University of Toronto

Faculty of Applied Science and Engineering

Vida Fit - Personal Fitness Tracked Documentation

Victoria Piroian

Executive Summary

Vida Fit is a health and fitness information application with the purpose of providing users with a platform to track their meals, sleeping patterns, and physical activity all while setting personal goals to improve their personal health and wellbeing.

The team developed the system architecture after examining the underlying context and motivation of the web application. First, requirements and use cases were developed which influenced the design of the architecture. The functional requirements include: entering entries about eating, sleeping, and exercise patterns, displaying information related to their entries, allowing for the update of previously entered entries, and the deletion of records. The non-functional requirements include that the system must be easily accessible from the web and easy to navigate and understand. Additionally, use cases for creating a profile, logging a meal, logging sleep information, logging an exercise, and adding a goal in progress were developed to provide an all-encompassing overview of the system.

The team used a Waterfall V-Model methodology to emphasize the significance of planning and testing to validate throughout the lifecycle of the web application. Further, this system incorporates static and dynamic structures which define the hierarchy and dependencies in the system as well as the interactions between elements.

UML diagrams were created to guide the system development and help visualize how the system functions. The UML use case diagram summarizes all five use cases described above: update profile, log meal, log exercise, log sleep, and log progress. The UML class diagram shows the connection of the User class to the four Diet, Sleep, Exercise, and Progress classes. Additionally, the UML state chart diagram depicts the Progress object as the user sets and completes many goals. The UML sequence diagram is focused on the log progress use case where interactions are exhibited between the server and user, and between Progress and User objects. The UML activity diagram details the workflow of adding, editing, or deleting a sleep entry as the Log Sleep use case is the area of focus.

Last, the team designed and implemented various test suites with an emphasis on the back end to ensure the quality of the web application. The team faced several challenges but managed to also learn many lessons over the duration of this project. The next steps and recommendations are to implement a login page to provide more security to users, incorporate a food catalog to reduce search time, provide users with a weekly summary of their progress, and send out push notifications to motivate the achievement of their goals.

Table of Contents

1.0 Web Application Overview

1.1 Purpose

1.1.1 Context

Starting post-secondary education can be a difficult transition for young adults. Many students have to face a multitude of systemic issues that might lead to "stress, physical inactivity, poor sleeping habits, over-consumption of alcohol and drugs, and poor nutrition" [1]. For instance, a study conducted by Western University in 2015 discovered that "only 13.2% of students met Canada's Food Guide Nutrition Guidelines and 17.2% met the Canadian Society of Exercise Physiology's Guidelines for Physical Activity". Moreover, it also found that "only 44.8% of students reported that sleep was not an issue". But one of the most alarming statistics from this study was that "only 3.3% of students met the recommendations set across all three health behaviours" [1].

1.1.2 Need for the system

During this transition stage, and even after that, the need to advocate for their own health care is of great importance. Young adults need the tools to keep track of these three health factors, not only to succeed in post-secondary education but to also learn to build better lifestyle habits to "lead both physically and mentally healthy lives" [1]. As such, the web app aims to simplify this tracking task to motivate students to monitor their health, see their progress, and achieve their health goals, which can encompass better sleeping habits, more physical activity, or eating nutritiously.

1.1.3 Gap in the system

Currently, there is steady growth in the mobile health app market, with more than 318,000 of these apps available worldwide [2]. Nonetheless, most of these apps are only available for Android and iOS devices and their web app version can be complex for users to understand, or they lack a web app version altogether. However, students spend roughly 40 hours a week on their laptops for different purposes, including coursework, Zoom classes, video games, and social media [3]. By providing a web app that is available on both their laptops and mobile devices, students will have greater flexibility and be more efficient when it comes to keeping track of their sleeping, eating, and physical activity habits.

1.2 Intended Audience

The web app aims to target young adults, ages 18 to 25, who are currently undergoing post-secondary education, as this demographic will get the most benefit from the app to self-regulate their health while they transition into adulthood and start advocating for their own health. For instance, a survey conducted in the US in 2018 showed that “a total of 64 percent of teens and young adults say they have used a health-related mobile app, with fitness apps being the most commonly reported (45 percent)” [4].

1.3 Content Summary

This section outlines the details of the system and explains how different components – front-end, back-end, use cases, and databases – fit together. The website has four main pages for users to interact with: diet, sleep, exercise, and progress tracking (see Appendix A). They are also able to update their own personal information and measurements.

The web application allows users to create a profile and track their eating habits, physical activity, sleeping patterns and overall progress goals. A user can create a profile for themselves, as well as add their meals, workouts, sleep periods and progress, but can also make updates to previous entities and delete entries they no longer want to have on their profile. Searching for specific entries using keywords is also available for each page.

When designing the back-end of Vida Fit, controllers, entities, and repositories were used to support CRUD functionalities. These functions were then connected to the front-end to allow users to interact with different elements on the user interface. Further, the back-end includes a stored database in the SQL file; SQL queries are used to extract specific entries in order to customize and personalize the view to a particular user.

Since the progress report, there have been several updates to the web application. The most significant change has been the removal of the catalog for the Diet page of the website. Previously, the team had planned to have a database of most common foods and their dietary information, but it became apparent this was unfeasible with the tools and timeframe given. Without the catalog tables, the web application no longer supports the calculation of statistics (ie. calories burned per hour) based on the information entered by users. Since the system does not automatically calculate statistics for the user, a Resources page was added to provide the users with helpful information such as calorie charts and the Canadian food guide. The requirements had to be reprioritized to align with the needs of the user. The team chose to focus on the most important functionalities to users including the CRUD operations, search functionalities, and providing the user with additional resources.

2.0 Summary of the Requirements

2.1 Requirements Overview

2.1.1 Purpose of the requirements

The requirements were identified to ensure that the needs of the targeted demographic were met. These requirements provided a foundation for the product vision and scope, given that they helped the team understand the design and functionality of the web app.

2.1.2 Requirements elicitation

To meet the needs of the user, the team performed two data collection techniques: questionnaires and interviews. The results gathered from these methods allowed the team to identify the participants' current use of health-tracking apps, and get more insights into various ways they track their sleeping, eating, and physical activity patterns. This helped the team decide on the following list of requirements:

Table 1. List of functional requirements

Functional requirements	
R1	The system must display all the main actions on the landing page.
R2	The system must allow users to enter manual entries about their eating, sleeping, and exercise patterns.
R3	The system must display information related to users eating, sleeping, and exercise records.
R4	The system must allow users to update previous entries about their eating, sleeping, and exercise.
R5	The system must allow users to delete records about their eating, sleeping, and exercise.
R6	The system must send reminders (push notifications) at the specific time set by the user.
R7	The system must allow users to enter, update, view, and delete their health plan and reflections about their wellness goals.
R8	The system must provide a weekly summary of the users' goals.
R9	The system must allow the user to edit their profile information.

Table 2. List of non-functional requirements

Non-functional requirements	
R1	The system shall be accessible from the web.
R2	The system shall have a response time below 10 seconds.
R3	The system shall be compatible with Windows and Macs for accessibility.
R4	The system shall have the date format for any record type as follows: date/month/year to ensure consistency across records.
R5	The system shall have an error rate for entering records of eating, sleeping, and exercise details below 10%.
R6	The system shall be easy to navigate and understand.

2.1.3 Decision process

From the two lists of requirements, the team decided to use a weighted decision matrix to vote on the functional and non-functional requirements that would be implemented to address the problem (Refer to Appendices A and B). From this voting process, the team was able to classify both functional and non-functional requirements into three categories: 'must-have' (6-7 votes), 'should-have' (4-5 votes), and 'could-have'(1-3 votes). These are presented in table 3 below.

Table 3. List of requirements

Must-have	Should-have	Could-have
• The system must allow users to enter manual entries about their eating, sleeping, and exercise patterns. • The system must display information related to users eating, sleeping, and exercise records. • The system must allow users to update previous	• The system must display all the main actions on the landing page. • The system shall have a response time below 10 seconds. • The system shall be compatible with Windows and Macs for	• The system must send reminders (push notifications) at the specific time set by the user. • The system must provide a weekly summary of the users' goals. • The system shall have the date format for any record

entries about their eating, sleeping, and exercise. ● The system must allow users to delete records about their eating, sleeping, and exercise. ● The system must allow users to enter, update, view, and delete their health plan and reflections on their wellness goals. ● The system must allow the user to edit their profile information. ● The system shall be accessible from the web.	accessibility. ● The system shall be easy to navigate and understand.	type as follows: date/month/year to ensure consistency across records. ● The system shall have an error rate for entering records of eating, sleeping, and exercise details below 10%.

From this list, all the 'must-have' requirements were implemented in order to satisfy all CRUD operations of the web app and offer four distinct functionalities. Moreover, these requirements cover all three health factors that are meant to help young adults advocate for their health. By implementing these requirements, the users will be able to track their progress and achieve their wellness goals in all three areas, improving and leading healthier lifestyles. On the other hand, the majority of the 'should-have' requirements were implemented, with the exception of the first requirement. In future iterations of the web app, the landing page could be further expanded to provide access to all the main actions in addition to the existing menu. Finally, none of the 'could-have' requirements were implemented as they fell outside the scope and resources of the project and hence were not prioritized.

2.2 Use Cases

Based on the functional and non-functional requirements identified above, the team developed five use cases that outline the processes of logging a meal, logging an exercise, logging a sleep period, logging a client's profile, and logging the client's progress.

The use cases represent the five main tasks a user would complete while using the web application. The use cases also directly relate to the requirements the team has identified as they incorporate all must-have elements of our functional and non-functional requirements.

The team chose to present these use cases to the reader because it provides background and foundation for how the system is supposed to behave. These use cases were pivotal in

influencing the design of the project. Further, they guided the establishment of our goals and helped define the complexity of the system. The team had to refine most of the use cases from the progress report to reflect the changes made in the architecture of the web application. These changes and new use cases are presented below.

Table 4. Use case for logging users

<table>
<tr><td colspan="2">Use Case Name: Log User</td><td>ID: 1.0</td><td>Important Level: high</td></tr>
<tr><td colspan="4">Primary Actor: Client</td></tr>
<tr><td colspan="4">Brief Description: This use case describes how the client can add, delete, or edit details of their user profile.</td></tr>
<tr><td colspan="4">Trigger: Client clicks either the “+” button or the “Edit” button next to one of the profiles on the User page.
Type: External</td></tr>
<tr><td colspan="2">Normal flow of events:
1. Client opens the Users page of their personal health application.
a. If new user, perform subflow S-1 to create a new profile.
b. If existing user wants to update their profile, perform subflow S-2 to update their profile.
c. If an existing user wants to delete their profile, perform subflow S-3 to delete the profile.</td><td colspan="2">Information for steps:</td></tr>
<tr><td colspan="2">Sub-flows:
S-1: Create new profile
1. Client clicks the “+” button.
2. The client enters a client ID, their first name, last name, weight, height, and age. However, only first, last name and ID are mandatory.
3. Client clicks on the “Save” button
4. This profile is then stored in the User relational table.

S-2: Edit existing profile information
1. Client clicks the “Edit” button of a specific entry on the User page.
2. The Client_ID is accessed from the</td><td colspan="2">Information for steps:

← client ID, first name, last name, age, weight, height, password

→ new user profile

→ Client_ID</td></tr>
</table>

User relational table. 3. Client edits attributes of their profile including first name, last name, age, weight and height. They have the ability to change any combination of these fields, the only restriction is that their first and last name cannot be null. Note, the user cannot update their password. 4. Client clicks on the "Save" button 5. The user tuple is updated in the User relational table. S-3: Delete user 1. Client clicks on the "Edit" button next to a user on the *User* page. 2. Client clicks on the "Remove User" button 3. The Client_ID is accessed from the User relational table. 4. The user tuple is deleted from the User relational table.	← first name, last name, age, weight, and height → updated user profile → Client_ID

Summary of Inputs		**Summary of Outputs**	
Description	**Source**	**Description**	**Destination**
New user's info (client ID, first name, last name, age, weight, height, password)	Client input	New user profile	User
Existing user's updated info (first name, last name, age, weight, and height)	Client input	Updated user profile	User
		Client_ID	User

Table 5. Use case for logging meals

Use Case Name: Log Meal	**ID:** 2	**Importance level:** low
Primary Actor: Client		
Brief Description: This use case describes how the client can add, delete, or edit details of a		

<table>
<tr><td colspan="2">meal.</td></tr>
<tr><td colspan="2">Trigger: Client clicks either the “+” button or the “Edit” button next to one of the meals on the Diet page.
Type: External</td></tr>
<tr><td>Normal flow of events:
1. Client opens the Diets page of their personal health application.
a. If user wants to add a new meal, perform subflow S-1 to add new meal.
b. If user wants to update an existing meal, perform subflow S-2 to update the meal.
c. If user wants to delete an existing meal, perform subflow S-3 to delete the meal.</td><td>Information for steps:</td></tr>
<tr><td>Sub-flows:
S-1: Add meal
1. Client clicks the “+” button.
2. The client enters their client ID, meal type, date, meal name, and serving amounts for calories, sugar, carbs, protein, and fat.
3. Client clicks on the “Save” button
4. The meal information is then stored in the Diet relational table.

S-2: Edit meal
1. Client clicks on the “Edit” button of a specific entry on the Diet page.
2. The userMealKey is retrieved from the Diet relational table.
3. Client edits generic information about the retrieved meal, such as the meal name, calories, sugar and macronutrient levels per serving.
4. Client clicks on the “Save” button
5. The meal tuple is updated in the Diet relational table.

S-3: Delete meal</td><td>Information for steps:

← client ID, meal type, date, meal name, calories, sugar, carbs, protein, fat

→ new meal is created

→ userMealKey

← meal name, calories, sugar, carbs, protein, fat

→ updated meal entry</td></tr>
</table>

<table>
<tr><td colspan="2">1. Client clicks on the “Edit” button next to a meal on the Diet page.
2. Client clicks on the “Remove Diet” button
3. The userMealKey is accessed from the Diet relational table.
4. The meal tuple is deleted from the Diet relational table.</td><td colspan="2">→ userMealKey</td></tr>
<tr><td colspan="2">Summary of Inputs</td><td colspan="2">Summary of Outputs</td></tr>
<tr><td>Description</td><td>Source</td><td>Description</td><td>Destination</td></tr>
<tr><td>New meal’s info (client ID, meal type, date, meal name, calories, sugar, carbs, protein, fat)</td><td>Client input</td><td>New meal entry</td><td>Diet</td></tr>
<tr><td rowspan="2">Existing meal’s updated info (meal name, calories, sugar, carbs, protein, fat)</td><td rowspan="2">Client input</td><td>Updated meal entry</td><td>Diet</td></tr>
<tr><td>userMealKey</td><td>Diet</td></tr>
</table>

Table 6. Use case for logging exercise

<table>
<tr><td colspan="2">Use Case Name: Log Exercise</td><td>ID: 3</td><td>Importance level: low</td></tr>
<tr><td colspan="4">Primary Actor: Client</td></tr>
<tr><td colspan="4">Brief Description: This use case describes how the client can add, delete, or edit details of an exercise.</td></tr>
<tr><td colspan="4">Trigger: Client clicks either the “+” button or the “Edit” button next to one of the exercises on the Exercise page.
Type: External</td></tr>
<tr><td colspan="2">Normal flow of events:
1. Client opens the Exercise page of their personal health application.
a. If user wants to add a new exercise, perform subflow S-1 to add new exercise.
b. If user wants to update an existing exercise, perform subflow S-2 to update the entry.</td><td colspan="2">Information for steps:</td></tr>
</table>

c. If user wants to delete an existing exercise, perform subflow S-3 to delete the entry.	
Sub-flows: S-1: Add exercise 1. Client clicks the "+" button. 2. The client enters their client ID, workout ID, date, workout name, duration, and satisfaction 3. Client clicks on the "Save" button 4. The exercise information is then stored in the Exercise relational table. S-2: Edit exercise 1. Client clicks on the "Edit" button of a specific entry on the *Exercise* page. 2. The userExerciseKey is retrieved from the Exercise relational table. 3. Client edits generic information about the retrieved exercise such as workout name, duration, and satisfaction 4. Client clicks on the "Save" button 5. The exercise tuple is updated in the Exercise relational table. S-3: Delete exercise 1. Client clicks on the "Edit" button next to an exercise on the *Exercise* page. 2. Client clicks on the "Remove Exercise" button 3. The userExerciseKey is accessed from the Exercise relational table. 4. The exercise tuple is deleted from the Exercise relational table.	**Information for steps:** ← client ID, workout ID, date, workout name, duration, and satisfaction → new exercise is created → userExerciseKey ← workout name, duration, and satisfaction → updated exercise entry → userExerciseKey

Summary of Inputs		**Summary of Outputs**	
Description	**Source**	**Description**	**Destination**
New exercise's info (client ID, workout ID, date, workout name, duration, and satisfaction)	Client input	New exercise entry	Exercise

Existing exercise's updated info (workout name, duration, and satisfaction)	Client input	Updated exercise entry	Exercise
		userExerciseKey	Exercise

Table 7. Use case for logging sleep

Use Case Name: Log Sleep	**ID:** 4	**Importance level:** low
Primary Actor: Existing client		
Brief Description: This use case describes how the client can add, delete, or edit details of a sleep period.		
Trigger: Client clicks either the "+" button or the "Edit" button next to one of the sleep entries on the *Sleep* page. **Type:** External		
Normal flow of events: 1. Client opens the Sleep page of their personal health application. a. If user wants to add a new sleep entry, perform subflow S-1 to add new entry. b. If user wants to update an existing sleep entry, perform subflow S-2 to update the entry. c. If user wants to delete an existing sleep entry, perform subflow S-3 to delete the entry.	**Information for steps:**	
Sub-flows: S-1: Add sleep entry 1. Client clicks the "+" button. 2. The client enters their client ID, date, duration, rest score, if they had a dream (true/false), if they woke up with an alarm (true/false), and number of naps. 3. Client clicks on the "Save" button 4. The sleep entry information is then stored in the Sleep relational table. S-2: Edit sleep entry	**Information for steps:** ← client ID, date, duration, rest score, dream, alarm, naps → new sleep entry is created	

1. Client clicks on the “Edit” button of a specific entry on the *Sleep* page. 2. The userSleepKey is retrieved from the Sleep relational table. 3. Client edits generic information about the retrieved sleep entry such as duration, rest score, if they had a dream (true/false), if they woke up with an alarm (true/false), and number of naps 4. Client clicks on the “Save” button 5. The sleep tuple is updated in the Sleep relational table. S-3: Delete sleep entry 1. Client clicks on the “Edit” button next to a sleep entry on the *Sleep* page. 2. Client clicks on the “Remove Sleep” button 3. The userSleepKey is accessed from the Sleep relational table. 4. The sleep tuple is deleted from the Sleep relational table.	→ userSleepKey ← duration, rest score, dream, alarm, naps → updated sleep entry → userSleepKey

Summary of Inputs		**Summary of Outputs**	
Description	**Source**	**Description**	**Destination**
New sleep entry’s info (client ID, date, duration, rest score, dream, alarm, naps)	Client input	New sleep entry	Sleep
Existing sleep entry’s updated info (duration, rest score, dream, alarm, naps)	Client input	Updated sleep entry	Sleep
		userSleepKey	Sleep

Table 8. Use case for logging progress

Use Case Name: Log progress	**ID:** 5	**Important Level:** medium
Primary Actor: Client		
Brief Description: This use case describes how the client can add, delete, or edit details of their goals related to sleep, exercise, and diet.		

<table>
<tr><td colspan="2">Trigger: Client clicks either the “+” button or the “Edit” button next to one of the progress entries on the Progress page.
Type: External</td></tr>
<tr><td>Normal flow of events:
1. Client opens the Progress page of their personal health application.
a. If user wants to add a new progress entry, perform subflow S-1 to add new entry.
b. If user wants to update an existing progress entry, perform subflow S-2 to update the entry.
c. If user wants to delete an existing progress entry, perform subflow S-3 to delete the entry.</td><td>Information for steps:</td></tr>
<tr><td>Sub-flows:
S-1: Add progress entry
1. Client clicks the “+” button.
2. The client enters their client ID, progress ID, goal, progress score and reflection.
3. Client clicks on the “Save” button
4. The progress entry information is then stored in the Progress relational table.

S-2: Edit progress entry
1. Client clicks on the “Edit” button of a specific entry on the Progress page.
2. The userProgressKey is retrieved from the Progress relational table.
3. Client edits generic information about the retrieved progress entry such as goal, progress score and reflection.
4. Client clicks on the “Save” button
5. The progress tuple is updated in the Progress relational table.

S-3: Delete progress entry
1. Client clicks on the “Edit” button next to a progress entry on the Progress page.
2. Client clicks on the “Remove</td><td>

← client ID, progress ID, goal, progress score and reflection

→ new progress entry is created

→ userProgressKey

← goal, progress score and reflection

→ updated progress entry</td></tr>
</table>

<table>
<tr><td colspan="2">Progress” button
3. The userProgressKey is accessed from the Progress relational table.
4. The progress tuple is deleted from the Progress relational table.</td><td colspan="2">→ userProgressKey</td></tr>
<tr><td colspan="2">Summary of Inputs</td><td colspan="2">Summary of Outputs</td></tr>
<tr><td>Description</td><td>Source</td><td>Description</td><td>Destination</td></tr>
<tr><td>New progress entry’s info (client ID, progress ID, goal, progress score and reflection)</td><td>Client input</td><td>New progress entry</td><td>Progress</td></tr>
<tr><td rowspan="2">Existing progress entry’s updated info (goal, progress score and reflection)</td><td rowspan="2">Client input</td><td>Updated progress entry</td><td>Progress</td></tr>
<tr><td>userProgressKey</td><td>Progress</td></tr>
</table>

3.0 Web App Design

3.1 Design Principles

Due to the nature and time frame of the project, it did not make sense to use rapid application development or agile approaches. The team used a waterfall method because it ensures the identification and understanding of the requirements before the team begins programming on the front and back ends. Furthermore, the V-Model waterfall methodology was used to ensure that there was a focus on planning and testing as a method of validation. This V-Model allowed the team to have a clear understanding of the tests that will be used before the coding process began. The main strength of this methodology is that it helped to prevent major bugs and errors when deploying the web application.

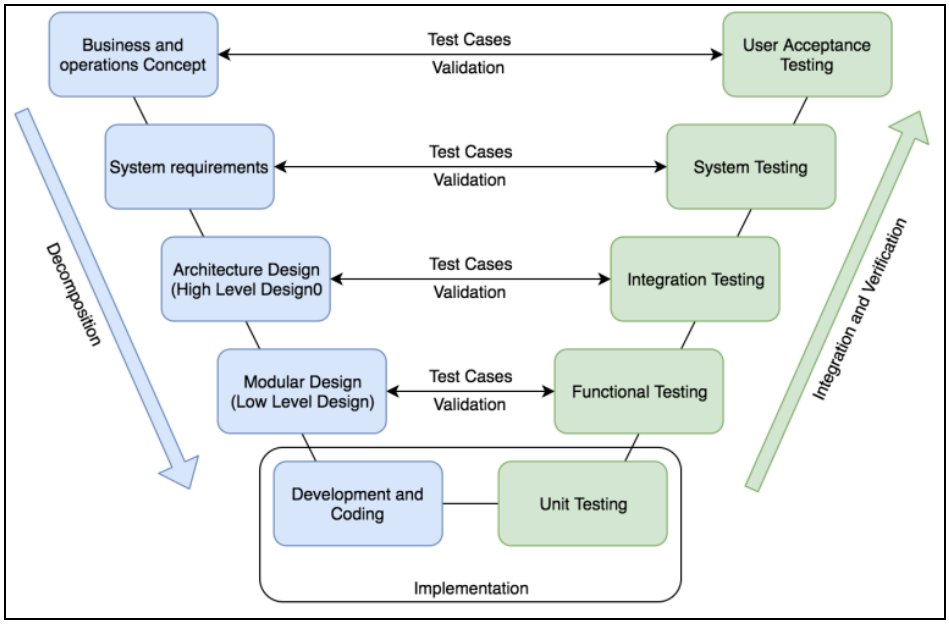

Figure 1. Visual representation of Waterfall V-Model [5]

Furthermore, the architecture of this project is based on the MVC (Model View Controller) Model. The team uses databases to store crucial information, webpages, and visuals to provide the user with an easy-to-navigate visual interface, and JDBC and Java implementations to connect the model and view providing the project with controllers. All components of the MVC Model will be applied to Vida Fit.

The architecture design of the system includes both static and dynamic structures. The control model follows a centralized control where one system has overall responsibility for control. The system specifically follows a Call-Return where the Main program is Users and has several classes that are managed by this central program. The basic architectural style of the system is the client-server model. The clients are represented by the users inputting information into the web app, and the servers are the diet, sleep, exercise, and progress pages of the website.

3.2 UML Use Case Diagram

The UML Use Case diagram summarizes all five of the use cases described in Section 2.2 (see Figure 2). The diagram includes all the appropriate use cases, actors, the communication between the two as well as the systems boundary. The appointment system is the Vida Fit System Web Application, where the actors are new users who create a new profile through the log user use case, as well as existing users who interact with each of the five use cases.

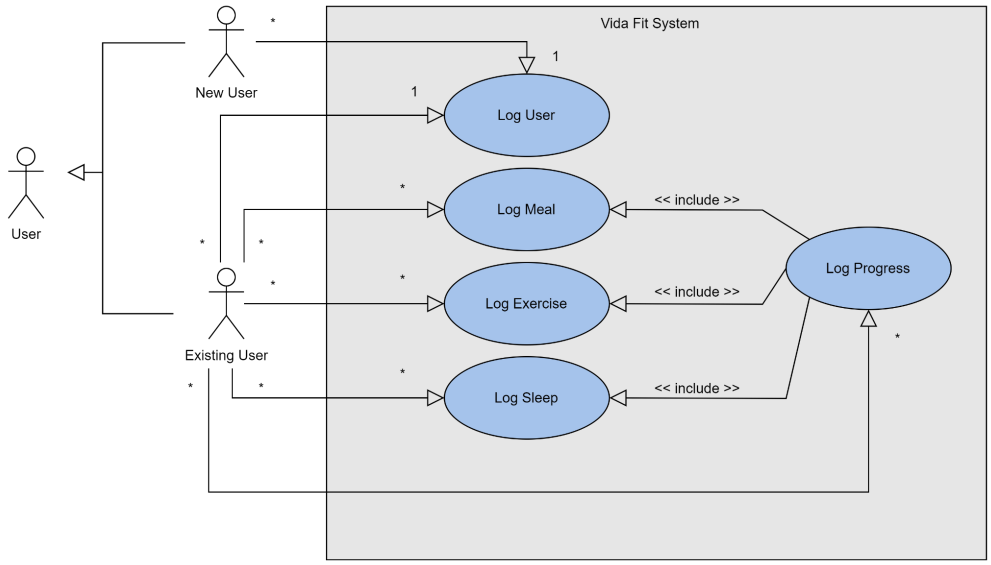

Figure 2: UML Use Case Diagram

3.3 UML Class Diagram

The UML class diagram for the entire system can be found in Figure 3 below. As a brief overview of the design decisions, the team's class diagram is centered around the User class which connects many of the other classes. Each of the Diet, Sleep, Exercise, and Progress classes are related to the primary User table through various interactions including a user going to sleep, eating, performing an exercise, or tracking progress. Each of those four classes has its corresponding composite keys, which uniquely identify each entry into their data tables. Each of the classes for the composite keys also interacts with the User table through their corresponding foreign keys. For instance, each entry in the Diet class is identified by its unique UserMealKey, which contains the ClientId from the User class.

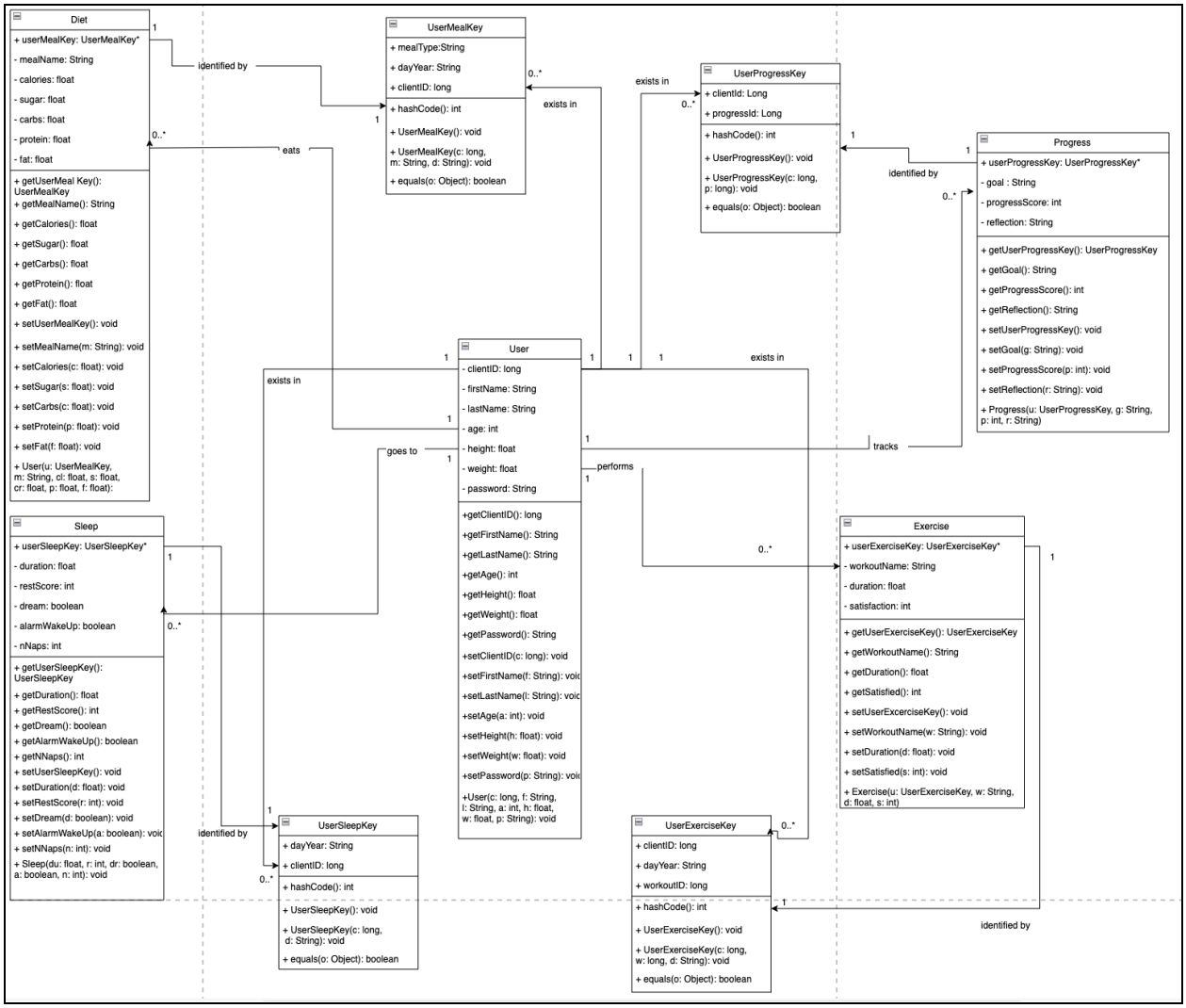

Figure 3: UML Class Diagram

3.4 UML State Chart Diagram

For the UML State chart diagram (Refer to Figure 4), the team has chosen to pick the Progress object to track a user as they input and complete various different goals related to their exercise, eating and sleep habits. One of the benefits of Vida Fit is that users are able to track their improvements in various different aspects of their lifestyle and so the team wanted to highlight the various flow of events, including different states and events that they will encounter while creating and tracking their progress goals.

The team has chosen to make the diagram general (without assigning it to a particular exercise, sleep, or diet goal) to highlight particular features and attributes of the Progress object and class. However, the flexibility of this object also allows for progress tracking across the user's diet, sleep, and exercise.

The states and events being analyzed in this diagram are the following: a goal being set, progress being made, that same progress being logged, and then when the user reaches their goal they update the progress to completed and create a reflection. Otherwise, they keep trying until the goal is reached and the progress is completed.

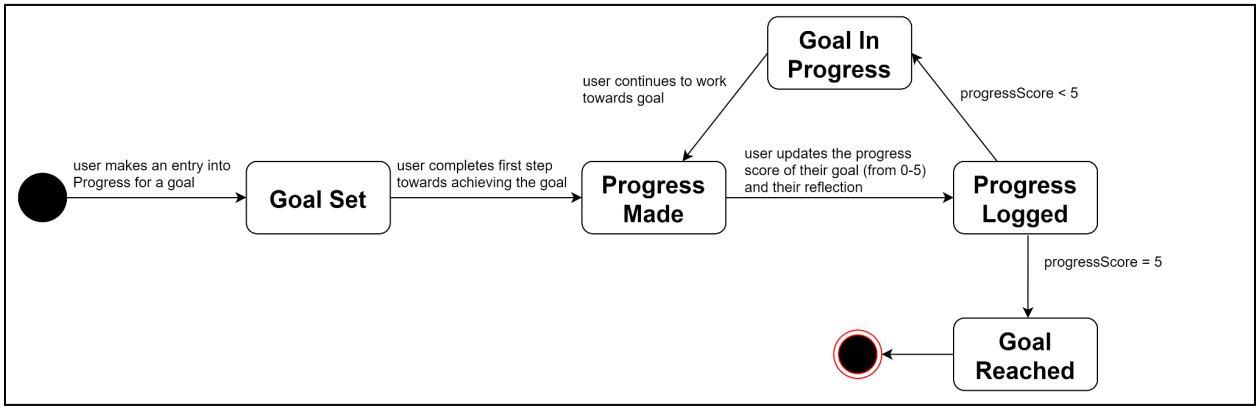

Figure 4: UML State chart diagram

3.5 UML Sequence Diagram

The team has chosen to select the Log Progress use case (ID: 5) for the UML sequence diagram as can be seen in Figure 5. The objects taking part in this interaction are the server and the user, including as well the Progress and User objects.

The use case is initiated when the user opens the progress tab and creates a new progress goal object that is related to the User through the clientId. The server then returns the progress that was just initialized and will contain its corresponding attributes including the new UserProgressKey and the goal. Once the object has been created through this interaction with the server, the user can now set the initial progress score and reflection, and the server will display the progressScore and reflection on the progress tab of the web application. The progressScore is used to route the progress object into one of two alternative routes. If the progressScore is less than 5, then the user will continue to make progress towards the goal until they reach progressScore = 5. The server then continues to update its score until they reach the goal, which is represented in the loop. If their score is already 5, the server returns the final progress score and then the user completes their final goal reflection and the server displays the completed goal on the application's progress tab.

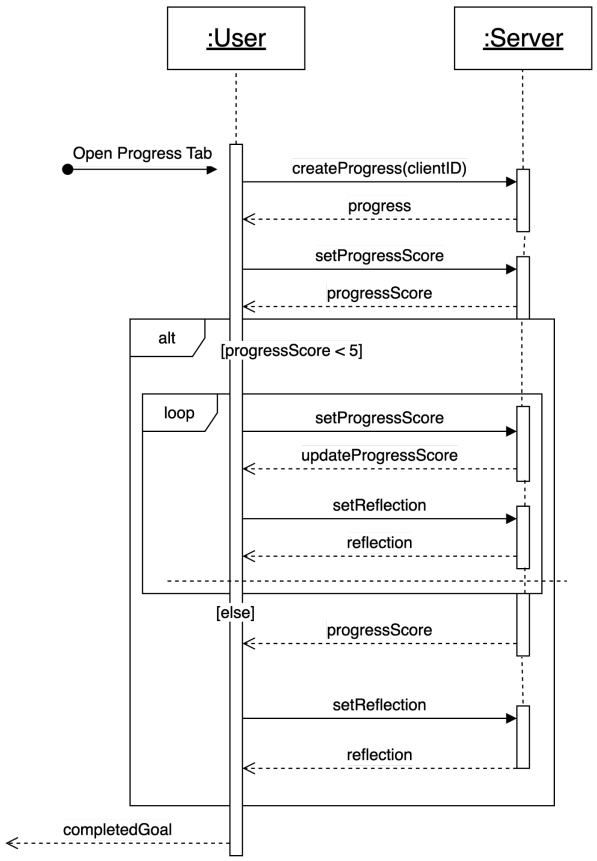

Figure 5: UML Sequence Diagram for Use Case 5

3.6 UML Activity Diagram

The Log Sleep use case (ID: 4) has been chosen to model the UML activity diagram (see Figure 6). The activity diagram models the workflow behind adding, editing, or deleting a sleep entry into the system. The various activities and their corresponding events are detailed below.

Initially, the client opens the Sleep tab which leads them to a decision point (branch). The guard conditions for this branch are decided based on whether or not the user has already created a sleep object for a given date. For instance, if the user wants to log a new sleep entry they would follow the ‘else’ branch which leads them to add a new sleep period and then enter the corresponding sleep information such as the sleep duration, rest score, whether they had a dream, whether they were woken up by an alarm, and the number of naps taken that day. When the user is accessing a Sleep object that already exists, they reach another decision point based on whether the user wants to update a sleep period or delete it. In the case where they want to edit it, they then select which sleep period they want to edit and update the desired sleep information. If they want to delete a sleep period, they select the period to delete. Each of these branches merge, and the sleep tab is then updated to reflect the changes to the necessary sleep periods.

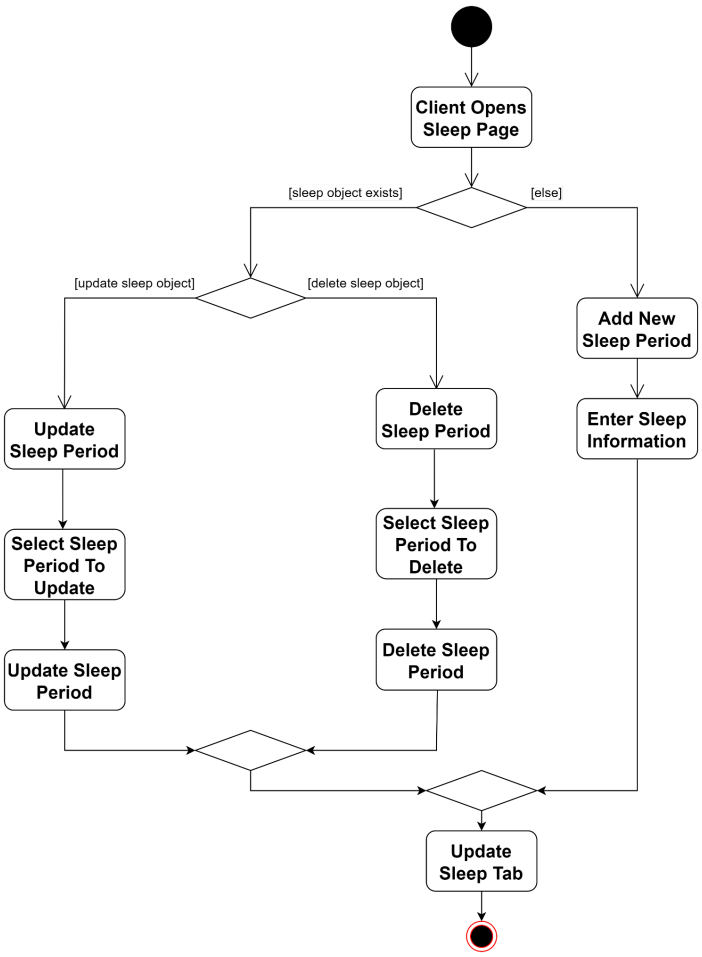

Figure 6: UML Activity Diagram for Use Case 4

4.0 Access by Wireless Devices

Currently, the web app is accessible from laptops, iPads/tablets, and mobile wireless devices. After running an analysis through Lighthouse to compare the performance of the web application on desktop versus mobile devices, it was determined that performance is weaker on mobile devices. Other metrics such as accessibility, best practices, and search engine optimization (SEO) are very high. These results are shown in Figure 7. There are several steps the team would need to take to modify the web app to better support use on mobile devices.

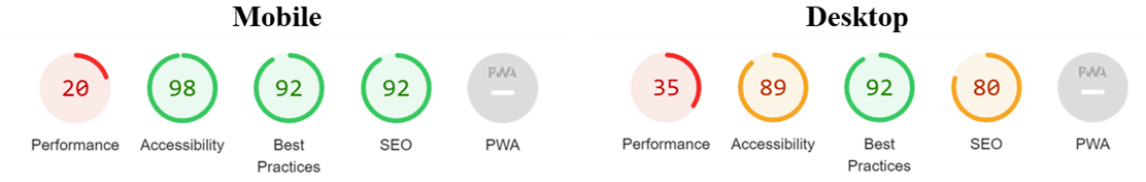

Figure 7. Mobile vs. Desktop Performance

The team would have to investigate the following factors to modify for better use on mobile devices: browser compatibility and capability, the user interface on a mobile device, data storage, and security/privacy. First, research would have to be conducted to indicate which browsers are compatible with HTML, Javascript, and Java. Once verified whether these applications are compatible and supported by mobile browsers, the team may investigate the next factor of transition.

Additionally, the user interface of the web app must be reevaluated to ensure that the website is responsive to feedback on a touch screen. The team would need to implement a user interface that is well suited for a mobile device screen as it is much smaller than a laptop screen. For example, some changes may include making image sizes smaller and stacking information vertically instead of horizontally for each line item. Additionally, changes would be made regarding the placement of certain features and buttons on each screen to allow for a seamless experience for the user. Further research would be conducted to collect data on optimal placements of buttons on each page. Next, the team must determine the database storage system that would be in place on mobile devices. Further investigation into the location of data (online vs. offline) would indicate how the new database system should be redesigned to fit a mobile device.

It is important to consider that certain functionalities are limited in the mobile version of the website. For example, users are unable to view the logs in a horizontal table format at a legible size without the need to scroll to the right/left. Additionally, the image of the Canadian Food Guide on the Resources page is not as compatible with the size of a mobile device, meaning that users may need to either zoom in to read the text or scroll to the right/left. While these functionalities present challenges, there are also future opportunities with the web application on mobile devices. For example, in the future, users may be able to take pictures of their meals on their mobile devices and add them to their diet log.

Efficiency and error management are two human factor principles that may be considered in the implementation of the mobile version of the web app. Ensuring that users can complete a task in the shortest amount of time possible and that error prevention is valued is pivotal to a good user experience.

5.0 Quality Assurance and Testing

Given that the team followed the Waterfall V-model methodology, the objective was to test all of the entity's CRUD operations for each of the five entities used in the project, following the methods shown in the labs. One test suite contained all the test cases needed for a specific entity. The method of unit testing was used to test smaller units in isolation from all other units. For

instance, in the case of the User Entity, the corresponding test suite contained the following tests cases:

- **getUser**
    - This test case aimed to perform a GET request using the user's ID to obtain and verify all the user's details contained in the database, including their ID, first name, last name, age, height, weight, and password.
- **addUser**
    - This test case aimed to perform a POST request to add information about a user to verify that the new user was indeed created in the corresponding repository and that the new user details were the correct ones, meaning they matched the information provided on the POST request.
- **deleteUser**
    - This test case aimed to perform a DELETE request to eliminate a user from the database using the user's ID, and verify that it was not found if a search was performed.
- **updateUser**
    - This test case aimed to perform a PUT request to update a few fields from an existing user and verify that they were correctly updated in the database.

We tried to conduct integration testing using the same operations for the rest of the entities, however the team had some difficulties with the composite keys, and only GET and DELETE requests were successfully implemented on the rest of the entities. For the rest of the CRUD operations and to search for specific entries, the team decided to manually test this using the front end. The team created 12 functioning test cases in total in order to validate functionality of the web application.

In order to assure the quality of the web application, the team would have implemented system testing and integration testing. System testing would use real inputs to test the entire system. Further, integration testing would have tested that all modules work together. This would have ensured the system functions properly as a whole and not just in the individual performance of each function. Additionally, system and integration testing would reduce bugs and help improve performance of the web application [6].

6.0 Discussion of Challenges

The team faced a plethora of challenges during the project. A brief summary of the most significant ones has been included below and has been categorized in two groups.

6.1 Technical Challenges

The greatest challenge of the coding aspect of the project was learning and applying front-end principles to the web application. To overcome this challenge of a lack of front-end coding knowledge (ie. HTML, CSS, etc.), several team members completed extensive independent research to help the other developers on the team complete the front-end coding and build the front-end of the web application.

Another challenge faced by the developers on the team was applying tutorial content to the context of our project. The team overcame this through trial and error by interpreting and testing specific lines of code to understand their functionality.

6.2 Non-Technical Challenges

Furthermore, the team had trouble dividing the work amongst members to ensure each member had an equal portion. Many team members wanted to be working on the development aspect of the web application, but this simply was not feasible given the complexity of the project. While the team found it difficult to measure how to equally assign portions of the project, superior communication skills of all team members allowed for this process to be as seamless as possible. Moreover, the team divided up the sections of the project in a strategic and optimal manner by assigning each member a portion that would play to their strengths. This allowed each member to feel confident and supported in the work they were completing.

Last, the team found it very difficult to abide by the time constraints and timeline of the project. The team felt the project was too large to be completed in the time given. Although the deadlines were quite far apart, the team had trouble managing the time and expectations of the project. To deal with this challenge, the team set many internal deadlines prior to original course deadlines to allow ample time to ask the professor and teaching assistants questions to help the web application exceed expectations. The team also acted efficiently by cutting requirements and features that were not pivotal to the web application's interface.

Although the team encountered numerous challenges during the project, all challenges were overcome due to the communication and support from each team member.

7.0 Lesson Learned

Examining the challenges the team encountered, there are several lessons learned and things that would have been done differently. First, the lack of a defined workflow and specific roles resulted in a bottleneck in the workflow. To address this issue, the team would spend more time dividing the work and setting hard deadlines at the beginning of the project. Further, the team found it difficult to outline what they believed the website would look like. Each team member had a different understanding of how the web application would function which resulted in

miscommunications when designing the system. To overcome this challenge the team members would spend more time outlining specifically how they imagine the web application to function, look, and interact.

There are several positive takeaways from this project. This project allowed the team to truly appreciate everything that goes into building a web application. The team has a better understanding and greater appreciation for web applications because of the practical understanding gained. This project has been greatly beneficial outside of the course scope. The CRUD principles and project management required in this project has helped each of us manage other projects better.

The next steps of this project include implementing a login page that gives users more security on the web application. This would allow for more security on the web application and keeps users' information safe. Further, the implementation of a food catalog would allow users to browse through common meals rather than manually inputting their meals which would reduce search time. Last, the team hopes to provide users with a weekly summary of their weekly progress and send push notifications to provide motivation to achieve the goals they set out, and also develop an AI/Machine Learning algorithm to recognize commonly searched features on the website; although this may be out of scope for this course, the team is looking to implement these functionalities in the future.

8.0 References

[1] “The body of the student: A deep dive into student health and Wellness,” *OUSA*, 24-Sep-2019. [Online]. Available: https://www.ousa.ca/blog_student_health_and_wellness. [Accessed: 01-Dec-2022].

[2] R. Hall, “Health apps: How mobile apps are improving our lives & well being,” MindSea, 02-Jun-2022. [Online]. Available: https://mindsea.com/health-apps/#:~:text=Share%20this%20Post%3A&text=According%20to%20IQVIA%2C%20there%20are,available%20just%20two%20years%20ago. [Accessed: 21-Oct-2022].

[3] S. Lumley, “Students spend 40 hours a week on laptops - for coursework, classes, and gaming,” *Mirror*, 18-Jul-2022. [Online]. Available: https://www.mirror.co.uk/tech/students-40-hours-week-laptop-27506582. [Accessed: 01-Dec-2022].

[4] J. Comstock, “Survey: Two-thirds of teens, young adults have used a health app,” MobiHealthNews, 01-Aug-2018. [Online]. Available: https://www.mobihealthnews.com/content/survey-two-thirds-teens-young-adults-have-used-health-app#:~:text=%E2%80%9CA%20total%20of%2064%20percent,%2C%E2%80%9D%20the%20report%27s%20authors%20wrote. [Accessed: 01-Dec-2022].

[5] Mohamed Sami, “The validation and Verification Model - the V-Model,” *Mohamed Sami*, 12-Feb-2019. [Online]. Available: https://melsatar.blog/2018/08/27/the-validation-and-verification-model-the-v-model/. [Accessed: 01-Dec-2022].

[6] “What is software testing and how does it work?,” *IBM*. [Online]. Available: https://www.ibm.com/topics/software-testing#:~:text=Software%20testing%20is%20the%20process,development%20costs%20and%20improving%20performance. [Accessed: 06-Dec-2022].

9.0 Appendices

Appendix A:

Figure 7: The Home Page

Figure 8: The User Page

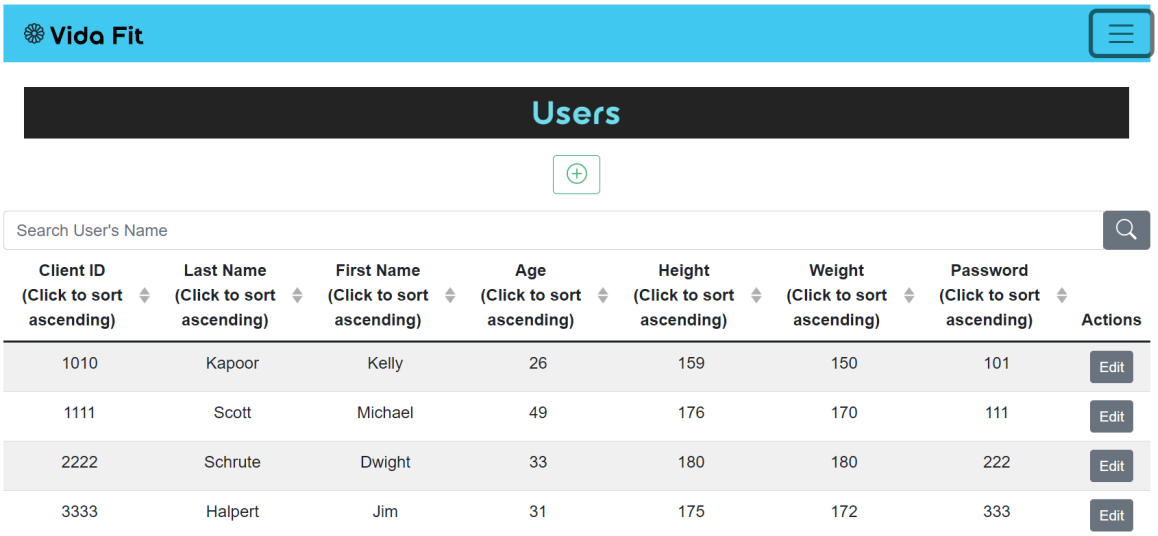

Figure 9: About Us Page

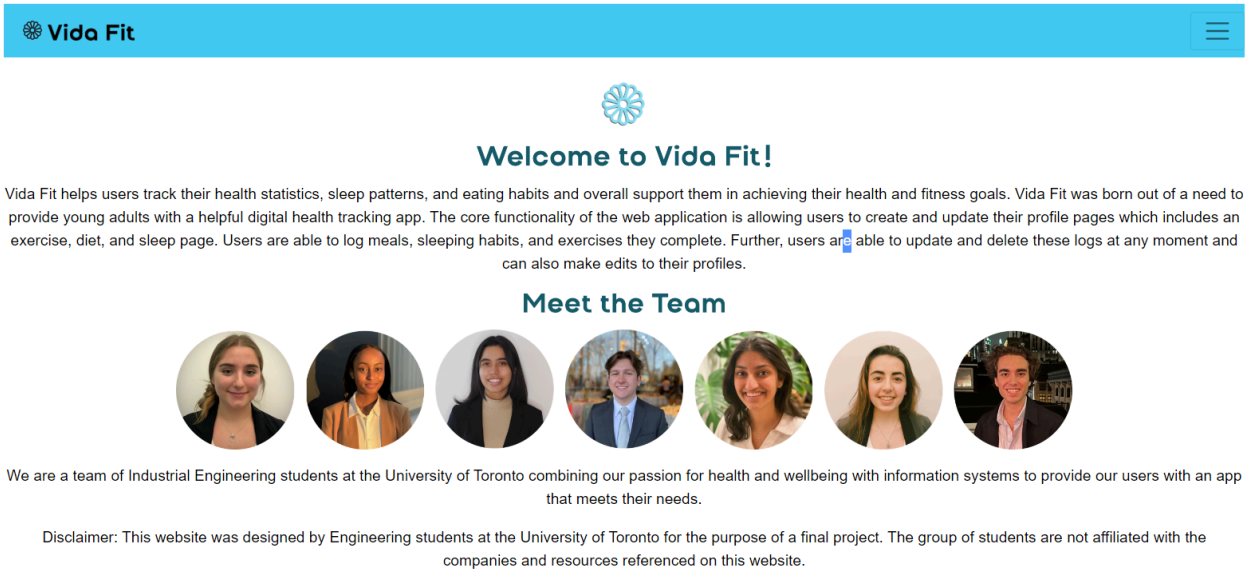

Figure 10: Exercise Page

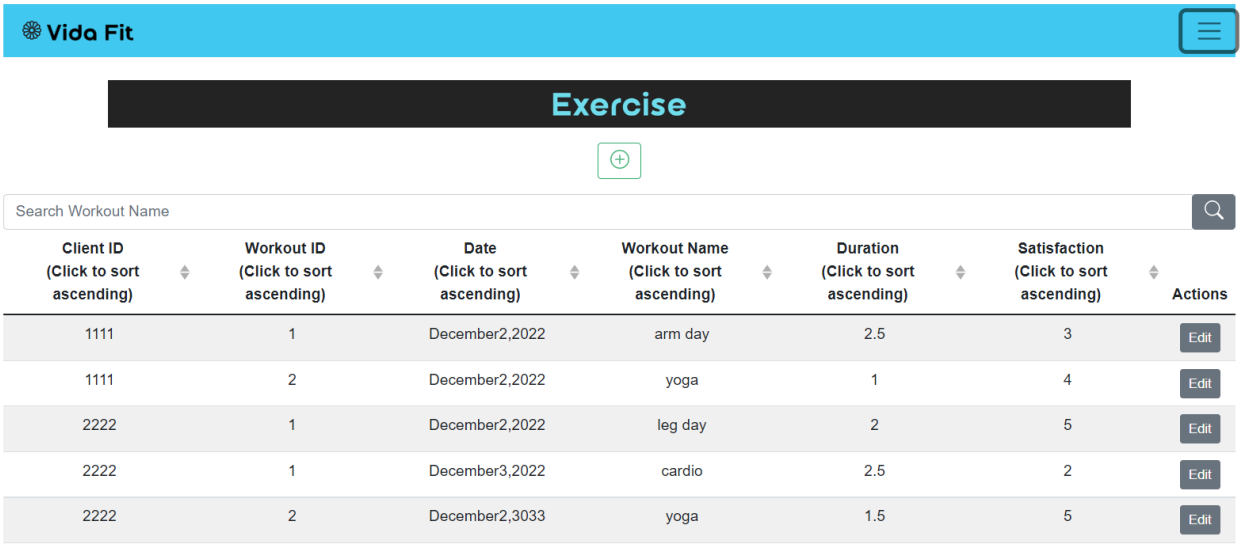

Figure 11: Diets Page

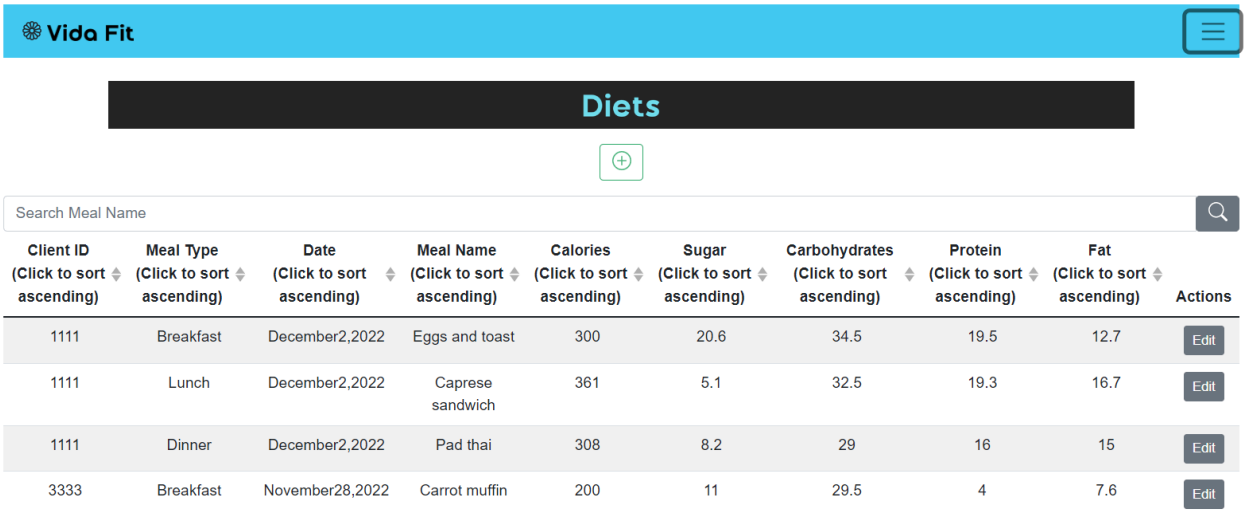

Figure 12: Progress Page

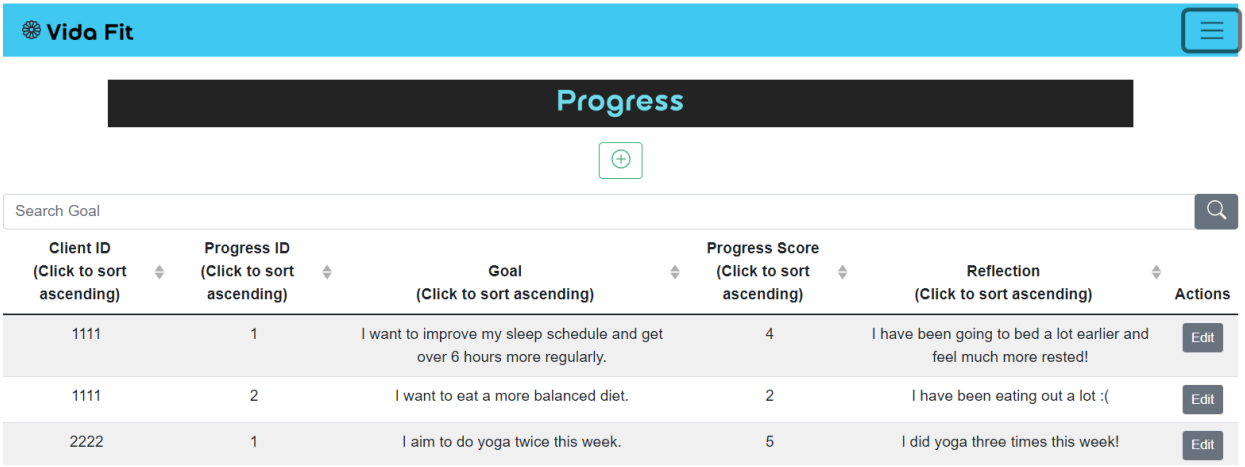

Figure 13: Sleep Page

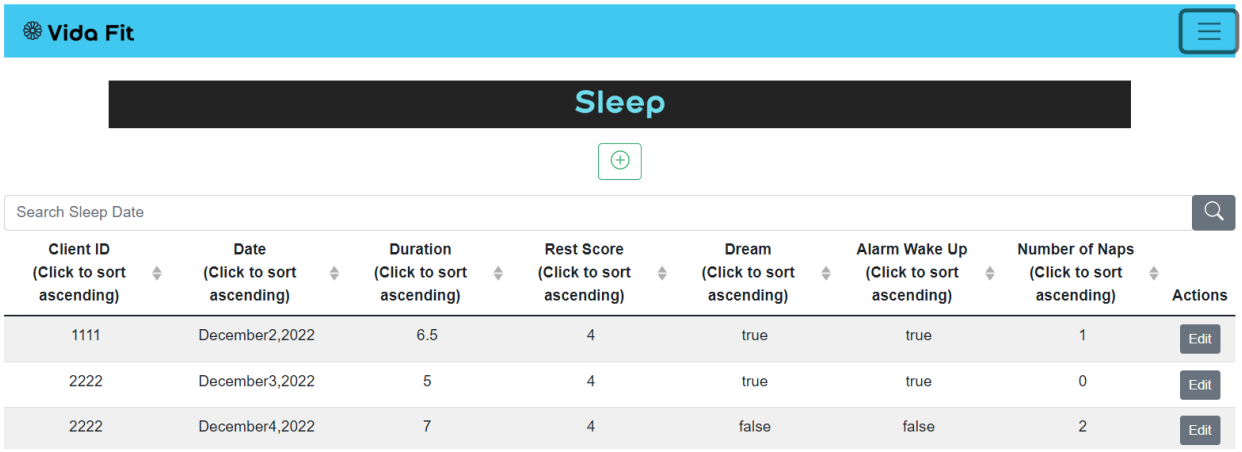

Figure 14: Resources Page