CM10313

Software Processes and Modelling Report

Group 19 - 2023

Thomas Canning	tc2169	Mcomp Computer Science
Alexander Agafonov	aa3853	Bsc Computer Science
Artiom Casian	ac3157	Bsc Computer Science
Harrison Crane	hc2188	Bsc Computer Science & AI
Alex Clarke	ac3431	Mcomp Computer Science

Abstract

(Thomas Canning, Alexander Agafonov, Harrison Crane, Artiom Casian, Alex Clarke)

We have created a wellbeing application to track a user's progress in 3 key areas: exercise, diet, and sleep. These 3 areas have been chosen as a result of feedback from our target audience, current university students, which we gathered through a questionnaire.

The key features revolve around users entering data into the application in each of the 3 categories. The user can track their progress over given time periods and can set their own goals. Additionally, we have added visualisations to help users better understand their progress and identify areas for improvement, including graphs and charts that provide an intuitive way to view progress over time. Including goals and visualisations is a functional requirement based on our research into how to best motivate users through software.

Our application is a desktop GUI application written in Java, with a Rust backend. The design of our application features a main menu with options to go to the exercise, diet, and sleep pages, and other pages for extraneous data. Each of these pages is where the user enters their data for the given category and can get a detailed view of their progress for that category.

To evolve our application, we would make use of the Terra API to integrate with wearables and automatically import fitness data after exercise. We would also evolve our goals system to reward the user for progression, such as by awarding customisation options.

Github pages of project:
frontend backend

Contents

Introduction

(Artiom Casian)

Personal Informatics, described by (Li, Medynskiy, et al., 2012), is "a class of software and hardware systems that help individuals collect personal information to improve self-understanding". Every year more people try to engage into a healthier lifestyle and keep track of their mental and physical health. Technology tries to follow this trend and support them, which leads to companies developing new gadgets that collect user's information and display it back to the user in an informative and engaging way. These two factors led to an increase in demand as well as awareness around the personal informatics software and much research has been conducted to explore this area.

According to (Li, Dey, and Forlizzi, 2010) the key and most useful features that users found helpful from interacting with different informatics systems were organisation of data, effect of app on the user's personal results and user-friendliness of the app. Organisation of data is a key aspect of an informatics system as it allows the user to quickly find information they are looking and see a detailed break-down of a specific data without having to search through different menus or windows. This not only saves time, but also allows users to gain a better understanding of their personal information and their wellbeing.

Secondly, the app must have a positive impact on the user's physical and mental wellbeing. This can be done through different features that the app has such as a goal/award system that motivates the user to improve and to achieve his own goals. (Fritz et al., 2014) have found that multiple users found the option to review personal records very helpful and engaging as it motivated them to set new records and see their progress over time. As well as sharing and trying to beat their friends' records added a competitive aspect to this feature.

And lastly the app should have a user-friendly design. After a tiring day or a workout, the user wants to quickly see his progress, add personal information, or share some data. This should not be a complex process and the user should find all the important information on the home screen. As suggested by (Klasnja et al., 2018), users found it very helpful when the data was automatically transferred to the app from their fitness gadgets such as smartwatches. Instead of imputing data by hand which can lead to errors and inconsistency in data, the smartwatch was sending data to the app which then updated the user's information automatically.

For our Personal Informatics software system, we will try and implement a general-purpose desktop application that would be keeping track of user's personal information and display data in an efficient and engaging manner. The Idea is to focus on 3 main areas which are: exercise, diet, and sleep. The app will be taking data inputs from the user, storing it, and returning it in a convenient and easy to read way. In addition to that, (Li, Medynskiy, et al., 2012) we plan to implement extra features such as goal/award system. The idea behind that is to motivate the user to access the app more often to see their progress. Overall, our Personal Informatics software system has the potential to be a comprehensive tool for users to track their personal information and make positive changes to their lifestyle.

Agile Software Process Planning and Management

(Thomas Canning)

3.1 What is agile and scrumban

Agile software development, as described by Abrahamsson (Abrahamsson et al., 2017), is a methodology attempting to answer the business industry's requirement for a faster and nimbler software development process. This is achieved through the primary objective of and agile development team being to continuously turn out tested working software at frequent intervals. For agile development, a close relationship between developers and clients is key, with the client continually tweaking their requirements throughout the development process, as opposed to a fixed contract explicitly stating requirements before development begins – "agile development is focused on delivering business value immediately as the project starts." (Abrahamsson et al., 2017).

Throughout our project, we have adopted a scrumban development methodology. This is a hybrid of two agile methodologies, being kanban and scrum. Kanban is a workflow management method that utilises kanban boards. A kanban board makes use of columns, and tasks are moved along these columns as they progress, starting from the backlog and moving to done.

Scrum helps teams deliver value incrementally, by making use of short cycles known as sprints. Sprints start with a sprint planning meeting, which is a collaborative effort between the scrum master (person who ensures the scrum framework is followed) and the product owner (accountable for effective product backlog management), and the rest of the development team. The selection of items chosen to be completed by the end of the sprint is known as the sprint backlog. The team starts work on the sprint by taking tasks - which are broken down programming jobs from the backlog - into an in-progress stage, and then to done. Throughout the sprint, the team checks on everyone's progress in a meeting call sprint standup, with the goal of working out any challenges that would impact the team's ability to deliver the sprint goal. A sprint is followed up by a sprint review, where the team demonstrates to the stakeholders the work they have completed in the sprint, which is then followed by an internal sprint retrospective, where the team reflects on their efforts in the sprint, discussing what went well and what can be improved for the next sprint.

3.2 How we carried out scrumban

For our scrumban methodology, we assigned a scrum master (Thomas Canning), and a product owner (Alexander Agafonov). The scrum master was responsible for taking meeting minutes and organising meetings, while the product owner was responsible for managing the product backlog and keeping the kanban board up to date.

To plan and track our sprints, we made use of a variety of features offered by GitHub. Whilst using GitHub to manage our git repository so we could easily work collaboratively on a single code base was important, we also made use of other GitHub tools to plan sprints. We kept track of our sprints using a kanban board for each sprint, which had a column for the sprint backlog chosen for that sprint, and then a series of columns to move tasks along as they progressed. An important column that helped to reduce bugs in the code, improve code quality, and ensure everyone in the team had a good understanding of the codebase was the awaiting review column. No task would be moved from this column to the done section until it had been talked about in a meeting and reviewed by at least 1 other group member. Our Kanban boards looked like the following:

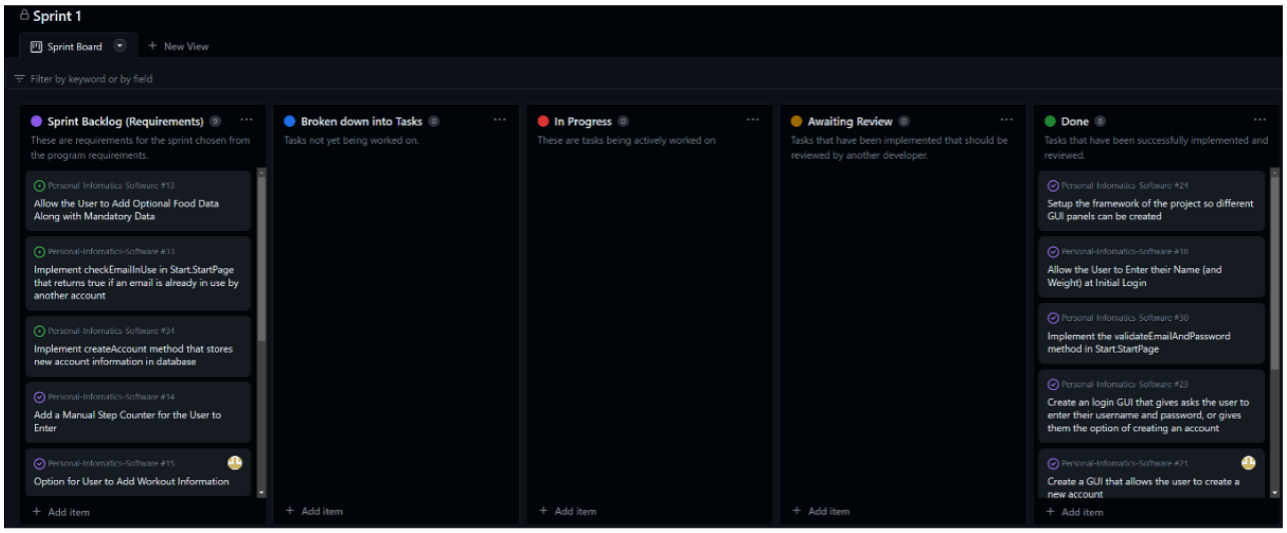


Figure 3.1: Sprint 1 Kanban board after the end of the sprint (full size in appendix)

Along with Kanban boards, we used the issues feature of GitHub to keep track of our product backlog. When we discussed a new feature in our meetings, we would add it as an issue which would create a list of all our requirements, which was then ordered by the product owner and moved to the kanban board if it was to be a part of that sprint.

We took the first two weeks to plan our project as a whole. We took this time to setup the GitHub page, creating a framework for how we would carry out our project as an agile process. We also began making a list of requirements. As part of setting up our project, we created a folder in the GitHub page where we would keep meeting minutes. Minutes from each meeting can be found in the appendix.

For our project, we had three sprints. Sprints lasted for approximately two weeks. One thing we found challenging was sticking to an exact two-week time schedule, due to people being away and easter breaks. We partly overcame this with online meetings, but we did have to deviate from a strict two-week time schedule as it was sometimes not practical. Every sprint began with one or more sprint planning meetings and had at least one sprint standup during the duration of the sprint, which was an official meeting where we discussed progress with each other and any challenges we'd faced. Every sprint then ended with a sprint retrospective meeting, where we discussed what we achieved in the sprint, what went well, what could be improved, and what needs to be moved to the next sprint.

We made a lot more progress in later sprints than we did in sprint 1, and this is mostly because we were unfamiliar with the swing GUI library at the start and later became a lot more familiar with it, speeding up programming. As a result, lots of the tasks from the sprint 1 backlog had to be moved to sprint 2. We aimed to have two meetings per week, on a Monday and a Friday. Whilst sprint standup is typically a daily event for software teams, this would be impractical for us as we have other commitments to focus on with our course.

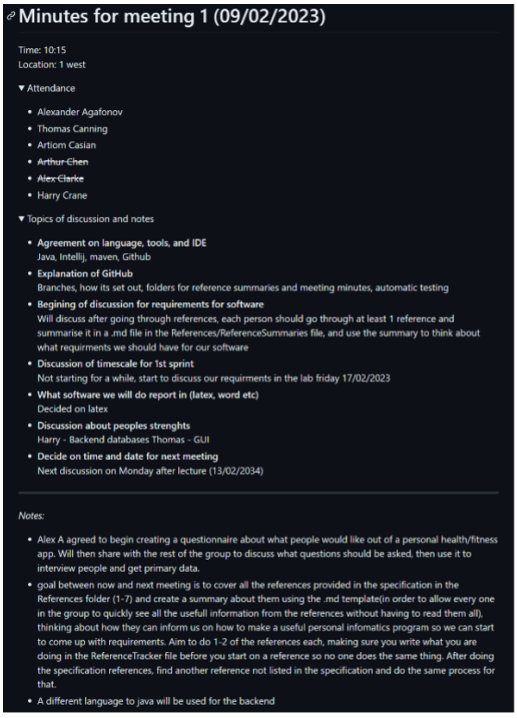


Figure 3.2: An example of meeting minutes, from the 1st meeting (full size in appendix)

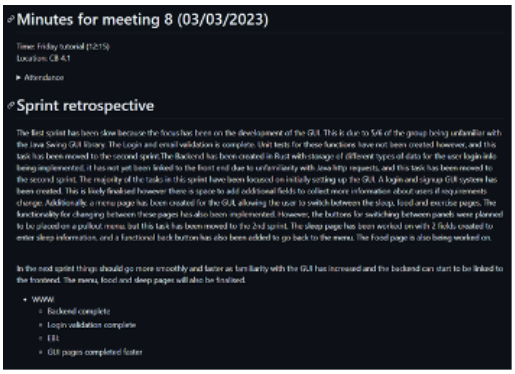


Figure 3.3: end of sprint 1 full size in appendix

Date	Topic
09/02/2023	Tools, coding language, assigning references to people to help come up with requirements
13/02/2023	Discussion of features we want creation of questionnaire
17/02/2023	Discussing java libraries, questionnaire responses, backend language, creation of requirements list
20/02/2023	Continued working on requirements, chose java swing library for GUI, created kanban board
24/02/2023	Sketched an initial GUI design of program, created a baseplate for the java code
27/02/2023	Started 1st sprint, decided what everyone's tasks are for sprint 1
03/03/2023	Sprint stand up
10/03/2023	Sprint retrospective and review, what needs to be carried over to sprint 2
20/03/2023	Planning sprint 2, created kanban board, moving uncompleted tasks from sprint 1
24/03/2023	Started 2nd sprint, talked about UML designs, came up with new requirements
31/03/2023	Sprint stand up
17/04/2023	Sprint retrospective and review, what features need to be moved to sprint 3
20/04/2023	Starting sprint 3
24/04/2023	Sprint stand up
28/04/2023	Sprint retrospective

Figure 3.4: A summary of the main topics of discussion in each meeting. Full meeting minutes can be found in the appendix.

3.3 Overview of sprints

Detailed sprint planning, sprint standups, and sprint retrospectives can be found in the appendix.

Sprint 1

The focus of the first sprint was establishing a framework for the GUI. We created a menu to access each of the subpages from, and a Subpage class that each subpage would extend. We also created and finished a GUI for the login and signup system. Our goals for this sprint had been much more ambitious, as we had planned to of also implemented a more complicated pull-out menu system and have the food page complete for instance, but these tasks had to be pushed back to later sprints. Progress in Sprint 1 was slower than anticipated because our group was unfamiliar with Java GUIs and as such, programming was slower.

Sprint 2

In sprint 2, all subpages but the goals subpage had been created. All data entry aspects of these pages had been completed, but much of the GUI for viewing and comparing data over time was still incomplete, such as adding charts. At this point, the backend had been setup, but had not yet been connected to the front end.

Sprint 3

We completed all the initial requirements we had set out to achieve in Sprint 3. We had a front end with all out features working, and a backend connected to our front end, meaning we had a fully working log in and sign-up system, amongst other things. Much of Sprint 3 was also focused on adding charts to our GUI to view data over time.

Overall, our approach to agile development was very effective at producing a product that met the requirements by the deadline and helped us to manage who was doing what easily as a team. Whilst sticking to the kanban methodology seemed to slow us down initially, as we wanted to jump straight into putting our ideas into code, it was a much faster approach at reaching a minimum viable product than a sequential development methodology such as waterfall, and dividing tasks into sprints made it easy to continuously adjust our requirements and improve our product.

Software Requirements Specification

(Thomas Canning, Alexander Agafonov, Harrison Crane, Artiom Casian)

4.1 Gathering Requirements

While we already had a set of initial set of requirements from the project specification, we thought we needed to get more requirements to create a useful and successful project. We used a combination of our own ideas, various online sources and papers, verbal interviews with people and questionnaires. We focused the questionnaires and interviews to fellow students since they were our target audience for this application. They were done in such a way that we could get a feeling of what our target audience wanted in this type of application. We then looked through the ideas and data to see which ones were manageable and reasonably doable for our scope. We decided to mainly focus on user experience; a wide range of health-related options and a system in which data entered weeks ago can be stored and viewed. This would make our application versatile and will give the application longevity in terms of usage

Figure 4.1: Screenshot of a questionnaire result (more in appendix)

4.2 Non-Functional Requirements

Tools used to create our software

1. Program will be written in Java and will be a desktop GUI application.
2. Program will be a Maven project to make downloading dependencies easy. We will include the following dependencies in the pom.xml.
   (a) “junit.jupiter for unit tests.
   (b) “jdatepicker” for a simple way of letting the user pick a date within the GUI.
   (c) “jcalendar” is required for the date picker to work.
   (d) “jfreechart” is used to plot line charts.
   (e) “okhttp” is used as the http client for connecting with the backend.
   (f) “jackson” is used to parse Json data.

3. Everyone will use IntelliJ as the IDE so we share the same configuration files making collaborating easier.
4. We will use git with one GitHub repository for the java front end and other project files and another for the backend.
5. We will also use GitHub for the agile software process planning and management, including:
6. Automatically running our unit tests each time someone pushes to the repository by making use of a GitHub workflow.
7. Using a kanban board for each sprint-to-sprint plan, as well as an overall planning board for the entire project to keep track of all tasks such as who is working on what on the report.
8. Taking meeting minutes as well as sprint standup in .md files.
9. Keeping a list of all the references used in the project.
10. Backend will be created using rust and the Rocket.rs framework.
11. Backend will connect to a MySQL server using the "sqlx" crate.
12. Backend will be deployed to a remote server using docker.

Testing

1. We will adopt a test-driven development approach for the parts of our code that can be easily unit tested, writing unit tests per function using the Junit library.
2. For our GUIs, every member of the group has thoroughly tested every other members' GUI code, with a variety of inputs and attempting to discover bugs occurring due to unexpected inputs.
3. We will run out tests automatically every time code is pushed to the git repository using a GitHub action.

Scalability

1. The program should be written in such a way that the GUI is easily scalable to include new features.
   (a) There will be a menu where each of the main sections of the software can be accessed from, e.g., exercise, food etc.
   (b) Each section will have its own class, which will extend some form of JPanel template so that each page has the same look and feel.
   (c) Every page will have a similar layout with buttons/text boxes in a vertical column.
   (d) Every page will make use of the same GUI components such as the chart and date picker components.
2. An effort will be made to not repeat code by having different parts of the program use the same methods if they can be reused, such as using the same methods for validating if a input is valid across the program.
3. The backend should be able to scale to support multiple users, store more data, and the addition of new HTTP requests

4.3 Functional Requirements

Login and signup system

1. The login GUI will be the first part of the program the user sees and will contain the following:
   (a) A text box to enter an email.
   (b) Another text box to enter a password.
   (c) A tick box to show the password.
   (d) A login button to submit the login details and attempt a login.
   (e) A signup button to go to a page to create a new account.
2. If the user successfully logs in, they'll be taken to the main menu of the program.

3. If the user enters incorrect account details, they will be informed with an incorrect email or password popup.
4. The signup system will consist of two pages.
5. The first page will consist of:
   (a) A text box to enter an email.
   (b) A text box to enter a password.
   (c) Another text box to re-enter the password to confirm it.
   (d) A show password text box.
   (e) A signup button.
6. Pressing the signup button will check whether valid details have been entered to create a new account. If so, the user will move onto a second signup page and have to enter more details, otherwise, the user will be told what is not correct in a popup window. The following checks are made:
   (a) Check passwords match.
   (b) Check email is not already in use with an existing account.
   (c) Check that an email is a valid format for an email.
   (d) Check no fields are empty.
   (e) Check that the password is secure enough.
7. The second signup page contains the following GUI elements for the user to enter additional account information:
   (a) A textbox to enter a name.
   (b) A date picker to enter Date of Birth.
   (c) A text box that only accepts a number to two decimal places to enter current weight.
   (d) A signup button.
8. If valid information has been entered, pressing the sign-up button will submit a request to the backend to create an account using all the information provided on the sign-up pages, and the user will be logged in and taken to the main menu.

Menu

1. There will be a menu page that has a pull-out menu with buttons to access each of the subpages for recording data from, and a logout button.
2. The rest of the menu will be dedicated to viewing stored data, such as in the form of charts, and what the user sees on their menu should be customisable.
3. The backend will be tested using the postman application to ensure that HTTP routes are working correctly.

Exercise

1. The exercise page will let you record a workout and view previous workout statistics.
2. It will contain the following GUI elements to record the following aspects of a workout:
   (a) A dropdown list to choose activity type.
   (b) A text box to enter exercise start time.
   (c) A text box to enter exercise end time.
   (d) A text box that only accepts numbers to two decimal places to enter distance of exercise.
   (e) A text box that only accepts numbers to two decimal places to enter elevation gain if it is relevant.
   (f) A date picker to choose date exercise happened on.
   (g) A submit button to push exercise data to backend.
   (h) A view exercise statistics button that takes user to a new page where they can view all exercise data.
3. The exercise statistics page will have text readouts of exercise data from past workouts, it will also show the number of steps taken today taken from the step tracker page (6). It will also have a button that will display exercise data in a GUI chart to make it more intuitive to track progress visually.

Sleep Tracker

The thesis written by (Carlhenricsdotter, 2022), emphasizes the significance of sleep as an integral component of a Personal Informatics System. This research highlights the crucial role of sleep in maintaining optimal physical and cognitive health, as well as its impact on the overall effectiveness of personal informatics solutions.

1. The sleep page lets you keep a record the amount of time you sleep for each night.
2. It contains the following GUI elements:
   (a) A text box to enter the time sleep started.
   (b) A text box to enter wakeup time.
   (c) A date picker to enter the date sleep started on.
   (d) A submit button to push sleep data to the database.
   (e) A button that displays a chart showing sleep data.
3. Sleep data can be viewed on the chart that shows the amount of time slept each night over a certain time period.

Weight Tracker

1. The Weight Tracker page will let you enter a weight on a given date, so you can easily track your weight over time.
2. It will contain the following GUI elements:
   (a) A text box only accepting a number to two decimal places to enter the weight.
   (b) A date picker to enter date weight recording was taken on.
   (c) A submit button to push weight data to backend.
   (d) A button that displays a chart showing weight over time.

Step Tracker

It has been mentioned in the article written by (Brabazon, 2015), that a step tracker is an essential component of any Personal Informatics system. That has influenced our decision to implement a similar feature.

1. The Step Tracker page lets a user keep track of the number of daily steps over time.
2. It will contain the following GUI elements:
   (a) A text box that only accepts digits to enter the number of steps taken.
   (b) A date picker to enter the date the step count was recorded on.
   (c) A submit button to push step count to the backend.
   (d) A button that displays a chart showing the number of steps taken each day over time.
3. The number of steps taken on the current day is also displayed in the exercise statistics section.

Food

1. The food page will let you track what you are eating each day. It will let you enter a name of the food you have eaten, and the number of calories that food contains so you can track total number of calories in and look back on your diet.
2. There will also be a section for tracking consumption of five fruit or vegetables items per day, after consuming a fruit or veg item, the user can click one of five red boxes to turn it green.
3. The page will contain the following GUI elements:
   (a) An array of five clickable buttons to track five a day.
   (b) A set of text boxes to enter information for recording food.
   (c) A date picker to enter the date the food information was recorded on.
   (d) A submit button to push food entry to the backend.
   (e) A food history button that takes you to a new page to view stored information about food history.
4. On the food history page, the user will be able to view what they ate on a particular day, as well as view calorie intake over time on a chart.

Goals

(Niess and Diefenbach, 2018) report reveals that goal setting strategies were well-received by most users, while commonly used strategies such as rewards failed to motivate participants to become more active.

1. The goals page will let you create a list of your own goals. Each goal consists of a description of the goal, a date that the goal will be achieved by a tick to mark the goal as complete, and a cross to delete the goal.
2. Goals that are marked as complete turn green.
3. Goals that have exceeded their complete by date turn red.
4. Logic makes sure that the due date of a goal can't be set to an earlier date than the current date.
5. The goals page consists of the following GUI elements:
   (a) A button that creates a new goal.
   (b) A vertical stack of goal elements that contains the buttons and text boxes described earlier.

Backend

1. The backend will be able to be connected to using the HTTP protocol.
2. It should include the functionality of multiple users, each with access to only their own data.
3. A new account will only be able to be created if the email used in the account is unique.
4. Date's will be formatted "
5. Once a user logs in, a user ID will be returned, this is then parsed into all other requests to get and store data for that user.
6. There should be HTTP routes for storing and saving data for exercise, sleep, weight, food, goals, and five a day.
7. All data returned from the backend will be in JSON form.
8. When possible, data will be returned in chronological order
9. GET requests will take parameters in the form of HTTP parameters.
10. POST requests will take parameters in the form of HTTP form data.

Design

(Alexander Agafonov, Harrison Crane, Alex Clarke)

When we started planning our project, we wanted to create an application that is accessible and simple for all users to understand. With this philosophy in mind, we decided to use the Java programming language to build our program. Java contains many libraries which can create elegant looking interfaces and contain features which are helpful in aiding the user experience.

5.1 Programming Paradigm

Our next step was to plan out our object-oriented design of our program as for our programming paradigm, it was specified in the specification to use object-oriented programming. We could split our initial program ideas into smaller programmable chunks and arrange our options into their own class. For example, since our project is based on personal health tracking, exercise and food sections can be individual classes. Another reason why object-oriented programming is useful is that it is very easy for us to build upon our current application in the case we decide to expand our initial requirements. Since we will be building upon our initial requirements after each sprint, we will need to be as efficient as possible to effectively slot these features in. Furthermore, object-oriented concepts such as inheritance will simplify the design of this project; we do not need to create redundant attributes and methods that will bloat the program, for example, sections such as food and sleep can be derived from one class which can set out the structure of the graphical interface.

5.2 GUI Libraries

We decided to use the swing GUI library as it is built into Java and provides all the features we need. We did consider potentially making an android application coded in Java, however as none of us are familiar with android studio we decided against that. Swing provides a simple appearance which is not intrusive for the user. It is highly customisable and provides much of the necessary features for us to implement our application. Swing does not have a component that makes choosing a date easily, so after research, we came across Jdatepicker, which is a GUI element that lets you choose a date from a calendar and for graphs we used Jfreechart, which will be further explained below. To ensure everyone on the team has the same library versions, we imported them though the pom.xml file, a file used by Maven to know information on and the configuration of the project.

5.3 Collecting Data

Collecting data is one of the main objectives of this project as data needs to be tracked to provide the user with accurate data from not only today, but also previous days past. To achieve this, we need two things: easy ways for a user to submit data and a way to store that data securely. To achieve the first point about a user submitting data, we need to create a GUI which makes this as easy as possible for the user. This includes simple a GUI interface which does not overwhelm the user. Proper prompts around text fields can make sure the correct data is submitted and for data such as dates, other methods of data submission can be used, for example, a calendar to select dates. The user should also be notified in the form of a helpful warning if the data they have submitted is incorrect to reduce confusion. Now that a user has entered data, it needs to be stored. We decided to store this on a database on a server. This allows for data integrity and ease of access to data, as well as reducing redundant data. An explanation on the design of the database can be found below in a separate section.

5.4 Graphs and Data Visualisation

One of the main requirements for this project is to allow a user to see their health and fitness routines progress. Now that data has been submitted and stored, it can be easily retrieved from the database. To plot this data on a graph, we used another library called Jfreechart, which allows us to create many different types of graphs such as bar charts and line graphs. This allows a user to visually see their data they previously entered on the graph and trends can be identified from the graph. For example, weight can be represented on a line graph, and you can easily tell the difference of weight between different days.

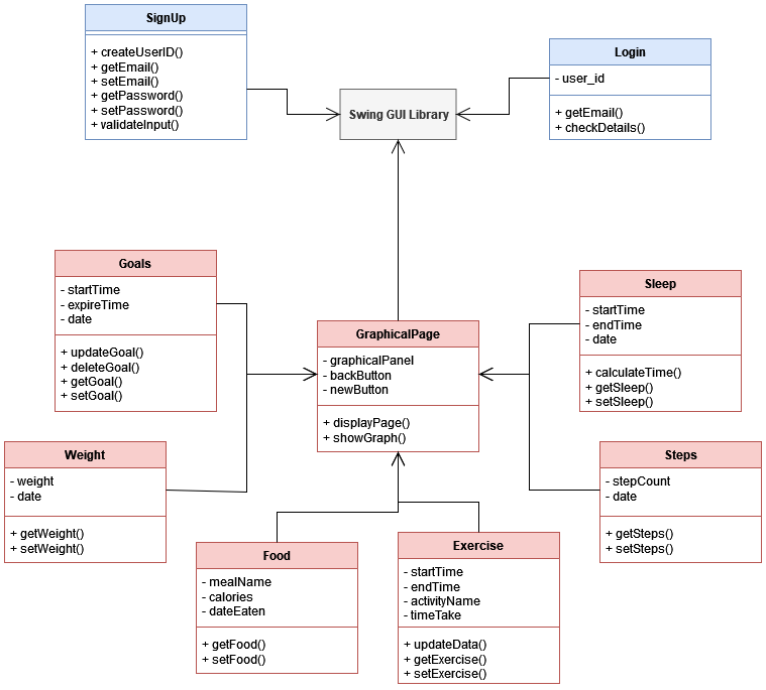
5.5 Goals and Achievements

Since we used object-oriented programming, we can easily create a section for goals by creating a "Goals" class. Because they will be created as objects, they can be made to be easily editable, for example, to change the description and target. Methods can also be designed to assist with deletion and completion of goals. The actual graphical design of representing goals will be very simple; there will be clear buttons to create, delete and complete a goal. There will be a clear text field so a user can enter the description of a goal.

5.6 Scalability and Maintenance

One of the main purposes of good programming design is to enable the final product to be easily maintained over the years, as well as being able to handle an increase in usage. To attempt this, we made sure to employ proper use of encapsulation; for example, everything related to goals will be in one class and everything related to sleep will be in another. Attributes and methods will have clear names describing their purpose and will have helpful comments and docstrings. This will make sections of code easy to understand for future developers. Furthermore, because of our database design, it can be easily scalable should demand for our application increases.

5.7 Class Diagram – Initial



This is our initial idea of our program and how the main classes interact with each other as shown in the class diagram. From Swing, we will be able to create user interfaces for all our classes. The classes will be inheriting and using the methods from classes in Swing, hence, there are arrows on the diagram to show this. The other major class is a general graphics page which will define the buttons and panels, a bit like a template. Classes such as Exercise and Goals inherit from this, then they can use the template to create their own interface, whilst incorporating their own features.

5.8 Sequence Diagram

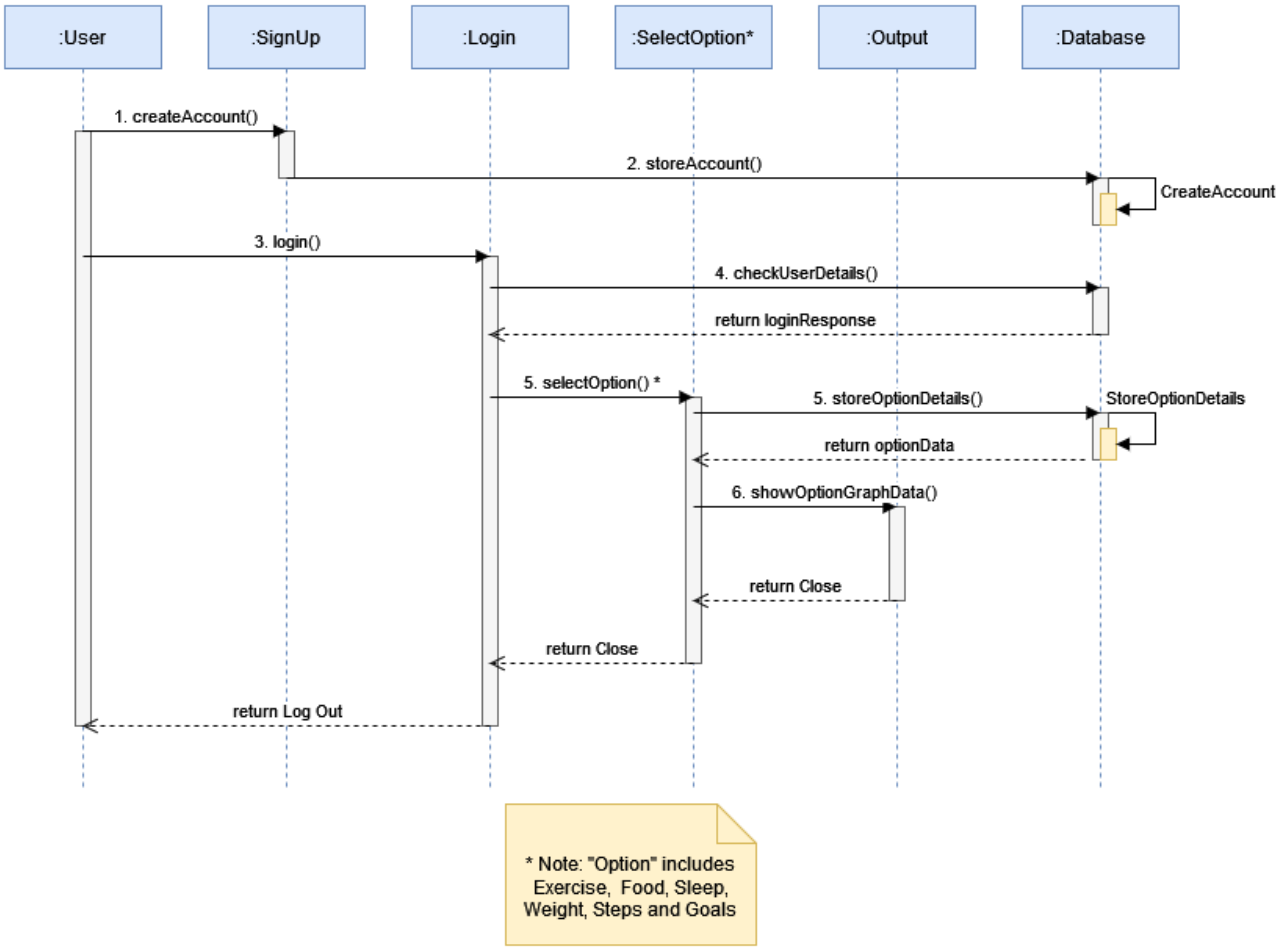


Figure 5.1: Sequence Diagram

The purpose of a sequence diagram is to show the messages sent between objects and how everything interacts with each other.

The diagram shows how a user would interact with our application. The user would create an account, which is then stored onto a database pending validation checks. If they have an account already, they can sign in. When signing in, the email and password are checked to see if they are valid and consistent to what appears in the database. If the user is successful, they will then be able to choose which option to enter data for: exercise, food, sleep, weight, steps or goals as explained in the note. They will then be able to enter and submit the relevant data for the option they chose. This data is then stored in the database because we want the user to be able to access past data when viewing a graph of the relevant option. A user can view this graph by pressing a button, in which another window with the graph will appear. Since this covers the utility of our program, the user can then quit the program and log out.

5.9 Class Diagram – Final

This class diagram of our final project is purely for documentation purposes to show how our project developed and expanded. Since this is the finished project, this diagram is more detailed with attributes showing their data type and methods showing parameters and return type. Both the frontend and backend class diagrams can be found in the appendix.

5.10 Frontend

This diagram refers to the frontend of the application, namely the user interfaces, inputs and outputs and validation. A difference between our initial conceptual class diagram and the final diagram is the inclusion of

folders, in which classes are in their related folders. The sections are the following:

- Program – this exists one level outside the other folders in the directory. This contains the "Main" file used to run the program and instantiate the objects to create the application, namely "MainFrame" and "MainGUIPanel".
- Start – this folder is focused on the login aspect. It has a login page and a sign-up page, as well a framework for the user interface. The main class tying these together is "StartPage" and is compositionally related to the other classes as they cannot exist without each other.
- MainProgram – this section deals with the fundamental part of our application: the options for submitting data. The options include exercise, food, sleep, weight, goals and steps. They inherit attributes and methods to create an interface from a class called "Subpage", as well as validation methods from the NonGUIElements folder. Furthermore, the Goals class has an inner class called "Goal". In this folder, there is also MainPage and Menu to create the main menu for selecting options, as well as TimeChart which is used to create graphs.
- NonGUIElements – This folder contains classes called Validation and DataSet. DataPoint is an inner class of DataSet.

All classes apart from Main inherit methods from library classes in some way, which is shown on the diagram using arrows to the "Library" section. Image can be found in the appendix labelled [5].

5.11 Backend

BackendCommunication is a separate folder in the same directory as the others and as the name suggests, it deals with the classes that communicate with the database i.e., the backend. The LoginBackend class inherits attributes and methods from the Requests and Json classes to aid with the communication with the database for login details. There is also WeightBackend, ExerciseBackend, FoodBackend and SleepBackend which are linked by aggregation with their respective data type: WeightData, ExerciseData, FoodData and SleepData. These are classes in the DataTypes folder. Image can be found in the appendix labelled [6].

5.12 Database Schema

This is the database schema to be used for storing data in the backend. This will be converted to MySql to create the tables for the database.

Figure 5.2: Database schema

5.13 Backend Requests

The backend application will be communicated with using HTTP requests, here is a plan of what those requests will be and what they will do. As you can see they line up with what is stored in the database.

Request Type	Request path	Description
GET	/check_email	Checks if an email is in use
POST	/signup	Signs up a new user
GET	/login	Logs in a user
GET	/meal	Gets meal data for a user
POST	/meal	Adds a new meal record for a user
GET	/weight	Gets the user's most recently recorded weight
GET	/weights	Gets all weight recordings for a user
POST	/weight	Adds new weight data for a user
GET	/sleep	Gets user's sleep data
POST	/sleep	Adds new sleep data for a user
GET	/exercise	Gets user's exercise data
POST	/exercise	Adds new exercise data for a user
GET	/goals	Gets goals for a user
POST	/goals	Adds new goals for a user
GET	/five_a_day	Gets five a day data for a user
POST	/five_a_day	Adds new five a day data for a user

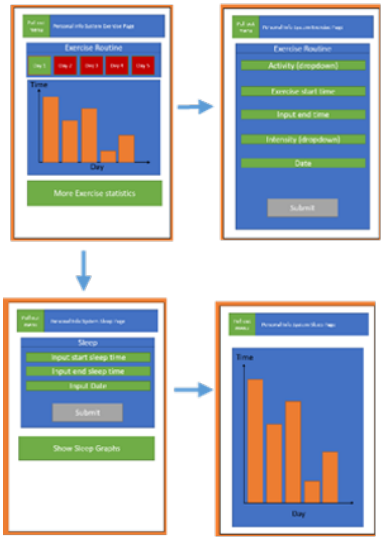
5.14 Prototype Designs

To make the prototype design of how we intended to make the program look like we decided on creating simplified page models within PowerPoint, this meant we could start coding and aim for the program to look ultimately like the PowerPoints.
To meet the requirements of the PI program, the decisions below have been taken:

This is a prototype of the first page that will be seen upon startup, this meets all the requirements for the login GUI. This is because as shown on the left it contains the Email and password inputs, The checkbox toggling password view and the login and signup buttons. The right image demonstrates the incorrect login details popup. In all the data input cases invalid details will be dealt with the same with related text

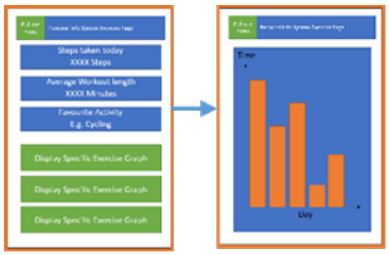
On the left this shows the sign-up page which is very similar in requirements to the login page however with a confirm password input. Once valid details have been provided the user is then shown the second sign up page which then asks for additional information, after this is provided and the last sign-up button is pressed the account is created the user is logged in and then brought to the main page.

The main page will essentially be a navigation page. This is where the user will be able to access the other pages / parts of the program. This is also where we plan to show user statistics such as averages and graphs. To navigate from here the user would use the pull-out menu to navigate to specific pages for data entry and more specialised data. The image on the right shows the prototype of how the sidebar will look.



Exercise Page: This contains graphs to view exercise history and a more exercise statistics button to view more specific / all data and recording workouts via clicking on one of the Day buttons that turn green on data input. The page has 6 user-modifiable fields: Activity, Start and end time, Intensity and Date to track to meet exercise requirement 1 and 2

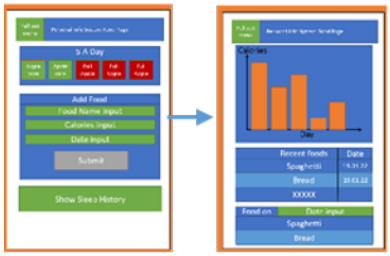
This is the more exercise statistics page. This is where more detailed information about the user's exercise data can be found and more specialised graphs. The image on the right is an example of how the specialised graph will be displayed. This is to meet the remaining exercise requirements.



Sleep Page: This is where the sleep information is collected using inputs such as start time, end time and the date to categorize the data. These inputs then allow for you to view the data later using the show sleep graphs button which then shows the data meeting all the sleep requirements.

Weight Page: This is where weight data is collected and shown, it is collected via the weight and date input which then allows for the graph to display data easily when clicked. The weight data will be shown in a similar way to the right screenshot which meets the requirements of the weight page.

Steps Page: This page is where steps are input by using the text input and the date input. The data can then be viewed with the steps graph button that then displays the date in a graph format on the user's screen meeting the requirements for Step tracker



Food Page: This page tracks your 5 a day consumption using a clickable button that switches colour and saves daily, what food you eat, and your daily calories and the date consumed. This is then displayed in graphs and a table using the show sleep history button using a chart, Recent foods and a way to search for foods users have had on specific dates which meets all food requirements.

Goals Page: This page contains user set goals for themselves. It allows them to add a new goal using the "Add Goal" button and the colour of the goal changes depending on how close to the due date it is and whether it has passed. It also can change to green if the tick button is clicked. The X on the left of the goal allows the user to delete that goal from the system. This meets all the requirements for the goals page

Overall, throughout the creation of the PowerPoint designs we have managed to meet all the requirements if all the features within are implemented for the GUI. This also shows a rough prototype of how the program will be traversed, used and what its used for and the ease of use within the GUI which makes it more user-friendly.

Software Testing

(Harrison Crane, Artiom Casian)

We will be conducting 4 different types of tests to ensure our application works correctly and follows the designed route.

Functional Tests:

Here we will ensure that the program meets all the predesigned requirements. Specifically, we test that:

- The user can create an account with a valid email and password or if they already have an account then they should be able to log into it.
- The main menu is working, and the user can access all the tabs.
- The food tab allows the user to keep track of their 5 a day as well as of the number of calories they've taken.
- The user should be able to input their data in the exercise tab.
- User should be able to store his sleep time in the sleep tab.

Integration Tests

Integration tests should verify that different modules or services used by your application work well together. Mainly we should ensure that:

- All the feature tabs interact well with the main menu tab.
- Backend interacts well with the frontend (backend being the database to store all the users' data and frontend being the program itself.)

Unit Tests

Unit tests are very low level and are designed to test very specific parts of the program. We will be using unit testing the following methods:

- Email validation method
- Password validation method
- Unit Input Validation (time and distance)

Usability Tests

We will conduct usability testing to ensure that the application is easy to use and navigate.

6.1 Evidence of Testing and Test Results

Functional Tests

Functional tests were run throughout the development process. Every time a new feature was implemented, we were running the program to ensure that everything is working as designed and if there were errors, we made sure to fix them.

Integration Tests

The backend was tested using the 'postman' application. It allows you to run HTTP requests to check functionality. It also lets you create automated tests using JavaScript to ensure the requests are functional. This software was chosen over manually testing the HTTP routes due to the ability to save different requests to be run multiple times. It also creates API documentation automatically which was very beneficial when connecting the frontend to the backend. The API documentation created can be found here on Postman.

Here is an example for storing then getting exercise data using postman:

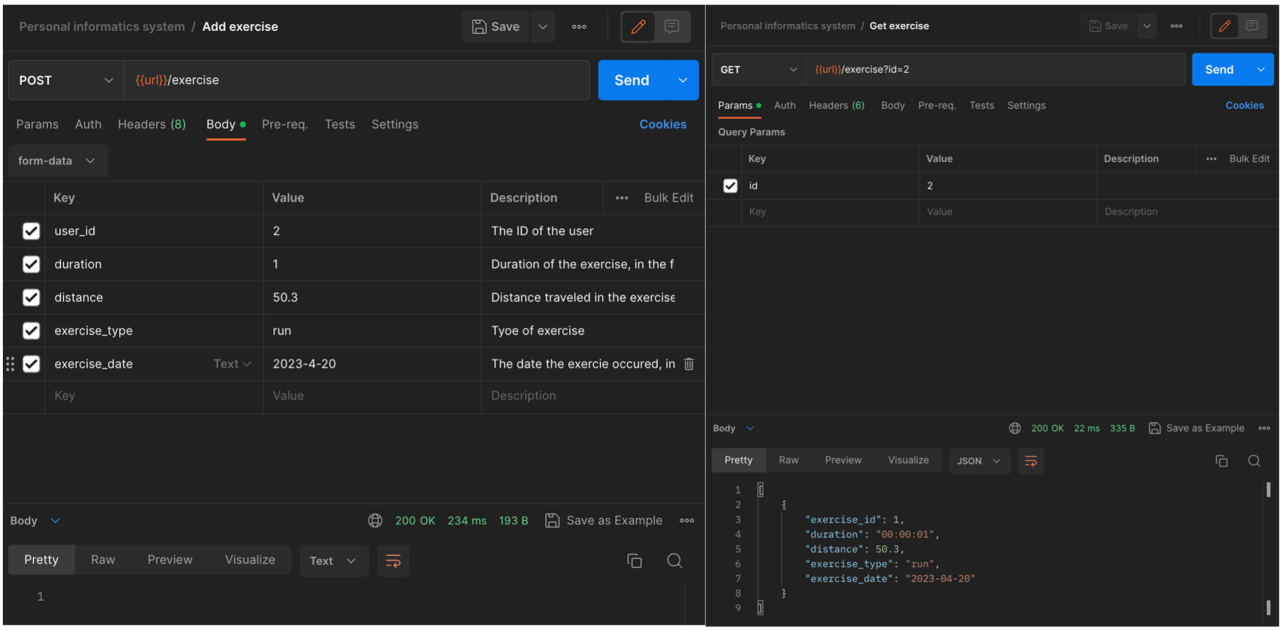


Figure 6.1: Testing exercise using postman

First a POST request is sent to store the exercise data, then a GET request to retrieve that data and therefore check that it has been stored.

Unit Tests

Unit tests were used on our two main validating methods that are: checkPasswordIsSecureEnough() and checkEmailIsValidFormat(), as well as unit tests functions for validating user input from different menus.

As the result of those unit tests, we ensured that the program works correctly and can handle wrong inputs without crashing or causing errors.

(Screenshots attached as evidence in the appendix)

Usability Tests

Usability of the program was tested at the end of the sprints to ensure that all methods and features work as planned.

Reflection and Conclusion

(Alexander Agafonov, Alex Clarke)

7.1 Reflection

The goal of this project is to create a wellbeing and health tracking app using an agile methodology., We can measure the success of the project by considering multiple things such as whether we met the requirements and how well we followed the agile methodology.
The table below shows a summary of the requirements and whether they have been met:

Requirements	How we met it / didn't meet it	Met
Non-Functional: Tools to create our software	Out of these requirements we managed to meet all these requirements which are confirmed by the two GitHub repositories linked above and the files and code within matching to meet the requirements.	yes
Non-functional: Testing	We met these requirements by bug-checking other developers code and features on commit, this can be shown with the kanban board history and the phases of the different issues passing through peer review. The Unit testing was also completed which is evidenced in the above page in the document	yes
Non-functional: Scalability	All the requirements here were met. This is evidenced within the video and PowerPoint designs. The program was designed with this clearly in mind and there is a main page with space and functionality to add more features and the entire program follows the single theme. It's also clear parts are reused within the program such as the graphs and date input. Lastly the backend is scalable due to the design of the MySQL database attached which is evidenced in the Design section of this report the database can be accessed across the world and so can support multiple users and store all current and new data via HTTP requests and so definitely meets the requirement	yes
Functional: Login and Signup System	All the requirements in this section are met, this evidenced within the video as the login system works bringing the user to the main page, validity checking the login details and the inputs ensure the sub-requirements are met. By the end of the video the signup system is evidenced, and the input validation is shown with the popup window. It's also clear the account was created within the program's backend as the user was logged in after signup therefore, we believe the requirements of this section have been met	yes
Functional: Menu	The requirements within this section have been partially met, Requirement 3 within this section is met and evidenced in the above section in this report however Requirement 1 and 2 could be said to have not been met. This is shown in the video as there is no pull-out bar menu and customisable menu.	1/2
Functional: Exercise	As evidenced by the video the exercise page displays all the functionality to meet the requirements including graphs using previous data and the steps page. It is shown the functionality of the exercise statistics is working as the input data in the video is shown clearly on the statistics page and in all the ways mentioned within the requirements	yes
Functional: Sleep Tracker	The sleep page is evidenced, and functionality clearly shown within the video showcasing the features within that clearly satisfy the requirements such as the sleep graph and data entry.	yes

Functional: Weight Tracker	Similarly, to the sleep page this is evidenced and functionality clearly within the video showcasing the data entry including date with a popout calendar and a chart button satisfying the requirements	yes
Functional: Step Tracker	The Step tracker is showcased within the video displaying data entry and showing charts satisfying the requirements	yes
Functional: Food	Evidenced by the video and prototype PowerPoint designs, the Food page displays the 5 a day functionality, the food entry and the food history button leading to the statistics page therefore satisfying all the requirements for this section	yes
Functional: Goals	Shown by the video the goals page demonstrates the functionality needed for the requirements such as the ability to add and delete goals, mark them complete and have due dates and the GUI for this is in the format asked for by the requirements and so meets all the requirements for this section	yes
Functional: Backend	The backend functionality is demonstrated within the video and evidenced within this document, it is clear the backend can be connected to, includes capability for multiple unique users simultaneously and returns information in the correct format from GET and POST requests by the unit testing and the video showcase functioning. It's also clear data retrieved such as date and user information are returned in a Json and in the correct date format as otherwise the graphs within the program would not function as intended. Functionality such as saving and retrieving data is also shown within the video as this is needed for the graphs to be created. Therefore, as all of this is met this section has met all its requirements	yes

Overview

From this we can see that we have met almost all of the requirements we have set ourselves for this project. The only requirements section that was not fully fulfilled was with the main menu, The requirements that were not met relate to the side bar menu within the program and a fully customisable menu. Due to time limitations and wanting to get the core functionality completed before this we decided to switch from this approach. The new approach we went with was a much simpler implementation of a main page and a back button to traverse the program.

In the end the product created was extremely close to the planned product and we have met all but 2 of the requirements. Therefore, on the requirements end we can consider this project a success.

Potential future improvements

Some of the things that we would potentially include if we continued this product would be API integration, The menu revamp with sidebar and overall, more automated functionality for the user.

7.2 Conclusion

After each sprint we discussed not only our tasks for the next sprint, but also an evaluation of our sprint backlog to see if the tasks on there were still within our scope. After the first sprint, we quickly realised that implementing Terra API would not be a good use of resources and time. This API was new for everyone in our group, so learning and implementing this API would be difficult. Furthermore, we did not have many devices to test the API on, so we did not know what to expect from this API, other than it could gather data from multiple sources efficiently.

Another idea we had at the start was to deploy our application onto a website as it would make the application more accessible. However, this idea was quickly dropped as this too would take a huge commitment to accomplish. It would not fit into our time constraints of our project. However, this would be a good idea to implement in the future as there are no downsides to making our application more accessible.

After the second sprint, we would be planning for our final sprint. Since it was the final sprint, we knew which tasks we could do or not do due to time constraints. The most notable omission was the addition of a reward shop system relating to goals. When a user completed a goal, they would receive coins which could then be spent in the shop. Other notable omissions were a pull-out menu and a scaling down of details for some of the options. For example, diagrams of food were not included to create cards of the food. They could be implemented in the future is we decide to continue development as these are useful and interesting features.

During development of the backend, we were initially going to an application called Docker to run it on our local machines. However, there were issues with this, and we could not get it to work. To get around this, we had to make a change and deployed it to a web server instead.

We believe our approach and implementation of the agile / scrumban methodology has been successful in creating an efficient, working application and we believe it was successful in meeting our target market of students. While we mostly implemented the methodology successfully, there was one main issues, namely having to extend the first and second sprints to ensure the tasks have been completed as an example. Because of this pushback of our backlog, development time of other features was reduced. In terms of sprint planning, we managed to do this effectively; every group member attended all the meeting scheduled. This made planning and assigning tasks very easy. The tasks will then be added to the sprint and kanban boards so everyone can see what tasks they need to complete. After a sprint was complete, we would all come together to do a sprint review to see how the sprint went and what needed to be done for the next sprint. This process was proven to be efficient as we created a functional product. The team spirit was good throughout the project. We kept detailed "meeting minutes" documents which describes what we discussed in our meetings. There are also screenshots of our sprint and kanban boards to show the development of our project through tasks.

We also believe we created a nice looking, simple user interface which achieved the goal of being as user-friendly as possible. The functions of each button and text field are clearly labelled and the interface itself does not have a complicated, messy look. To see the look of the application, refer to our video going through our program. Our programming structure was also effective as described in other sections. Our program is designed to be scalable and maintainable for the future.

Appendix

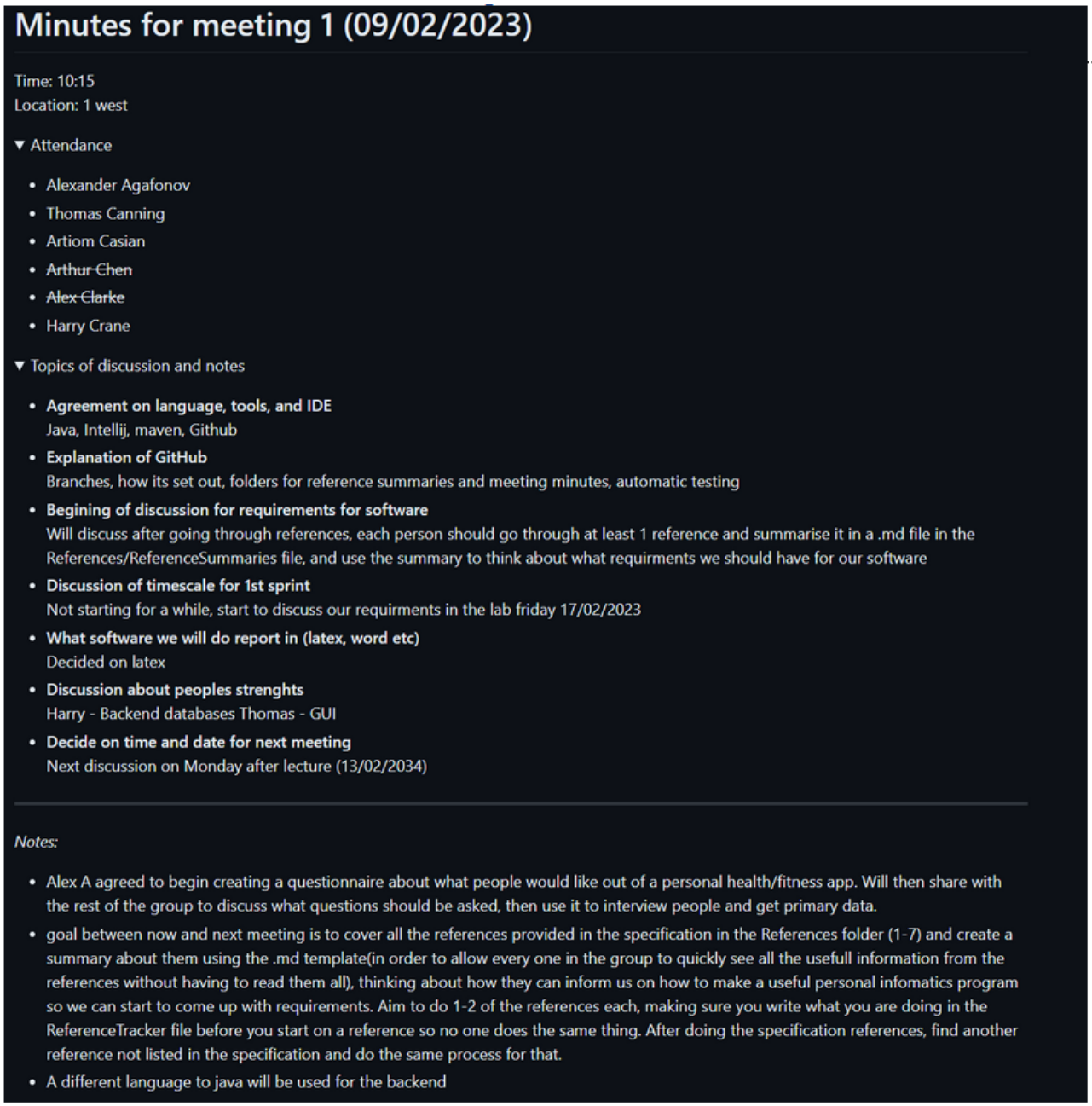
8.1 Group Contribution Form

Group Member	Contribution	Signature
Thomas Canning	10	
Alexander Agafonov	8	
Harry Crane	8	
Artiom Casian	8	
Alex Clarke	8	

8.2 Images

Meeting Minutes

[1]



Minutes for meeting 1 (09/02/2023)

Time: 10:15
Location: 1 west

▼ Attendance

- Alexander Agafonov
- Thomas Canning
- Artiom Casian
- ~~Arthur Chen~~
- ~~Alex Clarke~~
- Harry Crane

▼ Topics of discussion and notes

- **Agreement on language, tools, and IDE**
  Java, Intellij, maven, Github
- **Explanation of GitHub**
  Branches, how its set out, folders for reference summaries and meeting minutes, automatic testing
- **Begining of discussion for requirements for software**
  Will discuss after going through references, each person should go through at least 1 reference and summarise it in a .md file in the References/ReferenceSummaries file, and use the summary to think about what requirments we should have for our software
- **Discussion of timescale for 1st sprint**
  Not starting for a while, start to discuss our requirments in the lab friday 17/02/2023
- **What software we will do report in (latex, word etc)**
  Decided on latex
- **Discussion about peoples strenghts**
  Harry - Backend databases Thomas - GUI
- **Decide on time and date for next meeting**
  Next discussion on Monday after lecture (13/02/2034)

Notes:

- Alex A agreed to begin creating a questionnaire about what people would like out of a personal health/fitness app. Will then share with the rest of the group to discuss what questions should be asked, then use it to interview people and get primary data.
- goal between now and next meeting is to cover all the references provided in the specification in the References folder (1-7) and create a summary about them using the .md template(in order to allow every one in the group to quickly see all the usefull information from the references without having to read them all), thinking about how they can inform us on how to make a useful personal infomatics program so we can start to come up with requirements. Aim to do 1-2 of the references each, making sure you write what you are doing in the ReferenceTracker file before you start on a reference so no one does the same thing. After doing the specification references, find another reference not listed in the specification and do the same process for that.
- A different language to java will be used for the backend

Minutes for meeting 2 (13/02/2023)

Time: During Monday lecture (17:15)
Location: 8W1.1

▶ Attendance

▼ Topics of discussion and notes

- Discuss what has been learned from looking at the referances and start to think about requirements for our program.
- Work out what else needs to be done to get the marks for problem analysis section.
- Discuss questionnaire questions and potential requirements from "interviews" with other students.
- Work out when next meeting will be and what should be done before then.

Notes:

- Terra API for importing fitness data
- Agreed to use kanban board
- Decided on making a Wellbeing app
- First sprint will focus on developing a minimum viable product, later sprints will implement features like multiple users, logins, and more fitness and wellbeing features.
- Scrum master: Thomas C
- Product Owner: Alex A

Minutes for meeting 3 (17/02/2023)

Time: During Friday tutorial (12:15)
Location: CB 4.1

▶ Attendance

▼ Topics of discussion and notes

- Which Java library are we going to use for our frontend.

Java swing library, also able to make graphs using it to track data over time.

- Begin to gather requirements for program and think about the 1st sprint.
  Requirements we talked about have been collected in the markdown file in project requirements and Alex A will begin to add them to the Kanban board
- Discuss questionnaire results if any have been collected

1 questionaire response has been collected, link to the microsoft form has been put in the repository so anyone can use the questionaire.

- Work out when next meeting will be and what should be done before then.

Next lecture, but we will start having daily scrum on Wednesdays if needed

- Rust will be used as the language for the backend and a new repository will be created by Harry Crane to handle this

Minutes for meeting 4 (20/02/2023)

Time: During Monday Lecture (17:15)
Location: 8W1.1

▶ Attendance

▼ Topics of discussion and notes

- Brining requirements together and continuing planning 1st sprint

Java swing library will be used for GUI, along with a seperate library to make charts showing data over time.

Minutes for meeting 5 (24/02/2023)

Time: Friday tutorial (13:15)
Location: CB 4.1

▶ Attendance

▼ Topics of discussion and notes

- Discussion of what everyone will do in the first sprint
  Thomas - Login/signup GUI and Menu GUI Harry - Backend for account information
- Explanation of how the code is organised
  Main is the entry point of the program which creates an instance of MainFrame. MainFrame extends JFrame, and when the program is run an instance of StartPage(which extends MainGUIPanel) is added to the frame. StartPage switches between the various classes that extend JPanel to provide a GUI for logging in and creating an account. After the user logs in, MainFrame removes StartPage and adds MainPage which is the GUI for the main program (also extends MainGUIPanel). MainPage initially adds an instance of Menu where the user is able to go to any of the pages of the app (Food, Exercise, Sleep). When the user presses at button in Menu, MainPage removes the panel it is displaying and adds the newly selected panel.
- Design initial program
  An initial design of the program to be implemented in Sprint 1 has been created as a powerpoint in ProjectInformation

Minutes for meeting 6 (27/02/2023)

Time: Monday after maths lecture (14:05)
Location: EB balcony

▶ Attendance

Deciding what everyone is doing for sprint 1

- Alex A - Sleep Menu
- Alex C - Food Menu
- Artiom - Email and login verification / validation
- Thomas - Login/signup and Main menu GUI
- Harry - Backend / Login Details

Minutes for meeting 7 (03/03/2023)

Time: Friday tutorial (12:15)
Location: CB 4.1

▶ Attendance

▼ Topics of discussion and notes

Sprint standup

- Alex A - Begining work on sleep section
- Alex C - Begining work on food section
- Artiom - Begining work on Email and login verification / validation
- Thomas - Finsihed creating sign up and login page, working on main menu. Found a library that can be used for implementing a pullout menu
- Harry - Backend for user login details created including functions for logging in, checking email, and creating an account. Ready to be linked with frontend

Minutes for meeting 8 (10/03/2023)

Time: Friday tutorial (12:15)
Location: CB 4.1

▶ Attendance

Sprint retrospective

The first sprint has been slow because the focus has been on the development of the GUI. This is due to 5/6 of the group being unfamiliar with the Java Swing GUI library. The Login and email validation is complete. Unit tests for these functions have not been created however, and this task has been moved to the second sprint.The Backend has been created in Rust with storage of different types of data for the user login info being implemented, it has not yet been linked to the front end due to unfamiliarity with Java http requests, and this task has been moved to the second sprint. The majority of the tasks in this sprint have been focused on initially setting up the GUI. A login and signup GUI system has been created. This is likely finalised however there is space to add additional fields to collect more information about users if requirements change. Additionally, a menu page has been created for the GUI, allowing the user to switch between the sleep, food and exercise pages. The functionality for changing between these pages has also been implemented. However, the buttons for switiching between panels were planned to be placed on a pullout menu, but this task has been moved to the 2nd sprint. The sleep page has been worked on with 2 fields created to enter sleep information, and a functional back button has also been added to go back to the menu. The Food page is also being worked on.

In the next sprint things should go more smoothly and faster as familiarity with the GUI has increased and the backend can start to be linked to the frontend. The menu, food and sleep pages will also be finalised.

- WWW:
  - Backend complete
  - Login validation complete
  - EBI:
  - GUI pages completed faster

Sprint standup

- Alex A - Continuing work on sleep section, Created display for sleep, working on back button - Alex C - Continuing work on food section, Working on button panels and inputs - Artiom - Finished work on Email and login verification / validation, Creating Unit tests - Thomas - Worked on menu GUI, created functionality for switching pages in the program - Harry - Implementing storage of different data e.g. weight into the backend.

Minutes for meeting 9 (20/03/2023)

Time: Monday 14:15
Location: EB Balcony

▶ Attendance

Planning sprint 2

- Created Kanban board for sprint 2 and moved uncompleted tasks from sprint 1 to the new board.
- Updated designs for food and exercise GUIs in powerpoint
- Decided on what needs to be stored in the database for the new features
- Decided who is doing what for the next sprint:
  - Thomas: Exercise page
  - Harry: Updating backend with new data fields and connecting it to the front end
  - Alex A: continuing sleep GUI adding feature to store start and end times of sleep and track sleep over time
  - Alex C: continuing food GUI, adding ability to track 5 a day and food history
  - Artiom: Creating unit tests for user input, specifically for login, and unit tests for checking function that calculates length of sleep is correct

Minutes for meeting 10 (24/03/2023)

Time: Friday 12:15
Location: CB 4.1

▶ Attendance

Finishing sprint planning for sprint 2

Continueing selecting requirements from sprint backlog to move to in progress on kanban board for sprint 2. Requirements for sprint 2 revolve around developing sleep, exercise and food pages, specifically adding graphs to display.

Designed what the GUI will look like for the graphs.

Talked about how we will do UML designs for our software

Discussed timeline for future sprints

Sprint 2 will start for the next 2 week

Minutes for meeting 11 (31/03/2023)

Time: Friday 12:15
Location: Online

▶ Attendance

Sprint standup

- Alex A - Created sleep section with text boxes to enter sleep start and finish time
- Alex C - Created 5 a day button array for food section
- Artiom - Created tests for email and password validation methods
- Thomas - Created a weight section with a chart to display weight over time, and created data inputs for exercise page
- Harry - Working on docker for backend, and starting to add functionality for goals, weights, exercise, and food data for backend.

This sprint is progressing much more smoothly than the last sprint, as everyone is more familiar with the Java GUI library, and Harry is becoming more familiar with the tools for the backend.

Minutes for meeting 12 (17/04/2023)

Time: Monday after maths lecture (12:15)
Location: EB balcony

▶ Attendance

Sprint retrospective

More progress has been made in sprint 2 than in sprint 1 because of increased familiarity with Java and creating GUIs. All of the main GUI pages we want to create have been created apart from the goals page. Goals will be implemented in sprint 3. Apart from goals, all places where data is entered into the program have now been completed. The only feature that has not yet been able to be implemented in sprint 2 has been adding charts to the program to visualise data. This will be the focus of sprint 3, along with adding the goals section. Additionally, the backend has now been completed with functionality for all the data we want it to be able to store, however it has not yet been connected to the front end and this task will be moved over to sprint 3.

In the next sprint we should have a completed product with all the features we wanted to implement to meet the requirements. However, we are not sure at this point if we will have time to implement compatabilty with the terra API.

Minutes for meeting 13 (20/04/2023)

Time: Monday after maths lecture (13:15)
Location: EB balcony

▶ Attendance

▼ Topics of discussion and notes

Starting sprint 3 today, unfinished tasks from sprint 2 have been moved to the sprint 3 kanban board which has now been created. Added the rest of our requirements which haven't yet been completed to the sprint 3 backlog such as adding charts to all the sections to display data, and fully connecting the backend to the frontend.

Minutes for meeting 14 (24/04/2023)

Time: Monday 14:15
Location: East Building Balcony

▶ Attendance

Sprint standup sprint 3

- Thomas has finished goals section, adding a new page to the menu that gives the ability to add new goals. Each goal has a due date and a description of the goal. Goals can be marked as complete or deleted. Marking a goal as complete turns it green. If the due date of a goal passes and the goal has not been marked as complete, the goal turns red.
- Harry deployed the backend to a webserver and setup a MySql databse server so that the backend can be accessed. He also investigated and implemented JSON parsing in java using the jackson library so that the data retrieved from the backend can be used.
- Alex C has finished the Food section, Now allowing users to add food data to the program.
- The programming deadline for the project has been set as this Friday, from which no new code will be written. We will then finalise the report in the following.
- Discuessed everyones progress with the report and what will be completed before friday.

Minutes for meeting 15 (28/04/2023)

Time: Friday tutorial (12:15)
Location: CB 4.1

▶ Attendance

Sprint retrospective for sprint 3

We now have completed programming for our project. We have met every requirement needed for the coursework. The last week has involved finsihing adding charts to every section in the program so stored data can be reviewed and compared. We have also connected more of the backend. This sprint went the best compared to all the prior sprints, as by this point we are much more familiar with the swing GUI library.
What went well?
We are now at a stage where programming is complete, and we can now focus on the report. We have produced a final product that meets all the initial requirements, and has also grown from the initial requirements as new requirements have developed along the way.
What could have been improved
The workload was less evenly distributed in this sprint, with the majority of the coding for this sprint on the front end being done by Thomas, and all the code for the backend and connecting the backend to the front end being done by Harry. What could have been improved in this sprint is allocating the task of connecting the front end to the backend to someone else, in order to split work up better so we could have progressed through this sprint faster.

Kanban Boards

[2]

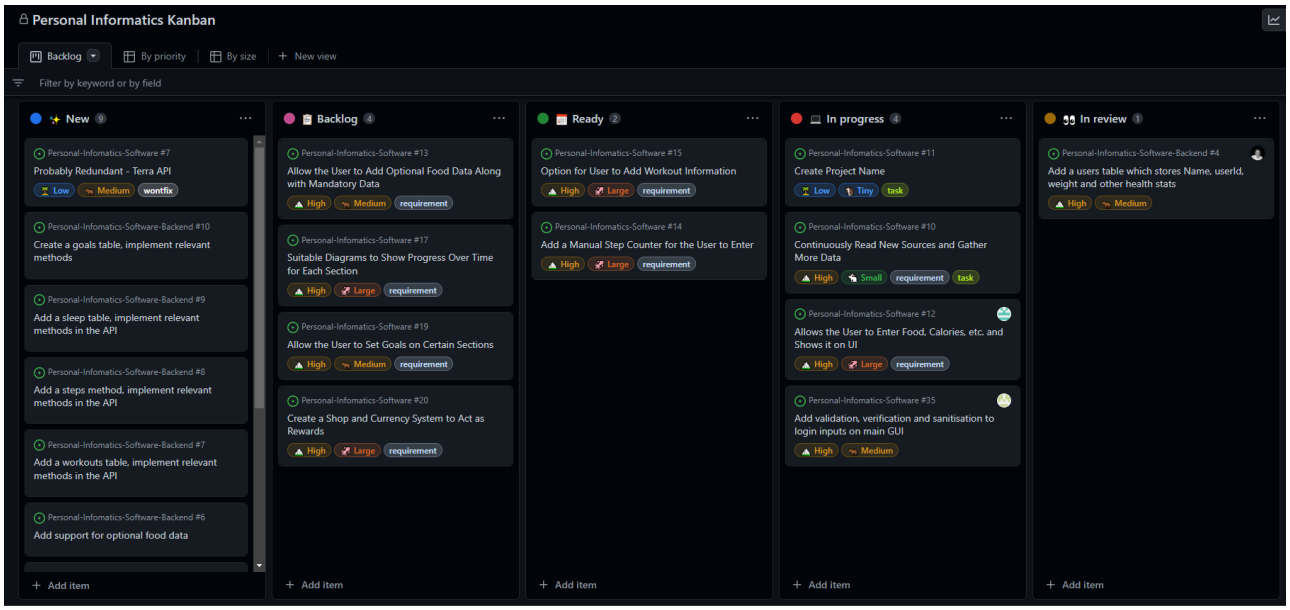


Figure 8.1: Our general overall kanban board.

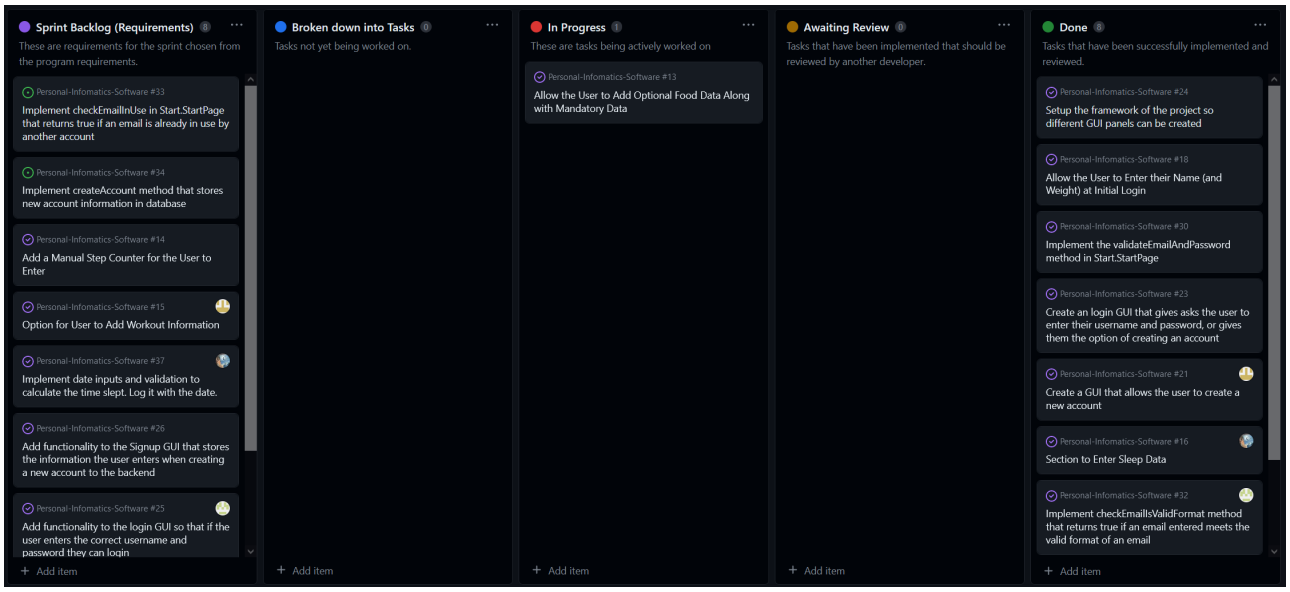


Figure 8.2: Sprint 1 kanban board.

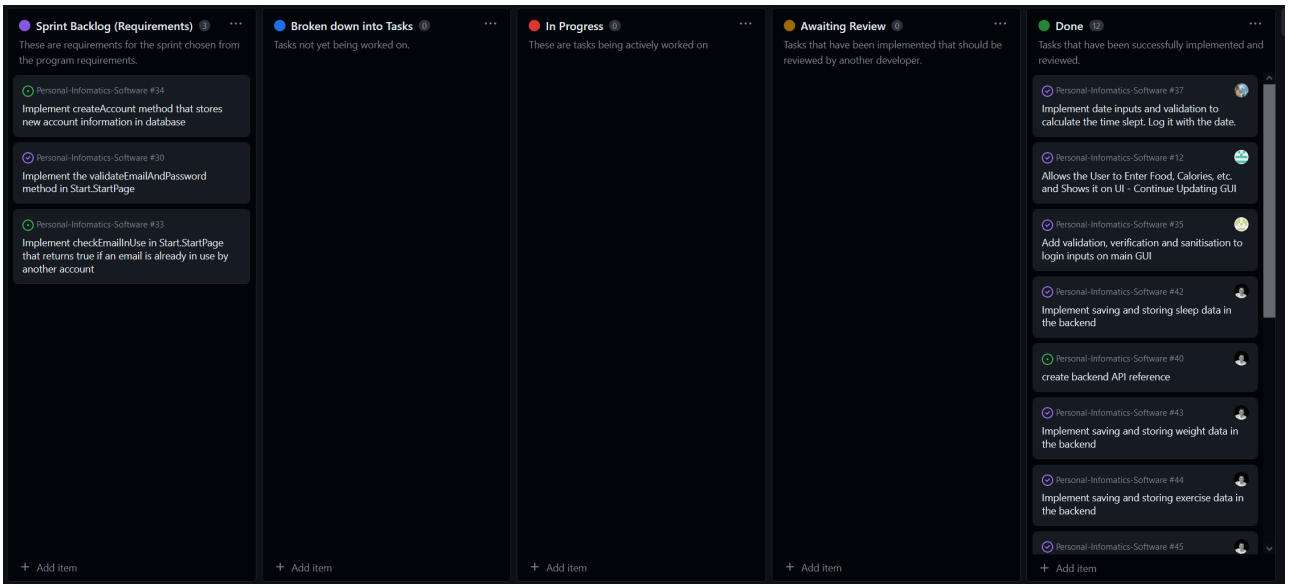


Figure 8.3: Sprint 2 kanban board.

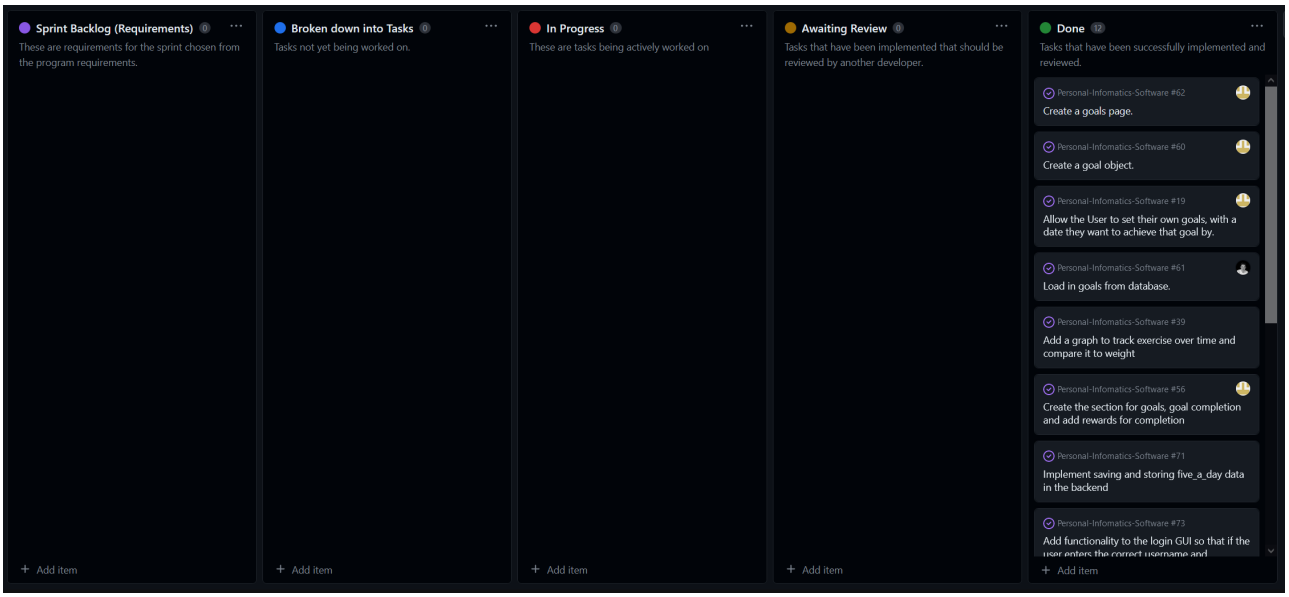


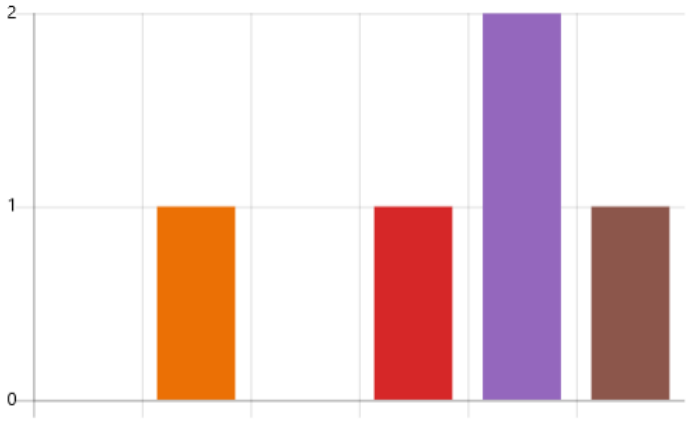
Figure 8.4: Sprint 3 kanban board.

Questionnaire

[3]

1. How many days do you exercise a week?

0	0
1	1
2	0
3	1
4	2
5+	1



2. On your average session, how long do you exercise for?

0 Hours	0
< 1 Hour	1
1-2 Hours	4
2+ Hours	0

3. Do you use tracker apps e.g. step counter?

Yes	2
No	3

4. If not, why?

3
Responses

Latest Responses
"Too much effort"
"useless "

5. What data do you want to track when looking to download a tracker app?

Steps	1
Calories	1
Sleep Data	1
Heart Rate	1
Something Else	1

6. Do you prefer using one app to track multiple things (as close to), or multiple apps to each track one thing?

One app to track multiple thin...	5
Multiple apps to each track on...	0

7. Do you care about the number of calories you are consuming and the 'healthiness' of the food (e.g. sugars, salt, protein etc.)?

Yes	3
No	0
From time to time	2

8. Is it beneficial for you to track food details like above (calories, carbs etc.)?

Yes	3
No	2

9. Do you, or do you want to track the number these details in food?

Yes	3
No	2

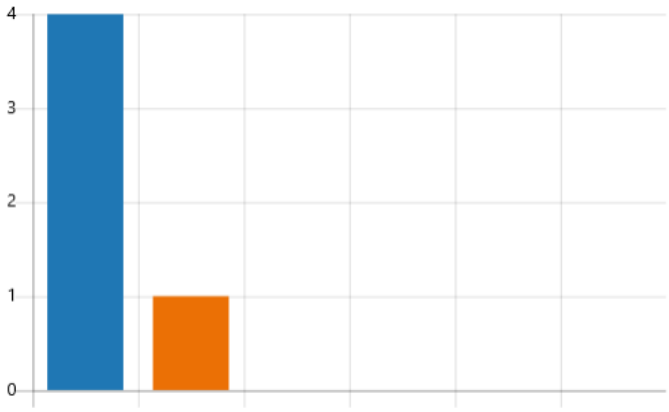
10. Are there other details you track, either health related or not?

2
Responses

Latest Responses
"*no*"

11. How often do you use a computer?

Everyday for a most of the day	4
Everyday but only for a short ...	1
Once every 2 days	0
Used between 3-5 days	0
Weekly	0
Never	0



Code Extracts

[4]

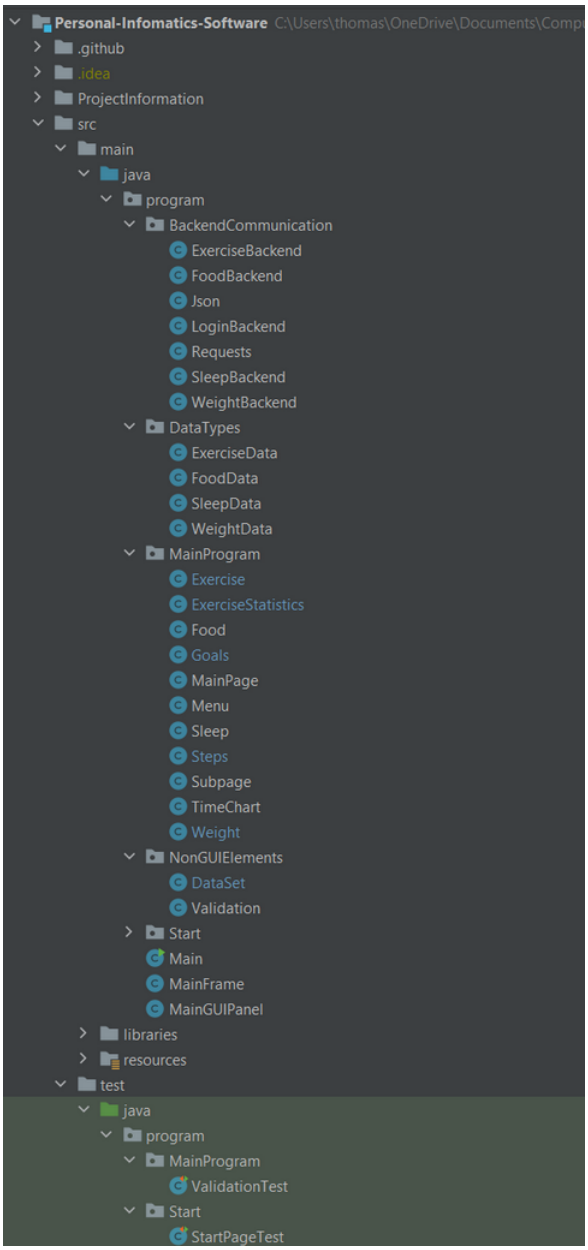


Figure 8.5: Structure of packages and classes in the Java code.

```java
public class MainFrame extends JFrame {
    public MainPage mainPage;
    6 usages
    public StartPage startPage;

    ThomasCanning
    public MainFrame() {

        //Setting up the main frame
        this.setTitle(Main.NAME);
        this.setExtendedState(JFrame.MAXIMIZED_BOTH); //Maximizes the frame on startup
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setMinimumSize(new Dimension( width: 400,  height: 620));
        ImageIcon frameIcon = new ImageIcon( filename: "src/main/resources/logo.png");
        this.setIconImage(frameIcon.getImage());
        startPage = new StartPage( mainFrame: this);
        mainPage = new MainPage( mainFrame: this);
        this.add(startPage);
        this.setVisible(true);

    }
    /**
     * Method to switch between the StartPage and MainPage JPanel's
     * @param panel The panel to be added to the frame
     */
    ThomasCanning
    public void SwitchPanel(JPanel panel){
        this.getContentPane().removeAll(); //Removes all components from the frame
        this.add(panel); //Adds the panel passed in to the frame
        this.revalidate();
        this.repaint(); //Refreshes the frame
    }
}
```

Figure 8.6: Mainframe class that is instantiated when program is run. mainPage and startPage are added to it. startPage deals with logging in and signing up, and mainPage is the GUI for everything after logging in.

```java
public class Subpage extends JPanel {
    protected JButton backButton;
    74 usages
    protected JPanel mainPanel;

    7 usages  ThomasCanning
    public Subpage() {
        this.setBackground(new Color( r: 230,  g: 126,  b: 34));
        this.setBorder(BorderFactory.createLineBorder(new Color( r: 52,  g: 73,  b: 94),  thickness: 3));
        this.setLayout(new BorderLayout());

        JPanel topPanel = new JPanel();
        topPanel.setLayout(new FlowLayout(FlowLayout.LEFT));
        ImageIcon backArrowIcon = new ImageIcon( filename: "src/main/resources/back-arrow.png");
        Image backArrowImage = backArrowIcon.getImage();
        Image newBackArrowImage = backArrowImage.getScaledInstance( width: 30,  height: 30, Image.SCALE_SMOOTH);
        ImageIcon backArrow = new ImageIcon(newBackArrowImage);
        backButton = new JButton(backArrow);
        topPanel.add(backButton);
        this.add(topPanel, BorderLayout.NORTH);

        mainPanel = new JPanel();
        mainPanel.setBorder(BorderFactory.createEmptyBorder( top: 50,  left: 100 ,  bottom: 100,  right: 100));
        this.add(mainPanel, BorderLayout.CENTER);
        mainPanel.setLayout(new GridBagLayout());

        this.setVisible(true);
    }
}
```

Figure 8.7: Subpage class that every subpage (e.g. exercise, food) extends from so they have the same look.

```java
public MainPage(MainFrame mainFrame) {
    super(mainFrame);
    exercisePanel = new Exercise( mainPage: this, mainFrame);
    foodPanel = new Food( mainPage: this, mainFrame);
    sleepPanel = new Sleep( mainPage: this, mainFrame);
    menuPanel = new Menu( mainPage: this, mainFrame);
    weightPanel = new Weight( mainPage: this, mainFrame);
    goalsPanel = new Goals( mainPage: this, mainFrame);
    startPage = mainFrame.startPage;
    stepsPanel = new Steps( mainPage: this, mainFrame);
    exerciseStatistics = new ExerciseStatistics( mainPage: this, mainFrame);

    this.add(menuPanel);
    }
```

Figure 8.8: An instance of each of the subpages of the main program is created in MainPage. Initially menu panel is then added to the MainPage as that is the first thing that displays. Every page takes the MainPage instance as a parameter (this) so that the back button can be programmed to take you back to the previous page.

```
public Weight(MainPage mainPage, MainFrame mainFrame) {super();

    JLabel weightLabel = new JLabel( text: "Weight Tracker");
    weightLabel.setFont(new Font( name: "Arial", Font.BOLD,  size: 25));
    weightLabel.setBorder(BorderFactory.createEmptyBorder( top: 0,  left: 0,  bottom: 30,  right: 0));

    //set layout to gridbaglayout and add components
    GridBagConstraints c = new GridBagConstraints();
    c.fill = GridBagConstraints.HORIZONTAL;
    c.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);

    c.gridx = 0;
    c.gridy = 0;
    mainPanel.add(weightLabel, c);

    c.gridx = 0;
    c.gridy = 1;
    mainPanel.add(new JLabel( text: "Enter weight:"), c);

    c.gridx = 0;
    c.gridy = 2;
    weightTextField = new JTextField();
    weightTextField.setPreferredSize(new Dimension( width: 200,  height: 20));
    weightTextField.setText("Enter weight in kg");
    mainPanel.add(weightTextField, c);
```

Figure 8.9: An extract from the weight Subpage showing how the subpages were organised with a GridBagLayout, with each GUI element being created and added to the JPanel.

```
weightTextField.addFocusListener((FocusAdapter) focusGained(e) → {
        super.focusGained(e);
        if (weightTextField.getText().equals("Enter weight in kg")) {
            weightTextField.setText("");
        }
});
//Only allow up to 3 digits before the decimal point and 2 digits after
weightTextField.addKeyListener((KeyAdapter) keyTyped(e) → {
        char c = e.getKeyChar();
        if (!(Character.isDigit(c) || c == '.')) {
            e.consume();
        } else if (c == '.' && weightTextField.getText().contains(".")) {
            e.consume();
        } else if (Character.isDigit(c) && weightTextField.getText().indexOf('.') != -1
                && weightTextField.getText().substring(weightTextField.getText().indexOf('.')).length() == 3) {
            e.consume();
        } else if (weightTextField.getText().length() >= 6) {
            e.consume();
        }
});
```

Figure 8.10: Another extract from the weight SubPage, showing how the text fields have action listeners that only let certain characters be entered. In the case of the weight text box, it only allows up to 3 digits before the decimal point and up to 2 after.

```java
public static Response post_request(String sub_dir, Dictionary<String, String> form) throws IOException {
    HttpUrl.Builder urlBuilder = Objects.requireNonNull(HttpUrl.parse( $this$parse: base_url + sub_dir)).newBuilder();

    // Create a new form and add the values from the passed in dictionary
    var body = new FormBody.Builder();
    for (Enumeration<String> enm = form.keys(); enm.hasMoreElements(); ) {
        var key = enm.nextElement();
        body.add(key, form.get(key));
    }

    // Create url and request
    String url = urlBuilder.build().toString();
    var request = new Request.Builder().url(url).post(body.build()).build();

    // Execute the request
    Call call = new OkHttpClient().newCall(request);

    // Return the response of the request
    return call.execute();
}
```

Figure 8.11: The Java code used for posting data to the backend.

```java
public static Response get_request(String sub_dir, Dictionary<String, String> parameters) throws IOException {
    HttpUrl.Builder urlBuilder = Objects.requireNonNull(HttpUrl.parse( $this$parse: base_url + sub_dir)).newBuilder();

    // Iterate over the parameters and add them as query parameters
    for (Enumeration<String> enm = parameters.keys(); enm.hasMoreElements(); ) {
        var key = enm.nextElement();
        urlBuilder.addQueryParameter(key, parameters.get(key));
    }

    // Create url and request
    String url = urlBuilder.build().toString();
    var request = new Request.Builder().url(url).get().build();

    // Execute the request
    Call call = new OkHttpClient().newCall(request);

    return call.execute();
}
```

Figure 8.12: The Java code used for fetching data from the backend.

Unit Tests - Code Extracts

```
class ValidationTest {

    no usages  ThomasCanning
    @Test
    void calculateTimeDifference() {
        assertEquals( expected: "00:00", Validation.calculateTimeDifference("12:00", "12:00"));
        assertEquals( expected: "00:01",Validation.calculateTimeDifference("12:00", "12:01"));
        assertEquals( expected: "11:02",Validation.calculateTimeDifference("01:00", "12:02"));
        assertEquals( expected: "23:23",Validation.calculateTimeDifference("00:00", "23:23"));
        assertEquals( expected: "00:00",Validation.calculateTimeDifference("25:00", "25:30"));
        assertEquals( expected: "00:00",Validation.calculateTimeDifference("0:10", "1:5"));
        assertEquals( expected: "00:00",Validation.calculateTimeDifference("04:101", "04:102"));
        assertEquals( expected: "00:00",Validation.calculateTimeDifference("abc", "abc"));
        assertEquals( expected: "00:00",Validation.calculateTimeDifference("1", "1"));

    }

    no usages  ThomasCanning
    @Test
    void validateTimeInput() {
        assertTrue(Validation.validateTimeInput( userTimeInput: "12:00"));
        assertTrue(Validation.validateTimeInput( userTimeInput: "00:00"));
        assertTrue(Validation.validateTimeInput( userTimeInput: "23:59"));
        assertFalse(Validation.validateTimeInput( userTimeInput: "12:60"));
        assertFalse(Validation.validateTimeInput( userTimeInput: "24:00"));
        assertFalse(Validation.validateTimeInput( userTimeInput: "12:00:00"));
        assertFalse(Validation.validateTimeInput( userTimeInput: "12:00am"));
        assertFalse(Validation.validateTimeInput( userTimeInput: "12:00pm"));
    }
```

Figure 8.13: An example of some of the code to test the functions that dealt with times, using JUnit for testing.

Figure 8.14: An extract from some of the JUnit tests for the password and email validity checks.

Final Class Diagrams

[5]

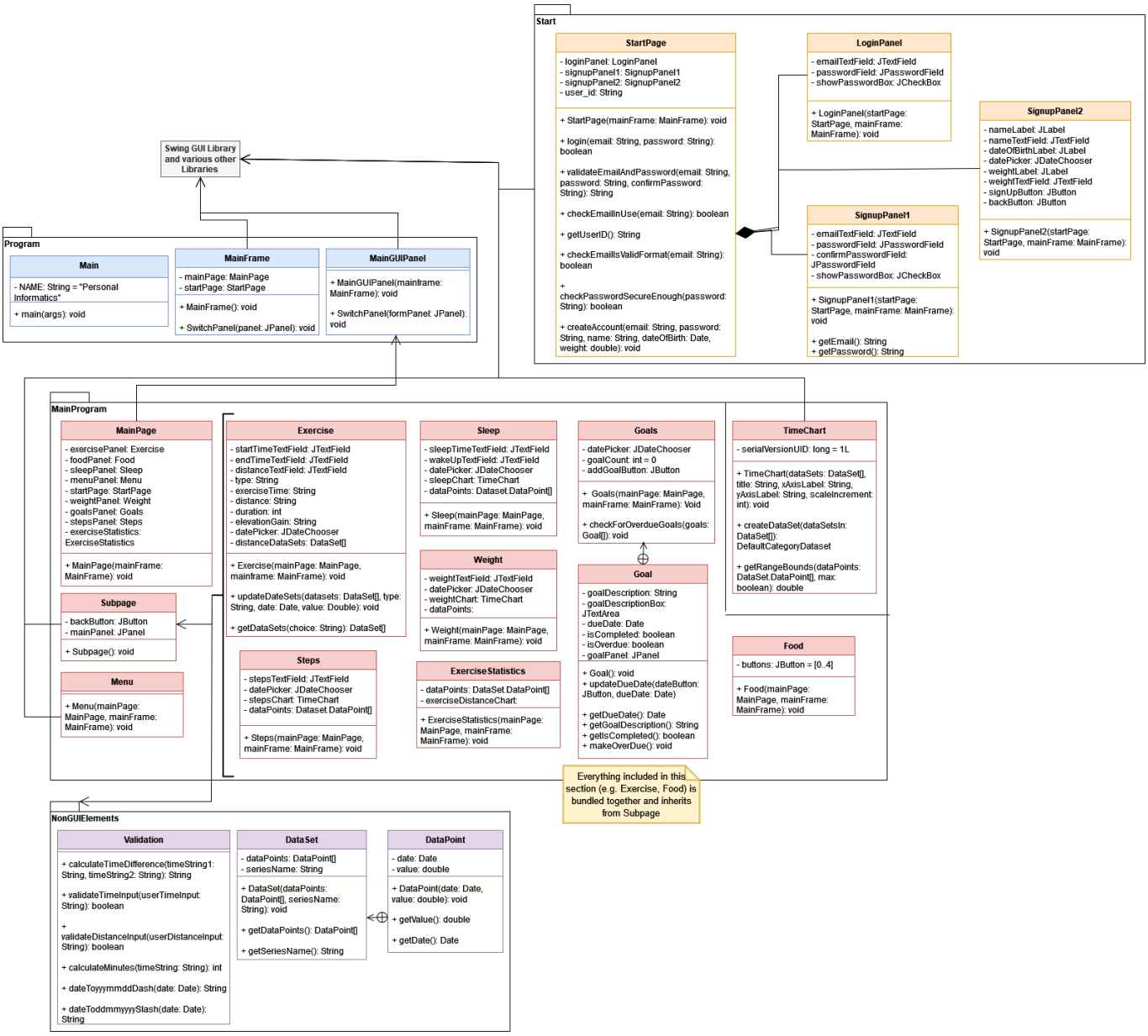


Figure 8.15: Full frontend class diagram.

[6]

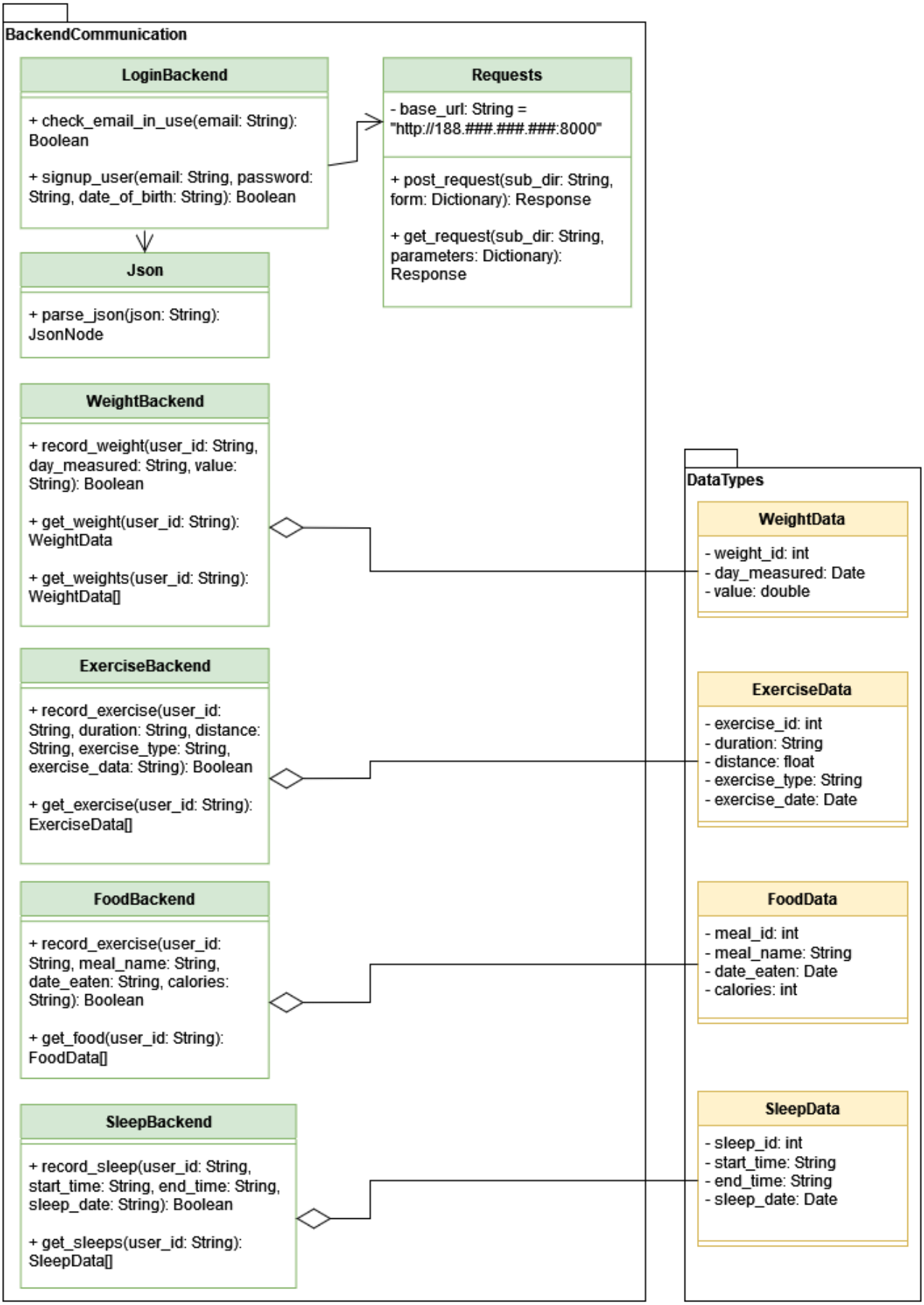


Figure 8.16: Full backend class diagram.

Bibliography

Abrahamsson, P., Salo, O., Ronkainen, J., and Warsta, J., 2017. *Agile software development methods: review and analysis*. arXiv: 1709.08439 [cs.SE].

Brabazon, T., 2015. Digital fitness: Self-monitored fitness and the commodification of movement. 48.

Carlhenricsdotter, E., 2022. The future design of personal informatics.

Fritz, T., Huang, E.M., Murphy, G.C., and Zimmermann, T., 2014. Persuasive technology in the real world: a study of long-term use of activity sensing devices for fitness. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* [Online]. Toronto Ontario Canada: ACM, pp.487–496. Available from: https://doi.org/10.1145/2556288.2557383 [Accessed April 29, 2023].

Klasnja, P., Consolvo, S., McDonald, D.W., Landay, J.A., and Pratt, W., 2018. Using Mobile & Personal Sensing Technologies to Support Health Behavior Change in Everyday Life: Lessons Learned, pp.338–342.

Li, I., Dey, A., and Forlizzi, J., 2010. A Stage-Based Model of Personal Informatics Systems, pp.557–566.

Li, I., Medynskiy, Y., Froehlich, J., and Larsen, J.E., 2012. Personal Informatics in Practice: Improving Quality of Life Through Data.

Niess, J. and Diefenbach, S., 2018. Practical Challenges for the Design of Personal Informatics Systems. *I-com* [Online], 17(3). Available from: https://doi.org/10.1515/icom-2018-0036 [Accessed April 30, 2023].