

University of Plymouth

School of Engineering,
Computing, and Mathematics

PRCO304

Final Stage Computing Project
2019/2020

MoodSage – Mood and Worry Tracking

Edward Gavin

10555083

BSc (Hons) Computer Science

Acknowledgements

I would like to thank my project supervisor, James Hayter for his guidance and help over the course of this project, his feedback was invaluable and helped the project become the best it could be.

I would also like thank my friends, family and girlfriend for their support throughout the entirety of this project, as well as my degree.

Abstract

This report describes a software project involving the development of an application intended for use in the field of Mental Health. It is focussed on both, supporting those suffering from anxiety and the Mental Health Professional helping clients on a journey of empowerment and recovery. The application is designed to provide the client with the ability to share personal data with a therapist who can monitor, analyse and advise them and aid them in understanding their feelings during regular one to one sessions.

The background of the project, in terms of the current market for this application is discussed whilst the current mental health situation in the UK is explored. The capabilities and limitations of current solutions drives the requirement for a more relevant tool from which client and Mental Health Professional can benefit. The legal, social and ethical considerations are stressed, and the report clearly identifies and defines how the potential issues are to be addressed.

The functional requirements of the application are outlined and are followed by a look into the overall design and architecture of the system, as well as the chosen technologies and method of approach.

The development process, going over the five main stages of this applications development is defined and discussed, presenting an overview of the challenges faced during the development phase and changes to the requirements which happened over time. This is complimented by an overview of how the product was tested, ensuring quality in the final version.

The report concludes with an evaluation, initially reflecting on how well the project met the objectives which were set out in section one, then discussing the architecture of the system, how the decisions made effected the final project and what changes could be made in a future version.

The author has included a set of appendices, referenced throughout the document, offering fidelity detail and information on the issues discussed within.

Contents

Acknowledgements	1
Abstract	1
1. Introduction	5
2. Background	5
2.1. Why Mood Tracking?	5
2.2. Competition.....	5
2.2.1 Overview	5
2.2.2 Limitations	6
2.3. Objectives	6
3. Legal, Social and Ethical Problems	8
3.1. Legal.....	8
3.1.1. General Data Protection Regulation.....	8
3.1.2. Licensing	9
3.2. Social and Ethical	9
4. Requirements	10
4.1. Functional Requirements.....	10
4.1.1. Core Requirements (MVP)	10
4.1.2 Desirable Requirements	11
4.1.3 Optional requirements	11
4.2 Non-Functional Requirements	11
4.2 Use Cases	12
5. Chosen Development Technologies.....	13
5.1. Mobile Application.....	13
5.2. Server and Database	13
5.3. Web Application.....	13
6. Design, Architecture and Implementation.....	14
6.1. User Interface Design	14
6.2. Design Principles and Patterns.....	15
6.2.1. SOLID	15
6.2.2. Dependency Injection.....	16
6.2. Server	16
6.3. Database	17
6.4. Mobile Application.....	18
6.5. Web Application.....	20

7. Method of Approach	22
7.1. Agile.....	22
7.2. Scrum	22
8. Project Management	23
8.1. Trello.....	23
8.2. Version Control	23
9. Development	25
9.1. Stage 1 - Research and Requirements.....	25
9.2. Stage 2 – Authentication and Account functionality	25
9.3. Stage 3 – Core functionality	25
9.4. Stage 4 – Desirable features	26
9.5. Stage 5 – Final touches and Report	27
10. Testing	28
10.1. Unit Testing.....	28
10.2. Test Driven Development and CICD.....	28
10.3. Usability Testing	29
10.4. Functional Testing	29
11. End-Project Report.....	30
11.1 Summary	30
11.2 Objectives Review	30
11.1. Changes	31
12. Project Post-Mortem.....	32
12.1 Objectives Evaluation	32
12.2 Technologies Evaluation.....	32
12.3 Developer Performance Evaluation	32
12.4 Project Management Evaluation	32
12.5 The Future	33
13. Conclusion	33
References.....	34
Appendix A. – User Guides	37
Appendix B. Project Management.....	49
Appendix C. SWOT Analysis.....	59
Appendix D. Third Party Resources Used.....	60
Appendix E. Logo.....	61
Appendix F. UI Design Breakdown	62

Appendix G. Initial Designs	64
Appendix H. API Routes.....	68
Appendix I. User Stories.....	69
Appendix J. Usability Testing Results	70
Appendix K. Functional Testing.....	76

Word Count: 10000

Links to code:

Mobile and web Application: <https://github.com/EdGavin98/MoodTracker>

Server: <https://github.com/EdGavin98/MoodTracker-Server/>

1. Introduction

Since 1993 there has been an upwards trend in the number of common mental health disorders, such as anxiety and depression, among adults within the United Kingdom (Baker, 2020). A study in 2017 showed that 12.8% of 5 to 19 year olds had at least one mental condition, with emotional disorders being the most common (Sadler, et al., 2017). On top of this, there is an issue with wait times for therapy; for instance, those who seek therapy with the NHS or associated services, can be left waiting for up to 6 weeks for their first appointment, with 12.7% of patients waiting up to 18 weeks (Community & Mental Health Team, 2019).

These statistics show a need for new tools and methods which can take advantage of the prevalence and accessibility of modern technology to give potential clients the tools they need to either help themselves, or to work with a therapist or other trusted person to improve their situation and gain a better understanding of why they feel the way they do.

The system presented in this report, known as MoodSage, aims to provide a solution to the problem by producing an Android application which can be used by those dealing with emotional difficulties to track how they feel and reflect upon it. It will allow the client to share the data with a therapist who can view it through a web application. However, it will also work without a therapist, so those who are stuck on a waiting list, or those who are just looking to try and help themselves will still be able to use it. The system will primarily focus on anxiety-based conditions.

2. Background

2.1. Why Mood Tracking?

Mood tracking and journaling is a common technique used with Cognitive Behavioural Therapy (CBT). Often this is performed by a therapist giving the client a sheet to fill out in between sessions, which is then discussed with the client the next time they meet. Digitalising this format could allow for clients and therapists to better understand the data which has been collected, as well as providing an opportunity for medical professionals and clients to use these techniques outside of sessions.

2.2. Competition

To complement the information discussed in this section, a SWOT analysis is available in Appendix C.

2.2.1 Overview

The concept of mood and worry tracking has been around for a while now and because of this there are many potential competitors already available for clients to download or use online, covering a range of potential techniques. Two of these are:

- Daylio
 - Daylio is a self-care bullet journal with mood tracking functionality (Daylio, 2020). Its primary focus is on self-help and allowing clients to better understand why they felt the way they did. It is one of the most

popular applications available within this category with over five million downloads (Google, 2020).

- SilverCloud
 - SilverCloud is an online CBT course which is available for people with mental health problems. Taking the form of an 8 week course, it allows the therapist to set tasks and work the client needs to go through, and further allows for therapist/client check in's where progress can be monitored and appropriate courses of action to best fit the client's needs can be adjusted (NHS, 2018).

2.2.2 Limitations

These applications do have limitations which this solution intends to resolve. For example, Daylio is primarily self-help oriented, and as such is much focused on just making sure the client can use the data. This means, those who are in therapy while using the application do not have an easy process of sharing their data with a therapist. This is because the client can only show the therapist the application during a therapy session, there is no opportunity for the therapist to be able to view the data between sessions and merely having the mobile app interface doesn't allow the therapist to dig deeper into the information and gain a better understanding of the client's challenges.

On the other hand, SilverCloud is a therapist monitored application and gives therapists a large opportunity to interact with the client, however, it still has limitations. SilverCloud uses a course-based structure and because of this, clients can only use it for a short amount of time. It is also only available through referral from the NHS or other organisations such as universities (NHS, 2018). These limitations mean that while it can provide help to those who need it, it still faces some of the pre-existing problems, such as the wait times for therapists and it doesn't provide clients with the ability to consistently self-track before and after therapy.

During further research into this market, more limitations of a wider array of applications were revealed. While many of the applications which exist on the market have a wide array of features for collecting data and allowing clients to reflect on it, they commonly lack features which give clients the opportunity to act on their feelings as well (Caldeira, et al., 2018); whether that be by sharing the data with someone or giving them advice or facilities that allow them to think about solutions to their problems. This is a problem as it means while many of the potential applications on the market help clients understand how they are feeling, they do not give them the help and guidance they need in acting on this new knowledge.

2.3. Objectives

The aim of this project is to create an application which can be used to help people understand their moods and worries, as well as share that information with a therapist. As such, the objectives that have been set out for it are:

- To research cognitive behavioural therapy techniques which could aid in the effectiveness of the application.

- To build an application which allows client to be able to track and reflect upon their moods and worries.
- To build an application which allows therapists to be able to view the data of any clients they may have.
- To design a user interface in a way which actively helps the client feel better.
- To create a way for clients to share this information with a therapist in a way they feel comfortable with.
- To follow agile project management techniques and build the application within the time allocated.

3. Legal, Social and Ethical Problems

3.1. Legal

3.1.1. General Data Protection Regulation

General Data Protection Regulation (GDPR) are the guidelines which exist in European Union (EU) law, defining how information should be stored, protected and processed. It applies to any application which processes the data of EU residents and as such, it must be considered when developing this application. GDPR is made up of 7 key principles (Information Commissioner's Office, 2020), these are:

- Lawfulness, fairness and transparency. There must be a lawful basis for collection of data and client data must not be used in an unexpected or misleading way.
- Purpose limitation. It must be clear what the data will be used for at the start and all purposes need to be documented. Data can only be used for a new purpose if you have consent from the client or if it is compatible with the original purpose
- Data minimisation. Data must be sufficient and relevant, limited only to what is necessary
- Accuracy. Steps must be taken to ensure that all personal data is accurate and remains so.
- Storage limitation. Data must not be stored for longer than needed and must be periodically reviewed.
- Integrity and confidentiality. Appropriate measures must be in place to protect the personal data that is held.
- Accountability. The product owner must take responsibility for what they do with personal data

The application complies with these by only storing that information which is needed. Personal details are limited to the client's name, which is used for personalisation within the application and to allow clients and therapists to identify each other, as well as their email which is used as a unique identifier for logging in and for adding an account link in an identifiable way. When a client account is deleted, all records associated with them are removed from the system as well, ensuring the data is kept only whilst required. Validation is also in place on the front and back end of the system, ensuring that client information stored in the system is correct and accurate.

Security is also a key point of the application, all API requests are sent over HTTPS so that data is encrypted while in transit and authentication tokens are also in use, with roles based authentication on all routes which could contain potentially sensitive data, ensuring that the system will only share data with those who have the required permissions. Passwords are also hashed and salted within the database using bcrypt so the clients' credentials will remain secure.

GDPR also advises the use of a Data Protection Officer (DPO), who can monitor data usage and ensure data is being used in accordance with the rules set out, for applications which deal with a large amount of client data. However, as this

application is only small now, this is not required, but it would be implemented if the application grew and gained a large client base.

3.1.2. Licensing

When developing an application, it is important to make sure the developer has the permissions required to use third-party assets. All the libraries, code and assets which are in use in this application either have licenses that allow for reuse or have had explicit permission granted for their reuse in this project. A full list of third-party resources in use can be seen in Appendix D.

3.2. Social and Ethical

There are several ethical and ethical concerns for self-tracking applications, especially regarding those which deal with a client's physical or mental health. Tamar Sharon (2017) broke down three common social and ethical debates regarding self-tracking, these are:

- Empowerment vs Surveillance and Discipline. This debate discusses whether self-tracking applications empower the client, allowing them to take more control over themselves and become an active participant in their own care rather than the receivers of advice, or whether they act as tools to monitor the client, increasing surveillance and medical overreach in places which they shouldn't. This project aims to address this by giving the client control over the account links, letting them remove it at any time and giving them the opportunity to decide what data they want to share.
- Improved Overall Health vs. The Disintegration of State and Collective Responsibility for Health. This debate focuses on the shift to self-tracking, whether it gives an ability to improve the overall health of the general population, or whether the expectation that citizens should care more for themselves is replacing the notion of state funding for social support and healthcare. However, this is addressed within the project, by giving clients the opportunity to still share that data with a therapist in a way which lets them see the detail.
- Greater (Self-) Knowledge vs. Reductionism and the Non-Impartiality of Numbers. The final debate mentioned discusses whether the data gained from these applications will help the client monitor their health better, or whether reducing everything down to numerical data will cause these applications to define the client's definition of health and potentially mislead them in some way. This is addressed by the application not only collecting numerical data, but also giving clients the opportunity to provide qualitative data by describing how they feel and providing reflection options which focus on their own thoughts, rather than just pure quantitative data.

4. Requirements

4.1. Functional Requirements

At the beginning of development, a set of functional requirements were created for the project which matched the objectives set out, these were divided up into the categories of Core, Desirable and Optional. Requirements in the core category represent the minimum viable product which would allow for the product to reach the objectives set out. Desirable requirements are those which would be nice to have should the Minimum Viable Product (MVP) be reached on time, and the optional requirements which are those most likely not to be included within the project but could be added to a potential future version. More potential features were added during the development of the project however, these were mostly added to the Desirable and Optional categories.

4.1.1. Core Requirements (MVP)

- Client/Patient Application:
 - A client should be able to register for the mobile application.
 - A client should be able to log in to the application.
 - A client should be able to log out the application.
 - A client should be able to delete their account.
 - A client should be able to log a mood in the application.
 - A client should be able to view a calendar showing their mood for each day.
 - A client should be able to log a worry in the application.
 - A client should be able to see all the worries they have logged.
 - A client should be able to filter the worries by date.
 - A client should be able to filter the worries by type.
 - A client should be able to filter the worries by severity.
 - A client should be able to accept a therapist
 - A client should be able to reject a therapist
 - A client should be able to remove an account link at any point.
 - A client should be able to view data while offline.
 - A client should be able to log data while offline.
- Therapist Application:
 - A therapist should be able to register for the web application.
 - A therapist should be able to log in to the web application.
 - A therapist should be able to see all their patients
 - A therapist should be able to see any pending patients.
 - A therapist should be able to add a patient.
 - A therapist should be able to remove a pending patient.
 - A therapist should be able to remove a linked patient.
 - A therapist should be able to view graphs for all a patient's data.
 - A therapist should be able to view graphs for the past month of a patient's data.
 - A therapist should be able to see a list of all a patient's moods.
 - A therapist should be able to see a list of all a patient's worries.

- Server:
 - The server should store client data.

4.1.2 Desirable Requirements

- Client/Patient Application:
 - A client should be able to edit their profile.
 - A client should be able to set a mood target.
 - A client should be able to set a worry target.
 - A client should be able to see if they have reached their mood target.
 - A client should be able to see if they have reached their worry target.
 - A client should be able to add solutions to their worries.
 - A client should be able to view solutions they have logged.
 - A client should be able to control if a therapist can see a mood.
 - A client should be able to control if a therapist can see a worry.
 - A client should be able to receive reminder notifications.
 - A client should be able to disable notifications.
 - A client should be able to customise the appearance of charts.
- Therapist application:
 - A therapist should be able to see solutions a patient has logged.
 - A therapist should be able to view a patient's targets

4.1.3 Optional requirements

- Client/Patient Application:
 - A client should be able to delete a mood.
 - A client should be able to delete a worry.
 - A client should be able to log a GAD-7 score.
 - A client should be able to use the app without syncing data to the server.
- Therapist Application:
 - A therapist should be able to leave a note for a client.
- Server:
 - The server should delete link requests after a set amount of time.

4.2 Non-Functional Requirements

Non-functional requirements are those which are not concerned with the specific functionality of a system, instead defining constraints the system as a whole must meet (Kotonya & Sommerville, 1998). The requirements set out for this system are:

- Usability - The applications should be easy and intuitive for a client, no matter their skill level
- Security - The system should be able to authorize and authenticate clients with roles-based authentication.
- Reliability - The system should have no bugs that could cause any part of it to fail and disrupt the client
- Scalability - The server should be able to scale up and down to match the demand

- Effectiveness - The system should be effective in when helping the client understand and control their anxiety
- Accuracy - The system should not allow the client to input any incorrect data
- Performance - The system should respond quickly to the client's actions
- Testability - All parts of the system should be testable
- Documentation – The system should be well documented

4.2 Use Cases

In order to better understand how the two users would interact with the system, a use case diagram (Figure 1) was created showing some high-level actions that each user type could make. Although it doesn't show how each user goes navigates the system, it shows where the two user's user cases meet, which helped when building these sections of the application.

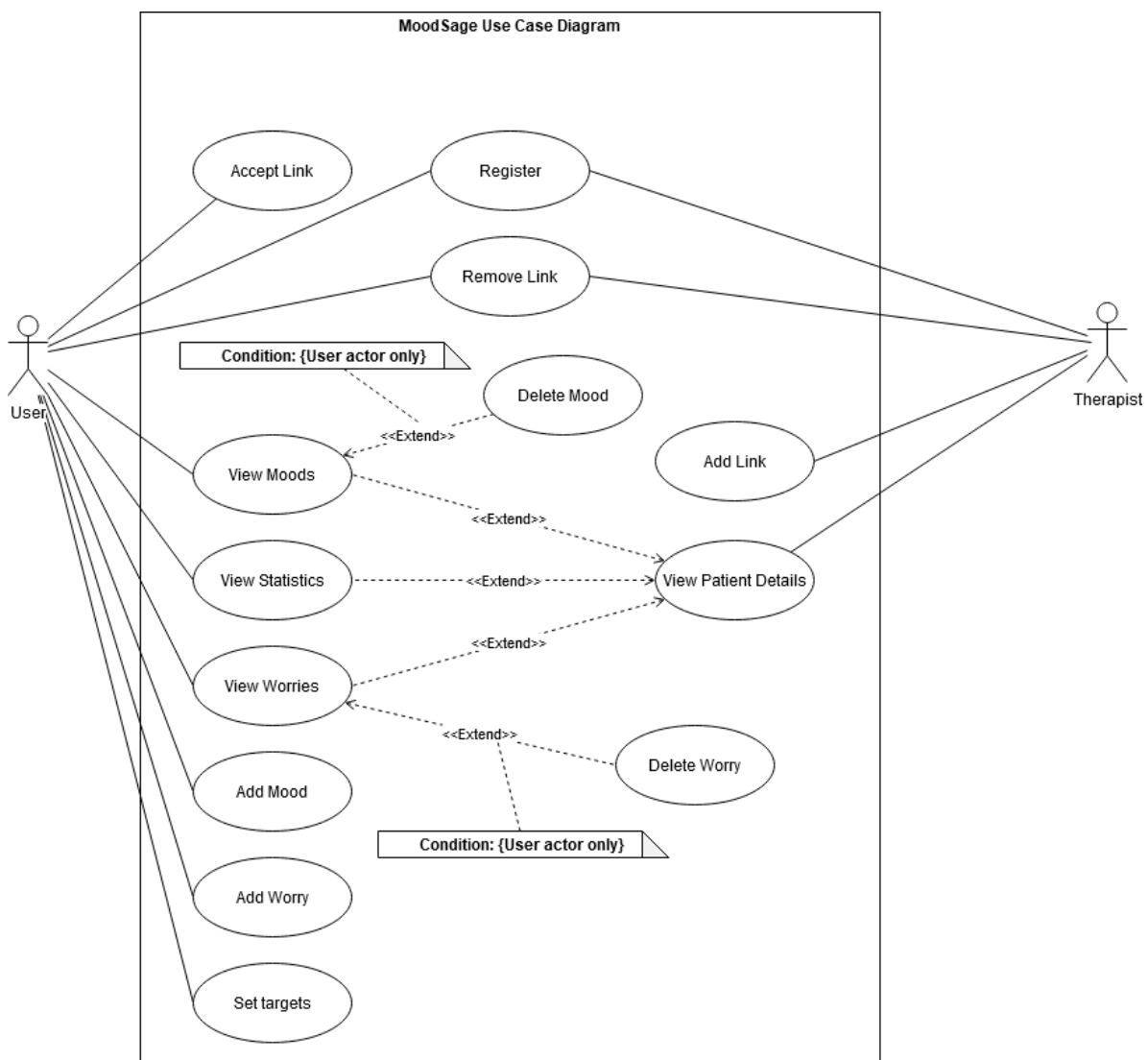


Figure 1: System use case diagram

5. Chosen Development Technologies

5.1. Mobile Application

Android was chosen as the platform for the mobile application, using Android Studio 3.6 as a development environment and Kotlin as the programming language. These technologies were chosen because:

- Cost – All of the development tools associated with Android are free to use and have no platform specific requirements.
- Access to Devices – The developer had no access to iOS devices, which would have meant relying solely on an emulator. This would not have given an accurate representation of the system.
- Current best practice - As of Google I/O 2019, Android development is Kotlin first and more than 50% of current professional developers use it in the development of applications (Haase, 2019).

5.2. Server and Database

The server-side component of the application was implemented using Node.js, hosted on Heroku, and the Database was implemented with the NoSQL database MongoDB, hosted using MongoDB Atlas. These technologies were chosen because:

- Cost – Atlas and Heroku both had generous free tiers, with Heroku also offering free access to paid tier services for students.
- Flexibility – Having a separated database and middleware API allowed for much greater control over how to use and access the data, as well as offering options for changing out various parts should the needs of the application change.

5.3. Web Application

The web application was developed using Angular, and the final version was hosted with and served to the client by the Node.js server, making it a full MongoDB, Express, Angular, and Node (MEAN) stack application. This technology was chosen because:

- Flexibility - The ability to break the site down in to small components means which new features can be added easily, and the site can even be easily reorganised should requirements ever change.
- Resources - Angular offers a lot of resources for building good and responsive User Interface (UI) using the material components library.
- Compatibility - Building the therapist application to run in a web browser guarantees a client's therapist would be able to access it no matter what platform they were on.

6. Design, Architecture and Implementation

Figure 2 shows the architecture of the overall system.

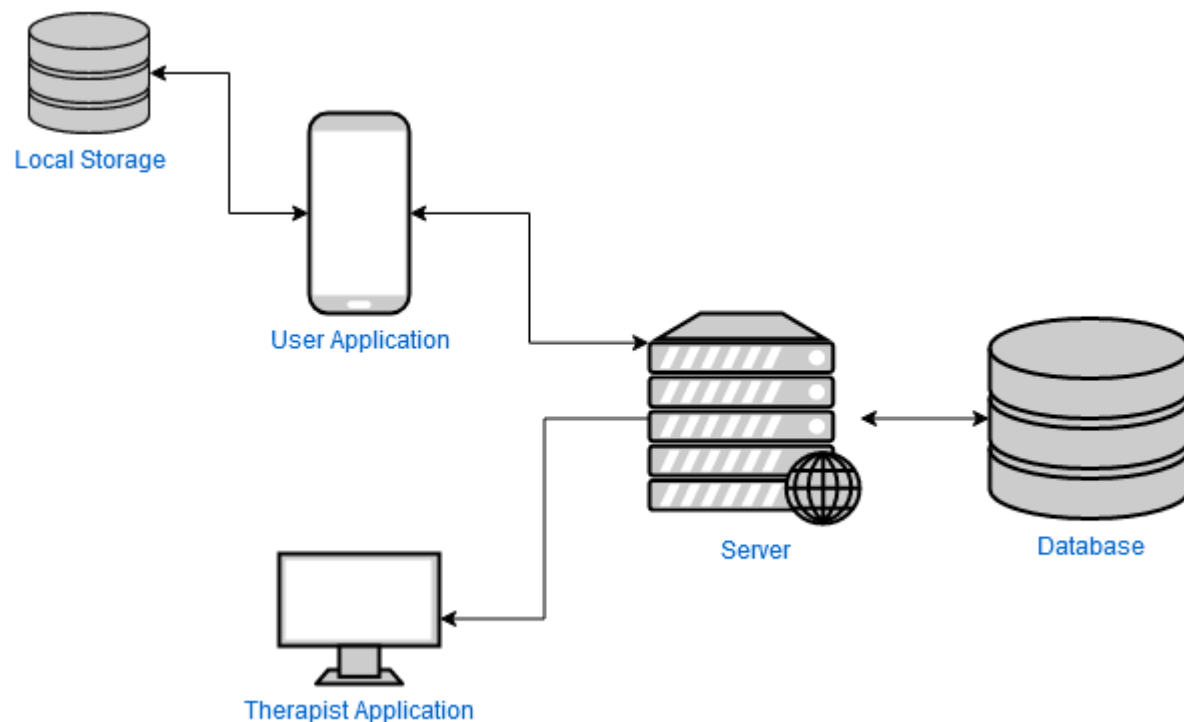


Figure 2: System architecture diagram

6.1. User Interface Design

As this was an application focused on helping clients feel better and relies on the client wanting to use it regularly, it was important the design of the client interface followed good Human-Computer Interaction (HCI) principles so that the app was intuitive and didn't cause any frustration with its users. Time was also spent researching the colour scheme of the application, to pick a colour which clients would associate with feeling happier and calmer and colours which they would like to spend time viewing.

Purple was chosen as the base colour for the application (Example in Figure 3). This is because it elicits feelings of relaxation and calmness, as well as comfort and happiness (Kaya & Epps, 2004). A colour that can elicit these emotions is important to have within the application, as it increases the likelihood of the application positively effecting the client. Green and Blue were also considered for the application, however in another study it was shown that out of the options, purple was the colour the most people would describe as pleasant (Simmons, 2011), and as such, appeared to be the best choice for this application and its Logo (Appendix E).

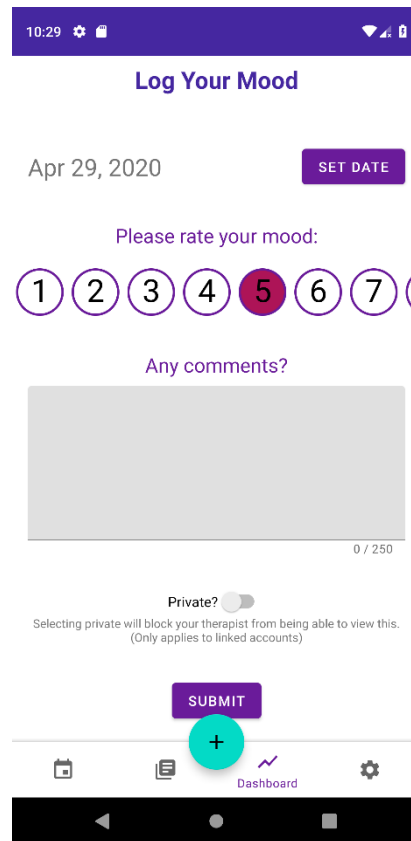


Figure 3: Example application screenshot

In order to ensure that the client interface was intuitive to use, the “8 Golden Rules of Interface Design” (Shneiderman, et al., 2010) were used as the guiding principles for UI design. These rules are:

1. Strive for consistency.
2. Enable frequent clients to take shortcuts.
3. Offer information feedback.
4. Design dialogue to yield closure.
5. Offer simple error handling.
6. Permit easy reversal of actions.
7. Support internal locus of control.
8. Reduce short-term memory load.

A full breakdown of these rules, as well as the original UI designs, can be seen in Appendices F and G.

6.2. Design Principles and Patterns

Following software design principles is important when writing code, it allows the code to stay readable, expandable and maintainable which is something that needs to be accounted for when developing a large project.

6.2.1. SOLID

SOLID is one set of design principles that were followed during the development of this application, this set consists of:

- **Single Responsibility.**
- **Open/Closed.**
- **Liskov Substitution.**
- **Interface Segregation.**
- **Dependency Inversion.**

Following these principles allows for the system to be maintainable and much more scalable in the future by having greatly reduced coupling between objects and much clearer code, especially when compared to a system that does not follow them (Singh & Hassan, 2015).

For some examples of these principles in use, the single responsibility principle can be seen in most classes in all applications, though most obviously within the repository layer of the Android application, where each repository class deals with one specific data type, and the same in the Angular services. As the API makes use of subclasses for the client and therapist types, the Liskov substitution principle can be seen where certain calls are able to be made using the parent 'Account' class and wouldn't be broken if the sub types are used.

Interface segregation is in use within the Android application as well, where items that require a small set of functionality are given an interface to that functionality instead of a whole class reference, this is so that other methods within the implementing class are not exposed where they are not needed.

6.2.2. Dependency Injection

Across the web and mobile application, the dependency injection pattern is used. Dependency injection is a design pattern where an object receives its dependencies rather than instantiating them itself (Razina & Janzen, 2007). This is often done using a dependency injection framework such as Dagger, which is used in the android application, although manual dependency injection is possible as well and is useful for unit testing. By using dependency injection, the developer can reduce coupling within the code, and make it easier to change out modules when new features need to be added or if mocks are needed during testing; this is especially pertinent when combined with strong adherence to the dependency inversion principle. This has the advantage of making code much more readable and maintainable, as instantiation logic is kept away from the rest of the code.

6.2. Server

The server is a Representational State Transfer (REST) API, and is an architecture style where resources on a server are exposed through use of URIs and actions are defined using HTTP verbs, such as POST, GET, PUT and DELETE representing the typical create, read, update and delete actions which most applications need to perform. REST APIs should only store the state of the resources which they are serving, leaving application state as the responsibility of the user (Richardson & Ruby, 2007).

In order to communicate the state of whether a client is logged in or not, the server uses JSON Web Tokens (JWT) for authentication. When a client provides valid

credentials to the “/auth/login” route, the server will provide a token which they can use to authenticate themselves on future requests by providing it within the HTTP Authorization using the Bearer authorization scheme. In order to prevent the tokens from being tampered with by a potential attacker, they are signed by the server’s secret key, this helps to offer a layer assurance that the client is who they say they are. JWTs were used because unlike session-based authentication, which keeps a record on the server of who has been logged in, they store no information on the server, allowing it to remain stateless.

For a full description of the available API routes and what they do, see Appendix H.

6.3. Database

MongoDB was chosen as the server-side database for the application. MongoDB is a NoSQL, document based and distributed database that uses a schema-less JSON-like design (MongoDB Inc, 2020). A NoSQL database was chosen as it offered a few advantages for this style of project over a traditional relational database system.

One of these advantages is due to the agile project management structure where features are added incrementally in each sprint. Using a database that does not have a strict schema means that new features can easily be added during sprints, without modifying the existing data that already exists (Bhogal & Choksi, 2015). This allows for a much smoother evolution of the prototype during development.

To allow for some guarantees of data consistency so related documents would still share similar fields once new features were added, Mongoose was used to provide validation and extra hooks for business logic on the database. This allowed for objects to be defined in a much more structured way, and made the code much more robust in terms of readability and maintainability, it also allowed for the definition of sub-types of documents, so that similar documents could be kept in the same collection without the risk of trying to access undefined fields. As MongoDB uses a JSON-like structure, a class diagram has been used to represent the final database design, as defined by mongoose, instead of a traditional Entity Relationship Diagram, this can be seen in figure 4.

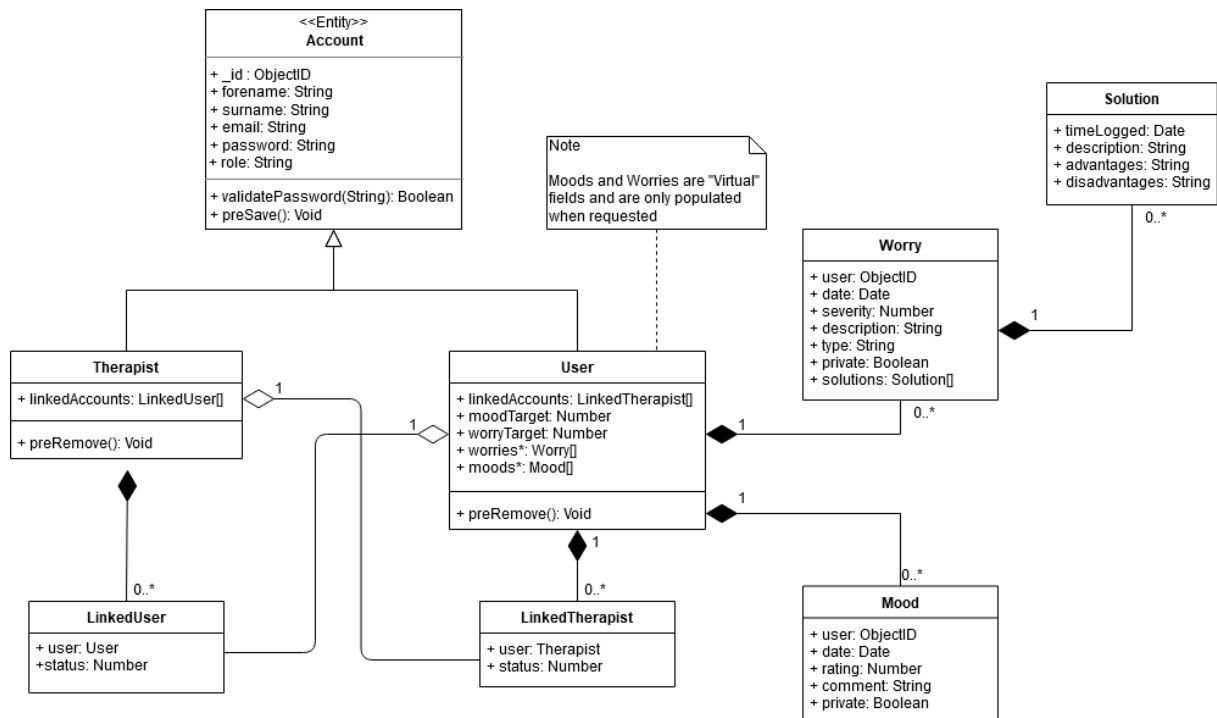


Figure 4: Class diagram of the final database design

6.4. Mobile Application

The overall architecture of the mobile application can be seen in figure 5, this design was used as it is the current best practice for developing native Android applications and is the pattern currently suggested by Google (Google Developers, 2019).

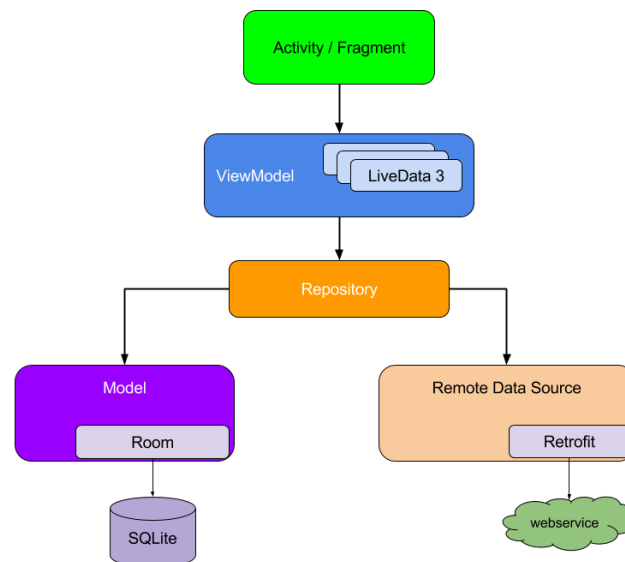


Figure 5: Overall architecture used by the android application. (Google Developers, 2019)

The architecture uses the MVVM pattern. MVVM splits the application into 3 major components (Moravcik, et al., 2012), each with a single responsibility, these are:

- Model - The model is the data layer of the application, handling the information that the application needs to use.
- ViewModel - The ViewModel interacts with the model, handling all the data that can be used within the UI and all the interaction logic. It exposes data to the views using observable data types, allowing for easy and seamless updating of the views.
- View - The view handles the presentation of data; it observes the data which has been exposed by the ViewModel and reacts to changes and events as the ViewModel emits them.

As the ViewModel is unaware of what views are observing it, this pattern allows for much looser coupling between these two areas of the application, allowing them to easily be changed out and tested as it just needs to be concerned with emitting events and data, rather than interacting with its view, like in other patterns such as Model-View-Presenter. Within this application databinding was used, this allowed most of the data presentation logic to be defined within the view XML, helping to keep the view class, which in android would be the fragment or activity class, clean.

The repository layer acted as a wrapper for the data layer of the application, allowing for a single point of access to both the network and local data sources allowing for them to easily be modified and changed out should the needs of the application change. For the local data source, Room was used, Room is an Object Relational Mapper (ORM) for a local SQLite database and allowed for easy local storage within the application. Although the remote database is NoSQL based, Room's nature as an ORM allowed for easy transitions between the types of data structure, and its inbuilt support for LiveData and Kotlin Coroutines offered far more development benefits a solution like Realm or ObjectBox. The structure of the local database can be seen in figure 6.

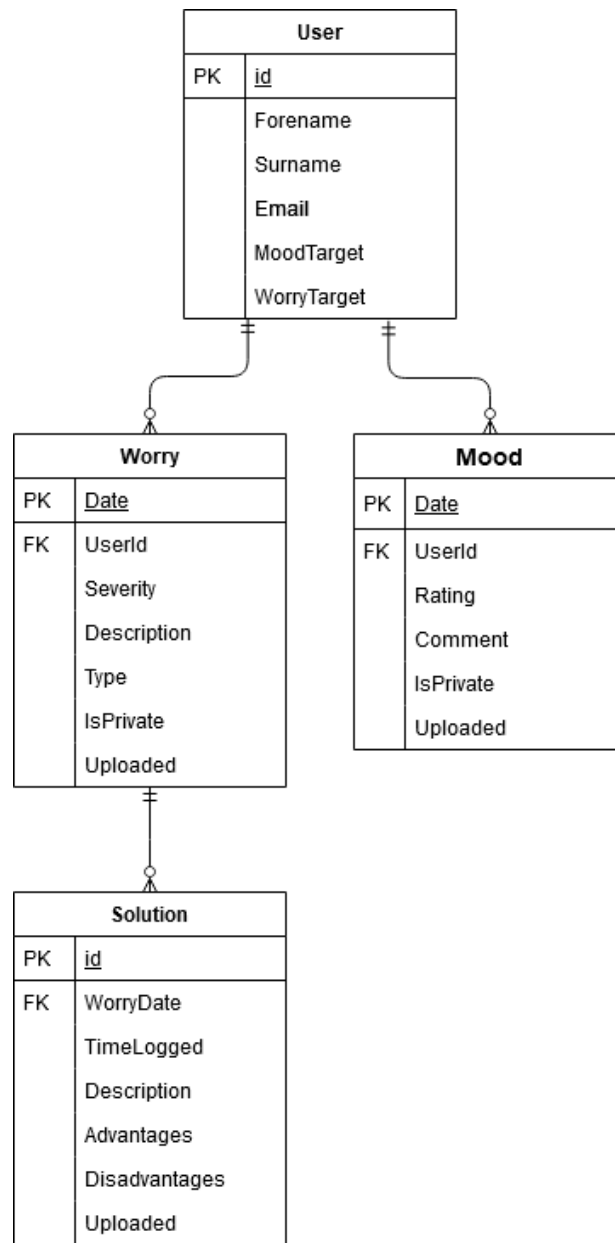


Figure 6: Entity Relationship Diagram of the local SQLite database

6.5. Web Application

For the Angular application, it was structured so each piece of functionality was broken off into its own component, which is a separate section of html with its own controller, which were grouped together into modules relating to their purpose. This structure was used as purpose-based modules allowed for easier understanding of where things should be within the application, helping with maintainability and readability of the code and then components representing a single piece of functionality allowed for them to follow the single responsibility principle much more closely, which in turn made them easier to understand and test.

Common functionality needed across the app, such as network calls and accessing data that needed to be shared across components, was put into services which could then be injected into the components when they were needed. Observable

data types were then used within the services for handling the main data and state of the application, this meant that if one component needed to update. For example, the clients list, this change would be immediately pushed to any other components which were observing the data.

7. Method of Approach

7.1. Agile

An agile approach was taken during the development phase of this project. Unlike the waterfall model, which is split into 5 phases consisting of; requirements, design, implementation, verification, and maintenance and won't let you return to a previous phase, agile takes a more incremental approach (McCormick, 2012). Agile breaks the program down into smaller chunks, focusing on delivering small parts of the software in small increments, rather than as one piece at the end of development.

This has the advantage of making it able to react changes in the requirements of the software without having to return to the drawing board. While it does have some disadvantages when it comes to much larger software project, where it could make it easier to go off track, for a small project such as the one being undertaken here, it fits perfectly and means that research can still be undertaken during the development process and as potential new features are worked out from this, they can be added to the backlog. As client testing would happen during the development of this application, Agile would also allow for this to have a much more significant effect on the overall application because user feedback and research could be immediately incorporated into the next increment of development, allowing for a much more fluid development process.

7.2. Scrum

Scrum is an agile methodology which focuses on delivering software at fixed intervals known as "sprints". A sprint is a set of development activities that takes place over a defined time (Schwaber, 1997), they contain the development of each proposed set of features as well as a review of the progress which was achieved and any adjustments which may transpire. Traditionally scrum teams have designated roles, such as scrum master and product owner however, as this was project was undertaken with a single developer, these aspects of scrum were not used.

In the case of this project, one-week sprints were used. This ensured that the developer could react quickly to any changes needed in the system and made sure that new features were consistently added to ensure a good pace of progress. Sprint reviews were carried out at the end of each sprint, reflecting upon the progress which was achieved and planning out the work that would be undertaken in the next sprint. The sprint reviews for this project can be seen in Appendix B.

8. Project Management

8.1. Trello

In order to keep the project organised and to keep track of what needed to be done, Trello was used. Trello is an online tool to help organise a project into a board which can tell you what is being worked on, what exists in the backlog and what has been completed as well as any other information which may need to be tracked (Trello, 2017).

The board for this project was organised in to 6 columns:

- User Stories (Appendix I), which acted as the product backlog.
- In Progress, which was used to show which clients stories were currently being worked on.
- Finished, showing which client stories had been implemented in to this application.
- Other tasks, which was used to track tasks which need to be undertaken during the project but weren't related to functionality.
- Completed tasks, which was used to store non-functional tasks which had been completed.
- Sprint Log, this task stored a write up of each sprint, detailing what was being worked on, when it was due as well as defining which label colour represented the items from this sprint.

Screenshots of the board can be seen in Appendix B, and a small example in figure 7.

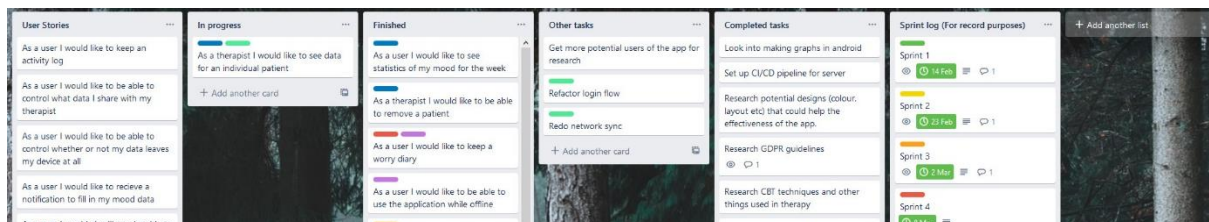


Figure 7: Trello board example

8.2. Version Control

As this was a major project with a large scope and multiple components, a version control system was needed in order to effectively manage changes to the code. Version control systems are a type of software that helps developers manage changes in the source code over time, allowing developers to return to any previous point in the project and easily revert breaking changes to the code (Atlassian, 2020).

Git was chosen as the version control system for this project, with repository hosting on GitHub. Using a remote repository as the master copy ensured there was always a recent copy of the source code available, making switching between development devices easier and protecting against any local data loss as well. As the system was being developed, a new branch was created for each new feature or large change that was being worked on and a pull request would then be opened to merge the

change back into the master branch when it was completed. Using this development structure meant that there was always a working version of the code available on the master branch and provided easy and key points to revert to if something went wrong during development.

9. Development

This section will go more into detail on the processes and changes involved in reaching the point the application is at now, as well as some smaller technologies which were used.

9.1. Stage 1 - Research and Requirements

The first stage of the project focused heavily on researching the market for this application, creating the initial set of core requirements, as well as researching potential hosting solutions for the server. During this period, time was also spent learning how to use various technologies the developer was unfamiliar with, such as Dagger and the various unit testing frameworks which would be in use across the different application types.

Requirements research occurred through a combination of market research and seeing what similar applications on the market were doing, as well as looking into research for common cognitive behavioural therapy techniques. This research influenced the edition of the worry diary functionality into the application, as well as the desirable functionality which extended from it, as it was discovered this was a very common technique used for anxiety patients and there was much potential functionality to gain from the current methods (Borkovec, 2006).

No significant challenges were faced over the course of this phase and by the end of it, requirements for the application had been decided on.

9.2. Stage 2 – Authentication and Account functionality

The second stage of development was for the first sprint and involved getting the server functional, as well as getting log in and authorisation functionality completed so the application could be designed around it from the start, rather than it being retroactively added to the application. This ensured security was enforced throughout the entire application.

During this stage, a few challenges were faced due to the developer not being very familiar with the technologies in use however, it did not cause any delays within the project and this stage was still completed within the planned timeframe.

9.3. Stage 3 – Core functionality

Stage 3 of development was the largest, and encompassed sprints two to eight, focusing on development of the core requirements of the application and reaching the MVP.

There were several challenges faced during this stage of the project, and many of the earlier sprints fell behind schedule due to a lack of familiarity with Kotlin, as well as technologies which were in use such as databinding as some binding adapters proved harder to implement than expected. However, stage four had been planned with enough of a buffer so as not to cause any issues with the development of the application and overall project progress.

One of the issues which did occur during this phase was within the repository layer of the Android application, due to it being the mediator between the remote and local data sources for the application, error handling, and communicating these errors to the client became an issue as there were many potential causes. Consequently, a large refactor ended up happening and the repository layer was changed so each method would return a `Resource<T>` instance, containing a message, status and the data they were fetching (if successful). This also led to the development of the `ApiCall<T>` class for use in the network sources and was created as the repository needed to know the status of the network call, and while Retrofit did provide its own `Response<T>` wrapper class, it didn't make sense for the abstraction to depend upon a library specific data type. It would make it harder to move to a different library as the interface would need to be re-written as well, going against the purpose of the dependency inversion principle.

During this phase of development, more research was still being undertaken for extra application features and some new ones were added to the application. For example, this application depends on clients regularly logging their data and as such, a method of helping it to become habitual was needed. One study into this area has shown that sending daily reminder notifications has the highest success rate when getting clients to log information daily however, it was not as good at making that process eventually become a habit as having a trigger event for the process (Stawarz, et al., 2015). As there is no trigger event for this application the notification approach was used as this should still allow habit formation, albeit over a longer period and should allow for more consistent results while building the habit.

9.4. Stage 4 – Desirable features

Once the core requirements of the application had been implemented, stage 4 of development began. This stage involved beginning to implement the desirable features of the application, as well as when the testing process began as the MVP was now complete.

There were a few challenges during this stage of development, for example, owing to the slower than anticipated progress from stage 3 some extra work had to be taken on each week in order to catch up and get as many of the desirable features into the application as possible. However, this was dealt with well and the project began to operate much more smoothly during this phase because of the Learning from Experience (LfE) to this point.

There was also an issue with the web application at this point in the development cycle. Due to unfamiliarity with this aspect of the technology, the way route authorization was handled within the application had to be fixed as redirects were not always activating, this led to the inclusion of Angular AuthGuards which were used to stop unauthenticated users from getting in to certain parts of the application, though the lack of an API key would have still stopped them from retrieving any data.

There was an issue with ensuring components were all acting on the same source of data so any updates would automatically be reflected across the page, this ended up being the structure, so instead of each page passing data down to the components.

Furthermore, they would all observe the main data source in the patient service and only charts and small detail components would take inputs, as they only needed a small portion of the data and the data they had wouldn't be updated by another component on the page.

Overall however, this stage brought the project back on track and all desirable functionality ended up being implemented into the application.

9.5. Stage 5 – Final touches and Report

During the final stage of the project, development of new features for the application ceased and the focus shifted on to fixing bugs and making quality of life improvements to the applications in order to make sure that they were ready for the project hand in. During this time the developer's focus shifted to writing the report as the major task, as well as assembling all the resources required for that. There were no major issues or challenges encountered during this phase of the project.

10. Testing

10.1. Unit Testing

Unit testing is the process of testing the smallest separate module in the system (Runeson, 2006) and is an important part of software development as it can help to ensure each module of the system is working correctly and lowers the risk of bugs as each part of the system is confirmed to work independently.

Unit tests were written for all 3 sections of this system, ensuring they work correctly and giving assurance any future changes to these parts would not break the pre-existing functionality which is available within the application. An example unit test from the Angular application can be seen in figure 8.

```
it("should disable submit if email is invalid", () => {
  component.editForm.get("email").setValue("invalidemailcom");
  fixture.detectChanges();

  let buttonDisabled = fixture.debugElement.query(By.css("#submit")).nativeElement.disabled;
  expect(buttonDisabled).toBeTruthy();
});
```

Figure 8: Example unit test

10.2. Test Driven Development and CICD

After the initial authentication and registration systems were developed for the server, a Test-Driven Development (TDD) approach was adopted. In TDD development follows a process of writing tests, writing code to make the tests pass, and finally writing refactoring the code to make it cleaner, this process can be seen visually in figure 9.

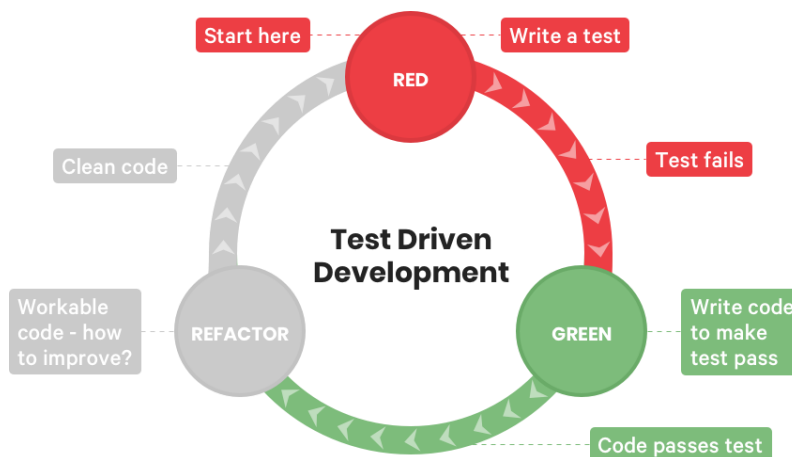


Figure 9: Visual example of Test-Driven Development (Chekalin, 2018)

The advantages of this approach are, it forces the developer to plan out how a piece of functionality should work prior to implementation. It also ensures any new code added to the system will work when it is finished and that it hasn't broken anything which already existed; this can be especially advantageous if you are doing a large scale refactor of the system.

Continuous Integration and Continuous Deployment (CICD), is a process which involves continuously merging all new code into a central repository and running a set of automated tests when these changes are merged into the master branch of the repository (Fowler, 2006). As with TDD, this ensures all new code added to the system will not break pre-existing functionality and everything new works. Subsequently, using a continuous deployment system all these changes can be deployed into the production server or application if the tests and any other defined constraints pass.

For this project, Travis CI was used to provide continuous integration and continuous deployment to the Heroku server. Travis was set up to watch the central repository, running the testing suite on any new commits which were pushed to it. When code was then merged with the master branch it would run the testing suite and if successful, trigger a new deployment on the server, making sure the API was consistently up to date with the newest working features.

10.3. Usability Testing

Once the MVP had been completed, usability testing was performed with real clients. In these tests, clients were given a set of tasks which they needed to complete that covered all areas of functionality within the application. They were then requested to fill out a questionnaire when they completed their testing for feedback to be gathered. Consequently, feedback was used to improve existing parts of the application and to guide development of new features within the application as well.

This process happened over the course of a few weeks, with new features and fixes being implemented as the results came back in. Most client testing used the task list and questionnaire format, though some client tests were more general purpose and related to long term usage of the application. The task lists, results and the changes made can be seen in Appendix J.

10.4. Functional Testing

Where unit testing is a “white box” testing technique, as tests are designed based upon the source code, functional testing is a “black box” technique, as tests are designed based on the specification and functionality of the application. In order to make sure a system functions as expected, it is important both cases are covered (Lui & Tan, 2009). During development of the application, functional testing was used to ensure features were working as they were made however, in order to ensure the application was completely functional at the end of development, a functional testing plan was created, converting each of the implemented functional requirements into a set of test cases involving failing conditions as well as the expected behaviour. The results of this testing can be seen in Appendix K.

11. End-Project Report

11.1 Summary

Overall the project was completed successfully. This project was started with the goal of producing an application to help people who are suffering from anxiety, giving them tools and resources to help manage their worries and communicate them with a therapist. Though there were some challenges along the way, causing the project to fall behind schedule, the project did meet all the core requirements which were set out, as well as all the desirable requirements.

11.2 Objectives Review

1. To research cognitive behavioural therapy techniques which could aid in the effectiveness of the application.

The objective has been met through the research which took place over the course of development. This research returned journaling and reflection as one of the common methods used within CBT. This was chosen as the direction for the application as being able to transfer it into an application format would allow for extraction of more data, giving the user more to reflect on in order to understand how they feel. This was then followed up by further research into specific competitors to this application, as well as the research from Caldeira et al (2018) in order to make sure this application fits the user profile.

2. To build an application which allows a client to be able to track and reflect upon their moods and worries.

This objective has been met through the completion of the MoodSage Android application. The application gives the client the ability to do what has been stated within the objective and goes further by giving them the ability to take positive action in being able to log solutions.

3. To build an application which allows therapists to be able to view the data of any clients they may have

The completion of the MoodSage therapist application fulfils this objective. The application gives therapists a simple view of all their clients and easy access to view their data and to conduct deeper analysis than would have been possible had the client just been showing them their view of the mobile application.

4. To design a user interface in a way which actively helps the client feel better

This objective was met through research in to colour theory and emotional association with specific colours, in order to choose a colour scheme that would help the user. Research was also undertaken in to HCI principles and good user interface design in order to make sure that the interface was easy to use and intuitive, so that it did not leave the client frustrated. Although this object was achieved, further research could also be undertaken into the use and effects of different shapes on user emotion and perception.

5. To create a way for clients to share information with a therapist in a way that they feel comfortable with

Through the development of the account links system, and the addition of privacy settings to the final user application, this objective has been met. The link system ensures that a client is aware of who they are linked with by displaying the necessary details and requiring the client to be the one who accepts the link. The privacy settings then add to this by allowing the clients to hide certain things if they don't feel completely comfortable yet.

6. To follow agile project management techniques and build the application within the time allocated

Overall, this objective was reached. Although some sprints within the application fell behind schedule, the application was completed within the time allocation for the project and it was completed in an agile and iterative manner. Overall success notwithstanding, future development with this project, or other agile style projects could mitigate the issues encountered by expanding how the product backlog is structured, breaking down user stories on the board into their core requirements and labelling them appropriately to be able to see where the development time needs to be spent with each one.

11.1. Changes

Initially, the application was going to be focused on helping people with depression as well as anxiety, however, during research it was decided that it would be better for the application to make its primary focus anxiety as it would allow for more in depth features and tailor what it displays to fit that purpose, rather than going more general. This was also decided due to the results of the market research, which showed there were far fewer solutions which gave a focus to anxiety when compared to those that focused on depression or, took a more generalised approach.

12. Project Post-Mortem

12.1 Objectives Evaluation

As previously mentioned, the aim of this project is to aid in management and tracking of moods and worries and to make it easier for that information to be shared with a therapist and the objectives were structured around this. All the objectives were met whilst also seeing an increase in the effectiveness of the final product.

12.2 Technologies Evaluation

Overall, the technologies which were chosen in order to develop the applications are considered the correct choice as they allowed the application to be completed satisfactorily and did not introduce any significant issues of their own.

Developing the application for Android allowed for the developer to be sure the application was working and that all features performed well. It also allowed the developer to explore an area of interest and gain experience with Kotlin, a new language.

The choice to use Angular for the website allowed for it to be built in a modular, scalable and maintainable way, with small individual components being combined to create each page. In hindsight, this helped keep the code base clear and easy to understand, while also providing a seamless experience for the therapist who would be using it.

Finally, Node.js and MongoDB were appropriate choices for the backend of the system. MongoDB's flexibility was correct for the agile management of the project, and although it increased the amount of work to do, building an API with Node.js gave much more control and flexibility to the system than what could have been achieved with an all-in-one solution such as Firebase.

12.3 Developer Performance Evaluation

During development of the application the developer attempted to spend between 30 and 40 hours per week on all aspects of the project. In most instances this was achievable however, there were some weeks where other commitments or unforeseen circumstances would get in the way. To compensate, extra time would usually be spent in the following weeks in order to recover the timeline which resulted in the recommended amount of project time being used.

The developer did face difficulties due to their unfamiliarity with dealing with a project of this scale, which did lead to rushing into some tasks and underestimating the challenges that would be faced, this ultimately led to delays in other tasks which should have potentially been done before. However, a lot has been learned from this project and the knowledge can be used in future endeavours.

12.4 Project Management Evaluation

Overall, Agile was the correct approach to take with this project as it allowed for iterative development of the deliverables and meant any changes in requirements as the project went on, were easy to deal with and resolve. Scrum however, may not

have been the best approach for this as a solo project, instead a project Agile methodology such as Kanban may have been more suitable as it provides more focus on the work currently in progress and greater flexibility with planning, which could have stopped the project from being affected by external factors as it had been.

12.5 The Future

In its current state, the application is in a formidable position to be able to help people understand how they are feeling and know how to recover and yet, there are still many other features which could be added to the application in order to increase its viability within the market and make it even more maintainable, these are:

- The addition of features such as streaks, in order to entice users in to coming back regularly.
- Multi-platform mobile support, either by developing a native iOS app, or using technologies such as Flutter. This would open the application up to even more potential users.
- The ability to export user data from the application as a .CSV file, so that users could import it into another program and explore the data in their own way.
- Create even more detailed weekly or monthly reports of a user, generated by the server at a fixed interval that could provide more in-depth analysis of the previous month than possible on a phone.

13. Conclusion

The goal of the project was to create an application which could help people keep track of how they are feeling and reflect upon it. The project has achieved all the objectives which were set out, as well as fully meeting the core and even desirable functional requirements. Although there were some setbacks during development, the project was able to recover the schedule and was delivered on time. This was the first time the developer had worked on a project of this scale, and considering what has been achieved, it is a success. In the future it is hoped the application could be developed further and eventually released on the market with more features and even stronger integration between the two main parts in order to become one of the leading solutions for mental health self-tracking on the market.

References

- Atlassian, 2020. *What is version control? - Atlassian Git Tutorial*. [Online]
Available at: <https://www.atlassian.com/git/tutorials/what-is-version-control>
[Accessed 4 May 2020].
- Baker, C., 2020. *Mental health statistics: prevalence, services and funding in England*, London: House of Commons Library.
- Bhogal, J. & Choksi, I., 2015. *Handling Big Data using NoSQL*. Gwangiu, IEEE.
- Borkovec, T. D., 2006. Applied relaxation and cognitive therapy for pathological worry and generalized anxiety disorder.. In: *Worry and its psychological disorders: Theory, assessment and treatment*. s.l.:s.n., pp. 273-287.
- Caldeira, C. et al., 2018. *Mobile apps for mood tracking: an analysis of features and user reviews*. s.l., American Medical Informatics Association., p. 495–504.
- Chekalin, D., 2018. *Advantages of Test Driven Development - Codica*. [Online]
Available at: <https://www.codica.com/blog/test-driven-development-benefits/>
[Accessed 4 May 2020].
- Community & Mental Health Team, 2019. *Psychological Therapies Report on the use of IAPT services, August 2019 Final Summary Report*, s.l.: NHS Digital.
- Daylio, 2020. *Daylio, Journal, Diary and Moodtracker*. [Online]
Available at: <https://daylio.net/>
[Accessed 2 May 2020].
- Fowler, M., 2006. *Continuous Integration*. [Online]
Available at: <https://martinfowler.com/articles/continuousIntegration.html>
[Accessed 4 May 2020].
- Google Developers, 2019. *Guide to app architecture | Android Developers*. [Online]
Available at: <https://developer.android.com/jetpack/docs/guide>
[Accessed 4 May 2020].
- Google, 2020. *Daylio - Diary, Journal, Mood Tracker*. [Online]
Available at: https://play.google.com/store/apps/details?id=net.daylio&hl=en_US
[Accessed 2 May 2020].
- Haase, C., 2019. *Android Developers Blog: Google I/O 2019: Empowering developers to build the best experiences on Android + Play*. [Online]
Available at: <https://android-developers.googleblog.com/2019/05/google-io-2019-empowering-developers-to-build-experiences-on-Android-Play.html>
[Accessed 12 May 2020].
- Information Commissioner's Office, 2020. *The Principles | ICO*. [Online]
Available at: <https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/principles/>
[Accessed 2 May 2020].

- Kaya, N. & Epps, H. H., 2004. Relationship Between Color and Emotion: A Study of College Students. *College Student Journal*, 38(3), p. 396+.
- Kotonya, G. & Sommerville, I., 1998. *Requirements Engineering: Processes and Techniques*. 1st ed. s.l.:Wiley.
- Lui, H. & Tan, H. B. K., 2009. Covering code behavior on input validation in functional testing. *Information and Software Technology*, 51(2), pp. 546 - 553.
- McCormick, M., 2012. *Waterfall vs. Agile Methodology*, s.l.: MPCS, Inc.
- MongoDB Inc, 2020. *The most popular database for modern apps | MongoDB*. [Online]
Available at: <https://www.mongodb.com/>
[Accessed 3 May 2020].
- Moravcik, O., Petrik, D., Skripcak, T. & Schreiber, P., 2012. Elements of the Modern Application Software Development. *International Journal of Computer Theory and Engineering*, 4(6), pp. 891 - 896.
- NHS, 2018. *Silvercloud course - NHS*. [Online]
Available at: <https://www.nhs.uk/apps-library/silvercloud/>
[Accessed 2 May 2020].
- Razina, E. & Janzen, D., 2007. *Effects of Dependency Injection on Maintainability*. Cambridge, Acta Press, pp. 7 - 12.
- Richardson, L. & Ruby, S., 2007. *RESTful Web Services*. 1st ed. Sebastopol: O'Reilly Media, Inc.
- Runeson, P., 2006. A survey of unit testing practices. *IEEE Software*, 23(4), pp. 22-29.
- Sadler, K. et al., 2017. *Mental Health of Children and Young People in England*, s.l.: NHS Digital.
- Schwaber, K., 1997. *SCRUM Development Process*. London, Springer, pp. 117-134.
- Sharon, T., 2017. Self-Tracking for Health and the Quantified Self: Re-Articulating Autonomy, Solidarity, and Authenticity in an Age of Personalized Healthcare. *Philosophy & Technology*, Volume 30, pp. 93 - 121.
- Shneiderman, B., Plaisant, C., Cohen, M. & Jacobs, S., 2010. *Designing the User Interface: Strategies for Effective Human-Computer Interaction..* 5th ed. s.l.:Pearson.
- Simmons, D. R., 2011. Colour and emotion. In: C. P. Biggam, C. A. Hough, C. J. Kay & D. R. Simmons, eds. *New Directions in Colour Studies*. Amsterdam/Philadelphia: John Benjamins Publishing Company, pp. 395 - 413.
- Singh, H. & Hassan, S. I., 2015. Effect of SOLID Design Principles on Quality of Software: An Empirical Assessment. *International Journal of Scientific & Engineering Research*, 6(4), pp. 1321-1324.

Stawarz, K., Cox, A. L. & Blandford, A., 2015. *Beyond Self-Tracking and Reminders: Designing Smartphone Apps That Support Habit Formation*. New York, Association for Computing Machinery, pp. 2653-2662.

Trello, 2017. *What is Trello? - Trello Help*. [Online]
Available at: <https://help.trello.com/article/708-what-is-trello>
[Accessed 4 May 2020].

Appendix A. – User Guides


Mobile Application

Installation

To install the application, download and install the APK on your device or open the project in Android Studio and build from there, loading on to a device or emulator of your choosing. The application targets a minimum API level of 22, meaning it should work on any phone running Android 5.1 or later.

Log In/Register

9:30 9:30


MoodSage

First name
Last name
Email
Password
Confirm password

Email
Password

SIGN IN

REGISTER

No account? Register now

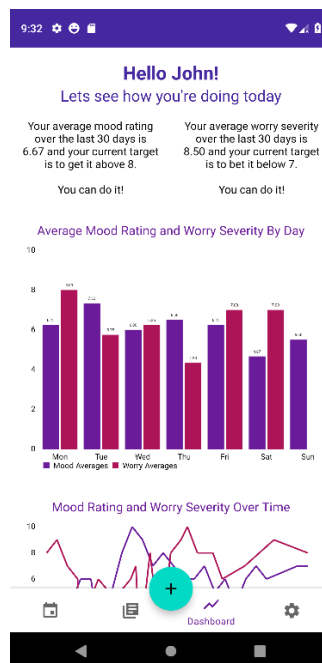
REGISTER

Password Must Contain:
8 characters or more
1 uppercase character
1 lowercase character
1 number

To log in to the application, simply enter your email and password in the labelled boxes and press sign in, the application will then proceed to a loading screen before taking you to the dashboard.

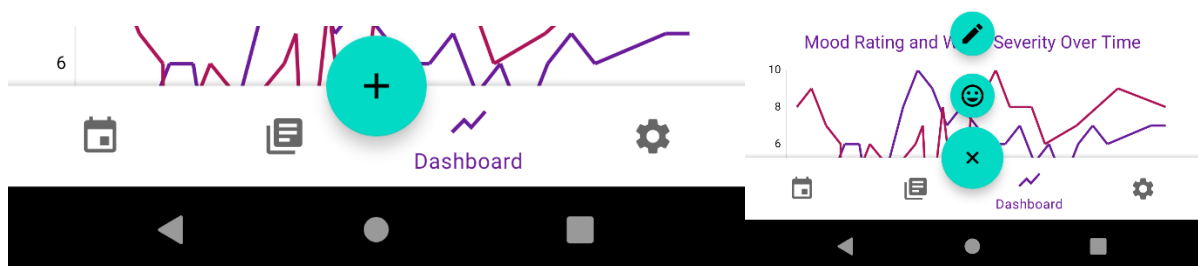
If you do not already have an account, press the register button at the bottom of the screen. This will take you to the account registration screen. From here, enter your details into the labelled boxes, make sure everything is correct and press register. If you are successful, you will receive a notification informing you and you will be taken back to the login screen, where you can follow the steps outlined above to log in. If you are unsuccessful, the application will inform you of what was incorrect, and you will remain on the registration screen.

The Dashboard



After logging in to the application you will be taken to the dashboard and until you log out of the application this is now where it will open to. The dashboard provides a general overview of your statistics, based off the statistics you have entered the application so far. The dashboard also provides an overview of your targets and how close you are to reaching them, if you do not have any targets set it will prompt you to go to your profile and add some.

Navigation



At the bottom of the screen is the navigation bar, this is where most of the navigation through the application will occur from. The calendar icon is for the settings screen, the books icon is for your worry diary, the graph icon, as visible in the images, is for the dashboard, and the gear icon is for settings. The middle button can be clicked on to expand it, this will reveal two more buttons, the button with the face will navigate to the "Log a Mood" screen, and the button with the pencil will navigate to the "Log a Worry" screen.

Log a mood

From the Log a Worry screen you can set the date of you worry, pick the rating using the horizontally scrollable rating picker and enter a short comment about your day. You are also able to set privacy settings for the date, if you have a therapist this means that they will not be able to see it.

When you have entered all your details press submit to save it, if you are not connected to the internet it will be uploaded to the server when you next get a connection. If there is an error while saving the application will inform you and leave the screen how it is. If it saves successfully, the screen will be cleared so that you can log another mood if you want.

If the date selected already as a mood, the submit button will be greyed out and disabled.

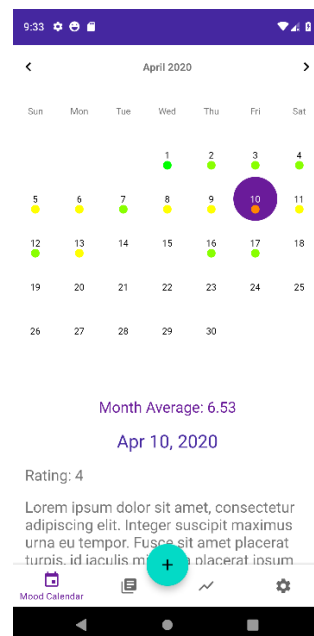
Log a worry

From the log a worry screen you can enter the details of the worry that you have been having. You can set the date and time, as well as the type of worry that it is. If you are unsure of what each type means, click the help icon and a description of each will pop up. Then you can set the severity of the worry, give a description of it and set your privacy settings for it.

When you have entered all your details press submit to save it, if you are not connected to the internet it will be uploaded to the server when you next get a connection. If there is an error while saving the application will inform you and leave the screen how it is. If it saves successfully, the screen will be cleared so that you can log another worry if you want.

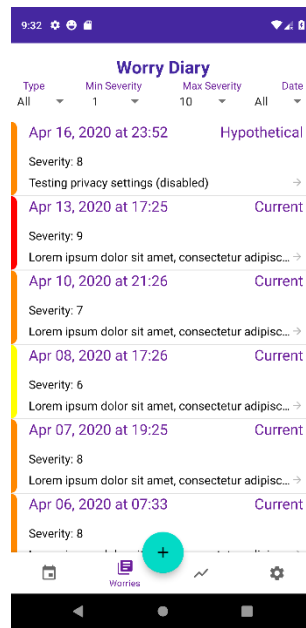
If the date and time selected already as a worry, the submit button will be greyed out and disabled.

The Mood Calendar



The mood calendar page will show you the moods you have logged for each day so far. It will also show the average mood for whatever month you are viewing, the month can be changed by clicking the arrows at the top of the page or by swiping left and right on the calendar. From this page you can click on each day and, if there is a mood it will be displayed below the calendar.

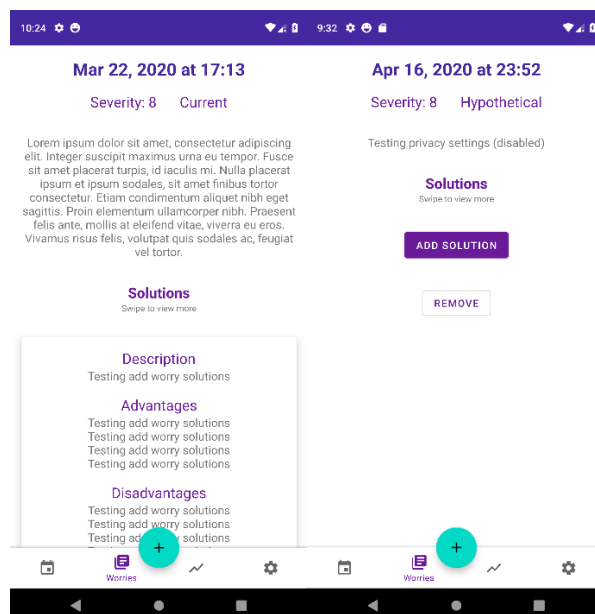
The Worry Diary



The worry diary will display all the worries that you have logged within the application so far. Using the filters at the top of the page you can set the conditions for which worries will be shown if for example you are trying to find a specific one, or you only want to see hypothetical worries from this week.

From this page you can also click on any of the worries you see, this will bring you to a new page with the full details of the worry, as well as any solutions you have logged.

Worry Details



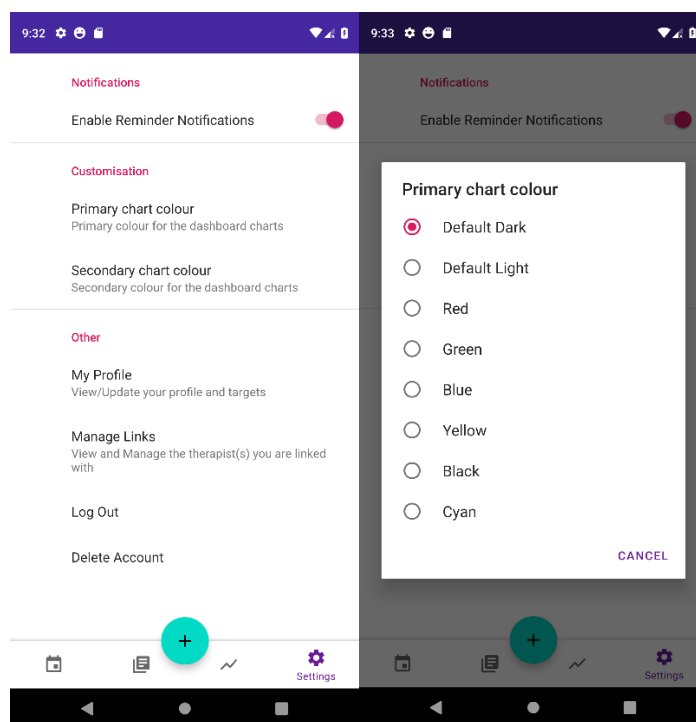
On the worry details page, you can see the full description of any worry that you have logged, as well as a horizontally scrollable list of all solutions that you have logged for it. If you would like to add a new solution to the worry, you can press the add solution button, or, if you would like to delete it, you can press the remove button.

Add solution

The screenshot displays the 'Add a Solution' interface. It features a purple header bar with the title 'Add a Solution'. Below this, there are four text input fields for user input: 'Description', 'Advantages', 'Advantages', and 'Disadvantages'. Each field includes a character count indicator (0 / 500). A purple 'SUBMIT' button is positioned at the bottom right of the form area. A green circular button with a white plus sign is located at the bottom center. The bottom navigation bar contains icons for 'Worries', a home icon, a settings icon, and another 'Worries' icon. The top status bar shows the time 9:32 and 10:33, along with various system icons.

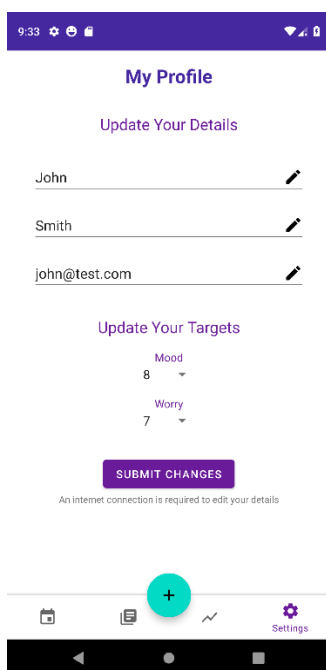
On the worry details screen, to log a new solution to a worry, enter the description, advantages and disadvantages of your solution into the labelled boxes and press Submit. If your attempt is unsuccessful, the application will inform you where the error is, and you can fix it and try again. If the attempt is successful, you will be notified and the screen will be cleared, allowing you to log another one. Press your devices back button to navigate back to your worry diary or navigate away using the bottom navigation bar.

Settings



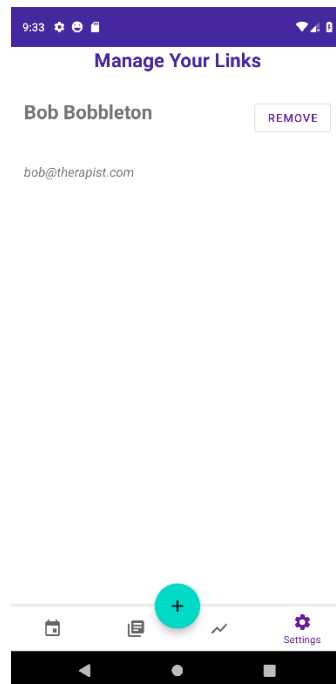
On the settings screen you can modify your preferences for the application, enabling/disabling notifications and setting your preferred colours for the charts on the dashboard. You can also navigate to your profile and to the account links page, as well as logging in to the application or deleting your account.

Profile



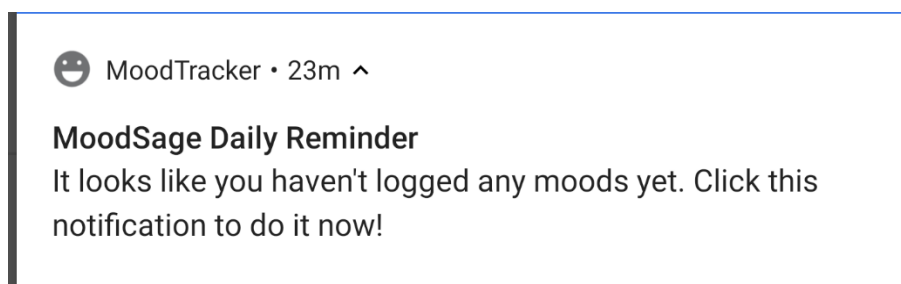
From the profile screen you can view and update your details if they change, as well as setting your targets for your moods and worries. As stated in the application, an internet connection is required to update your profile. When you submit the application will notify you on the success status.

Manage Links



If your account is linked with a therapist, the manage links screen will show the current account(s) you are linked with. Any pending requests will have accept and remove buttons, and all active links will have a remove button so you can stop access at any point.

Notifications



If you have notifications enabled, MoodSage will send you a reminder at 5pm every day reminding you to log your mood if you haven't already. You can click on the notification to go to the app, or swipe it away and do it later.

Web application

The web application can be accessed at: <https://api.stsoft.tech>

Home Screen



On the home screen of the application you will be greeted with the MoodSage logo, from here you can use the log in button if you wish to log in to the application, or you can use the register button if you do not have an account yet.

Log in/Register

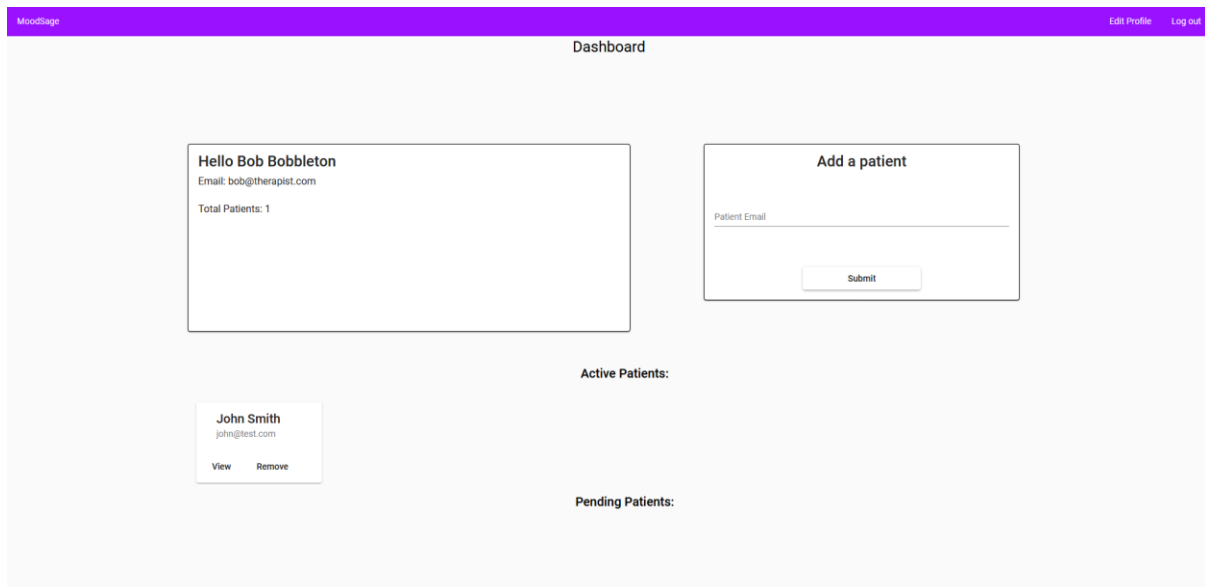
Register	Login
<p>First name *</p> <p>Surname *</p> <p>Email *</p> <p>Password *</p> <p><small>Password must contain:</small> 4 Characters 1 Uppercase 1 Lowercase 1 Number</p> <p>Confirm Password *</p> <p>Submit</p>	<p>Email *</p> <p>Password *</p> <p>Log In</p>

On the registration screen there will be a form to enter your details, if anything is invalid the fields will be highlighted in red and you will not be able to submit. Once you have finished entering your details, press submit. If account creation is successful you will be notified and taken to the log in screen, if not you will see a notification at the bottom of the screen.

On the log in screen, enter the email and password associated with your account into the labelled boxes. If anything is invalid the fields will be highlighted in red and you will not be able to submit. Once you have entered your details, press log in and

you will be redirected to your dashboard if you are successful, or notified if you are not.

Dashboard



The dashboard is your home screen when logged in to the application, from here you can see your details and the number of patients you have. You can also see a list of all of your active and pending patients, with the option to view the details of those who are active and the option to remove both types.

On this screen you are also able to add a new patient by typing their email into the labelled text field and pressing submit. If it was successful the patient will be added to pending and you will be notified, if it is not successful then you will receive a notification with the reason why.

Patient profile (Statistics view)



On the patient profile page you can view the details of the patient and the information that they have logged. At the top of the page is the patients details, as well as a selection of statistics about the patient.

Below this is a tabbed array for statistics and details. On the statistics page you will be able to see a selection of graphs showing various information the user has logged. All of the graphs have the option to just show data from the last month, some also have other options such as filtering by category, showing targets and changing the graph type.

On the ratings over time graph it is also possible to click on specific data points, doing so will bring up a card with details about the mood or worry that was clicked on.

Patient profile (Details view)

Statistics			Details		
Day	Average (All time)	Average (This Month)	Day	Average (All time)	Average (This Month)
Monday	8.50	9.00	Monday	6.25	6.00
Tuesday	5.75	6.00	Tuesday	7.55	6.00
Wednesday	6.25	6.00	Wednesday	6.00	6.00
Thursday	2.50	6.00	Thursday	6.50	7.00
Friday	7.25	6.00	Friday	6.25	7.00
Saturday	7.50	6.00	Saturday	4.67	6.00
Sunday	6.75	6.00	Sunday	5.50	6.00

Date	Type	Score	Date	Rating
Sunday 22 March 2020 at 17:15	Current	8	Sunday 22 March 2020	4
Monday 23 March 2020 at 17:14	Current	6	Monday 23 March 2020	5
Tuesday 24 March 2020 at 17:14	Hypothetical	7	Tuesday 24 March 2020	5
Wednesday 25 March 2020 at 17:16	Hypothetical	6	Wednesday 25 March 2020	4
Thursday 26 March 2020 at 17:17	Hypothetical	5	Thursday 26 March 2020	6
Friday 27 March 2020 at 17:17	Current	4	Friday 27 March 2020	6
Saturday 28 March 2020 at 18:18	Current	6	Saturday 28 March 2020	1
Sunday 29 March 2020 at 18:18	Hypothetical	5	Sunday 29 March 2020	5
Monday 30 March 2020 at 18:21	Current	4	Monday 30 March 2020	8
Tuesday 31 March 2020 at 18:19	Hypothetical	5	Tuesday 31 March 2020	10

On the details view you are shown some stats about moods and worries on each day of the week. You are also given a list of the moods and worries that the patient has logged, each item in these lists is clickable to reveal more details about the mood or worry. You can also change the sort order of lists by clicking the heading at the top, and you can go between pages by using the controls at the bottom of each list, which will also allow you to change the number shown per page.

Edit Profile

Logout Edit Profile Logout

Edit Your Profile

First Name *
Bob

Surname *
Bobertson

Email *
bob@therapist.com

Submit

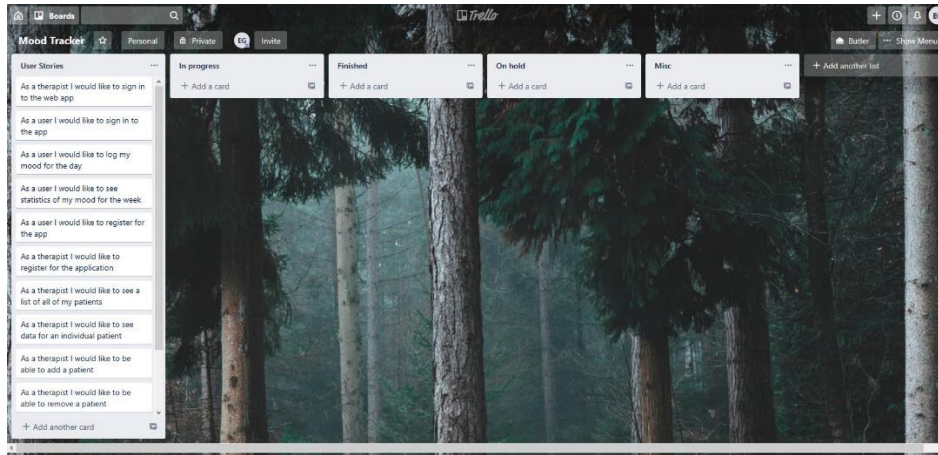
Delete Account

On the edit profile page you can change and update your data. If any of the fields are invalid they will be highlighted in red and you will not be able to submit your changes. When you are done though, hit submit and the changes will be saved, or you will be notified if you weren't successful. You are also able to delete your profile from this page.

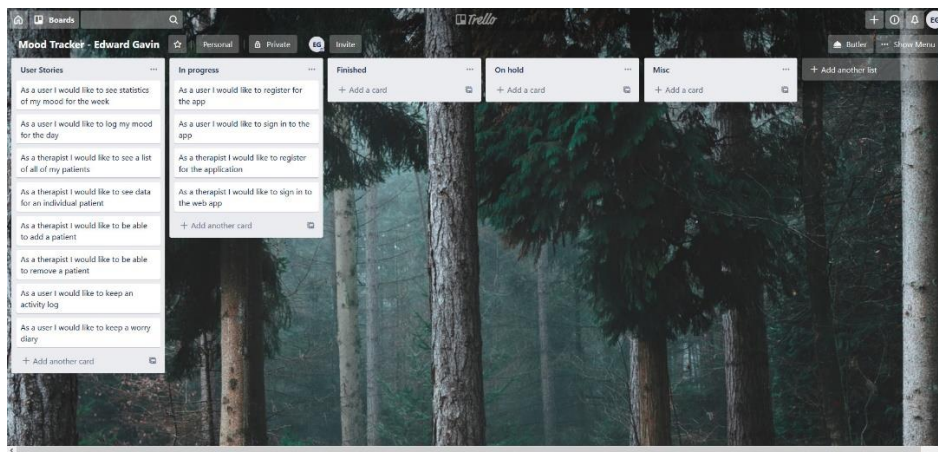
Appendix B. Project Management

Trello Screenshots

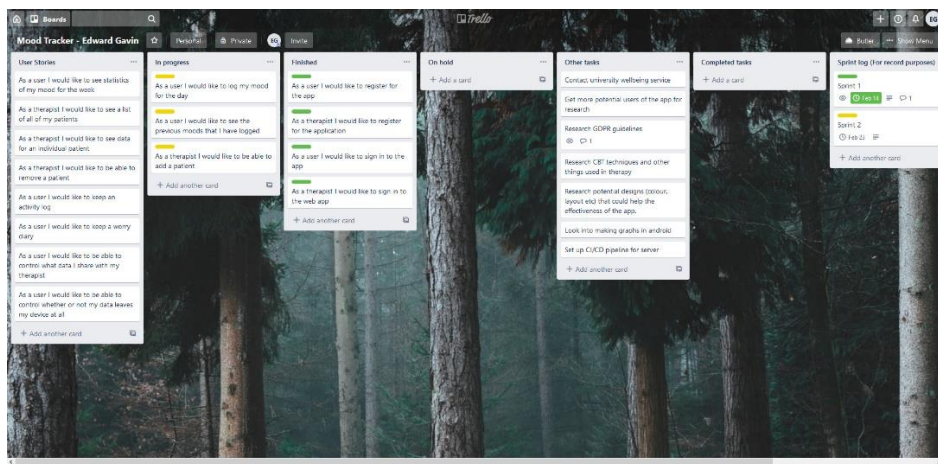
Trello 1



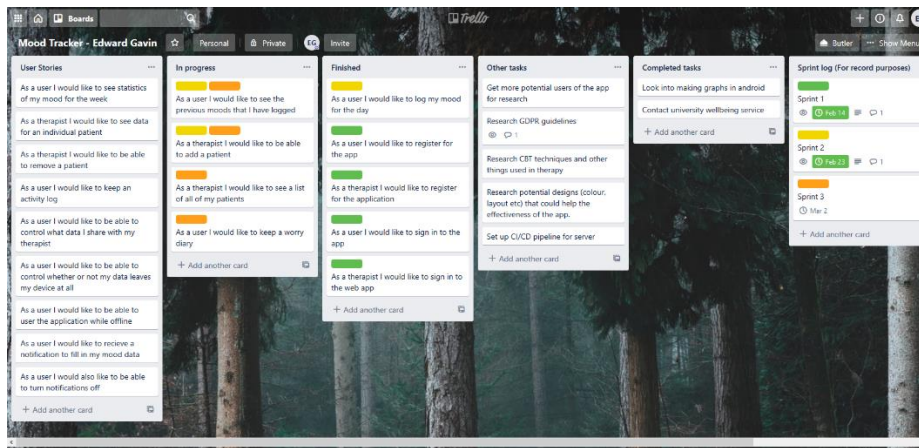
Trello 2/3 – Due to no functional progress being made, the board was the same between these two weeks.



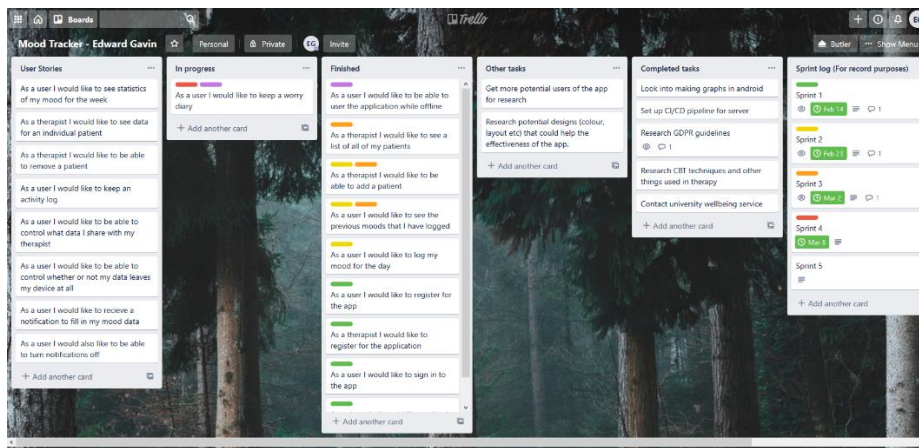
Trello 4.



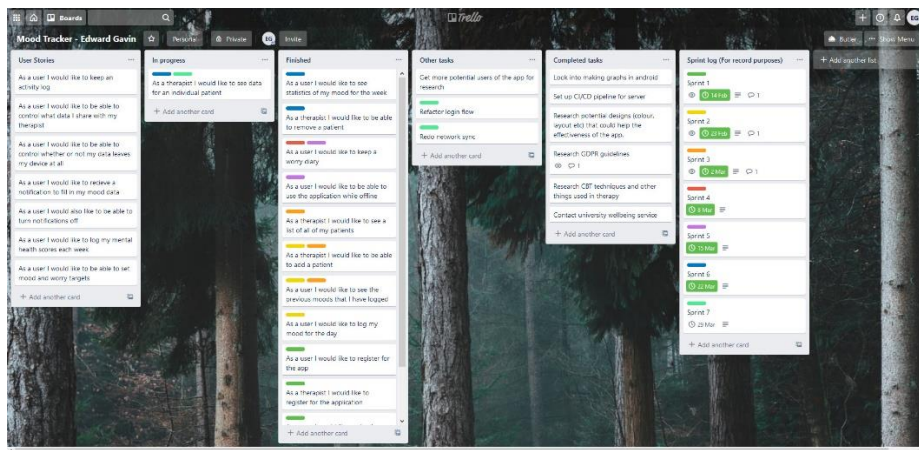
Trello 5



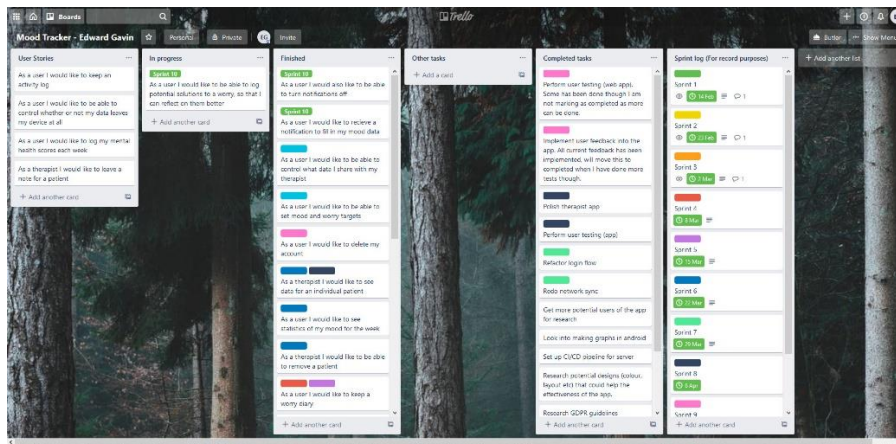
Trello 6



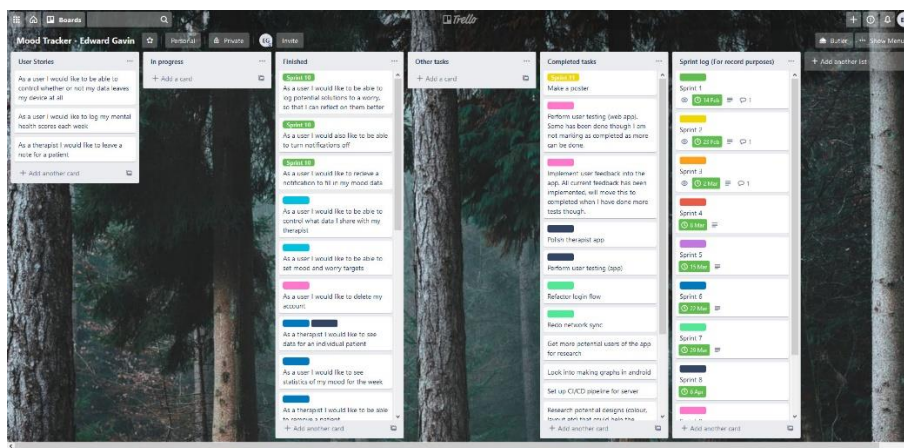
Trello 7



Trello 8



Trello 9



Sprint Reviews

Sprint 1.

Dates: Project Start (27th Jan) to 14th Feb)

Tasks:

- Log in (User and Therapist)
- Register (User and Therapist)

Notes:

First sprint will also involve set up, research and deciding on specific technologies, meaning it will be longer than subsequent sprints)

Outcome:

All tasks for this sprint were completed within the specified time frame. Authorisation has been set up on both apps, passwords are encrypted, and https is in use for all connections. Currently database is minimal for user information as I've only added what could potentially be needed, flexibility of MongoDB will allow me to add whatever I need later if a new feature is added.

Sprint 2.

Dates: Friday 14th Feb - Sunday 23rd

Tasks:

- Implement core functionality of the mobile app (Log and View Moods)
- Add the ability for a therapist to add a patient to their list

Non-functional tasks:

- Contact relevant users for each side of the app
- Research more into legal issues such as GDPR
- Research into social and ethical parts as well.
- More design research now that Auth functionality exists

Outcome:

More difficulties were encountered on this sprint than I had anticipated. Choosing which persistence library to use ended up causing a lot of trouble, as both Realm and ObjectBox were valid options considering the NoSQL backend. However, after a day of trying out a small implementation of each, ObjectBox's lack of features and Realms lack of support for coroutines lead me to return to using Room, the android ORM for SQLite due to its tight integration with the android system, specifically LiveData and coroutines. Research was conducted though, both into current options that are available on the market, as well as current research into CBT.

Sprint 3.

Dates: 23rd Feb – 2nd March

Tasks:

- Finish unfinished tasks from sprint 2
- Add the ability for patients and therapists to view and manage their links

Non-functional tasks:

- Re-contact university wellbeing
- Perform market research

Outcome:

A few problems were encountered during the sprint, though most were overcome with some relative ease. Market research was good, new ideas gotten for the app.

Sprint 4.

Dates: 2nd March - 8th March

Tasks:

- Add worry logging into the mobile application

Non-functional tasks:

- Add unit tests and set up CI for the server

Outcome:

Some challenges were encountered during this sprint regarding technology, was unable to complete worry diary functionality due to this, so that will be carried over into the next week. Server now has CI pipeline set up for continuous integration.

Sprint 5.

Dates: 8th March – 15th March

Tasks:

- Continue adding worry diary functionality
- Complete the network sync functionality of the application

Non-functional tasks:

- Increase test coverage

Outcome:

Overall this sprint was a success. Worry diary functionality was completed this week and network sync functionality was finished, allowing for offline use of the application.

Sprint 6.

Dates: 15th March – 22nd March

Tasks:

- Create initial dashboard for the user application
- Add user statistics viewing to the therapist application (This will be ongoing through a few sprints)
- Add the ability for therapists to remove patients

Non-functional tasks:

- Do some more research for extension features

Outcome:

Overall this sprint was a success. The initial dashboard was implemented into the user application, and the statistics area on the therapist application has parity with the current information available. Therapists are now also able to remove patients. Research turned up some interesting data on notifications and habit formation, will look to include this in the application going forwards.

Sprint 7.

Dates: 22nd March – 29th March

Tasks:

- None this week

Non-functional tasks:

- Refactor the login flow of the application
- Redo network sync to be more efficient

Outcome:

Had to move around a lot this week due to the current situation, as such, functional tasks were put on hold. However, the non-functional tasks and more were completed, the application now uses fragments on the login screen and goes to a loading screen before the dashboard, rather than fetching data while on the dashboard. Network sync now queues workers right after adding, rather than doing it periodically. On the back of the network sync changes, the repository layer was also refactored, now all methods return Resource classes which can contain the status as

well as the data. Was also able to do a large-scale UI refactor, making the whole application look a lot cleaner.

Sprint 8.

Dates: 29th March – 5th April

Tasks:

- Finish patient view on the therapist app

Non-functional tasks:

- Start performing user testing on the user application

Outcome:

Therapist app is completely up to it's MVP as well now so user testing on that can begin soon, a little later than hoped but not too much of a setback. Was able to get some initial feedback for the mobile app tests, though will continue to test over the next few weeks and implement liked features as more responses come through.

Sprint 9.

Dates: 5th April – 12th April

Tasks:

- None this week

Non-functional tasks:

- Implement bug fixes from android app testing
- Get the web app hosted and started user testing for that

Outcome:

Overall progress was good this sprint, ended up polishing the web application a bit more before hosting it. Now has some cool interactivity on the graphs so you can bring up mood or worry data easily, also added AuthGuards so that routes can't be accessed unless authenticated. Web app is now hosted with the API so should be able to get some more test data for that now as well as the android application.

Sprint 10.

Dates: 12th April – 19th April

Tasks:

- Add data sharing controls to the android application
- Add targets to the android application

Non-functional tasks:

- Implement web app feedback if any is received
- Perform more testing

Outcome:

Overall this sprint went well. Users can now add targets on their accounts and see this on their dash (they will be prompted to set some if they haven't). Users can also control what data the therapist sees. Was also able to refactor the back end a bit and clean up the code while I was adding the server-side requirements for these features. Was also able to add a bit of branding to both applications and was able to allow therapists to see a user's targets as well on the relevant graph.

Sprint 11.

Dates: 19th April – 26th April

Tasks:

- Add notification functionality to the user application
- Add the ability to add solutions to worries

Non-functional tasks:

- None this week

Outcome:

More strong progress this sprint. Both notifications and the ability to add solutions to a worry were added this week, I was also able to make a few small bug fixes and tweaks to each application. Need to start thinking about winding down going forwards now and putting more focus into report writing.

Sprint 12. - Final

Dates: 26th April – 3rd May

Tasks:

- Add more stats relating to worries and solutions on the dashboard
- Add solution-based stats to the therapist app

Non-functional tasks:

- Make a poster for the showcase
- Improve test coverage

Outcome:

Good progress, all functionality was completed within the time allotted and both apps are now in a finished state. A poster was also created for the project showcase and

has been uploaded to the DLE. The final few 10 days of this project will no longer follow the sprint structure as they are mostly going to be based around writing the report and making sure the application is as bug free as it can be before I submit it.

Risk Analysis

<i>Risk</i>	<i>How to Manage it</i>
<i>Data loss</i>	Strong use of version control will happen through the entire development process. At the end of the day all commits should be push to the remote repository to ensure that at most, a device failure would cause the loss of a days' worth of data.
<i>Illness or Emergency</i>	In the event of this happening, the project supervisor would be notified immediately. If the illness was severe enough to make it that no work could be accomplished. An extenuating circumstances application would be considered.
<i>Difficulty with new technologies</i>	As there are technologies involved within the project that are new to the developer, difficulties are to be expected. The first period of the project will be spent becoming familiar with these technologies, if there are any difficulties after this, then the management plan for scheduling setbacks will be used.
<i>Scheduling setbacks</i>	In the event of severe difficulties, or other unforeseen circumstances that could cause delays within the project, enough time has been left at the end to ensure that there is a buffer, allowing for some scheduling setbacks to not effect the progress of the project.

Appendix C. SWOT Analysis

In order to better understand MoodSage's place in the market, this SWOT analysis was undertaken early in the project to identify where it would be able to sit.

<p style="text-align: center;">Strengths</p> <ul style="list-style-type: none"> • Application features are based on research into current CBT techniques and are shown to be effective • Minimum product is structured so that it would be an effective tool, further development will only help to improve it • User interface is designed to help the user, not just facilitate their actions • As a native application it should have much greater performance 	<p style="text-align: center;">Weaknesses</p> <ul style="list-style-type: none"> • Android only could lock out a large portion of the market • Although allowing for more specialised features, primary focus on anxiety will further restrict the market • Will need to have the trust of users in order to gain a share in the market, as such, certain monetisation schemes such as personalised advertisements, could not be used as users would feel we are using their data for money.
<p style="text-align: center;">Opportunities</p> <ul style="list-style-type: none"> • There are no applications that provide this level of integration between clients and their therapists when looking at long term solutions. • Integration with therapist side does open an opportunity for promoting the application through them, in order to build an initial user base. 	<p style="text-align: center;">Threats</p> <ul style="list-style-type: none"> • Heavy saturation within the market, could be hard to gain any share if this application was released • As well as existing saturation, main competitors, such as Daylio already have large, established user bases, could cause problems as the application would need to convert a lot of users in order to be successful.

Appendix D. Third Party Resources Used

Android

- Room - <https://developer.android.com/topic/libraries/architecture/room>
- Retrofit - <https://square.github.io/retrofit/>
- Moshi - <https://github.com/square/moshi/>
- Dagger 2 - <https://dagger.dev/>
- Work Manager - <https://developer.android.com/topic/libraries/architecture/workmanager>
- Material Calendar View - <https://github.com/prolificinteractive/material-calendarview>
- Jetpack Navigation - <https://developer.android.com/guide/navigation/navigation-getting-started>
- MpAndroidChart - <https://github.com/PhilJay/MPAndroidChart>
- ThreeTenBP/ABP - <https://github.com/ThreeTen/threetenbp/>
<https://github.com/JakeWharton/ThreeTenABP>
- MockK – <https://mockk.io/>
- Hamcrest – <http://hamcrest.org/>
- Robolectric - <http://robolectric.org/>
- GetOrAwaitValue – <https://codelabs.developers.google.com/codelabs/advanced-android-kotlin-training-testing-basics/#8>

Node

- Bcrypt - <https://www.npmjs.com/package/bcrypt>
- Cors - <https://expressjs.com/en/resources/middleware/cors.html>
- Express - <https://expressjs.com>
- Jsonwebtoken - <https://www.npmjs.com/package/jsonwebtoken>
- Mongoose - <https://mongoosejs.com/>
- Passport - <http://www.passportjs.org/>
- Mocha - <https://mochajs.org/>
- Chai - <https://www.chaijs.com>
- Supertest - <https://www.npmjs.com/package/supertest>

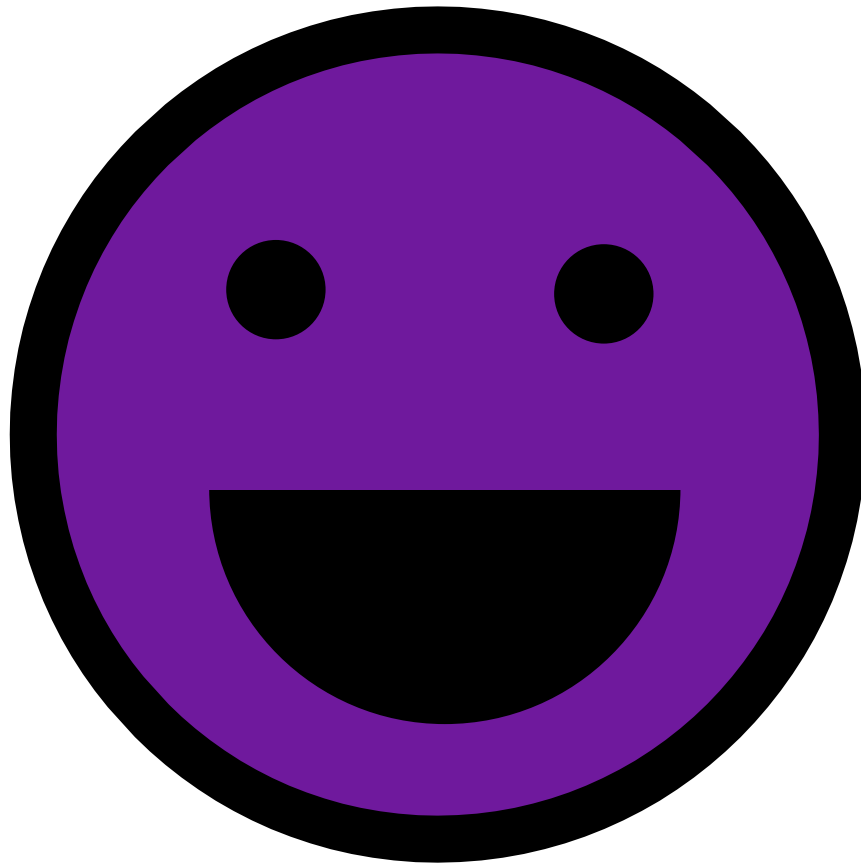
Angular

- Material components - <https://material.angular.io/>
- Ng2 charts - <https://valor-software.com/ng2-charts/>
- Chartjs-plugin-annotation - <https://www.npmjs.com/package/chartjs-plugin-annotation>
- Ngx-cookie-service - <https://www.npmjs.com/package/ngx-cookie-service>

Icons

- Material Icons - <https://material.io/resources/icons/?style=baseline>

Appendix E. Logo



MoodSage

In order to be recognisable, the Logo uses a simple design with a happy face to clearly represent what the application is trying to achieve. It uses purple, the core colour of the application to allow the user to build an association within their head.

Appendix F. UI Design Breakdown

In order to build effective user interfaces, strong principles must be followed when designing them. Shneiderman's "8 Golden Rules of Interface Design" (Shneiderman, et al., 2010) provide these principles, and by following them, the overall design of the applications interface can be made much more intuitive for the user.

The first rule is to "strive for consistency" within the design of the application. This can be achieved by making buttons with similar actions look the same, make sequences of actions consistent and making menus and other areas of the application consistent. As an example of this from the Android application, other than the strict colour scheme, the buttons have been designed so that users can identify their function easily once they have seen one example (Figure A-1). Constructive actions within the application, such as creating a new mood or worry, have purple buttons with white text. This is then reversed for buttons that perform destructive actions within the application, where they now have purple text and a white background, helping to symbolize, even when glancing at them, that they do opposing actions.



Figure F-1: Example of two buttons from the application.

As another example of the way these rules have been followed within the application, rules 3 and 4 say to "offer informative feedback" and to "design dialog to yield closure." Aside from the simpler examples, such as click animations on buttons, this application has been designed to be constantly communicating with the user in some way when they perform an action. For example, when the user attempts to change their profile details, the application will communicate the status of this with them, whether it succeeds or fails. In any of the information logging areas, the application will also clear the screen if the user was successful in logging the information, but leave it if there was an error, while still notifying the user of the status regardless. This style of clear communication with the user allows them to be sure that their actions went through and be confident in then moving to another section of the application to do something else.

To "support the internal locus of control" is the seventh rule. Because users prefer to feel in control of a system, rather than the system telling them where to go, it is important that they are the ones making the actions within the application. The application supports this by not making any redirects or changing screens without the users input. All non-navigation actions within the application do not provide any forced navigation to another screen, with the exception of logging in and out of the application, where the user would expect to be taken to a new screen or when the action would destroy the data that screen is presenting.

As well as the rules set out by Shneiderman, Gerlach and Kuo (1991) discuss the use of mental models and metaphors in building effective user interfaces. Mental models are the client's own ideas of how the system should function, based off their

usage to date, or based off usage of similar applications in the past. It is important to support these ideas of how the application should function and to build the application around that, rather than forcing the client to learn a new workflow or style. An example of this is the usage of the bottom navigation bar for providing the main navigation functionality (Figure F-2). This style was used because many modern applications use this navigation style, and because of this, many clients will immediately understand how to navigate through the application.

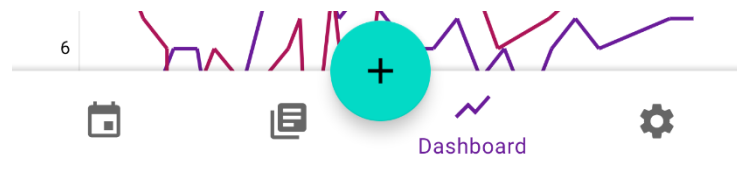


Figure F-2: The application's bottom navigation bar

Using metaphors within an application involves using symbols, drawn from the real world or other things the user might be familiar with, in order to effectively convey what a specific button or section of the application will do. In using metaphors, you can make your interface less cluttered and less text based, instead giving users the ability to understand the functions of the application by relating it to their real-world experiences, making the application feel more recognisable and familiar to them. Some examples of this usage can also be seen in figure F-2.

In conclusion, these principles have been used throughout the development of both applications in order to help the user understand the application and in order for it to become more familiar to them, hopefully increasing their willingness to use the application, and the trust that they have with it. Further research into HCI could still be performed though in order to make the interface even better.

References

Gerlach, J. H. & Kuo, F.-Y., 1991. Understanding HumanComputer Interaction for Information Systems Design. *MIS Quarterly*, 15(4), pp. 527 - 549.

Shneiderman, B., Plaisant, C., Cohen, M. & Jacobs, S., 2010. *Designing the User Interface: Strategies for Effective Human-Computer Interaction..* 5th ed. s.l.:Pearson.

Appendix G. Initial Designs

Before development of many parts of the application took place, a rough outline would be created of the expected UI in order to provide guidance while creating it.

Mobile App



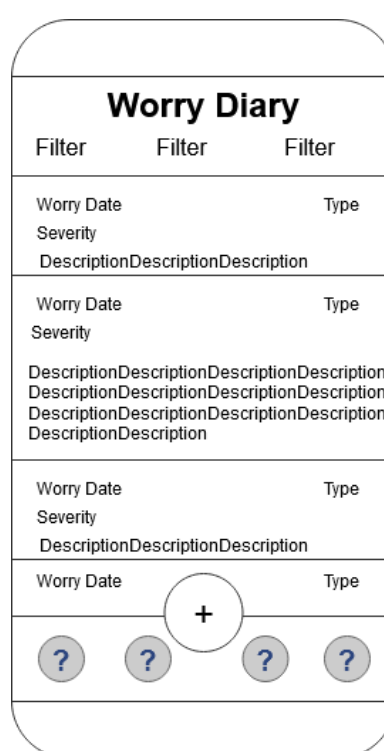
Mobile app login screen design. It features a large placeholder for a profile picture at the top. Below it are input fields for 'Email' and a password field (indicated by asterisks). At the bottom, there is a blue 'Log In' button, a link for 'No Account?', and a blue 'Register' button.



Mobile app register screen design. It has a title 'Register' at the top. Below the title are input fields for 'Name', 'Last Name', and 'Email'. There are two password fields (indicated by asterisks). At the bottom, there is a blue 'Register' button.



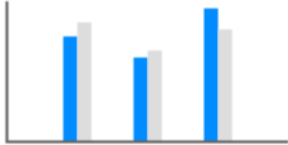
Mobile app mood diary screen design. It features a calendar for February 2020. The calendar shows dates from 1 to 28. The date 24 is highlighted in blue. Below the calendar, there is a section titled 'Mood Date' with a 'Rating: 6'. Below the rating, there are four input fields for 'Comments'. At the bottom, there is a large circular button with a '+' sign and four circular buttons with question marks.




Mobile app worry diary screen design. It has a title 'Worry Diary' at the top. Below the title, there are three 'Filter' buttons. The screen is divided into three sections, each containing input fields for 'Worry Date', 'Severity', 'Type', and 'Description'. At the bottom, there is a large circular button with a '+' sign and four circular buttons with question marks.

Dashboard

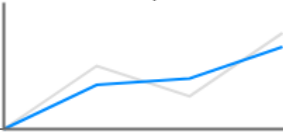
Graph 1



Graph 2



Graph 3



Profile

Forename

Surname

Email

Targets

Mood

Worry

Submit Changes

+

?

?

?

?

Log a Mood

Date and time

Set Date

Type

☐ Current ☐ Hypothetical

Severity

1

Comments

Submit

+

?

?

?

?

Log a Worry

Date and time

Set Date

Type

☐ Current ☐ Hypothetical

Severity

1

Comments

Submit

+

?

?

?

?

Web App

Page 1

TherapistApp.com

MoodSage

Profile Log out

Dashboard

Therapist Name

Email

Other details

Add a patient

Add

Patients

Patient Name

Patient email

Status: Active

View Remove

Patient Name

Patient email

Status: Active

View Remove

Patient Name

Patient email

Status: Pending

Remove

Patient Name

Patient email

Status: Active

View Remove

Page 1

TherapistApp.com

MoodSage

Profile Log out

Patient Name

Patient email

Other info

Quick Stats Box

Stats

StatisticsDetails

Worry	Stats

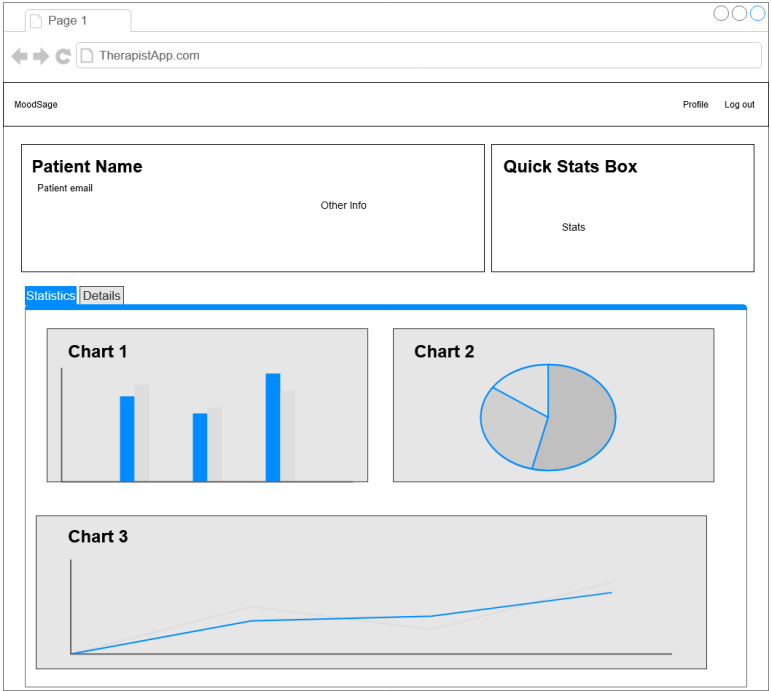
Worry
Worry
Worry

Mood	Stats

Mood
Mood
Mood

Worry Date	Type	Severity
Description		

Mood Date	Rating
Comments	



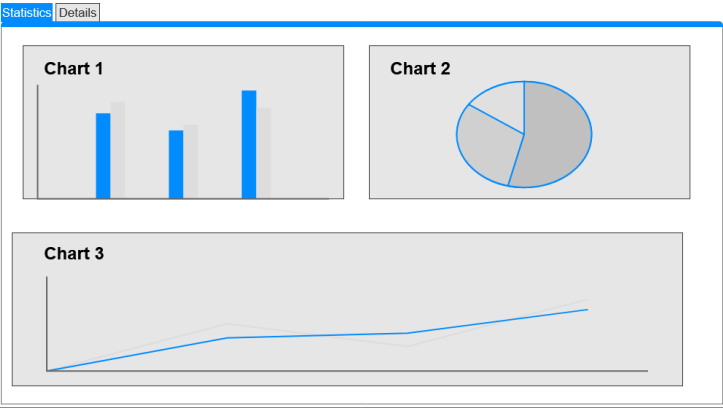
Patient Name

Patient email

Other Info

Quick Stats Box

Stats



Appendix H. API Routes

Route	Verb	Description	Patient	Therapist
/auth/login	POST	Authenticate, valid request returns JWT		
/auth/register	POST	Register a new account on the server		
/api/profile	GET	Returns user's profile details		
	PUT	Amends user's profile details		
	DELETE	Deletes a profile from the server		
/api/profile/links	GET	Get all accounts that are linked with a user		
	POST	Add a new account link		
/api/profile/link/:id	PUT	Update account link status		
	DELETE	Remove an account link		
/api/profile/moods	GET	Returns all a user's moods		
	POST	Adds a new mod to a user's profile		
/api/profile/moods/:date	DELETE	Deletes a specific mood from a user		
/api/profile/worries	GET	Returns all a user's worries		
	POST	Adds a new worry to a user's profile		
/api/profile/worries/:date	DELETE	Deletes a specific worry from a user		
/api/profile/worries/:date/solutions	POST	Add a new solution to a specific worry		
/patient/:id	GET	Get all of a patients data		

Appendix I. User Stories

- As a therapist I would like to sign into the web app
- As a user I would like to sign into the mobile app
- As a therapist I would like to register for the application
- As a user I would like to register for the application
- As a user I would like to log my mood for the day
- As a user I would like to see the previous moods that I have logged
- As a therapist I would like to be able to add a patient
- As a therapist I would like to see a list of all my patients
- As a user I would like to be able to use the application while I am offline
- As a user I would like to keep a worry diary
- As a therapist I would like to be able to remove a patient
- As a user I would like to see statistics of my mood for the week
- As a therapist I would like to see data for an individual patient
- As a user I would like to delete my account
- As a user I would like to be able to set mood and worry targets so that I can better track my progress
- As a user I would like to be able to control what data I share with my therapist
- As a user I would like to receive a notification to fill in my mood data so that I do not forget
- As a user I would like to be able to turn notifications off
- As a user I would like to be able to log potential solutions to worries so that I can reflect on them better

Appendix J. Usability Testing Results

Mobile Application

Tasks List:

Set One: New account

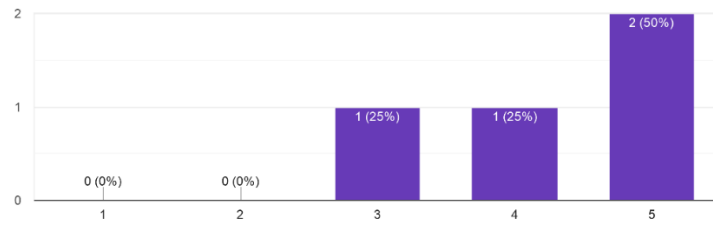
- Register an account and sign in to the app
- Log a new mood (any details)
 - View the mood you have just logged
- Log a new worry (any details)
 - View the worry that you have just logged
- Disable the internet connection on your device and log a new worry
 - View the worries again, what's different.
 - Without leaving the screen, re-enable the internet connection on your device and wait a few seconds. Does anything change?
- Log out of the application.

Set Two: Existing account

- Sign into the app:
 - email: john@test.com
 - password: 123456
- View all the graphs on the dashboard
 - Does this user have more hypothetical or current worries
 - Is this user's overall mood going up or down
 - On what day are the users worries most severe
- View all the moods this user has logged for each day
 - View the details of a low rated mood
- View all the worries that a user has logged
 - Filter the worries so that you can see only current worries with a severity of 5 or greater, posted in the last week
- View the current therapist account(s) that are linked with this account
- Explore the app on your own for a bit.

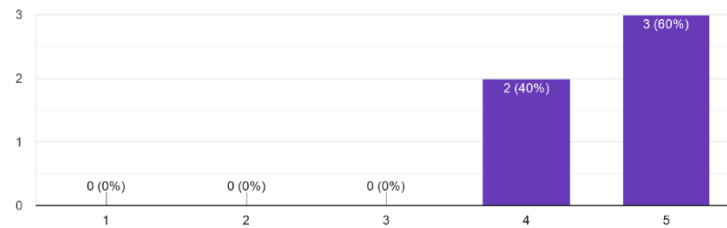
How easy was it to register an account and sign in?

4 responses



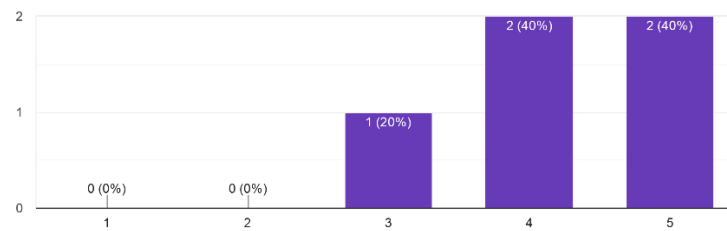
How easy was the process of adding a mood and viewing it?

5 responses



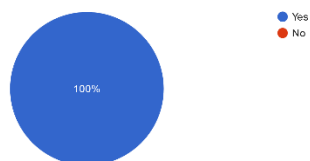
How easy was the process of adding a new worry and viewing it?

5 responses



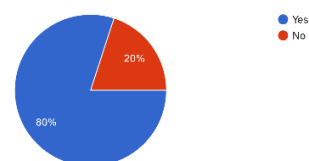
Did the automated sync work correctly when you turned your connection back on?

5 responses



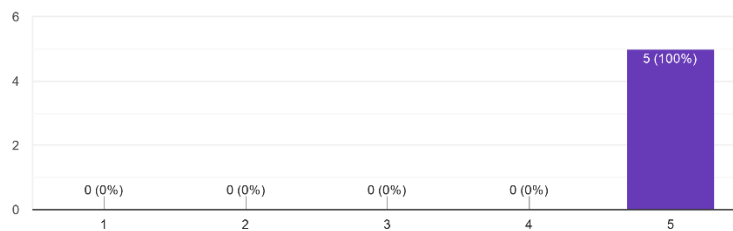
Were you able to easily tell which moods were synced and which weren't?

5 responses



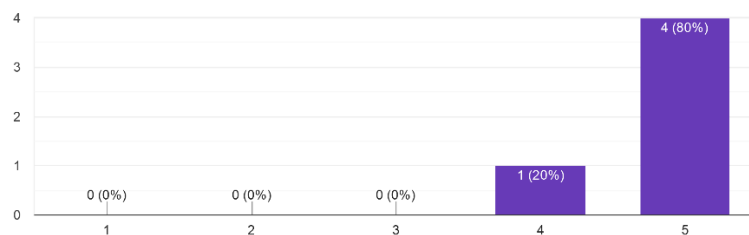
How easy was it to log out of the application?

5 responses



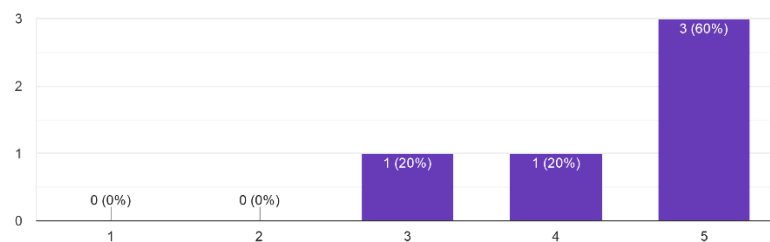
How easy was it to get the information in the task list from the dashboard?

5 responses



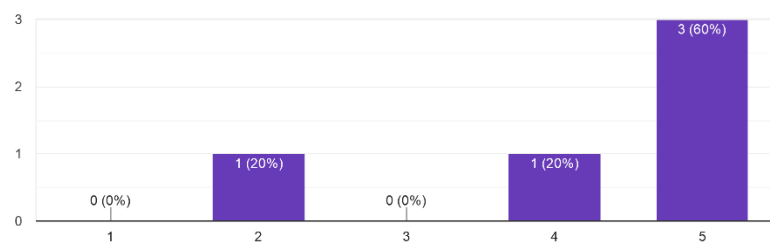
How easy was it to pick a low rated day on the calendar page?

5 responses



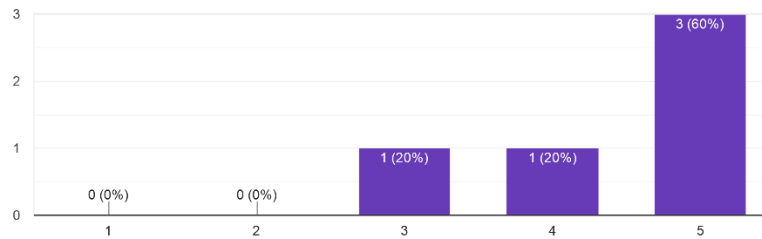
How easy was it to use the filter options on the worry diary?

5 responses



How easy was it to find the page to manage your links?

5 responses



Overall, the reception to the mobile application was very positive, though there were still some bugs and other issues reported by the testers. Some of these were simple and easy to fix, such as the “Log a Worry” page incorrectly being titled “Log a Thought,” which did confuse some of those who were testing the application. Other testers suggested some simple features to add as well, such as showing the average mood rating for the current month on the calendar page, which were incorporated into the final application.

Another issue that was reported with the application, was that one of the filter options in the worry page didn’t do what it said it would. The filter was just titled “Severity,” giving the impression that it would filter to that exact severity, however, the filter set the minimum severity that would be shown on the worry diary. In response to this feedback, the button was re-titled to “Min Severity” and a new filter option was added that would set the maximum severity to be shown.

There was also an issue for some users with finding the page to manage account links, in the written feedback this was reported to be from not understanding what it was. Because of this, a small subtitle was added to the button on the settings page, explaining its purpose, though in a full release of the application, details like this would be explained on the store page as well.

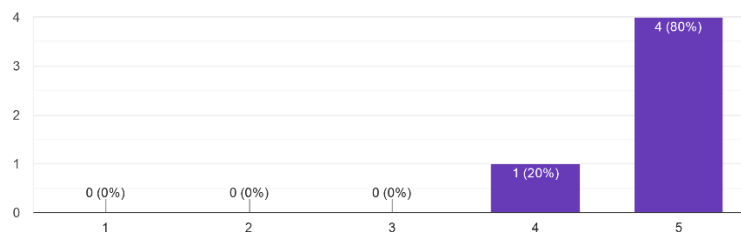
Web Application

Task List:

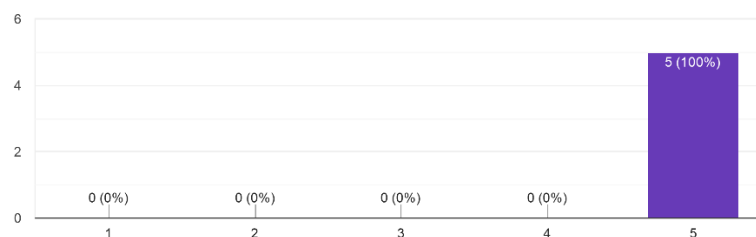
- Register for an account
 - Log in to MoodSage with these Details (Using this account instead as it has a pre-linked patient)
 - Email: bob@therapist.com
 - Password: 123456
- From the dashboard, add a patient (Email: jim@user.com)
- Remove the patient you just added.
- View the details for John Smith
 - View the graphs for John Smiths data
 - View Johns details in a written format.
 - Can you interact with anything on this tab to see more details?
 - Going back to the statistics page, can you view specific details for an individual mood or worries?
- Log out of the application

Results:

How easy was the registration process?
5 responses

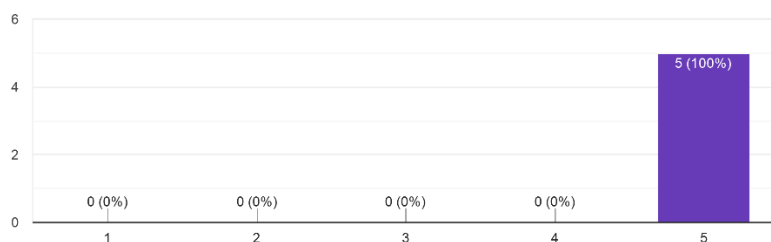


How easy was the log in process?
5 responses



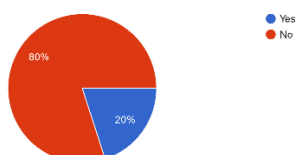
How easy was it to add, remove and view a patient?

5 responses



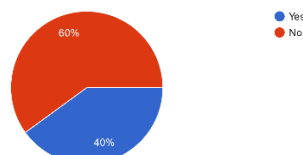
Was it obvious that you could click on an individual mood or worry on the line graph to pull up more details about it?

5 responses



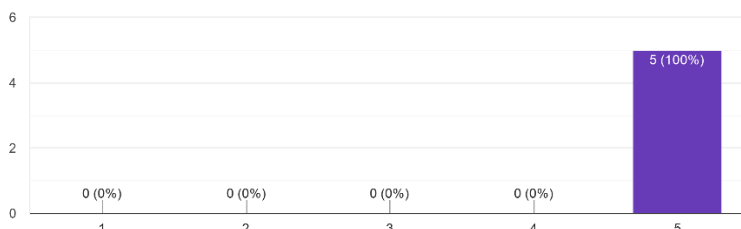
On the details page, was it obvious that you could click on a mood or worry in the list to expand it and view the description?

5 responses



How easy are all of the stats to understand?

5 responses



Overall, the reception to this application was very positive and those who tested it found it very intuitive to use. One bug was reported from this testing, which was that the log out button would not function properly unless the tester clicked exactly on the text as clicking on the surrounding area would only log them out but wouldn't navigate them away from the page, this was quickly fixed.

The other problems with the application that the testers reported, was to do with the less obvious functionality of the application where you could click on certain parts to reveal more details. These two questions both had an overly negative response, as such, a few changes were made to the web application. On the graph, a subtitle was added notifying the user that they could click on any of the data points. On the table, as well as changing the cursor type, the background of each row was highlighted, making it more obvious that it could be interacted with.

Appendix K. Functional Testing

Mobile application

Test No.	Description	Directions	Inputs	Expected Result	Actual Result
1	Register for an account	1. Click "Register" 2. Enter details 3. Click "Register"	First Name: Edward Last Name: Gavin Email: test@email.com Password: P4ssword Confirm: P4ssword	Account should be created successfully and user returned to the login screen	Application functioned as expected
2	Register for an account with no details	1. Click "Register" 3. Click "Register"	N/A	An error message should be displayed saying that there is no name, user should remain on the registration screen	Application functioned as expected
3	Register for an account with invalid email	1. Click "Register" 2. Enter details 3. Click "Register"	First Name: Edward Last Name: Gavin Email: testemailcom Password: P4ssword Confirm: P4ssword	An error message should be displayed saying that the email is invalid, user should remain on the registration screen	Application functioned as expected
4	Register for an account with invalid password	1. Click "Register" 2. Enter details 3. Click "Register"	First Name: Edward Last Name: Gavin Email: test@email.com Password: password Confirm: password	An error message should be displayed saying that the password is invalid, user should remain on the registration screen	Application functioned as expected
5	Register for an account with non-matching passwords	1. Click "Register" 2. Enter details 3. Click "Register"	First Name: Edward Last Name: Gavin Email: test@email.com Password: Passw0rd Confirm: P4ssword	An error message should be and say the passwords do not match, user should remain on the registration screen	Application functioned as expected
6	Log in	1. Enter details 2. Click "Sign In"	email: test@email.com password: P4ssword	The application should progress to the loading screen, then the dashboard.	Application functioned as expected
7	Log in with no connection	1. Enter details 2. Click "Sign In"	email: test@email.com password: P4ssword	The application should display a message that there is a network error.	Application functioned as expected

8	Log in with wrong details	1. Enter details 2. Click "Sign In"	email: not@account.com password: password	The application should display a message that the email/password is invalid	Application functioned as expected
9	Log out	1. Enter details 2. Click "Sign In" 3. Click "Settings" 4. Click "Log Out" 5. Click "Yes" when prompted	email: test@email.com password: P4ssword	The application should log out, returning the user to the log in screen and clear the back stack.	Application functioned as expected
10	Delete account	1. Enter details 2. Click "Sign In" 3. Click "Settings" 4. Click "Delete Account" 5. Click "Yes" when prompted	email: test@email.com password: P4ssword	The application should return to the user to the log in screen and clear the back stack.	Application functioned as expected
11	Log a mood	1. Enter details 2. Click "Sign In" 3. Click the centre + button 4. Click "Log a Mood" 5. Enter details 6. Click "Submit"	email: test@email.com password: P4ssword date: 10/05/2020 rating: 6 comment: test private: false	The application should say the mood has been successfully logged and clear the screen.	Application functioned as expected
12	View previously logged moods	1. Enter details 2. Click "Sign In" 3. Click the centre + button 4. Click "Log a Mood" 5. Enter details 6. Click "Submit" 7. Click "Calendar"	email: test@email.com password: P4ssword date: 10/05/2020 rating: 6 comment: test private: false	The application should display a yellow circle on May 10 th .	Application functioned as expected
13	View a specific previous mood	1. Enter details 2. Click "Sign In" 3. Click the centre + button 4. Click "Log a Mood" 5. Enter details 6. Click "Submit" 7. Click "Calendar" 8. Click on May 10th	email: test@email.com password: P4ssword date: 10/05/2020 rating: 6 comment: test private: false	The application should display the details of the mood below the calendar.	Application functioned as expected
14	Log a worry	1. Enter details 2. Click "Sign In" 3. Click the centre + button	email: test@email.com password: P4ssword	The application should display an error saying the worry has been	Application functioned as expected

		4. Click "Log a Worry" 5. Enter details 6. Click "Submit"	date: 10/05/2020 at 17:00 type: current severity: 7 description: test private: false	logged successfully and reset the screen.	
15	Log a worry with no description	1. Enter details 2. Click Sign In 3. Click the centre + button 4. Click Log a Worry 5. Enter details 6. Click "Submit"	email: test@email.com password: P4ssword date: 10/05/2020 at 17:00 type: current severity: 7 description: test private: false	The application should display an error message and leave the screen as it is.	Application functioned as expected
16	View all logged worries	1. Enter details 2. Click "Sign In" 3. Click the centre + button 4. Click Log a Worry 5. Enter details 6. Click "Submit" 3. Click on "Worry Diary"	email: test@email.com password: P4ssword date: 10/05/2020 at 17:00 type: current severity: 7 description: test private: false	The application should display the worry in the diary, with an orange bar on the left.	Application functioned as expected
17	View all worries with filters	1. Enter details 2. Click "Sign In" 3. Click on "Worry Diary" 4. Set type to hypothetical 5. Set type to all 6. Set max severity to 8	email: test@email.com password: P4ssword date: 10/05/2020 at 17:00 type: current severity: 7 description: test private: false date: 09/05/2020 at 17:00 type: hypothetical severity: 9 description: test private: false	The application should first display 2 worries, then 1, then 2 again and then 1 again.	Application functioned as expected
18	View account links	1. Enter details 2. Click "Sign In" 3. Click "Settings" 4. Click "Manage Links"	email: test@email.com password: P4ssword	The application	Application functioned as expected

19	View account links with no internet connection	1. Enter details 2. Click "Sign In" 3. Click "Settings" 4. Click "Manage Links"	email: test@email.com password: P4ssword (Add this account through therapist app)	The application should display the current request to the user.	Application functioned as expected
20	Accept a therapist	1. Enter details 2. Click "Sign In" 3. Click "Settings" 4. Click "Manage Links" 5. Click "Accept"	email: test@email.com password: P4ssword (Add this account through therapist app)	The application should accept the link and then refresh the list to reflect the changes.	Application functioned as expected
21	Reject a therapist	1. Enter details 2. Click "Sign In" 3. Click "Settings" 4. Click "Manage Links" 5. Click "Remove"	email: test@email.com password: P4ssword (Add this account through therapist app)	The application should reject the link, the list should now be empty	Application functioned as expected
22	Remove an account link	1. Enter details 2. Click "Sign In" 3. Click "Settings" 4. Click "Manage Links" 5. Click "Accept" 6. Click "Remove"	email: test@email.com password: P4ssword (Add this account through therapist app)	The application should remove the link, the list should now be empty.	Application functioned as expected
23	Log a mood while offline	1. Enter details 2. Click "Sign In" 3. Click the centre + button 4. Click "Log a Worry" 5. Enter details 6. Click "Submit"	email: test@email.com password: P4ssword date: 12/05/2020 rating: 6 comment: test private: false	The application should still save the mood and upload it when connection is restored.	Application functioned as expected
24	Log a worry while offline	1. Enter details 2. Click "Sign In" 3. Click the centre + button 4. Click "Log a Mood" 5. Enter details 6. Click "Submit" 7. Click on "Worry Diary"	email: test@email.com password: P4ssword date: 12/05/2020 at 17:00 type: current severity: 7 description: test private: false	The application should still save the worry and upload it when connection is restored. There should also be a cloud icon with a line through it displayed next to the worry, this	Application functioned as expected

		8. Go online		should disappear when the mood has been synced.	
25	Edit profile	1. Enter Details 2. Click "Sign In" 3. Click "Settings" 4. Click "My Profile" 5. Edit details 6. Click "Submit Changes"	email: test@email.com password: P4ssword First name: Ed	The application should save the changes to the profile.	Application functioned as expected
26	Edit profile but leave name blank	1. Enter details 2. Click "Sign In" 3. Click "Settings" 4. Click "My Profile" 5. Remove first name 6. Click "Submit Changes"	email: test@email.com password: P4ssword	The application should reject the profile changes and state the name is invalid.	Application functioned as expected
27	Edit profile with invalid email	1. Enter details 2. Click "Sign In" 3. Click "Settings" 4. Click "My Profile" 5. Enter details 6. Click "Submit Changes"	email: test@email.com password: P4ssword email: testemailcom	The application should reject the changes and say the email is invalid.	Application functioned as expected
28	Edit profile while offline	1. Enter details 2. Click "Sign In" 3. Click "Settings" 4. Click "My Profile" 5. Edit details 6. Click "Submit Changes"	email: test@email.com password: P4ssword First name: Ed	The application should reject the changes and inform the user they do not have a connection.	Application functioned as expected
29	Set mood and worry targets	1. Enter details 2. Click "Sign In" 3. Click "Settings" 4. Click "My Profile" 5. Set mood and worry targets 6. Click "Submit Changes"	email: test@email.com password: P4ssword Mood target: 4 Worry target: 5	The application should update the mood and worry targets. The text on the dashboard should no longer be telling the user to set them.	Application functioned as expected
30	View target progress	1. Enter details 2. Click "Sign In" 3. Click "Settings" 4. Click "My Profile" 5. Set mood and worry targets 6. Click "Submit Changes"	email: test@email.com password: P4ssword Mood target: 4 Worry target: 5	The text on the dashboard should show the compare the users targets to their average for this month.	Application functioned as expected

31	Add a solution to a worry	<ol style="list-style-type: none"> 1. Enter details 2. Click "Sign In" 3. Click the centre + button 4. Click "Log a Worry" 5. Enter Details 6. Click "Submit" 7. Click on "Worry Diary" 8. Click on a worry 9. Click on "Add Solution" 10. Enter details 11. Click "Submit" 	<p>email: test@email.com password: P4ssword</p> <p>date: 10/05/2020 at 17:00 type: current severity: 7 description: test private: false</p> <p>description: test advantages: test disadvantages: test</p>	The application should successfully add the solution and then clear the screen.	Application functioned as expected
32	Add a solution to a worry with no data	<ol style="list-style-type: none"> 1. Enter details 2. Click Sign In 3. Click the centre + button 4. Click Log a Worry 5. Enter Details 6. Click "Submit" 7. Click on "Worry Diary" 8. Click on a worry 9. Click on "Add Solution" 11. Click "Submit" 	<p>email: test@email.com password: P4ssword</p> <p>date: 10/05/2020 at 17:00 type: current severity: 7 description: test private: false</p>	The application should reject the solution and inform the user that it is missing data.	Application functioned as expected
33	Add a solution to a worry with no connection	<ol style="list-style-type: none"> 1. Enter details 2. Click "Sign In" 3. Click the centre + button 4. Click "Log a Worry" 5. Enter details 6. Click "Submit" 7. Click on "Worry Diary" 8. Click on a worry 9. Click on "Add Solution" 10. Enter details 11. Click "Submit" 	<p>email: test@email.com password: P4ssword</p> <p>date: 10/05/2020 at 17:00 type: current severity: 7 description: test private: false</p> <p>description: test advantages: test disadvantages: test</p>	The application should save the solution and upload when connection is restored.	Application functioned as expected
34	Receive notifications	<ol style="list-style-type: none"> 1. Enter details 2. Click "Sign In" 	<p>email: test@email.com password: P4ssword</p>	Reminder notification should appear at 5pm after you sign in	Application functioned as expected

35	Disable notifications	1. Enter details 2. Click "Sign In" 3. Go to "Settings" 4. Uncheck "Enable Notifications"	email: test@email.com password: P4ssword	Reminder should no longer appear at 5pm	Application functioned as expected
36	Customise chart appearance	1. Enter details 2. Click "Sign In" 3. Click on "Settings" 4. Set secondary colour to cyan 5. Click on "Dashboard"	email: test@email.com password: P4ssword	Charts on the dashboard should now be purple and cyan	Application functioned as expected
37	Create private mood	1. Enter details 2. Click "Sign In" 3. Click the centre + button 4. Click "Log a Mood" 5. Enter details 6. Click "Submit"	email: test@email.com password: P4ssword date: 12/05/2020 rating: 6 comment: test private: true	The application should save this mood as normal; it should have a key icon next to it in the calendar view and it should not be sent to the therapist.	
38	Create private worry	1. Enter details 2. Click "Sign In" 3. Click the centre + button 4. Click "Log a Worry" 5. Enter details 6. Click "Submit" 7. Click on "Worry Diary"	email: test@email.com password: P4ssword date: 10/05/2020 at 17:00 type: current severity: 7 description: test private: true	The application should save this worry as normal. It should have a key icon next to it in the diary and it should not be sent to the therapist.	

Web App

Test No.	Description	Directions	Inputs	Expected Result	Actual Result
1	Register for an account	1. Click "Register" 2. Enter details 3. Click "Submit"	First name: Dave Surname: Test Email: dave@test.com Password: Passw0rd Confirm password: Passw0rd	Account should be created, user should be given an alert and then taken to the log in screen.	Application functioned as expected
2	Register for an account with no details	1. Click "Register" 3. Click "Submit"	N/A	Submit button should not be clickable	Application functioned as expected
3	Register for an account with invalid email	1. Click "Register" 2. Enter details 3. Click "Submit"	First name: Dave Surname: Test Email: dave123com Password: Passw0rd Confirm password: Passw0rd	Submit button should not be clickable, email box should display an error.	Application functioned as expected
4	Register for an account with invalid password	1. Click "Register" 2. Enter details 3. Click "Submit"	First name: Dave Surname: Test Email: dave@test.com Password: password Confirm password: password	Submit button should not be clickable, password box should display an error	Application functioned as expected
5	Register for an account with non-matching password	1. Click "Register" 2. Enter details 3. Click "Submit"	First name: Dave Surname: Test Email: dave@test.com Password: Passw9rd Confirm password: Passw0rd	Submit button should not be clickable, password box should display an error	Application functioned as expected
6	Log in	1. Click "Log in" 2. Enter details 3. Click "Log in"	Email: bob@therapist.com Password: 123456	The application should take the user to the dashboard	Application functioned as expected
7	Log in with wrong password	1. Click "Log in" 2. Enter details 3. Click "Log in"	Email: bob@therapist.com Password: password	The application should reject the login and display a message that their credentials are incorrect.	Application functioned as expected
8	View list of all accepted and pending patients	1. Click "Log in" 2. Enter details 3. Click "Log in"	Email: bob@therapist.com Password: 123456	The dashboard display John Test as an active patient, with no pending patients	Application functioned as expected

9	Add a new patient	1. Click "Log in" 2. Enter details 3. Click "Log in"	Email: bob@therapist.com Password: 123456 Patient email: jeff@test.com	Patient should be added and appear in pending.	Application functioned as expected
10	Add a new patient that doesn't exist	1. Click "Log in" 2. Enter details 3. Click "Log in" 4. Enter details into "Add a Patient" box 5. Click "Submit"	Email: bob@therapist.com Password: 123456 Patient email: bill531@test.com	Application should reject the addition and display an error	Application functioned as expected
11	Add a therapist as a patient	1. Click "Log in" 2. Enter details 3. Click "Log in" 4. Enter details into "Add a Patient" box 5. Click "Submit"	Email: bob@therapist.com Password: 123456 Patient email: kev@therapist.com	Application should reject the addition and display an error	Application functioned as expected
12	Add self as a patient	1. Click "Log in" 2. Enter details 3. Click "Log in" 4. Enter details into "Add a Patient" box 5. Click "Submit"	Email: bob@therapist.com Password: 123456 Patient email: bob@therapist.com	Application should reject the addition and display an error	Application functioned as expected
13	Remove a patient	1. Click "Log in" 2. Enter details 3. Click "Log in" 4. Enter details into "Add a Patient" box 5. Click "Submit" 6. Click on "Remove" in the new patients entry	Email: bob@therapist.com Password: 123456 Patient email: jeff@test.com	Application should prompt the user to confirm, and then display a message that the user has been removed. User should then be removed from list.	Application functioned as expected
14	View a patient's details	1. Click "Log in" 2. Enter details 3. Click "Log in" 4. Click "View" on John Tests card	Email: bob@therapist.com Password: 123456	Application should be taken to the details page for John Test	Application functioned as expected
15	Filter graphs last month's data only	1. Click "Log in" 2. Enter details 3. Click "Log in" 4. Click "View" on John Tests card 5. Click on "Statistics" 6. Click on "Last Month" on any graph	Email: bob@therapist.com Password: 123456	Graph should now only show the data from the current date till 1 month ago.	Application functioned as expected

16	View patients worries in a list	1. Click "Log in" 2. Enter details 3. Click "Log in" 4. Click "View" on John Tests card 5. Click on "Details"	Email: bob@therapist.com Password: 123456	Application should bring up a list of the patients worries alongside other details	Application functioned as expected
17	View details for a patient's worry	1. Click "Log in" 2. Enter details 3. Click "Log in" 4. Click "View" on John Tests card 5. Click on "Details" 6. Click on a worry in the list	Email: bob@therapist.com Password: 123456	The worry should be expanded to now show more details	Application functioned as expected
18	View patients' moods in a list	1. Click "Log in" 2. Enter details 3. Click "Log in" 4. Click "View" on John Tests card 5. Click on "Details"	Email: bob@therapist.com Password: 123456	Application should bring up a list of the patients worries alongside other details	Application functioned as expected
19	View details for a patient's mood	1. Click "Log in" 2. Enter details 3. Click "Log in" 4. Click "View" on John Tests card 5. Click on "Details" 6. Click on a mood in the list	Email: bob@therapist.com Password: 123456	The mood should be expanded to now show more details	Application functioned as expected
20	View all solutions to a patient's worry	1. Click "Log in" 2. Enter details 3. Click "Log in" 4. Click "View" on John Tests card 5. Click on "Details" 6. Click on a worry in the list	Email: bob@therapist.com Password: 123456	The expanded worry detail should show all solutions with arrows to cycle through them.	Application functioned as expected
21	View patient's mood target on a graph	1. Click "Log in" 2. Enter details 3. Click "Log in" 4. Click "View" on John Tests card 5. Click on "Statistics" 6. Click on "Show mood target" on the ratings over time graph	Email: bob@therapist.com Password: 123456	The mood target should be displayed as a line on top of the graph	Application functioned as expected
22	View patient's	1. Click "Log in" 2. Enter details 3. Click "Log in"	Email: bob@therapist.com Password: 123456	The worry target should be displayed as a	Application functioned as expected

	worry target on a graph	4. Click "View" on John Tests card 5. Click on "Statistics" 6. Click on "Show worry target" on the ratings over time graph		line on top of the graph	
--	----------------------------	---	--	-----------------------------	--