities/:email	POST	{
			strava_activities: [
			{ activity1: test},
			{ activity2: test},
				{
				email: “test1@gmail.com”
				}
					Pass

			] }		

4.	/api/delete/:email	DELETE	nil	{
				email: “test1@gmail.com”
				}
					Pass

7.3.2.5. Forget Password and Reset Password

Test Case ID		IT-04		**Date Tested**	01/11/2023
Test Case Description		Activity flow for strava integrations and import strava activities			
Test Parameters					**Result**
No.	**Endpoint**	**Method**	**Body**	**Params**	**Pass/Fail**

1.	/api/post	POST	{
			username: “testuser1”,
			email:
			“test1@gmail.com”,
			password: “qwerty123”,
			age: 22,
			weight: 58,
			height: 162
			}
				nil	Pass

2	/api/resetUserPassword/:email	GET	nil	{
				email: “test1@gmail.com”
				}
					Pass

3.	/api/getToken/:token	GET	{
			strava_activities: [
			{ activity1: test},
			{ activity2: test},
			]
			}
				{
				token: variable.token
				}
					Pass

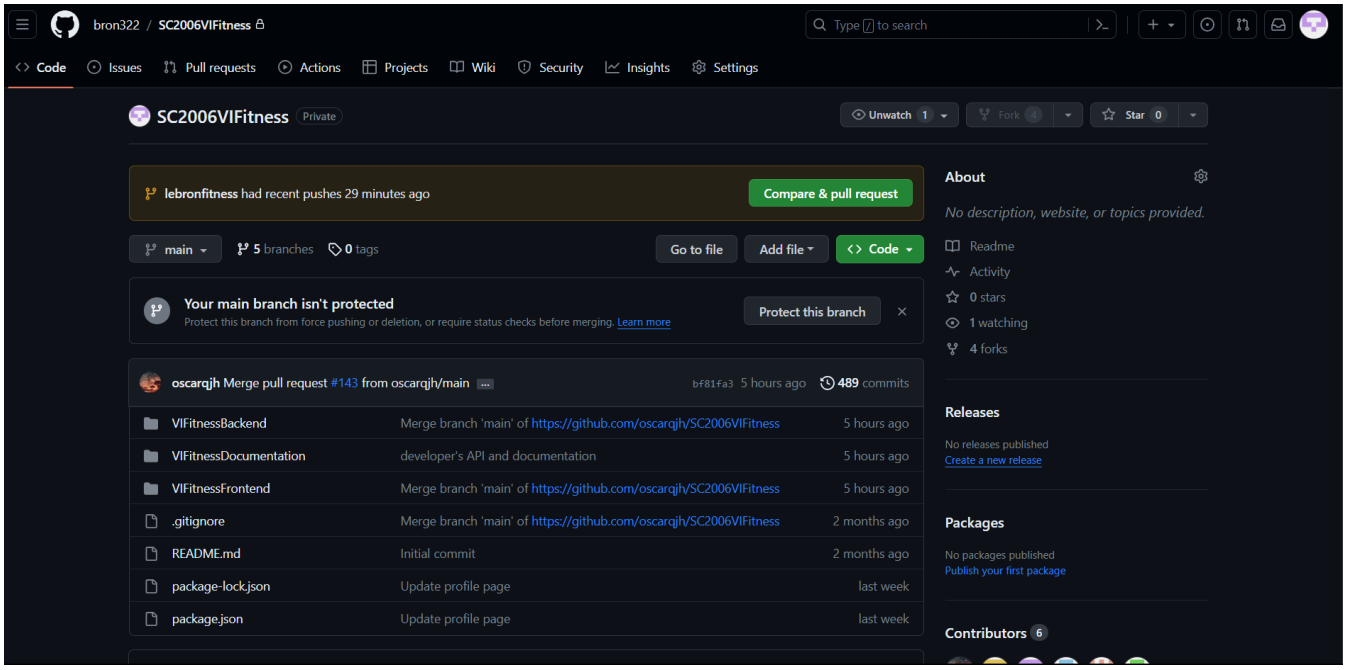
	/api/resetPassword/:id	POST	{
			newPassword: “qwerty”,
			token: variable.token
			}
				{
				id: variable.id
				}
					Pass

4.	/api/delete/:email	DELETE	nil	{
				email: “test1@gmail.com”
				}
					Pass

8. Software Engineering Practices

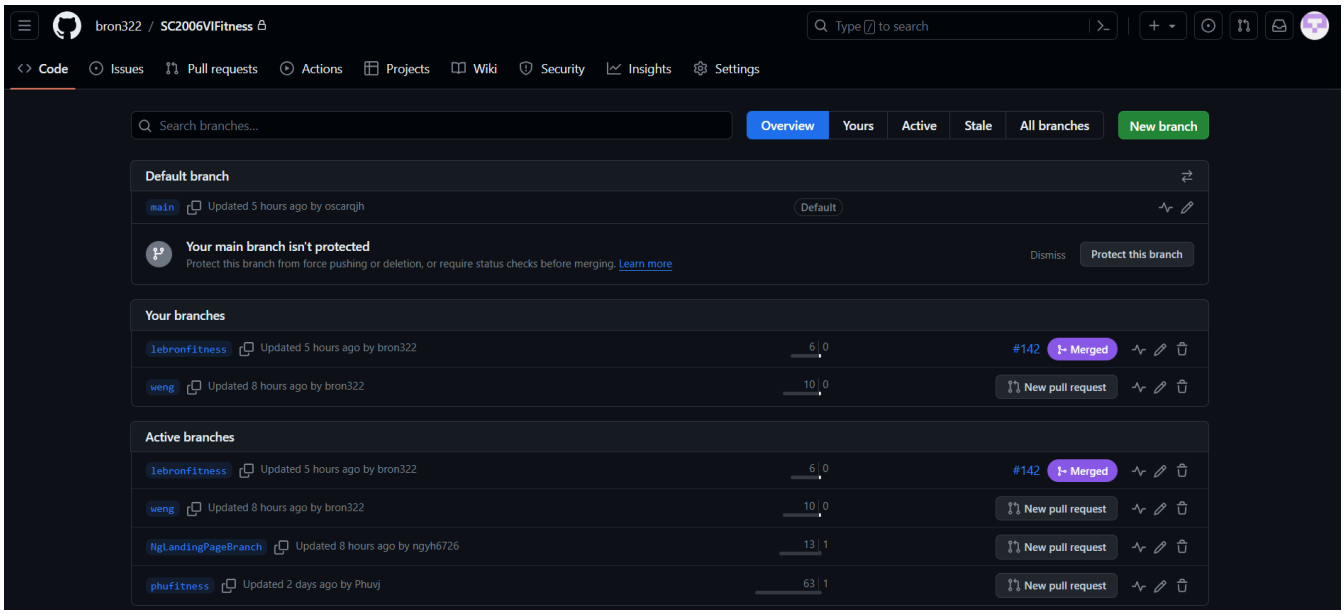
8.1. Version Control

8.1.1. Platform



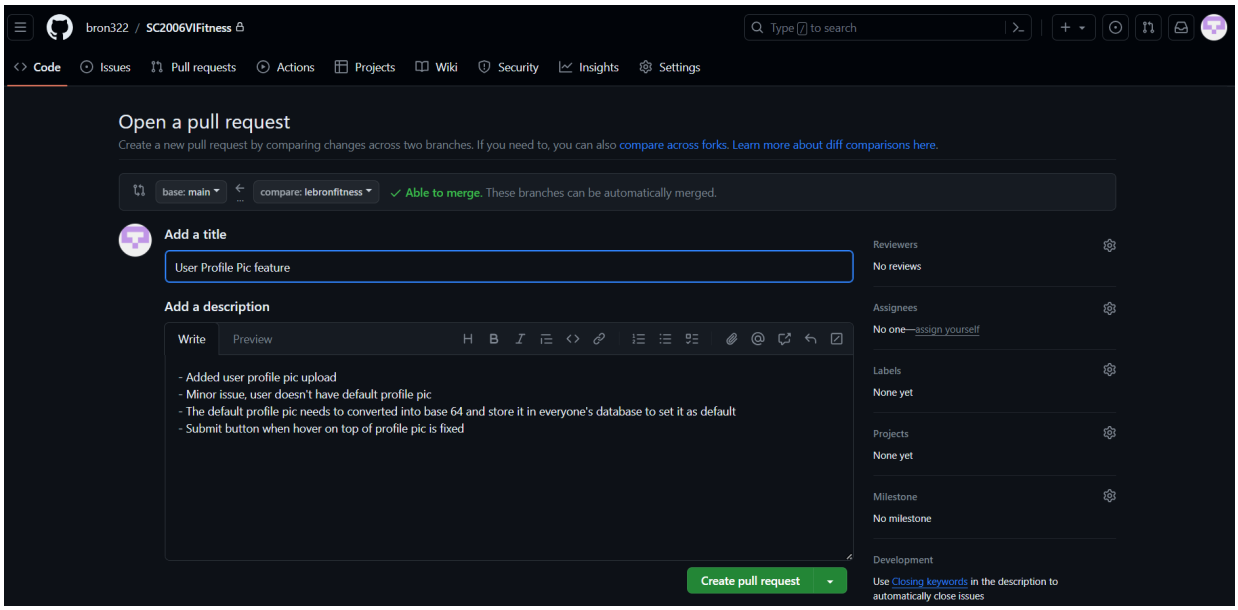
The primary revision control tool our team utilizes is Git, and GitHub acts as our main cloud-based repository for the project. Leveraging GitHub's capabilities to store, oversee, and trace modifications in project code, we've organized our code management and version control processes. After considering different collaboration workflows with Git, such as the Forking Workflow and Centralized Workflow, our team decided to embrace the Git Feature Branch Workflow (which will be explained in 8.1.2) due to the project's scope and the collaborative pair programming structure.

8.1.2. Branching



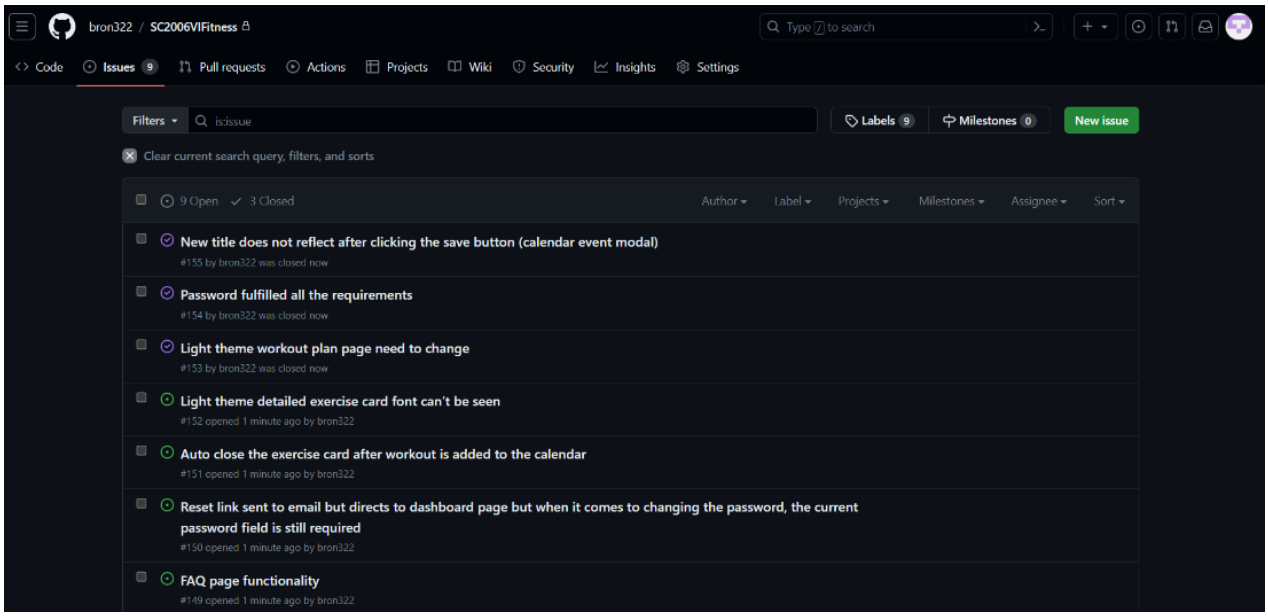
The main branch, referred to as "main," consistently contains stable and functional code. When working on new features, we fetch the latest code from the main branch and establish a distinct branch, usually named after the feature and developer. Using separate branches ensures that code remains isolated, reducing complexities during the debugging phase. After completing the feature, the modifications are committed to the feature branch, followed by a pull request to merge them into the main branch. This methodology ensures a methodical and structured approach to integrating new changes into the codebase.

8.1.3. Pull Request



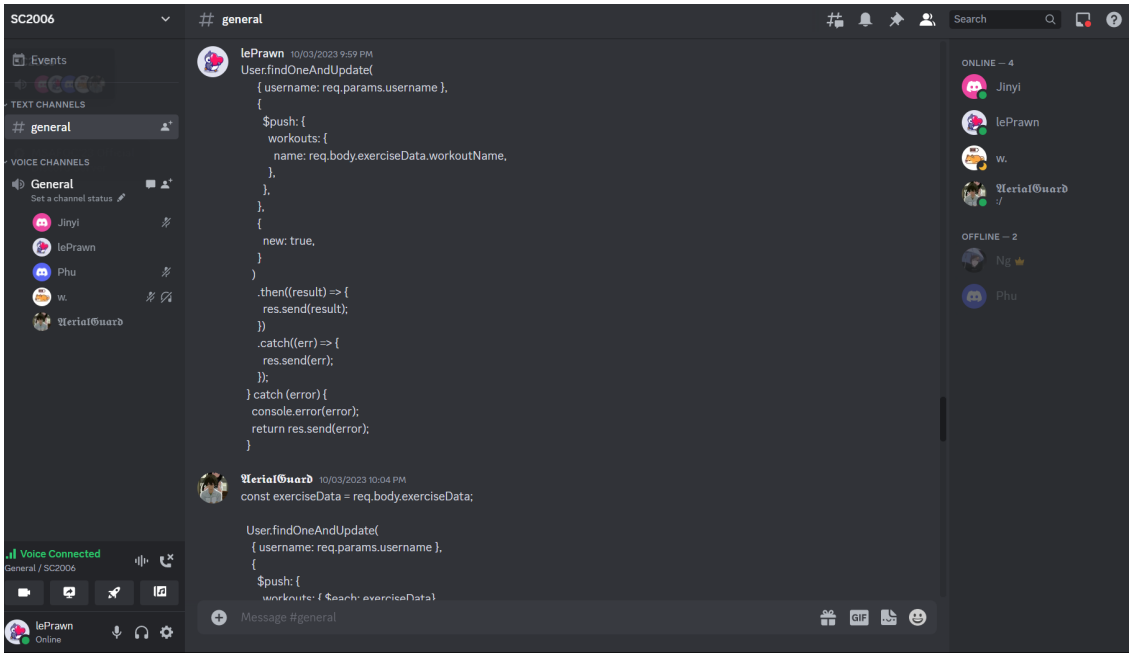
When making updates to the main branch in Git, the pull request functionality comes in handy. The developer responsible for a specific feature initiates a pull request and designates their teammates as the code reviewer. This approach encourages efficient communication and ensures proper resolution of merge conflicts, preventing the introduction of unnecessary bugs. The pull request is deemed finished only after the code undergoes review, receives approval, and any conflicts are addressed. Adhering to this process helps uphold code quality and mitigates the risk of errors going unnoticed.

8.1.4. Git issues



Git Issues functionality was widely used in our group to monitor any ongoing issues or existing bugs within the application. Tags were utilized to categorize the nature of the issue, whether it involved fixes or potential enhancements. The issue would only be marked as closed once the problem had been successfully addressed.

8.1.5. Weekly Sprints



To maintain a consistent workflow and manage merge conflicts effectively, we organize weekly sprints. These sprints act as regular milestones to review our ongoing progress and address any challenges arising from merge conflicts. This iterative method allows us to closely monitor the development process, detect conflicts at an early stage, and promptly find resolutions. The weekly sprints foster efficient collaboration and establish a well-defined structure for handling merge conflicts, leading to seamless code integration and reducing potential delays in our development workflow.

VIFitness' SCRUM sprint log can be found in the pdf documents attached, or can be accessed via https://drive.google.com/file/d/16q23XlMh9j91szRvuU1wxjvhrwe3T_9r/view?usp=sharing

8.2. Software Development Life Cycle (SDLC) Processes

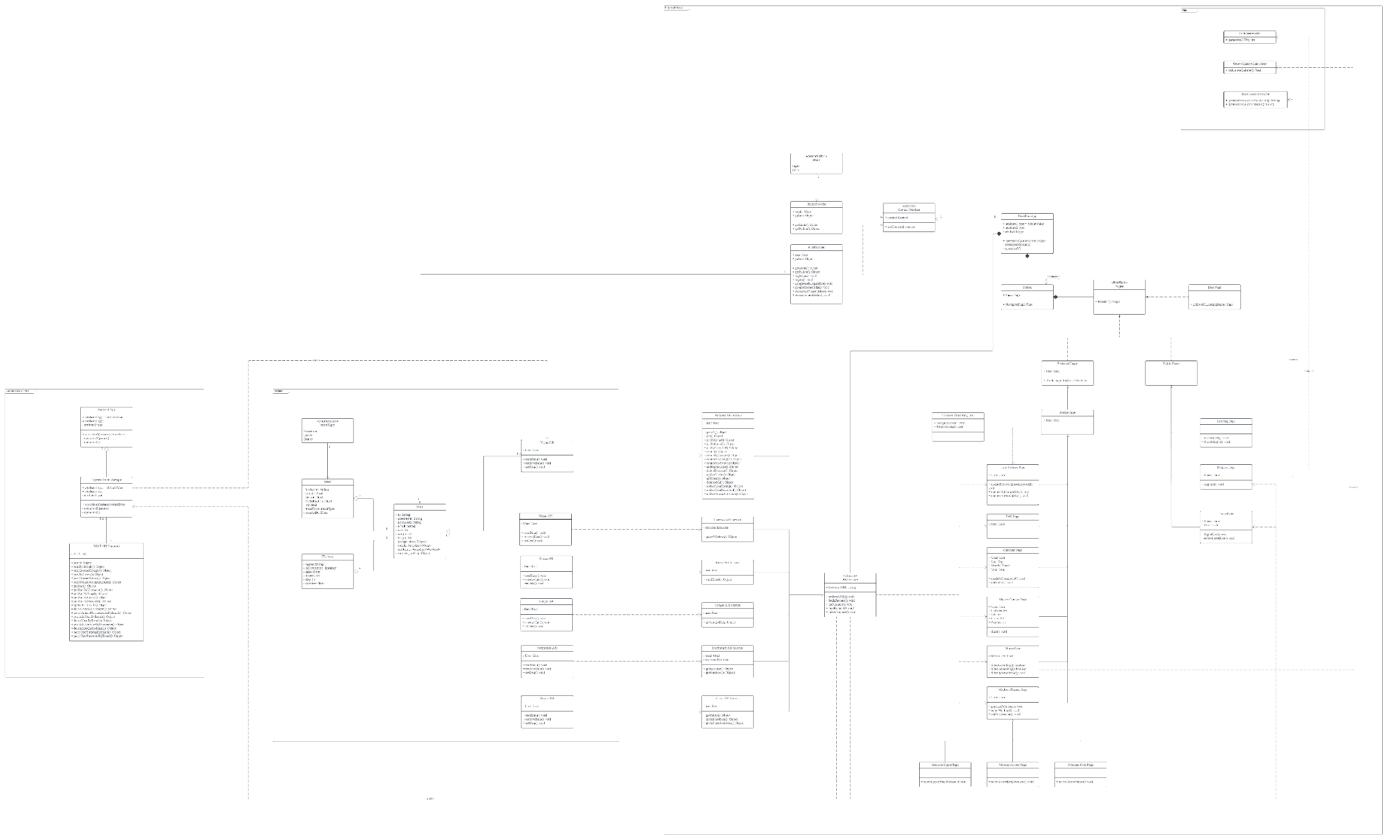
9. Appendix A: Data Dictionary

Term	Definition
User	The end user who has a registered account and is using services provided by VIFitness web application
VIFitness	Name of the web application in this project
Strava	Strava is an American internet service for tracking physical exercise which incorporates social network features. It started out tracking mostly outdoor cycling and running activities using Global Positioning System data, but now incorporates several dozen other exercise types, including indoor activities
Strava Activities	Activities posted by the user on their Strava account. These activities will be fetched into VIFitness's database with the permission of the user for the purpose of tracking caloric expenses of a user.
Account	A user account is comprised of a username, password and any information. related to the user, which is stored inside VIFitness's database, enabling the user to access various features of the web application and deposit information into the database.
Username	An alias given to a user's account
Password	A string of characters (letters, numbers, special symbols) that allow the user to access their account
OAuth 2.0	OAuth 2.0 is the industry-standard authorization protocol that enhances security by enabling user authentication without the necessity of storing passwords in our database. Instead, passwords are securely managed by external companies like Google or Twitter.
Macros tracker and planner	A feature in the web application that allows users to plan their meals and track nutritional and caloric intake of their meals
Nutritional Values	Includes information such as Calorie, saturated fat, trans fat, poly saturated fat, monounsaturated fat, cholesterol, sodium, dietary fiber, sugars, protein, vitamins, calcium, iron, potassium, caffeine and other micronutrients
Meal query	Users enter the description of a meal e.g., "a set of chicken chop with fries and salad". Nutritional value information will be return through API request.
Workout planner	A feature in the web application that allows users to plan their workout schedule on a calendar

10. Appendix B: Analysis Models

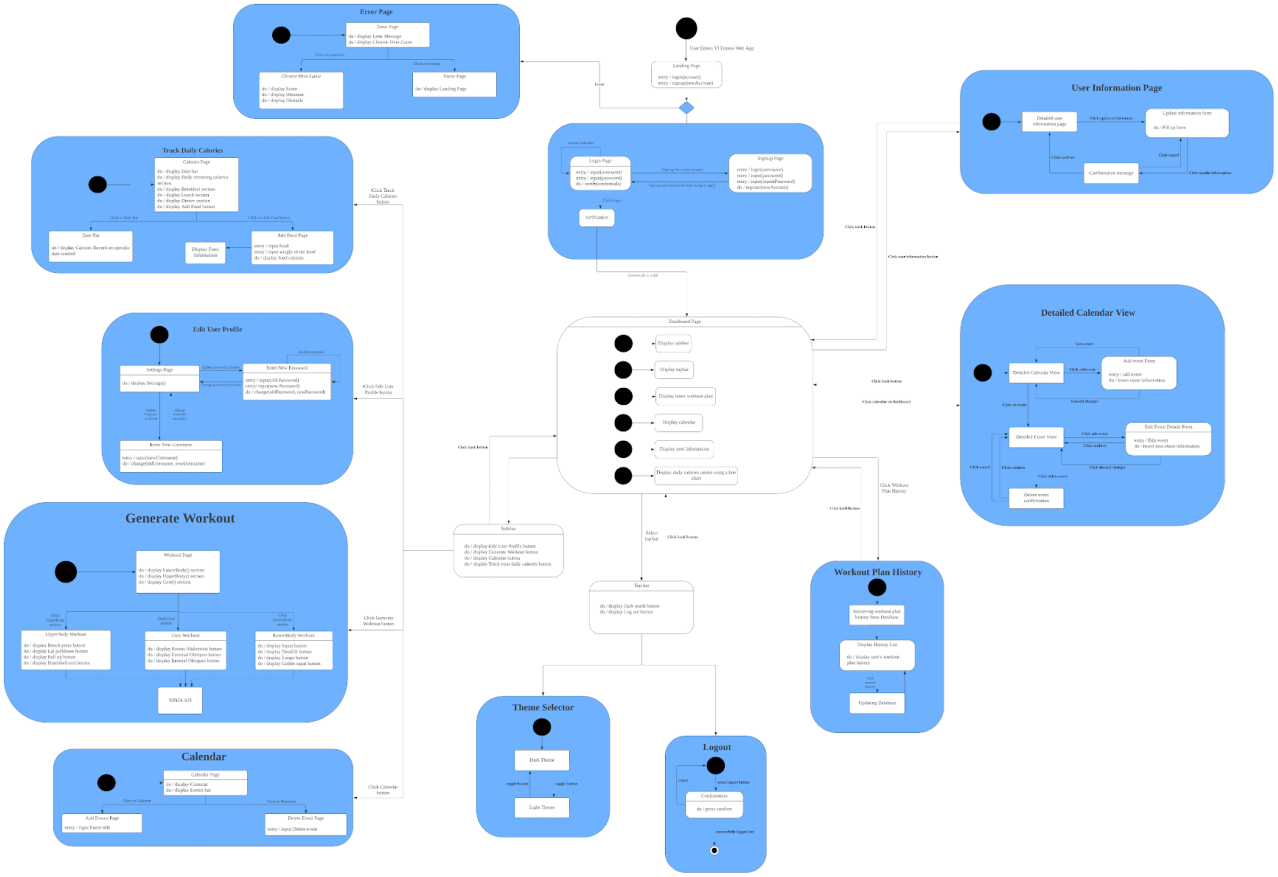
10.1. B1. Class Diagram

High-definition image can be accessed via
https://drive.google.com/file/d/1IK4YrUrANZmjCRWfpqB9xCGC3F26EoPT/view?usp=sharing



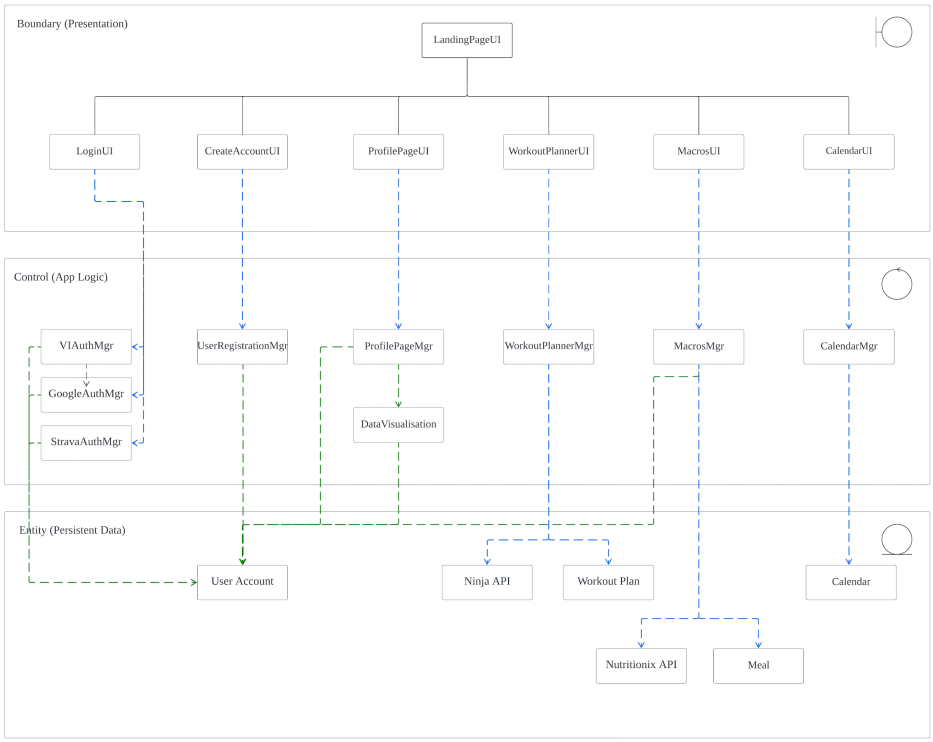
10.2. B2. State Machine Diagram

High-definition image can be accessed via
https://drive.google.com/file/d/1WqHZsLB6zSz0ANY6fRRpCCdDTgeJ33pT/view?usp=sharing



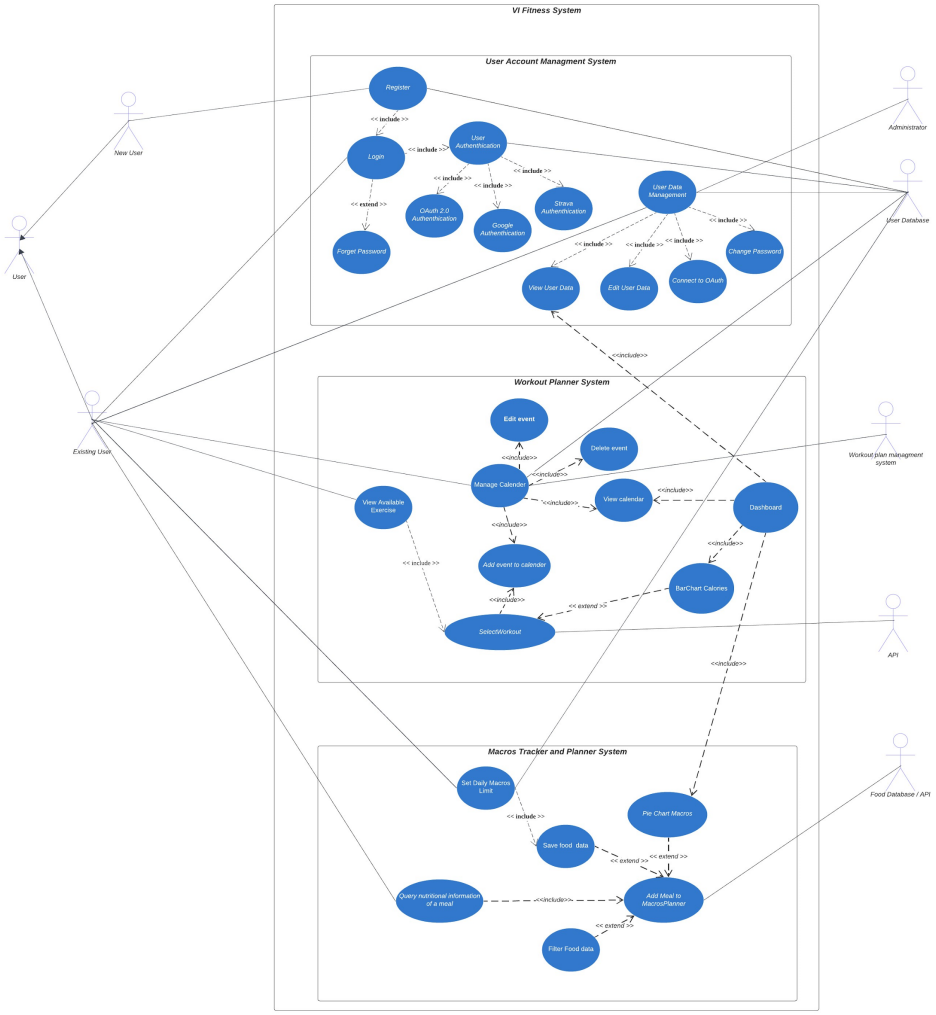
10.3. B3. System Architecture Diagram

High-definition image can be accessed via
https://drive.google.com/file/d/1lwXWMxWci36sJjH_MsV1vSP2elptj36e/view?usp=sharing



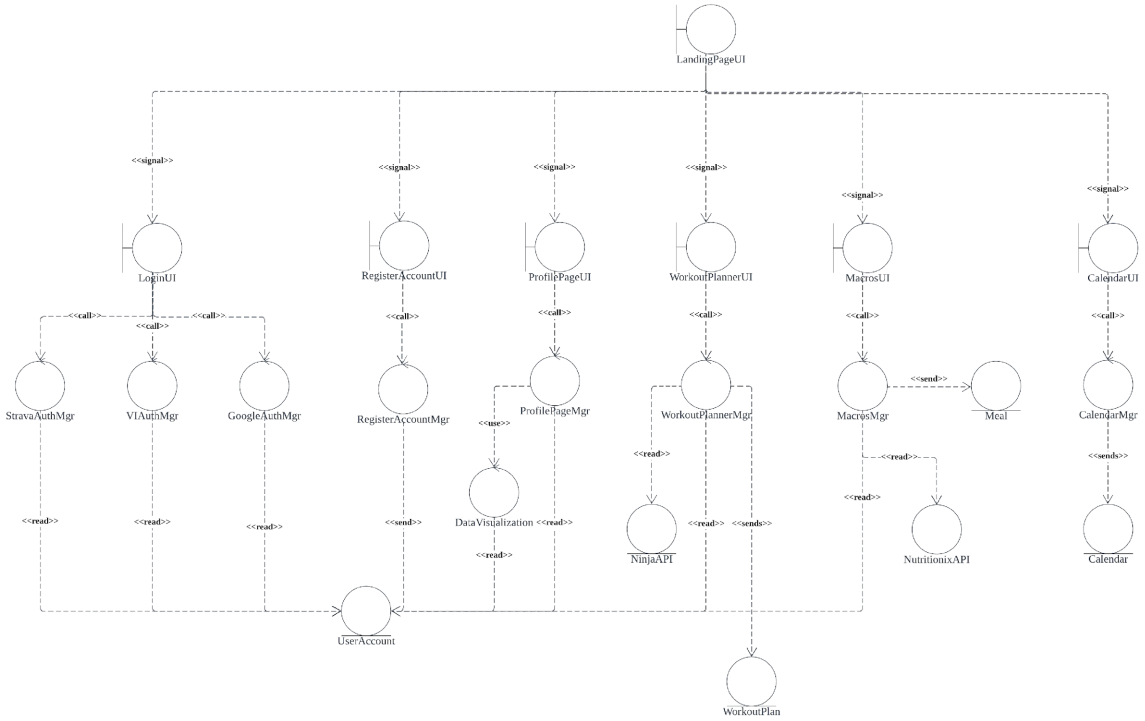
10.4. B4. Use Case Diagram

High-definition image can be accessed via
https://drive.google.com/file/d/1LOcPhRIACv8RoyL3vl189efp7gn7xlzS/view?usp=sharing



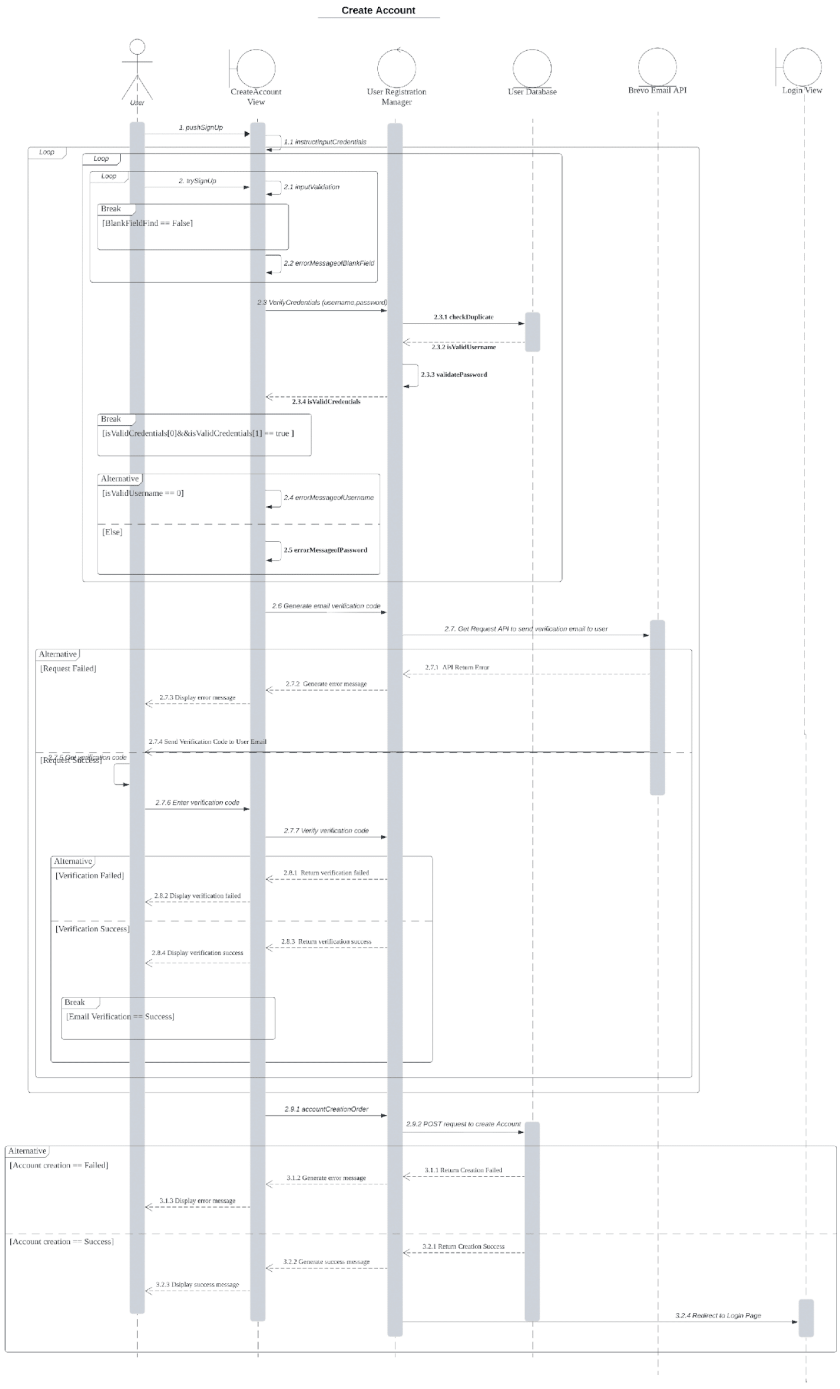
10.5. B5. Key Boundary Class Diagram

High-definition image can be accessed via
https://drive.google.com/file/d/1xGdAJEZ9qKz2xFGrZ-WczFJ68bTULSCq/view?usp=sharing



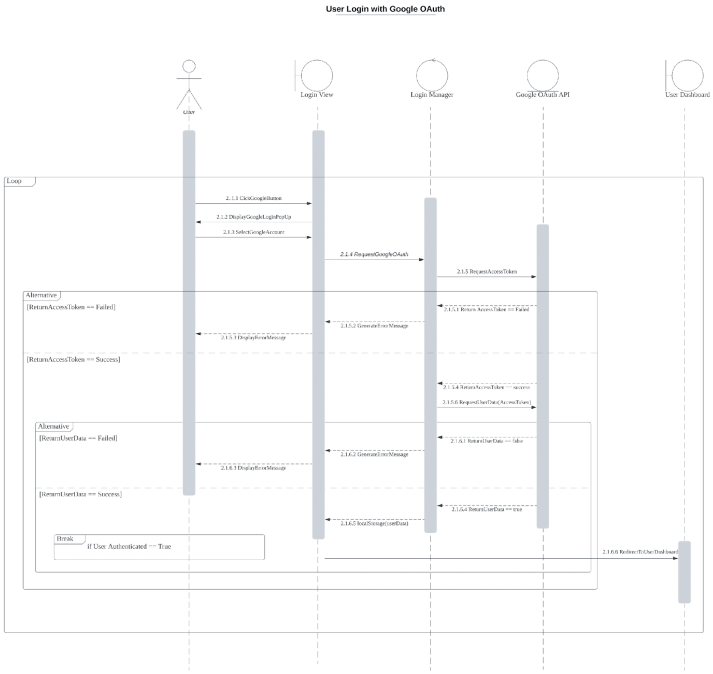
10.6. B6. Register Account Sequence Diagram

High-definition image can be accessed via
https://drive.google.com/file/d/1xc1EVbCsmFG5MKPS_dHiEelMfw5yL9Fi/view?usp=sharing



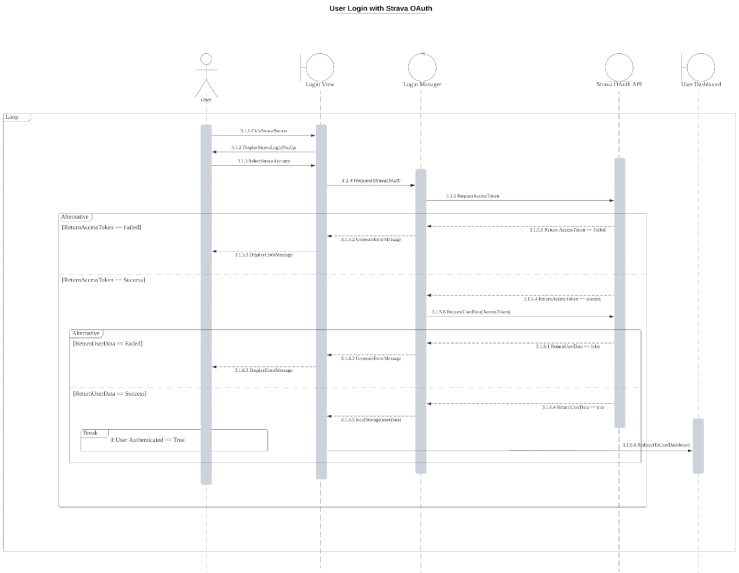
10.7. B7. Google OAuth Sequence Diagram

High-definition image can be accessed via
https://drive.google.com/file/d/1IM7s9iIzCQYPsTdbgAzuUwmgZIVI9V2P/view?usp=drive_link



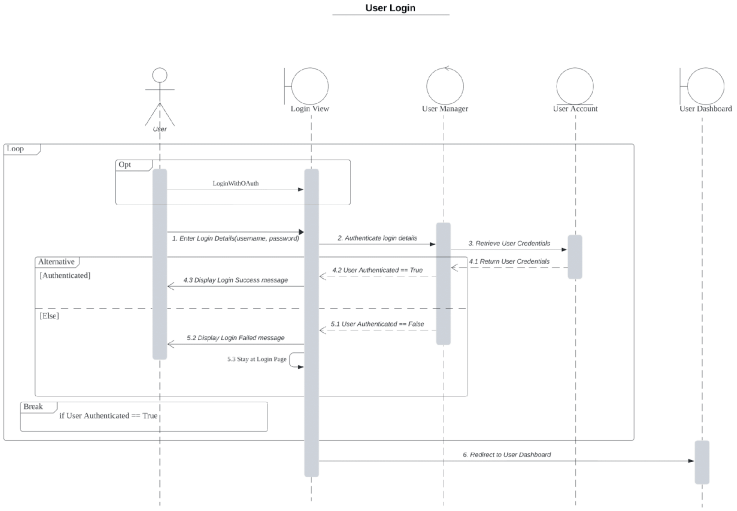
10.8. B8. Strava OAuth Sequence Diagram

High-definition image can be accessed via
https://drive.google.com/file/d/1ItS280ri3fobM7TTHnMW_OynkPbAykWk/view?usp=drive_link



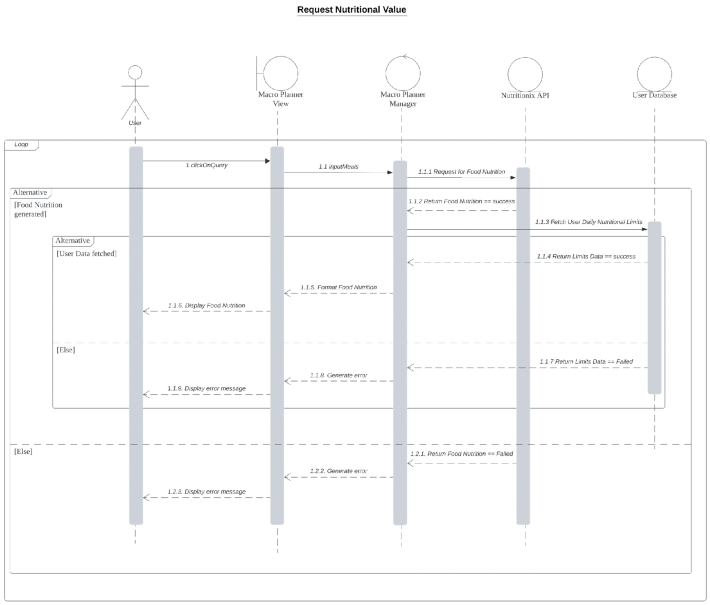
10.9. B9. User Login Sequence Diagram

High-definition image can be accessed via
https://drive.google.com/file/d/1tXSAuRIdsmGsRMQ0nAB-LBibLkovYiyv/view?usp=drive_link



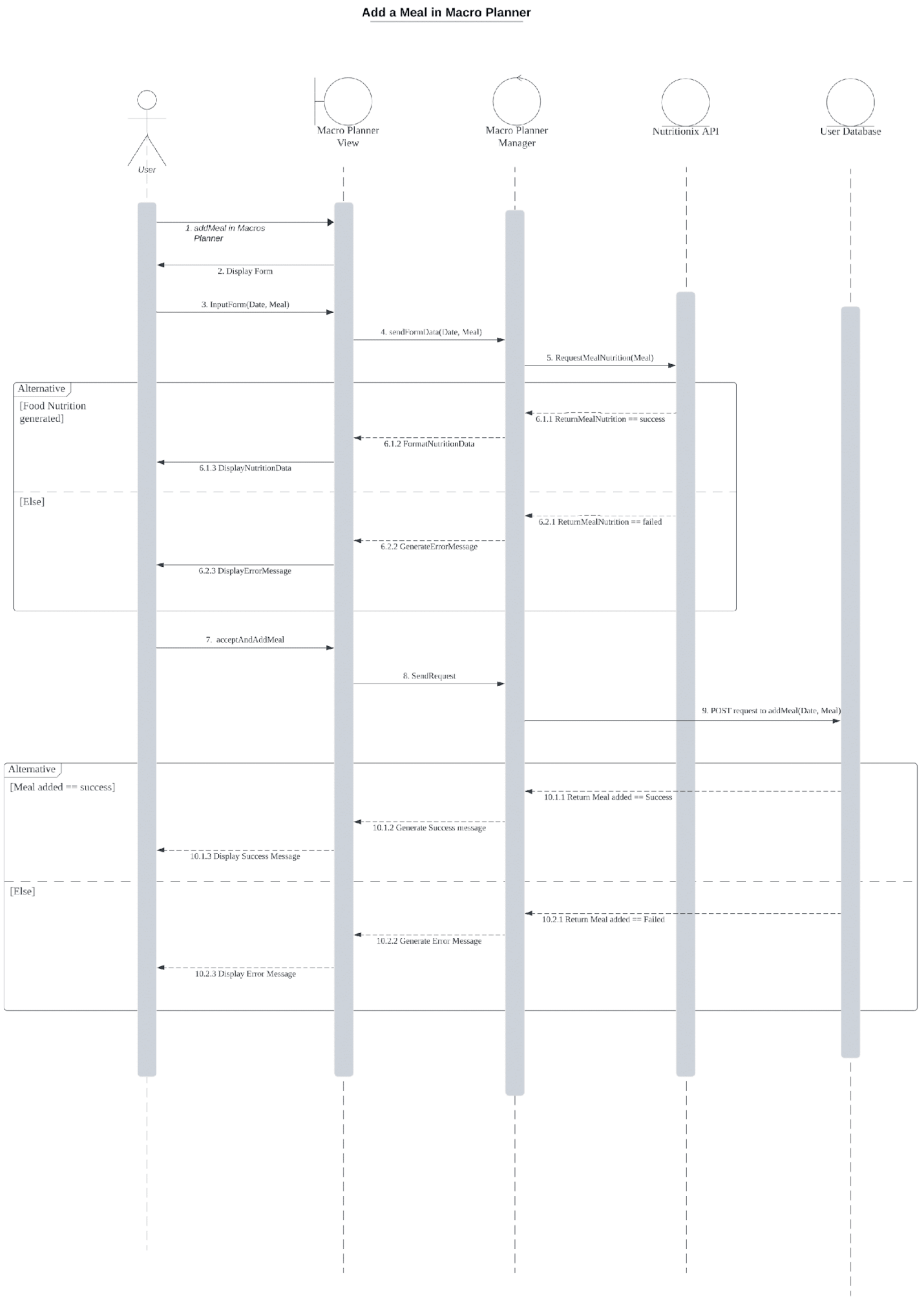
10.10. B10. Query Meal Sequence Diagram

High-definition image can be accessed via
https://drive.google.com/file/d/1x66MfLvSV5VEVRedTFmCsEmaNkwDwy_p/view?usp=drive_link



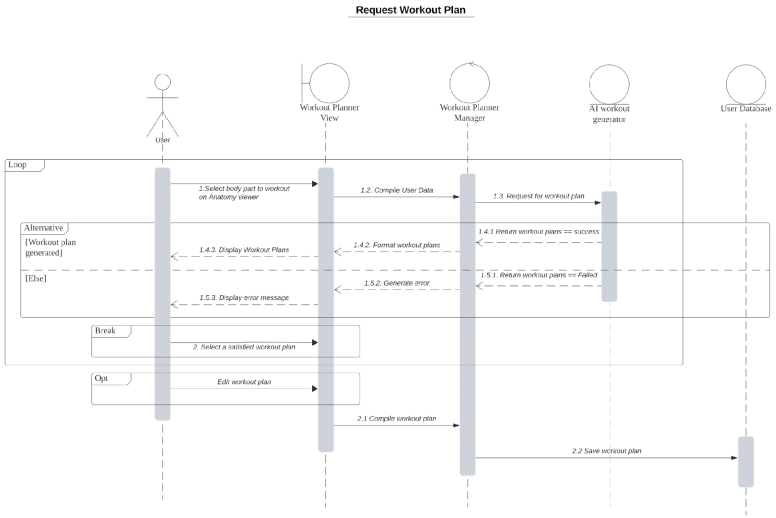
10.11. B11. Add Meal Sequence Diagram

High-definition image can be accessed via
https://drive.google.com/file/d/1cVXG3PCCJ3odfoSMrmwtrcAXGl67awit/view?usp=sharing



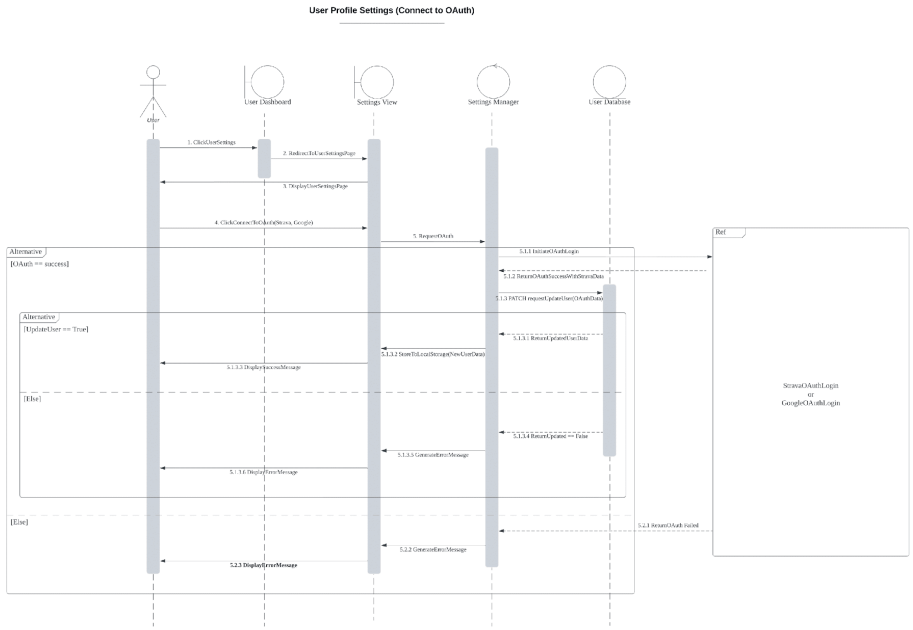
10.12. B12. Generate Workout Sequence Diagram

High-definition image can be accessed via
https://drive.google.com/file/d/1sRcrECFKYte-OcDDJ6s7mTVRZ2WKyY1N/view?usp=drive_link



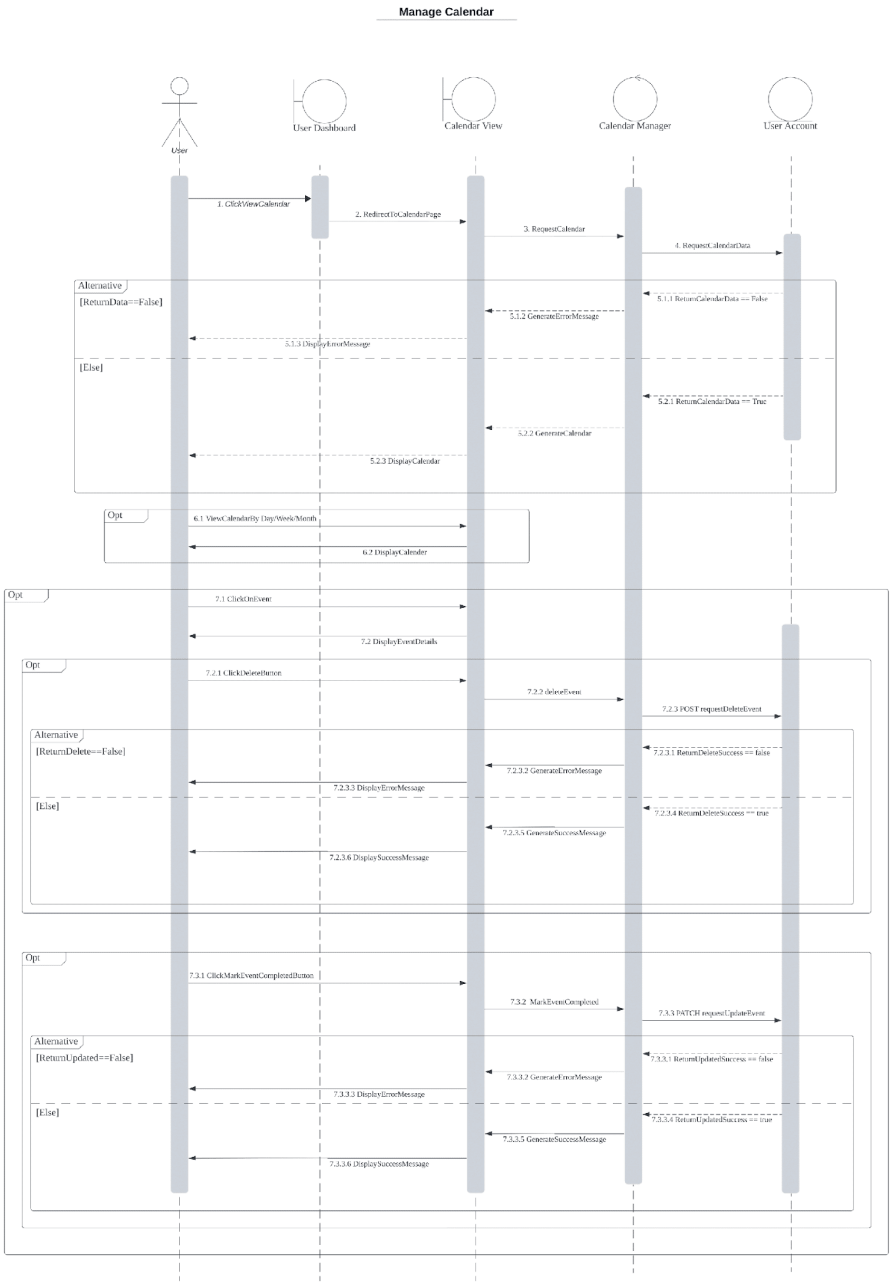
10.13.B13. User Settings Connect to OAuth Sequence Diagram

High-definition image can be accessed via
https://drive.google.com/file/d/13GDJhfcn5xj1wAt9G0nvofAeMc6L9owJ/view?usp=sharing



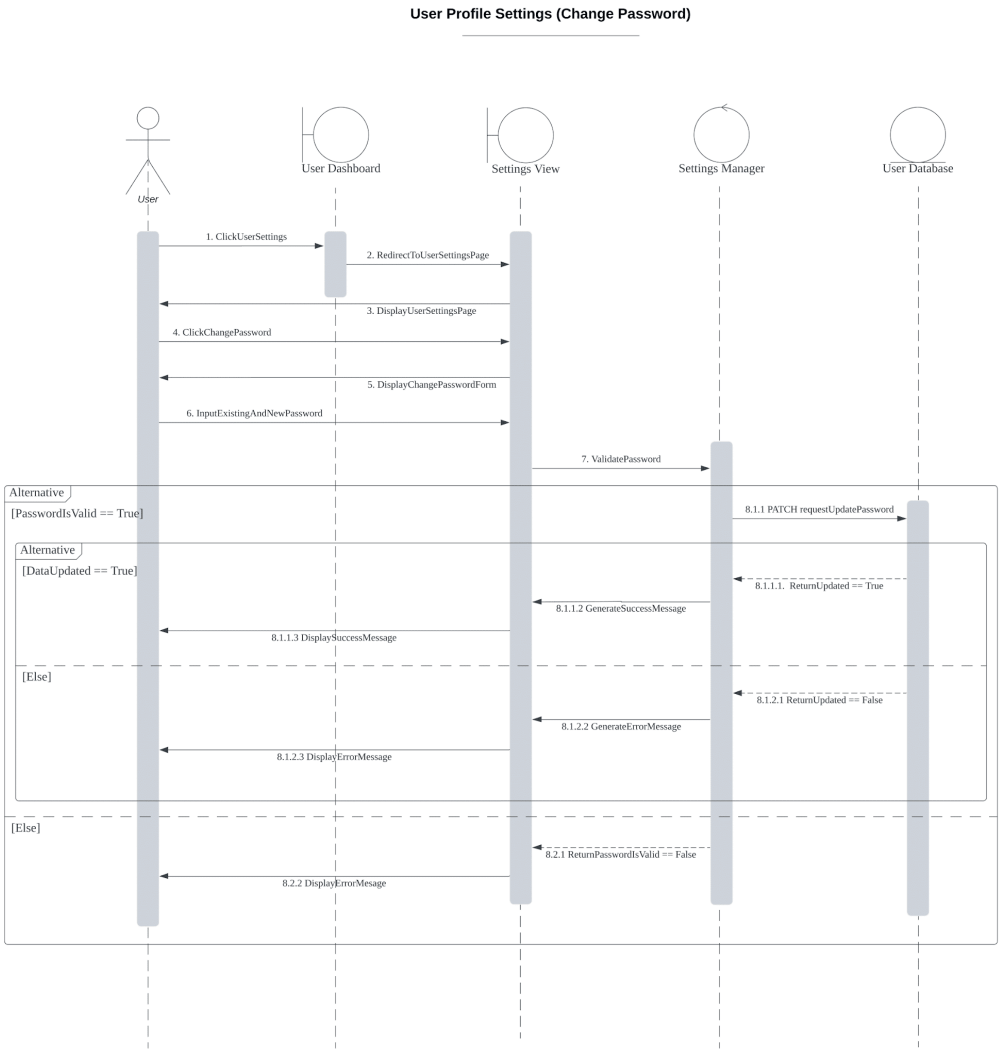
10.14. B14. Manage Calendar Sequence Diagram

High-definition image can be accessed via https://drive.google.com/file/d/1DVVDah36wP86Y_HL6LQYdmECcPKaZ_A3/view?usp=drive_link



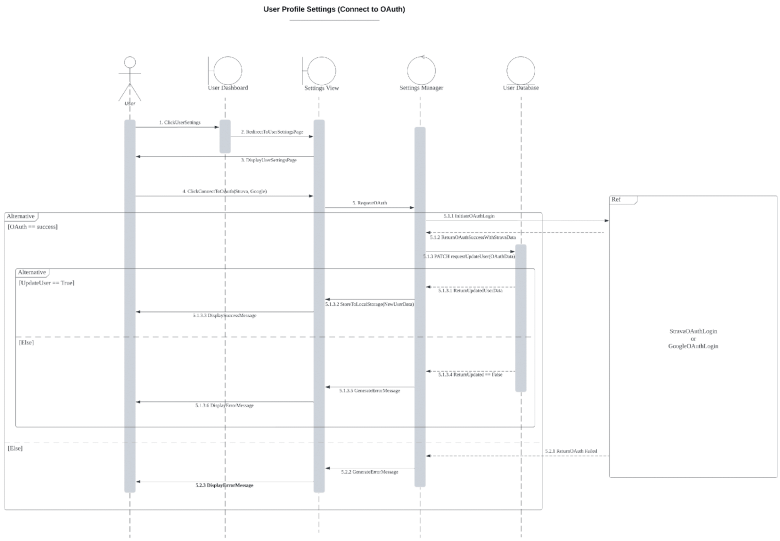
10.15. B15. User Settings Change Password Sequence Diagram

High-definition image can be accessed via
https://drive.google.com/file/d/1_fHq8OvPiRn2x92uF3RSR75aelvfZxMn/view?usp=sharing



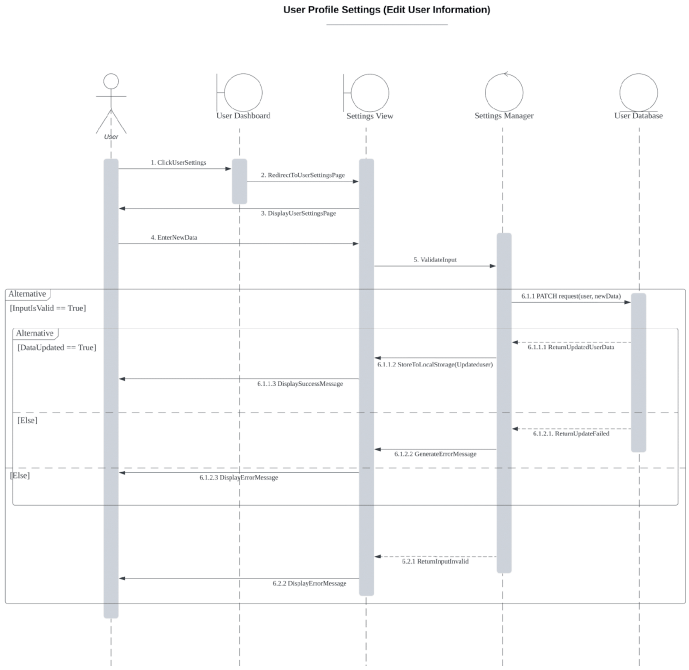
10.16. B16. User Settings Connect to OAuth Sequence Diagram

High-definition image can be accessed via
https://drive.google.com/file/d/13GDJhfcn5xj1wAt9G0nvofAeMc6L9owJ/view?usp=sharing



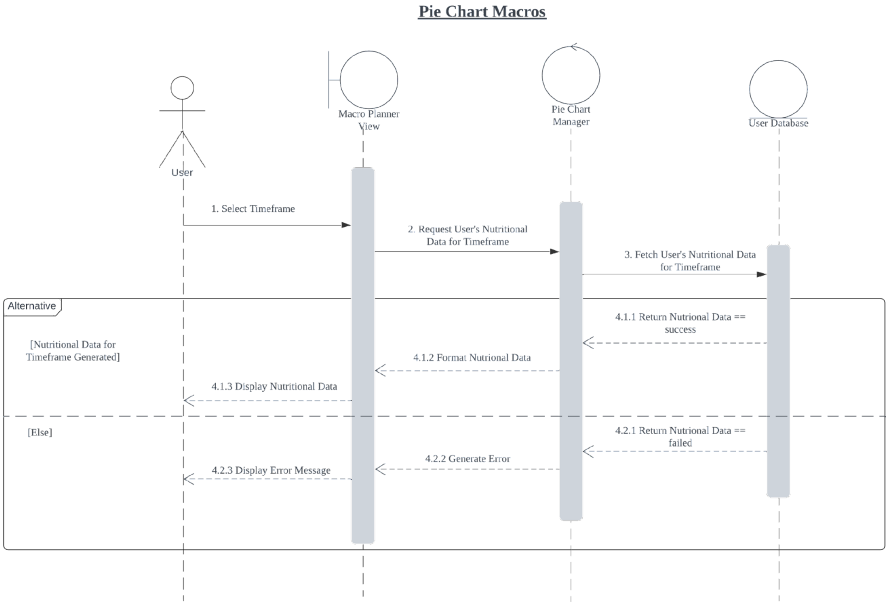
10.17. B17. User Settings Update User Information Sequence Diagram

High-definition image can be accessed via
https://drive.google.com/file/d/1_NN0vCNegvOeByZ-os1pxasFxTFPWMZa/view?usp=sharing



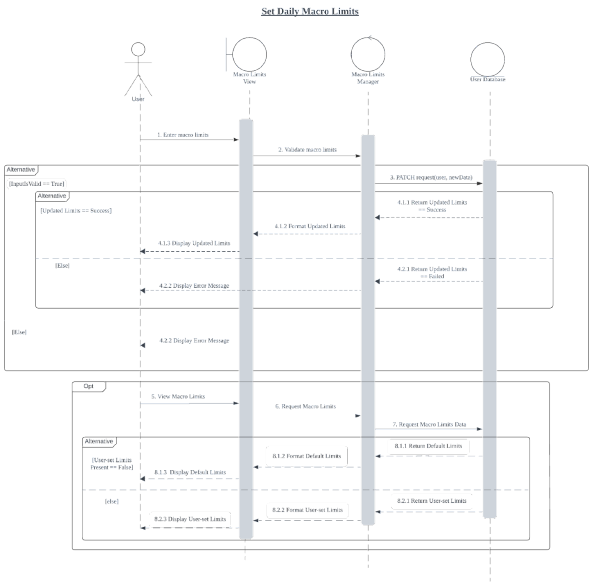
10.18.B18. Pie Chart Macros Sequence Diagram

High-definition image can be accessed via
https://drive.google.com/file/d/1aJeNDzNUm1QUyxOHvvQaFxV5zHd1do-a/view?usp=sharing



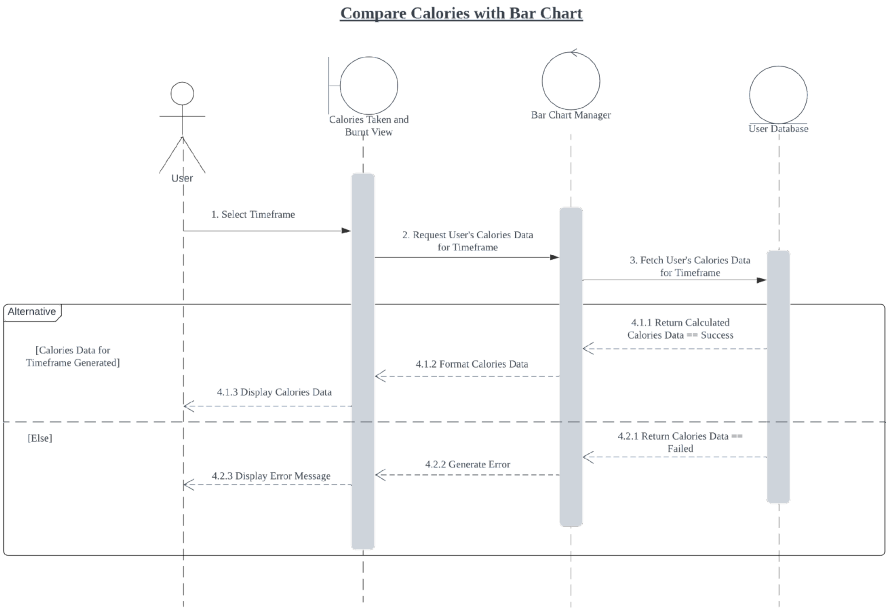
10.19.B19. Set Daily Macros Limits Sequence Diagram

High-definition image can be accessed via
https://drive.google.com/file/d/1Th6AkFCZ5ieLy4CthdMSYiH6NBDBrKdW/view?usp=sharing



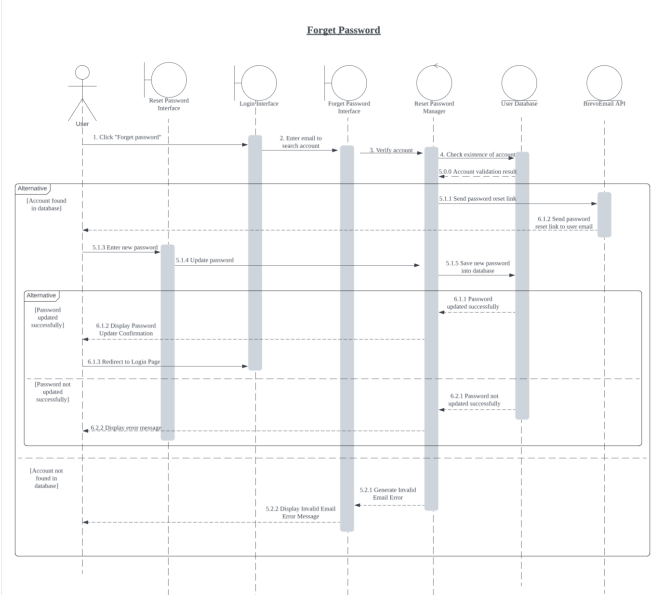
10.20.B20. Compare Calories with Bar Chart Sequence Diagram

High-definition image can be accessed via
https://drive.google.com/file/d/1WI6hJ25qvyY4oOttz3us2J7vYGqT2bIs/view?usp=sharing



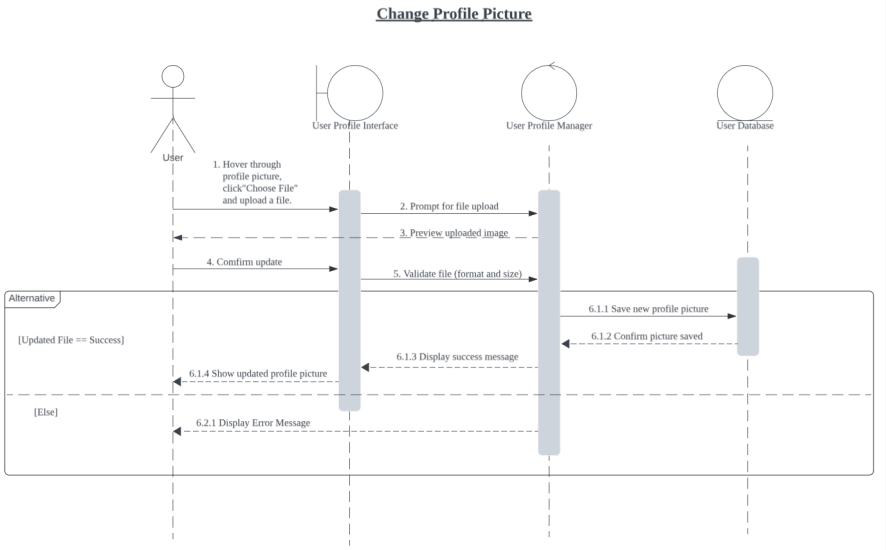
10.21.B21. Forget Password Sequence Diagram

High-definition image can be accessed via
https://drive.google.com/file/d/1ceu5FPTx7ivMKDrYqj3BAxk--IGpESgc/view?usp=drive_link



10.22.B22. Change Profile Picture Sequence Diagram

High-definition image can be accessed via
https://drive.google.com/file/d/1pKKwWffZcC2lhjWIhziBuxZedTLsT1Ug/view?usp=drive_link



11. Appendix C: To Be Determined List

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