CMP-5012B Software Engineering I

Group Project Stage 1 – Report

https://bitbucket.org/zmf18gwu/cmp-5012b-part1/branch/ToBeMarked

Design and implementation of a health tracker application.

Group Number: 12

Date: Last edited 17/03/2020

Group Members: Jonathan (Buzz) Embley-Riches 100237137

Linfeng (Chris) Chen 100254216

Ranhui (Harris) Zhang 100256949

James Burrell 100263300

1 Problem Description.

1.1 Main objective and overview.

Our initial proposal for this coursework assignment was to choose the scenario in which we were to develop a health tracker application for use by a large health insurance company. The main goal of the application would be to allow the user to track information on their diet and fitness regime. This would then be processed by the user would be given simple goals and a view of their habits over time. It would also present basic health information, with the presumed goal of making the user more healthy, and in doing so make less health-related claims to the insurance company. The application should also have the potential to be integrated to be offered alongside an insurance product.

However, we decided we would follow a different scenario and develop a more full-featured application, focused only on providing the user with a health tracker, with no relation to it being used by an insurance company. In doing so, the requirements for providing basic health information and potential integration with the insurance company would be ignored and replaced with adding more detailed and varied goals, as well as better tracking ability. From this we will be developing a stand-alone health tracker application, taking inspiration from existing systems like Samsung Health[1] and MyFitnessPal[2].

With this new problem in mind, we will be carrying out and documenting a software development process, with the final product being a fully working browser based health tracker application and a full written document describing the development process. Following the coursework guidelines, we will be conducting initial research into the problem area where we analyse current solutions and their features, and using this to produce our own set of requirements for our application. After this, user focused development techniques will be used to produce a set of user stories, personas and use case descriptions to better guide the development. Object-oriented analysis (OOA), object-oriented design (OOD) and Model-View-Controller (MVC) design will be used to lay out the foundation of how our program is to be implemented. We will produce an initial desktop prototype, and later make a full web-based version. We will also be using an iterative development strategy, Scrum, to aid in the project.

The first deliverable of the assignment will be programmed using the JavaFX software platform, alongside a PostgreSQL database to store user data. The second web-based application using Python's Django framework.

[1] https://www.samsung.com/uk/samsung-health/
[2] https://www.myfitnesspal.com/

1.1 Analysis of similar systems.

We have selected 3 current health and fitness apps available to the public to analyse. We have preliminarily identified their key features in order to see which of them are common to all 3. We then went on to pick out features that were possible to implement in our own solution, and then features we could include if needed.

Application name.	Key features.	Key features worth including in our own application.	Key features we could include.
MyFitnessPal[3]	• Food database. • Barcode scanner. • Restaurant logger. • Food insights (healthy eating guidance). • Calorie counter. • Macronutrients tracker. • Food diary. • Fluid intake tracking. • Weight goals. • Calorie goals. • Exercise logging. • User groups. • Progress reports. • Device connection.	• Food database. • Calorie counter. • Food diary. • Fluid intake tracking. • Weight goals. • Calorie goals. • Exercise logging. • User groups. • Progress reports.	• Macronutrient tracker.
Google Fit[4]	• Step tracker. • Heart rate monitor. • Workout tracker. • Goal monitoring. • Fitness coaching • User groups. • Fitness reports. • Device connection.	• Workout tracker. • Goal monitoring. • User groups. • Fitness reports.	
Samsung Health[5]	• Sleep tracking. • Heart rate monitoring. • Blood glucose tracking. • Caffeine intake tracking. • Device connection. • Automatic workout tracking. • Cross device syncing. • User groups. • Challenges and milestones.	• Caffeine intake tracking. • User groups. • Challenges and milestones.	

As a brief note, many features in these apps that are common to all three have not been included in the *"Key features worth including in our own application."* column. For example, all three apps have the ability to connect with devices such as smart watches to track heart rate. Due to the nature of the assignment, we will not be including features like this, and will be discussed more clearly in the MoSCoW section.

In the next part of the analysis, we will discuss the main features of these 3 applications in order to get a general view of the applications and how they are related to our project.

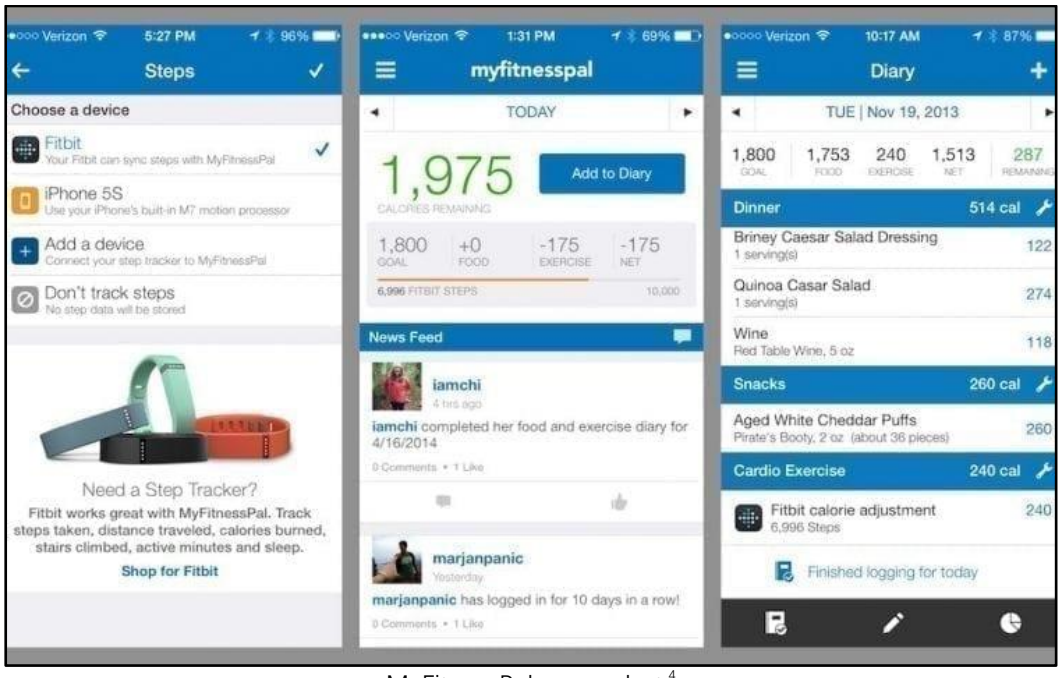
MyFitnessPal

MyFitnessPal is one of the most popular web-based exercise and fitness social media applications[3]. It helps you keep track of users' daily food intake, calculating nutrition, calories and vitamins contained in the food. This also helps users to find out what their diet is missing or what to reduce from it.
Other than many outstanding recipes, fitness training guides and lots of health tips, MyFitnessPal can also connect to many other training tracker devices or mobile apps, for example, Fitbit, Lumo Lift and Polar Loop. These devices track users' steps, calorie consumption and active times during a day.

MyFitnessPal has a huge food database that contains more than 11 million foods including global items and cuisines. Other advanced functions are barcode scanners and recipe importers. It can scan barcodes, save meals and recipes, and use 'Quick Tools' for fast and easy food tracking. With the recipe importer, users can easily import the nutritional information for the recipe other users cook. MyFitnessPal also automatically calculates the calories in the foods, meals and recipes with its calorie counter. Its fluid intake tracker also saves users' recently logged amounts of consumed water.

MyFitnessPal provides community functions like Facebook. Users can find and follow their friends, share their exercises or daily diets, and see friends' progress. With joining the community, users can find motivation, support, tips, and advice in our active forums.

The application can chart users' progress and view nutrition reports of calories, macros, and nutrient intake.



MyFitnessPal screenshot.[4]

[3] https://www.dummies.com/health/exercise/what-is-myfitnesspal/
[4] https://blog.myfitnesspal.com/now-you-can-track-your-steps-in-myfitnesspal/

Google Fit

Google fit is a health tracker app that collects users' biometric stats and daily activities to manage users' health and fitness. It has easy-to-use user interface and simple look.

Google Fit uses "move minutes" and "heart points" to track user's activities, thus motivating users to do more activities. It checks "move minutes" and "heart points" to see if the user meets their goal each day, and helps the user adjust their goal progress so users can keep challenging themselves. It helps users get customized tips and actionable coaching based on the user's activity.

Google Fit allows users to get connected with other mobile apps and devices, to give the user a holistic view of the user's health in order to keep track of the user's progress.

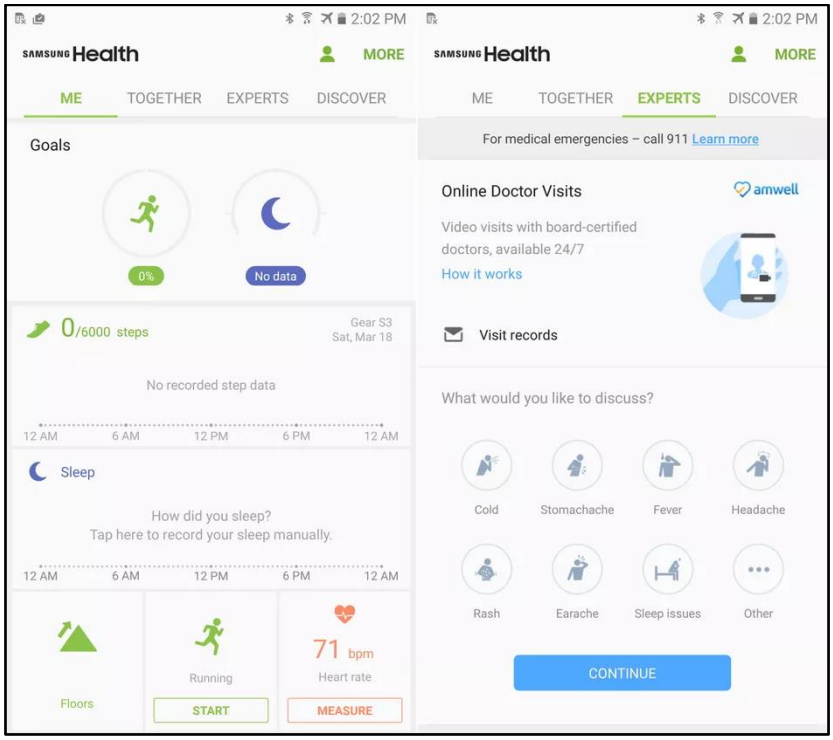
GoogleFit screenshot.[5]

[5] https://www.techcommuters.com/google-fit-review/

Samsung Health

Samsung Health provides core features to help users to keep fit and healthy. It records and analyzes users' daily activities and habits to help maintain a successful diet.

Samsung Health records various bits of information like food, caffeine and water intake details to help users create a balanced lifestyle. It allows users to compete with their friends and check their rank. If connected with smart phones with the relevant sensors, Samsung Health can measure temperature and humidity, UV intensity and heart rate.



Samsung Health screenshots.[6]

[6] https://www.cnet.com/how-to/tips-and-tricks-to-get-the-most-out-of-samsung-health/

1.3 Feature matrix of similar systems.

Key feature.	Our application.	MyFitnessPal	Google Fit	Samsung Health
Login system.	✓	✓	✓	✓
User health information record.	✓	✓	✓	✓
Step tracker.	✓	✓	✓	✓
Water tracker.	✓	✓	✕	✓
Recipe importer.	✓	✓	✕	✓
Food diary.	✓	✓	✓	✓
Weight goals.	✓	✓	✕	✓
Calorie goals.	✓	✓	✕	✓
Exercise tracker.	✓	✓	✓	✓
Exercise goals.	✓	✓	✓	✓
Email communication.	✓	✕	✕	✕
Food database.	✓	✓	✕	✓
Barcode scanner.	✕	✓	✕	✕
Calorie counter.	✓	✓	✓	✓
Coaching.	✓	✓	✓	✓
User group.	✓	✓	✓	✓
Sleep tracker.	✓	✓	✓	✓
Tracking sync between multiple devices.	✕	✓	✓	✓

1.4 MoSCoW Analysis

Must Have.	Should have.	Could have.	Won't have.
User Accounts	Food database	Macro counter	Barcode scanner
Personal information	Calorie counter	Sleep counter	Device connection
Exercise capture	Water tracking		Heart rate counter
Diet capture	Coaching		Stress tracker
Goal capture			Glucose counter
Regular weight capture			Automatic workout tracking
User groups			
User reports			
Group reports			

This MoSCoW analysis has identified the crucial features our solution must implement. Without these features, we believe the application would be a poor solution to the problem presented in section 1.1, as it does not provide the basic functions a health tracker application needs to. In addition, without these features the solution would not be inline with the coursework specification, and so would result in a poor mark. As for features the application should and could have, they have been placed there as they extend the function of our app, but don't necessarily prevent it's main function if they were not included.

For example, implementing a food database and in turn a corresponding macronutrient counter would be useful to provide detailed information on a users meal. However, it then presents the problem of; where would we source this amount of data in the given time to implement our solution? Because of this we have placed them in '*Should*' for food database, and '*Could*' for macro counter, as it would require more effort to implement. The feature of sleep counter/tracking was placed in the 'Could' column for a different reason. At this stage in the project, we think that data will be stored alongside a specific date in a similar manner to a dictionary, and because of this implementing a sleep tracker would be problematic as sleep durations typically cross over two dates, so may require extra logic work.

Finally, the '*Won't*' column contains features that simply will not be implemented because of the resources we are given and also what is expected of the solution in the context of the scenario. For instance, all of the features in that column require external hardware in the form of sensors or input devices to work, which are unavailable. Whilst these features are often present in the applications we analysed, our requirements from the scenario do not need us to produce as complex or full featured app as those.

2 User stories and use case.

2.1 User stories.

User account create.	#1	HIGH
As an end user.		
I want to be able to create an account by entering my email address, password and then all of my personal information.		
So that I can have a personal account tailored to me, that allows me to access the app without having to re-enter all my details every time.		

User account login.	#2	HIGH
As an end user.		
I want to be able to login to an already created account.		
So that I can have access to all the apps features and all my personal information.		

Navigate to another tab.	#3	HIGH
As an end user.		
I want to be able to click on a menu.		
So that I can easily navigate the various sections of the app.		

Exercise tracker.	#4	HIGH
As an end user.		
I want to be able to view any of my past exercises I have logged.		
So that I can keep up to date with my routine workouts and make sure that I am always improving myself.		

Sleep tracker.	#5	HIGH
As an end user.		
I want to be able to input the amount of sleep I have had on a certain day.		
So that I can monitor the amount of music I've had over a certain amount of time.		

Fluid intake tracker.	#6	HIGH
As an end user		
I want to be able to input the amount of fluid I have consumed in a day.		
So that I can track that I am drinking a healthy amount each day.		

Regular weight tracker.	#7	HIGH
As an end user		
I want to be able to constantly check my weight and be notified when I have to check my weight.		
So that I can check my weight against my current weight goal and view the progress I have made.		

Exercise tracker.	#8	HIGH
As an end user.		
I want to be able to enter any exercise that I participate in, and the duration of that exercise.		
So that I can track my exercise progress, to make sure I stay active to help increase my health.		

Diet input.	#9	HIGH
As an end user.		
I want to be able to enter all the food I consume a day in a diet diary.		
So that I can track my daily food intake, to help with dieting and general healthy eating.		

Goal input.	#10	HIGH
As an end user.		
I want to be able to set different types of goals for myself for different parts of the app.		
So that I can have clear goals that I need to complete, which will motivate me to get healthier.		

User group creation.	#11	MED
As an end user.		
I want to be able to create a new group with other users.		
So that I can begin to share information and work towards a group goal.		

User group joining.	#12	MED
As an end user.		
I want to be able to join a pre-existing		

User reports.	#13	MED
As an end user		
I want to be able to see reports generated about my information for various app sections.		
So that I can keep up to date with my progress.		

Group reports.	#14	MED
As an end user		
I want to be able to see my user groups current progress towards their goal.		
So that I can keep up to date with my progress compared to my user groups progress and provide motivation to members who haven't reached their goals.		

Food database.	#15	MED
As an end user		
I want to be able to input the food that I consume during the day.		
So that I can save the information for faster input another time.		

Calorie counter.	#16	MED
As an end user		
I want to be able to track the amount of calories I consume on a daily basis and give me an estimate of the total.		
So that I can keep a record of how much I consume and change my diet based on the results.		

2.2 Use case.

USE CASE NAME	**User Account Create**	
Goal in Context	To allow a user to create an account, which will allow them to enter personal information and have a place to login and securely access all the features of the app.	
Scope & Level	Create an account system.	
Preconditions	System is running and the user account file is loaded.	
Success End Condition	A new user account is created.	
Failed End Condition	Invalid account creation.	
Primary Actor	User	
Trigger	Actor selecting “Sign up”.	
SUCCESS SCENARIO	**Step**	**Action**
	1	The system displays a window requesting log in or sign up.
	2	The actor selects the sign up option.
	3	The actor enters an email and password they would like to use for their account.
	4	The system verifies that the email is the correct format and passwords match.
	5	The actor enters all their personal information requested from the system.
	6	The system verifies the entered information is valid.
	7	The system stores the information and confirms, then displays the main window
ALTERNATIVE SCENARIO	**Step**	**Branching action**
	4a	The system displays an error message, and a helpful tip. Asks for re-entry.
	6a	The system displays an error message, and a helpful tip. Asks for re-entry.
RELATED INFORMATION		
Priority	Top priority	
Performance Target	-	
Frequency	Infrequent	

Subordinate Use Cases	-
Channel to Primary Actor	User interface.
Secondary Actors	-
Channel to Secondary Actors	-
OPEN ISSUES	Verifying that the email is live and valid?
SCHEDULE	Due date is version 1.0 release
AUTHOR	Linfeng Chen, Ranhui Zhang, James Burrell, Buzz Embley-Riches.

2.3 Personas

We have developed these personas for use throughout the development process. When a feature is considered, the personas will be referred to to see how they would interact that said feature.

Persona 1

Name: Dan Johnson
Age: 25
Sex: Male
Occupation: Office worker
Family: Lives with Girlfriend

Likes: Staying fit and healthy, Enjoying food and social events.
Dislikes: Managing his diet, keeping track of how much exercise he does.

Biography: Dan likes to stay fit and be in full control of his life. He lives a busy life, juggling his work life, his social life with his Girlfriend and his need to stay fit. Dan likes to find ways to help manage his time, he is an avid app user and loves how they can help him out in his day to day life.

Persona 2

Name: Emma Rutherford
Age: 58
Sex: Female
Occupation: Works in a Cafe
Family: Lives home alone

Likes: Walking her dog, chatting with friends and walking marathons.
Dislikes: Overly complicated technology, giving out her personal information to strangers.
Biography: Emma is a strong willed walkathlon participant. She loves walking and strives to do it everyday, whether it's with her friends, her dog or participating in charity events, she is always trying to

improve her speed. Emma is not great with technology but still tries to use it as she can see the benefits of it but due to her old age she is no professional.

Persona 3

Name: Ethan Geary
Age: 22
Sex: Male
Occupation: Full time student

Likes: Drinking at the pub, heavy gym sessions and playing computer games.
Dislikes: Old outdated technology, apps full of bugs and unhealthy eating.

Biography: Ethan loves having a good time and trying to look after his body when he is not drinking. He is a strict gym attendee and tries to workout as much as possible between going to university and going out drinking. Sometimes he finds it hard to keep track of his workout times and his eating schedule.

3 Object Oriented Analysis (OOA)

3.1 Initial Development

Using slides 24 and 25 from lecture two on OOA, we used a similar object modelling technique by Loomis and Rumbaugh[7] to come up with the basis for our object oriented design. The steps below show the stages to this process.

Stage 1

From our list of user stories, we identified the 'nouns' that would be needed to form our initial classes. Below you can see an example of a user story taken with the 'nouns' that would later need to be classes identified.

Exercise tracker.	#8	HIGH
As an end user.		
I want to be able to enter any exercise that I participate in, and the duration of that exercise.		
So that I can track my exercise progress, to make sure I stay active to help increase my health.		

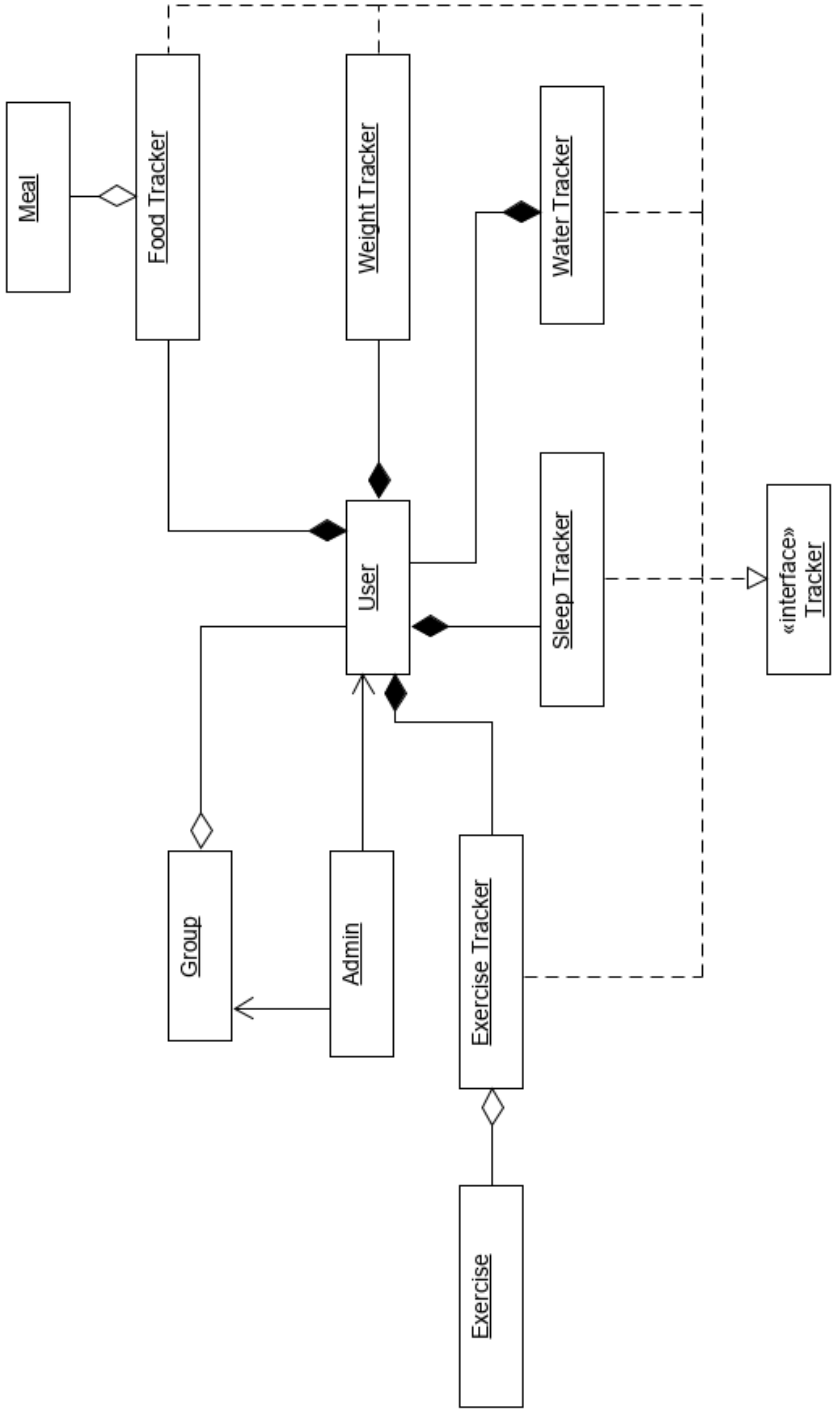
As you can see from this highlighted use case, 3 clear classes can be seen from this, ExerciseTracker, User and Exercise. We repeated this process for the rest of the user stories and came up with a more comprehensive list.

Noun	Class Purpose
User	Stores individual information for each person using the application, as well as any methods that use this to generate further data, ie BMI. Allows the same instance of the application to be used by many users.
Exercise	Stores information about a particular exercise that a User has completed, like its type, duration, start and end times.
Meal	Stores information about what and when someone a User has eaten. Will likely include a list of foods the user can add to and their calorie count.
Group	Stores information on a joinable Group. Name of the Group, it's members, it's admin and the Group goal will be stored.
Tracker (Water, Exercise, Food, Weight, Sleep)	Stores information on a User's involvement with each aspect of their health. (Water, exercise, food, weight and sleep). Will be divided into separate classes for each one. Each tracker contains a HashMap with a date then what happened on that date, for example, ran 1km.
Admin	Will be a class used for initialization and management of the users profile, but will remain hidden to the user.

[7] *M.E.S. Loomis, A.V. Shah, J.E. Rumbaugh. An Object Modelling Technique for Conceptual Design. ECOOP '87 Proceedings of the European Conference on Object-Oriented Programming Pages 192-202, 1987.

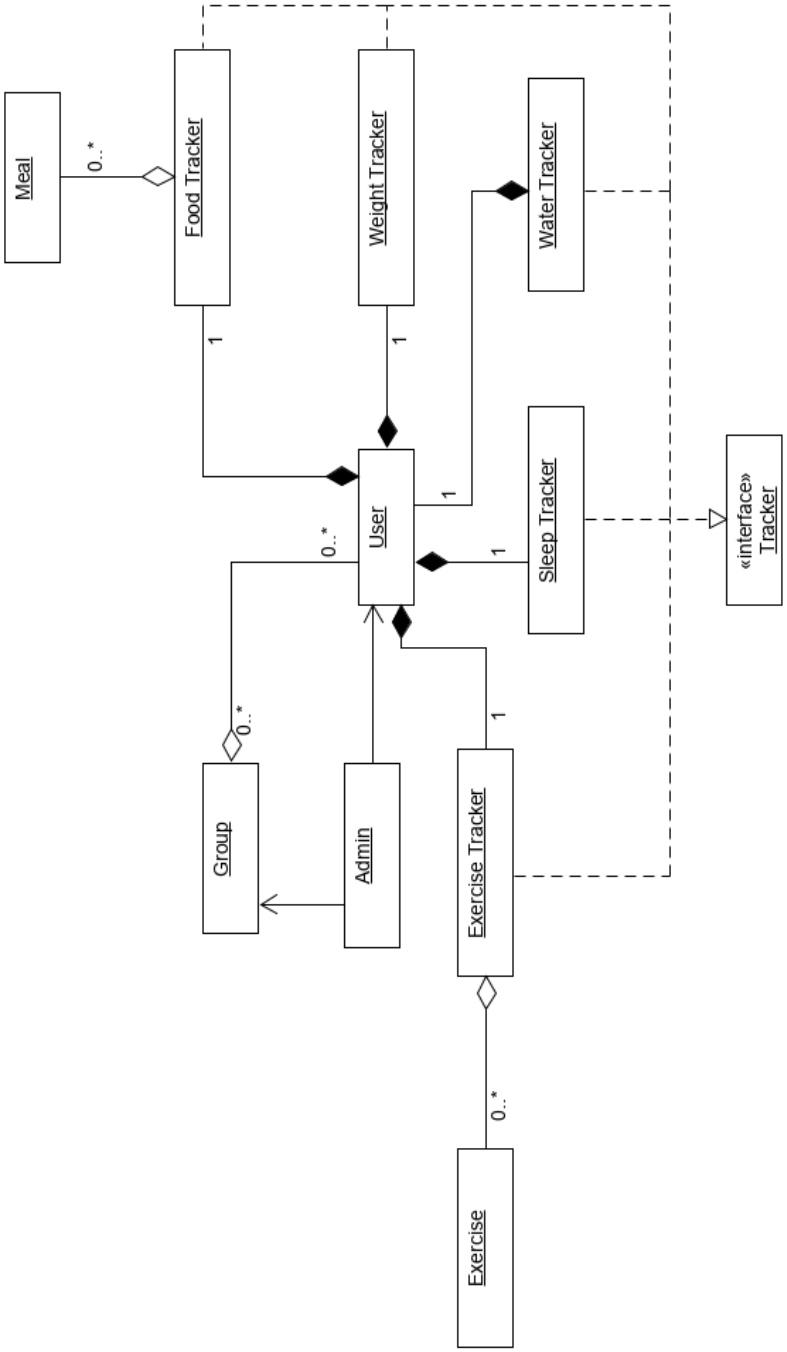
Stage 2

We then looked at the relationships between each class and how they would interact, and produced this diagram.



Stage 3

After the relationships were identified, we then added their expected cardinalities. The diagram below shows our final high level diagram for the Model part of our program.



3.2 Explanation of Class Relationships

Our application is based on the idea that only one person can use only one instance of the application at a time. For each user, a tracker for each aspect of health monitoring is made. This means one tracker for Weight, Sleep, Water Intake, Exercise and Food Consumption, for each user. The User class is composed of instances of these 5 trackers, hence the composition relationship indicated by the black diamond in the diagram. Because these trackers all hold similar functionality, we have decided to define a Tracker interface with a set of methods that all trackers must have but can be modified to suit their needs.

A User will have the ability to join many Groups, and a Group will contain many Users, showing an aggregation relationship. A User also has the ability to not be part of a Group, and a Group can contain no members, hence the cardinality of this relationship is 0...n on both sides.

Finally, the Exercise and Food trackers allow the User to store information on many different Exercises and Foods respectively, hence another aggregation relationship with a cardinality of 0...n is shown.

4 Object Oriented Design

4.1 Initial Design

After our OOA stage, we began to define the most obvious attributes and methods that would be needed for the main functionality of our application. Once we began to implement these classes, it became aware to us that many more classes would have to be added in order for the full range of requirements to be implemented. Because of this, we often returned to the UML diagram to modify, add and remove elements.

This is a link to the final UML class diagram for the Model part of our implementation.

4.2 MVC Diagram

Our MVC architecture allows the three separate parts of our program to not depend heavily on one-another. This means the program is loosely coupled meaning each element can be modified during development with little to no consequences on the other elements. It will also allow us to clearly lay out our code so it is more readable and therefore easier to build upon. The diagram below shows our MVC UML diagram. The methods, attributes and detailed relations have been removed for visual clarity, as this diagram is meant to show nothing more than our MVC architecture.

Model-View-Controller UML class diagram.

5 Initial Implementation

5.1 Basic Information

The final implementation of our Health Tracker application was implemented using the Java programming language, with the JavaFX software platform used to produce the user interface. User data from our application is stored in a postgreSQL database, which can be viewed and edited by a developer using pgAdmin 4[8]. The database communicates with our application using the JDBC[9] API.

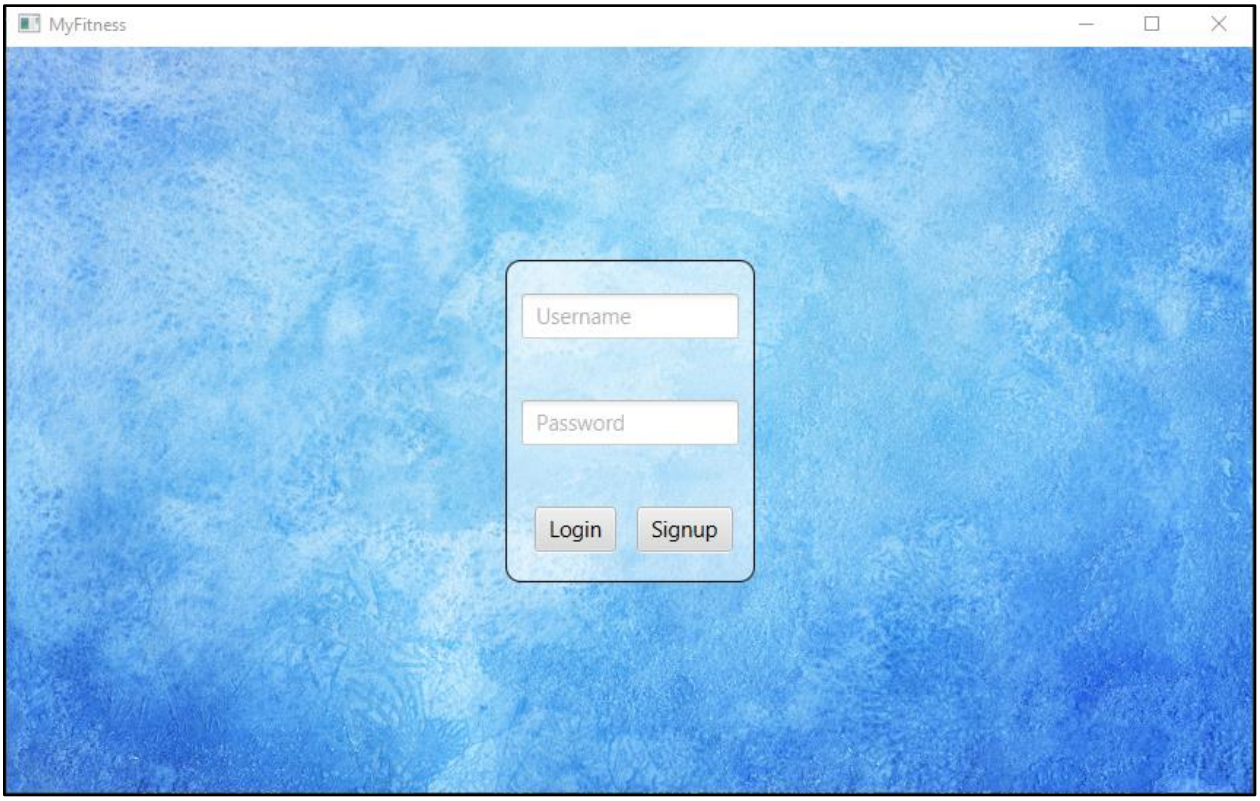
The application was programmed entirely using the IntelliJ IDE[10] from JetBrains and the database was created using pgAdmin 4.

[8] https://www.pgadmin.org/
[9] https://docs.oracle.com/javase/tutorial/jdbc/basics/gettingstarted.html
[10] https://www.jetbrains.com/idea/

5.2 Functionality and their relations to the requirements.

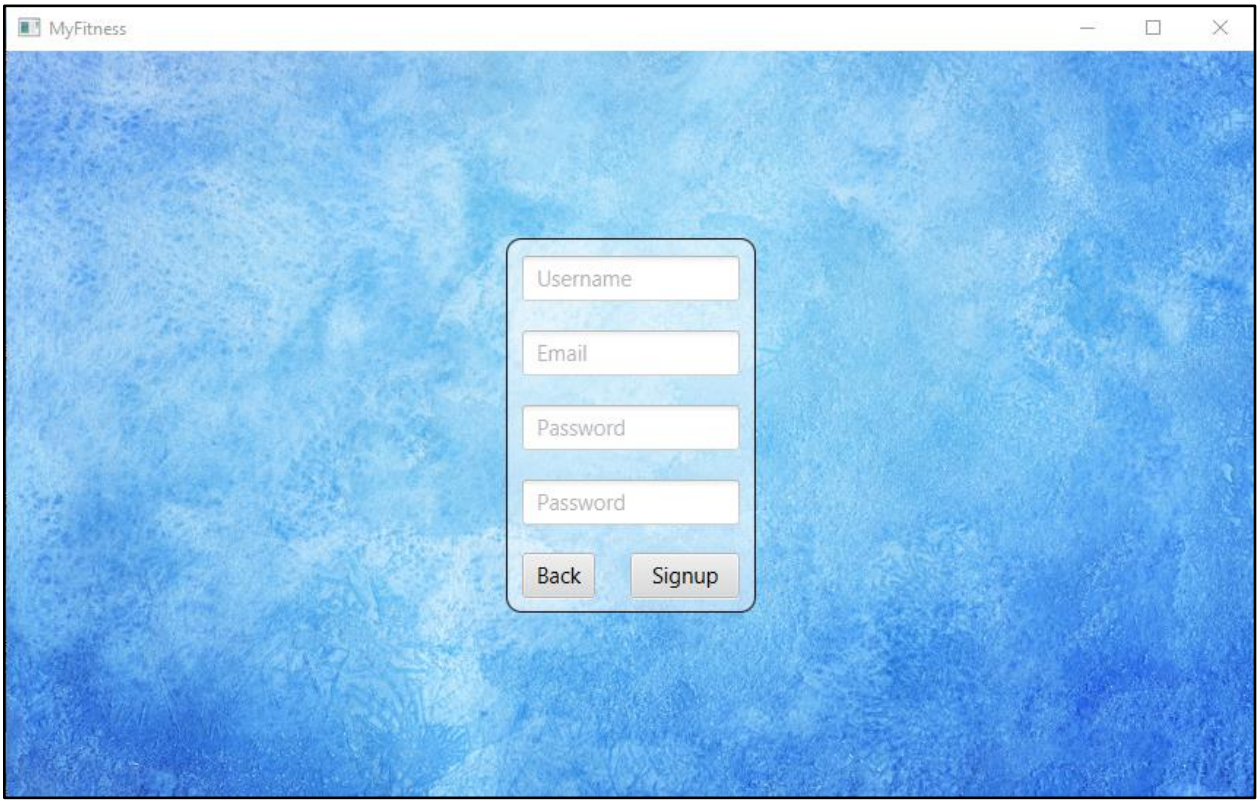
For each feature of our application shown in the screenshots below, the MoSCoW requirement it meets is highlighted in the table below it.

1. When the program is loaded, the user is shown a screen where they can either login, or click a button to sign up.



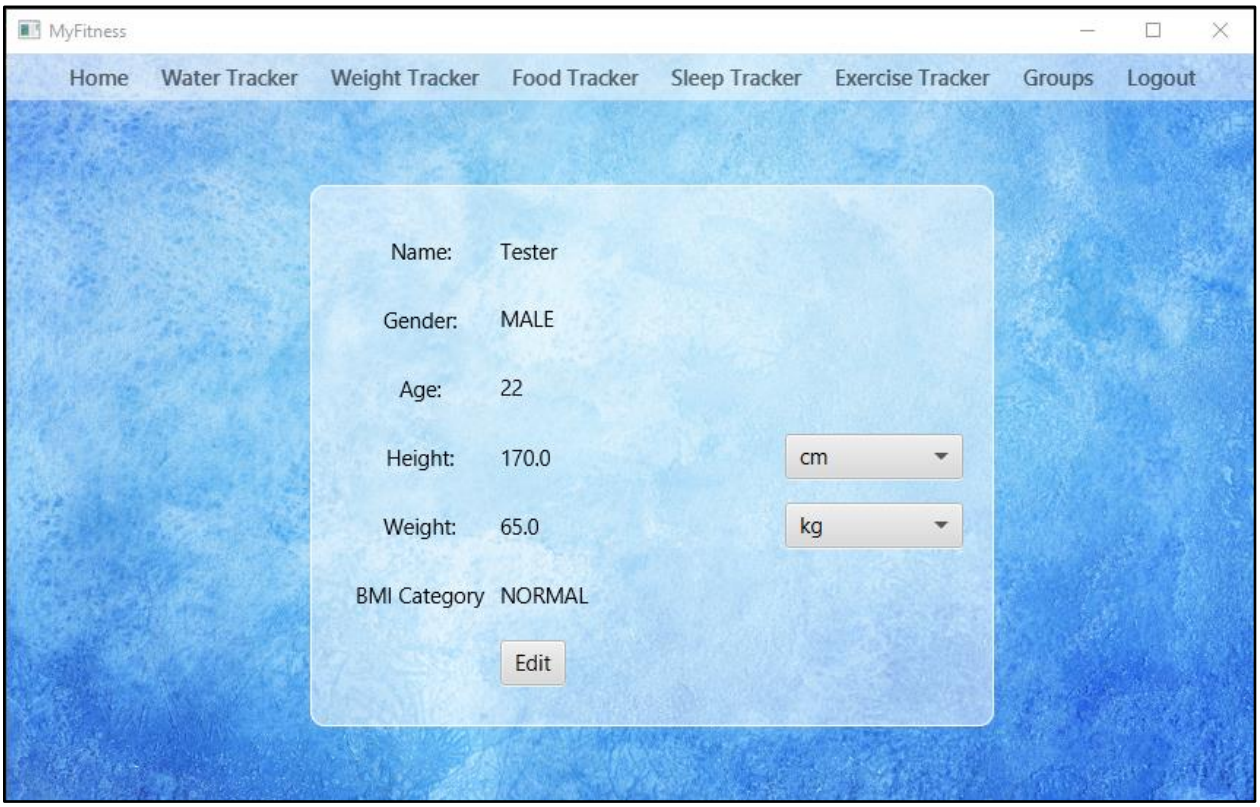
Must Have.	Should have.	Could have.	Won't have.
User Accounts	Food database	Macro counter	Barcode scanner
Personal information	Calorie counter	Sleep counter	Device connection
Exercise capture	Water tracking		Heart rate counter
Diet capture	Coaching		Stress tracker
Goal capture			Glucose counter
Regular weight capture			Automatic workout tracking
User groups			
User reports			
Group reports			

2. When signing up, the user enters their information, it is checked for validity, and if correct the account is created.



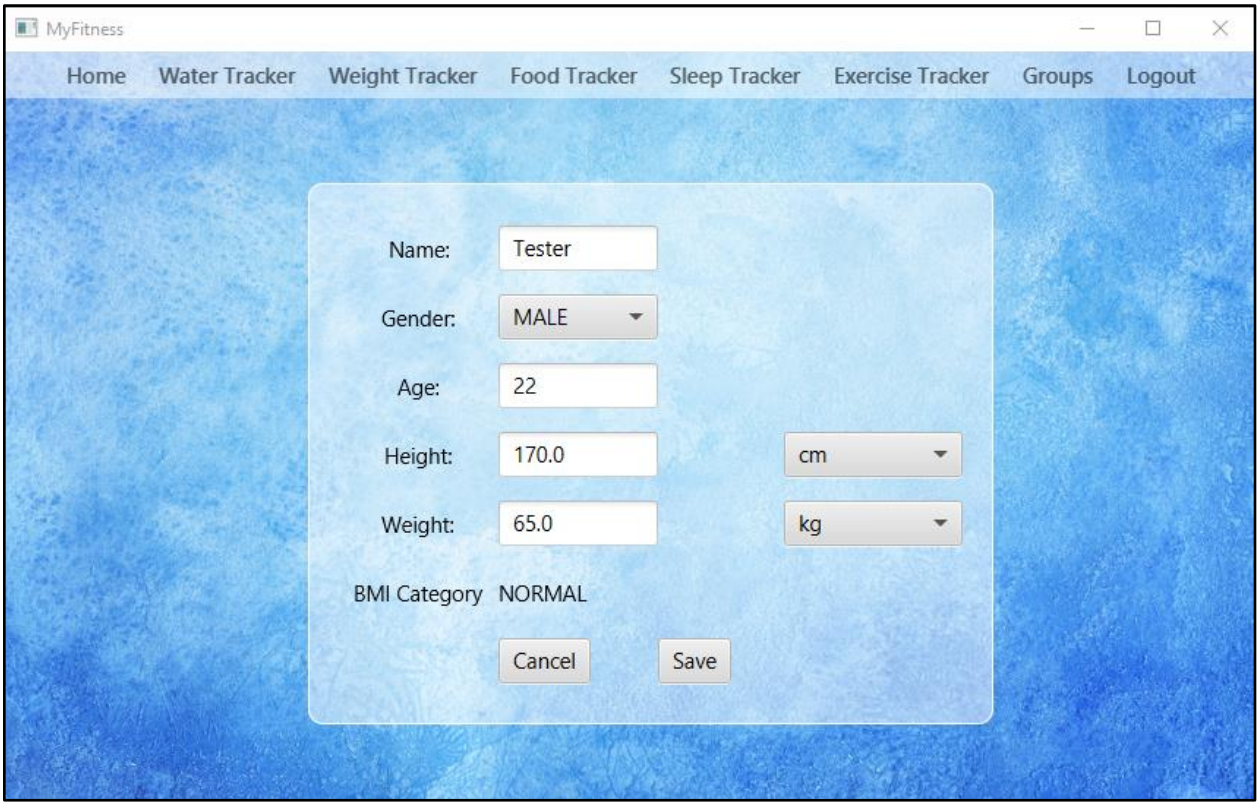
Must Have.	Should have.	Could have.	Won't have.
User Accounts	Food database	Macro counter	Barcode scanner
Personal information	Calorie counter	Sleep counter	Device connection
Exercise capture	Water tracking		Heart rate counter
Diet capture	Coaching		Stress tracker
Goal capture			Glucose counter
Regular weight capture			Automatic workout tracking
User groups			
User reports			
Group reports			

3. The user is initially presented with the home screen, showing their personal information, unit preferences.



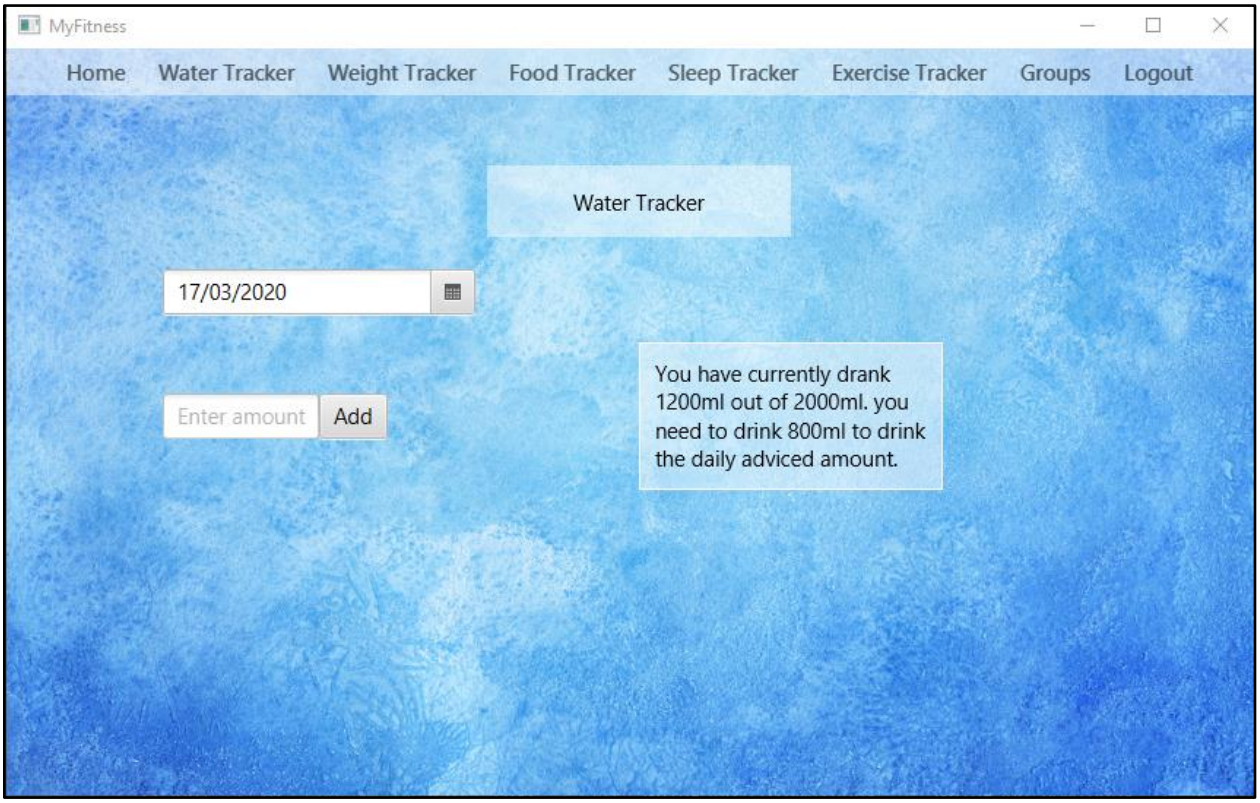
Must Have.	Should have.	Could have.	Won't have.
User Accounts	Food database	Macro counter	Barcode scanner
Personal information	Calorie counter	Sleep counter	Device connection
Exercise capture	Water tracking		Heart rate counter
Diet capture	Coaching		Stress tracker
Goal capture			Glucose counter
Regular weight capture			Automatic workout tracking
User groups			
User reports			
Group reports			

4. The home page has a button that can be pressed in order to edit personal details.



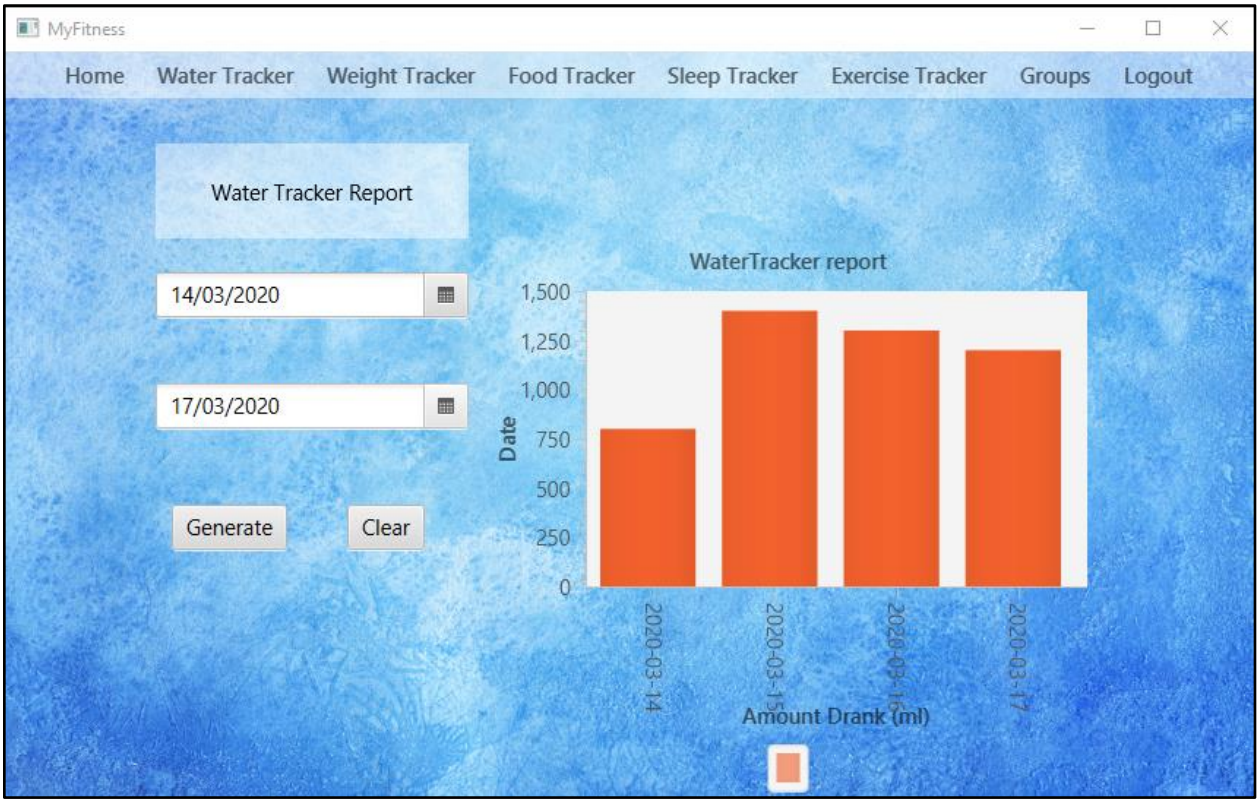
Must Have.	Should have.	Could have.	Won't have.
User Accounts	Food database	Macro counter	Barcode scanner
Personal information	Calorie counter	Sleep counter	Device connection
Exercise capture	Water tracking		Heart rate counter
Diet capture	Coaching		Stress tracker
Goal capture			Glucose counter
Regular weight capture			Automatic workout tracking
User groups			
User reports			
Group reports			

5. The user can enter an amount of water and a date to add to.



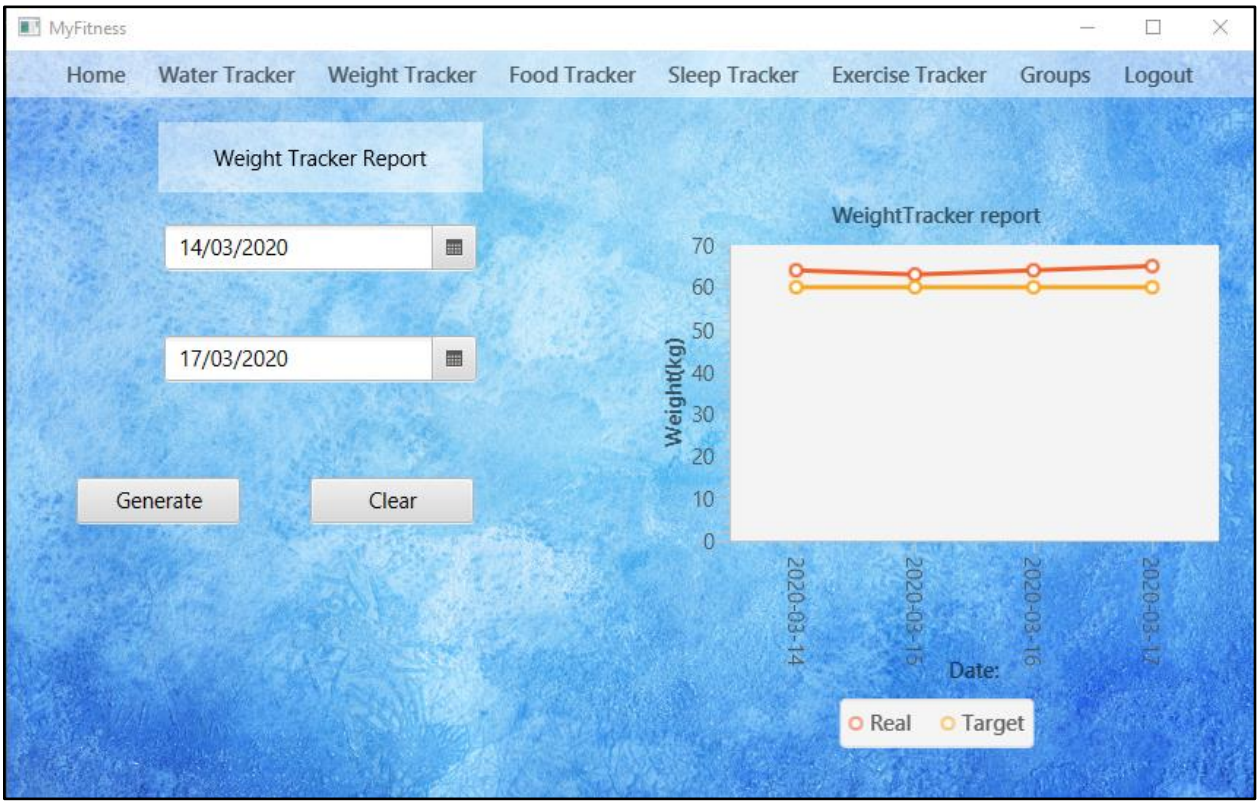
Must Have.	Should have.	Could have.	Won't have.
User Accounts	Food database	Macro counter	Barcode scanner
Personal information	Calorie counter	Sleep counter	Device connection
Exercise capture	Water tracking		Heart rate counter
Diet capture	Coaching		Stress tracker
Goal capture			Glucose counter
Regular weight capture			Automatic workout tracking
User groups			
User reports			
Group reports			

6. A report on the amount of water consumed over two selected dates can be produced.



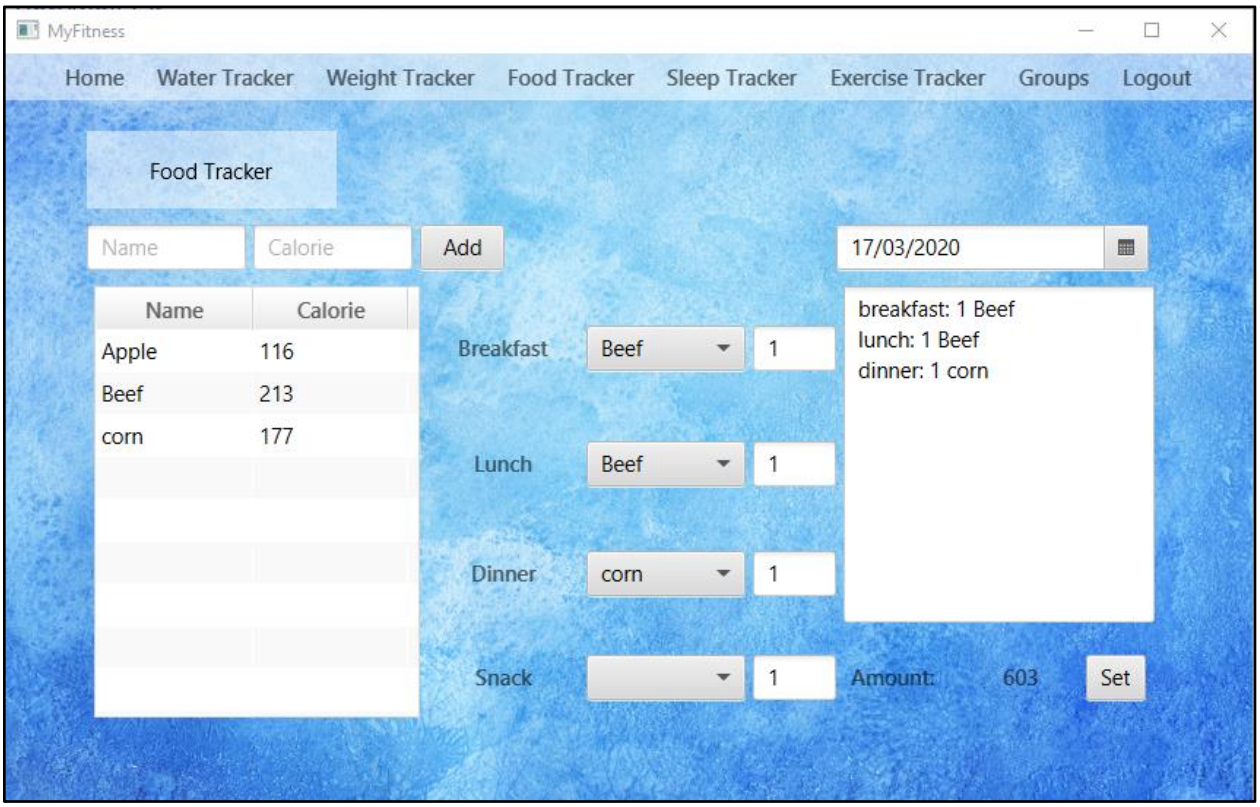
Must Have.	Should have.	Could have.	Won't have.
User Accounts	Food database	Macro counter	Barcode scanner
Personal information	Calorie counter	Sleep counter	Device connection
Exercise capture	Water tracking		Heart rate counter
Diet capture	Coaching		Stress tracker
Goal capture			Glucose counter
Regular weight capture			Automatic workout tracking
User groups			
User reports			
Group reports			

7. The weight tracker tab allows the user to generate a report on how their weight has changed over two inputted dates. (The user can change their weight for the current date on the home page.)



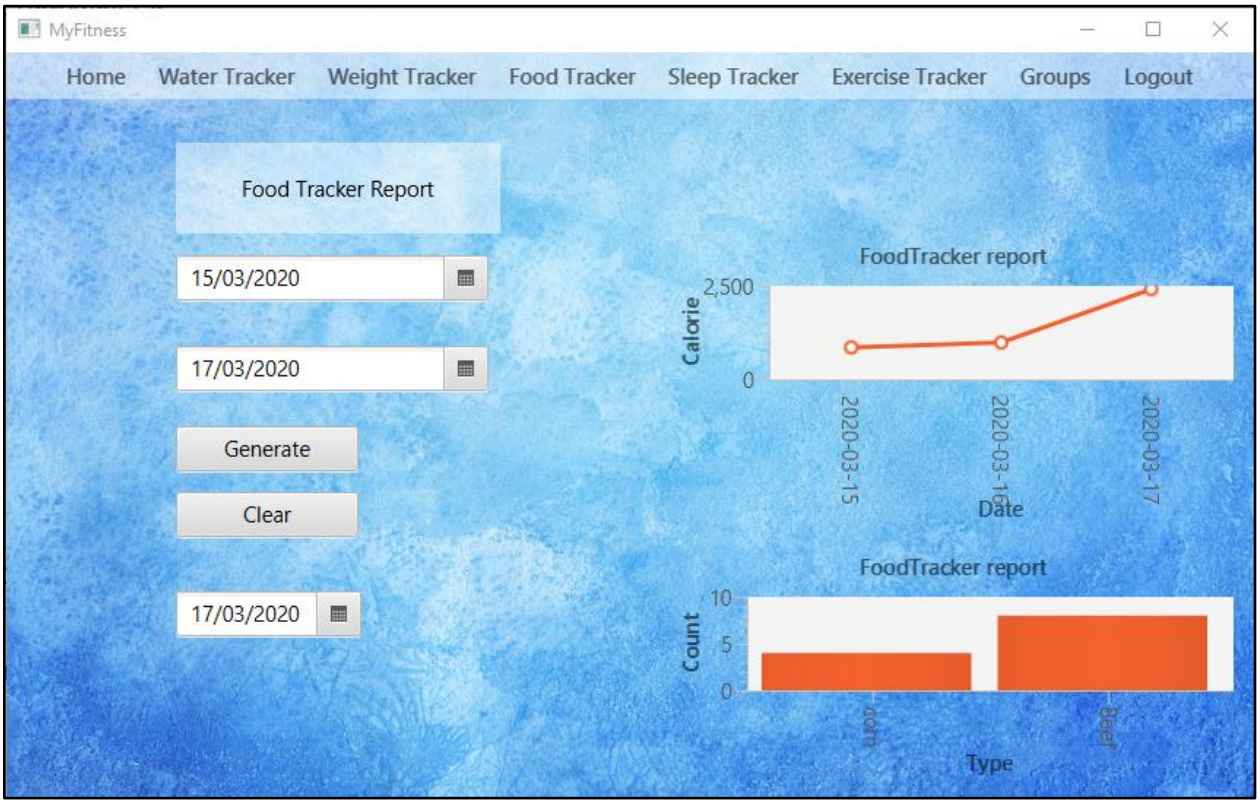
Must Have.	Should have.	Could have.	Won't have.
User Accounts	Food database	Macro counter	Barcode scanner
Personal information	Calorie counter	Sleep counter	Device connection
Exercise capture	Water tracking		Heart rate counter
Diet capture	Coaching		Stress tracker
Goal capture			Glucose counter
Regular weight capture			Automatic workout tracking
User groups			
User reports			
Group reports			

8. Foods can be added to the database along with a calorie value. These foods can be added to meals and the total amount of calories for that meal can be set and recorded for a specified date.



Must Have.	Should have.	Could have.	Won't have.
User Accounts	Food database	Macro counter	Barcode scanner
Personal information	Calorie counter	Sleep counter	Device connection
Exercise capture	Water tracking		Heart rate counter
Diet capture	Coaching		Stress tracker
Goal capture			Glucose counter
Regular weight capture			Automatic workout tracking
User groups			
User reports			
Group reports			

9. The food tracker report tab can be used to generate a report between two inputted dates. This report shows how many calories the user has consumed on each day between the two dates. It also allows the user to see how many of a certain food item was eaten on a specified day.

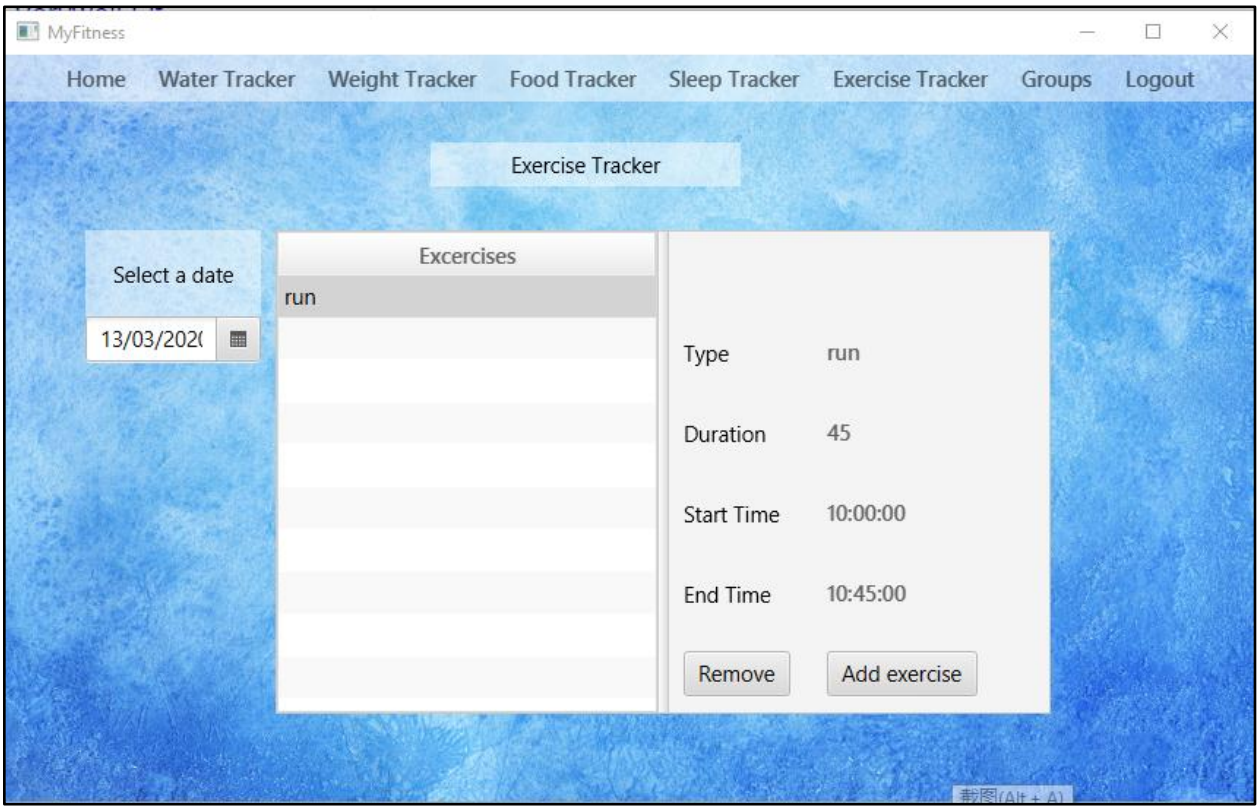


Must Have.	Should have.	Could have.	Won't have.
User Accounts	Food database	Macro counter	Barcode scanner
Personal information	Calorie counter	Sleep counter	Device connection
Exercise capture	Water tracking		Heart rate counter
Diet capture	Coaching		Stress tracker
Goal capture			Glucose counter
Regular weight capture			Automatic workout tracking
User groups			
User reports			
Group reports			

10. The sleep tracker can be used to input how much sleep a user has had on a certain day. They can then see how much sleep they have had between two dates.

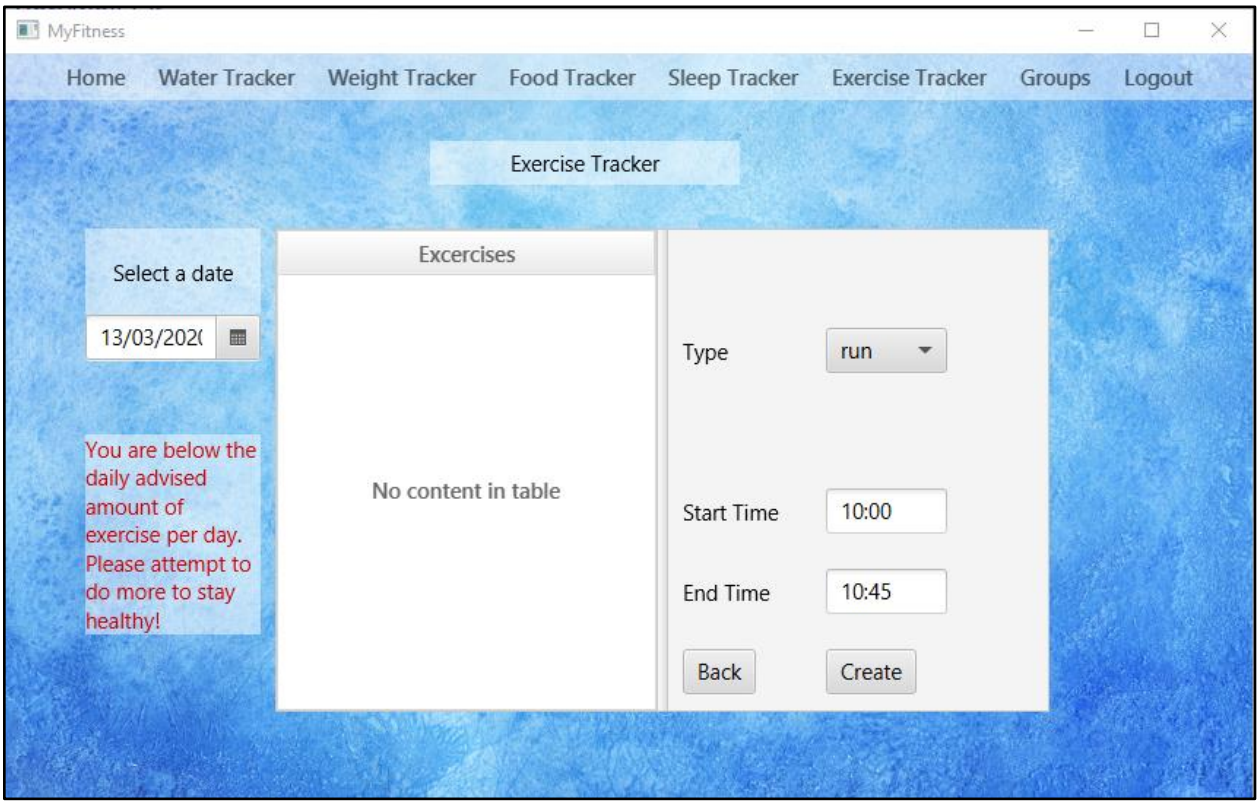
Must Have.	Should have.	Could have.	Won't have.
User Accounts	Food database	Macro counter	Barcode scanner
Personal information	Calorie counter	Sleep counter	Device connection
Exercise capture	Water tracking		Heart rate counter
Diet capture	Coaching		Stress tracker
Goal capture			Glucose counter
Regular weight capture			Automatic workout tracking
User groups			
User reports			
Group reports			

11. Exercise can be logged with an inputted type, start and end time at a certain date. The list of exercises performed on a particular day can be seen.



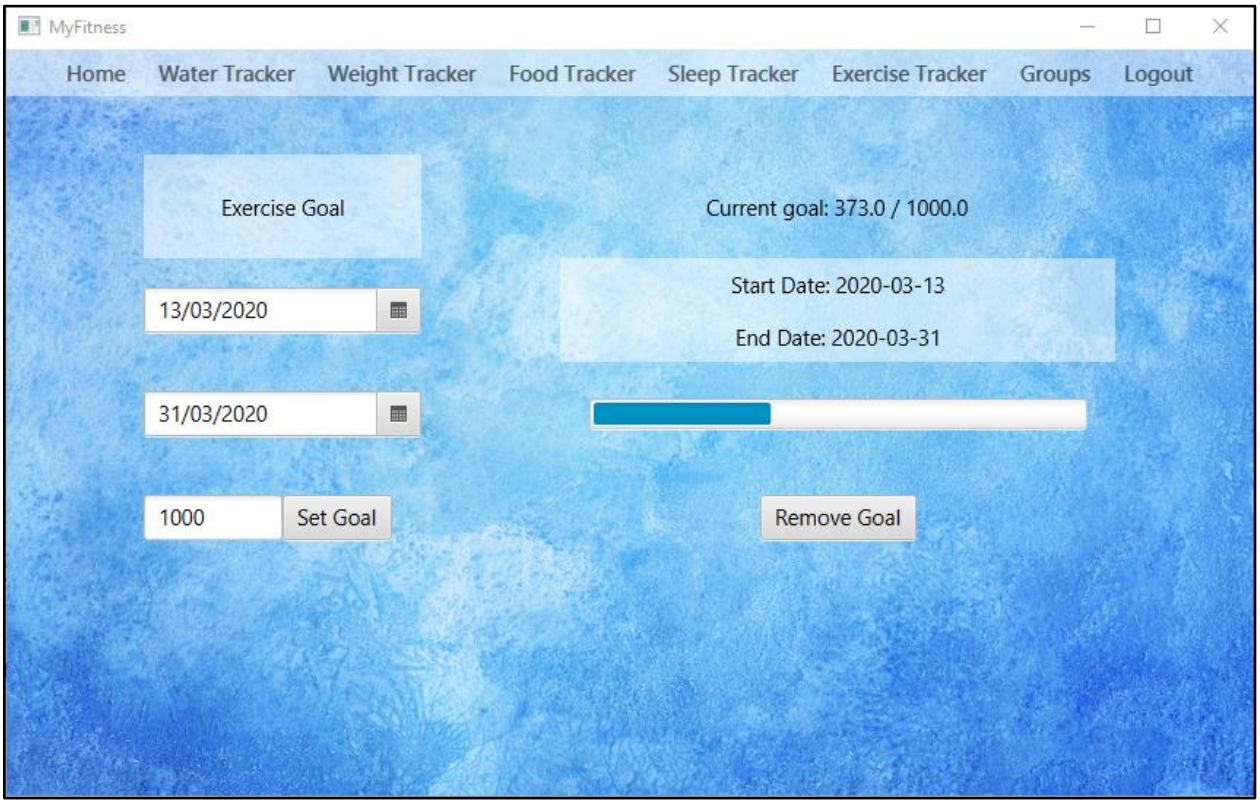
Must Have.	Should have.	Could have.	Won't have.
User Accounts	Food database	Macro counter	Barcode scanner
Personal information	Calorie counter	Sleep counter	Device connection
Exercise capture	Water tracking		Heart rate counter
Diet capture	Coaching		Stress tracker
Goal capture			Glucose counter
Regular weight capture			Automatic workout tracking
User groups			
User reports			
Group reports			

12. If the user is below their recommended daily amount of exercise, a prompt is shown indicating how much more exercise they should do.



Must Have.	Should have.	Could have.	Won't have.
User Accounts	Food database	Macro counter	Barcode scanner
Personal information	Calorie counter	Sleep counter	Device connection
Exercise capture	Water tracking		Heart rate counter
Diet capture	Coaching		Stress tracker
Goal capture			Glucose counter
Regular weight capture			Automatic workout tracking
User groups			
User reports			
Group reports			

13. The user has the ability to enter a goal of how many minutes of exercise they would aim to do between two inputted dates. A progress bar shows how close they are to their goal, and any exercise done between the goal dates is added up and displayed.

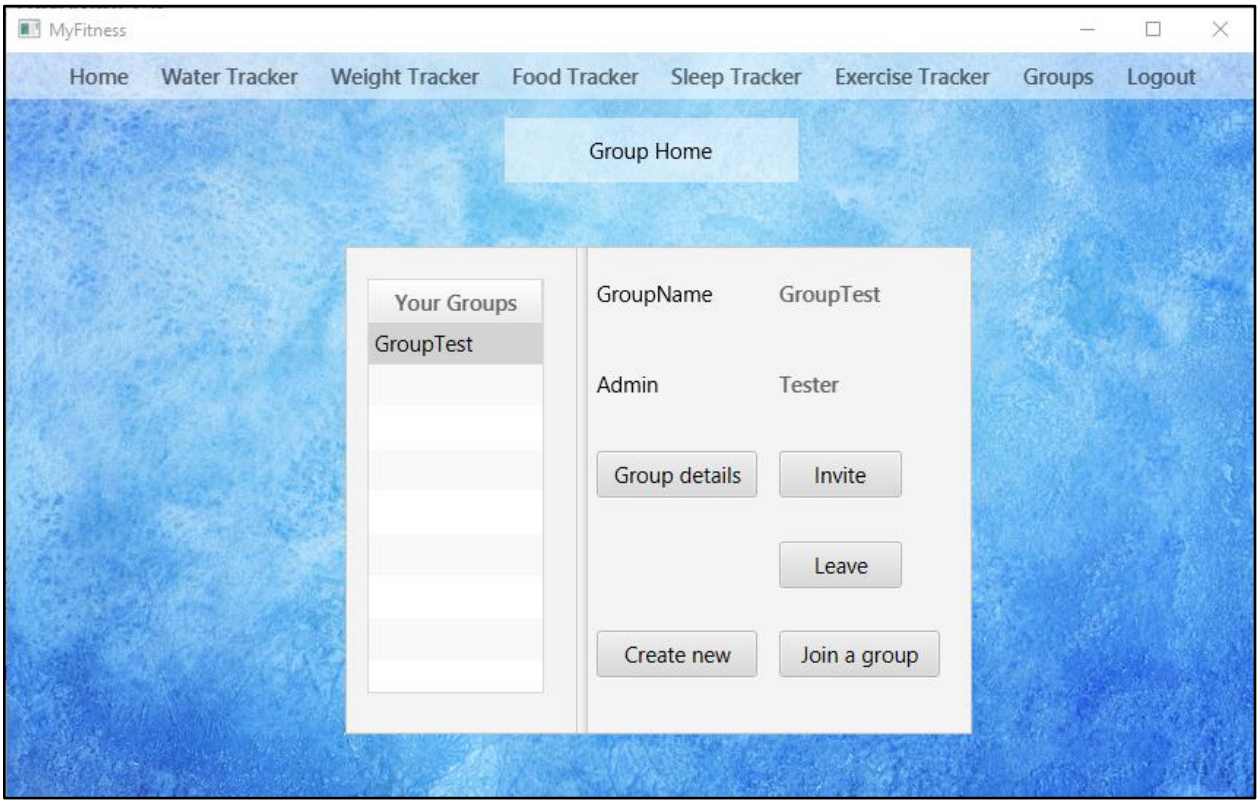


Must Have.	Should have.	Could have.	Won't have.
User Accounts	Food database	Macro counter	Barcode scanner
Personal information	Calorie counter	Sleep counter	Device connection
Exercise capture	Water tracking		Heart rate counter
Diet capture	Coaching		Stress tracker
Goal capture			Glucose counter
Regular weight capture			Automatic workout tracking
User groups			
User reports			
Group reports			

14. Much like the other sections, a report can be generated on how much exercise the user has done between two dates.

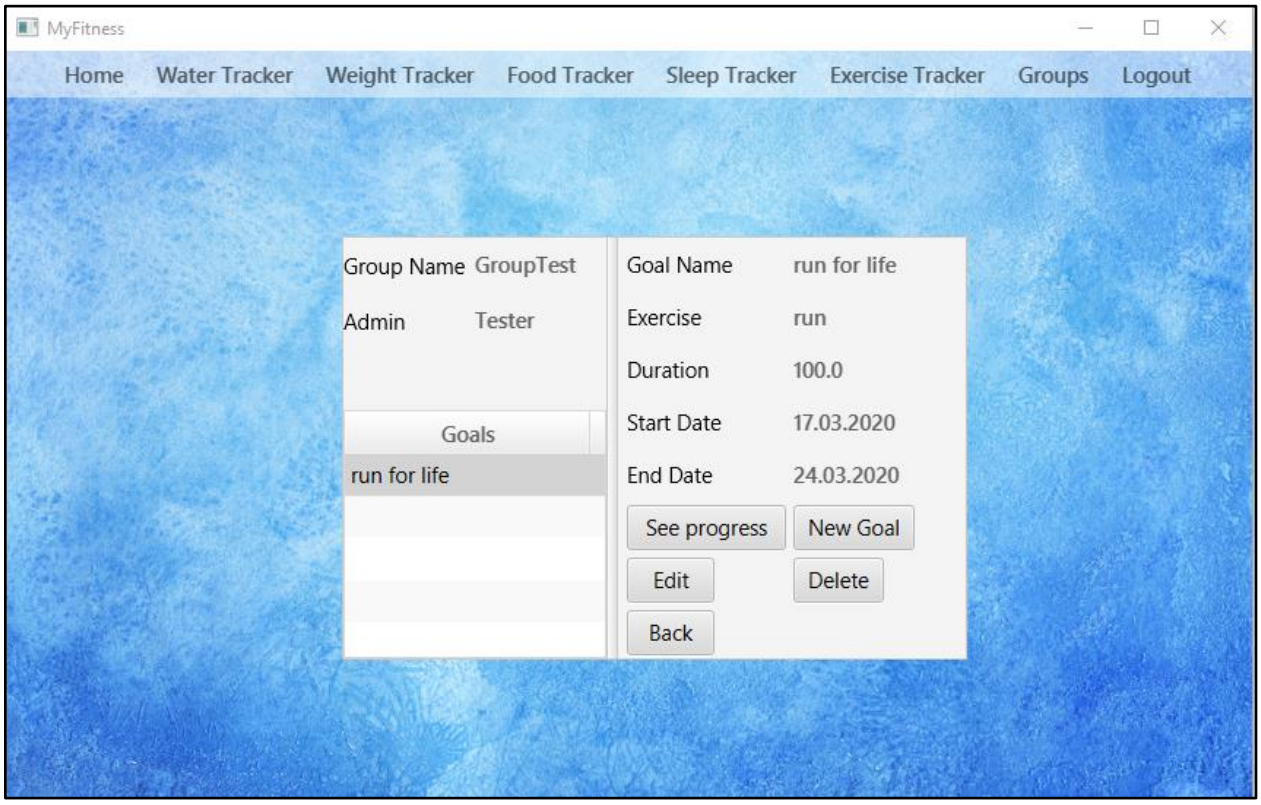
Must Have.	Should have.	Could have.	Won't have.
User Accounts	Food database	Macro counter	Barcode scanner
Personal information	Calorie counter	Sleep counter	Device connection
Exercise capture	Water tracking		Heart rate counter
Diet capture	Coaching		Stress tracker
Goal capture			Glucose counter
Regular weight capture			Automatic workout tracking
User groups			
User reports			
Group reports			

15. The following page allows the user to make, join, invite another user to a group or leave one. A list of groups the user is a part of is shown on the left. Details on the groups' name and admin user are also shown.



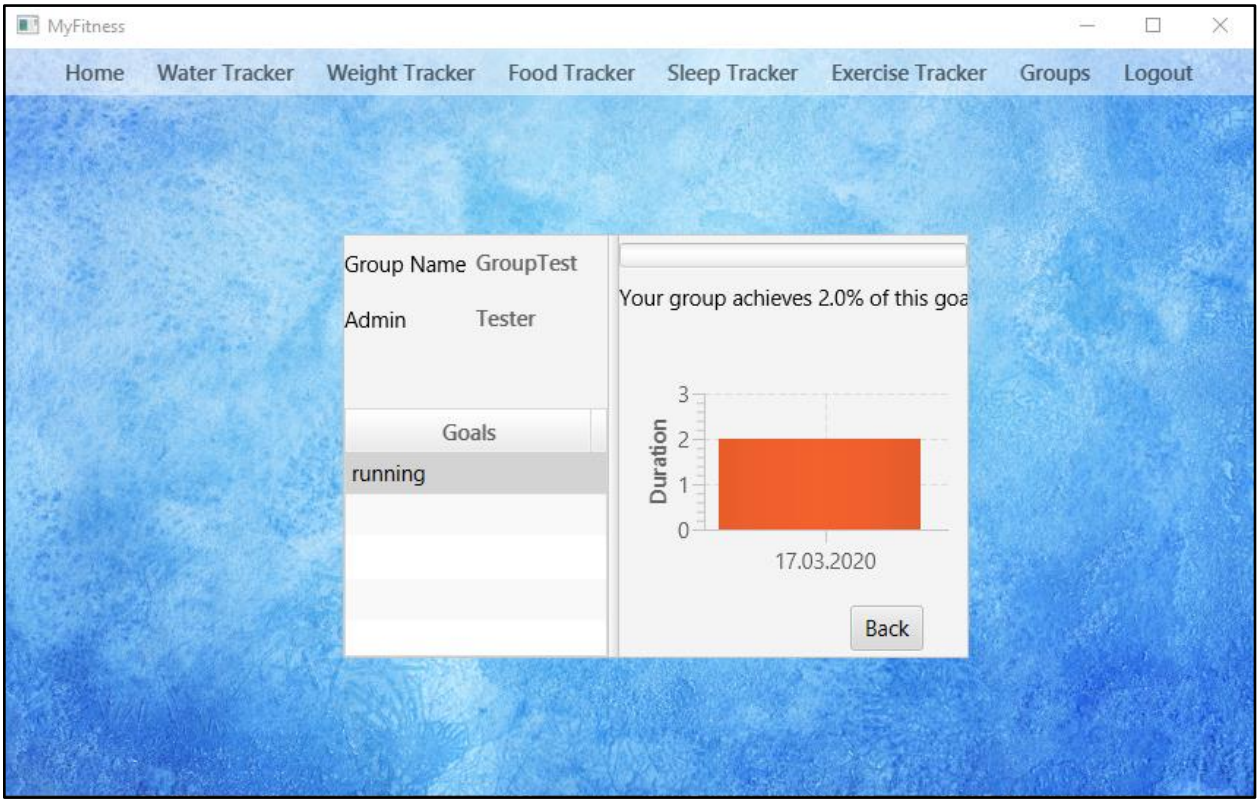
Must Have.	Should have.	Could have.	Won't have.
User Accounts	Food database	Macro counter	Barcode scanner
Personal information	Calorie counter	Sleep counter	Device connection
Exercise capture	Water tracking		Heart rate counter
Diet capture	Coaching		Stress tracker
Goal capture			Glucose counter
Regular weight capture			Automatic workout tracking
User groups			
User reports			
Group reports			

16. A group goal can be set by any user, and each goal is shown in the box in the bottom left. Details on the goal are displayed on the left. Only the group's admin can delete a goal. Each user's exercise amount between the goal dates counts towards the progress of the group goal.



Must Have.	Should have.	Could have.	Won't have.
User Accounts	Food database	Macro counter	Barcode scanner
Personal information	Calorie counter	Sleep counter	Device connection
Exercise capture	Water tracking		Heart rate counter
Diet capture	Coaching		Stress tracker
Goal capture			Glucose counter
Regular weight capture			Automatic workout tracking
User groups			
User reports			
Group reports			

17. The group goal progress can be displayed along with a percentage and progress bar.



Must Have.	Should have.	Could have.	Won't have.
User Accounts	Food database	Macro counter	Barcode scanner
Personal information	Calorie counter	Sleep counter	Device connection
Exercise capture	Water tracking		Heart rate counter
Diet capture	Coaching		Stress tracker
Goal capture			Glucose counter
Regular weight capture			Automatic workout tracking
User groups			
User reports			
Group reports			

As you can see from these screenshots, all of our Must have requirements have been met, as well as all from Should have. Sleep counter was also implemented from the Could have column. As for the macronutrient counter feature, this was not decided on being implemented. This is because each food inputted is done from the user and not pulled from an external, already complete database. This means that a user would have to enter the macronutrient information of a food they wished to use in the application, which is information an average person doesn't possess. In addition to this, we found that the user interface we had designed was already fairly crowded on the food tab, and displaying even more information would have left it hard to read.

Below is the MoSCoW requirement table from part 1, with the full amount of features we deemed to have successfully implemented highlighted in red.

Must Have.	Should have.	Could have.	Won't have.
User Accounts	Food database	Macro counter	Barcode scanner
Personal information	Calorie counter	Sleep counter	Device connection
Exercise capture	Water tracking		Heart rate counter
Diet capture	Coaching		Stress tracker
Goal capture			Glucose counter
Regular weight capture			Automatic workout tracking
User groups			
User reports			
Group reports			

5.3 Unit Testing

We have included an example of the unit tests we ran on our code throughout the development process in order to test the functionality of the methods responsible for an implemented feature.

1. **Testing CalculateBMI**

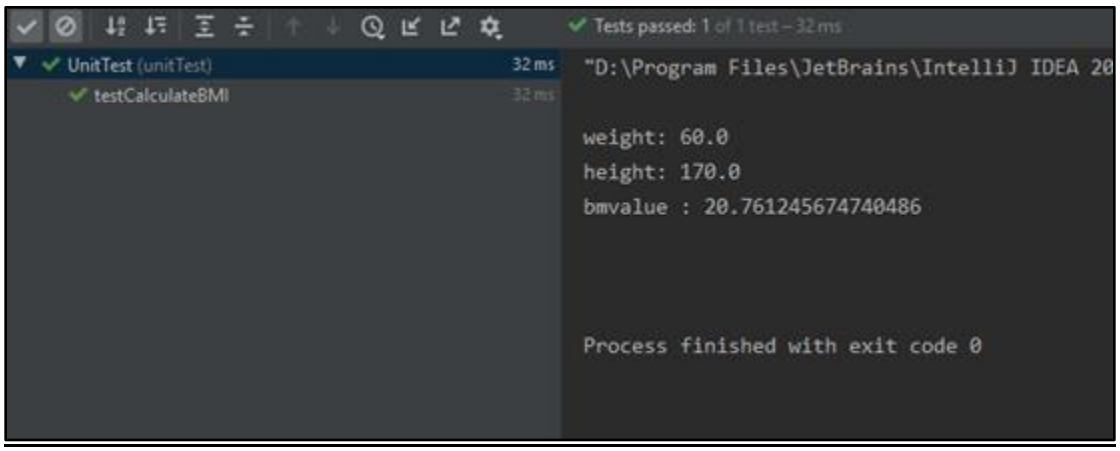
   a) The method calculateBMI in the User class is tested.

   b) Tester code:

```
23      @Test
24      public void testCalculateBMI()
25      {
26
27          User tester =new User();
28          tester.setHeight(170);
29          tester.setWeight(60);
30
31          assertEquals( expected: 20.76,tester.calculateBMI(), delta: 0.02);
32      }
```

   c) Code base: user.User.java (Line 106).

   d) Test result: pass.



2. Testing JoinGroup

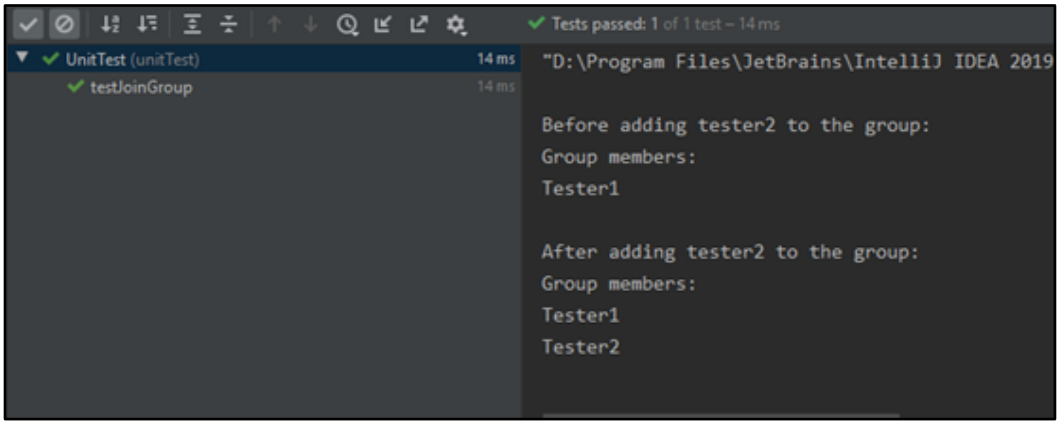
a) The method addUser in class Group is tested.

b) Tester code:

```
38      @Test
39      public void testJoinGroup()
40      {
41          //Create a new user
42          User tester1=new User();
43          tester1.setName("Tester1");
44          User tester2=new User();
45          //Create second new user
46          tester2.setName("Tester2");
47          //create a new group
48          Group testGroup=new Group(tester1, name: "test Group");
49
50          //before adding tester2 to the group
51          System.out.println("Before adding tester2 to the group:");
52          System.out.println("Group members:");
53          for(User u:testGroup.getUsers())
54          {
55              System.out.println(u.getName());
56          }
57          System.out.println();
58          //after adding tester2 to the group
59          System.out.println("After adding tester2 to the group:");
60          System.out.println("Group members:");
61          testGroup.addUser(tester2);
62          for(User u:testGroup.getUsers())
63          {
64              System.out.println(u.getName());
65          }
66      }
```

c) Code base: group.Group.java (Line 43).

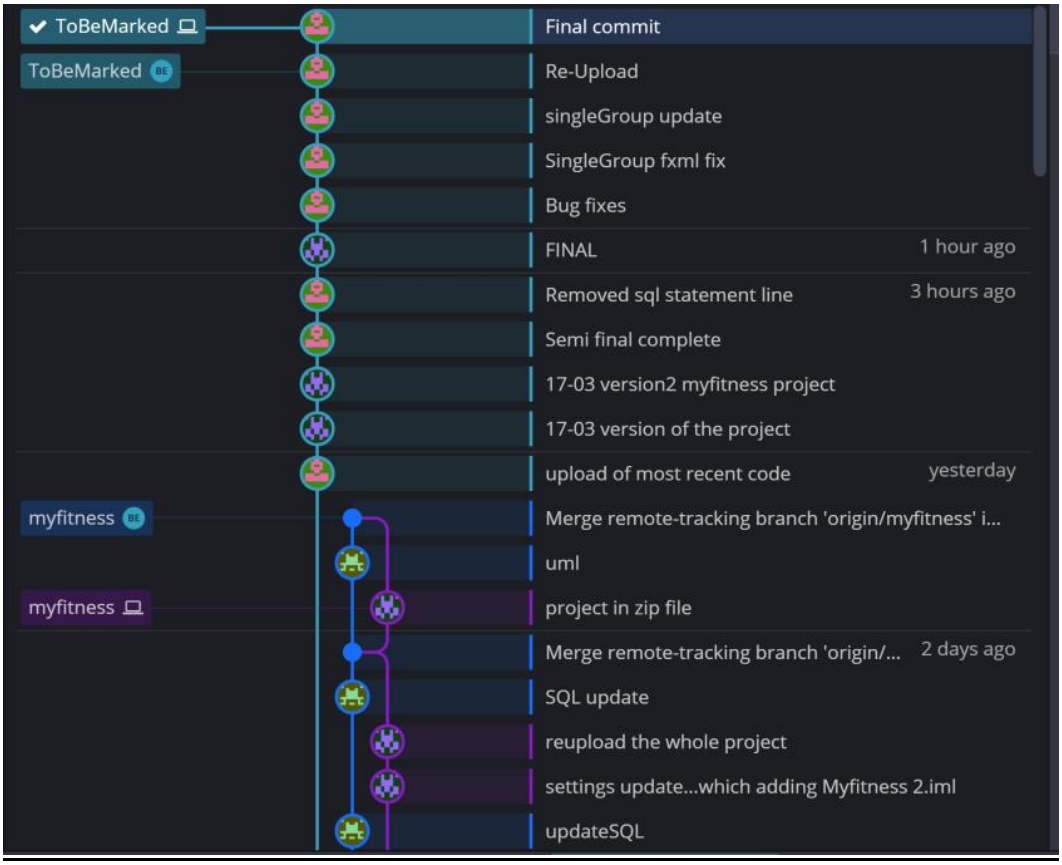
d) Test result: pass.



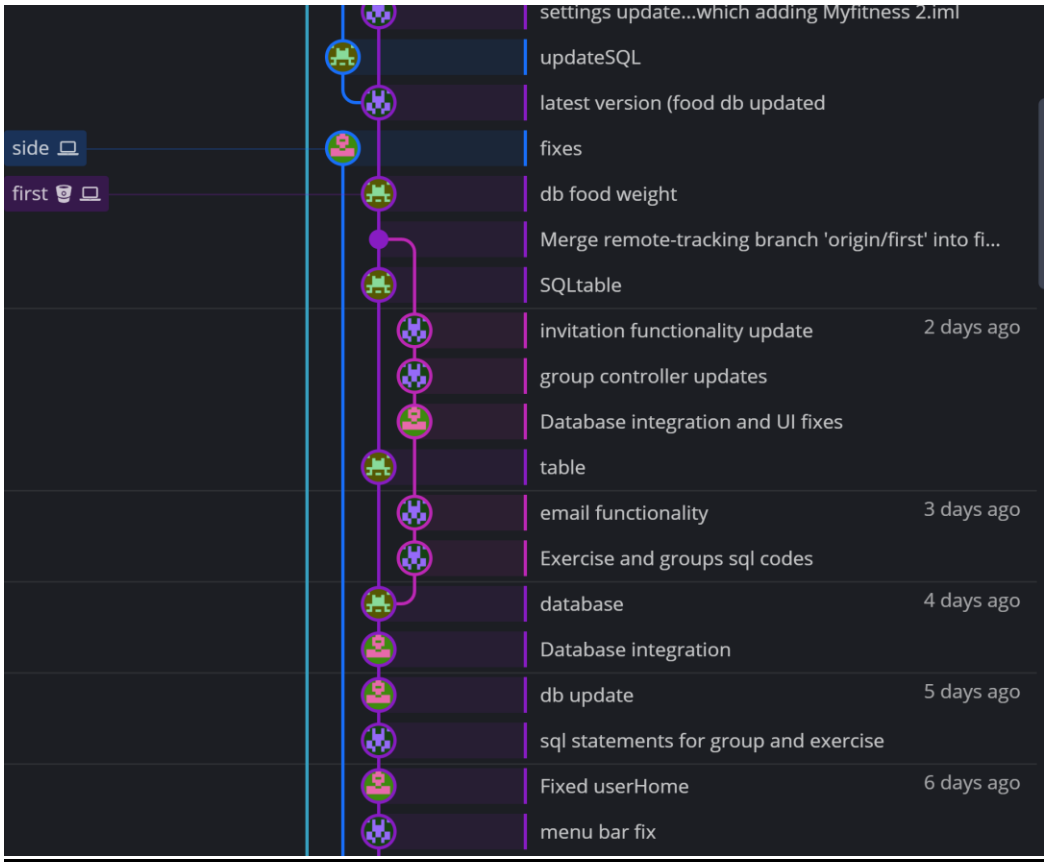
5.4 Evidence of Iterative development.

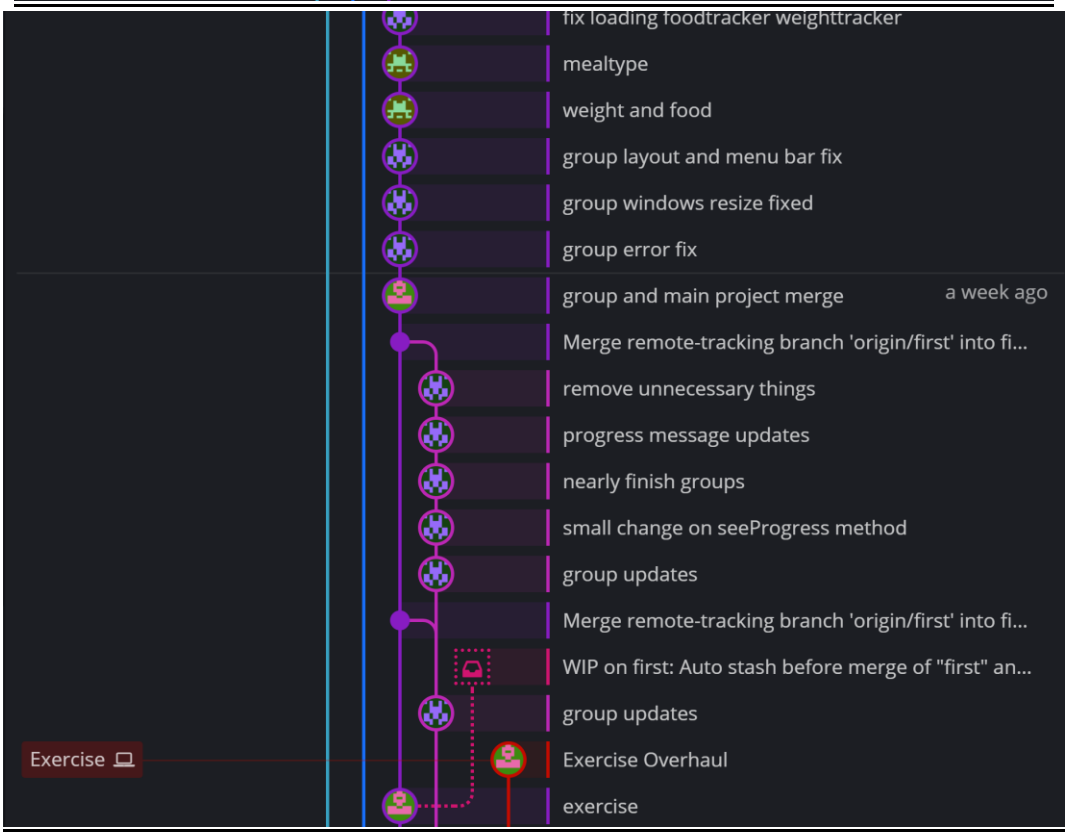
Throughout the development process we frequently went back to the planning, design, implementation and testing phases to make changes based on the problems we have found, taking an iterative development approach. In order to log these changes, we used a Git Repository tool BitBucket[11] to easily document these changes. Below are some screenshots of our BitBucket commits, showing brief comments on what changes were made and the date of them. We frequently met up in Scrum style meetings to discuss progress, group code and debug. As well as setting new targets to achieve by the next meeting.

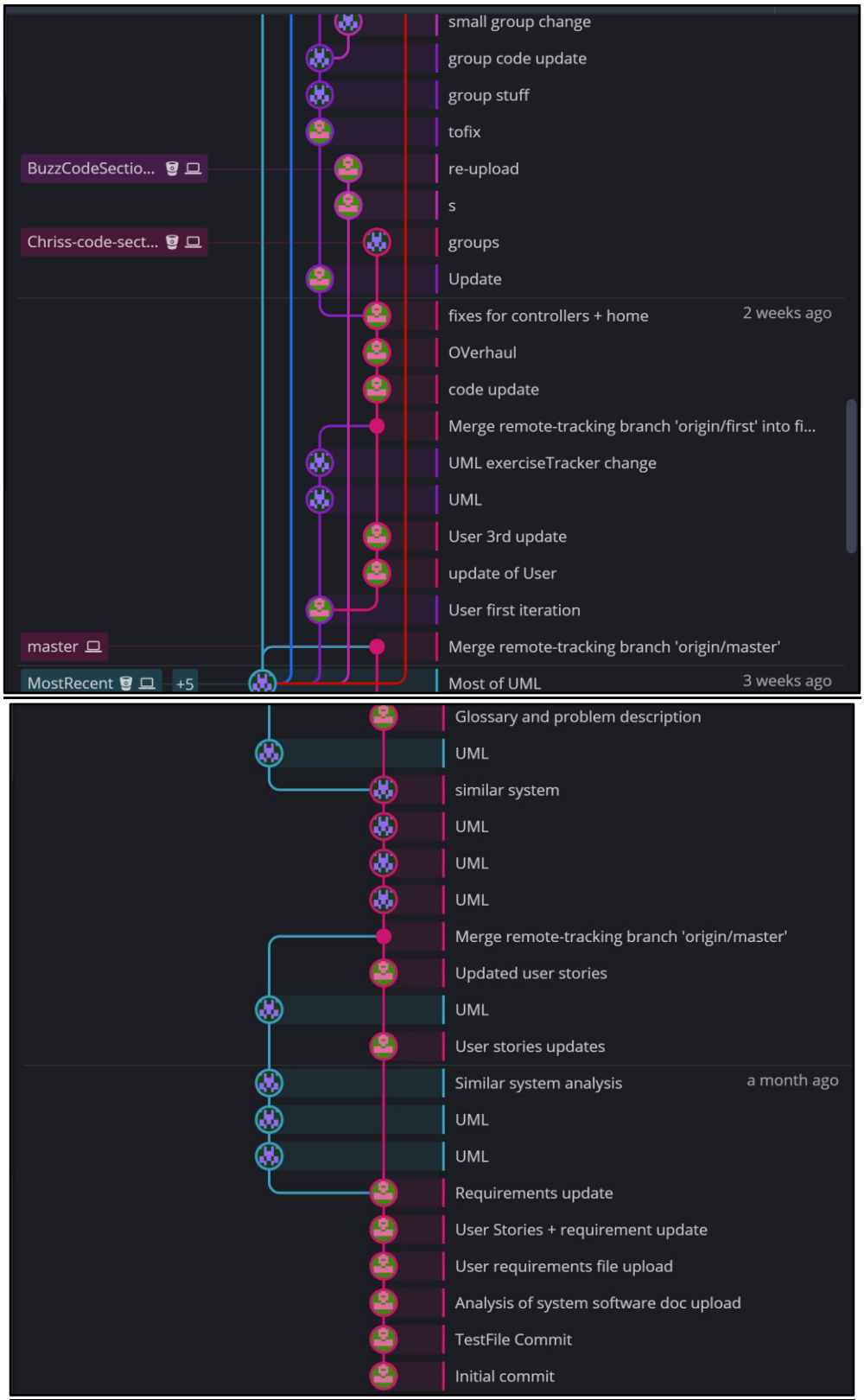
End of development.



[11] https://bitbucket.org/product/


settings update...which adding Myfitness 2.iml
updateSQL
latest version (food db updated
side
fixes
first
db food weight
Merge remote-tracking branch 'origin/first' into fi...
SQLtable
invitation functionality update
2 days ago
group controller updates
Database integration and UI fixes
table
email functionality
3 days ago
Exercise and groups sql codes
database
4 days ago
Database integration
db update
5 days ago
sql statements for group and exercise
Fixed userHome
6 days ago
menu bar fix


fix loading foodtracker weighttracker
mealtype
weight and food
group layout and menu bar fix
group windows resize fixed
group error fix
group and main project merge
a week ago
Merge remote-tracking branch 'origin/first' into fi...
remove unnecessary things
progress message updates
nearly finish groups
small change on seeProgress method
group updates
Merge remote-tracking branch 'origin/first' into fi...
WIP on first: Auto stash before merge of "first" an...
group updates
Exercise
Exercise Overhaul
exercise



Start of development process

6 Stage 2 Preparation

6.1 How will our stage 1 prototype be ported into a web-based application?

As mentioned in part 1.1, we will be using the Python programming language and it's Django framework to implement our web-based application. We have decided to switch to Python for stage 2 as all members of the team have previous experience using the language for web-based application development. In addition the language supplies versatile but easy to learn frameworks that will be able to support all features from our application in stage 1. The first stage of implementation would be to convert the Model sections of our current application from Java to Python. Adapting our prototype to be a web-based application should not require any changes in how we process data collected from the user. Processes like object creation, calculations for aspects like BMI or for coaching purposes will remain the same. The only expected changes for data processing would be due to differences in programming language between Java and Python. As for data storage, the database we have created for stage 1 can be used for stage 2, and in theory requires no changing. Instead of using the Java Database Connectivity (JDBC) API, our change in programming language will most likely result in the use of Python's psycopg2 API for database access and manipulation, as we all have previous experience using it. In terms of architecture, the Model-View-Controller (MVC) used for stage 1 will need to be converted to the Model-Template-View architecture preferred by Django. We have used the JavaFX software platform to produce a user interface for our first stage, however this will be replaced with Django rendered HTML templates with corresponding CSS files for formatting.

Glossary of terms

Word	Description
User	People using software. Used to record user account information, physical information, and contact information.
Group	Created for many users, you can share personal experiences and set common exercise goals for multiple users.
Water tracker	Track and record users' daily drinking water.
Weight tracker	Track and record users' daily weight changes.
Exercise tracker	Track and record users' daily, weekly and monthly exercise.
Exercise	Select the item of exercise and the duration of the exercise.
Food tracker	Track and record users' daily diet.
Meal	Add or select food and choose the type of food to calculate total calories.
Sleep tracker	Track and record user sleep quality.

Tracker	Contains the current user and the user's goals. An abstract class for all tracker classes.
Database connector	Connect java code with database.
WeightDB	For weight page data storage operations.
FoodDB	For food page data storage operations.
FileTool	For meals table data storage operations.
Email	When a user invites another user to the group, a message is sent
AdminClass	For sleep, group, water, exercise, user, data storage operations.
Goal	An aim or purpose, the user sets the goal and goes to achieve it.

7. Group contribution

7.1 Your student ID: 100237137

7.2 Please provide the % contribution of each group member within your group without consulting the other members, i.e. from your own perspective (delete entries below which exceed your group size and replace "Group member x" with the member's name).

Jonathan (Buzz) Edward Embley-Riches : 25%

James Burrel: 25%

Linfeng (Chris) Chen: 25%

Ranhui (Harris) Zhang: 25%

100 %

7.3 Please provide what activities you did and what parts you contributed to the final deliverable:

· **activities:**

- Organised the group meetings.
- Decided on agile method and our initial approach with group members,
- Designed UX & UI with group members.
- Designed Class model with group members.
- Designed MVC with group members.
- Designed Database design with group members.
- Aided construction of report with group members
- Coded sections (see below).

· **contributions**

(Full contribution, all below done by myself with some assistance from other members)

- WaterTracker.Java
- WaterTrackerController.java
- WaterTrackerReportController.java
- WaterTrackerPage.fxml
- WaterTrackerReport.fxml
- SleepTracker.java
- SleepTrackerController.java
- SleepTrackerPage.fxml
- SleepTrackerReport.fxml
- SleepTrackerReportController.java
- ExerciseHomeController.java
- ExerciseGoalController.java
- ExerciseReportController.java
- exerciseHome.fxml
- exerciseGoalPage.fxml

- exerciseTrackerReport.fxml
- User.Java
- HomeController.java
- userHome.fxml
- LoginController.java
- loginPage.fxml
- SignUpController.java
- signupPage.fxml
- AdminClass.java
- main.java