Electronics and Computer Science
Faculty of Engineering and Physical Sciences
University of Southampton

Lewis Hawkins - lh5g19
Max Burgess - mwmb1g19
Michelle Moh - mjhm1a18
Peik Lim - pxl1g19
Thomas Hoad - tdh1g19

September 2022 - January 2023

Tracking fatigue after brain injury in real-time using a smart-phone app and sensors/wearables.

Project supervisor: Mark Weal
Second examiner: Mercedes Arguello Casteleiro

A Group Design Project Report submitted for the awards of:

Computer Science MEng
and
Electrical and Electronic Engineering MEng

Statement of Originality

- I have read and understood the ECS Academic Integrity information and the University's Academic Integrity Guidance for Students.
- I am aware that failure to act in accordance with the Regulations Governing Academic Integrity may lead to the imposition of penalties which, for the most serious cases, may include termination of the programme.
- I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.

We expect you to acknowledge all sources of information (e.g. ideas, algorithms, data) using citations. You must also put quotation marks around any sections of text that you have copied without paraphrasing. If any figures or tables have been taken or modified from another source, you must explain this in the caption and cite the original source.

I have acknowledged all sources and identified any content taken from elsewhere.

If you have used any code (e.g. open-source code), reference designs, or similar resources that have been produced by anyone else, you must list them in the box below. In the report, you must explain what was used and how it relates to the work you have done.

I have not used any resources produced by anyone else.

You can consult with module teaching staff/demonstrators, but you should not show anyone else your work (this includes uploading your work to publicly-accessible repositories e.g. Github, unless expressly permitted by the module leader), or help them to do theirs. For individual assignments, we expect you to work on your own. For group assignments, we expect that you work only with your allocated group. You must get permission in writing from the module teaching staff before you seek outside assistance, e.g. a proofreading service, and declare it here.

I did all the work myself and have not helped anyone else.

We expect that you have not fabricated, modified or distorted any data, evidence, references, experimental results, or other material used or presented in the report. You must clearly describe your experiments and how the results were obtained, and include all data, source code and/or designs (either in the report, or submitted as a separate file) so that your results could be reproduced.

The material in the report is genuine, and I have included all my data/-code/designs.

We expect that you have not previously submitted any part of this work for another assessment. You must get permission in writing from the module teaching staff before re-using any of your previously submitted work for this assessment.

I have not submitted any part of this work for another assessment.

If your work involved research/studies (including surveys) on human participants, their cells or data, or on animals, you must have been granted ethical approval before the work was carried out, and any experiments must have followed these requirements. You must give details of this in the report, and list the ethical approval reference number(s) in the box below.

78456

Contents

Chapter 1

Introduction

Firstly, this report introduces the problem statement of the project, the solution proposed to counter the problem, the collaboration with our external customer, Dr Leisle Ezekiel and project supervision under Dr Mark Weal. This is then followed by a general overview of an existing prototype and the requirements of the new prototype set out by our external customer and concludes with a brief insight into the product's description. The body of the report breaks into initial chapters documenting the overall system design of the prototype, describing the development and build of each feature in the prototype. The second half of the report contains chapters which document both internal and external testing of the app, compares the project specification to the end-product, an overview of how the project was managed, ending with a conclusion of the project and a proposal of future development for the app.

1.1 Problem Statement

Nearly 400,000 individuals are admitted to hospitals because of a brain injury. Acquired Brain Injury (ABI) is classified as injury sustained by the brain after birth and is usually categorised as traumatic and non-traumatic i.e a fall, tumour or a stroke[14]. The aftereffects of a brain injury vary from one individual to another depending on the severity of the injury. This impacts the normality of their lives, preventing the continuation of work, social and leisure activities. The effects affect mainly their cognitive abilities i.e concentration span, memory retention, slurred speech and etc. However, the common effect of a brain injury found in most patients is fatigue. Managing fatigue and improving the impact it has on daily life is rather difficult due to its nature of being unpredictable and individuals that have deteriorating cognitive abilities may find it difficult to monitor and be more aware of their own fatigue levels.

1.2 Solution

To address this issue, our solution moving forward is to build a mobile app that can aid patients with brain injury, the ability to gain a greater

understanding of their fatigue levels and learn how to manage them. This will be done through activity-based surveys and the monitoring of vitals with the help of wearables such as a FitBit or an Apple Watch. By incorporating these features onto the app, this provides a pathway for patients to have more awareness of their energy levels and to monitor their energy levels. The app will then allow them to gain a better understanding of what activities may potentially be affecting their fatigue levels more so than others and prompt the patients to make changes to the activities to improve the impact that fatigue has on their daily lives.

1.3 Collaboration

This project was done in collaboration with Dr Leisle Ezekiel and under the supervision of Dr Mark Weal. Dr Leisle Ezekiel is a registered Occupational Therpist at the University of Southampton in the School of Health Sciences with research centred around social participation after brain injury, experiences of fatigue post brain injury and ecological momentary assessment of daily activities and fatigue following brain injury or long COVID. Her research also utilises participatory and collaborative approaches such as centred design and she is interested in a wider spectrum of mHealth to support self-management of long-term health conditions.

1.3.1 Integration With An Existing Prototype

Prior to the commencement of this project, a working prototype app using an android operating system was developed and was tested by people with brain injury. This prototype consisted of repeated surveys where people could log their activities and fatigue at intervals throughout the day, sampled ambient noise and physical activity transitions, and concluded with a short reaction time test which was an indicator for mental fatigue.

1.3.2 Purpose of New Prototype

A new prototype was needed by the external partner to address design issues and to further the development of the app. Suggestions for the new prototype included the following: collecting physical activity and sleep data from external devices such as a FitBit, providing daily summaries of the responses from the user surveys, reaction time scores, allowing customisation of the app by the user, provide an enhanced short reaction test to increase user's engagement, tracking of environmental noise, possibly capturing phone use data that may indicate fatigue- a slow response or increased errors on tasks.

1.4 Product Description

As the new prototype is an extension of the previous one, there will be some elements that will remain the same. The prototype will still consist of

the surveys where the users can report their fatigue using a n-point Numerical Rating Scaled (NRS), log in their activities by participating in an activity questionnaire which will collect information regarding the user's current/most recent activity and a reaction time test also known as a Psychomotor Vigilance Test (PVT) as a means of assessing any changes in levels of fatigue.
The additional elements include a notification system which prompts the user to take a survey reporting their current levels of fatigue and activity, a dashboard page where users will be able to view a summary of their daily activities which is made up of the responses from the questionnaire i.e the NRS and PVT results, giving the user the ability to synchronise data from their wearables such as a FitBit.
According to our partner's published research, there was emphasis on allowing more user customisation to the app. The main features of user customisation were centred around allowing users to set the time frame in which they would want to receive the notifications prompts, a time when a daily summary would be ready to view, and allowing the user to switch between dark and normal mode. Last but not least, there is a settings page that housed these user customisation features making it easier for the user to locate.

Chapter 2

System Design

In the previous chapter, we highlighted the need to build an Android mobile application that could help patients with brain injury gain a greater understanding of their day-to-day fatigue levels. As with all software projects, every human-centered product design project starts with the user interface (UI) design process. Considering that the UI would act at as sole point-of-contact between the user and app interface, it is crucial that we design an app that balances functionality with the ease of use. As we walk through this chapter, the key functionalities of the app will be explained more thoroughly in their individual subsections. This subsection in particular aims to walk the reader through the timeline of the UI design process and additionally, briefly explain the design aspects that were taken into account.

As mentioned in section 1.3.1 - 1.3.2, a working prototype of the app had already been previously developed by the customer. The previous prototype consisted of several essential features that would allow the user a more personal and individual understanding of their fatigue and the conditions that lead to fatigue. Some examples of these features are the Ecological Momentary Assessment (EMA) and the Psychomotor Vigilance Test (PVT). These features will be explained in further detail in the upcoming sections.

2.1 Terminology

2.1.1 User Survey as an Ecological Momentary Assessment (EMA)

An ecological momentary assessment, put simply, is a method of assessing a subject's current experiences and behaviours in the individual's normal/-natural environment. This is typically done by repeatedly sampling people's thoughts and activities several times throughout the day, during or close to the time they carry out that activity. The action of repeatedly sampling subjects in their naturalistic environments is especially useful for users to understand and report changes in behaviour, symptoms, and cognitions as they carry out different tasks. The method of collection, whether by using a specific piece of technology or using a diary, may vary. However, since our

goal is to extend on the previous prototype and to further the development of the mobile app, we decided to deliver the EMA through the app and integrate it as one of the app's main features.

[Later: talk about the benefits of using a SMART EMA]

2.2 UI Design

2.2.1 Early-Stage Design (Low Fidelity)

In the early stages of the project, we had the opportunity to meet Dr Leisle to discuss user and app requirements. During this meeting we were able to view the previous prototype and its features, identify user needs, and most importantly get a brief idea of what the final product will look like. We were also provided some useful resources that will be used later in the project, such as the user-survey questionnaire that Dr Leisle had designed in her previous user-centered study. To aid the ideation stage of the design process, we took to the drawing board to sketch out the basics of the user interface before deep-diving into prototyping and developing the app. Basic sketches of the app were done digitally on a sketching app on a tablet.

[Insert images of sketches here]

From the images above, it can be observed that only a minimal number of the visual attributes of the final product were drawn. Of all sketches, the least expressed page/feature at this stage was the dashboard (also called the daily summaries page). The dashboard was ideated to be the go-to page for users to view their responses to their fatigue surveys, so that users can monitor and better understand their fatigue levels. At the time, we had an understanding that we will require a feature that would allow users to do so, but we were still in the brainstorming process on how to relay the information back to the user in a simple yet informative way. There are a multitude of ways for us achieve this, such as using charts/graphs or just by displaying text on a UI card (Android CardView) based on their responses. As this was still in the early stages of the project timeline, we decided to prioritise exploring what types of data we will be able to retrieve such as survey responses and data from a FitBit before designing the dashboard on a higher level. The same applies for the sketch of the PVT (also called the reaction time test) as there were several methods of implementing the test.

By observing the other sketches, we take note of the key elements of each page, such as the survey question/prompt, instructions for the user, user response and input methods (buttons and sliders), and also navigation and progress bar statuses. The entire process of creating low fidelity sketches such

as the ones shown above only took 10-15 minutes at most and did not require any setup. Being able to draft sketches without too much effort allowed us to quickly switch between methods of implementing certain features, making it extremely advantageous of formulating a clearer expectation of the final product.

2.2.2 Prototyping on Figma

After drafting out the sketches, we took to prototyping the app in more detail on Figma. Here we will explain the individual components that make up every page/feature. It might be useful to note that the prototypes shown here may differ from the final product design in terms of the layouts, fonts, font size, navigation buttons and so on. This is the result of continuous improvements being made to the app over time, after getting design feedback from within the team, Dr Leisle and our supervisor Dr Mark. Any design changes will be highlighted in chapter 3.

Homepage: The initial prototype for the home page of the application included two buttons that will allow users to navigate to two major parts of the app. Intuitively, upon clicking on the first button, the user will be navigated to take the EMA survey; upon clicking on the second button, the user will be brought to the dashboard where they can view their user survey responses. On top of the page, the message "Hello, what would you like to do today?" greets the user and lets the user know what actions he/she can take. The final design largely took on our prototype.

User survey: The user survey part of the app will collect user responses mainly via two input methods: sliders for numeric rating scales (i.e. 0-10 or 0-100 point scales) and buttons to input user's selections to particular questions. In the prototype, the first part of the survey (can be seen in [INSERT FIGURE NUMBER HERE] consists of a question asking the user what their energy levels are. Instructions are given to the user to drag a numeric pointer scale slider (between 0-100) according to how energized they are feeling. Above the slider, a picture of a battery is included as visual aid. A progress bar is also shown on the top of the page to give the user an idea of how much of the survey is left. On the bottom of the page, a next button is included to allow the user to navigate to the next question of the survey.

In this part of the survey, we ask questions about the user's current or recent activity (an activity that was done in the last 5 minutes or so). The questions and answers we have used in the survey had been given to us by Dr Leisle and had already been designed to capture as accurately the recent activities of a user, without the need of asking too many questions. This is firstly done by inquiring the user's current location (e.g. at home or somewhere else). After that, a more general question follows, asking about the broader category of the activity they are engaged in (e.g. social activities or working and studying). Finally, we get into the specifics of the activity the

user is carrying out (e.g. cooking or buying groceries). By doing it structurally this way, we eliminate the need to include the hundreds of activities that a person could be doing. This prevents users from getting overwhelmed by the number of possible selections and keeps the survey simple. From the screenshots of the prototype above, buttons are once again used here as the main method of collecting user responses.

Moving on, the last set of questions of the survey are specific to the activity that the user just carried out. Questions like how much effort did it take, how have your energy levels been affected, and how fatigued are you, are prompted to the user. Depending on the question, sliders and buttons are used to collect user input data just like the previous pages that we have implemented in the prototype. The same elements that were mentioned before, such as the progress bar, back button, and next button, are shown on each page in the prototype. After answering the final question, the user should be able to click the "submit" button on the bottom part of the screen to end the survey. The final design can be seen in Section 3.2.3.

Reaction time test: After a user has completed the survey questions, they will be invited to participate in a reaction time test. The Figma prototype of the reaction time test only consists of four screens, all of which are static images and do not showcase how the test works. Taking a look at the screenshot in [INSERT FIGURE HERE], the first screen contains some prompts inviting the user to take the reaction time test. The user has two choices here, either to go to the reaction time test or to skip it. If the user chooses to skip the test, they shall be navigated back to the homepage. If the user chooses to take the test, then they are navigated to the second screen (see figure NUMBER HERE), where a prompt provides instructions to the user to tap on the screen when it changes colour. A start button is included below the prompt for the user to start the test when they are ready. When the user taps on start, they move onto the third screen, where the actual reaction time test takes place. In the centre of the screen, the prototype shows a grey area – this area would be the part of the screen that will undergo a colour change. It is also the area where the user is meant to tap when the stimulus occurs.

Not pictured in the prototype is the actual mechanics of the reaction time test, where each colour-change-then-tap cycle will be repeated for a specific duration. More of this will be further discussed in Chapter 3. In the prototyping stage, we also aimed to display the user response times after each iteration, hence it is included in the prototype. Once finished with the test, the user can click on done and head to the last page (i.e. the fourth screen) of the reaction time test, where the average response time is displayed. From there, the user can navigate to the dashboard to view their survey responses and reaction time test scores.

2.3 Reaction Time Test

2.4 Data Dashboard

Another vital section of the application is the data dashboard – an area of the app where the user is able to see their responses to fatigue surveys and have their data visually displayed to them. It was important for us to include a data dashboard in the app so that the users could review the history of their responses, with the aim being that they could use the dashboard to monitor and better understand the causes of their fatigue.

We had multiple goals when designing this section of the application. Firstly, we needed the data dashboard to show daily summaries of the user's fatigue in addition to the full data collected in the time since the user first began to record responses to surveys. While the daily summaries allow the user to reflect on their recent fatigue at the end of the day, we also want to provide users with the ability to look back over a larger time frame and at specific dates and times to view responses they may have otherwise forgotten about. We would also aim to separate these sections clearly in the name of creating a simple experience for the user. Furthermore, we had the goal that both of the previously mentioned parts of the data dashboard would display data related to the user's energy levels, reaction time and wearables in a visual manner that is not cluttered or confusing to the user. This visual presentation of the various data related to fatigue will aid users in spotting any trends or patterns in their fatigue over time which could lead to them being able to better manage or control how they experience their fatigue. The data should be shown in a helpful way in order to allow users to achieve this as easily as possible. Similarly, we had the goal that the user should be able to view the full data from every time they answered a fatigue survey including the date and time that the survey was taken and the individual response to each survey question. Once again this was to be done in a way that is intuitive and helpful to the user and that does not cause any confusion itself. This would further aid the user in analysing the causes of their fatigue by reminding them how they answered surveys from a time that they may not otherwise remember.

The solution that we designed uses a horizontal bar with two tabs to allow the user to select to view the 'daily summary' or the 'lifetime' section. As the user does not need to be able to see both of these sections simultaneously, we are able to use the remaining space on the screen below the tabs to display the contents of the user's selected section. We then decided that this space would be divided vertically into an area for graphs of different data and an area for a list of the user's survey responses. A first sketch of this is shown below.

The top of each section would be home to an area for a series of charts that would provide a visualisation of the user's fatigue data. This graph section would be home to three different line graphs – one for the user's energy level

(Question 1) in each survey response, one for the user's reaction time result and one for data collected from a user's wearable, specifically a Fitbit. Each of the graphs would individually take up the width of the screen with the user having the ability to scroll horizontally to change the graph that would be shown. Furthermore, each of the graphs would be given a unique colour to assist the user in not getting lost or confused when navigating between the graphs as they get more familiar with the application.

We decided that the remaining section beneath the graphs is where the user will be able to view the data from previously answered surveys. For the 'daily summary' section the user will only be able to see their response to surveys taken on the date of the most recent full day whereas for the 'lifetime' section the user will be able to see details of every survey they have ever taken. In both cases the user will be presented with a vertical list of shortened versions of each survey response, sorted by the date and time that the survey was taken. The user will be able to scroll up and down through these briefs before selecting a response to view the full data in a popup.

Our initial proposed solution for the dashboard, as presented in our progress report on the 16th of November, included a single graph at the top of the screen with the tabs and separate sections below that. [add image?]. We moved on from this solution when we confirmed that we would be presenting multiple types of data visually. Furthermore, this does not allow for a clear enough separation between 'daily summary' and 'lifetime' data graphs. While we could also implement the solution from the final implementation here, with six individual graphs to scroll between, this would not meet our goal of presenting the data in a way that is not confusing. Other alternative solutions for the dashboard included having just a single graph to display the three different types of data in each section. We decided against this approach as it would not have met our goal of creating an uncluttered display. Plotting more than two types of data on the same line graph introduces a difficulty with drawing scales on the graph axes and therefore while having a collection of lines on the same pair of axes may have made identifying some trends over time somewhat easier, we decided in favour of separating the data into separate individual graphs with their own colours and scales.

2.5 Settings Page

We also decided to include a settings page in our application as we felt that it was important that the user is able to customize their experience with the application, including giving the user some control over the timing and frequency of the notifications that are sent to the user's phone to remind them to take a fatigue survey. Furthermore, we decided that the user should be able to block out certain times of the day when they do not want to receive notifications, such as when they are working. These features were crucial to include in the application because they allow users to only receive notifications at times when they are able to take the survey and are therefore happy to log their responses multiple times every day. Additionally, we wanted the settings

page to contain the normal features and adhere to the normal conventions that are commonly found in settings pages, in order to increase the user's ability to use the app with ease and without confusion.

The solution we designed would incorporate the use of a drop-down menu followed by a series of sliders to present the user with a clear method to adjust their notification preferences. This would include a slider to specify the earliest and latest time of the day between which they would like to receive alerts, a 'do not disturb' slider which would prevent the user from receiving alerts between two specified times and a slider to let the user set the time of day when they receive a notification to inform them that their new daily summary is available to view in the dashboard. These settings would be placed in a 'notification preferences' section at the top of the settings page and would precede the page's remaining contents, including an FAQs section and a button to let the user log out of their account.

Each slider would be clearly labelled to communicate to the user the changes they are able to make, and any other buttons would also be given a text label and a small icon to increase the user's familiarity with the app. Moreover, the 'do not disturb' slider would be made red to ensure that the user understands that they would *not* be receiving notifications during this period. Similarly, the log out button would also be red and would be the very final button located at the end of the page as is commonly found in many other applications.

Chapter 3

Development and Builds

3.1 Development Setup

The main aim of our development was to get started with a basic version of the app that could run user surveys. The surveys created a large amount of dependencies on other features we wanted to implement and this basic template was important to establish quickly. We went in without any experience of app development so it was critical we came up with an approach that we we could all get comfortable with quite quickly.

3.1.1 IDE and Emulators

To develop the app, we decided on using Android Studio with Java. We discovered that IntelliJ had Android Studio integration so we used this as a platform that we were familiar with, provided Git controls and would also provide all the Android Studio features. The app development provided us with a template app to get started with which meant we could immediately start working on features right out of the gate. Most of us are very strong with Java, so even though Kotlin is quite a popular app development language and was an option for us, we decided to go the familiar route.

Shown below is a look at our development environment. Here we can see an example of how we are using the Android Studio tools such as setting up an emulator to test our app with ease. We decided collectively on using the same emulated device so that our displays would all look the same as each other. The device we simulated was a Pixel 5 running Android 11, API 30. We chose these specifications because API 30 is compatible with a solid 40 per cent of devices and provided us with essential tools that were required for notifications. We considered using a device running Android 8 or 9 to give us compatibility with 90 to 75 per cent of devices but the overall UI look wasn't as clean and made notifications significantly more challenging to display consistently.

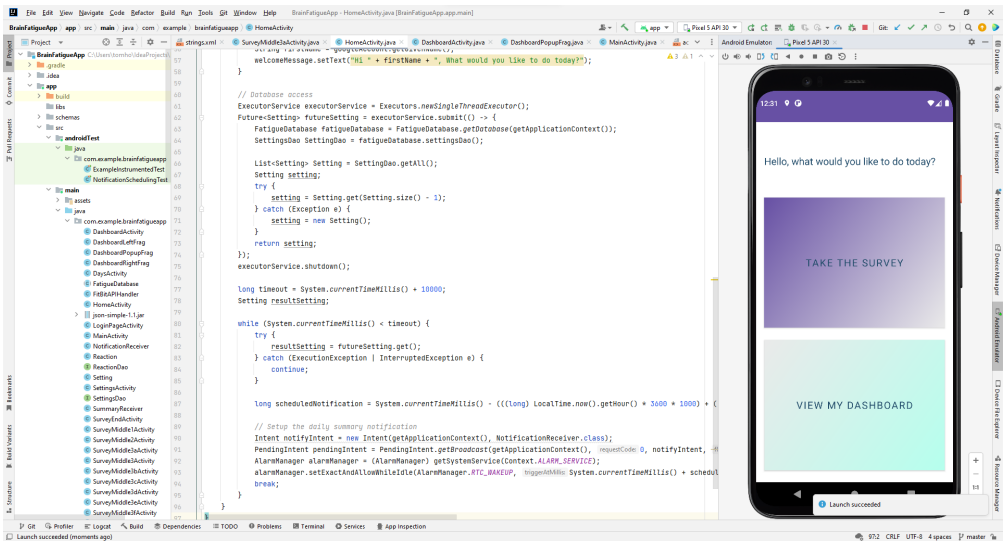

Figure 3.1: IntelliJ Development Environment and Emulator

In addition to the Emulator setup within IntelliJ, we also had a couple of our own Android devices that we used for additional testing during development. Setting up the app on these devices was very easy. Android provides development settings that allowed us to download our app from IntelliJ to the device over WiFi and it became fully functional. This was very beneficial for getting a proper hands-on feel of the app, as well as allowing us to test notifications being sent without having to leave an emulator and hence a computer on overnight.

3.1.2 GitHub Repository

To work collaboratively, we set up a private GitHub to store our work. We used the methodology of creating feature branches for each idea, creating commits on this branch, and then creating a pull request when ready to merge into the master branch. During development, we set up these pull requests to also squash the commits before a merge which also formalised our Git history.

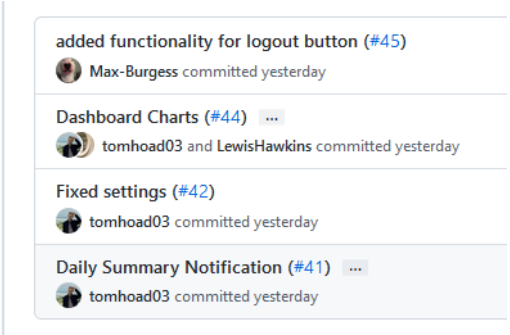

Figure 3.2: Git Pull Request History

We encountered a few problems with Git throughout development, but most issues were quite resolvable using IntelliJs useful Git integration features. Using pull requests rather than all working on master meant we had very few conflicts arising and meant we always had a branch with a working product we could demonstrate or test with at any given time.

3.1.3 Activity Layout

A main principle of using Android Studio is developing the app using Activities and Fragments. Activities act as a new page whereas fragments can be used to display more space on a single activity. Through our development we setup a template using mostly activities, with the use of fragments within the dashboard. Shown below is the layout of activities that make up our app.

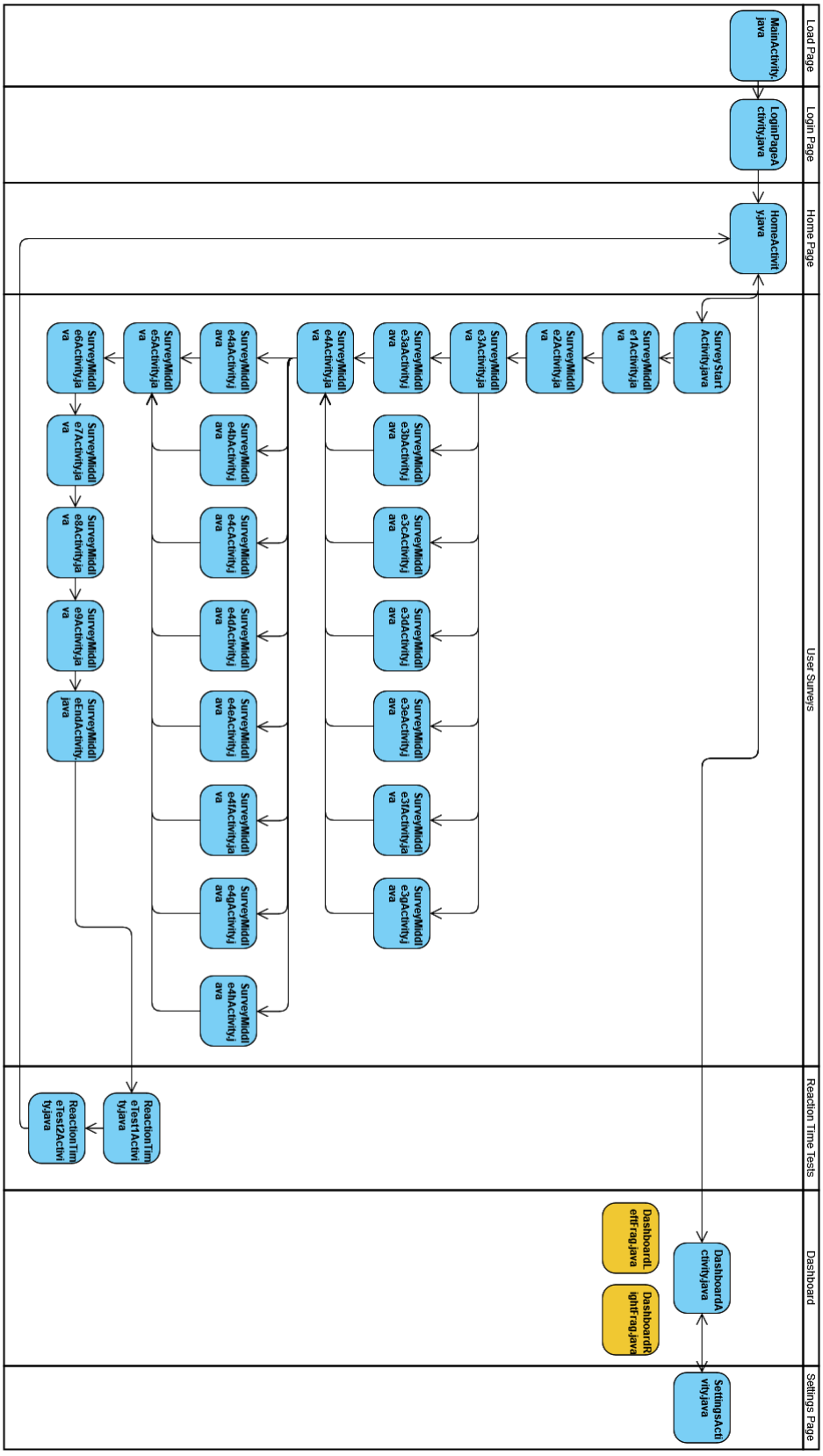

Figure 3.3: Activities Layout

3.2 User Surveys

3.2.1 Survey Development

The development of our survey largely took on our UI design and question structure given in the preparation. We had to make a few changes here or there from our design for elements that didn't look right when translated to a proper device and we went through several iterations of the design.

The user surveys were the first part of our development so, a lot of the UI development was brand new to us and took a while to adjust. With a lot of elements on the page, we had a tough time making sure they were arranged correctly and that layout translated to an actual device. This task was done collaboratively and over the entire duration of the project as we got more comfortable with our development skills and learnt new tricks. Incremental feedback was useful to make gradual improvements over time that ultimately helped us match up the surveys to our initial design.

Shown below is an image during the early stages of user survey development. At this point there are only general layout features added, not all questions are fleshed out and the look of the design is only half done.

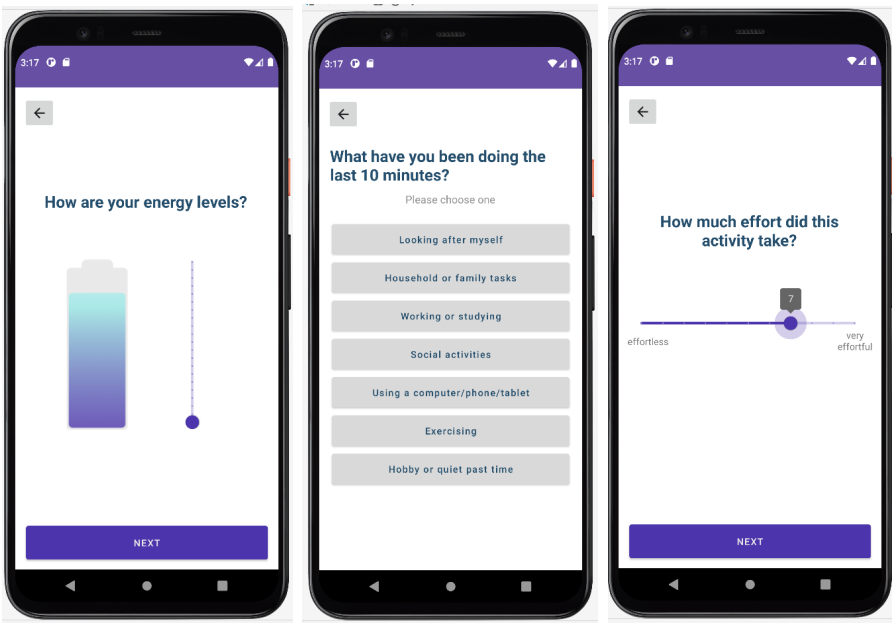

Figure 3.4: Early User Survey Pages

3.2.2 Survey Navigation and Results Collection

Once we had the initial development completed, we needed to get some kind of survey result collection implemented. Whilst perfecting the UI was important, we also needed to make sure the data was available for other parts of the app such as the daily summary and the lifetime dashboard. To collect the information we created a Java class called SurveyResult.java shown in the appendix B. We used the activities chained together to create a SurveyResult

object with the values inside to reflect the button clicked on. Navigating back in the survey made sure to properly update the results. When the end of the survey is reached, the object is saved to the app database and marked with a timestamp.

Android development uses intents to navigate apps which directly works with Android devices' built-in home and back button. During the initial stage of development, we designed our back button for the survey which we discovered was a bad course of action. Referring to this article [12] we discovered that it's a waste of space and also messes with the stack tracking inherent to Android. The screen real estate we gained from the decision to remove the button was very beneficial and we used it to make improvements later on. An important and final part of our user survey design navigation design was to program the app to block off returning to the user survey once it has been submitted. This prevents a user from repeatedly submitting the same survey and confusing the results.

3.2.3 Response to Design Feedback

During a customer-supervisor meeting we had at the beginning of development, we went through the user survey design and received feedback to improve the UI. These changes were largely implemented in the final version of the design. We also came up with a few improvements of our own to make the user experience better. These changes include:

- Improved question wording and order.
- Improving slider variety and scaling to give numbers and a key.
- Cleanly attaches the reaction time tests to the end of the survey.
- Removing our custom back buttons, replacing them with a survey progress bar.
- Implementing our custom colour scheme to provide a more comfortable user experience.

Shown below are a few examples from our final user survey design:

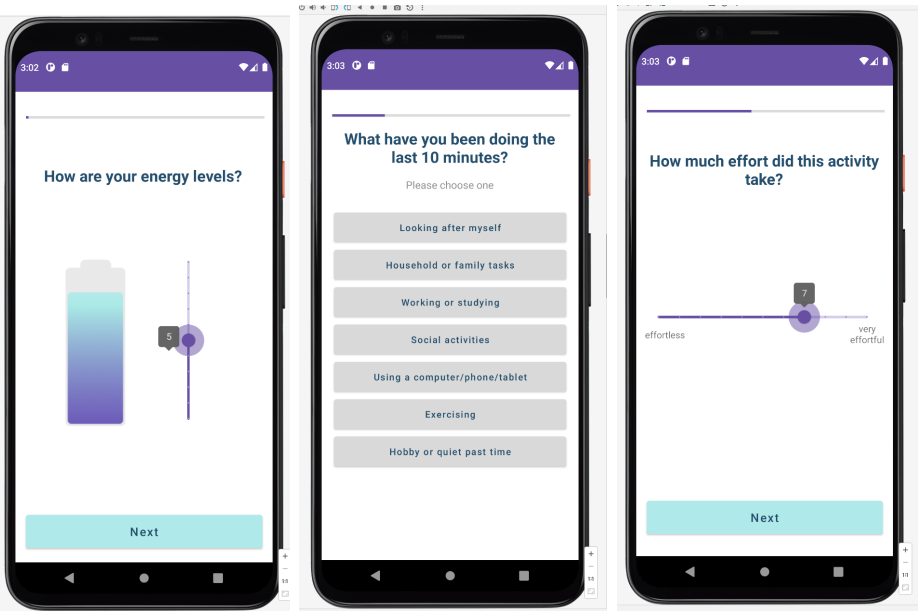

Figure 3.5: Final User Survey Pages

3.3 Reaction Time Tests

3.4 Notifications

The notifications were a critical feature for the app and were perhaps the hardest part to design well. It took us a good length of time to get these systems in place correctly and they ended up taking longer than we had planned in our Gantt chart. This was partly because of its dependency on the settings page which was designed late on, but also because Google is continually scaling down how certain background functionality works in Android apps.

We have multiple different notifications for this app. Firstly, we have a brain fatigue survey notification that goes off a number of times a day at fairly regular intervals. Secondly, we have a daily summary notification that will give users a rundown of their surveys in the last 24 hours. Finally, we have a second survey notification that designed to go off in extreme scenarios such as too much noise, too much movement or a large change in fitbit data.

3.4.1 Notification Design

The notification design and building all happen within one Java class, NotificationReciever.java which is highlighted in appendix B. Within this class, we lay out the design features for a notification. We also use another receiver to send the summary notification which has a very similar design to this. The following features apply to all notifications:

- Vibrate when notification is received.

- Forces the notification to appear as a heads-up display.
- Categorises the notification as a phone call, increasing general importance.
- Sets up the notification to launch the app to a specific activity when interacted with.
- Sends the notification even if others are suppressed.
- Dismisses the notification when the user interacts with it.
- Prevents the user from dismissing the notification any other way than interacting with it.

As a result, this provides us with a notification that will always persist on the screen for the user, meaning they have to interact and start the brain fatigue survey to remove it. We decided on this design choice because we placed high importance on these notifications, and we wanted them to get a guaranteed response from the user. It also meant that implementing noise response notifications would allow for an immediate interruption. As a result, this is how the notification will appear to the user.

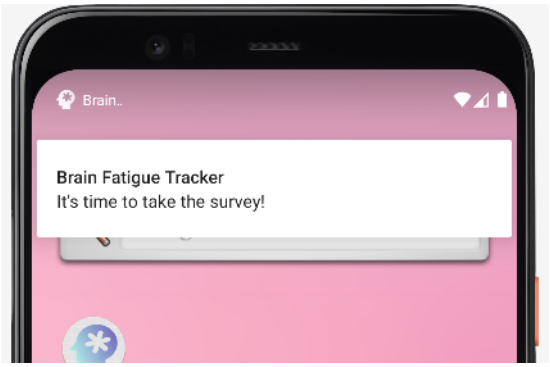

Figure 3.6: Fatigue Survey Example Notification

3.4.2 Notification Persistence

A key part of the survey notifications is to make sure that they are seen by the user and that they will act upon it. This was quite challenging to achieve, especially as Google is in general cutting back techniques for processes that run in the background. These are efforts aimed to conserve battery life as much as possible but this meant we had to work around several restrictions. Whilst all these changes seem to have been implemented in newer versions of Android, we needed to work with them rather than avoiding them by working in an out-of-date version. We got quite creative with our notifications system and ultimately it gave us a lot of flexibility.

Survey notifications should be regular, semi-predictable and customisable. We managed to achieve this by calculating when each notification was going

to occur after every completion of the survey. The phone looks at the current time, the available times and the desired frequency of notification and works out the next available time to send the location. We alert the user to when the next notification aims to arrive so that they can get a general idea of what time it will arrive. Given Android's sleep measures, we noticed there was some variance from this calculated time which we appreciated in some regards and became quite annoyed at in others.

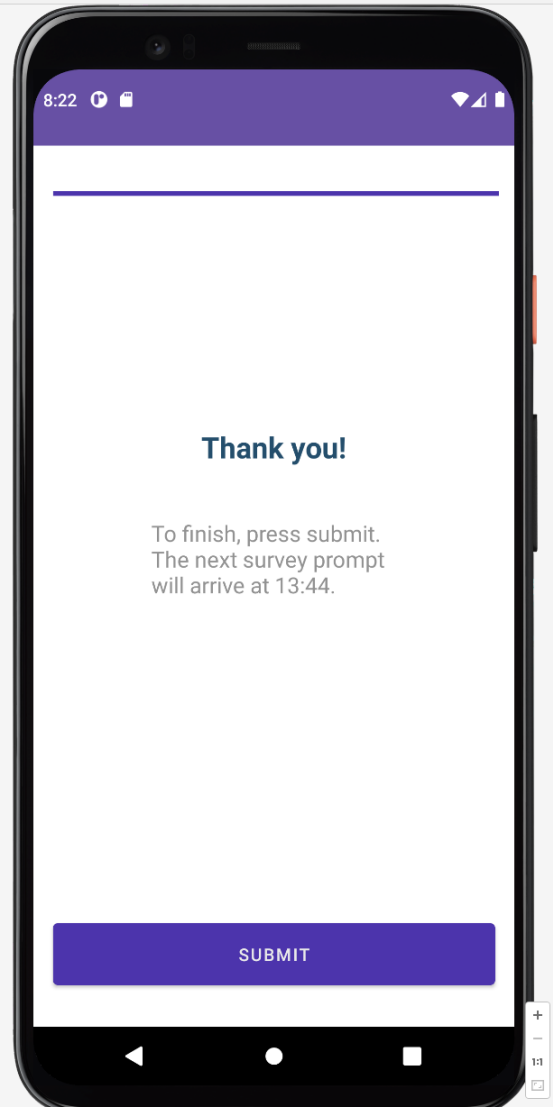

Figure 3.7: Fatigue Survey Notification Warning

The code that demonstrates our alarm scheduling can be found in the survey end activity, appendix B. In this scenario, we use:

```
alarmManager.setExactAndAllowWhileIdle(...);
```

We found this to be the best way to set as precise an alarm as possible. This will require the user to complete the survey when it arrives to send another notification as repeating alarm notifications were always imprecise and sent whenever the system was awake.

3.4.3 Notification Controls in Settings

Once we had the notification system sending regular notifications, we wanted a way to control when it would send them. For this, we implemented some sliders in the settings page and a database table to store the notification times. We established three different factors as to when to send the notification.

- Interval - how often a notification is sent. (1h, 1h30m, 2h, 2h30m, 3h)
- Awake Time - the times of day to send the notifications. (6 am to 12 pm at the extremes, sliders allow 1h adjustments.)
- Do Not Disturb - a block in the day to not send a notification. (At least 1h interval, sliders allow 1h adjustments.)

This leads the app to then calculate when the next notification is sent. This calculation is given in a function shown in the code of appendix B. To explain simply, the app works out the next available time to send the notification, the interval length of time from now. The interval time is only taken off the available time. For example, if a 2h interval is set, the survey is finished at 10 pm and the awake time is set to 8 am to 11 pm, the next notification should arrive at 9 am. There are ways we can make this more random by adding a random number (+-15 minutes) to the end of the notification send time or always sending trying to send at least 3 notifications in the day which would not take long to implement.

3.4.4 Emergency Notifications

The final part of our notification system was designing and implementing an emergency notification system. This idea outlined by our customer explored how survey notifications should be sent in extreme scenarios such as large amounts of noise or activity. Implementing this idea was quite difficult and required setting up the app as a foreground task so that it would run constantly. With how far we got with this feature, we have been unable to fully test the functionality in extreme scenarios when the app gets put to sleep by Android itself to see if it will work.

The version we achieved was one that monitors the background volume. This requires the use of the microphone to measure the highest decibel level in 1 seconds intervals. If the app detects a decibel level greater than a given threshold, it will set off a notification. The notification itself is on a roughly 20 second timeout to prevent being flooded with notifications. Given that decibel is a dimensionless unit, we had to follow an arbitrary function to convert our seemingly null value amplitudes received from the microphone into arbitrary values that lined up with a linear scale. The following equation calculates our volume values, where the constant 32768 (the amplitude value limit) is used to bring the value up to a normal value.

$$Volume = 20 \log_{10}(\frac{Amplitude}{32768})$$

The code for the emergency notifications can be found in appendix B.

The result here is not a fully complete emergency notification system but a proof of concept for definite. Ideally, we would like to expand this system to check the fitbit activity, number of steps, energy levels, etc. Adding these checks would be quite simple as most of the work for having this run in the background is now established.

3.5 Data Dashboard

3.6 Daily Summary

3.7 FitBit Integration

3.7.1 Selection of Device

For the sensors/wearables part of this project title and requirements it was quickly decided that using a already on the market wearable would be preferable to designing our own wearable from scratch , this was for a few reasons:

1. We are only given a few weeks for this project so any wearable we did design would likely be a very early prototype which is a problem because:
   (a) Our customer intends to use the product of this GDP for research so having something that is unfinished leaves them unable to do that
   (b) Even if the prototype was somehow fully functional and worked perfectly there would only be one of it which isn't helpful as even if they managed to run multiple participants with this one prototype given each of the participants would need a few weeks or months with it it would take years to end up with a sample size still too low to publish any meaningful research from..
2. Massive companies like Garmin and FitBit will have access to huge labs full of R&D and will be able to produce sensors and wearables many orders of magnitude more sophisticated than anything we could make.
3. A lot of people already own smart watches and other wearables such as FitBit which means much less money would have to be spent by any researcher using our GDP as a lot of the participants would already own the equipment needed.

Taking into account the decision the next design choice to be made is which one to focus on integrating with, this is because sadly all these companies use different proprietary software and format their data differently so you cannot

create something to work with all of them at once and due again to the time constraints of this GDP resources need to be focused to where they will be most productive. This led us to decide to use Fitbit. The reason for this decision is that two companies own over 80% of the smartwatch market - Apple with 46% and FitBit with 38% [16] which means if we want to pick a wearable a large amount of the population will have we will have to pick one of these two. Of these two FitBit is the better choice for a few main reasons. One of these is price, FitBit devices are considerably cheaper than Apple watches which can have prices up to £1,200 with even their cheapest device with the cheapest configuration still being £249.99 [10] Meanwhile even buying a brand new Fitbit direct from their website at full price you can get them as cheap as £49.99 [11] which is important both for helping our customer get their research done for an acceptable price and for this GDP where we have a relatively small budget which if we wanted to buy one of these devices for testing, even the cheapest apple watch would eat our entire budget. The second main reason for picking Fitbit over apple is that it contains a lot more of the info we need as it is a true Fitness wearable that contains all the info we need. This is opposed to Apple which is designed to be a more general purpose smartwatch and whilst the higher end watches that cost around £1,000 do contain most the features the FitBit devices have the £250 one which is within our budget is missing key metrics such as being unable to give ECG (Electrocardiogram) data[9]. Even if this data was recorded by the device Apple make their devices almost deliberately obtuse to work with with any other device or program which isn't made by Apple to try to encourage consumers to stay within the "Apple Ecosystem". This could make it difficult for us to make our system work with theirs and get the info it needs.

3.7.2 Integration of Device with the App

Now that it was decided that FitBit devices would be used as the wearable portion of this project it was important to consider how this will be done. The original plan was to implement some sort of Bluetooth connectivity so that while you are using the app it could connect to the wearable and get data from it. This however turned out not to be possible as this is not allowed by FitBit due to security concerns. As this was not possible we decided to see how FitBit themselves do it within their dedicated app as they are likely to know the best way to do app-device connectivity with their wearables. On their app the way they do it is through a "sync feature" where you log in using your FitBit credentials then press a "sync" button, shown below, and then the app pulls the information from the FitBit API.

FitBit give, on their website, a tutorial on how to use their API for personal projects, apps and for server applications with different protocols for each.

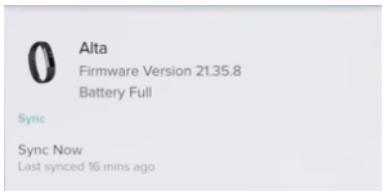

Figure 3.8: Fitbit App Sync feature

3.8 Storing the Data

With the amount of data we are collecting for the app, we needed a way to store it that wasn't going to be too costly to the performance of the app. Android offers a few ways to store information but the key technique we used here was to create a database using Room [13].

3.8.1 Room Database

The database design, as shown below, follows the schema recommended by Android Developers to follow. There is one abstract database that uses multiple Data Access Objects (DAOs) that use custom-written SQL statements to interact with the database. Within the database, there are a collection of tables to store several Java objects. In this case, we are storing the results of the survey, the settings page and the reaction results.

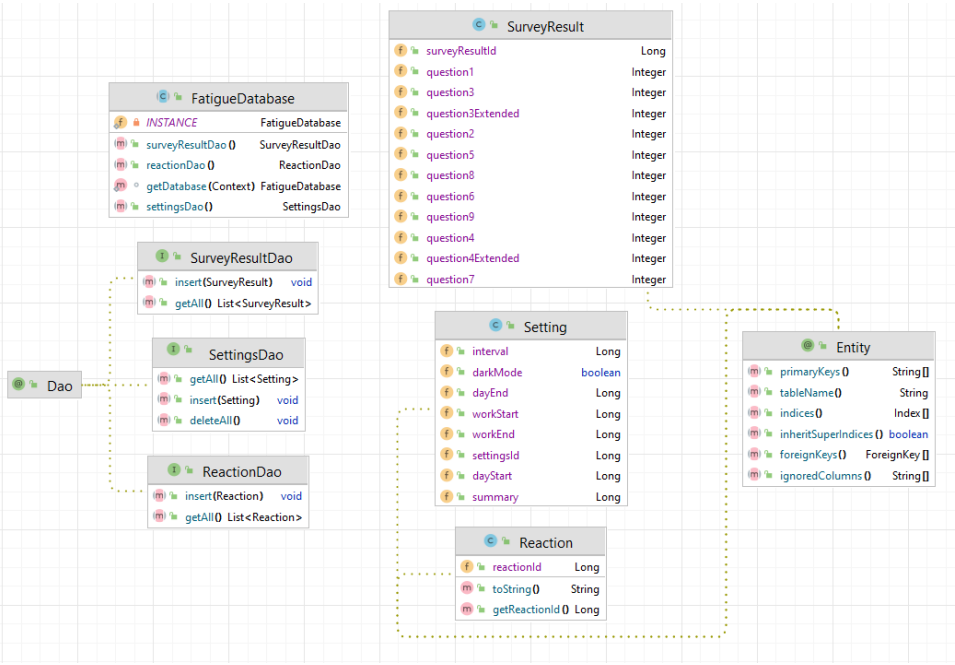

Figure 3.9: Room Database Tables

The database must be accessed on a separate thread than the app runs on, so we had to be careful in how we used it. For our scenario, we decided to use Java Executors which run in parallel to the rest of the app. For those

scenarios when we wanted to wait for information from the server, we used Java Futures to receive a promised result.

Whilst most of this seems overly technical, the result is a very efficient way of storing the data. We noticed that we were easily able to upgrade the database over time using Room and we also noticed the data within would persist beyond just a few changes to the app's code making it resistant to any updates we made. If there was anything we could do to expand this feature of the app, it would be to start encrypting the data. Whilst an actor may have a hard time getting into the database, it'd provide an extra layer of security that could be quite easily developed. We are not storing a user's personal or sensitive data within this database so it wasn't a critical objective for us.

3.9 Login Page

The layout originally planned for the login page and implemented in the first few iterations of the app is shown below. It contains a username and password field along with integrated logins with google and Facebook. The reasoning for the integrated logins is that it has been shown in studies that people with traumatic brain injury can have impaired memory which means having a system which doesn't require them to remember another login could be immensely helpful[2][3]. The reason Facebook and Google were chosen specifically is because a Google account is guaranteed to be at hand for all users of our app as it is the account type android uses and Facebook is the most popular social media amongst the older portion of the population who are those more likely to suffer from traumatic brain injury[7][8].

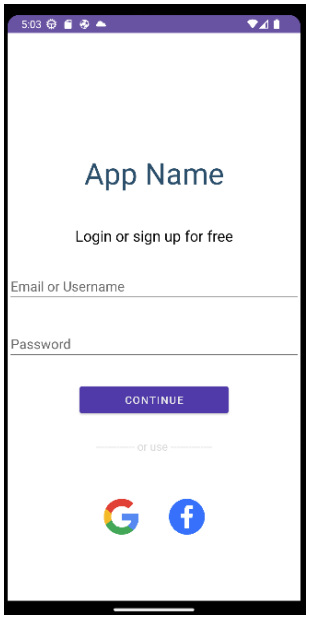

Figure 3.10: Room Database Tables

The first major update and deviation from the plan happened fairly early on in development when we realised if we set up a system for the username and passwords to be stored somewhere securely which would probably require some sort of upkeep cost and/or supervision from a developer. This would not suit our customers needs as they intend to use this for research after this GDP is over and the team who worked on this app have moved onto other things. After consideration of this the username and password fields were removed and in its place was added a option to continue to the app without logging in. Selecting the continue without logging in option means less data can be accessed and some personalisation features will no longer work, for instance they will see a generic greeting on the homepage rather than a personalised one, however these are cosmetic and do not effect the functionality of the app.

The second update to the login page after it's inception came during testing in which it was discovered the Facebook integrated login system we used was deprecated by Facebook themselves as an anti-phishing measure. This meant it had lost all functionality unless additional steps were taken by the Facebook user to go into their account settings to re-enable it. This is discussed further in section 4.1.1.1. After this was removed we were left with just two options- google integrated login and continue without login. The small icon for the google integrated login that can be seen in the first iteration were also decided to be too small so these were changed into thick bars that go almost the full width of the screen which should help those with issues with their visual and motor functions to use it more easily. This was the final update to the login page which left us with the page as it looks today which can be seen below.

3.10 Settings Page

Chapter 4

Testing

4.1 Internal Testing

For the internal testing of a App of this nature where multiple people are contributing their own sections it is important to have tests that both cover each person's individual contributions along with making sure that each persons sections work together without issues arising. To cover these main areas the testing was split into the commonly used Unit, Integration and System testing system which splits the testing into these 3 sections:

Unit testing, which is the testing of small components of the program, usually made by just the one person, in isolation. These tests are useful as they can often be made simple and automated so that they can be run every time a change is made to make sure nothing has been broken during the changes.[1]

Integration testing, which starts running the individual units together in bigger sections to try to uncover any issues in their interactions which cannot be seen in unit testing alone. This is especially important in an environment where multiple people are working on different units as they may of had different ideas of how the units will work together which can cause unintended behaviour in the app.[1]

Systems testing, which is the testing of the full, completed system and making sure that it meets its requirements. In our case this means testing and using the app in its entirety, in an emulator on installed on a mobile to make sure it provides the functionality required for our customer [1]

4.1.1 Unit Testing

A non-exhaustive list of some of the unit tests carried out is as follows:

Test name	Description	Expected outcome	Result
Google Sign-in button	Creating the Google sign-in button which will send you to the google integrated sign in	Screen should become greyed out and a menu should pop up listing all your Google accounts awaiting you to sign into one of them	Pass
Facebook Sign-in button	Creating the Facebook sign-in button which will send you to the Facebook integrated sign in	Should navigate you to the in-built browser to log into Facebook	Fail - Discussed below table

4.1.1.1 Facebook Sign-in Problems

Creation of the integrated Facebook log-in for the app went smoothly as the Facebook documentation explains how to do it in a simple and straightforward manner. However once we begun testing it the login simply would not work, upon researching why this is it was discovered that in October of 2021 Facebook decided to disable sign-in from embedded browsers due to a sharp increase in phishing attacks using the feature.[18]. This can be fixed by making the person log in go into their user settings on Facebook and re-enable it however this is something not all users of our app would be able to do necessarily and even if they did manage it this would allow them to be targeted by the phishing attacks Facebook disabled this feature in order to prevent. There are other ways to login using Facebook for instance if they have the Facebook app also on the device you can use the "Android App Switch" feature to login. However, as this would likely raise new issues ,such what happens if someone doesn't have the app, and the Fact that everyone on a android phone is guaranteed to have a google account we decided to instead remove the Facebook login from the Sign-in page.

4.1.2 Integration Testing

Test name	Description	Expected outcome	Result
Going Through the survey	Going through the survey so make sure each individual question navigates to the next one correctly	Should go one by one through the survey in a way that follows the layout designed in Figma	Pass
Checking the dashboard screen for past surveys	Check the dashboard screen to see that surveys you have taken in the past are displaying there	If the database has integrated with the dashboard correctly you should be able to see all your past survey results	Pass
Notification upon results being ready to view	Checking that once the results of that days surveys are ready to view for that user they are sent a notification for it on their phone.	A notification telling them about the results being ready should pop up at the top of the screen	Pass

4.1.3 System Testing

Test name	Description	Expected outcome	Result
Complete a survey to show in the dashboard	Go all the way through the survey giving answers and then navigate to the dashboard to see the results	If this works as intended a new instance should be added to the list of past surveys and should reflect the answers you gave	Pass
Logout and Login	Use the logout button in the settings menu to logout and then once you are on the login screen login with google again	If the logout worked correctly when you press the login with google button on the login page instead of it remembering you and immediately navigating to the home page it should ask you to enter you details as if it were your first time logging in.	Pass
Test the notification time frames	Go into the settings and adjust the time frames for when the app is allowed to send you notifications.	The app should send you your next notification after the no notifications time you specified is over	Pass.

4.2 External Testing

4.2.1 Ethics Approval

One of the aims that was set out as part of the project scope was to ask the public to test the end product's functionality and usability. Before submission of the ethics application form, there were a few key factors that needed to be considered: the right number of people, what group of people were we asking, how many days the person would be using the app, the types of data being displayed on the app, how we were planning on obtaining the feedback and steps on how users will be using the app.

As the testing and gathering of feedback of the application could only be done once the development of the application was completed, we decided as a team that 5-10 people would be just the right amount to document how people felt about using the app and the improvements that could be done if there were any. After deciding the number of people, we then had to decide if we were going to be asking students from a specific department such as the Occupational Therapists (OT) department, or just generally around the university campus. The OT department was considered as an option because according to studies and research done by our customer, the OT department was very much involved in the process of developing the predecessor application as they had a better understanding of patients that had sustained brain injury before.

The other option was to ask working adults, however, as this project only focused on developing an app that could aid patients, it was not required for us to assess the answers recorded by the users and give feedback based on their answers in a clinical manner. Thus, we decided to limit the scope of testing the application with general students around the university or with students from the operational therapist department.

A major part of the project involved incorporating wearables and sensors such as a FitBit hence one factor that had to be considered was how long each user could test with the FitBit as we only we had a limited supply. The proposed idea was to have users use the FitBit for over a period of five days to test whether mainly: the notification alerts would prompt the user at different times of the day and if the data from the FitBit was synchronised with the app. The data that was collected would be the users' response to the activity survey questions in the app and the users' heart rate and step count from the FitBit. (needs the full dataset that we're collecting)

Before the testing period commences, participants will be asked if they would like to partake in this study and if so, will be given a Participation Information Sheet (PIS) which contains the outline of what the study is about and why the study is being done. This allows participants to gain a better idea of how the study will be conducted before deciding to participate. If the participants are willing to participate, they will be given a consent form and it's mandatory for the user to sign before the testing period commences. Then, the participants will be taught how to navigate and operate through the app on a mobile phone. The testing period will be five days, where the user will be prompted by notification alerts throughout each day to answer the activity survey questions:

- Question One will ask the participant how they are feeling in terms of energy levels on a scale of 0-10.
- Question Two will ask where the participant is at the moment?
- Question Three will ask what the participant is doing right now? The participant must pick the closest category.
- Question Four branches into more specific activities based on the activity chosen in Question Three. (these specific activities can be seen in

Appendix #)

- Question Five will ask how effortful the activity was in Question Four take.
- Question Six will ask the participant how fatigued they are at the moment.
- Question Seven will ask the participant how much they enjoyed the activity.
- Question Eight will ask the participant how the activity affected their energy levels.
- Question Nine will ask the participant to choose the option that best describes their fatigue from physical, mental or a mixture of both fatigues.
- Question Ten will ask the participant to take an interactive reaction time test.

At the end of each day within the testing period, the participant will receive a summary of how their fatigue was affected by the activities they did throughout the day. This will be represented to the participant as a separate dashboard section of the app. The steps are then repeated throughout the testing period and once it ends, the participants will be asked to take part in a survey which allows them to provide any type of feedback regarding how they felt about the app's design, the activity questions being asked, the overall transition of the app and anything in general can be voiced through this survey.However, the testing of the app was placed at the very end of our technical objectives as the app may not be complete, and the possibility of our ethics application being rejected. If the ethics form was disapproved, there had to be time spent on revising the application and resubmitting it again. Even though the testing of the app was the final leg of the project, there was still a possibility that the ethics would be approved thus, we decided to submit it and work on the app simultaneously.

4.2.2 Setbacks

First submission was disapproved as parts of the application made the study misleading namely, the study title which was the title of this project, made it seem like participants will have experienced brain injury and was not reflecting what the study was intended to do. There was also a lack in detail explaining how participants will be wearing sensors/wearables when they take part. Along with the main ethics application form, there was a document called the Participation Information Sheet which lacked detail in which just stating that the participants were volunteers was insufficient. The ethics committee pointed out they needed to know more information about inclusion/exclusion criteria i.e, age, the link to brain injury which again was a confusion lead by the title of the study. There were other smaller adjustments that needed to be made such as the formatting of the PIS as there was a mixture of font colours, and deletable information that was not deleted. Lastly, the risk section was supposed to be in terms of the participants, but

it was written in terms of risk with the technology, however it was suggested that it may link to risks with the participants like frustration.

The ethics application was edited based on the feedback and resubmitted with the title changed to: "Evaluating the usability of a mobile application designed to help people track their levels of fatigue". This came back disapproved as well with similar feedback mentioning that the study design within the application still implies that brain injury patients will be the participants. It was advised that our sentences had to be edited in a way which mentions that the app will be developed with a non-clinical population and had applications in the treatment of brain injuries. The ethics committee also wanted the 5-10 participants from the Occupational Therapy Dept. to be made clear whether they were to be staff/students and not patients. Another area that needed more refining was the section mentioned previously in the PIS, asking for the characteristics of the participants as stating they were volunteers was insufficient. The age, additional inclusion/exclusion criteria such as gender and visual function were alluded hence it had to be made clear that gender was not an eligibility requirement at all in the PIS and participants who felt like they did not fit the criteria stated in the consent form they could choose to opt out. After editing the ethics application form, the third attempt came back successful.

In summary, the ethics application was successful after the third attempt at submission. This was mainly due to the implication of the study requiring participants to have had brain injury before in order to be eligible to participate and not being specific enough about the characteristics of participants i.e inclusion/exclusion criteria. The issue was then solved once the title was changed and the characteristics of the participants were further specified where gender was not considered as an eligibility requirement.

4.2.3 Feedback

4.2.3.1 Results

4.2.3.2 Analysis

Chapter 5

Project Specification Outcomes

5.1 Technical Objectives

5.2 Stretch Objectives

We had 5 stretch objectives for this GDP after speaking to the customer at the start. These are objectives that don't have any effect on the customers ability to conduct the research they need with out app but would make the user experience a lot more seamless. These objectives were:

1. To design and implement a long-term calendar tracking past results
2. To design and implement charts to show long-term fatigue progress and trends over varying timescales.
3. Implement additional compatibility for unusual/unreleased external devices.
4. Design and implement a new survey for significant impromptu symptom/activity recordings.
5. Design and implement an Android widget(s) to display daily summaries without opening the app.

The first objective, "To design and implement a long-term calendar tracking past results" was not achieved. The app does store the user's results long term and they can be accessed however they are shown in a list on the dashboard and there was no additional page for a nice calendar view of them implemented due to time constraints. The impact of this on the research should be minimal as the information is still there, just not presented as nicely as it would be if the calendar had been implemented. The main negative impact of this falls upon the user as they will not be able to see the long term effects of certain activities on their fatigue. This should not have too much of a detrimental impact on this app however and in some cases may be a good thing. As is noted by B.M Hood (2010) and M. Shermer (2011) people have a predisposition to draw causation from correlation and as is said by Dobelli (2013) this predisposition "leads us astray practically every day"[4][5][6]. This could be problematic for the users of this app if they see a

correlation between a particular activity and fatigue using our apps long term trends and start altering their behaviour and avoiding things in day to day life which may not actually be the cause of their fatigue. Therefore, in retrospect it is possible not having this feature is for the best as it means the only people who are likely to see the final correlations are the researchers who will be knowledgeable enough on the area to avoid making any false assertions.

The second objective is similar to the first but in a chart form rather than a calendar. This one was part implemented as the app's dashboard does show a chart of your past results however there is no option to change the timescale it shows currently and because of reasons mentioned previously it does not show the user any trends or correlations.

The third objective, similar to the last was part implemented. We were unable to implement compatibility for more than just the FitBit family of devices within time as every company has a different system so this would require completely rewriting a new copy of the all the code for the FitBit integration but for this new companies protocols and formatting. However the section of this objective about compatibility for "unreleased external devices." was implemented. This is due to the procedure mentioned in 3.7.2 where instead of using a direct connection to the device itself we instead have connected to the FitBit API and get the devices data from there. This mean even if new devices are released by FitBit, as long as they have their data uploaded to the API, our app will still be able to work with it.

The forth objective Was not completed. The main app survey was the one given to us by the customer and is the one all the results on the dashboard are based upon. Whilst the idea of having multiple surveys to keep users engaged was an appealing one at the start, due to the fact the entire app revolves around this one survey and the whole purpose of the app is to gather data using this one survey, adding another survey that people could choose to take instead quickly became unfeasible and would require extensive consultation with the customer to make sure it still collected the core data they needed.

The fifth and final objective was not achieved. Android is incredibly restrictive with stuff you are allowed to show to the user while your app is closed. Even getting the notifications done was a challenge as android likes to send the notifications out in batches and then leave the user alone before the next batch to try to prevent users being spammed by an endless stream of notifications at all hours. Widgets are, understandably, even more restricted than notifications because they become a semi-permanent fixture of your home screen. After research and realising this it was decided time was better spent elsewhere and instead it would be better to make it very quick to access the dashboard from within the app to make the widget unnecessary as the information can be accessed in around 5 seconds even without it. As is shown in previous chapters the dashboard button is on of the two main buttons on the home screen so this quick and easy access was achieved.

5.3 Gantt Chart Comparison

A lot has changed since our initial Gantt chart plan. Whilst we believe we met a large number of our technical objectives, our stretch objectives were less accomplished, although we have very much achieved a couple of them as they slowly formed part of our main goals. Our initial plan tracked a steady line of progress across the term, taking at most three weeks for each task. However, we discovered in practice that most of our development would become back-loaded. This was not intentional but we could have planned better for this. We severely underestimated the dependency created by setting up the app in a medium none of us were that familiar with. Even though we are comfortable with Java, getting used to merging XML designs and activity flow was a challenge. Shown below is the comparison of our original Gantt chart to the final plan. This data is based on our Git commit history and record logs of work done.

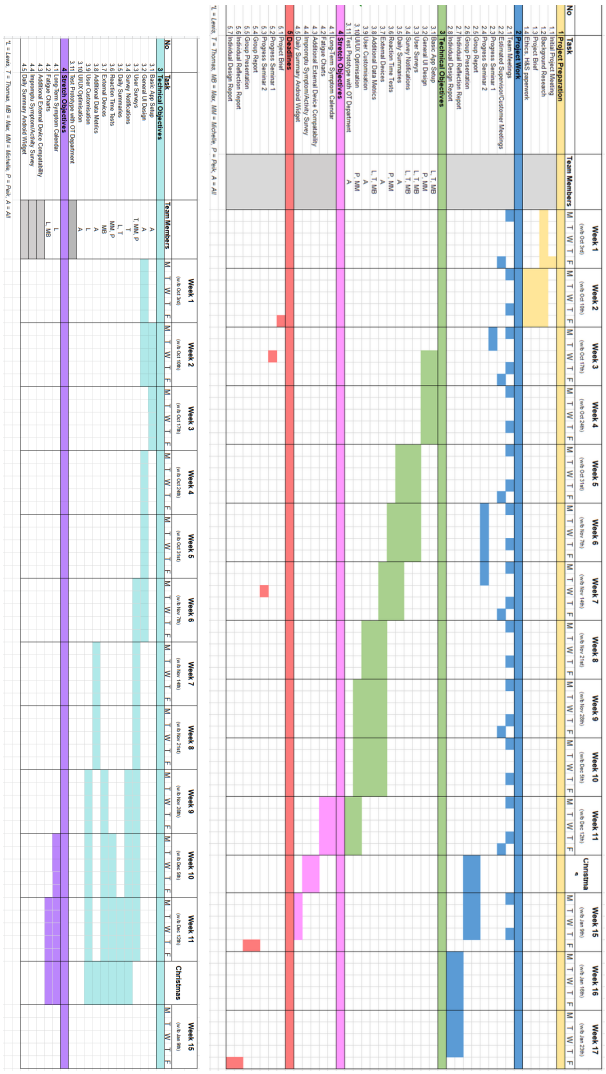

Figure 5.1: Gantt Chart Comparison

We can see from this graph that the bulk of the time was taken up by

the user surveys and initial UI design. The user surveys were a pivotal part of the app so we spent longer on them to perfect them but also took our time in polishing so the hours spent later on were less and sporadic. We also discovered that because the user surveys provided most of the data, we needed to work on this task in conjunction with another. The best example of this was setting up the database to store information that could be collected by the dashboard. The UI design also has a lot of time logged because this task evolved into multiple others such as the login page, dashboard design and settings page. The time logged under other tasks also became these pages even if not specifically planned under that name.

Another big part of the Gantt chart we can see here is the time spent on stretch objectives. We discovered in early December that having a dashboard that displayed information was progressing well and began to quickly and efficiently expand this feature to provide graphs, long term results and daily summaries. This started to come under our stretch objectives somewhat unintentionally. There are perhaps lots more things we could have planned for these stretch objectives goals but these could very easily be added onto the dashboard.

Chapter 6

Project Management

6.1 Project Planning

6.1.1 Customer Planning

In the beginning, we tried to come up with a basic layout and idea of the parts needed for our project. A key stage in this process was organising a couple of meetings with our customer to understand the progress already made and where we could take those ideas further. Through this process, we set up a Teams channel for communication and meetings where we were able to receive a few documents from our customer, including their thesis as well as one specifying the user survey contents. We found this especially useful to establish a first draft UI design and begin setting up the app following a basic structure. A later meeting helped provide critical feedback on this UI design which we then expanded on and prepared to develop fully. Our communication with our customer was quite front-loaded as once we received a direction it was quite clear where we wanted to take it, but we occasionally conversed on some expectations.

6.1.2 Group Planning

Our group planning was established from our first in-person meeting. We set up a Discord group to provide an informal place to call, share files and discuss ideas. This platform became an essential part of our project planning and we all used this incredibly frequently. We discovered during the first month or two that in-person meetings were going to be tricky with one person having Covid for a week and another person spraining their ankle making getting onto the University campus difficult. During this time we realised our reliance on Discord. Meetings became much more regular and more impromptu.

6.1.3 Code Planning

As discussed during the development setup section, we used a private GitHub repository to all work collaboratively. To make sure we didn't conflict with our work, we set up meetings to organise who was working on what and created our branches to work on. When we finished that feature, we

would pull, squash and merge that branch into the master whilst everyone else updated their branches with the new changes. This meant we could work collaboratively but also separately without breaking other people's builds. The usage of branches meant we could be very aware of everyone else's progress and work planned.

6.2 Division of Responsibilities

6.2.1 Technical Contribution

6.2.2 Report Authorship

Section	Subsection	Contributor	Word Count
Introduction	Problem Statement	-	0
”	Solution	-	0
”	Collaboration	-	0
”	Problem Statement	-	0
Design	Terminology	-	0
”	UI Design	-	0
”	Reaction Time Test	-	0
”	Data Dashboard	-	0
”	Settings Page	-	0
Development	Development Setup	-	0
”	User Surveys	-	0
”	Reaction Time Tests	-	0
”	Notifications	-	0
”	Data Dashboard	-	0
”	Daily Summary	-	0
”	FitBit Integration	-	0
”	Storing the Data	-	0
”	Login Page	-	0
”	Settings Page	-	0
Testing	Internal Testing	-	0
”	External Testing	-	0
Spec Outcomes	Technical Objectives	-	0
”	Stretch Objectives	-	0
”	Gantt Chart Comparison	-	0
Management	Project Planning	-	0
”	Division of Responsibilities	-	0
”	Risk Assessment Outcomes	-	0
”	GDPR and Ethics Compliance	-	0
Conclusion	Conclusion of Project	-	0
”	Future Work	-	0
Total	-	-	0

6.2.3 Poster and Presentation Contribution

6.3 Risk Assessment Outcomes

6.4 GDPR and Ethics Compliance

The University of Southampton has strict policies, processes and guidance that must be adhered to in order for the University to be compliant when it comes to data protection. The Data Protection Act(DPA) is a law designed to protect personal data stored on computers or in an organised filing system.

According to the Information Commissioner's Office(ICO), it describes personal data as information about any particular living individual. This could include anyone, including a customer, client, employee, partner, member, supporter, business contact, public official or member of the public[15]. Personal data does not have to be 'private' information and if the information includes information that could make an individual identifiable, it still counts as personal data. This includes: name, location details, email address, date of birth, online identifiers such as mobile device IDs or IP addresses and any other identifiers including physical, genetic, economic, mental or cultural. Special category personal data covers more sensitive information which includes: race or ethical origin, political origin, religious beliefs, Trade Union memberships, health data and sexual orientation [17]. The ICO also states that the law applies to any 'processing of personal data'. Processing refers to anything to do with data including collecting, recording, storing, using, analysing, combining, disclosing, or deleting it.

It is University policy that ANY study involving human participants, questionnaires, lab studies, field observations and more, must be submitted for ethics review. As such in our project, we needed to collect feedback in the form of a survey once the app was complete to test the usability and the functionality of the app. This process required collection of some personal data such as their names and university email addresses. Therefore, in order to be compliant with the University's Data Protection Policy, we submitted an ethics application demonstrating how informed consent will be communicated, to identify risks and demonstrate how these risks will be addressed, how data would be gathered and protected. The university also suggests the following to help with compliance: appropriately create, store, use, share, archive and destroy personal data using the following methods such as the following: using password protected computers and documents, encrypt personal information, make back-up copies and secure them securely, log off or lock your computer, report suspicious cyber incidents and shred confidential waste. The process of submitting the ethics application is detailed in the external testing in Chapter 4 of this report.

Chapter 7

Conclusion

7.1 Conclusion of Project

7.2 Proposal of Future Work

There is a few areas within this app that could be expanded upon were we given more time or can be done by a future group working on this topic.

The most obvious area of extension would be to add support for more potential wearable devices to be added. This may be done by either adding integration for other wearable fitness devices besides Fitbit, such as Garmin or an Apple watch, Or designing a wearable from scratch that is designed specifically for the fatigue tracking app. Another option that sticks with the theme of getting more data into the app would be to try to integrate the app with other common apps used to track various fitness and activity metrics such as "My Fitness Pal" or "Strava" each of which have millions of users ¡add source¿ and would have the added benefit of keeping all the tracking within the phone removing the need for any wearable devices. This could make any research using the app more convenient and save money for the researchers as they don't have to make sure everyone taking part in The study has one of the wearables.

A topic that would also be good for extension would be to create some sort of server system that could store all of the survey result data that the researchers could access. This could make it a lot easier for them to gather data and allow them to more easily monitor the participants process while the trial is ongoing. This would also make life easier for the participants as once they got set up for the trial they might not ever have to meet the researchers again in person which lifts a lot of the geographic restraints normally associated with running the trials. If this was combined with the last potential piece of future work of integrating this app with other exercise tracking apps then this would lift both the equipment restriction and the location restriction which could make research using the app incredibly scale-able. The only thing that would limit study size at that point would be finding enough willing volunteers.

If there is future work done in this area one area that could also be expanded could be the survey format itself. At the moment it is the same survey every single time which may cause participants to get bored and may harm participation and engagement levels over the long term. There would be multiple ways to do this such as having a larger set of questions they are only asked a subset of each time. The app could also then be designed to ask the questions which get the most varied answers from the user. This is because these questions are likely to be the ones that both keep the user the most engaged and provide the most information for the researchers as answering the same question in the same way numerous times both disengages users and doesn't give the researchers much to work with.

Bibliography

[1] IEEE. "IEEE Standard Glossary of Software Engineering Terminology". In: (1990), pp. 7–74. DOI: https://doi.org/10.1109/IEEESTD.1990.101064.

[2] G. Kinsella et al. "Everyday memory following traumatic brain injury". In: (1996).

[3] J.L. Mathias et al. "Prospective and declarative memory problems following moderate and severe traumatic brain injury". In: (2009).

[4] Bruce M. Hood. *The Science of Superstition : How the Developing Brain Creates Supernatural Beliefs*. 2010.

[5] Michael Shermer. *The believing brain: from ghosts and gods to politics and conspiracies ; how we construct beliefs and reinforce them as truths*. 2011.

[6] Rolf Dobelli. *The Art of Thinking Clearly: Better Thinking, Better Decisions*. 2013.

[7] Centers for Disease Control and Prevention. "Traumatic brain injury surveillance report". In: (2019).

[8] S. Dixon. "Main social networking site preference in the United Kingdom (UK) 2020, by age". In: (2022).

[9] Apple. *Apple Watch SE Features*. URL: https://www.apple.com/uk/apple-watch-se/.

[10] Apple. *Shop Apple Watch*. URL: https://www.apple.com/uk/shop/buy-watch.

[11] Fitbit. *Fitbit Store*. URL: https://www.fitbit.com/global/uk/products/trackers.

[12] Google. *Provide custom back navigation - Android Developers*. URL: https://developer.android.com/guide/navigation/navigation-custom-back#:~:text=All%5C%20Android%5C%20devices%5C%20provide%5C%20a,button%5C%20or%5C%20a%5C%20software%5C%20button.

[13] Google. *Save data in a local database using Room - Android Developers*. URL: https://developer.android.com/training/data-storage/room.

[14] Headway. *Acquired Brain Injury: The Numbers Behind the Disability*. URL: https://www.headway.org.uk/about-brain-injury/.

[15] licensed under the Open Government Licence v3.0. Information Commissioner's Office [Guide to Data Protection 14th October 2022]. *Some basic concepts*. URL: https://ico.org.uk/for-organisations/guide-to-data-protection/introduction-to-dpa-2018/some-basic-concepts/.

[16] Jovan. *Smartwatch Market Share: The World's Top Brands*. URL: https://kommandotech.com/guides/smartwatch-market-share/.

[17] University of Southampton. *Data Protection and GDPR*. URL: https://www.southampton.ac.uk/isolutions/staff/data-protection-and-gdpr.page.

[18] Julie Yuan. *authentication on Android embedded browsers*. URL: https://developers.facebook.com/blog/post/2021/06/28/deprecating-support-fb-login-authentication-android-embedded-browsers/.

Appendix A

Original Project Brief

Each year in the UK over 300,000 are admitted to hospital because of a brain injury. Around half will experience troublesome and persistent fatigue that prevents their resumption of work, social and leisure activities. Managing fatigue and the impact of fatigue on daily life is difficult, partly because fatigue fluctuates and can be difficult to predict, but also because people with brain injury may have cognitive challenges that affect their ability to keep track of and learn about their fatigue. To address this problem, we developed a working prototype app (android) that was tested by people with brain injury. The prototype included repeated surveys where people report their fatigue and activity at intervals across the day, sampled ambient noise and physical activity transitions, and delivered a short reaction time test (as an indicator of mental fatigue). A new prototype is needed to address design issues and further develop the app. These might include: to integrate data from external devices such as Fitbits, to provide daily summaries of the users survey responses, reaction time scores, and physical activity and sleep data from Fitbit, to allow some customisation by the user, to provide an enhanced short reaction time test to increase user engagement, track environmental information such as noise, possibly capture phone use data that might indicate fatigue – a slowing of response or increased errors on tasks.

Appendix B

Brain Fatigue App Code

Not all the code is available in this section, only those that we have used to reference during the development chapter.

NotificationReceiver.java

```
public class NotificationReceiver extends BroadcastReceiver {
    private int NOTIFICATION_ID = 3;

    @Override
    public void onReceive(Context context, Intent ignore) {
        // Create a notification intent
        Intent intent = new Intent(context,
            SurveyStartActivity.class);
        intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        TaskStackBuilder stackBuilder = TaskStackBuilder.
            create(context);
        stackBuilder.addNextIntentWithParentStack(intent);
        PendingIntent pendingIntent = stackBuilder.
            getPendingIntent(0, PendingIntent.
            FLAG_UPDATE_CURRENT);

        // Build the notification
        NotificationCompat.Builder builder = new
            NotificationCompat.Builder(context, "")
                .setSmallIcon(R.drawable.
                    ic_notification_vector)
                .setContentTitle("Brain␣Fatigue␣Tracker")
                .setContentText("It's␣time␣to␣take␣the␣survey!
                    ")
                .setDefaults(Notification.DEFAULT_VIBRATE)
                .setPriority(NotificationCompat.PRIORITY_HIGH)
                .setCategory(NotificationCompat.CATEGORY_CALL)
                .setFullScreenIntent(pendingIntent, true)
                .setAutoCancel(true)
                .setOngoing(true);

        // Display the notification
        NotificationManager notificationManager = (
            NotificationManager) context.getSystemService(
            Context.NOTIFICATION_SERVICE);
        notificationManager.notify(NOTIFICATION_ID, builder.
            build());
```

```
            NOTIFICATION_ID++;
        }
    }
```

SurveyEndActivity.java

```
public class SurveyEndActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_survey_end);

        // Hide action bar
        if (getSupportActionBar() != null)
            getSupportActionBar().hide();

        // Database access
        ExecutorService executorService1 = Executors.
            newSingleThreadExecutor();
        Future<Setting> futureSetting = executorService1.
            submit(() -> {
            FatigueDatabase fatigueDatabase = FatigueDatabase.
                getDatabase(getApplicationContext());
            SettingsDao settingsDao = fatigueDatabase.
                settingsDao();

            List<Setting> settings = settingsDao.getAll();
            Setting setting;
            try {
                setting = settings.get(settings.size() - 1);
            } catch (Exception e) {
                setting = new Setting();
            }
            return setting;
        });
        executorService1.shutdown();

        // Schedule the next notification
        long timeout = System.currentTimeMillis() + 10000;
        Setting resultSetting;

        while (System.currentTimeMillis() < timeout) {
            try {
                resultSetting = futureSetting.get();
            } catch (ExecutionException | InterruptedException
                e) {
                continue;
            }

            long scheduledNotification = scheduleNotification(
                System.currentTimeMillis(), resultSetting.
                getInterval(), resultSetting.getDayStart(),
                resultSetting.getDayEnd(), resultSetting.
                getWorkStart(), resultSetting.getWorkEnd());

            // Notify the user of the next notification
            SimpleDateFormat sdf = new SimpleDateFormat("HH:mm
```

```
            ", Locale.UK);
        String submitPromptString = "To␣finish,␣press␣
            submit.␣The␣next␣survey␣prompt␣will␣arrive␣at␣"
             + sdf.format(System.currentTimeMillis() +
            scheduledNotification) + ".";

        final TextView submitPromptText = findViewById(R.
            id.endSurveySubMessage);
        submitPromptText.setText(submitPromptString);

        // Next button
        final Button surveySubmitBtn = findViewById(R.id.
            activity_survey_end_submit_button);
        surveySubmitBtn.setOnClickListener(v -> {
            ExecutorService executorService2 = Executors.
                newSingleThreadExecutor();
            executorService2.execute(() -> {
                FatigueDatabase fatigueDatabase =
                    FatigueDatabase.getDatabase(
                    getApplicationContext());
                SurveyResultDao surveyResultDao =
                    fatigueDatabase.surveyResultDao();

                SurveyResult surveyResult = (SurveyResult)
                     getIntent().getSerializableExtra("
                    survey_result");
                surveyResultDao.insert(surveyResult);
            });
            executorService2.shutdown();

            // Setup the next notification
            Intent notifyIntent = new Intent(
                getApplicationContext(),
                NotificationReceiver.class);
            PendingIntent pendingIntent = PendingIntent.
                getBroadcast(getApplicationContext(), 0,
                notifyIntent, PendingIntent.
                FLAG_UPDATE_CURRENT | PendingIntent.
                FLAG_IMMUTABLE);
            AlarmManager alarmManager = (AlarmManager)
                getSystemService(Context.ALARM_SERVICE);
            alarmManager.setExactAndAllowWhileIdle(
                AlarmManager.RTC_WAKEUP, System.
                currentTimeMillis() + scheduledNotification
                , pendingIntent);

            Intent intent = new Intent(SurveyEndActivity.
                this, HomeActivity.class);
            intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP
                );
            startActivity(intent);
        });
        break;
    }
}

public Long scheduleNotification(Long startTime, Long
    interval, Long dayStart, Long dayEnd, Long workStart,
```

```
        Long workEnd) {
         long remainingTime = interval;
         long alarmTime = startTime;

         while (remainingTime > 0) {
             if (alarmTime >= dayStart && alarmTime < workStart
                 ) {
                 long gap = workStart - alarmTime;
                 if (gap > remainingTime) {
                     alarmTime += remainingTime;
                     break;
                 } else {
                     alarmTime = workEnd;
                     remainingTime -= gap;
                 }
             } else {
                 long gap = dayEnd - alarmTime;
                 if (gap > remainingTime) {
                     alarmTime += remainingTime;
                     break;
                 } else {
                     alarmTime = dayStart;
                     remainingTime -= gap;
                 }
             }
         }

         return alarmTime;
     }
}
```

SurveyResult.java

```
@Entity
public class SurveyResult implements Serializable {
    @PrimaryKey
    public Long surveyResultId; // Corresponds to the time (in
        milliseconds) the survey is started
    @ColumnInfo(name = "question1")

    public Integer question1; // 0 to 100 in 5 point intervals
        (slider)
    @ColumnInfo(name = "question2")
    public Integer question2; // 1 or 2
    @ColumnInfo(name = "question3")
    public Integer question3; // 1 to 7 (should be -1 of
       question 2 = 2)
    @ColumnInfo(name = "question3_extended")
    public Integer question3Extended; // 1 to 5, 6 to 13, 14
       to 15, 16 to 18, 19 to 24, 25 to 27, 28 to 32 (range
       corresponds to value in question 3, should be -1 of
       question 2 = 2)
    @ColumnInfo(name = "question4")
    public Integer question4; // 1 to 8 (should be -1 of
       question 2 = 1)
    @ColumnInfo(name = "question4_extended")
    public Integer question4Extended; // 1 to 3, 4 to 7, 8 to
       10, 11 to 15, 16 to 20, 21 to 26, 27 to 33, 34 to 39 (
```

```
    range corresponds to value in question 4, should be -1
    of question 2 = 1)
@ColumnInfo(name = "question5")
public Integer question5; // 0 to 10 in 1 point intervals
    (slider)
@ColumnInfo(name = "question6")
public Integer question6; // 0 to 10 in 1 point intervals
    (slider)
@ColumnInfo(name = "question7")
public Integer question7; // 0 to 10 in 1 point intervals
    (slider)
@ColumnInfo(name = "question8")
public Integer question8; // 1 to 4
@ColumnInfo(name = "question9")
public Integer question9; // 1 to 3

public SurveyResult(Long surveyResultId) {
    this.surveyResultId = surveyResultId;
    this.question1 = -1;
    this.question2 = -1;
    this.question3 = -1;
    this.question3Extended = -1;
    this.question4 = -1;
    this.question4Extended = -1;
    this.question5 = -1;
    this.question6 = -1;
    this.question7 = -1;
    this.question8 = -1;
    this.question9 = -1;
}

@Override
public String toString() {
    return "SurveyResult{" +
            "surveyResultId=" + surveyResultId +
            ",␣question1=" + question1 +
            ",␣question2=" + question2 +
            ",␣question3=" + question3 +
            ",␣question3Extended=" + question3Extended +
            ",␣question4=" + question4 +
            ",␣question4Extended=" + question4Extended +
            ",␣question5=" + question5 +
            ",␣question6=" + question6 +
            ",␣question7=" + question7 +
            ",␣question8=" + question8 +
            ",␣question9=" + question9 +
            '}';
}

public Long getSurveyResultId() {
    return surveyResultId;
}

public String getSurveyResultIdAsString() {
    // Convert the id for this survey result to a date and
        time
    SimpleDateFormat sdf = new SimpleDateFormat("d/M␣HH:mm
        ", Locale.UK); // Should be d/M/yy to 'year-proof'
```

```
            dates
        GregorianCalendar calendar = new GregorianCalendar(
            TimeZone.getTimeZone("Europe/London"));
        calendar.setTimeInMillis(this.getSurveyResultId());
        return sdf.format(calendar.getTime());
    }
```

EmergencyService.java

```
public class EmergencyService extends Service {
    private static final int NOTIFICATION_ID = 2;

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int
        startId){
        ExecutorService executorService = Executors.
            newSingleThreadExecutor();
        executorService.execute(() -> {
            try {
                MediaRecorder mediaRecorder = new
                    MediaRecorder();
                mediaRecorder.setAudioSource(MediaRecorder.
                    AudioSource.MIC);
                mediaRecorder.setOutputFormat(MediaRecorder.
                    OutputFormat.THREE_GPP);
                mediaRecorder.setAudioEncoder(MediaRecorder.
                    AudioEncoder.DEFAULT);
                mediaRecorder.setOutputFile(
                    getExternalCacheDir().getAbsolutePath() + "
                    /audio.3gp");

                mediaRecorder.prepare();
                mediaRecorder.start();

                Timer timer = new Timer();
                timer.scheduleAtFixedRate(new RecorderTask(
                    mediaRecorder), 0, 500);
            } catch (IOException | RuntimeException e) {
                e.printStackTrace();
                Log.d("Audio", "Fail:␣" + e);
            }
        });

        startForeground();
        return super.onStartCommand(intent, flags, startId);
    }

    private class RecorderTask extends TimerTask {
        private final MediaRecorder mediaRecorder;
        private long timeout = System.currentTimeMillis();

        public RecorderTask(MediaRecorder mediaRecorder) {
```

```
        this.mediaRecorder = mediaRecorder;
    }

    public double getAmplitude() {
        if (mediaRecorder != null) {
            return mediaRecorder.getMaxAmplitude();
        } else {
            return 0;
        }
    }

    public void run() {
        double amplitude = 20 * Math.log10(getAmplitude()
            / 32768);

        if (amplitude > -5) {
            Log.d("Audio", "Pass: " + amplitude);

            if (System.currentTimeMillis() > timeout) {
                timeout = System.currentTimeMillis() +
                    10000;
                // Create a notification intent
                Intent intent = new Intent(
                    getApplicationContext(),
                    SurveyStartActivity.class);
                intent.setFlags(Intent.
                    FLAG_ACTIVITY_CLEAR_TOP);
                TaskStackBuilder stackBuilder =
                    TaskStackBuilder.create(
                    getApplicationContext());
                stackBuilder.addNextIntentWithParentStack(
                    intent);
                PendingIntent pendingIntent = stackBuilder
                    .getPendingIntent(0, PendingIntent.
                    FLAG_UPDATE_CURRENT);

                // Build the notification
                NotificationCompat.Builder builder = new
                    NotificationCompat.Builder(
                    getApplicationContext(), "")
                        .setSmallIcon(R.drawable.
                            ic_notification_vector)
                        .setContentTitle("Brain Fatigue
                            Tracker")
                        .setContentText("We detected a
                            significant jump in volume!")
                        .setDefaults(Notification.
                            DEFAULT_VIBRATE)
                        .setPriority(NotificationCompat.
                            PRIORITY_HIGH)
                        .setCategory(NotificationCompat.
                            CATEGORY_CALL)
                        .setFullScreenIntent(pendingIntent
                            , true)
                        .setAutoCancel(true)
                        .setOngoing(true);

                // Display the notification
```

```
                    NotificationManager notificationManager =
                        (NotificationManager)
                        getApplicationContext().
                        getSystemService(Context.
                        NOTIFICATION_SERVICE);
                    notificationManager.notify(2, builder.
                        build());
                }
            }
        }
    }

    private void startForeground() {
        Intent notificationIntent = new Intent(this,
            MainActivity.class);

        PendingIntent pendingIntent = PendingIntent.
            getActivity(this, 0,
                notificationIntent, PendingIntent.
                    FLAG_IMMUTABLE);

        startForeground(NOTIFICATION_ID, new
            NotificationCompat.Builder(this, "BrainFatigueApp")
                .setSmallIcon(R.drawable.
                    ic_notification_vector)
                .setContentIntent(pendingIntent)
                .setDefaults(Notification.FLAG_AUTO_CANCEL)
                .build());
    }
}
```