A PI System to Track Sleep, Productivity & Mood

Abstract

Personal Informatic systems encompasses technology which aids users in tracking their personal data. The problem our team faced was to attempt to create such a system geared towards university students. We also had to ensure that we followed an Agile software development life cycle involving a minimum of three sprints. Our system was created within four sprints, and focuses on allowing a university student to track their sleep, productivity and quality of day. In our system, the user is able to rate productivity and quality of day out of 10 for the day, and give the amount of hours they slept for sleep quality. The user is able to compare this data on a graph over time, or can compare each variable against each other, such as choosing to plot sleep quality against productivity for the last week. To manage productivity, the user is able to create a list of tasks they would like to complete as well, checking them off when completed. Currently, the user is able to set three goals, one concerning each PI variable, and will achieve a bronze, silver or gold badge depending on how well they meet their goal. Enhancements to the system include greater personalisation for each user. Currently, the user can reflect on the graphs showing their data themselves, but the system being able to identify possible trends, and indicating such trends to the user, could allow for a deeper understanding of their data and its connections.

Table of Contents

Title and Abstract 1

Table of Contents 2

Introduction 3

Agile Software Process Planning and Management 5

Software Requirements Specification 7

Design 12

Software Testing 15

Reflection and Conclusion 19

References 23

Appendix A - Key Questionnaire Responses 25

Appendix B - Test Table 30

Introduction

Personal Informatics (PI) refers to systems that allow users to track relevant information about their lives for a multitude of reasons: self-improvement, self-reflection, and for motivation, as a few examples. Effectively, the goal is to aid users in improving self-knowledge, allowing them to act on this knowledge in a positive way. (Li et al., 2011). For instance, there are various technologies, such as Sleep Cycle (2009), that are designed to track a user's sleep quality in the hopes of improving their sleep schedule and helping them feel more rested in the morning. There has been much investigation into the effects of PI systems, with regards to how positively it can affect users. There are those who have looked into possible negative side effects, such as Ian Li (2012), who revealed that software developers need to ensure that their PI system deals with sensitive data, such as weight, in a way that is respectful to the user. Dijk et al. (2017) examined whether PI systems are currently successful in providing users with the drive to act on improving in some way. Promisingly, they found that the future of PI systems seemed bright, despite concluding that we needed a better understanding of the effects of PI systems to encourage users optimally. In our project, we have attempted to design a system that aims to improve the quality of a student's day by tracking and monitoring their sleep quality, productivity, as well as how good they thought their day was.

In creating our system, we conducted research to determine features that can make a PI system more effective for its users, in the sense that they would be more motivated to improve on their behaviour. One study, which focused on enhancing fitness through wearable technologies, emphasised the distinction between reflexive technologies and persuasive ones, where the latter explicitly hopes to change a user's behaviour in a beneficial way (Fritz et al., 2014). In making a PI system persuasive, the study discussed the benefits of a goal and reward system in motivating users. For example, the study revealed how some users would walk more even if they had already been active simply because they were close to getting a system reward for reaching a certain number of steps. The types of system rewards were also discussed; collecting badges were explicitly revealed as a source of motivation. We can see this through looking at current, successful PI systems used today. Fitbit, for example, motivates users in one way by allowing them to unlock badges depending on their activity throughout the day, such as 'the hiking boots' badge, which the user can achieve by walking 35,000 steps in one day. (Kosecki, 2017).

Furthermore, personalisation is also important in creating an effective PI system. It appears that if the user is able to have a personalised experience with the system, the user is more likely to reach their goals and use the system long-term (Gulotta et al., 2016). How this personalisation is achieved depends on the system; personalisation

could be incorporated by allowing the user to set their own goals, for instance. Many researchers have looked at the integration of personalisation to help users take action to change their behaviour. Rapp et al. (2018), for example, suggested that users of PI systems, especially those new to the idea, would benefit from recommendations on how to act based on their data, as they may be unsure of how to best act to improve. Lee et al. (2005) also explored personalisation, arguing that it could be found through encouraging users to reflect on their own goals and desires, allowing users to discover what they think is important for themselves.

From prior research, another characteristic of PI systems that can help make the system effective is the ability to present numeric feedback to the users. For example, Fritz et al. (2014) found that users who were able to see their PI data numerically were more motivated to partake in certain activities in order to improve the number shown to them; specifically, they found that, in some cases, people would aim to walk more simply to increase their step count. Interestingly, they also concluded that the user's understanding of the number shown to them was a source of motivation even if they did not completely understand the meaning of the data. That is to say, the desire to increase the number, despite what it represented, was enough to motivate users to change behaviour.

Moreover, the main intent behind our particular PI software system is to allow the user, a university student, to track and reflect on the quality of their day, and be able to compare this with their sleep quality and productivity. With a major difficulty of university students being the ability to stay productive and maintain a good sleep schedule, as well as being able to find a work-life balance, we aim to allow our users to specifically compare how productively they spent their day, as well as how many hours they slept, with their overall rating of the day, in order to reflect on how each variable affected the other. Apart from collecting data on our targeted users through surveys, we decided on such a topic through our own research concerning university students, where we commonly found that the majority of students suffer from poor sleep quality, which places them under risk of developing mental health issues while affecting their studies and enjoyment through the day (Foulkes, 2019). As such, we wanted to allow users to explore the relationships between their enjoyment through the day and how many hours of sleep they had during the night. With productivity also being a key part of a students' day, we also wanted our users to explore this relationship too.

Agile Software Process Planning and Management

In this coursework, we were required to follow an Agile Lifecycle for the development of our software. The Agile Alliance defines agile software development as a set of frameworks and practices which follow the Manifesto for Agile Software Development and its 12 principles (Agile Alliance, n.d). This approach is based on iterative development, whereby the project scope and requirements are laid out at the beginning of the development, and then a team of software developers work on creating the software that fulfils the requirements through a certain number of sprints. These sprints last a few weeks each, and consist of developers choosing software requirements to try to meet by the end of each sprint.

As a part of creating our PI system, we ensured to follow an agile process through four sprints in total, steadily developing and evolving our system. At the beginning of each sprint, we selected from the product backlog, the features of the highest priority that were essential to the system, and assigned each member the task of fulfilling one of these requirements. Although at the end of each sprint there should be a testable product, our first sprint lasted a week in length, and was used to create concrete GUI and class designs for our system, such that we knew exactly how our system would work, and how each of the classes would react with each other. The other three sprints followed the normal routine, where we were able to test our implemented features by the end of each sprint.

During each sprint we held scrum meetings on every Monday and Friday, used to help manage the progress within each sprint and handle any changes we needed to make to our system. Our first meeting of each sprint was our sprint planning meeting, and we ended each sprint with our sprint review. We can take our second sprint as an example. Our second sprint was 2 weeks long. In our sprint planning meeting we concretely decided what features we wanted to work on. During this meeting, we decided that we would tackle the coding of the GUI, and the implementation of the code to allow the user to enter, and store their data on their system using text files. Effectively, we decided that we would handle the 'Viewing and Collecting Tracking Data' section of our requirements specification. So, from the product backlog we decided on tackling features such as the user being able to enter a productivity rating for their day and have this stored in a file on their system (along with the other PI variables we were tracking). Then, during the sprint, we held scrum meetings to formally catch up on each other's progress, discussing any difficulties we were facing and working together to attempt to tackle them.

At the end of our sprints, we held our sprint reviews. For the second sprint, this was held on March 27th, where we discussed the functionality accomplished. During our second sprint, we realised that the order in which the user entered their PI data would impact how it was stored in the text file, and that this inconsistency could lead to

problems. Furthermore, this realisation meant that allowing the user to enter and store their data properly could not be finished during the sprint. However, in our sprint review we noted that this, although unfinished, could be tackled in the third sprint and would not be harmful to the further development of our software. Other features, such as plotting graphs of the PI variables against time, could be implemented and tested against test data in the correct format, while this issue was sorted. To plan, track and keep in continuous contact during sprints we mainly used Trello and Discord outside of the meetings held on Microsoft Teams. Below reveals a screenshot of our Trello page during sprint 3, which lasted between 29th March-9th April. (In this sprint, we focused on implementing the 'Goals and Achievements' as well as the 'Managing Tasks' functionality.)

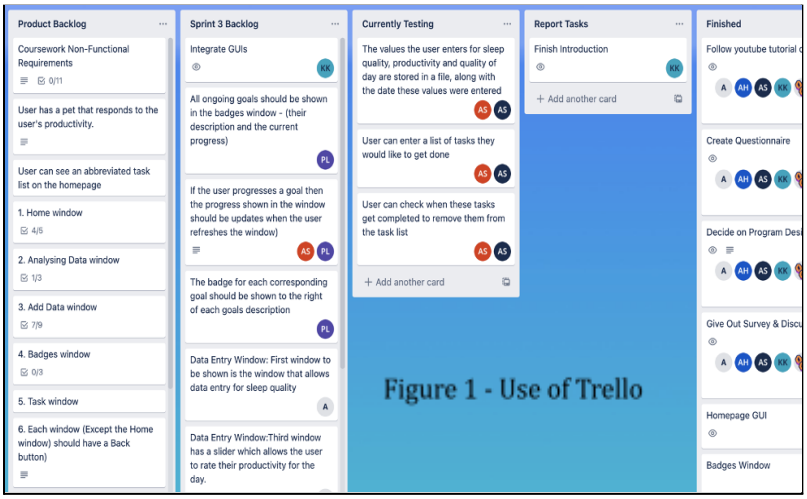


Figure 1 - Use of Trello

We used Trello to note the features of our entire product backlog, as well as the features to be tackled during each sprint. For example, in sprint 3 one of the features included was that when the user wanted to enter their PI data, the first window shown to them asked the user for the data on their sleep quality - this was added because of the realisation in sprint 2 that the order in which the user entered their data was important in our system due to how we wanted the data to be stored. We also used this sprint to tackle features to do with the 'Task Management' and 'Goals and Achievements' section of our software specification requirement.

Continuous testing was also an important part of each sprint, whereby at the end of sprints 2, 3 and 4 we had testable functionality. For example, during sprint 4, which lasted from the 12th-19th April, we tackled the implementation of the graph functionality for our system. At the end of the sprint, we tested the graphing feature against our test table to determine the success of the feature.

Figure 2 reveals a visual breakdown of the amount of work tackled during each sprint, and shows how successful we were in meeting each sprint's goals. As expected, there were cases where we fell behind due to new challenges that appeared in sprints, most notably in sprint 2. Overall, we followed an agile process through the creation of our requirements backlog, which we used to develop a sprint backlog for each sprint. Scrum meetings were used to manage the evolution of our system, and to ensure that we remained on target to produce a testable system that could track the user's sleep, productivity and quality of day.

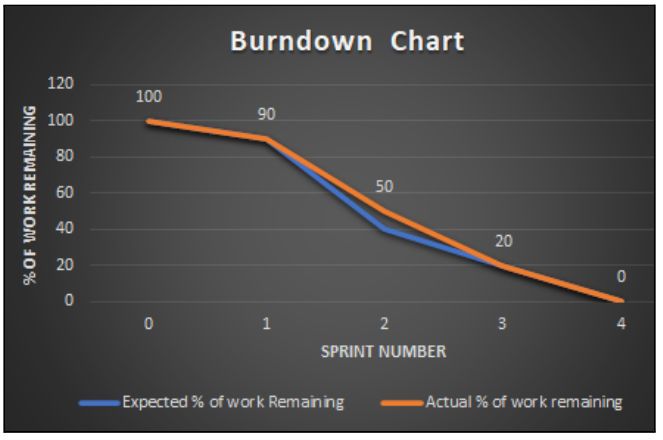
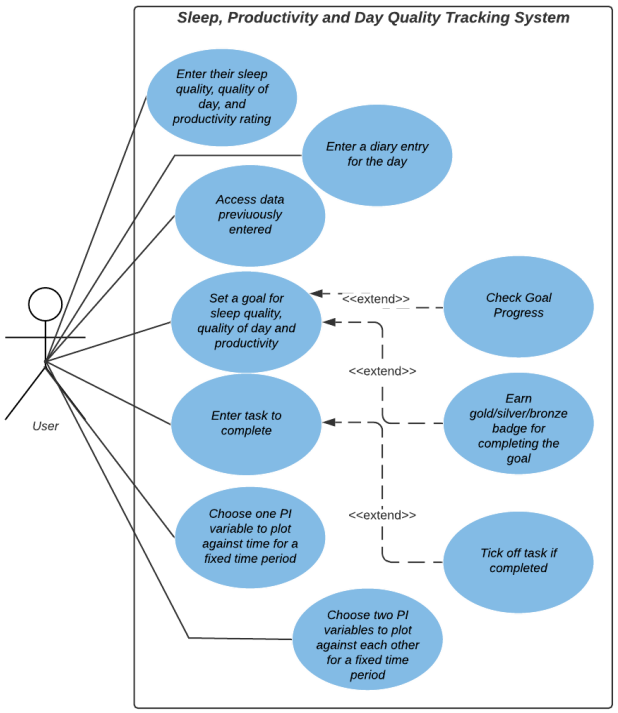


Figure 2 - Burndown Chart

Software Requirements Specification

At first, we analysed the software requirements given in the coursework specifications and we discussed our initial thoughts. One of our main concerns was what data our PI System would be tracking and whether relationships could be extracted from the chosen data. We also decided that our PI System would be one dealing with the user's quality of day. Domain research was then carried out by each of us and we successfully identified the following variables that university students would consider important to track: Amount of work done, hours of sleep, amount of exercise, amount of socialising and nutrition.

Following these discussions, we sent out a survey to gather more information and the respondents were asked to rate the significance of the above mentioned variables in determining their quality of day. Taking into account the survey's 43 responses, we narrowed down to productivity, hours of sleep and quality of day as the three parameters our PI System would track.



Subsequently, members of the group brainstormed several features that could be included in the software based on the three variables we decided to track and the theme of our system. Using these ideas, we established our specifications requirements, classifying them into functional and non-functional ones, which can be found in the table below. Figure 1 is a use case Diagram which shows the interactions between the user and the system's features.

Figure 3: Use Case Diagram

Figure 1 shows the features of our PI System, which has been designed considering university students as our target users. Our system requires data entry at only one point in the day which suits the busy lifestyles of most students. Ratings of their day and productivity are done on a scale of 1-10, which is a fairly easy and quick way to enter data. The user can also view their badges' progress and also the graphs showing the relationship between any two chosen tracked parameters. Our personal informatics system, being simplistic and easy to use, makes it fitting for university students.

Functional Requirements		
1. Viewing and Collecting Tracking Data		
1.1	**Requirement Name:** Storing User's Personal Data	Author: Group 10 Priority: HIGH Dependencies: None Source: Coursework Specification
	Description: The system must be able to store data on the three main variables for each day: quality of day, sleep quality and productivity. This must be stored in a file on the user's system where each line represents the data stored for the day which the user entered. On each line, the data in the file must be stored in the following format: 'Date:22/03/21 Sleep:8 Productivity:7 QoD:8'	
1.2	**Requirement Name:** Collecting Personal Data	Author: Group 10 Priority: HIGH Dependencies: None Source: Coursework Specification
	Description: The user must be able to manually enter data on their sleep quality, productivity and day quality for each day. Each day, the user must be able to rate their productivity and quality of day out of 10. For sleep quality, the user must be able to enter the amount of hours they slept.	
1.3	**Requirement Name:** Access to Data	Author: Group 10 Priority: HIGH Dependencies: 1.1 Source: Coursework Specification
	Description: The user must have access to the files that store their personal data.	
1.4	**Requirement Name:** Diary Entry	Author: Group 10 Priority: MEDIUM Dependencies: 1.2 Source: Group 10
	Description: The user should be able to enter a short diary entry in the same window where they rate the quality of their day.	
2. Identifying Trends and Relationships in PI Data		
2.1	**Requirement Name:** Comparing Data over Time	Author: Group 10 Priority: HIGH Dependencies: 1.1, 1.2 Source: Coursework Specification
	Description: The user must be able to compare their data over time. The user must be able to navigate to a window where they can select a variable out of sleep quality, productivity, quality of day to compare over a specific time period that they also choose. This must then be plotted on a graph to be shown to the user. For example, the user may want to see a graph of their productivity rating against time for the past week.	
2.2	**Requirement Name:** Comparing Different PI Data	Author: Group 10 Priority: MEDIUM

	Description: The user should be able to compare different kinds of PI data within a fixed period of time. The user should be able to select two PI variables (such as sleep quality vs productivity) to plot a graph of these two variables, for a fixed period of time that they also should be able to choose, such as a week.	Dependencies: 1.1, 1.2 Source: Coursework Specification
3. Goals and Achievements		
3.1	**Requirement Name:** Goal Creation	Author: Group 10 Priority: HIGH Dependencies: 3.4 Source: Coursework Specification
	Description: The user must be allowed to set a weekly goal for each data variable each week. For example, the user must be allowed to enter that they would like to have an average productivity rating of at least 7 for the week.	
3.2	**Requirement Name:** Goal Updating	Author: Group 10 Priority: LOW Dependencies: 3.1 Source: Coursework Specification
	Description: The user should be able to change the weekly goal they set for the week.	
3.3	**Requirement Name:** Managing Goals	Author: Group 10 Priority: High Dependencies: 3.1 Source: Coursework Specification
	Description: The user must be able to see the progress they are making with regards to their goals set through a progress bar. Depending on the number of days the user has achieved his goals, a percentage will be calculated and the corresponding progress bar will be displayed. For example, if the goal of the user is to sleep at least 8 hours every night and he has done so for 3 days, then a progress bar showing 42.9% (obtained from 3/7 * 100)will be shown.	
3.4	**Requirement Name:** Badges	Author: Group 10 Priority: HIGH Dependencies: 3.1
	Description: The user must receive either a bronze, silver or gold badge depending on the number of days they complete their goal. These badges grades, bronze, silver and gold correspond to the user fulfilling his goals for 7, 30 and 60 days respectively. The colour of the badge grade is indicative of the level the user is currently at for each category(Sleep, Productivity and Quality of Day). When the user has achieved the gold badge, he goes back to the bronze level for that particular category.	
3.5	**Requirement Name**: Viewing Badges Collected	Author: Group 10 Priority: MEDIUM Dependencies: Source: Group 10
	The user should be able to see the amount of badges they have collected thus far in using the system. The number of badges obtained will be displayed under the “Badges Awarded” section for each level, with the colour of the label indicating the grade.	

4. Task Management		
4.1	**Requirement Name**: Managing Tasks	Author: Group 10 Priority: MEDIUM Dependencies: None Source: Group 10
	The user should be able to navigate to a task window, where they can type in tasks that they would like to complete, which should be shown as a list (an abbreviated list should also be shown on the homepage of the system). They should be able to check tasks off when they have completed them, upon which they should disappear from the task list.	

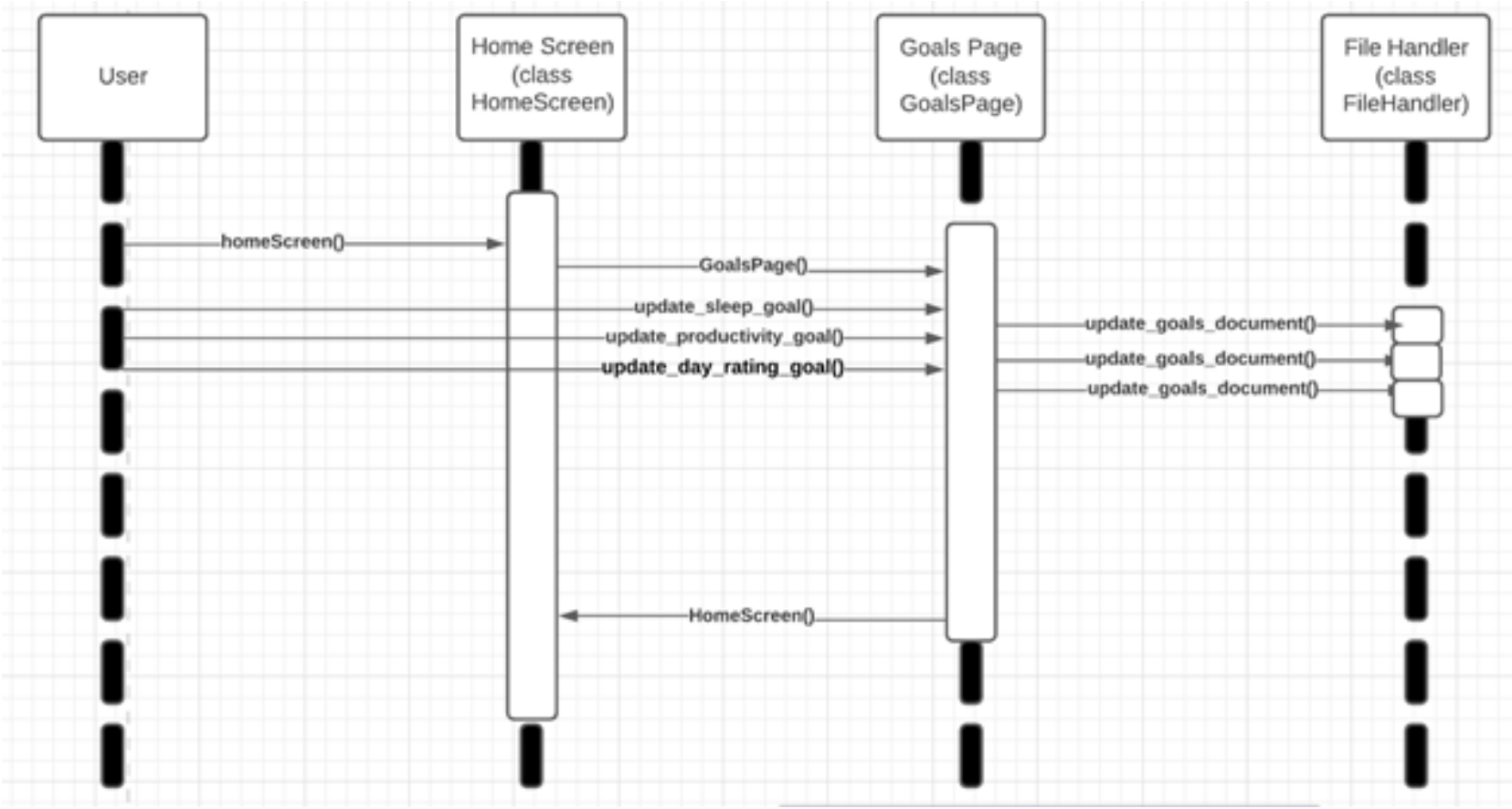
Non-Functional Requirements		
1. Software Development Process		
1.1	**Requirement Name:** Following Scrum Methodologies	Author: SJ Priority: HIGH Source: Coursework Specification
	Description: The software process must be consistent with Scrum methodology.	
1.2	**Requirement Name:** Sprints Number	Author: SJ Priority: MEDIUM Source: Coursework Specification
	Description: Software development should include at least three sprints.	
1.3	**Requirement Name:** Sprint Length	Author: LW Priority: MEDIUM Dependencies: 1.2 Source: Coursework Specification
	Description: Each sprint should last between 1 and 3 weeks.	
1.4	**Requirement Name:** Requirement Review	Author: SJ Priority: High Source: Coursework Specification
	Description: Must regularly review the functional requirements associated with the system.	
2. Expanding Initial Requirements		
2.1	**Requirement Name:** Expanding on Initial Requirements	Author: LW Priority: HIGH Source: Coursework Specification
	Description: Must expand upon all initial requirements, based on PI research.	
2.2	**Requirement Name:** Expanding on initial Functional Requirements	Author: LW Priority: HIGH Source: Coursework Specification
	Must expand on initial functional requirements set in the coursework specification adding additional functionality to the system to deliver the features.	

2.2	**Requirement Name:** Additional Requirement Gathering	Author: LW Priority: HIGH Source: Coursework Specification
	Additional requirements must be gathered through appropriate techniques, such as holding interviews or carrying out surveys.	
3. Background Research		
3.1	**Requirement Name:** Cite 3 Articles	Author: SJ Priority: HIGH Source: Coursework Specification
	Must read and cite at least three articles in the area of PI, at least one of which must be drawn from the reference section of this coursework document.	
3.2	**Requirement Name:** Cite 6 Articles	Author: LW Priority: HISource: Coursework Specification
	Should read and cite at least six 6 articles of any kind.	
4. Testing		
4.1	**Requirement Name:** Test-Driven Development	Author: LW Priority: HIGH Source: Coursework Specification
	Must adopt a test-driven development approach, ensuring the system is being tested in each sprint, including the production of test plans.	
4.2	**Requirement Name:** Testing Evidence	Author: SJ Priority: HIGH Source: Coursework Specification
	Must provide evidence of testing in some way, such as through test tables.	
5. System Characteristics		
5.1	**Requirement Name:** System Response Time	Author: LWPriority: HIGH Source: Coursework Specification
	The system must have a quick response time to registering user input, responding within milliseconds..	
5.2	**Requirement Name:** Compatibility	Author: SJ Priority: HIGH Dependencies: None Source: Coursework Specification
	The system must be able to run on multiple systems: Windows, macOS and Linux.	

Design

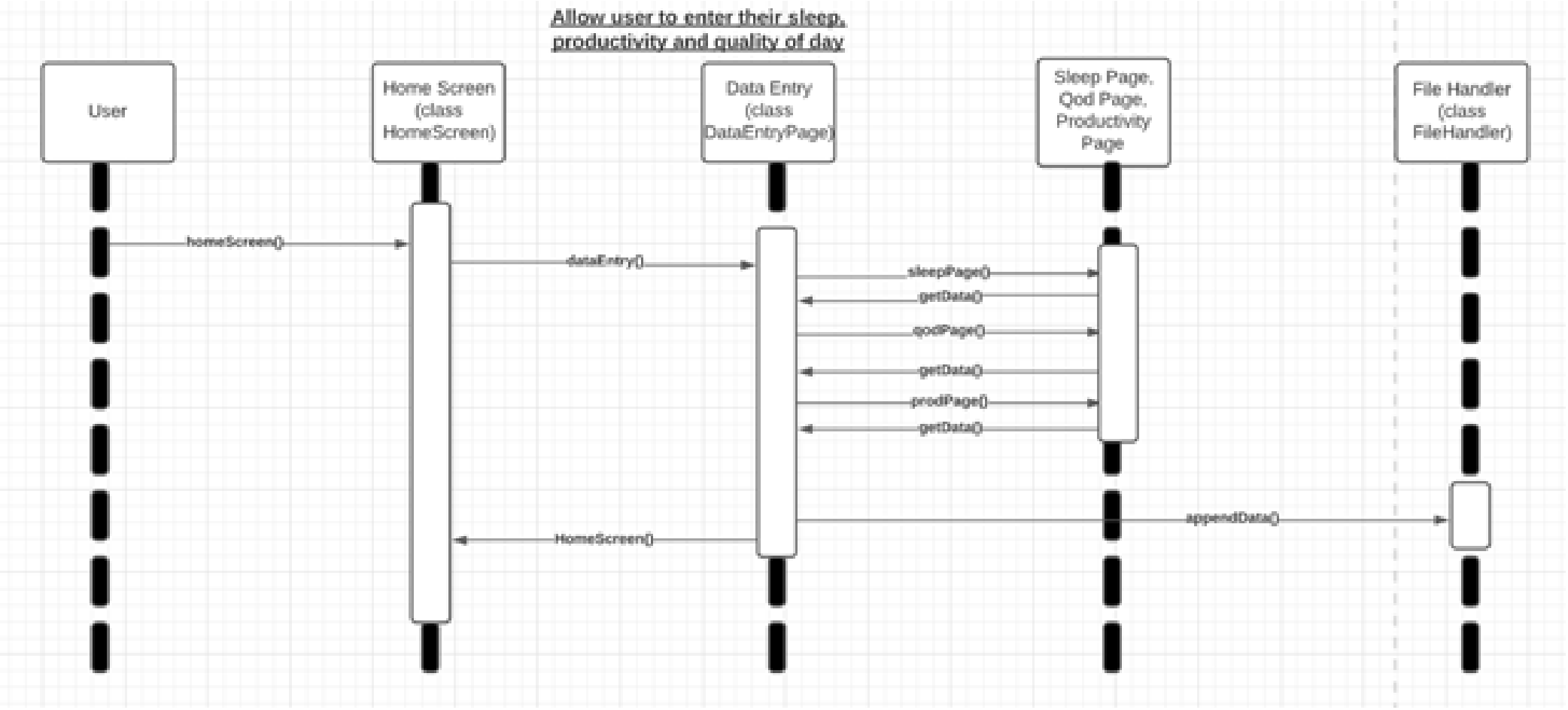
UML Sequence Diagrams

Diagram 1 - Goal Management



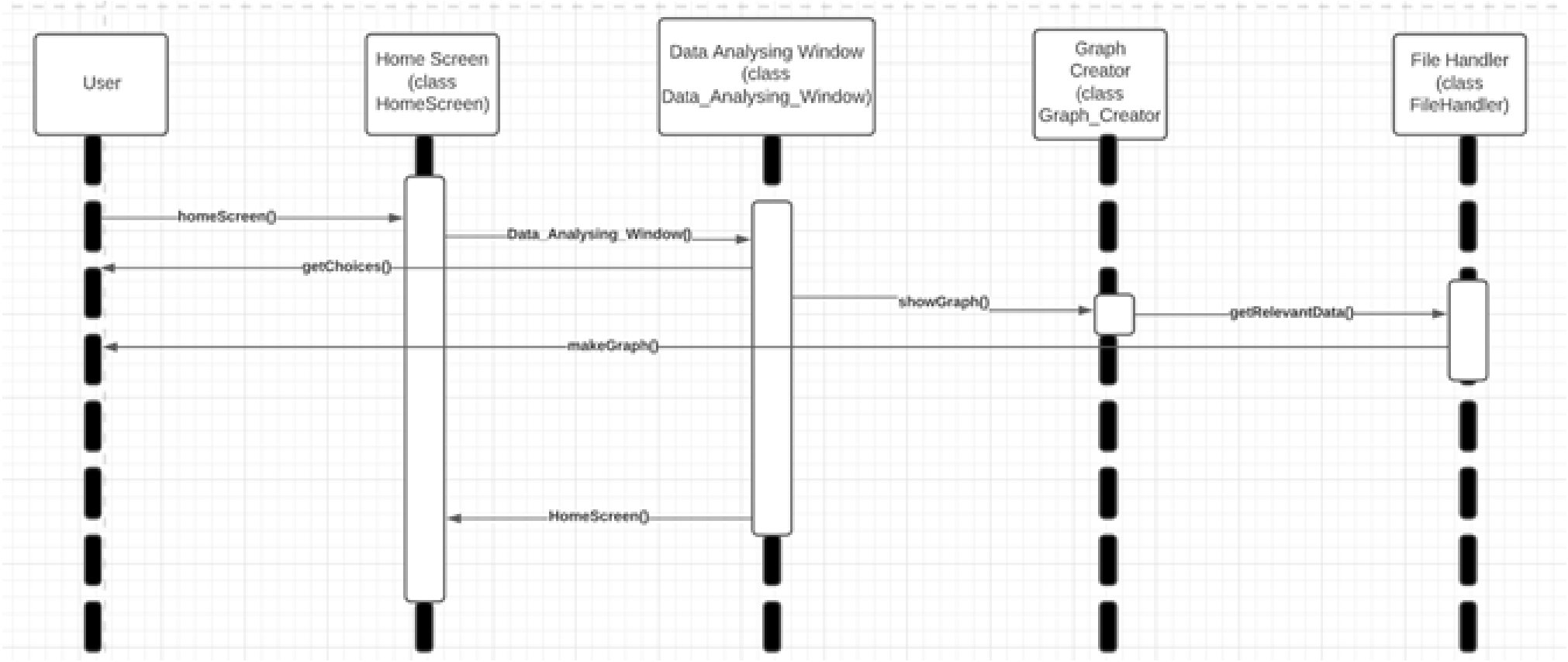
This diagram shows how the user can enter their goals, which are then stored in a document, and the user can then return to the home screen to change their goals if need be. This design specifically tackles 3.1 in the functional requirements: goal creation.

Diagram 2 - Data Entry



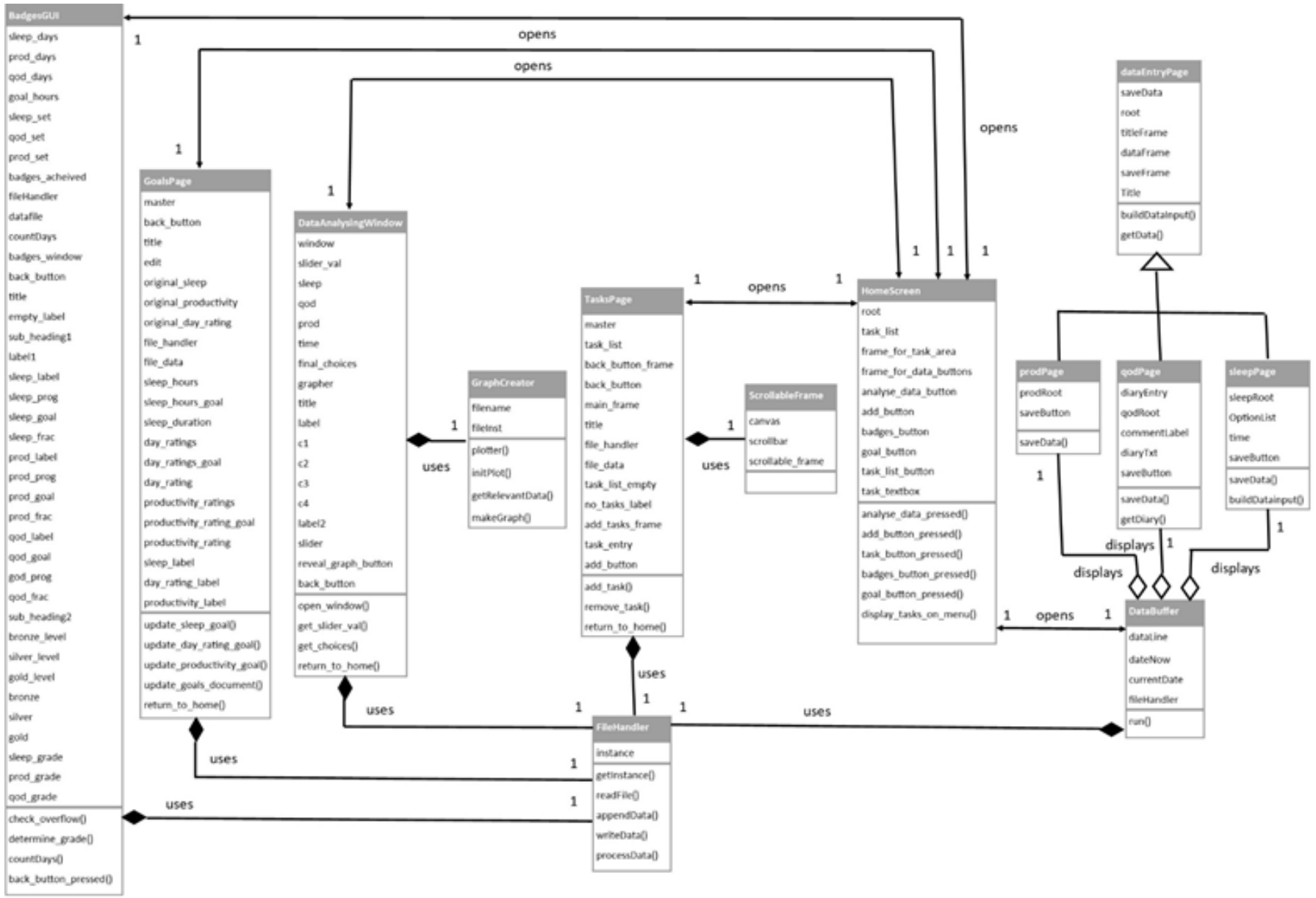
This diagram shows how the user can enter their sleep time, as well as productivity and quality of day ratings into the PI system, tackling 1.1, 1.2 and 1.3 of the functional requirements. This date is then stored in a file and the user can return to the home screen.

Diagram 3 - Correlation Graph



This diagram shows how the user can choose which variables they would like to be compared and then a graph is shown to them which shows the relationship between these variables. Here we are dealing with section 2 (Identifying Trends and Relationships in PI Data) of the requirements specification.

Class Diagram



Class Diagram

This Class diagram on the previous page shows the relationships between the classes involved in our PI system. The classes BadgesGUI, GoalPage, DataAnalysingWindow, DataBuffer and TaskPage are all associated with the class Homescreen in both directions. The HomeScreen opens each of the pages controlled by these classes and when each of those pages are exited the class opens HomeScreen. These classes are also comprised of FileHandler. DataAnalysingWindow is comprised of GraphCreator as it uses it to build the graphs that it displays and TaskPage is comprised of ScrollableFrame. Finally the DataBuffer is associated with prodPage, qodPage and sleepPage through aggregation as it cycles through each one to display them in order. These three pages inherit from dataEntryPage.

Design justification

Our design meets the requirement of allowing the user to enter a goal for each day/week for each variable our P.I system tracks(sleep, productivity and quality of day) which are then stored in a document that can be accessed by other areas of the program. This is shown in diagram 1 of the UML sequence diagrams.

Our second requirement is allowing the user to enter the quality of day, productivity or sleep on each day they use the program, the design of the program allows each of these variables to be updated by the user and the data is then added to a file where it can be retrieved later on. This is shown in diagram 2 of the UML sequence diagrams.

Another requirement of the program was that it can create a graph showing the user the correlation between two variables of their choice, the program does this by collecting the user's choices of variables and then passing these to the class responsible for creating the graph. The user is then shown the graph and after, can return to the home screen. This is shown in diagram 3 of the UML sequence diagram.

Software Testing

Testing remained a continuous process in the development of our system. Most of our tests passed quickly as a lot of them were visual confirmations of what was on the board. However, a few tests required more effort to solve with multiple ideas being put on the table and needing research. (The full test table showing all of our test results can be found in Appendix D).

Test Cases 3.16

3. Add Data windows	3.16) Verify that if the user does not complete all the windows then no data is added to the corresponding text file	1.1, 1.2		Go through all the windows and don't complete at least one of them, repeat the process until all three windows have not been completed at least once	If any window is not completed the rest should still show up like normal. But when the last window is completed (or not completed) and the user goes back to the Home window, no new data should be added to the corresponding text file	If any window is not completed the rest still show up like normal. But when the user goes back to the Home window, no new data is added to the corresponding text file	Pass

This test makes sure that the user adds all the data the system needs to function and does not skip even one piece of data (as this will cause a crash if certain functions of the program are run). At first it was difficult to figure out how we should check for this as the user could not complete any one of the three windows where they add data. We also had to decide what would happen if the user doesn't add data, does the user still complete the other pages or are they redirected back to the Home window. In the end, through a few meetings and breaking down the different situations that would result in this test case failing we decided that it would be best if no data is added to the corresponding text file until the user completes (or does not complete) the last window and is redirected back to the Home window. This allows us to do all our checking in one go and not have to worry about adding new redirecting functionality.

Test Case 5.1:

5.Task window	5.1) Verify that the Tasks window has all the current tasks showing	4.1		Show the Task window	After showing the Task window, all the current tasks should be shown in a list type order	After showing the Task window, all the current tasks are shown in a list type order	Pass

This test verifies that all the user's tasks for the day are shown on the window. This is so the user can see the tasks they have still to do or check them to signify the task has been completed. The problem with this is that the window the user is shown is a fixed size whereas how many tasks there are to show is unknown. At first we talked about showing a fixed number of tasks with buttons to show the next set and previous set of tasks. But after some research by the team that made the Task window we found a way to show any number of tasks using a scrollable frame. The scrollable frame does not require extra buttons and results in a much cleaner look to the GUI.

Test Case 5.9:

5.Task window	5.9) Verify that if the "Add task" button is clicked but the text box to the left is empty the user should be shown an error message telling them to add a task description and the same thing should happen if the user tries to enter duplicate tasks	4.1		Show the Task window and click the "Add task" button	After showing the Task window and clicking the "Add task" button the user should get an error message telling them that they have to add a task description before they can add the task to the day's list and the same thing should happen if the user tries to enter duplicate tasks	After showing the Task window and clicking the "Add task" button the user gets an error message saying that they have to add a task description before they can add the task to the day's list and the same thing happened when the user tried to enter a duplicate task	Pass

This test was not initially in the table until after the testing had begun. The testing team during their test of the Task window thought of this test whilst performing test case 5.5 and 5.6. As these tests included the text box where the user can enter their task description, the team realised that there is no verification that a task description is not empty or if the task already exists when the user adds a task. This was a quick fix but when the bug was found we were under time pressure to finish, so rather than one person fixing the verification for duplicate and empty tasks both we decided that one person would fix the verification for duplicate tasks and another for empty tasks. This split of the work allowed us to fix the bug and test it thoroughly in time.

Reflection and Conclusion

Personal informatics systems have had a huge surge in popularity recently, and much research has been completed to identify the optimal ways to track personal user data and provide them with valuable feedback. Our software implantation focused on three key elements of a user's day. The first was the time they spent asleep, the second and third were their ratings of their productivity throughout the day and of their thoughts on the overall quality of their day as a whole respectively.

Our target audience being university students, we conducted research through questionnaires on students in our year group. Results showed that out of 43 responses, the modal day rating out of 5 was a 3, and 84% of students agreed they would possibly use a PI system to help track and improve their quality of day. We also found that 70% of students wanted to spend more time studying. As a team we were curious about the relationships between these two factors and how they related to sleep, so considering this research and other independent research we decided tracking these three factors would be our core aim throughout this project.

In hindsight, this research method had certain flaws. Firstly, in the spirit of keeping the questionnaires concise for our stakeholder, their answers often lacked detail. For example, when asking students to give a rating out of five in answer to this question, "how would you rate the productivity of your day today?'", we did not follow up on why this is the case. Clearly, different students would have different ideas on how to classify how productive they have been. For example, they might consider the time spent watching lectures, time spent answering practice questions, and time spent working out, as well as others in their final answer. We could have achieved a more comprehensive response had we instead chosen to perform an interview with a range of students, as well as our original questionnaire.

Following our research, we started laying out our product backlog. We did a good job of identifying the core requirements of our software, and noting them down in a basic, yet understandable way. We also discussed priorities to requirements, for example we placed a high priority on having a home screen window, but a lower priority on an interactive pet on this said window (that we indeed did not complete due to time constraints). This helped us stay organized and focused throughout our sprints and ensured we had fully functional software, with the most important features, at the end of the process.

Our agile process had four well-structured sprints in total. At the start of each sprint, we would hold a sprint planning meeting, to discuss what we would tackle and designate tasks to individuals, then we would start a sprint. All but the first sprint were two weeks long, and sprint meetings were held twice a week, to keep up to date with how everyone

was getting along, and any problems that were faced. We all found this approach to group-work to be very convenient, and as individuals could choose what tasks they preferred to work on, we had only a small number of queries throughout. At the end of each sprint, we would review the sprint, demonstrate working software and briefly think about the next one.

Although at the end of each sprint we should have had a testable product, our first sprint lasted a week, and was used to create concrete GUI and class designs for our system, so that we knew exactly how our system would work and how classes interact. Though this resulted in some very good ideas by group members, it did not follow agile principles and could possibly have been coordinated in a more optimal manner, for example being discussed in an extended group meeting, instead of being dedicated to a full sprint. However, being our very first sprint, it helped to give the team an idea of the length of time certain tasks required and helped us to think about how we could split our product backlog into manageable sprints in the future.

Sprint two was dedicated to the coding of the GUI designs of the system and implementing the code to allow the user to enter and store their data on their system. This was a very well-coordinated sprint, and although individuals were tasked with learning a new Python GUI package – Tkniter, as well as designing separate GUIs, everyone coped very well, and we had a fully working separate GUIs for each window at the end of the sprint. Even outside of meetings, we kept the team up to date with our designs by posting pictures on Discord, and modifying them appropriately based on feedback.

In sprint 3 we agreed that our software should feel more personal to the user. Although sensitive data is being tracked, the user should feel a certain closeness to the software. We had already thought about having an interactive pet, so we decided upon allowing the user to add an optional diary entry, after giving their rating for the day. They could then, for example weeks later view a certain high rated day, and read through some of their personal thoughts at the time. We also found some small bugs in our code, and an issue in the data handling of the text files where we were storing our data. We added the requirement to fix these issues as well as to implement our new diary feature to the already populated sprint backlog and were then ready to start. Again, the sprint was focussed and well-handled by all team members.

Sprint four was slightly less productive than the previous two. We managed to get graph functionality working, and the GUI as whole to run seamlessly as a single piece of software. Although we had discussed developing an interactive pet for the home screen, we decided it should not be prioritized as highly as the graph functionality. Also, we prioritized some new tasks we thought up in the sprint plan such as using a consistent font and text size throughout our software, to achieve a more appealing design. Sprint four was our last sprint in this coursework, but should we have continued, we would

have commenced work on the pet feature in sprint five, along with other features, and new ideas such as some mentioned below.

Looking back on the features and functionality on the system's requirements specification, we have created, what can be described as, a very solid back-bone. To clarify, we designed and made a program, which theoretically could be published as a mobile application and fulfill its core requirements. However, there are a lot more things we could add, and had discussed doing so but eventually neglecting to do so for time reasons. Everything we have implemented is what makes this app functional and this would allow us in the future, to build more features on top.

A good example would be further customization. At the moment, we have a fairly bare bones GUI, which is completely functional, allowing the user to easily navigate to different sections of the program. It is, however, not a very interesting program to look at. We could add customization options that would enable the user to change the background with pre-set ones such as solid colors, stock images or even allow them to upload ones themselves. We could go further and allow them to personalize the look and layout of the buttons, allowing the program to completely cater to the users wants and needs, which is important in a Personal Informatics app, as this is an app they will be using on a daily basis.

Another feature we could add is a more interactive/engaging reward and tracking system. We have the ability to analyse data through the badges, show graphs and keep track of tasks and goals, but again, it is a fairly bare bones system. Something we could implement is a feature that pulls all this information together, making understanding patterns, and how well users are keeping to targets and healthy goals, more fun and interactive. One idea is to have a graphical pet on the front page, which represents all these metrics. If users were keeping to their goals, the pet would be visually happy; and if users were recently sleep-deprived, it would be looking tired and sad. If they had not done enough tasks and had low productivity it would be visibly stressed, along with more ideas. This would be done in a way to try and encourage the user to stick to good habits and take care of their pet. The badges could translate into currency, allowing the user to buy things like snacks, clothing and other accessories. This would just be one way that we could nicely wrap up all the information into a nice looking front page, which easily conveys all of it and encourages the user to be active with the app and aim for better habits.

When it comes to the design of our system, we planned it out very thoroughly and ended up with a very well made system. Considering the GUI, apart from what has already been mentioned, it serves its current objective perfectly, allowing clear use and easy navigation throughout the program. The design of the implementation was that, for each window, which would be arrived at through buttons, the current window would be destroyed and the new one constructed. This allowed each window to have up to date

information from different areas of the program, and not be too dissimilar to what would happen on a phone. The only slightly jaring place is the graph window. When this is created, it creates an entirely new window with just the graph on top of the current GUI, which would not translate well into an app. Realistically, it needs to appear within the same window, then be allowed to be overridden by any new requests by the user to change the graph. This would mean changing the implementation, allowing the windows to be dynamically modified whilst open.

Early on we discussed the storage and movement of data, which resulted in a robust system. There is a class that handles everything related to data, such as reading, writing, file handling, formatting etc. There is an agreed upon format for storage, and accessing data is done in a clear way. A key design point here is that no data is passed between different windows. It is always read from a text file, and if needed, the file itself is updated. For different aspects of the program we create different files. This method has worked flawlessly for us, simplifying the whole process. One could argue as our system gets more complicated and the scope grows, using a flat, text file system does not scale well, and a more sophisticated structure like a database would be more suitable.

In terms of our testing, we did a fairly thorough job throughout. Certain criticisms could include the fact that the majority of our tests would be classified as unit or integration tests, and doing more acceptance tests - by getting feedback from our clients would help better the program significantly. However, what we have done ensures that everything we set out to do is functional and working. Another improvement we could have made to our test-driven approach is using the ‘unittest’ module available for python that would automate our tests for us.

To conclude, our group of eight adopted the ‘Scrum’ agile approach to develop software for a Personal Informatics system targeted at current university students. We worked together to research, specify, design, develop and test software that collects three forms of PI data (sleep duration, productivity, and quality of day) from a user, and allows them to view relationships between this data. Our main goal was to provide our users with insightful information about how their productivity and rating of a certain day depended on the sleep they had the night before, and how their productivity relates to their overall day rating. We hoped that users having access to this information could improve certain habitual behaviours, and generally, help them get more out of their lives. We all thoroughly enjoyed working together on this project, and learned a great deal, not only from research, but more importantly from one another, and all look forward to more group projects in the future.

References

1. Agile Alliance, 2019. *What is Agile Software Development?* [Online]. Agile Alliance. Available from: https://www.agilealliance.org/agile101/ [Accessed 2 April 2021].
2. Foulkes, L., 2019. *Majority of university students report poor quality sleep, putting them at higher risk of mental health problems* [Online]. Available from: https://theconversation.com/majority-of-university-students-report-poor-quality-sleep-putting-them-at-higher-risk-of-mental-health-problems-110746#:~:text=In%20fact%2C%20up%20to%2060 [Accessed 2 February 2021].
3. Fritz, T., Huang, E.M., Murphy, G.C. and Zimmermann, T., 2014. Persuasive technology in the real world. *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14* [Online]. Available from: https://doi.org/10.1145/2556288.2557383.
4. Gulotta, R., Forlizzi, J., Yang, R. and Newman, M., 2016. *Fostering Engagement with Personal Informatics Systems* [Online]. Available from: https://mwnewman.people.si.umich.edu/fulltext/dis2016_longtermengage.pdf
5. 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* [Online], 32(5-6), pp.268–296. Available from: https://doi.org/10.1080/07370024.2016.1276456.
6. KOSECKI, D., 2017. *Presenting the Official List of Fitbit Badges. How Many Do You Have?* [Online]. Available from: https://blog.fitbit.com/fitbit-badges/.
7. Lee, M., Kim, J., Forlizzi, J. and Kiesler, S., 2015. Personalization Revisited: A Reflective Approach Helps People Better Personalize Health Services and Motivates Them To Increase Physical Activity. *Personalization Revisited: A Reflective Approach Helps People Better Personalize Health Services and Motivates Them To Increase Physical Activity* [Online]. Available from: https://doi.org/10.1145/2750858.2807552.
8. Li, I., 2012. *Thinking About Side Effects of Personal Informatics Systems* [Online]. Available from:

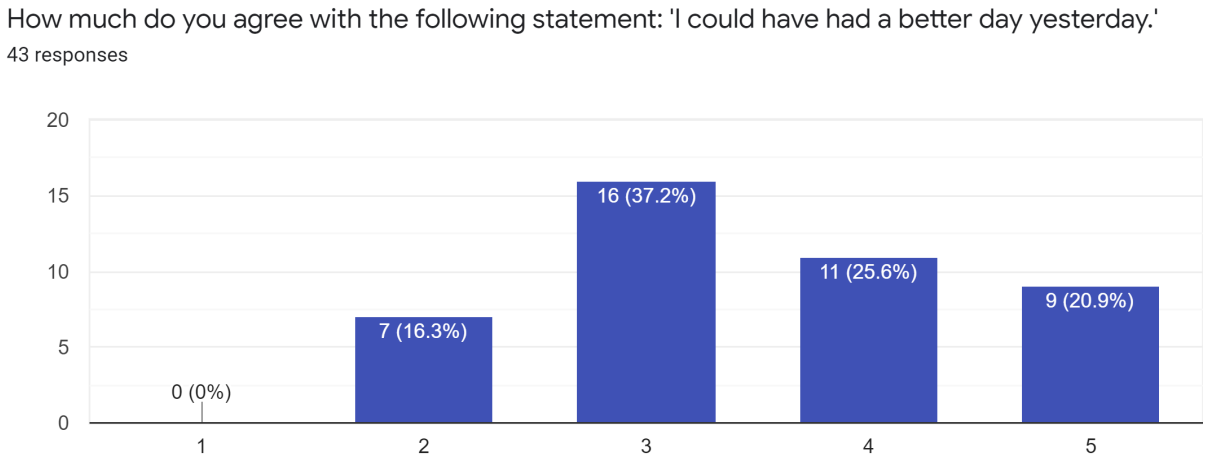
https://quantifiedself.com/blog/thinking-about-side-effects-of-personal-informatics-systems/ [Accessed 1 February 2021].

9. Li, I., Dey, A., Forlizzi, J., Höök, K. and Medynskiy, Y., 2011. Personal informatics and HCI. *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems - CHI EA '11* [Online]. Available from: https://doi.org/10.1145/1979742.1979573
10. Rapp, A., Marcengo, A., Buriano, L., Ruffo, G., Lai, M. and Cena, F., 2018. Designing a personal informatics system for users without experience in self-tracking: a case study. *Behaviour & Information Technology* [Online], 37(4), pp.335–366. Available from: https://doi.org/10.1080/0144929x.2018.1436592.
11. Sleep Cycle AB, 2009. *Sleep Cycle* (6.14.3) [Mobile App] Available from: https://apps.apple.com/us/app/sleep-cycle-sleep-tracker/id320606217 [Accessed 27 February 2021]

Appendix C - Key Questionnaire Responses

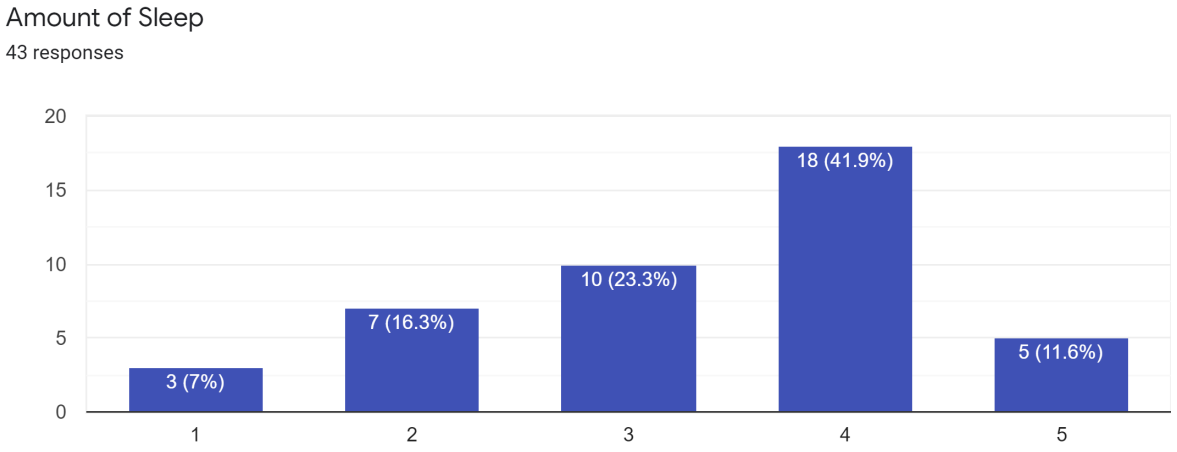
(Full survey can be found at https://docs.google.com/forms/d/1pUDBjH3XKPNG_SlH2ClMTtK8GfUmOWTPbapOYzwFGVA/edit)

1.



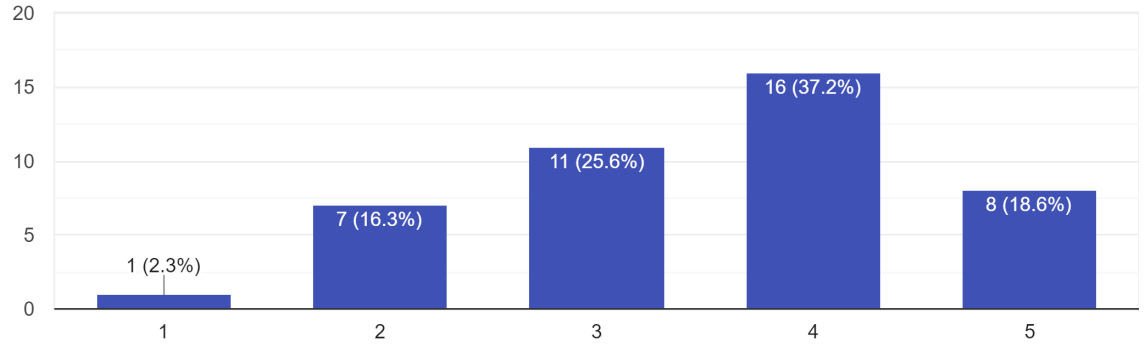
2.

Rate the following in terms of how important they are in determining the quality of your day (5 being most important):



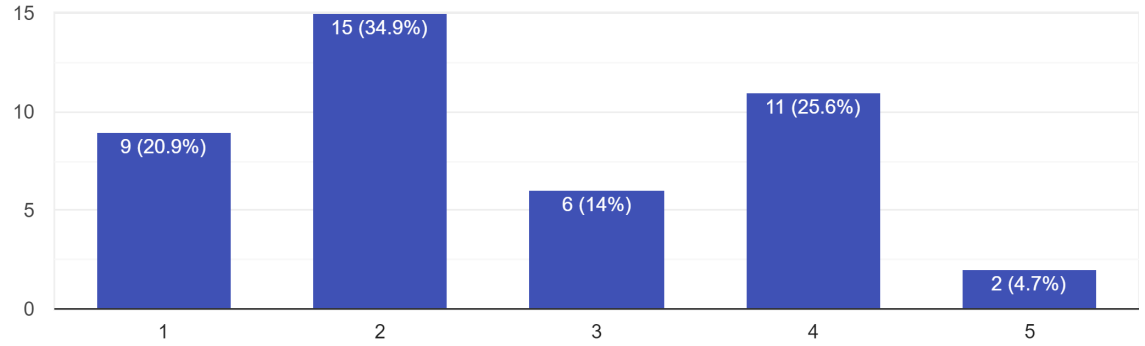
Amount of Work

43 responses



Amount of Exercise

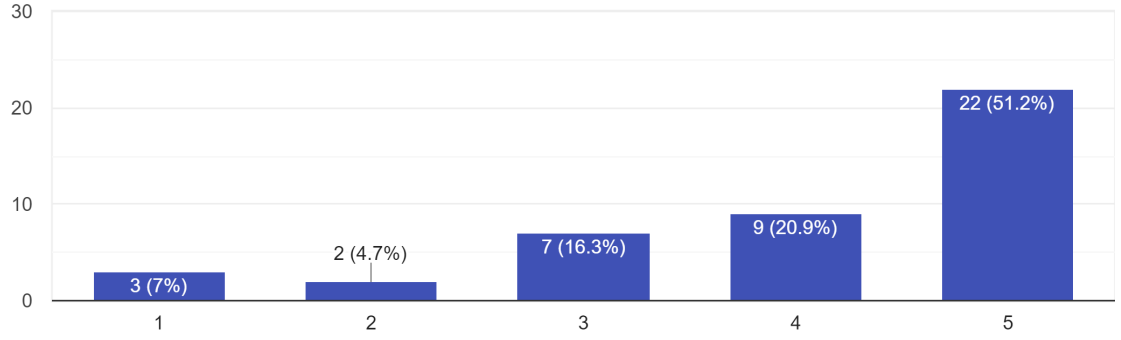
43 responses



3.

How much do you agree with the following statement: 'I could be more productive throughout the day.'

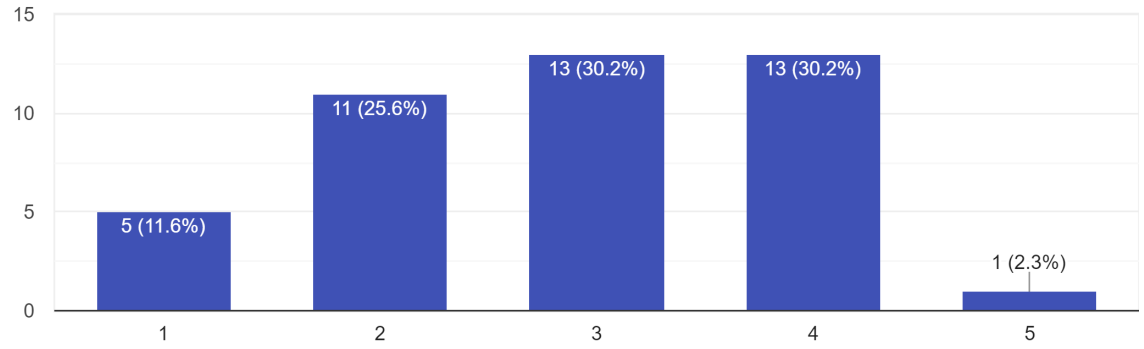
43 responses



4.

How would you rate the productivity of your day today?

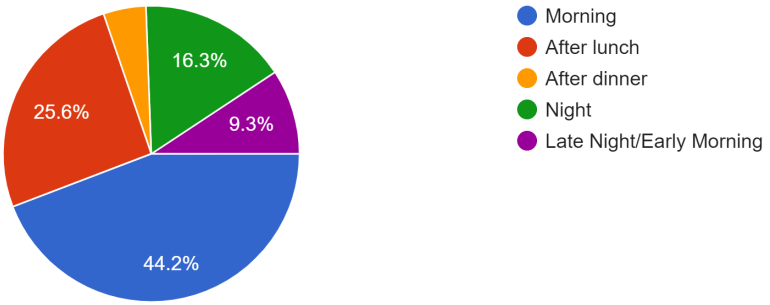
43 responses



5.

Generally, what time of day do you feel most productive?

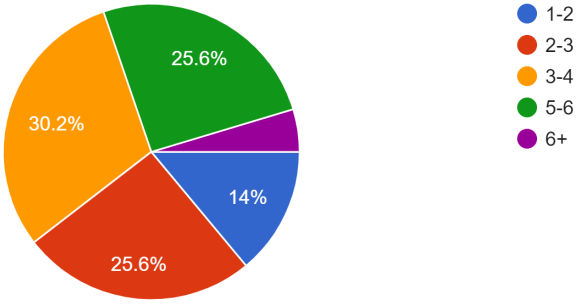
43 responses



6.

How many hours of studying a day do you consider to be productive?

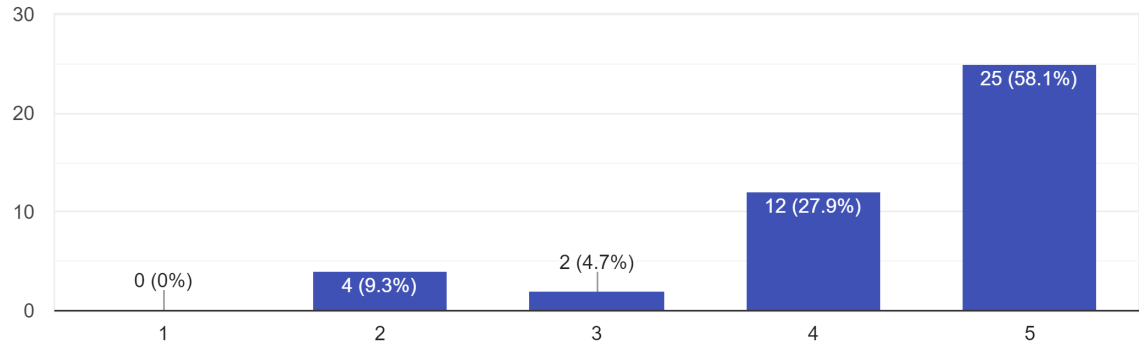
43 responses



7.

How closely do you think sleep is related to productivity?

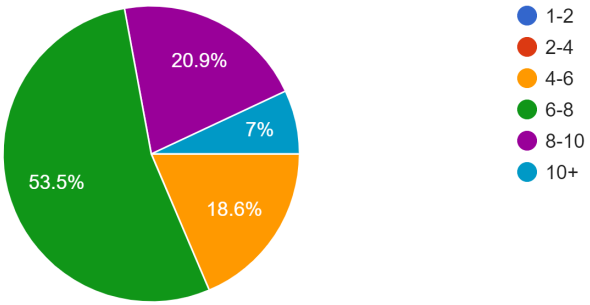
43 responses



8.

How many hours do you sleep, on average, per day?

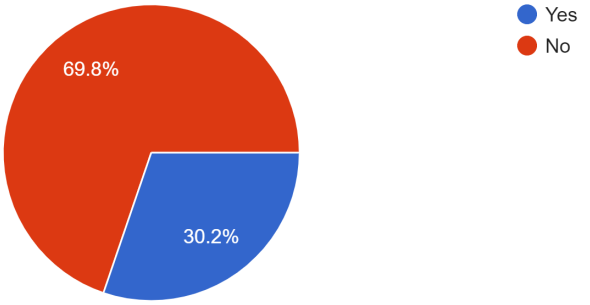
43 responses



9.

Have you ever used a PI System (such as the Fitbit app) to track information about yourself before?

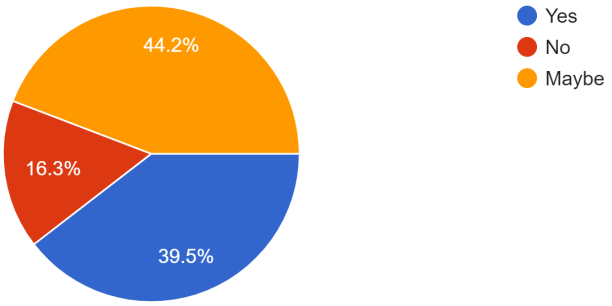
43 responses



10.

Would you consider using a PI system to help track and improve your quality of day?

43 responses



Appendix D - Test Table

Test Scenario and Number	Test Case and Number	Functional Requirements Tested	Pre-Conditions (TC = Test Case)	Test Steps	Expected Result	Actual Result	Pass/ Fail
1. Home window	1.1) Verify that a Task List button is shown			Show the Home window	The “Task list” button is shown at the top of the window	The “Tasks” button is shown at the top of the window	Pass
	1.2) Verify that the Task List button when clicked switches windows to the Task window		TC 1.1	Show the Home window and then click on the “Task” button	When clicked the Home window is replaced with the “Task” window	the Home window is replaced with the “Tasks” window	Pass
	1.3) Verify that an Analyse Data button is shown			Show the Home window	The “Analyse Data” button is shown at the bottom left of the window	The “Analyse Data” button is shown at the bottom left of the window	Pass
	1.4) Verify that the “Analyse Data” button when clicked switches windows to the “Analysing Data” window		TC 1.3	Show the Home window and the click on the “Analyse Data” window	When clicked the Home window is replaced with the “Analyse Data” window	When clicked the Home window is replaced with the “Analyse Data” window	Pass
	1.5) Verify that the “Badges“ button is shown			Show the Home window	The “Badges” button is shown at the bottom right of the window	The “Badges” button is shown at the bottom right of the window	Pass

	1.6) Verify that the “Badges” button when clicked switches to the “Badges” window		TC 1.5	Show the Home window and click on the “Badges” button	When clicked the Home window is replaced with the “Badges” window	When clicked the Home window is replaced with the “Badges” window	Pass
	1.7) Verify that the “Diary Add Button” is shown			Show the Home window	The “Diary Add Button” is shown centred horizontally and just above the “Analysing Data” and “Badges” buttons	The “Diary Add Button” is shown directly above the “Analysing Data” button	Pass
	1.8) Verify that the “Diary Add” button when clicked switches windows to the first “Add Data” window		TC 1.7	Show the Home window and then click on the “Diary Add” button	When clicked the Home window is replaced with the “Add data” window	When clicked the Home window is replaced with the “Data Entry” window	Pass
	1.9) Verify that the day’s tasks are shown in a scrollable frame under the “Tasks” button			Show the Home window	When the Home window is shown an area under the “Tasks” button shows the day’s tasks, and the user is able to use a scroll bar to see all the tasks	When the Home window is shown an area under the “Tasks” button shows the day’s tasks, but the user is not able to use a scroll bar to see all the tasks	Pass

2. Analysing Data window	2.1) The user is able to see 4 checkboxes each labelled “Sleep”, “Quality of Day”, “Productivity” and “Time” respectively	2.1, 2.2		Show the Analysing Data window	When the Analysing Data window is shown the user should see 4 checkboxes in the middle of the screen. The checkboxes should have a label to the right of them saying “Sleep”, “Quality of Day”, “Productivity” and “Time” respectively	When the Analysing Data window is shown the user can see 4 checkboxes in the middle of the screen. The checkboxes should have a label to the right of them saying “Sleep”, “Quality of Day”, “Productivity” and “Time” respectively	Pass
	2.2) The user should be able to click the checkboxes which are the variables to be used for the graphs	2.1, 2.2	TC 2.1	Show the Analysing Data window then click the checkboxes	When the boxes are clicked they should be highlighted which signifies that the user has clicked them, and the program has recorded the input	When the boxes are clicked they should be highlighted which signifies that the user has clicked them, and the program has recorded the input	Pass
	2.3) A slide bar should be shown at the bottom of the screen which shows “0” at the left end and “20” at the right end	2.1		Show the Analysing Data window	When the Analysing Data window is shown the user should see a slide bar (like a horizontal scroll bar) which has “0” at the left end and “20” at the right end	When the Analysing Data window is shown the user can see a slide bar (like a horizontal scroll bar) which has “0” at the left end and “20” at the right end	Pass
	2.4) The slide bar should have a label “Time period (weeks)”	2.1	TC 2.3	Show the Analysing Data window	When the Analysing Data window is shown the user should see a “Time period (weeks)” label	When the Analysing Data window is shown the user can see a “Time period (weeks)” label to the left of the slide bar	Pass

	2.5) A Reveal Graph button is shown at the bottom of the screen	2.1, 2.1		Show the Analysing Data window	When the Analysing Data window is shown a button labelled “Reveal Graph” is shown at the bottom of the screen	When the Analysing Data window is shown a button labelled “Reveal Graph” is shown at the bottom of the screen and is aligned to the left	Pass
	2.6) When the Reveal Graph button is clicked it should show the user a graph which shows the relationship between variables based on the checkboxes the user clicked	2.1, 2.2	TC 2.1, 2.2, 2.5	Show the Analysing Data window and click the Reveal Graph button	When the Reveal Graph button is clicked a graph is displayed which shows the relationship between two variables. The variables will be based on the checkboxes the user clicked on the Analysing Data window	When the Reveal Graph button is clicked a graph is displayed which shows the relationship between two variables. The variables will be based on the checkboxes the user clicked on the Analysing Data window	Pass
	2.7) When the Reveal Graph button is clicked and more than two checkboxes have been clicked, the user should be told that only two checkboxes should be clicked, and a graph should not be displayed	2.1, 2.2	TC 2.1, 2.2, 2.5	Show the Analysing Data window, check three boxes and then click the Reveal Graph button. Repeat with four checkboxes clicked	When the Reveal Graph button is clicked, the user should be told that only two checkboxes should be clicked, and a graph should not be displayed. This should happen with both three checkboxes clicked and four checkboxes clicked	When the Reveal Graph button is clicked, the user should be told that only two checkboxes should be clicked, and a graph is not displayed.	Pass

	2.8) When the Reveal Graph button is clicked and less than two checkboxes have been clicked, the user should be told that two checkboxes should be clicked, and a graph should not be displayed	2.1, 2.2	TC 2.1, 2.2, 2.5	Show the Analysing Data window and click only one checkbox then click the Reveal Graph button	When the Reveal Graph button is clicked the user should be told that they need to click two checkboxes and a graph should not be displayed	When the Reveal Graph button is clicked the user should be told that they need to click two checkboxes and a graph is not displayed	Pass
	2.9) There should be a back button to go back to the Home window			Show the Analysing Data window	When the Analysing Data window is shown a button should be shown that has some label signifying it is the back button	When the Analysing Data window is shown at the top right of the window a button is there which has some label signifying it is the back button	Pass
	2.10) When the Back Button is clicked it should take the user back to the Home window		TC 2.9	Show the Analysing Data window and then click the Back button	When the button is clicked the Home window should replace the Analysing Data window	When the button is clicked the Home window should replace the Analysing Data window	Pass
3. Add Data windows	3.1) Verify that the first window shown after the user clicks the Diary Add button is a window which allows the user to add an entry for Sleep Quality	1.1		Click the Diary Add button on the Home window	When the button is clicked the Home window should be replaced with a window that allows the user to add an entry for Sleep Quality	When the button is clicked the Home window is replaced with a window that allows the user to add an entry for Sleep Quality	Pass

	3.2) Verify that the user is able to use a drop down menu to select a time	1.1, 1.2		Show the first Add Data window	When the first Add Data window is shown the user should see a box with an arrow to the right and after clicking the arrow the user should get values which they can select from. The values should be times	When the first Add Data window is shown the user can see a box with a button to the right and after clicking the arrow the user should get values which they can select from. The values should be times	Pass
	3.3) Verify that the user has two drop down menus aligned vertically on the right side of the first Add Data window	1.1, 1.2	TC 3.2	Show the first Add Data window	When the first Add Data window is shown the user should see two drop down menu stacked on top of each other (with a bit of space in between them) placed on the right side of the window	When the first Add Data window is shown the user can see two drop down menu stacked on top of each other (with a bit of space in between them) placed on the right side of the window	Pass
	3.4) Verify that in the first Add Data window, the first drop down menu is labelled “Start Time:” and the second one is labelled “End Time:”	1.1, 1.2	TC 3.2, 3.3	Show the first Add Data window	When the first Add Data window is shown to the left of the drop down menus they should be labelled “Start Time:” and “End Time:” respectively	When the first Add Data window is shown to the left of the drop down menus they are labelled “Start Time:” and “End Time:” respectively	Pass
	3.5) Verify that at the bottom of the first Add Data window a button labelled “Save” should be shown and it should be centred	1.1		Show the first Add Data window	When the first Add Data window is shown at the bottom of the window a button is shown. It should be centred and have a label “Save”	When the first Add Data window is shown at the bottom of the window a button is shown. It is centred and have a label “Save”	Pass

	3.6) Verify that when the Save button is clicked on the first Add Data window it changes the window to the second Add Data window	1.1	TC 3.5	Show the first Add Data window and click the Save button	When the button is clicked the first Add Data window should be replaced with the second Add Data window	When the button is clicked the first Add Data window is replaced with the second Add Data window	Pass
	3.7) Verify that the third Add Data window has a slider (centred horizontally and vertically). The part of the slider that moves should have a label above it which starts at “1” and ends at “10”	1.1, 1.2		Show the second Add Data window	When the third Add Data window is shown, a slider should be shown which is centred horizontally and vertically. The label above the part of the slider which moves should change as the slider moves. Starting at 1 then ending at 10	When the third Add Data window is shown, a slider is shown which is centred horizontally and vertically. The label above the part of the slider which moves change as the slider moves. Starting at 1 then ending at 10	Pass
	3.8) Verify that above the slider in the third Add Data window there is a label that says, “How do you feel about your day today?”	1.1, 1.2	TC 3.7	Show the second Add Data window	When the third Add Data window is shown above the slider there should be a label that says, “How do you feel about your day today?”	When the third Add Data window is shown above the slider there is a label that says, “How do you feel about your day today?”	Pass

	3.9) Verify that on the bottom half of the third Add Data window has a text box which allows the user to enter data using their keyboard	1.1, 1.2, 1.4		Show the second Add Data window and type something into the text box	When the third Add Data window is shown, on its bottom half there should be a white text box which after the user clicks inside they can type using their keyboard. The user's keypresses should be shown on the text box, so the user knows that their keypresses have been processed	When the third Add Data window is shown, on its bottom half there is a white text box which after the user clicks inside they can type using their keyboard. The user's keypresses are shown on the text box	Pass
	3.10) Verify that on the second Add Data window at the bottom there is a button labelled "Save"	1.1		Show the second Add Data window	When the second Add Data window is shown, at the bottom of the window there should be a button displayed with label "Save"	When the second Add Data window is shown, at the bottom of the window there is a button displayed with label "Save"	Pass
	3.11) Verify that after clicking the Save button on the second Add Data window the window is replaced with the third Add Data window	1.1	TC 3.10	Show the second Add Data window and click on the Save button	After clicking the button the window should be replaced with the third Add Data window	After clicking the button the window is replaced with the third Add Data window	Pass

	3.12) Verify that the second Add Data window has a slider on the bottom half of the window. The part of the slider which moves has a label above it which should start at “1” and end at “10”	1.1, 1.2		Show the third Add Data window	When the second Add Data window is shown, a slider should be shown which is centred horizontally and vertically. The label above the part of the slider which moves should change as the slider moves. Starting at 1 then ending at 10	When the second Add Data window is shown, a slider is shown which is centred horizontally and vertically. The label above the part of the slider which moves changes as the slider moves. Starting at 1 then ending at 10	Pass
	3.13) Verify that the slider in the second Add Data window has a label above it which says, “How do you rate your productivity today?”	1.1, 1.2	TC 3.12	Show the third Add Data window	When the second Add Data window is shown, above the slider there should be a label which says, “How do you rate your productivity today?”	When the second Add Data window is shown, above the slider there should be a label which says, “How do you rate your productivity today?”	Pass
	3.14) Verify that the second Add Data window has a button at the bottom (centred horizontally) with label “Save”	1.1		Show the third Add Data window	When the second Add Data window is shown, at the bottom of the window there should be a button (centred horizontally) which has a label on it which says “Save”	When the second Add Data window is shown, at the bottom of the window there is a button (centred horizontally) which has a label on it which says “Save”	Pass
	3.15) Verify that after clicking the Save button on the third Add Data window the window is replaced with the Home window		TC 3.14	Show the third Add Data window and click on the Save button	After clicking the button the third Add Data window should be replaced with the Home window	After clicking the button the third Add Data window should be replaced with the Home window	Pass

	3.16) Verify that if the user does not complete all the windows then no data is added to the corresponding text file	1.1, 1.2		Go through all the windows and don't complete at least one of them, repeat the process until all three windows have not been completed at least once	If any window is not completed the rest should still show up like normal. But when the last window is completed (or not completed) and the user goes back to the Home window, no new data should be added to the corresponding text file	If any window is not completed the rest still show up like normal. But when the user goes back to the Home window, no new data is added to the corresponding text file	Pass
4. Badges window	4.1) Verify that the Badges window has all the current goals showing	3.4		Show the Badges window with some goals ongoing	When the Badges window is shown the current goals must have their description shown. Each description must be stacked on top of the other with some space underneath each goal for a progress bar	When the Badges window is shown the current goals have their description shown. Each description is stacked on top of the other with some space underneath each goal for a progress bar	Pass
	4.2) Verify that under each goal description there is a progress bar which shows the current progress till completion of the goal	3.4	TC 4.1	Show the Badges window with some goals ongoing	When the Badges window is shown, under each goal description there should be a green progress bar showing how much of the goal has been completed by the user	When the Badges window is shown, under each goal description there is a green progress bar showing how much of the goal has been completed by the user	Pass

	4.3) Verify that the progress bar updates each time the user revisits the window after progressing a goal	3.4	TC 4.1, 4.2	Show the Badges window and take note of the current progress of a tests goal, then go back to the Home window and change the progress of the test goal, go back to the Badges window. Change the goal to progress forward and back.	After the goal progress has been changed by the user, once the user opens the Badges window the progress bar should update accordingly.	After the goal progress has been changed, once the Badges window is shown the progress bar updates accordingly.	Pass
	4.4) Verify that to the right of the goals description there is a corresponding badge which shows the Badge grade the user has for that particular goal	3.4	TC 4.1	Show the Badges window	After showing the Badges window, to the right of each goal description there should be a box coloured bronze, silver or gold which signifies the user's Badge grade for that goal	After showing the Badges window, to the right of each goal description there is a box coloured bronze, silver or gold which signifies the user's Badge grade for that goal	Pass
	4.5) Verify that the user is able to see all the different badges they have collected at the bottom of window under a title "Badges Awarded"	3.5		Show the Badges window	After showing the Badges window, at the bottom of the window verify that the number of badges collected matches with what the user has for their goals and how much they have achieved their goals	After showing the Badges window, at the bottom of the window the number of badges collected matches with what the user had inputted for their goals and how much they have achieved their goals	Pass

	4.6) Verify that the number of each type of badge collected matches with what the user has inputted for their goals and how much they have achieved their goals	3.5	TC 4.5	Show the Badges window	After showing the Badges window, verify that the number shown of each type of badge collected matches what the user has inputted for their goals and how much they have achieved their goals	After showing the Badges window, the number shown of each type of badge collected matched what the user had inputted for their goals and how much they have achieved their goals	Pass
5. Task window	5.1) Verify that the Tasks window has all the current tasks showing	4.1		Show the Task window	After showing the Task window, all the current tasks should be shown in a list type order	After showing the Task window, all the current tasks are shown in a list type order	Pass
	5.2) Verify that to the left of each tasks there is a checkbox	4.1		Show the Task window	After showing the Task window, to the left of each task shown there should be a checkbox	After showing the Task window, to the left of each task shown there is a checkbox	Pass
	5.3) Verify that after clicking a checkbox, the task should disappear and that the task should also disappear from the corresponding text file	4.1		Show the Task window and click one of the checkboxes next to a task	After showing the Task window and clicking a checkbox the task should disappear and any task below should move up to fill the gap. Then open the corresponding text file and the task should also be gone and any tasks below should have moved up to fill the gap	After showing the Task window and clicking a checkbox the task disappears and any task below are moved up to fill the gap. Then opening the corresponding text file shows that the task is gone, and any tasks below have moved up to fill the gap	Pass

	5.4) Verify that if there are no tasks some text should be shown to the user that says “No active tasks” on the Task window	4.1		Show the Task window after making sure that there are no tasks for the day	After showing the Task window where the tasks should be shown there should instead be text that says “No active tasks”	After showing the Task window and making sure that there are no active tasks, where the tasks should be shown there is instead text that says “No active tasks”	Pass
	5.5) Verify that if there are no tasks then after adding a task the text that says “No active tasks” should be replaced with the task	4.1		Show the Task window and after making sure that there are no tasks for the day, add a task	After showing the Task window add a task and the text that says “No active tasks” should be replaced with the added task	After showing the Task window and adding a task, the text that says “No active tasks” is replaced with the added task	Pass
	5.6) Verify that at the bottom of the Task window there should be a text box where the user can enter the description of a task	4.1		Show the Task window and at the bottom of the page if there is a white text box area start typing in it	After showing the Task window and checking that at the bottom of the window there is a text box area, after typing in it and check that each keystroke is recorded and shown in the text box	After showing the Task window and at the bottom of the window there is a text box area, after typing in it each keystroke is recorded and shown in the text box	Pass
	5.7) Verify that to the right of the text box from 5.6 there is a “Add task” button	4.1	TC 5.6	Show the Task window	After showing the Task window, to the right of the text box there should be a button that says “Add task”	After showing the Task window, to the right of the text box there is a button that says “Add task”	Pass

	5.8) Verify that after clicking the “Add task” button a new task is created and shown to the user	4.1		Show the Task window, add a task description in the text box and then click the “Add task” button	After showing the Task window and adding a task, the new task should be shown under the other tasks (if there were any) and the task should be added to the corresponding file holding the day’s tasks	After showing the Task window and adding a task, the new task is shown under the other tasks (if there were any) and the task is added to the corresponding file holding the day’s tasks	Pass
	5.9) Verify that if the “Add task” button is clicked but the text box to the left is empty the user should be shown an error message telling them to add a task description and the same thing should happen if the user tries to enter duplicate tasks	4.1		Show the Task window and click the “Add task” button	After showing the Task window and clicking the “Add task” button the user should get an error message telling them that they have to add a task description before they can add the task to the day’s list and the same thing should happen if the user tries to enter duplicate tasks	After showing the Task window and clicking the “Add task” button the user gets an error message saying that they have to add a task description before they can add the task to the day’s list and the same thing happened when the user tried to enter a duplicate task	Pass
	5.10) Verify that if more tasks that can fit into the screen are shown then a scroll bar shows up which the user can use to see the other tasks	4.1		Show the Task window and add tasks until the scroll bar shows up, then use the scroll bar to see all the tasks	After showing the Task window and adding enough tasks to show the scroll bar, using the scroll bar by dragging it up shows tasks at the top of the list and dragging the scroll bar down shows tasks at the bottom	After showing the Task window and adding enough tasks a scroll bar shows up, using the scroll bar by dragging it up shows tasks at the top of the list and dragging the scroll bar down shows tasks at the bottom	Pass

	5.11) Verify that at the top right of the Task window there is a Back button that allows the user to go back to the Home window	4.1		Show the Task window and click the Back button	After showing the Task window and clicking the Back button the Task window is replaced with the Home window	After showing the Task window and clicking the Back button the Task window is replaced with the Home window	Pass
6. Goals window	6.1) Verify that user has three drop down buttons on the rightmost part of the window where they can enter their goal for sleep, productivity, and quality of day	3.1, 3.2		Show the Goals window	After showing the Goals window check the rightmost part of the window and there should be three drop down menus stacked on top of each other	After showing the Goals window check the rightmost part of the window and there are three drop down menus stacked on top of each other	Pass
	6.2) Verify that the first drop down menu has a label to its left saying “Sleep an average of * hours a day” where * is the value of the drop down menu which the user chooses	3.1, 3.2	TC 6.1	Show the Goals window	After showing the Goals window, check that the first drop down menu has text to the left of it saying, “Sleep an average of * hours a day”, the value of * should match the value the user selects from the drop down menu	After showing the Goals window and checking that the first drop down menu has text to the left of it saying, “Sleep an average of * hours a day”, the value of * matches the value the user selects from the drop down menu	Pass

	6.3) Verify that the second drop down menu has a label to its left saying, “Achieve an average day rating of *”, the value of * should be the value selected by the user from the drop down menu	3.1, 3.2	TC 6.1	Show the Goals window	After showing the Goals window, check that the second drop down menu has text to the left of it saying, “Achieve an average day rating of *”, the value of * should match the value the user selects from the drop down menu	After showing the Goals window and checking that the second drop down menu has text to the left of it saying, “Achieve an average day rating of *”, the value of * matches the value the user selects from the drop down menu	Pass
	6.4) Verify that the third drop down menu has a label to its left saying, “Maintain an average productivity of *%” where * should be the value the user selects from the drop down menu	3.1, 3.2	TC 6.1	Show the Goals window	After showing the Goals window, check that the third drop down menu has text to the left of it saying, “Maintain an average productivity of *%” the value of * should be the value the user selects from the drop down menu	After showing the Goals window and checking that the third drop down menu has text to the left of it saying, “Maintain an average productivity of *%” the value of * matches the value the user selects from the drop down menu	Pass
	6.5) Verify that when the user changes the value of any drop down menu the value in the corresponding label changes as well and the value in the text file changes as well	3.1, 3.2	TC 6.1, 6.2, 6.3, 6.4	Show the Goals window and change the value of each drop down menu one by one, check the value in the corresponding label and text file after each change	After showing the Goals window and changing each drop down menu’s value one by one the value in the corresponding label should change as well and the value in the text file should change too	After showing the Goals window and changing each drop down menu’s value one by one the value in the corresponding label changes as well and the value in the text file changes too	Pass