Custom Personal Informatics System Report

Programming 2 (CM12005)
Group 13

Akim Komarnitskii
(ak3625, BSc CS Y1)

Jeet Meher
(jm3522, MComp CS Y1)

Ollie Vahid
(ov247, BSc CS Y1)

Alex Herrera
(ah3208, MComp CS Y1)

Kateryna Dzemliuk
(kd794, BSc CS Y1)

Tom Boyce
(tb2301, BSc CS Y1)

James Sheppard
(js4209, MComp CS Y1)

Martin Darwall
(md2353, BSc CS Y1)

Chien Nguyen
(ctn32, BSc CS Y1)

Abstract: This document outlines the development of a Custom Personal Informatics (PI) System designed to aid university students in managing and improving their personal health and productivity. The need for such a system is derived from the growing understanding that simple changes in daily routines can significantly impact an individual's overall well-being and effectiveness. The system's development followed the Agile methodology, with features incrementally added and refined over five completed sprints. Our PI system is designed to track critical metrics such as sleep, water intake, steps, and productive hours. Each of these metrics was chosen based on extensive initial research including surveys and interviews with the target demographic, ensuring the system's relevance and utility. As of the third sprint, the system successfully integrates manual data entry with automated data collection via the Fitbit API, offering users a comprehensive overview of their habits. It also provides motivational tools like goal setting and achievement tracking to encourage continual use and engagement. Future work on the system will focus on enhancing user interaction by refining the UI for greater intuitiveness and adding features that allow users to compare their performance with peers, fostering a competitive yet supportive environment. By continuously evolving the system based on user feedback and emerging research, we aim to increase the positive impact of the system on users' lives.

Contents

1 Introduction **1**

2 Agile Software Process Planning and Management **2**

3 Specification of Software Requirements **3**
3.1 Gathering System Requirements 4
3.2 Specific Domain of Application 4
3.3 Data Fields Chosen 4
3.4 Motivating Users 5
3.5 Comparing User Data 6
3.6 Viewing and Collecting Tracking Data 6
3.7 Scalability and Performance 6
3.8 Structuring the Software Development Process 6
3.9 Testing 6
3.10 Requirements Table 7

4 Design **7**
4.1 Database ERD Model 7
4.2 UML Class Diagram 9
4.3 UML Flow Diagram 11
4.4 User Interface 12

5 Software Testing and Verification **16**
5.1 Manual Testing 16
5.2 Unit Testing 17

6 Reflection and Conclusion **18**
6.1 System Development 18
6.2 Software Process 20
6.3 Conclusion 20

7 References **21**

8 Appendix: Group Contribution Form **22**

9 Appendix: Survey **23**

10 Appendix: Interview Transcripts **24**

11 Appendix: Testing **26**
11.1 Testing Plan 26
11.2 Manual Testing 27
11.3 Unit Testing 28

12 Appendix: Contribution Table **32**

13 Appendix: Minutes **34**
13.1 Minutes From 25/03 . . . 34
13.1.1 Attendance . . . 34
13.1.2 Meeting Notes . . . 34
13.1.3 First Sprint (25/03 — 01/04) . . . 34
13.1.4 Next Week Discussion Points . . . 35
13.2 Minutes From 01/04 . . . 35
13.2.1 Attendance . . . 35
13.2.2 Discussion Topics . . . 35
13.2.3 Second Sprint (01/04 — 08/04) . . . 36
13.3 Minutes From 08/04 . . . 37
13.3.1 Attendance . . . 37
13.3.2 Discussion Topics . . . 37
13.3.3 Third Sprint (08/04 — 15/04) . . . 37
13.4 Minutes From 15/04 . . . 38
13.4.1 Attendance . . . 38
13.4.2 Discussion Topics . . . 39
13.4.3 Fourth Sprint (15/04 — 22/04) . . . 39
13.5 Minutes From 19/04 . . . 40
13.5.1 Attendance . . . 40
13.5.2 Discussion Topics . . . 40
13.6 Minutes From 22/04 . . . 41
13.6.1 Attendance . . . 41
13.6.2 Progress Review and Final Sprint (22/04 — 03/05) . . . 41
13.7 Minutes From 29/04 . . . 42
13.7.1 Attendance . . . 42
13.7.2 Progress Review and Report Finalization . . . 42

1 Introduction

Increasingly, medical advice and research is showing that minor changes to an individual's daily routines can have positive effects on their overall health, wellbeing and productivity. For example, we are often reminded of the advantages of walking 10,000 steps (Tudor-Locke et al., 2011), sleeping for at least 7 hours (Watson et al., 2015) and drinking at least 6 cups of water (NHS, 2023) every day. However, as discussed by Madore and Wagner (2019), processing multiple tasks concurrently can lead to reduced efficiency. Therefore, tools to aid in the tracking and pursuing of these personal development goals are likely to help an individual reach their goals more effectively.

PI(Personal Informatics) software is software designed to help collect and analyse data about an individual with the aim of promoting self-understanding and betterment (Li, I., 2012)[1]. A PI system could be an ideal solution to this issue as it could be very useful in allowing an individual to monitor and streamline their progress towards their goals for self-improvement.

However, in designing a PI system it is important to ensure user engagement; if a user stops using a system then the system cannot help them. Kersten-van Dijk et al. (2017) found that the studies they reviewed which discussed users dropping out of using a PI system reported dropout rates of 7-44%. This suggests that a significant proportion of people might not feel motivated to continue using PI software. Furthermore, Jones and Kelly (2018) found that presenting users with too much information could leave them feeling overwhelmed; Rapp and Cena (2016) found that first-time users of PI systems could find the act of recording their data burdensome; and Potapov et al. (2021) discovered that teenagers often found PI systems either controlling or confusing depending on the way they were designed.

Fortunately, some of these studies, as well as many others, have investigated how to make a PI software system effective for its users and proposed ways of designing systems with this in mind. For instance, Jones and Kelly (2018) suggested that, to avoid overwhelming users, a system should only show the user information that is interesting to them. They found that 'interesting information' was that which was surprising, useful or statistically significant. It was also found that users found it more interesting when insights could be provided between aspects of their life, rather than within them. Rapp and Cena (2016) recommended providing users with tailored summaries and targets; control over their data; and reviews of past data. In this way they suggested that feelings of freedom and nostalgia might be fostered, increasing positive views and connections with the software. Potapov et al. emphasise the importance of finding an appropriate balance between constraints to orient the user towards positive goals and freedom to allow them to use the tool in a way that suits them. Additionally, a study by Loerakker et al. (2023) showed that promoting self-compassion through the design of a Personal Informatics system can foster positive self-reflection while reducing the chances of rumination and cessation of goal pursuit. The study showed that framing data positively can promote self-compassion. For example, highlighting positive achievements rather than criticising poorer performance.

Our intention is to create a PI system aimed at current university students. To decide on a list of metrics for our system to track, we investigated the interests of our chosen demographic by means of a Google Forms survey (see Appendix 9). The survey, which was sent to students online, revealed that 60% of respondents were most interested in tracking their health, followed by 30% who were most interested in tracking their own learning and development. In addition, when asked to rate their interest in tracking their data for self-improvement on a scale of 1-10, 55% of participants rated their interest at 8 or above. This reinforced our belief that a PI system would be popular among our target audience.

[1] Article from the specification.

Furthermore, we conducted some individual interviews of students involving more specific questions about health and desirable PI system features (see Appendix 10). These interviews revealed an interest in graphs and a simple user interface which we will take into account when designing our application. All the students we interviewed also expressed a desire to alter the number of hours of sleep they got on a daily basis. Based on these investigations, as well as research into Fitbit and Garmin software, we decided that our system would track hours of sleep, water intake, steps taken and hours spent productively.

Our system will allow data to be entered manually. Some metrics will also be able to collect data automatically via the Fitbit API to reduce the burden of data entry. Additionally, our software will allow users to set personal goals and see useful data about their performance. This data will be able to showcase correlation in each metric against time individually and between separate metrics. This functionality will all be accessed via a desktop application with a graphical user interface. We hope that, by tracking these four important aspects of student life, our system will provide interesting and useful insights that guide our users towards better health and productivity while ensuring that they are not overwhelmed by too much data or the need to balance their goals.

2 Agile Software Process Planning and Management

To carry out this project in an efficient and organised manner, we followed Agile practices and values using the Scrum framework. This involved using a series of week-long sprints to break up and work through the continuously evolving product backlog. Between each sprint, a meeting was held to discuss the outcome of the previous sprint, go over any changes to the product backlog, produce a sprint backlog and divide this sprint backlog between the group members for completion during the next sprint. To maintain clarity and access to information, minutes were taken of these meetings and stored in the project GitHub repository. This meant that developers could always access information about the progress of other developers and their allocated tasks (See Appendix 13). During sprints, group members worked either individually or as part of a smaller group to complete their allocated element of the sprint backlog. Regular scrum meetings were also used to allow group members to communicate progress, plans, obstacles and key information. We assigned a scrum master to gain a full understanding of Agile values and methodologies and ensure the group's adherence to them. A product owner was also chosen to be in charge of managing the product backlog and organising sprints.

Our first sprint was devoted towards research into PI systems and what students wanted in one. Three group members were assigned to researching PI systems, two to market research, two to gauging interest in students, one to compiling the initial project backlog and one to researching the most appropriate programming language to use for the project. By the end of the week, all members had fulfilled the duties assigned to them. The research was discussed as a group and used to produce the idea for our system. As this idea evolved, it was converted into a list of functional requirements and added to the product backlog.

The second sprint consisted of designing and beginning to create our application. A member of the team set up a framework for our project on GitHub and two team members were assigned the task of creating a UI (User Interface) diagram for our application. Having done this, the UI designers presented the result in a scrum meeting and, based on this, other members of the group created a coding plan, started creating the UI and implemented some basic functionality. Meanwhile, a group member worked on creating a database and demonstrating the use of it; another researched and showcased the use of a suitable API; and the remaining people began the writing of the report. The scrum meetings held throughout the week were key in ensuring that all the programmers could stay abreast of any important changes made by others with impacts on their own work, as well as allowing the sharing and discussion of issues.

At the end of the sprint, all available team members attended a Google Meet meeting in which all progress was discussed. The sprint was very successful, with every individual achieving their designated goals. Based on the API research conducted, it was decided that the Fitbit API was most suitable for our system. Additionally, it was concluded that an alternative database design would be more efficient. These evolutions of the idea of our project meant that the product backlog was updated.

The sprint that followed largely prioritised the implementation of desired functionalities from the product backlog. Five members of the group were individually allocated to specific aspects of the program. This included implementing the productivity tracker, achievements, goals, graphs and the Fitbit API integration. One developer was also assigned to redesigning the database to be more efficient. The other available group members were given sections of the report to work on. During the frequent scrum meetings in this sprint, reporting back helped maintain a full understanding of the latest changes among the programmers. Unfortunately, due to technical difficulties, the sprint backlog was not entirely completed. These incomplete tasks were added to the sprint backlog of the next sprint. Other than some software features, all tasks were completed successfully and the next sprint was discussed and planned as usual.

The fourth sprint had very similar aims to the previous one, with team members allocated to report writing and the development of features, including those which were not completed during the previous week. By the end of the sprint, more elements of our system's functionality had been implemented and the report completed further so these parts of the product backlog were marked as complete. However, the design section of the report and the achievement functionality of the system were not completed due to other commitments and the work tracking section was not fully functional due to technical difficulties. The completion of these requirements was moved to the following week's sprint.

The fifth, and final, sprint targeted the completion of as yet unimplemented features and work on the report document. These responsibilities were split so that roughly half the group was working on each. As well as the programming of parts of the system, focus was also placed on merging all the functionality together to create our system as a whole. To align the completion of the project with our deadline and allow enough time for the required work, this final sprint took place over 11 days, between the 22nd of April and the 3rd of May. By the end of the sprint, the entirety of the sprint and product backlogs had been completed and the project as a whole was finished.

By following the Scrum framework we were able to efficiently divide our work into manageable quantities and maintain constant progress towards a timely completion, clearly demonstrating the effectiveness of the methodology.

3 Specification of Software Requirements

This section establishes the core requirements for our Personal Informatics system, focusing on helping students keep up with their fitness and study goals. We utilised feedback from surveys (see Appendix 9), talks with users, market research, and academic research articles to figure out what features our system should have. Our aim is to make a system that assists students in improving both their physical health and studying habits whilst not being overly invasive.

In the following sections, the notation *SRQ* will be used to denote requirements from the coursework's specification, while *ARQ* will represent Additional Requirements. Refer to Section 3.10 for a quick-reference table of requirements.

3.1 Gathering System Requirements

Initially, we asked people what features they would look for in an personal improvement application. From these interviews (see Appendices 9 and 10), we identified that most students use PI systems for tracking their fitness (60%) and their learning progress (30%), with the other 10% being interested in their environmental impact. This helped us decide which area of personal informatics our system should focus on. Next, we looked at what users stated about our initial ideas as well as comparing our ideas to expert findings from our listed articles (SRQ 3.1, 3.2). Additionally, many students told us they prefer applications that are intuitive but still allow for great insight and control into goals and targets. Also, they wished to connect with their friends and peers through the app and liked the idea of seeing all their information in a single area. Furthermore, we also researched into competitor systems (such as Garmin and Fitbit) to identify both good and bad features of these systems.

After this, we made an initial requirement list of what functionality our app must contain (SRQ 2.1, 2.3) based on the data gathered. We must ensure our system is perfect for tracking health and study time, allows users to share their progress with friends, and keeps all their data safe and private. Furthermore, we must regularly reflect on our requirements to ensure that they properly suit our vision for the system, adapting them whenever tests, feedback, or other factors suggest so.

3.2 Specific Domain of Application

Our PI system is targeted towards the student demographic, addressing their unique challenges such as managing academic deadlines alongside maintaining a balanced lifestyle.

Students grapple with the dual challenges of academic accountability and physical well-being. The system shall provide a selection of tools for effective study habits and simple health tracking (SRQ 2.1).

3.3 Data Fields Chosen

Our app is tailored towards individuals who require assistance managing their time healthily as we are aware that a large proportion of students have difficulty maintaining a healthy work-life balance. Our app aims to help them track key aspects of the student lifestyle. The fields we have chosen to track as well as how they shall be accessed are as follows (SRQ 5.1, 5.3):

- **Tracking sleep time**: The average student struggles to manage sleep, studying, and being social and out of these three, sleep is usually the area in which students choose to disregard in order to create more time. However, a lack of sleep can be detrimental on all aspects of life and as such it is imperative that we ensure our users are aware of how little sleep they may be getting across a given time frame. We are aware that some students may also be getting too much sleep which can lead to them feeling lethargic throughout the day and as such our system should also provide a solution for these users. Sleep quality also plays a large role in this but is incredibly difficult to track unless specialist equipment is used. Additionally, it is usually influenced by factors our system cannot control such as noise from other tenants and temperature. This field will require a user to manually enter their data. Allowing a user to set goals for this field will enable them to prioritise the quantity of sleep they are getting each night. Our system should enable our users to obtain a consistent, healthy sleep pattern.

- **Tracking step count**: The average student spends a large proportion of their day sedentary, whether that be from napping after a lecture, studying hard for an exam, or sat behind a computer trying to finish coursework. This can cause students to not get out frequently for events other than their lectures and the occasional social gathering. This has hidden consequences: less time outside walking means less vitamin D, less opportunity to make new friends, increased risk of certain diseases such as heart disease. It is imperative to emphasise to our users who do not perform enough steps in a given time period that this behaviour is not healthy. Our system will allow a user to view trends in their step data and clearly visualise how little (or many) steps they are actually performing.

- **Tracking work done**: Students often report one of two extremes when it comes to studying: overstudying and losing out on other aspects of the student life or understudying and suffering academically. Neither of these are healthy for a student. To potentially fix this behaviour, our system should allow a user to set study goals. This allows the understudying students to have a target to achieve in a given time period, motivating them to allocate more time to study. Furthermore, this allows the overstudying students to know that they have studied a sufficient amount for a given time period and deserve a break.

- **Tracking water intake**: Water intake has many hidden benefits: helps with weight loss by making you feel more full, increases energy levels, improves skin health. Many students may neglect drinking water as they need an energy boost via caffeinated products. By setting clear goals of how much water a user should drink, it increases the likelihood they select to drink water over a less healthy alternative.

- **Achievements and Goals**: These will be the methods by which goals for a user to achieve are set. Their main purpose is to motivate a user to change some aspect of their life. Through our research, we discovered that by providing some sort of challenge, a user becomes more motivated to complete a goal, even if no physical reward (aside from health benefits) is provided. By allowing the user to set these goals we ensure that the user finds these achievements to be possible and further increases the likelihood they change some of their bad habits. Additionally, by having system set achievements, a user may be driven to greater heights than they think they can achieve.

3.4 Motivating Users

Our system will allow users to set and manage goals (SRQ 7.1, 7.2, 7.3). This can be whatever a user feels is appropriate for any of the given fields including the time frame in which they wish to complete the goals. In addition to this, our system will implement achievements. This will be separate from goals and will be set by our system. For example, a goal for step count could be "walk 10,000 steps today". Achievements are key to our system as they provide a reason for a user to improve specific fields without aimlessly trying to increase some number (SRQ 7.4). Furthermore, these achievements rotate daily, ensuring a user never runs our of goals to achieve.

Additionally, we set additional requirements for our system to contain a range of social features (ARQ 1) including a leader board and groups. This was requested by many of the students we interviewed who wished to use friendly competition as a motivator (see Appendix 10). Unfortunately, we set the priority for this requirement to low as it did not directly influence the core behaviour of our system. During the creation of our system, it was decided that this feature was too complex to implement given the time limit and unexpected workload of our higher priority requirements.

3.5 Comparing User Data

Our system must ensure a user can easily view and compare all key data fields (SRQ 5.2). This should be done through graphs displaying all fields a user selects on the same axis. This graph should dynamically change size to fit the upper bound of the data for a given user, allowing a user to easily identify any correlation between the data fields (SRQ 6.2).

The graph must allow for a user to specify a time frame (SRQ 6.2) for the data to allow such that it allows the user to identify potential outside factors that our system cannot consider. For example, an event in a given time frame such as a coursework deadline could influence sleep count.

3.6 Viewing and Collecting Tracking Data

Users will be given the option to synchronise their step data, both present and past from a Fitbit account via the Fitbit API(ARQ 2). We must ensure that potential users without the relevant hardware can still use our system and must be done in an intuitive way in order to allow our system to be accessible for a greater amount of students. This will also allow for easy gathering of data fields which cannot be tracked via physical devices (SRQ 5.1). As only the study time field can be accessed directly via the Fitbit API, manual entry is necessary (SRQ 5.3).

To ensure that our users understand what data we are storing, each user shall be allowed to view the data we have stored on them for each of these key fields(SRQ 5.2). In addition, the security of this data is paramount. We must ensure it cannot be accessed by any other end users.

3.7 Scalability and Performance

Scalability (ARQ 3) will not be a problem for our system as it will run entirely off of the user's computer. Additionally, in order to allow for completion of our additional requirements of adding multi-user functionality, our system must allow for easy scalability. This should be done through a well constructed database. We must ensure that this system performs smoothly and efficiently to not hinder a user's experience (ARQ 4).

3.8 Structuring the Software Development Process

Our development of this project must very closely follow the agile and SCRUM development approach. This will be done to ensure our system is consistently improving and changing to match what our users need. Refer to the Agile Software Process Planning and Management section for more details (covers all of SRQ 1).

3.9 Testing

We must extensively test each code version to ensure that our users will not run into any bugs or other issues whilst using our system. This shall be driven largely by simulating the experience an average user will have. Refer to Section 5 for more details (covers all of SRQ 4).

3.10 Requirements Table

Requirement (RQ)	Notation	Dependencies	Priority
Follow SCRUM ideology	SRQ 1.1	—	High
Include at least one sprint	SRQ 1.2	1.1, 1.3	Medium
Sprints must last 1-3 weeks	SRQ 1.3	1.1, 1.2	Medium
Review requirements regularly	SRQ 1.4	—	High
Expand on initial requirements	SRQ 2.1	3.2	High
Additional functionality for our features	SRQ 2.2	—	High
Establish additional requirements	SRQ 2.3	2.1, 2.2	High
Cite 3 articles on target field	SRQ 3.1	—	High
Read and cite 6 articles	SRQ 3.2	—	Medium
Testing driven approach	SRQ 4.1	—	High
Provide testing evidence	SRQ 4.2	—	High
Store relevant user data	SRQ 5.1	3.1, 3.2	High
Allow users to access data	SRQ 5.2	—	High
Allow for manual data entry	SRQ 5.3	—	Medium
Compare user data	SRQ 6.1	5.1	High
Allow comparison of data	SRQ 6.2	3.1, 3.2, 5.1	High
Allow for managing of goals	SRQ 7.1	5.1	High
Add a time aspect to goals	SRQ 7.2	5.1, 7.1	Medium
Allow for updating of a goal	SRQ 7.3	7.1	Low
Motivating features	SRQ 7.4	7.1	Medium
Social features	ARQ 1	2.1	High
Access data Via API	ARQ 2	2.1, 5.1	Medium
Create a scalable system	ARQ 3	2.1	Medium
System must run smoothly	ARQ 4	2.1	Low
Easy data input	ARQ 5	2.1, 5.1, 5.3	High

Table 1: Priorities and Dependencies

4 Design

The following section shows how we plan to turn our initial requirements into a complete program where all of our features are included. Furthermore, it contains many important diagrams and images detailing the specifics of each feature in our system.

4.1 Database ERD Model

Our Entity-Relationship Diagram (ERD) demonstrates the structure of our PI system's database. This ERD has been created to suit our system's needs and to be used as a guide during development to ensure database functionality.

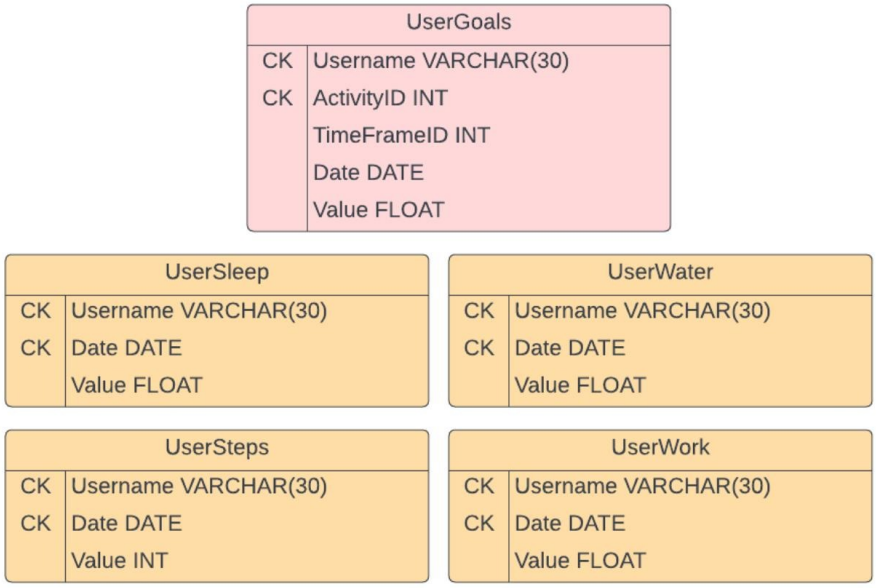


Figure 1: Database ERD Model

Figure 1 describes the following components:

- `UserStep` Table: This table records all step data that the system is provided. By using a compound key of the `Username` and `Date` fields each user's step count can be stored for a given day, with the count being stored in the `Value` field. By taking these daily inputs, our system can obtain enough data to assist in the improvement of the habits of a given user and help them set realistic goals. As the data can also be inserted through an API, this table aims to cover ARQ 2.
- `UserSleep` Table: This table is used to record all sleep data that the system is provided. By using a compound key of the `Username` and `Date` fields each user's sleep time can be stored for a given day, with the total time being stored in the `Value` field. By taking these daily inputs, our system can obtain enough data to assist in the improvement of the habits of a given user and help them identify when their sleeping time is either too low or high.
- `UserWork` Table: This table records all entries of both working and studying a given user performs. By using a compound key of the `Username` and `Date` fields each user's daily work time can be stored. The `Value` field holds the time data for each entry. By collecting large amounts of this data over an extended period of time, our system can support a user in identifying when they should be using their time more effectively.
- `UserWater` Table: All entries for the Water field are stored here. By using a compund key of the `Username` and `Date` fields each user's water intake can be stored and monitored, with the amount of water intake being stored in the `Value` field.

- `UserGoals` Table: The UserGoals table holds data related to the goals set by users. With attributes of `Username`, `ActivityID` forming a compound key, as well as `TimeFrameID`, `Date`, and `Value` storing the data that makes up each goal, this table supports the setting and monitoring of personalised goals. `UserGoals` covers SRQ 7.1, 7.2 and 7.3.

`UserStep`, `UserSleep`, `UserWork` and `UserWater` are all aimed to satisfy ARQ 5 and SRQ 5.1, 5.2 and 5.3.

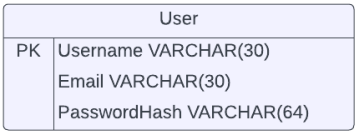


Figure 2: User Class in the Database Model

Figure 2 describes our `User` Table which sits at the centre of our database. The User table stores personal information of the user to allow us to differentiate between users. Attributes include `Username`, `Email`, and `PasswordHash`, which are fundamental for ensuring secure user authentication and system access. This table acts as the primary table with which other tables associate, establishing one-to-many relationships. Each user can associate with multiple records in the linked tables, reflecting the different amount of activities and metrics tracked by the system. Ultimately, the requirement of a multi-user system was removed due to limitations of time and other unexpected issues with development. However, the current database architecture has been deliberately designed so that a multi-user interface could be easily integrated , should it be developed in the future. This table covers ARQ 1.

4.2 UML Class Diagram

The class diagram of our PI system (Figure 3) shows the fundamental components of our system and their relationships, highlighting the system's design to efficiently manage and process user data. Central to this architecture is the `Database` class, which acts as the core for all data manipulation and data queries within the system. This class is critical for the handling of data and supports various functionalities essential to user interaction and data integrity. We utilised the Dapper module in C#, which allows the direct return or insertion of a list of C# objects from an SQL query, rather than handling them as tuples. Accordingly, based on the ERD, all classes are designed to be easily integrated with the Dapper package.

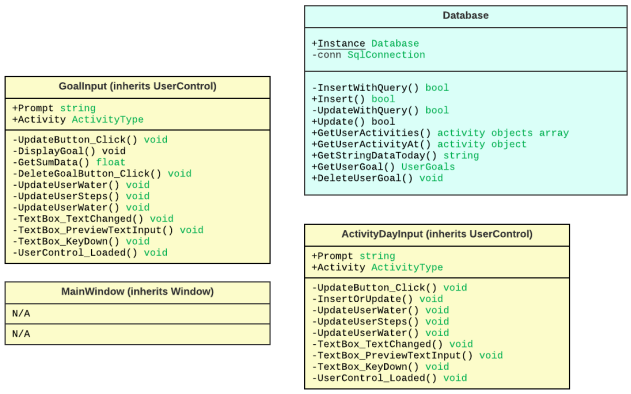


Figure 3: UML Class Diagram

The `Database` class ensures a singleton pattern for database management, creating a single, globally accessible instance throughout the application lifecycle. It manages the database connectivity, allowing secure and persistent connections to the data store. Data manipulation methods such as `InsertWithQuery()`, `Insert()`, `UpdateWithQuery()`, and `Update()` allow for efficient data insertion and updating within the database. Retrieval methods like `GetUserActivities()` and `GetUserActivityAt()` fetch activity data for users, crucial for generating reports and insights. Utility methods including `GetStringDataToday()`, `GetUserGoal()`, and `DeleteUserGoal()` provide additional functionality for managing daily data strings and user goals, enhancing the system's responsiveness to user interactions. These methods were essential for SRQ 6.1 and 6.2, as well as directly adhering to SRQ 5.1 and 5.2.

The `ActivityDayInput` and `GoalInput classes`, are specifically designed for handling the input and update functionalities related to daily user activities and goal management, respectively. The `ActivityDayInput` Class includes a Prompt string for user guidance and an Activity `ActivityType` to specify the type of activity data being entered. It comprises various user interaction handlers such as `UpdateButton_Click()` and `InsertOrUpdate()` for data submission, alongside `UpdateUserWater()` and `UpdateUserSteps()` for updating specific activity metrics. Text handling methods like `TextBox_TextChanged()`, `TextBox_KeyDown()` and `TextBox_PreviewTextInput()` ensure data integrity and user input validation.

The GoalInput Class mirrors the ActivityDayInput class in providing prompts and activity type specifications tailored towards user goals. It features goal-specific functionalities such as `DisplayGoal()` for visualising set goals, `DeleteGoalButton_Click()` for goal modification, and `GetSumData()` for data aggregation. It inherits text handling and initialisation methods from ActivityDayInput, ensuring a consistent and user-friendly interface. Whilst ActivityDayInput stands mainly by SRQ 6.1 and 5.1, GoalInput follows SRQ 7.1, 7.2 and 7.3.

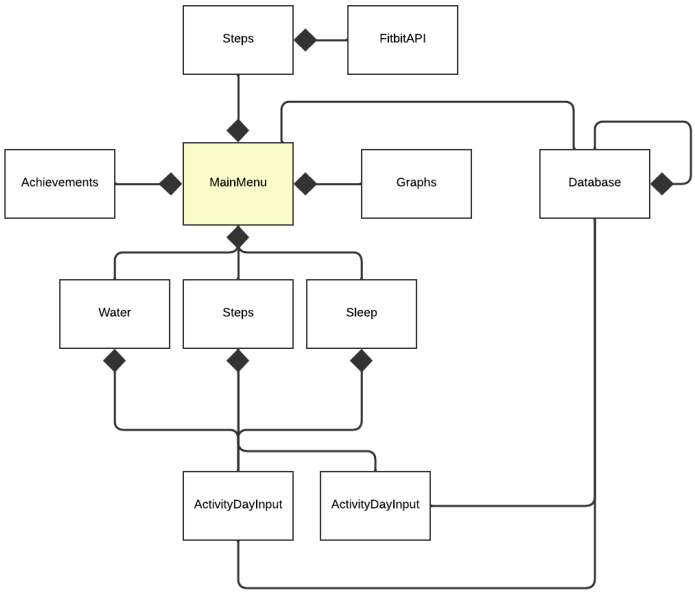


Figure 4: UML Flow Diagram

4.3 UML Flow Diagram

The flow diagram of our PI system (Figure 4) illustrates the interactions between classes, essential for efficient functionality and user experience. At the core is the MainMenu class, which integrates features like tracking steps, achievements, water intake, and sleep via separate classes (Step, Achievement, Water, Sleep). These classes manage specific activities and are tightly integrated with `ActivityDayInput` and `GoalInput` controls for data entry and goal management.

For example, when users input their daily water intake through the `ActivityDayInput` tailored for water, this control processes and validates the data, then interacts with the Water class and the Database to store this information. Similarly, the `GoalInput` facilitates setting and updating goals, interacting directly with the Database to ensure goals are current and reflective of user activity.

The association between the `ActivityDayInput`, `GoalInput`, and the `Database` ensures immediate data reflection in the system, supporting features like activity comparison and trend analysis. The `Database` manages its instance, maintaining data integrity and security, and acts as the core for all user data. This structure ensures seamless data flow across the system, fulfilling functional and non-functional requirements, and enhancing overall user interaction. The association between these elements are the groundwork for all requirements SRQ 5, SRQ 6 and SRQ 7, as well as ARQ 4.

4.4 User Interface

This section outlines the design of the user interface (UI) for our PI system, focusing on creating engaging user experience. We aim to bridge the gap between the system's technical capabilities and the user's needs through a simple yet sensible interface. The UI design is designed to facilitate easy navigation and interaction, so that users can efficiently manage and track their personal data. Here, we discuss the guiding principles of our UI design and how they enhance usability of the system.

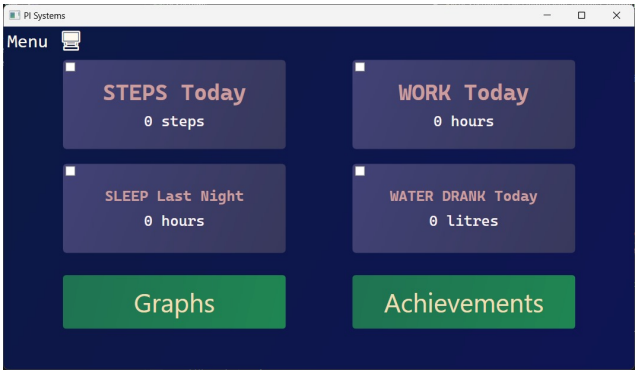


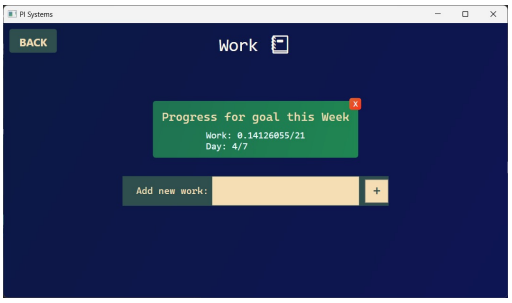
Figure 5: Main Menu of the Application

Figure 5 illustrates the main menu of our application. Positioned in the top left corner of the screen, the label `"Menu"` accompanied by a small computer icon indicates to the user that they are currently viewing the main menu. The background of the entire menu is a dark blue, a soft colour is chosen to minimise eye strain during long use of the app.

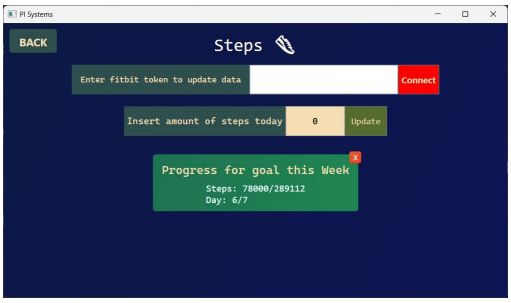
The main menu features four prominent buttons, each corresponding to the core functionalities of the Personal Informatics system: steps, work, sleep, and water. Additionally, two smaller green buttons provide access to sub-functionalities: graphs and achievements. Each of the four larger buttons displays the type of statistic it relates to, with today's data (or the previous night's for sleep hours) shown in smaller font size. The design adheres to SRQ 5.2 and 6.1.

A unique aspect of the design is a small white box located in the top left corner of each main functionalities button. These boxes can be marked by user when they want to see the related data in line graphs, and therefore allows for SRQ 6.2 to function smoothly. Moreover, several boxes can be ticked at once, allowing users to see the relativity of different type of stats in one graph.

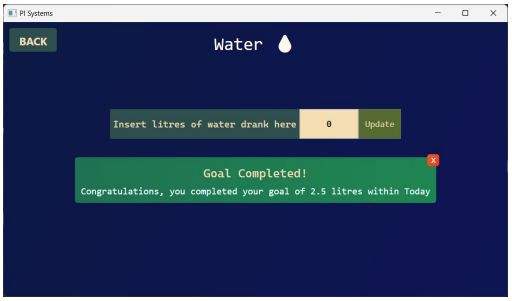
Figure 6 displays the four main functionality screen, which is designed to be simpler and more focused compared to the main menu. In the top left corner, a dark green `"BACK"` button allows users to easily return to the main menu.



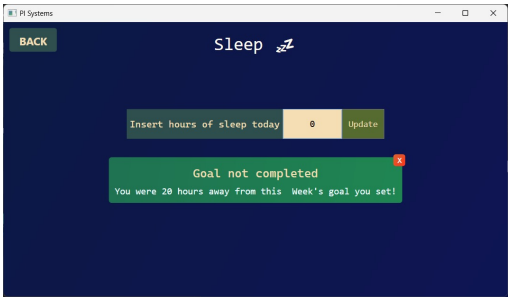
(a) Work Section Interface



(b) Steps Section Interface



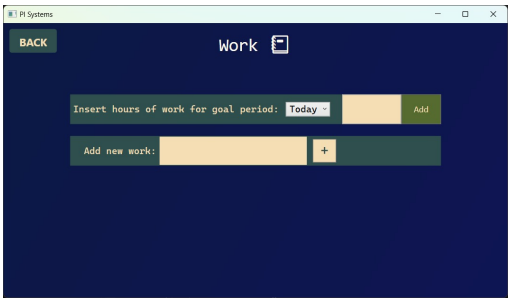
(c) Water Section Interface



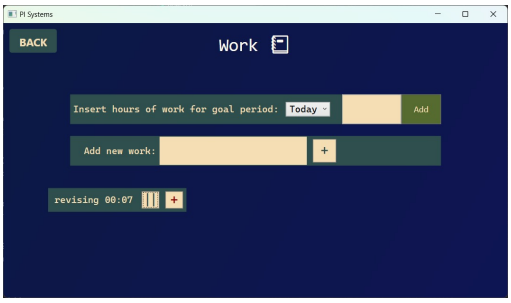
(d) Sleep Section Interface

Figure 6: Main Functionalities Interface

We can clearly see from figure 6a that the heading `"WORK"` is prominently displayed at the top of the screen, accompanied by a book icon. Centered on the screen is a large green box that tracks and displays the user's progress on their weekly achievements. Under the heading `"Progress for goal this Week"` of the box, the interface shows the amount of work completed and the specific day of the week that the goal was set. This box can be closed using the small red `"x"` button located at its top right corner, providing a way to dismiss the details if needed. Directly below this display, there is an input bar where users can `"Add new work"` they have completed that day. Adjacent to this bar, a plus button is the way to confirm the work users inserted and start to time it, allowing users to easily update their daily work achievements. This was designed in this format for ARQ 5.



(a) Add Work bar



(b) Add work successfully

Figure 7: Add Work Interface

Figure 7 illustrates how the functionality Work works. When users dismiss the progress box, the will see a new input bar, which they can use to add the amount of time that they work for the time period they chose in the bar (figure 7a). Another way here is to insert the type of work users want to do in the `"Add new work"` bar and then press the plus button. The work will appear and the time start running, users can pause it whenever they like. After finishing the work, users can press the red plus button to add the time they have been working for to the progress. They can easily check it in the main menu screen.

Figure 6b presents the steps interface, which maintains a similar design with the work screen but introduces several specific modifications. The heading `"Step"` is displayed at the top of the screen, distinguished by a shoe icon next to it, visually represents the category of the section. A notable difference in layout is the placement of the progress box, which is positioned below the input bars rather than above. This adjustment allows for immediate and easy data entry upon accessing the screen, complying with ARQ 5.

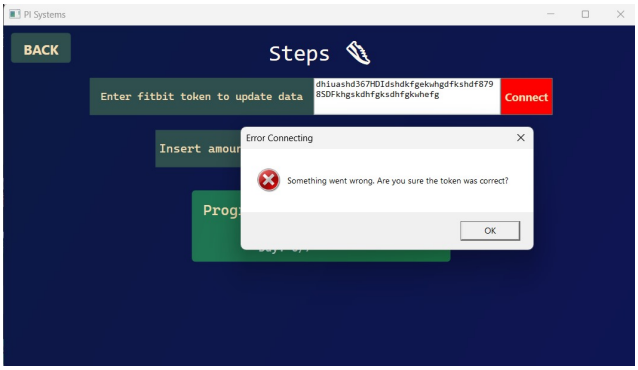


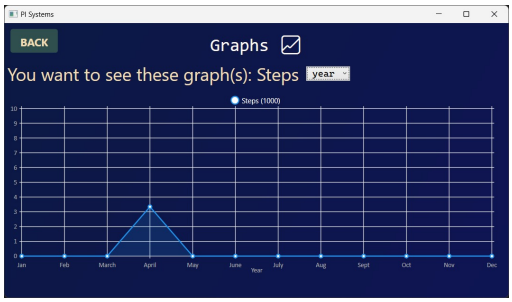
Figure 8: Error inserting wrong API Token format

Additionally, a new input bar for inserting an API token has been added to facilitate the integration of step data from external devices or services and fulfilling ARQ 2. Directly next to the API token input bar are `"Connect"` and `"Update"` buttons. The `"Connect"` button is used for verifying the API token, while the `"Update"` button adjacent to the Step input bar enables users to refresh and submit their step count data. Any incorrect format in the input will trigger an error message, as shown in figure 8, ensuring users are aware of input specifications and can correct their entries accordingly.

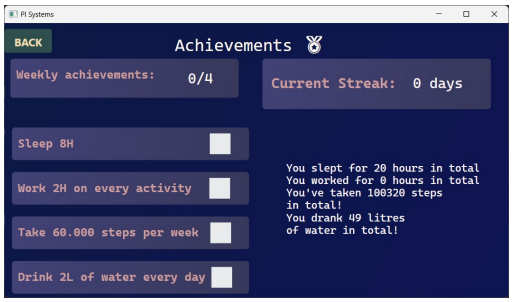
Figures 6c and 6d illustrate the design of the Water and Sleep screens, respectively, which are consistent in layout with the other main functionalities of the app. Both screens feature a `"BACK"` button in the top left corner for easy navigation. The Water screen is identified by a heading labeled `"Water"` accompanied by a water drop icon, and the Sleep screen is marked by a heading labeled `"Sleep"` along with a `"zzz"` icon. Each screen includes an input bar for users to enter their respective data values easily.

In Figure 6c, showcasing the Water functionality, the progress box is displayed upon achieving a set goal. It prominently features the message `"Goal Completed!"` in bold golden text, accompanied by a congratulatory message that also specifies the goal achieved. This positive reinforcement is designed to motivate users and acknowledge their accomplishments, as SRQ 7.4 demands.

Conversely, Figure 6d details the Sleep functionality when a goal is not met. The progress box in this scenario displays a bold, golden headline `"Goal Not Completed"` at the top. Below this heading, the screen informs users of the shortfall, specifying the amount needed to achieve the goal. In the depicted example, it indicates that the user was 20 hours short of reaching the weekly sleep goal. This feedback is intended to encourage users to adjust their habits to meet their sleep targets in the future.



(a) Graphs Section Interface



(b) Achievements Section Interface

Figure 9: Two other functionalities

Figure 9a showcases the Graph interface of the application, which is designed to provide a visual representation of user data over time. The screen includes a `"BACK"` button located at the top left corner, allowing users to easily navigate back to the main menu. The heading `"Graphs"`, marked with a small graph icon, indicates the functionality of the interface.

The main feature of this interface is a large line graph, titled at the top to specify the type of data being displayed. Users have the flexibility to select the timeline for the graph, ranging from a day to a year, with the current example set to display data over a year. The timeline is plotted on the horizontal axis, while the specific metrics related to the user's activities are displayed along the vertical axis. The graph utilises a light blue line on a gridded white background, enhancing readability and making it easier for users to interpret their data trends. The graph is also capable of showing multiple line at once with different colour. This feature makes it easier for users to compare among different types of statistic (figure 10). Through this interface, SRQ 6.1 and 6.2 are followed.

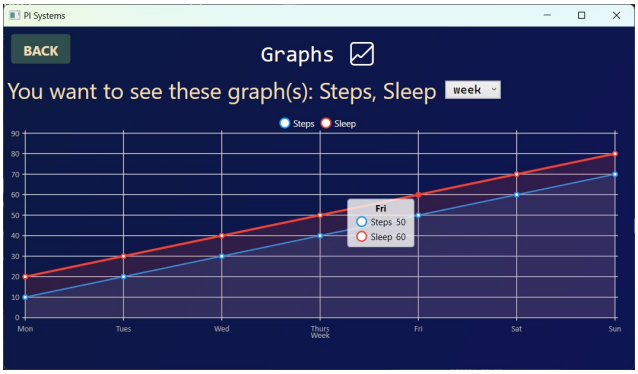


Figure 10: More than one line graph can be shown at a time

Figure 9b presents the Achievements interface, which aims to motivate and inform users about their progress in meeting set goals. Similar to other interfaces, it includes a `"BACK"` button at the top left for returning to the main menu and is labeled with the heading `"Achievements"`, accompanied by a medal icon to symbolise success and recognition.

The interface is divided into two main sections. On the left, a box at the top displays the number of achievements completed within the week, indicated as `"0/4"` in this example, suggesting no goals have been met yet. Below this summary, four boxes represent the achievements tied to the four main functionalities of the system (steps, work, sleep, and water). Each box includes a white checkbox that is ticked off when the corresponding daily goal is achieved.

On the right half of the screen, a white text box provides a summary of the total activities completed for each functionality during the week, assisting users in tracking their weekly progress. Additionally, a streak counter is placed above the activity summaries, encouraging users to consistently meet their daily goals and build long-term habits.

Together, these interfaces support the app's objective to engage users actively in tracking their progress and achieving their health and productivity goals, perfectly complying with SRQ 7.1 and allowing users to compare data with others, complying with ARQ 1.

5 Software Testing and Verification

During the initial planning of this project, the approach to testing was discussed and a testing plan was created, see Figure 13 in Appendix 11. It was agreed that, initially, each time a developer completed a feature from the product backlog, it would be thoroughly white-box tested by the developer using a variety of specifically chosen manual test cases and screenshots would be taken to record the results. If all tests produced the desired results, the developer would consider the feature complete. If any tests were unsuccessful, however, they would debug and fix the problem before applying all tests again. If the error was found in another developer's code during joint work, the error and the test results would be provided to the person responsible and the programmers involved would collaborate on the debugging. In this way, the component's functionality given both valid and invalid inputs would be ensured.

By testing each element before combining it with other elements of the system, the manual testing process as a whole would comprise a bottom-up integration testing approach. On the completion of a section of the product backlog, the developer would send the screenshots produced in testing to the product owner. The product owner would then analyse the screenshots to ensure that the evidenced functionality aligned with the requirements of the product backlog; to provide feedback to the developers; and to inform future decisions and alterations to the product backlog. Additionally, the product owner would manually test the system functionality during sprint planning meetings to provide a form of black-box testing.

Once the system as a whole was developed, the second stage of testing would be implemented, in which unit tests would be designed and used by developers to more thoroughly scour the system for bugs and inconsistencies. These automated and time-efficient tests would effectively provide a second layer of protection against errors in the code. With the system completed, these tests would be able to target the complete testing of the interactions between the database, the API and the system itself.

Our approach to testing was very effective in ensuring a lack of bugs in our code. The programmers in our group were very good at ensuring that their code initially contained no bugs so the manual testing stage revealed very few errors and unit tests even fewer. However, the clearly defined two-stage approach to testing made the process much smoother and more robust and provided confidence in the stability of the final product.

5.1 Manual Testing

The manual testing of our system can be considered in two main parts due to the program's design. Firstly, that of consistent data entry and the corresponding graph. And, secondly, the testing of the responsiveness and behaviour of goals and achievements. This is reflected in the test plan in Appendix 11. The data entry portion of this testing required the intelligent selection of inputs to test general cases, edge cases and invalid input. An example of this was the following test case used to check the response of input boxes to incomplete inputs (See Figure 15). In this case an error was found, as the input caused the program to crash. The developer was quickly able to remedy this by adding an extra condition to the processing code.

Description	Test	Expected Result
Incomplete inputs are handled gracefully	Input a decimal point	Input is treated as a zero

Table 2: Decimal Point Bug

The second section of the manual testing phase involved the testing of the Achievement and Goal sections. Data input was less important to this part, serving purely to allow the reaching of different possibilities within the system. For example, in the test cases below, the developer used manual inputs to get to a state of having an achievement and then leave that state. In this example, the programmer discovered that if an achievement's requirement was reached and then the data was updated so that the requirement was no longer reached, some results were displayed incorrectly on the Achievements page (see Figure 14). The source of this issue was swiftly located and removed by the developer on discovery.

Description	Test	Expected Result
Achievement page highlights completed achievements	Input data which completes an achievement	Achievement page highlights completion and shows correct progress
Achievement page does not highlight previously completed achievements which are no longer complete	Input data which completes an achievement then edit data so that achievement is not completed	Achievement page reverts highlighting and progress label

Table 3: Achievements Bug

A relatively major issue that was found in this area was goals not correctly expiring once their dedicated end time was reached (See Figure 16). Once the test case below had been checked, the bug was quickly found and solved.

Description	Test	Expected Result
Goals expire once end time has been reached	Allow a goal to end uncompleted	Goal shows failure and ends

Table 4: Goals Bug

5.2 Unit Testing

Unit testing was also divided into two parts in line with our test plan: direct data entry testing and API integration testing. In function, direct data entry unit testing was effectively a more robust and exhaustive version of the manual testing of the same functionality, covering standard inputs, edge cases and invalid inputs. For example, Figure 18 shows a unit test designed to check that updating the number of steps a user has completed has the correct effect on the database. Similarly, a unit test ensuring correct database editing upon the creation of a new goal can be seen in figure 19. A variety of further unit tests and their results can be found in Appendix 11. Pleasingly, this unit testing did not uncover any issues with our code, showcasing the effectiveness of our manual testing and the initial programming of the system.

The unit testing of the API's interaction was similar to the first set of unit tests in terms of the tests themselves (see Figures 22 and 23). For each action that the API could be involved in, a unit test was designed to ensure that the effect on the database caused by the action was identical to the effect which would be caused by the same action performed manually.

6 Reflection and Conclusion

With our PI system now complete and the majority of the specified requirements met, we can reflect on the merits and failures of our system and development approach. To do this we will critically analyse the way our system was designed and specified, as well as the approach we followed during development. By doing this we will be able to conclude how successful our project was, consider future alterations to our system and analyse ways in which our approach could be improved. Learning from this, we will be able to develop systems more successfully and efficiently in the future. Unfortunately, due to the deadline for the completion of our project, it is not possible to utilise our system for a significant period of time in order to properly gauge how well it works from a user's perspective but we can still evaluate it from a critical viewpoint and make educated speculations about its effectiveness.

6.1 System Development

The creation of the specification for our system began with research to establish the interests of our target demographic. The biggest parts of this research were an online form distributed to our peers and some interviews (see Appendices 9 and 10). The results of these were very helpful and were the driving force behind our choice of tracking metrics. However, while this approach was sound and successful in gathering information, only around twenty individuals filled in the form and we only managed to interview two individuals. This is a large enough number of people that we feel comfortable with using the results as a guide, especially given the clear majority in the opinions of the form respondents. However, in future investigations, ensuring a larger sample size would improve the reliability of our results and, at the very least, make us more certain of our decisions.

Our choice of tracking metrics was informed both by data gathering and by research into other PI systems. This meant that our final list of four metrics reflected the desires of our peers, as well as our informed opinions on the most beneficial choices. On reflection, these four metrics still seem to provide an excellent and balanced summary of an individual's physical and mental health in a way that appears likely to provide useful insights and achievable targets. We are therefore confident that our choice of metrics is sound. However, due to the variety of research responses, especially from the interviewees, it was not possible to take the views of every individual into account; there were a few features suggested during the interviews which were not added to the specification, either due to the need to limit the scope of the system or because they didn't align with the conclusions of other respondents or with our research. In future, looking back on this research or through conducting more thorough research, we could implement extra features over a longer period if a large proportion of our user base shared a desire for a particular feature.

One key requirement we specified for our system, which is a very important feature of any PI system, was the ability to log data. This specification was clearly met, with the user able to log their progress in all four metrics. The manual data entry process is also very simple and intuitive. Furthermore, by implementing it in this way we have made the data entry aspect of our PI system as effortless as possible, thereby making the use of our system more pleasant for the user. The system also has the ability to use the Fitbit API to gather data, as specified in the requirements. While this does work well and fulfils the requirements, Fitbit refresh API tokens on a regular basis. This means that users are regularly required to re-enter their Fitbit API token for our system to gather their data, reducing the benefits of the automated tracking. In future, it would be beneficial to look into avoiding this problem so that the effort required on the part of the user is minimised. Avoiding this is presumably possible as the Fitbit mobile app is reportedly built using this API and does not require repeated token entry from users.

Another important PI feature outlined by our specification was the intuitive display of user data and progress. This requirement was successfully met through the use of a software library specialising in graphs. Using this library we managed to create aesthetic, interactive and detailed graphs for chosen all metrics chosen by the user, giving users insight into health variations in their lifestyle. A possible area for improvement in this section would be the highlighting of particularly healthy periods of time to motivate the user, as this has been shown to promote positive reflection (Loerakker et al., 2023). However, the functionality of this feature certainly meets the requirements we set out.

Another key feature of our requirements is the inclusion of goals and achievements. This was also successfully implemented and provides an effective way to track progress towards goals and to provide motivation, making it a worthwhile part of the specification and the system. In the future, a possible improvement would be the ability to view other users' achievements within the application. Whilst successes can be displayed in-person or via the sharing of screenshots, a built-in system would make this easier and could help spread word of the system to more students who might find it useful. Additionally, the ability to have multiple goals per activity could be beneficial for some users. For example those who would like to have a short-term goal and a longer term goal.

The database our system uses is designed to intuitively integrate with the structure of our software. Additionally, it is quick and easy to understand and access. The addition of future metrics to our system would not be challenging to implement, simply requiring an associated table. However, whilst our database is intuitive, it could be normalised further by adding a bridging table with activity IDs. The benefit of this change would be increased efficiency and ease of further development. Adding new metrics would likely only require an extra record in the Activities table. However, there would also be downsides to this approach; understanding the layout of the database would be less simple, relying on keeping track of foreign keys and more complicated database queries.

Our class design closely resembles that of our database and is therefore similarly intuitive. If we were to alter the database design, careful alterations to the interface between the database and the code could maintain the clarity of data storage from a programming perspective by minimising changes in the class design.

As per our specification, the PI system is a locally-run PC application. This is beneficial to users when it comes to the viewing of the graphical statistics provided by the program as the majority of computers running the program will likely have a larger screen than a tablet or smartphone. However, it also requires that users log data using a large device. This could become quite burdensome, especially when away from home due to the these requirements. Given the widespread use of smartphones which are carried almost everywhere, a future improvement to our system would be to allow the tracking of data on mobile devices, making the process much more portable and accessible.

Our approach to the test-driven development of our system worked effectively in practice, with developers finding most of the bugs in their own code during testing and the finished product working effectively with no known issues. By using white-box testing the developers were able to apply tests based on their in-depth knowledge of the processes involved, thereby anticipating problems others would not. However, on one occasion, a bug went unnoticed by the initial testing and was only picked up by the product owner. This implies that, in future, a combination of white-box and black-box testing might prove to be more robust due to the diverse perspectives provided by group work.

6.2 Software Process

To develop this product we followed the Scrum framework and the Agile methodology of software development with the aim of allowing us to structure and manage the creation of the system in an effective and methodical way. We managed to complete five week-long sprints during the development process with the sprint backlog successfully cleared during the majority of the sprints and mostly cleared during the rest. This process ensured that we kept working at a sensible pace whilst adhering to the specification. It also allowed the system requirements to evolve during development, leading to a significantly improved final product.

However, our development process was by no means perfect. One way in which we could have improved our adherence to the Scrum framework would have been the inclusion of more frequent Scrum meetings. During our sprints, frequent Scrum meetings were held to discuss progress but they did not always occur on a daily basis. Holding daily scrum meetings would have been impossible given our busy and clashing schedules, especially during the Easter Break when developers had significant plans outside of work. However, more Scrum meetings could have been held on occasion and may have improved communication and understanding within the group.

Another notable way the Agile methodology allowed us to evolve our requirements was in the design of the system's database. In a sprint planning session during the development process, we established that our database could be improved by adjusting the design. The product backlog was updated to include this new design and the change was implemented as part of the sprint that followed, making the database more efficient. In this way, the Agile methodology helped us avoid future problems and work by allowing flexibility during the development process.

One part of the process that worked effectively for our group was the sprint planning weekly meetings. By having a member of the team run the meetings, and through open communication, we managed to always ensure that every member of the team had something to do and could work on them effectively during the sprint so that the group as a whole produced meaningful results during the sprint and finished the project before the deadline.

6.3 Conclusion

In conclusion, through carefully considered design and adherence to Agile methods, we managed to successfully produce a working PI system in an organised way. We believe that this system is intuitive, informative and robust. Furthermore, our approach to testing and development was effective, however testing could have been improved by incorporating a more collaborative system. Therefore, the development of this system has primarily been a success. Despite this, there are areas in which the system could be improved in the future to increase ease of use which, given a longer development time, could have been addressed accordingly.

7 References

[1] Jones, S. L. and Kelly, R., 2018. Dealing With Information Overload in Multifaceted Personal Informatics Systems. *Human–computer interaction*, 33(1), pp. 1–48. https://doi.org/10.1080/07370024.2017.1302334

[2] Li, I., Medynskiy, Y., Froehlich, J., and Larsen, J., 2012 Personal informatics in practice: improving quality of life through data. *Human–computer interaction*, 33(1), pp. 2799–2802. https://dl.acm.org/doi/10.1145/2212776.2212724

[3] Kersten-van Dijk, E. T., Westerink, J. H. D. M., Beute, F. and IJsselsteijn, W. A., 2017. Personal Informatics, Self-Insight, and Behavior Change: A Critical Review of Current Literature. *Human–computer interaction*, 32(5–6), pp. 268–296.

[4] Potapov, K., Vasalou, A., Lee, V. and Marshall, P., 2021. What do Teens Make of Personal Informatics? Young People's Responses to Self-Tracking Practices for Self-Determined Motives. *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 8-13 May 2021, New York. Association for Computing Machinery, pp. 1–10.

[5] Loerakker, M.B., Niess, J., Bentvelzen, M. and Woźniak, P.W., 2023. Designing Data Visualisations for Self-Compassion in Personal Informatics. *Proceedings of the ACM on interactive, mobile, wearable and ubiquitous technologies*, 7(4), pp. 1–22.

[6] Madore, K. P. and Wagner, A. D., 2019. Multicosts of Multitasking. *Cerebrum:the dana forum on brain science*[Online], 4(19). Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7075496/#sec-a.c.ftitle [Accessed 5 April 2024]

[7] National Health Service, 2023. *Water, drinks and hydration* [Online] Available from: https://www.nhs.uk/live-well/eat-well/food-guidelines-and-food-labels/water-drinks-nutrition [Accessed 5 April 2024]

[8] Rapp, A. and Cena, F., 2016. Personal informatics for everyday life: How users without prior self-tracking experience engage with personal data. *International journal of human-computer studies*, 94, pp. 1-17

[9] Tudor-Locke, C., Craig, C. L., Brown, W. J., Clemes, S. A., De Cocker, K., Giles-Corti, B., Hatano, Y., Inoue, S., Matsudo, S. M., Mutrie, N., Oppert, J. M., Rowe, D. A., Schmidt, M. D., Schofield, G. M., Spence, J. C., Teixeira, P. J., Tully, M. A. and Blair, S. N., 2011. How many steps/day are enough? For adults. *The international journal of behavioral nutrition and physical activity* [Online], 8(79). Available from: https://doi.org/10.1186/1479-5868-8-79

[10] Watson, N. F., Badr, M. S., Belenky, G., Bliwise, D. L., Buxton, O. M., Buysse, D., Dinges, D. F., Gangwisch, J., Grandner, M. A., Kushida, C., Malhotra, R. K., Martin, J. L., Patel, S. R., Quan, S. F. and Tasali, E., 2015. Recommended Amount of Sleep for a Healthy Adult: A Joint Consensus Statement of the American Academy of Sleep Medicine and Sleep Research Society. *Sleep*, 38(6), pp.843–844.

8 Appendix: Group Contribution Form

Name	Email	Contribution	Comment
Akim	ak3625	10	Led meetings, essentially managed the whole project and was constantly giving feedback
Alex	ah3208	8	Good group member who fully participated
James	js4209	8	Good group member who fully participated
Jeet	jm3522	10	Took on most of the technical element of the project and was always working on the program
Katrya	kd794	8	Good group member who fully participated
Martin	md2353	8	Good group member who fully participated
Ollie	ov247	8	Good group member who fully participated
Tom	tb2301	8	Good group member who fully participated
Thang	ctn32	8	Good group member who fully participated

For additional information refer to Appendices 12 and 13.

9 Appendix: Survey

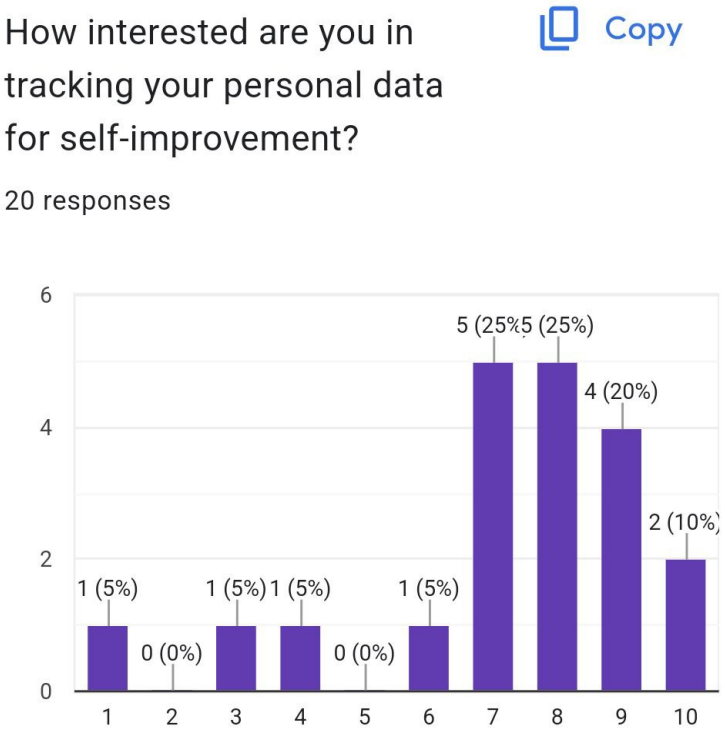


Figure 11: Survey Bar Chart

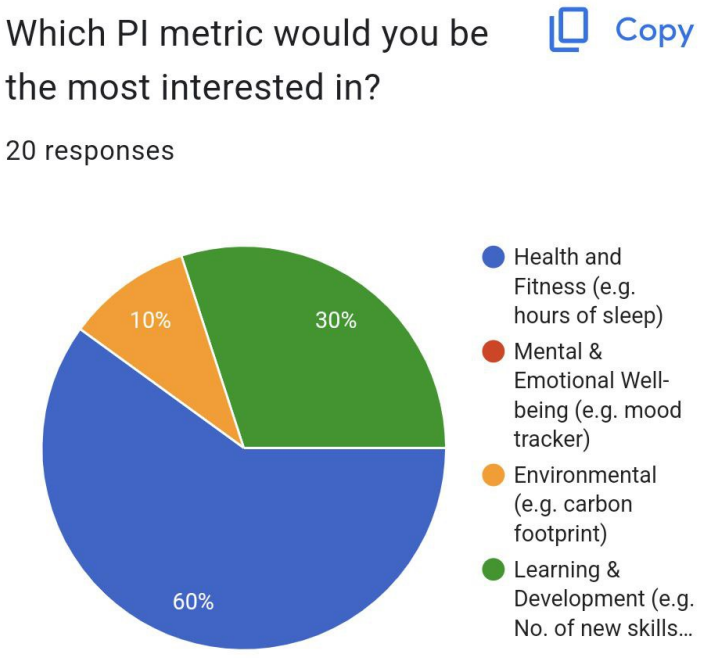


Figure 12: Survey Pie Chart

10 Appendix: Interview Transcripts

Interviewee: Edwin Smith

Q1: What are you looking for in an app to help you improve your sleep schedule?

> Something that relaxes me.

Q2: How many hours of sleep do you get on average, per day?

> Around 10 hours of sleep, however, I want to reduce this amount.

Q3: Do you think that your calorie intake is greater than, less than, or around the same as the daily calorie intake of the average adult?

> Probably more, but I am happy as I am. My metabolism is high and the daily average also depends on your physical build.

Q4: Would you be open to participating in challenges/competitions within the app to help you meet your fitness goals? If so, what kind of challenges are you looking for?

> Yes, I would be down for challenges, especially amongst other people or friends. Since I am an avid gym enthusiast, it would be cool to see a "who can eat 'n' amount of calories in a day" kind of challenge. Note that this is only for bulking reasons, however you can implement better calorie intakes depending on what the person wants. Having a challenge based on the number of steps or distance traveled would improve my cardio a lot, especially if a friend of mine could also join in.

Q5: How important is the user interface for you? Will you like a simple minimalistic design, or a UI consisting of more options and possibly more complex?

> I would like a simple but powerful UI, which allows you to see a bunch of data visualised. I would like to see a leaderboard of my friends, and a weekly challenges button so it keeps everyone busy.

Q6: Any other information you believe could be useful?

> No.

Interviewee: Oliver Niles

Q7: Are there any visualisations or reports you would like to see to help you understand your fitness data better?

> Yes, I would like to see line graphs where I can customise the range of data, ranging from days to years. Every week I would like a report which outlines the key data points of my fitness in order to determine what I can do to improve.

Q8: What kind of feedback/encouragement would you like to see from the app, and how would you like this to be displayed?

> You could implement a point system as a form of encouragement, where you could customise your account and compare with other friends on the app. I am more likely to use the app if it is more competitive with other people.

Q9: How important are social features, like being able to view other people's achievements, like your friends?

> Quite important to be able to view other people's achievements, but you could make it so your account is not visible to other people as well. However, it is important to focus on yourself because sometimes it can be unrealistic especially if the person you are comparing with is not at a level that you are at.

Q10: Do you have any fitness/health goals that you would like the app to help you achieve?

> Yes, I would like to be able to gain muscle over the next year by tracking my calorie intake and exercises that I have done. I'd also like to be able to get more sleep using the app's recommended bedtime schedule for my goals.

Q11: Any other information you believe could be useful?

> After I input the calories of a meal, I would like to have the option to save the meal as a preset which I can add to any day, which automatically adds the calories of that meal to my daily calories count so that I don't have to type it in each time.

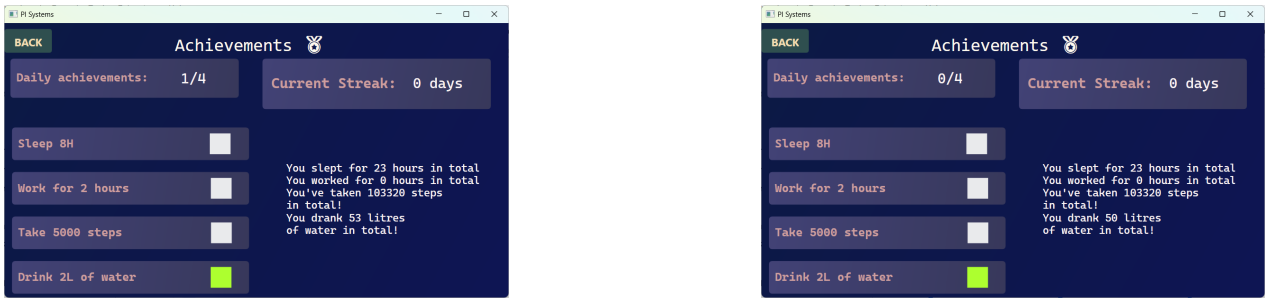
11 Appendix: Testing

11.1 Testing Plan

<table>
<tr><th>Objectives of Testing</th><th>Scope of Testing</th><th>Testing Approach</th><th>Test Enviroment</th><th>Test Data</th><th>Evidence</th></tr>
<tr><td>- Data pulled form DB should not be altered.</td><td rowspan="2">Stage 1, Sprint 2-4</td><td>Manual Test, Enter Data and check home screen and graph</td><td rowspan="4">Visual Studio, multiple Window's OS in multiple laptops.</td><td>Edge cases, empty inputs and standard data.</td><td>Screenshots</td></tr>
<tr><td>- UI should display the expected.</td><td>Manual Test, Change the state of completion of achievemnts and compare results.</td><td>Achievements acquisition.</td><td>Screenshots</td></tr>
<tr><td>- Data Manully entered must be shown to be recorded in the DB flawlessly.</td><td rowspan="2">Stage 2, Sprint 5</td><td>Unit Test, Check through dummy data the proficiency of our system.</td><td>Edge cases, empty inputs and standard data.</td><td>Output Log</td></tr>
<tr><td>- Data entered through an API should be equivalent to manully entered data in DB</td><td>Unit Test, Run a unit test over the Database and API.</td><td>Standard data</td><td>Output Log</td></tr>
<tr><td>Responsible: James and Alex</td><td>Responsible: Akim</td><td>Responsible: Martin and Jeet</td><td>Responsible: Group Effort</td><td>Responsible: James</td><td>Responsible: Jeet and Martin</td></tr>
</table>

Figure 13: Testing Plan

11.2 Manual Testing



(a) Achievement Correct (b) Achievement Incorrect

Figure 14: Achievement Test

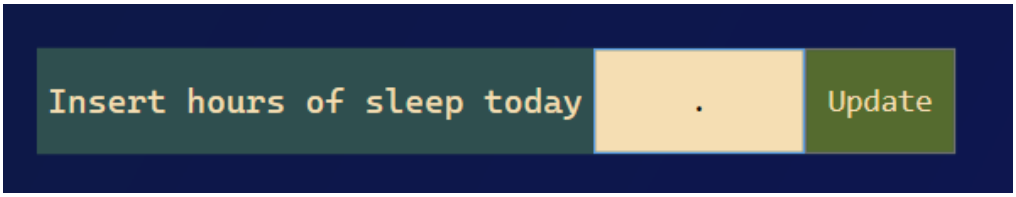


Figure 15: Decimal Point Error

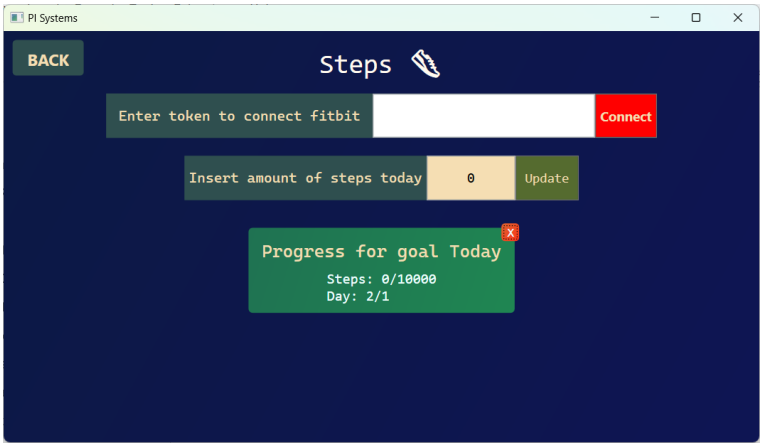


Figure 16: Goal Not Expiring

11.3 Unit Testing

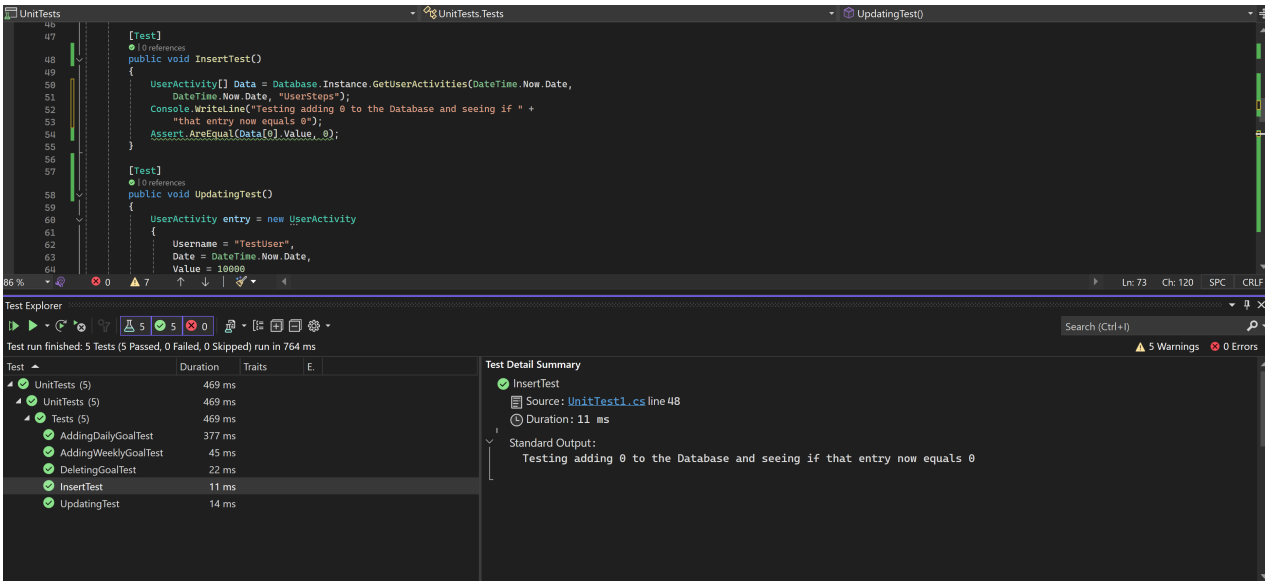


Figure 17: A test demonstrating an Insertion into the Steps Table in the Database

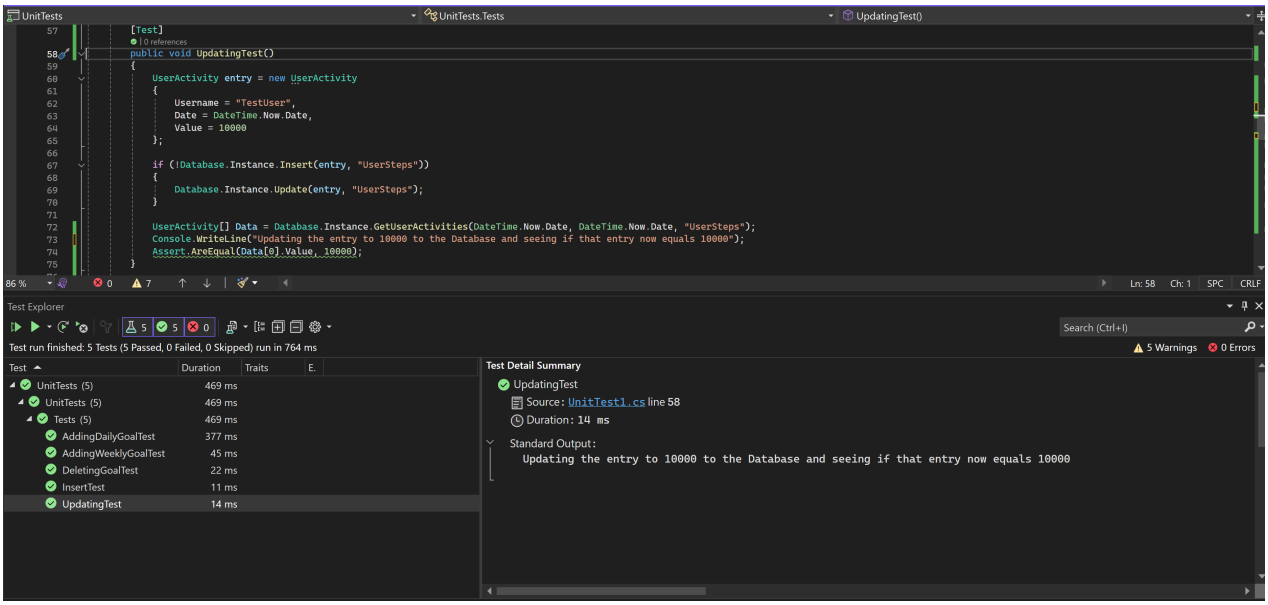


Figure 18: A test demonstrating an Update to the previous Insertion

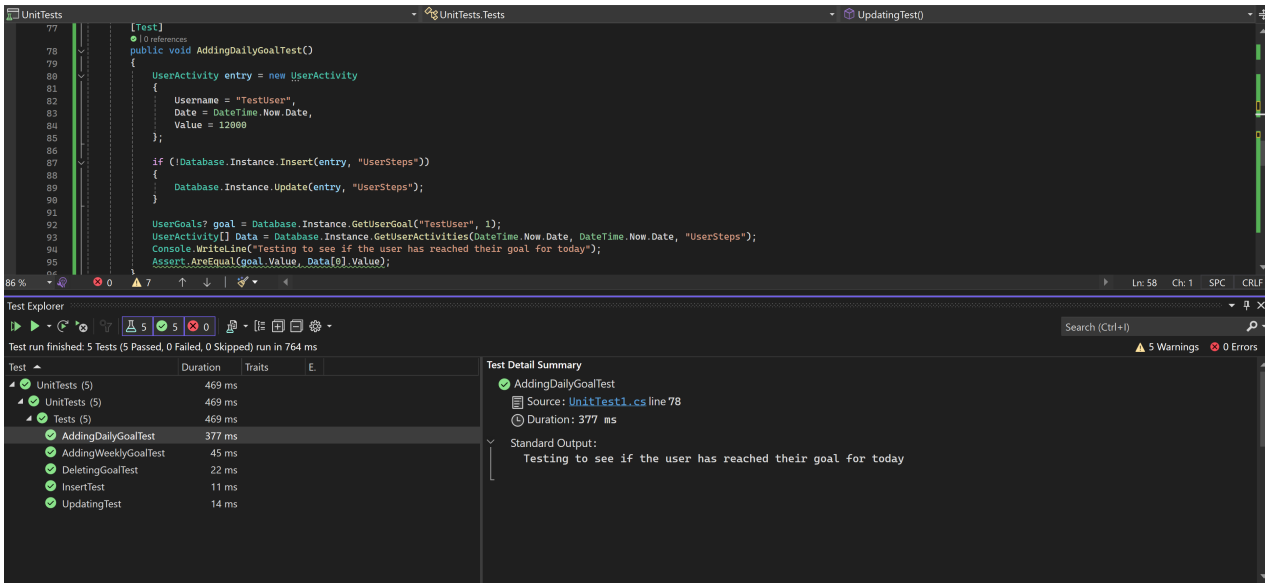


Figure 19: A test demonstrating an Insertion of a Daily Goal in the Database

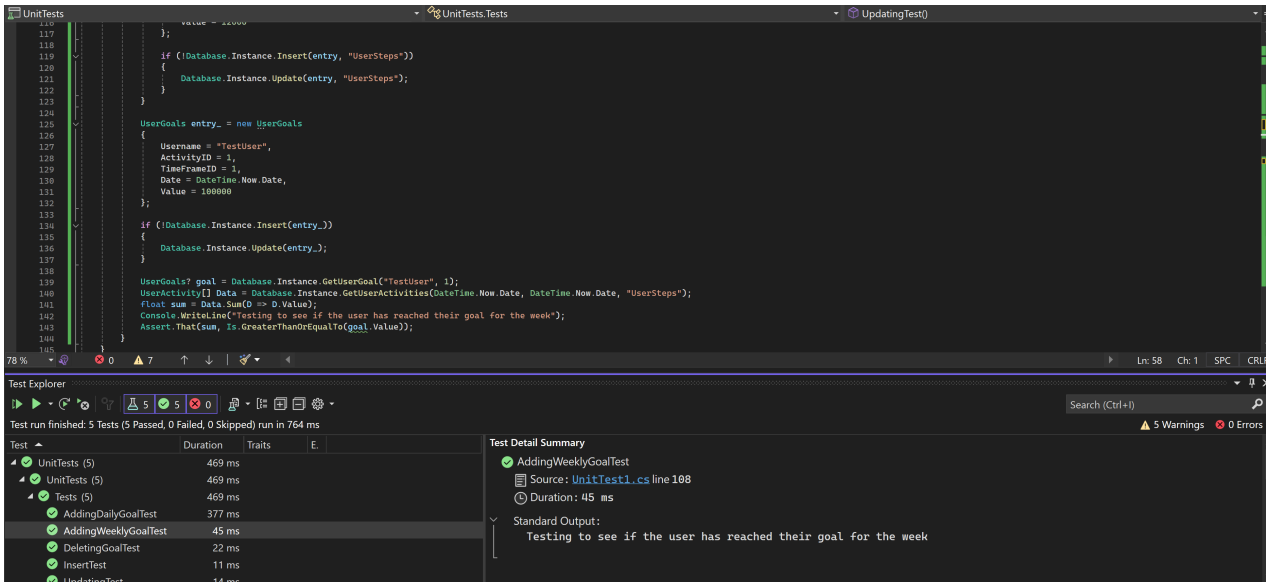


Figure 20: A test demonstrating an Insertion of a Weekly Goal in the Database

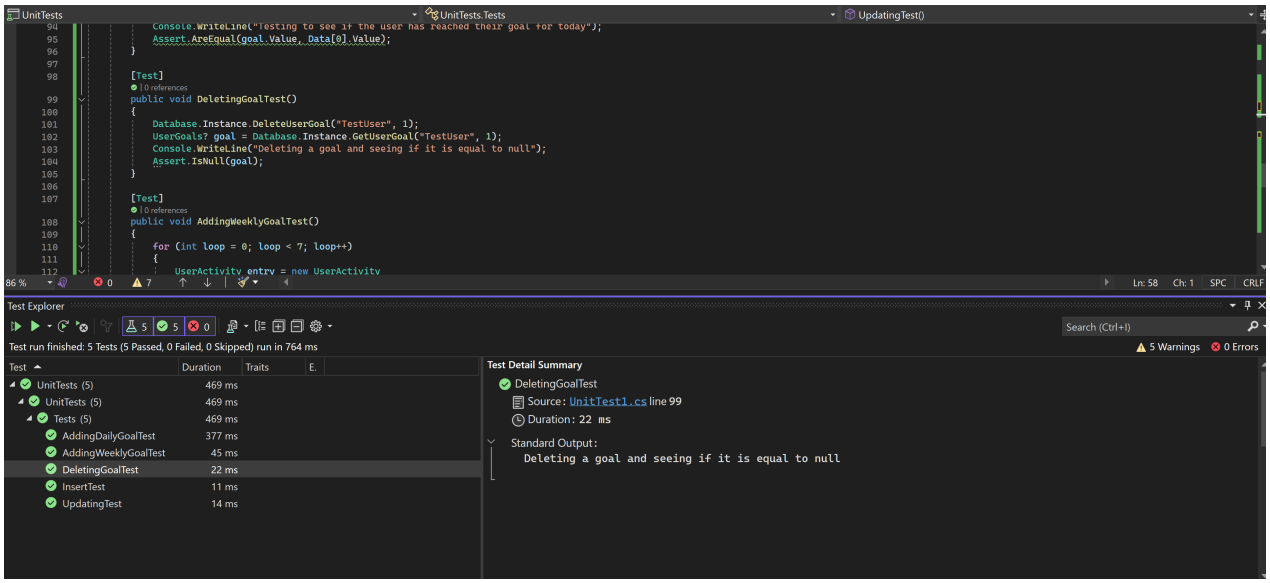


Figure 21: A test demonstrating the deletion of a Goal in the Database

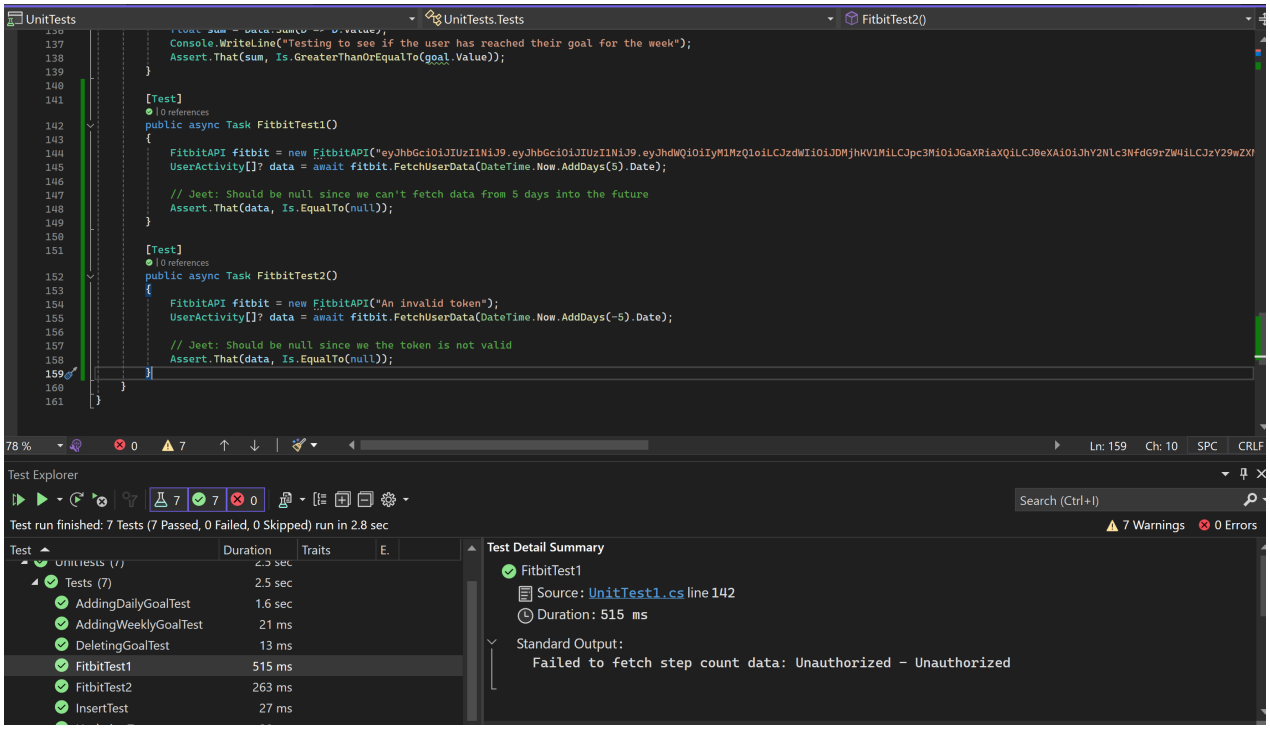


Figure 22: A test demonstrating a connection to the FitBit API but trying to access information that doesn't exist

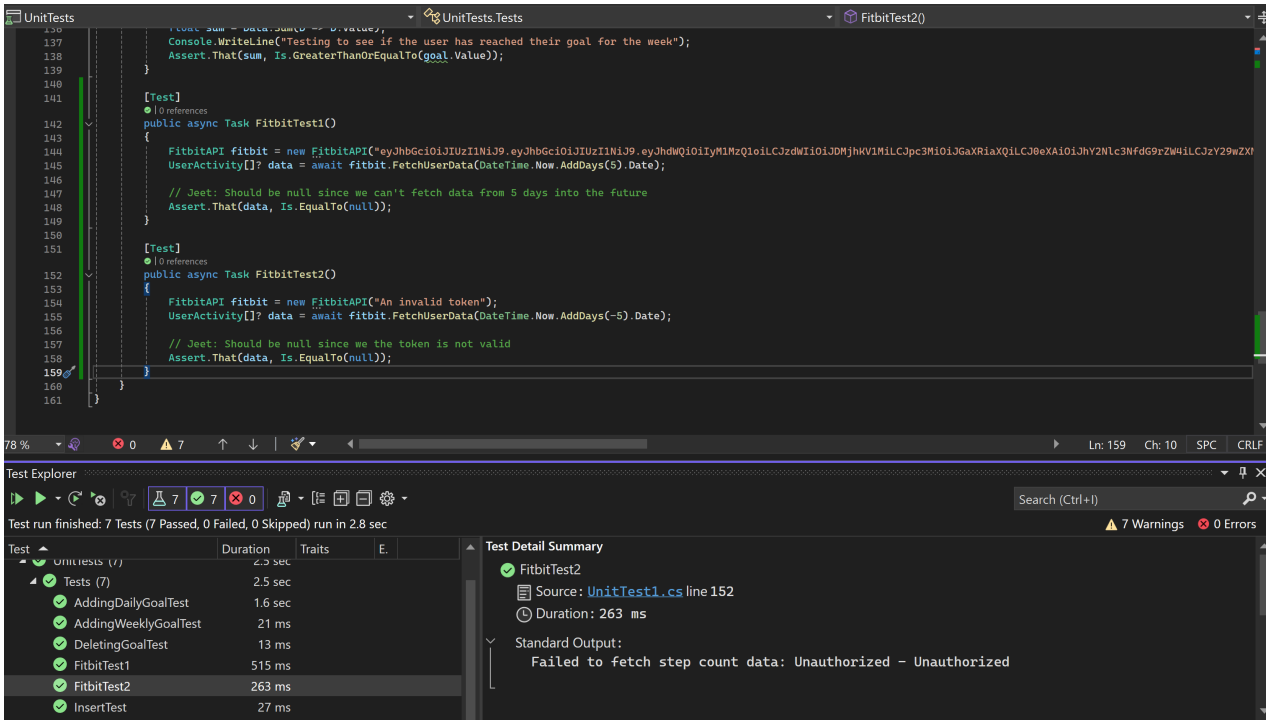


Figure 23: A test demonstrating a connection to the FitBit API but using an invalid token

12 Appendix: Contribution Table

Name	Week 1	Week 2	Week 3	Week 4	Week 5
Akim	Led meeting, took down minutes, created Github Repo, created latex report framework and created and ran a survey	Led meeting, took down minutes, assigned workload, created UI diagram and presented to the Technical Team	Led meeting, took down minutes, assigned workload and reviewed the documentation	Led meeting, took down minutes, assigned workload and reviewed the documentation	Led meeting, took down minutes, assigned workload and reviewed and improved the documentation
Alex	Researched PI structures, found out what data to collect and researched the effective of PI systems	Created UI diagram with Akim	Attempted to work on UI	Documented the functional requirements and justifications for each	Adding requirements to the Design section
James	Reviewed, commented and created priorities to each given requirement and added some additional ones	Worked on Steps and Work UI and added some functionality	Added more functionality to the work page e.g. timers	Added ability to delete work goals and all work done adds to a total work value	Reviewed and provided feedback on the report
Jeet	Interviewed people about PI systems and created transcript about key data gathered	Made database ERD, UML class diagrams, researched SCRUM and finished Menu UI	Updated the way databases work and created goal table that works with the UI	Connected Fitbit API with the app, where the user can input their token to download all their data	Made a Work table that works with James' UI and added some functionality to the update button
Katrya	Made table comparing appropriate languages for the UI design	Researched databases and C# connections	N/A	Created achievements UI and functionality	Polished up achievements UI

Martin	Made summary on research into Fitbit software	Wrote Introduction section	Wrote Agile software section of report	Started Testing section and updated Agile section	Wrote reflection, conclusion sections and updated Agile section
Ollie	Made summary on two articles on PI systems	Created the UI, made theme and researched graphing	Applied graphs to the UI with each statistic	Connected graphs to database and added the ability to change time period	Screenshotted prototype 1, integrated work into graphs and polished graphs UI
Tom	Made summary on research into Garmin software	Created basic program showing how Fitbit API could be used	Rephrased specification section and added more detail to documentation	More rephrasing in specification section, finished non-functional requirements and polished other parts	Updated specification section to reflect our program, requirements table
Thang	Made summary on two articles on PI systems	Wrote first half of specification section and drafted second	Started writing design section and finished ERD description	Completed design description with UML models and added non-functional requirement justifications	Adjusted UML description and finished UI design description

13 Appendix: Minutes

13.1 Minutes From 25/03

13.1.1 Attendance

- Taking attendance.

13.1.2 Meeting Notes

- James is currently responsible for making the group contribution table and the attendance table.
- The plan for the first sprint was discussed as a group discussion. See the next section for details.
- With everything discussed, team members have started working on their tasks.

13.1.3 First Sprint (25/03 — 01/04)

Tasks:

1. **Gather Articles:**
   - Each member finds two articles and provides a brief summary of each.
   - Everyone must research at least one article by this Wednesday.
   - **Responsible:**
     - Alex — will provide at least one article from the specification.
     - Thang
     - Ollie
2. **Interviewing:**
   - **Responsible:**
     - Jeet — for live interviewing and taking transcripts.
     - Akim — creates and distributes a Google Form/survey.
3. **Market Research:**
   - Potentially provide reference(s).
   - **Responsible:**
     - Tom — researches one competitor and provides a summary of his findings.
     - Martin — researches another competitor and provides a summary of his findings.
4. **Draft the Initial Requirements:**
   - **Responsible:** James

5. **Start the Report in LaTeX:**
    - **Responsible:** Akim — provides the initial framework.
6. **Research Programming Languages:**
    - **Responsible:** Katrya — to research languages including Python, C#, Java, C++, and JavaScript (in case we decide to create a website instead of implementing on PC), and others if necessary. Create a table listing different characteristics to compare and contrast, and provide a brief summary/conclusion.

13.1.4 Next Week Discussion Points

- Finalize the initial version of the spec, including:
    - What are the requirements?
    - Deciding on the PI to use.
- Which language to use? Is Python suited for the job? Consider if it's only suitable as a prototype.

13.2 Minutes From 01/04

13.2.1 Attendance

- Taking attendance.

13.2.2 Discussion Topics

- **Requirements Table:** Discussion on the requirements table prepared by James.
- **SCRUM Master:** Jeet has volunteered to be the SCRUM master, with responsibilities including in-depth SCRUM research and advising on future strategy steps.
- **Meeting Leader:** Akim has volunteered to be the meeting leader for all future meetings.
- **Writing Assignments:**
    - Alex will complete an article from the specification by today.
    - Alex, Ollie, and Thang are tasked with inserting references into the requirements table by the end of the week.
- **Programming Language:** C# has been selected as the programming language for the project.
- **Development Roles:**
    - Jeet and Ollie have been appointed as front-end developers.
- **Sprint Planning:** Discussion about the second sprint.

13.2.3 Second Sprint (01/04 — 08/04)

Tasks:

1. **Create a UI Diagram by Wednesday:**
   - **Responsible:**
     - Akim
     - Alex
2. **Create Framework for the Project on GitHub:**
   - **Responsible:**
     - Jeet
3. **After the UI Diagram is Finished, Create a Coding Plan:**
   - It should include both front-end and back-end.
   - **Responsible:**
     - Jeet
     - Ollie
     - James
4. **Potentially Start the UI for the Menu:**
   - **Responsible:**
     - Jeet and/or Ollie
5. **Research APIs:**
   - Create a small program with comments showing off basic functionality.
   - Share the program on GitHub in the `research` directory (create if doesn't exist).
   - **Responsible:**
     - Tom
6. **Start Writing the Content of the Report:**
   - **The Introduction section:**
     - Martin
   - **The Software Requirements Specification:**
     - Thang
7. **Create a Basic DB Using SQL and C#:**
   - A small program with comments that describes basic operations.
   - Share the program on GitHub in the `research` directory (create if doesn't exist).
   - **Responsible:**
     - Katrya

13.3 Minutes From 08/04

13.3.1 Attendance

- Taking attendance.

13.3.2 Discussion Topics

- **Discussing the Report:**
  - Martin has completed his task and will merge the report branch into the master after the meeting.
  - Thang has completed his task and will upload his work on GitHub by the end of today.
- **Presentations:**
  - Jeet presented the UML diagram.
  - Jeet presented the ERD.
  - Jeet walked everyone through the code.
  - Oliver presented how the graphs will be implemented.
- **APIs:**
  - Tom discussed the research he has done on APIs and will upload it to GitHub today.

13.3.3 Third Sprint (08/04 — 15/04)

Coding tasks:

1. Potentially fully code the **Graphs** part.
   - **Responsible:**
     - Oliver
2. Potentially fully code the **Achievements** part.
   - **Responsible:**
     - Alex
3. Potentially fully code the **Work** part:
   - **Responsible:**
     - James
4. Fully implement the **goals for all metrics** apart from Work.
   - **Responsible:**
     - Jeet

5. Implement the **API for Steps**:
    - **Responsible:**
        - Akim or Jeet (to be decided)
6. Consider **changing the DB design**:
    - **Responsible:**
        - Jeet

Report tasks:

7. **Go through the report and give feedback** to Thang and Martin:
    - **Responsible:**
        - Akim
8. Go through the report part of the spec and **allocate work** to Tom:
    - **Deadline:** Wednesday
    - **Responsible:**
        - Akim
9. Complete the **Introduction Section**:
    - **Responsible:**
        - Martin
10. Start the **SCRUM Section**:
    - **Responsible:**
        - Martin
11. Start the **Design Section**:
    - **Responsible:**
        - Thang

13.4 Minutes From 15/04

13.4.1 Attendance

- Taking attendance.

13.4.2 Discussion Topics

- **Ollie's Presentation on Graphs:**
  - Ollie completed the basic version of the graphs, which currently lacks functionality.
- **James' Update on Database Integration:**
  - James added database integration to the Work page.
- **Jeet's Contributions:**
  - Implemented goals for all metrics.
  - Simplified the database design.
  - Began integrating the API.
- **General Discussions:**
  - The team discussed the report.
  - The Scrum development process was reviewed.
  - Testing plans were outlined; testing will commence after product completion next week.
  - Reviewed the project requirements.

13.4.3 Fourth Sprint (15/04 — 22/04)

Coding tasks:

1. Finish the **Graphs**.
   - **Responsible:** Ollie
2. Develop the **Achievements UI** and Start **Functional Implementation**.
   - **Responsible:**
     - Katrya
     - Potentially Jeet (see #5)
3. Complete the **Work Section**.
   - **Responsible:**
     - Potentially James (confirmation by end of today)
4. Finish the **API Integration**.
   - **Responsible:**
     - Jeet

5. Assist with **Achievements Functionality**.
    - **Responsible:**
        - Jeet

Report tasks:

6. **Review** the Report and **Provide Feedback**.
    - **Responsible:**
        - Akim
7. **Assign Tasks** for Martin and Alex.
    - **Deadline:**
        - Wednesday
    - **Responsible:**
        - Akim
8. Finish the **Requirements Section**.
    - **Responsible:**
        - Tom
9. Finish the **Design Section**.
    - **Responsible:**
        - Thang
10. **Update the Scrum Section**.
    - **Responsible:**
        - Martin

13.5 Minutes From 19/04

13.5.1 Attendance

- Taking attendance.

13.5.2 Discussion Topics

- Discussing progress.
- **API**:
    - Jeet is almost done, will finish the API integration by Sunday.
- Start the software testing section of the report.
    - **Responsible**:
        * Martin
- Everyone else is continuing with their work for Monday.

13.6 Minutes From 22/04

13.6.1 Attendance

- Taking attendance.

13.6.2 Progress Review and Final Sprint (22/04 — 03/05)

Coding:

- Ollie has finished the **Graphs** except for the Work part, which has not been fully implemented yet by James.
- **Achievements UI** has been completed, but **not the functional part**, which will be done by Wednesday.
  - **Responsible**: Katrya
  - **Deadline**: Wednesday
- The **Work** part is not fully finished yet, it has to be linked to the DB and properly merged with the rest of the code by Wednesday.
  - **Responsible**: James
  - **Backup**: Ollie
  - **Deadline**: Wednesday
- Jeet has **fully integrated the API part**.
  - Some **bugs** have been discovered by the product owner; they **will be fixed shortly**.
- Jeet will **merge everything into the master branch** by Wednesday.
  - **Responsible**: Jeet
  - **Deadline**: Wednesday
- Jeet will implement the table in the DB for Achievements by Friday.
  - **Responsible**: Jeet
  - **Deadline**: Friday

Report:

- Tom has almost finished the **Requirements section**, however it has to be redone in the form of a table.
  - **Responsible**: Tom
  - **Deadline**: Wednesday

- Thang and Alex have not been able to complete the **Design section** by today, but they will fully complete it and push to GitHub by no later than Wednesday.
  - **Responsible**: Thang, Alex
  - **Deadline**: Wednesday
- Martin has updated the **Scrum section** for the previous week, and will do the same for this week.
  - **Responsible**: Martin
- Start the **Reflection and Conclusion** section of the report
  - **Responsible**: Martin
- Give feedback on the **Testing section** and let Martin know what to do next
  - **Responsible**: Akim
  - **Deadline**: Wednesday

13.7 Minutes From 29/04

13.7.1 Attendance

- Taking attendance.

13.7.2 Progress Review and Report Finalization

- Upload the **updated ERD** to the Google doc.
  - **Responsible**: Jeet
- Update the **ERD** in the report.
  - **Responsible**: Thang
- Insert **ARQ/SRQ notation** to the rest of Thang's Design section.
  - **Responsible**: Alex
- Complete the **Requirements** section now.
  - **Responsible**: Tom
- Update **SCRUM** and complete the **Testing** section.
  - **Responsible**: Martin
- Go through the report and give feedback.
  - **Responsible**: James, Akim

- Include **appendices** in the report.
  - **Responsible**: Akim
- Ollie successfully integrated the **work section with the graphs** and implemented **dynamic upper bound** for the graphs.