A cross-platform mobile Mood Tracker/Diary App

David Ryan
17417364

Final Year Project – 2021
B.Sc. Multimedia, Mobile & Web Development

Department of Computer Science
Maynooth University
Maynooth, Co. Kildare
Ireland

A thesis submitted in partial fulfilment of the requirements for the B.Sc. Multimedia, Mobile & Web Development.

Supervisor: *Dr. John Keating*

Contents

Declaration

I hereby certify that this material, which I now submit for assessment on the program of study as part of B.Sc. Multimedia, Mobile & Web Development qualification, is *entirely* my own work and has not been taken from the work of others - save and to the extent that such work has been cited and acknowledged within the text of my work.

I hereby acknowledge and accept that this thesis may be distributed to future final year students, as an example of the standard expected of final year projects.

Signed: Date:

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. John Keating, for assigning this project to me and guiding me throughout the process. I have been able to build a deeper understanding of good User Interface/User Experience (UI/UX) practices and gained experience in cross-platform mobile application development, both fields I have an interest in, as a direct result of his contributions. He has kept me on the right track through both the design and development stages, and his input was of benefit to the app produced.

I would also like to thank each of the testers who took part in the UI/UX tests, especially Dr. Ralf Bierig who asked his students to take part. They each contributed both their insights and time, in an already stressful year. I greatly appreciate their efforts and hope they found the experience beneficial to their own learning.

Abstract

Mood tracking is a technique in positive psychology that includes the tracking, recording, and analysis of a person's mood. Psychotherapists usually recommend it to patients who want to improve their mental health. It can be used personally or to support a counselling programme with a professional psychotherapist. This project addresses the facilitation of this process, for both previously mentioned contexts, in a manner that would allow a user to easily begin practicing this technique by means of the design and development of a cross-platform mobile application. Positive mood can enhance cardiovascular, hormonal and immune functions, promote healthy behaviours such as better sleep and exercise, and lead to more open-minded thinking and effective problem solving. The aim of this project is to build an application that improves upon existing apps that assist users with this practice, and allows users to maximize the benefits of taking a proactive approach to self-management of mood.

List of Figures

Chapter one: Introduction

1.1 Topic addressed in this project

This project is concerned with the development of a Mood Tracker/Diary mobile application, using Dart/Flutter as a cross-platform development toolkit. This project involves the development of the front-end and the design of an appropriate user interface for the given context. The app will communicate with a database via an API that has been created in a past student's final year project.

Mood tracking is a technique in positive psychology that includes the tracking, recording, and analysis of a person's mood. Psychotherapists usually recommend it to patients who want to improve their mental health. It can be used personally or to support a counselling programme with a professional psychotherapist. This project addresses the facilitation of this process, for both previously mentioned contexts, in a manner that would allow a user to easily begin practicing this technique.

1.2 Motivation

The World Health Organization (WHO) stress the mental element of health in their definition of health as defined in their constitution: "Health is a state of complete physical, mental and social well-being and not merely the absence of disease or infirmity". They go on to outline the magnitude of people suffering from mental disorders, stating: "Today, about 450 million people suffer from a mental or behavioural disorder" (World Health Organization, 2003).

The COVID-19 pandemic and the measures in place to combat the virus have created an environment of high-stress and anxiety, much to the detriment of people's mental health, especially those living with a mood disorder. Current psychological distress levels are elevated in individuals with mood disorder due to financial issues, adverse lifestyle changes, and worry for health and wellbeing of family/loved ones (Van Rheenen, Tamsyn E et al., 2020). Similarly, those who may have not struggled with their own mood in the past are now finding themselves experiencing negative symptoms during this pandemic. 4 in 10 adults in the US have reported symptoms of anxiety or depression during the pandemic – this is up from a figure of 1 in 10 adults reporting these symptoms between January to June of 2019 (Panchal, Nirmita et al., 2021). It is clear, that in these unstable times, people are more likely to find themselves in a state of unstable mental health with their own mood following suit. The Adult Psychiatric Morbidity Survey (APMS) 2007 (n = 7403), found a 13.9% population rate of mood instability, making it out to be an important part of psychopathology and that a better understanding of its nature, origins and correlations would be valuable (Broome, M R et al., 2015).

Mood tracking is a technique used to gain a better understanding of one's own mood and can enable pattern recognition, opening the possibility of proactive action rather than reactive action regarding self-management of emotional well-being. Positive mood can enhance cardiovascular,

hormonal and immune functions, promote healthy behaviours such as better sleep and exercise, and lead to more open-minded thinking and effective problem solving (Caldeira, Clara et al., 2018). A format for a person to carry out this practice in positive psychology, be it on their own initiative or following the advice of a professional psychotherapist, could prove beneficial for their overall health. Keeping a documented, dated log of one's mood would increase the accuracy of analysis and allow for better reflection, making mood tracking a more rewarding experience for the individual. Psychotherapists would also benefit from an accurate log of their patients' as it overcomes the issue of a patient's difficulty recalling how they felt on a previous date.

1.3 Problem statement

A model for capturing mood had been created in the previous work that set the foundation for this project through the development of a mood tracking API. This allowed for focus on the design issues presented by the requirement of a user interface built for capturing the data set out by the API, and for allowing each of the back-end's functionalities to be available in the front-end of an application in the context of a mobile device.

A user had to be identified so that their needs and wants could be fulfilled in the design. Naturally, different users expect different things, so what is to be incorporated and what is to be ignored must be realised from an early stage. A mood tracking app could have potential uses for both adults and children; however, these groups have a very contrasting set of user requirements. The process of designing an application to be used by children poses many new challenges. Furthermore, there is not a single user group for children. What an 8-year-old wants out of an app is vastly different to the wants of a 14-year-old.

Colour is a huge aspect of designing user interfaces. The app proposed in this project could benefit from each mood value having a corresponding colour, for the purposes of quicker recognition by the user and building their relationship with each of the mood values and how they are presented throughout the interface. This raises an issue whether a potential user's predetermined relationship with a colour conflicts with what mood the design correlates that colour to. Colour-blindness and other visual impairments must be considered to ensure an app that is accessible and not exclusionary from a design perspective.

Accessibility intersects both User Interface (UI) and User Experience (UX). UI refers to the presentation and interactivity while UX deals with the relationship the user builds with the app through using it. The application must be a pleasure to use and not pose any unnecessary cognitive load. The mood tracker must perform as a user would expect a mood tracker to perform.

Moving from design to technological issues, cross-platform capability requires testing for both iOS and Android platforms, as well as consideration for the many devices and screens sizes that may be displaying the app. Technological issues also include that the front-end correctly consumes, and feeds, the API. There are some issues in the API that will have to be repaired, such as broken endpoints, that likely arose from not being maintained after release.

1.4 Approach

For an effective design to be produced, a well-defined plan consisting of a series of consecutive stages, with each building upon the strengths and weaknesses of the prior, is required. This begins, as most projects do, with extensive research into existing solutions to the problem. Existing apps, such as Daylio and eMoods, were reviewed and it was considered how this project could fit in among that environment, ideally providing something the others lack in. A 2018 study by Caldeira, Clara et al., which analysed the features of many mood tracking apps was invaluable to this research. The study found that while these apps offer opportunities to collect data and reflect on it through data visualisation, user experience could be improved by extending support for the preparation and action stages of mood tracking. Preparation would involve informing a user on mood tracking before they begin the process. Action means empowering the user to take a course of action after the reflection stage, by providing advice or guides. The aim of this project is to improve upon other mood tracking apps and allow users to maximise the benefits of mood tracking by supporting these stages.

The study also found that these apps were used by both the general population and people with mental conditions. The general population tended to use the apps to learn about their mood patterns and cope with stress, while users with existing mental conditions also monitored their symptoms and medication. They used the tracked data personally, but also would communicate the data with their psychotherapist. This information would assist in deciding on features to be implemented but also in more accurately creating user stories and user personas.

The design process of this project is as important as the development stage. Both are iterative processes, and while changes were expected to be made to the UI/UX dimensions of the app during the development stage, it was important to complete each stage of the UI/UX process before any development began. User research, including the above-mentioned study, formed the theoretical foundations for the creation of user personas, fictional characters that represent a selection of the app's audience that includes their goals and pains. Two user personas were created in this stage, an adult who is more of a casual user, and a child that attends a counselling service. The user personas led to the creation of user stories told from the personas' perspectives. These stories function to realise the required features to satisfy user requirements, and typically follow the format of "As a [type of user], I want [some goal] so that [some reason]". At this stage, it was realized that this children user group would need an entirely separate app to fulfil their needs, so it was decided to focus the design on the adult age range.

The next stages included brainstorming interface ideas, sketching low-fidelity wireframes on paper and then drawing on the strengths of all the wireframes to progress to a high-fidelity wireframe. The free web application, Figma, was utilised to design high-fidelity, interactive wireframes which gave a real sense of what the developed app could look like. Here accessibility issues could be overcome by using plug-ins to ensure that the interface works for those with visual impairments. The main priority in this regard is that there is enough contrast between text and background to ensure legibility. A contrast checker ensured contrast throughout the design is to

the standard set by the Web Content Accessibility Guidelines (W3C, 2008).

From a technological perspective, Flutter, developed by Google and built on the Dart language, was chosen as a framework for cross-platform mobile application development. Flutter is unique and extremely powerful in that one codebase is all that is needed to build for both iOS and Android. The API from the past project was developed as a Python web application using Flask as a framework.

1.5 Metrics

Following agile development practices, testing will be carried out each time a feature is created or changed. It is important to test that each feature functions as desired at the time of coding so that the development can progress in an incremental fashion.

Regarding UI/UX, the application will be evaluated by a process of user testing. Test data will be of a qualitative nature, gathered by testers attempting to complete a set of tasks in their own time. They will be given the opportunity by means of a survey and a dedicated Microsoft Teams channel to voice their opinions, raise their frustrations, discuss what they liked and did not like, and identify any bugs they encountered during testing.

1.6 Project

Some of the significant achievements from this project were:

1. Seeing through the development of my first mobile application using a new framework, from the stages of printing 'Hello, World!' to rolling out to the Google Play store.
2. Gaining some experience in back-end technologies by repairing features of the API that had broken, and by refactoring the web-app to be deployed to Heroku.
3. Seeing the impact of UI/UX driven design and discovering how ideas sketched on paper can incrementally develop into a fully-fledged application.
4. **Learning a lot about positive psychology and coming to understand more about mood disorders and what can help those living with them.**

1.7 Summary

Having introduced the topic addressed in this project, outlining the motivations for the development of a mood tracking app, and discussing the approach to be taken in solving the problem of this task; the following sections of this report will further expand on the research carried out and expand on the analysis of the problem before considering the solution and outlining the testing process.

Chapter two: Technical Background

2.1 Topic material

To build an effective tracker of mood, it is important to have a solid grasp on what is meant by mood. A knowledge of this concept and the effect of practicing tracking mood should be a foundation for any decisions taken throughout the design and development process. A model for capturing mood had already been built, and the aim of this project was to build on that work. However, further research was conducted to develop an understanding of the decisions made that led the model to the form that it took. The model drew inspiration from Plutchik's wheel of emotions and Dr. Gloria Willcox' feeling wheel. Plutchik's wheel of emotions is used as a tool to identify and summarize emotions to their primary classes (Mohsin et al., 2019). It sets out primary, core emotions; how these emotions can combine to form new, secondary emotions; and how an emotion changes as it intensifies. Willcox' feeling wheel has proven useful in assisting clients to learn how to identify and express feelings (Willcox, 1982). The inner labels of this circle – namely happy, sad, surprised, disgusted, angry, fearful, bad – are the input values for the mood parameter as set out by the previously designed model for mood.

Contrastingly, it is proposed that it is possible to make a relatively consistent distinction between mood and emotion based on their intensity and duration (Totterdell et al., 1997). Emotions are said to be brief but intense experiences activated by situational factors, while moods are said to be prolonged but less intense experiences. In 1985, Watson & Tellegen developed a two-factor model for mood that employs a two-dimensional spectrum which splits moods into four categories (Desmet, 2015). This would suggest a much different approach to capturing mood. However, through the process of creating user stories as part of this project, it was discovered that a user would be likely to quickly use the mood tracker during a moment of intense emotional reaction. This falls more in line with the definition for emotion which led to the continuation of the use of Willcox' core emotions.

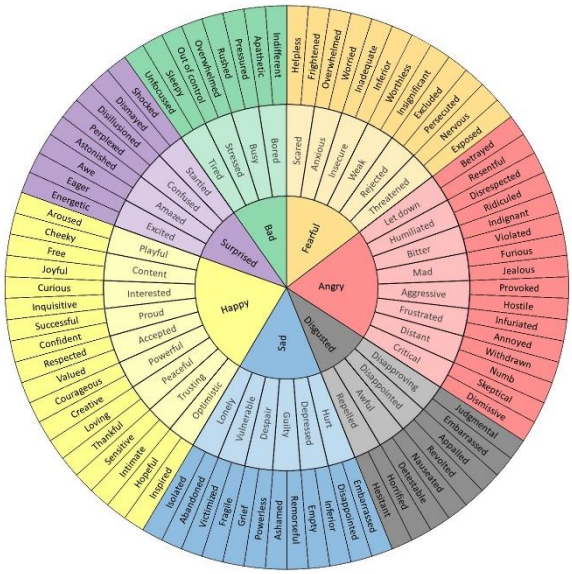


Figure 1 - Feeling wheel set out by Willcox

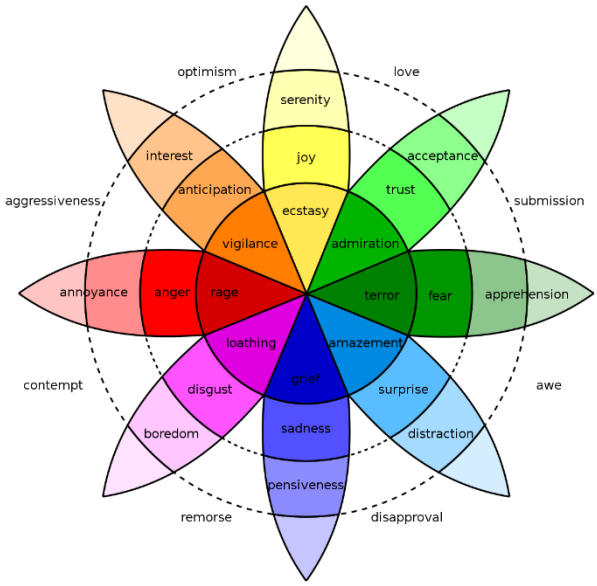


Figure 3 - Pluctchik's Feeling Wheel

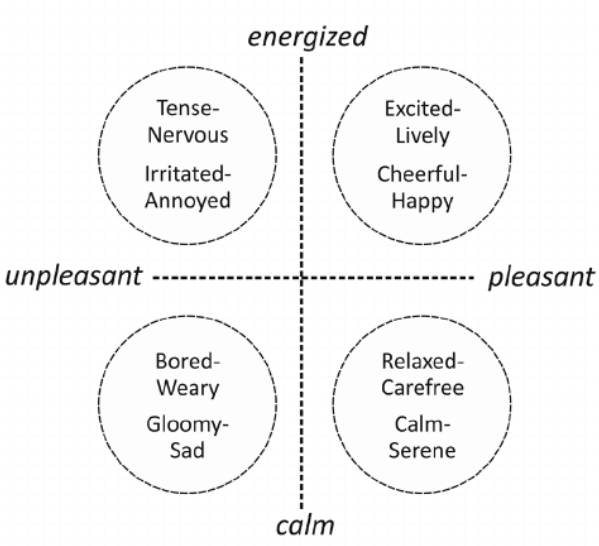


Figure 2 - Watson & Tellegen's two-dimensional model for mood

When designing, choosing colours is an important discussion to have as it can spur drastically evocative responses in users. The emotional response to colour is well enough understood that companies such as McDonalds have carefully chosen to use reds and yellows in their branding, colours that are known to increase appetite (Satcharoen, 2017). Choosing colours becomes an even more delicate process when the data being collected should be an accurate representation of the user's mood at that time. It is important that the colour a mood is represented by is an accurate portrayal of that mood. The colour should not conflict with the user's mood, having the potential to influence how they are feeling and prohibiting accurate tracking. The UX of the app would be negatively impacted if a user feeling extreme anger was to be presented with an interface full of happy colours.

For this reason, each mood-representing colour should be rooted in colour psychology. Conveniently, Plutchik's wheel can be used by designers to act as a "colour palette" for emotional design (Interaction Design Foundation, 2021), which led to the colours set out by Plutchik to be used throughout this project's design in order to represent their corresponding mood. Similarly, background colours require a similar amount of caution to be taken. A white background is standard to ensure high contrast between it and any text. This leaves only a primary colour for UI components such as top bars and button backgrounds. Research into this outlined green to be the best contender. Green is considered an emotionally calming colour (Kurt et al., 2014). Greens are found as relaxing as they are reminiscent of those found in nature (Eiseman, 2006).

Following the research into colour, reading was carried out into accessibility as there is overlap in these two topics. Colour blindness affects approximately 1 in 12 men and 1 in 200 women (Colour Blind Awareness, 2021). Additionally, at least 2.2 billion people in the world have a near or

distance vision impairment (World Health Organization, 2021). This makes clear the importance of designing with visual impairments in mind. For an app to be user friendly, it must be friendly towards all users, not a select few.

Furthermore, an investigation into existing software with a similar aim to this project found several popular mood tracking applications available on both the App Store and Google Play store, such as Daylio and eMoods. An already mentioned analysis of mobile apps for mood tracking by Clara Caldeira et al. provided in-depth investigation onto what is provided by these apps, but also, more importantly, where these apps were lacking in terms of supporting their users' mood tracking journey. This study also identified four stages of the mood tracking process:

1. ***Preparation*** **–** Planning that occurs before users start collecting personal data.
2. ***Collection*** **–** The recording of users' data.
3. ***Reflection*** **–** Making sense of and learning from personal data.
4. ***Action*** **–** When users act based on the insights gained through reflection.

A more practical approach was also taken as part of this research by downloading mood tracking apps on both mobile platforms to gain hands-on experience.

2.2 Technical material

The backbone of the technical material used for this project was the Flutter documentation. From setting up the environment to sample code for specific widgets, the documentation for Flutter provided by Google was frequently referred to for learning the framework and Dart, the language it is built upon.

The Android developer documentation assisted in getting the application built and deployed to Android devices. It was with the knowledge found here that a build could be uploaded to the Google Play store to be distributed for internal testing.

Several packages for Flutter produced by the community were utilised for this application. Each had its own documentation on pub.dev, the package manager for Dart. Figure 4 outlines this project's dependencies.

```
dependencies:
  flutter:
    sdk: flutter
  cupertino_icons: ^1.0.0
  feather_icons_flutter: ^4.7.4
  google_fonts: ^2.0.0
  http: ^0.13.0
  async: ^2.4.2
  fl_chart: ^0.20.1
  flutter_calendar_carousel: ^1.5.3
  path_provider: ^2.0.1
  csv: ^4.1.0
  flutter_email_sender: ^5.0.0
  url_launcher: ^6.0.2
  flutter_launcher_icons: ^0.9.0
  flutter_svg: ^0.21.0-nullsafety.0
```

Figure 4 - Utilised Packages

Like most other projects in this field, StackOverflow was a great resource as problems arose throughout development. The discussions had on this forum consisting of several potential solutions to frequently occurring issues provided a wealth of information and a place to go when progress hit a brick wall.

Regarding the API and back-end technologies serving this project, the documentation for MongoDB and Flask assisted in altering the server code to update deprecated code and repair broken endpoints. Additionally, the Heroku documentation allowed me to refactor the API so that it was deployable as a web application.

2.3 Summary

Research in the areas of both topical and technical material was essential to develop a solid backbone regarding the practical work that was to be carried out. Having gained a knowledge on mood and past solutions to computationally modelling it, an understanding could be found for the model for mood that the API was built upon. The following research into design issues including colour and contrast, paired with research into the technologies that would be tools used to build the software, are what directed the approach taken in designing and developing a cross-platform mood tracking app. The research in these two areas formed symbiotic constraints that were to be worked within, in that technological limitations spurred design constraints and vice versa.

Chapter three: The Problem

3.1 Problem analysis

A UI/UX problem begins with the user. Designing with a lacking understanding of the user requirements is equivalent to shooting in the dark. User research should be the fuel to an analysis of a problem such as the one being dealt with in this project. Following the research mentioned in previous sections, the techniques deployed to ensure user-centred design and develop an understanding of user requirements included user stories and user personas.

A good starting place to quickly get an idea of potential users is to develop user stories. These are short statements that aptly identify who the user is, what they need and why they need it. They are a practical tool that quickly get to the point. User stories follow the format of "As a [type of user], I want [some goal] so that [some reason]". There were two user stories developed as part of the problem analysis:

1. *As someone with a busy work life and an interest in my mental health, I want to be able to quickly see trends in my moods and their causes, so that I can develop a greater understanding of my own psychology and be more in control of my moods.*
2. *As a child who attends a counselling service, I want something that can help me understand my mood and how to put it into words, so that I can better explain how I am feeling.*

User personas build upon these user stories, fleshing out the identity of the user by realising their goals and frustrations. The creation of user personas helps with problem analysis as it humanizes the idea of the end user and allows the designer to ask themselves "what would [name] want?". The user personas created were based on the above user stories and can be seen included in Appendix 4.

Use cases were also created to discover how a user would trigger activities within the app, and how the app should respond as a result. Use cases set out a clear sequence of events that should occur that allow a user to complete a task. The use cases used to realise how a user would interact with this app are as follows:

1. **Activity:** Enter current mood
    a. The user logs in to the app.
    b. They use a dial to input how they are feeling.
    c. The user clicks a button to add the entry.
    d. The user chooses to not fill in information yet.
    e. *Alternative:* The user chooses to fill in all relevant data.
    f. The user saves the entry for mood.

2. **Activity:** Update previous mood
    a. The user logs in to the app.
    b. The user selects a recent mood entry from the Home Screen.
    c. The user updates the information in the relevant field.
    d. The user saves the updated data.

3. **Activity:** See log of previous moods
    a. The user logs in to the app.
    b. The user navigates to a Calendar Screen.
    c. The user finds the date they are looking for and the mood they felt that day.

4. **Activity:** Get statistics based on mood entries
    a. The user logs in to the app.
    b. The user navigates to a Statistics Screen.
    c. The user reviews data visualisation generated from their inputs.

5. **Activity:** Export mood entries
    a. The user logs in to the app.
    b. The user navigates to a Settings Screen.
    c. The user triggers the export function.
    d. The user inputs a target email address.
    e. Their mood entries are converted to CSV and emailed to target email.

Using the produced user stories, user personas and use cases as tools to attain a clear idea of the problem and the user requirements that would be needed to solve it, the following list of features to satisfy user requirements were set out in compliance with the four stages of mood tracking:

1. [Preparation Stage] A set of screens informing the user on mood tracking, its benefits, and how to use the app.
2. [Collection Stage] The ability to log data relevant to the user's mood that does not take up too much time.
3. [Reflection Stage] Visual data representation that enables insights into trends in the user's mood.
4. [Reflection Stage] A calendar that allows a user to see how they felt on previous dates to overcome possible difficulty with recall.
5. [Action Stage] Available resources that provide advice or guide the user towards acting on their mood, such as information on helplines.
6. [Action Stage] The ability to send their history of tracked data to themselves or to a professional psychotherapist.

3.2 Project UML documentation

Figure 5 is a UML class diagram that outlines the classes that were created for this application. A user object is instantiated using the User class' factory method when the login method of the ApiService is called. The ApiService parses inputted data to JSON and handles calls to the API and the responses it receives. The other ApiService methods function in a similar way, returning the relevant types or error messages. The User object will make a call to the getEntry() method in the ApiService, which returns an Entry object from the returned JSON. The Entry class is also used to return a List of type Entry when a call to the ApiService's getEntryList() is called.

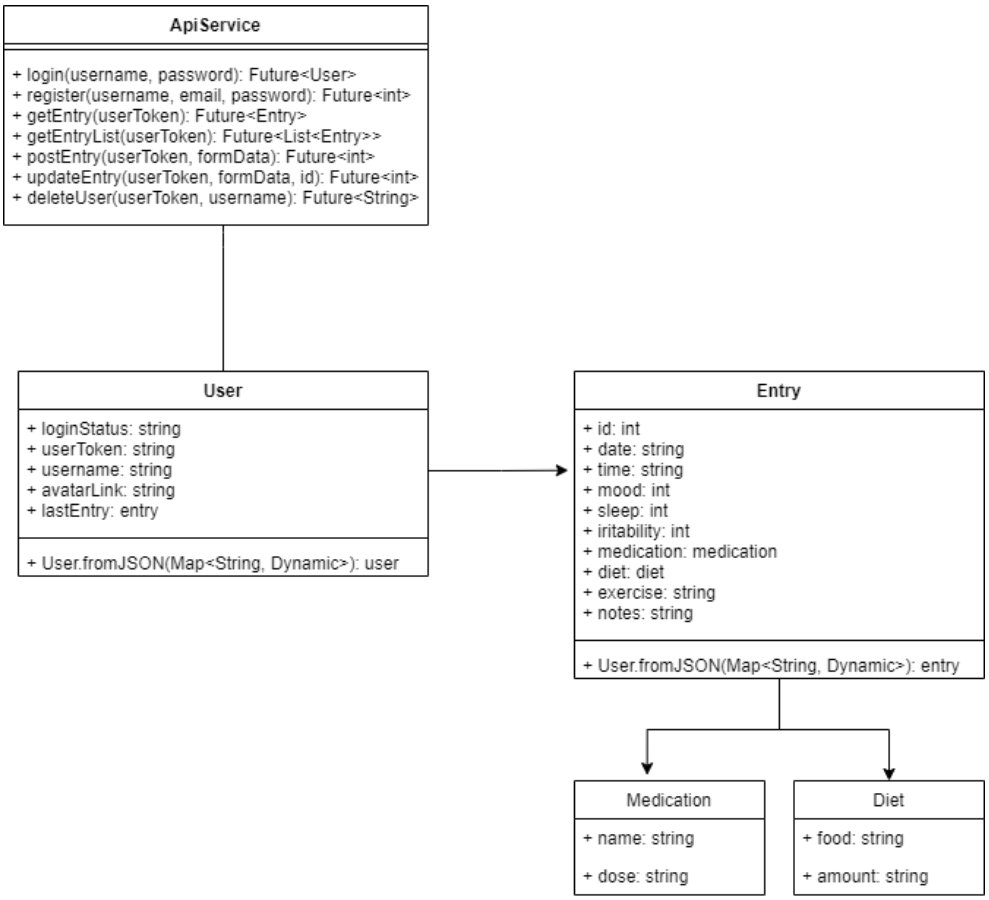


Figure 5 - UML Class diagram for Mood Tracking App

Figure 6 is a UML sequence diagram showing a typical user flow upon opening the application for the first time. The user begins by registering via the interface. This invokes a call to the ApiService, which then encodes the data to a JSON format and sends a HTTP POST request to the API. In this situation, the validation is successful and the new user is created. A successful response is sent back as far as the UI which automatically logs the user in without the need for the user to input the same data again. Following a successful login, the UI renders the Home screen where the user presses the 'Add Entry' button. A form is built by the UI and after being submitted by the user and validated, the data is sent to the ApiService which again packs it up into JSON and issues a post request to the API. The API adds the data to the database and a success code is sent back. When the UI receives the successful response, it renders a component to the screen, indicating this to the user.

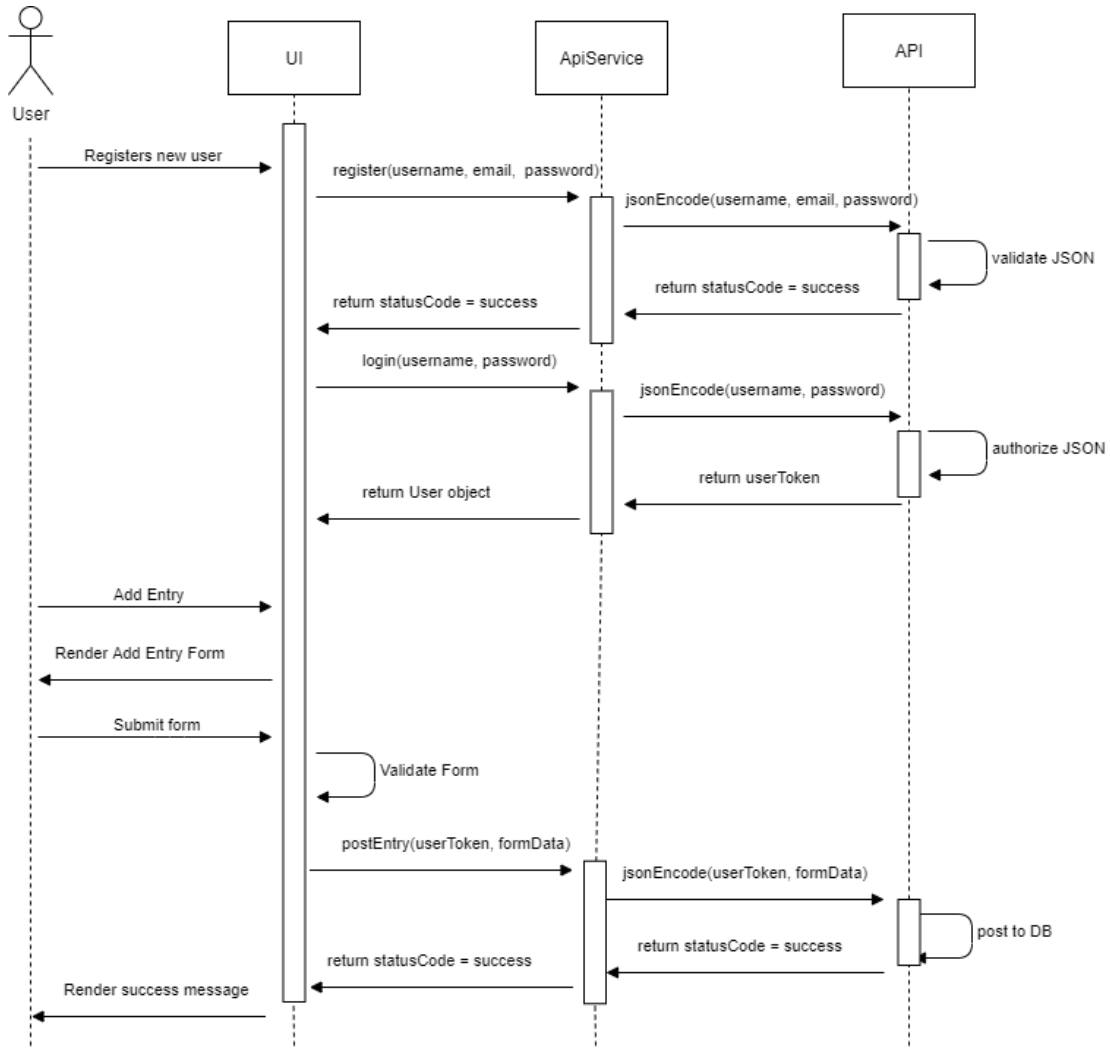


Figure 6 - UML Sequence Diagram for Mood Tracking App

UML Sequence diagrams for each interaction are included in Appendix 2.

3.3 Summary

The problem at hand has been analysed through a process of user research, realising user requirements through user stories, user personas and use cases which led to the formation of a list of features that would fulfil the requirements set out by a mood tracking app. The four stages of mood tracking as discovered in chapter 2.1 of this report were used as a tool in forming this list of features and would ensure that the project stayed on course in pursuing the aim of improving upon existing mood tracking apps in the preparation and action stages. The UML documentation produced elaborates on the structure of the project and each user interaction in a more technical manner.

Chapter four: The Solution

4.1 Introduction

This chapter identifies, discusses and justifies the decisions made in the production of a solution that satisfies the problem as set out and analysed in the previous chapter. The discussion is organised in a top-down fashion, beginning by outlining the solution on an architectural level, progressing on to high-level and low-level discussion respectively and, finally, a more in-depth look at the solution's implementation. The purpose of this approach to the discussion of the solution is to establish an incremental understanding of the structure and content of the solution, only getting deeper into detail after a higher level is discussed.

4.2 Architectural Level

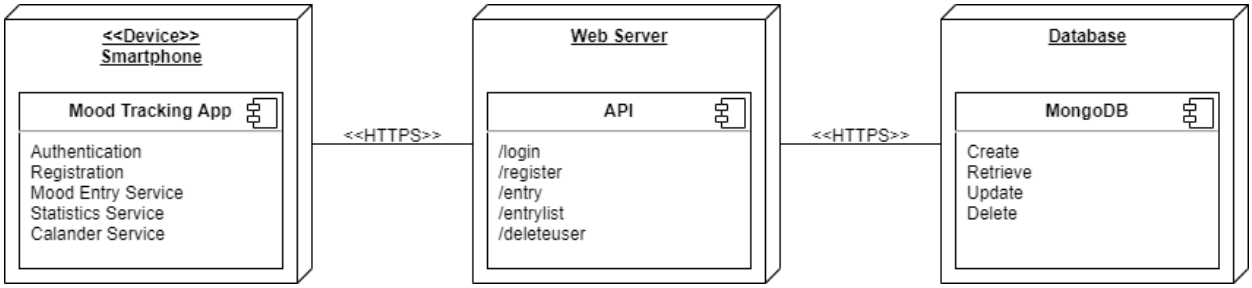


Figure 7 - Architectural Design

Figure 7 is a deployment diagram for this project, outlining how the high-level components – the app, the API, and the database – interact with each other to deliver the available features. The app running on the user's mobile device reaches out to the API, running on a web server, which in turn uses HTTP verbs to interact with the database.

The architecture of the API was developed using the Flask micro-web framework for Python, along with a MongoDB database for storage as part of the previous project. This project focused on the development of an accompanying front-end, so that the functionality of the API was available to a much broader audience.

The front-end was developed using Flutter as an application framework which requires the Dart language to be used throughout the codebase. Although Dart is a less popular language in comparison to the likes of Javascript or C#, the admittedly small learning curve is rewarded by a single codebase being compatible for builds on Android and iOS with close to native performance. React Native, which utilises Javascript, offers similar advantages, however the increasingly popular Flutter has the competitive edge in areas such as performance and development speed.

As seen in Figure 7, the design pattern used in this project was that of Model View Controller (MVC). This allows for the separation of logic into interconnected elements. The View, in this case the UI as rendered by Flutter, displays representations of data as it changes and available

functionality. The Model, in this case the MongoDB database, is the application's data storage. MongoDB is a document database, which means it stores data in JSON-like documents. This allowed for easy handling and minimal parsing for the controller. The Controller, in this case the API developed in Python and using the Flask framework, accepts input and converts it to commands for the model or view.

Determining MVC as a design pattern allowed for nicely segmented components to be developed. Each piece of code had a clearly defined role within the application architecture. The model manages data and takes input from the controller. The view renders presentation of the model. The controller responds to user input and performs interactions on the model.

4.3 High Level

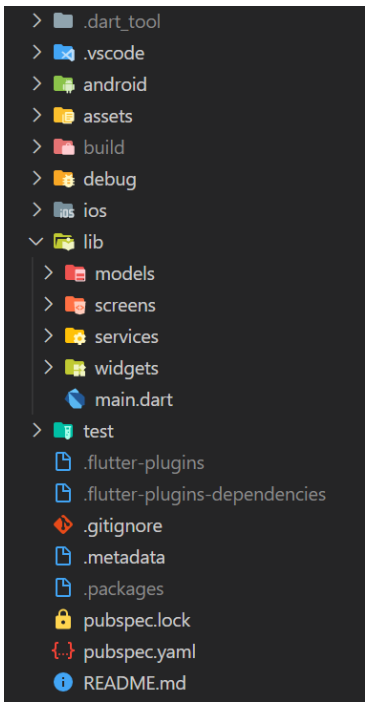


Figure 8 - Project Structure

Figure 8 shows the project structure. Most of the contents are created when opening a new project in Flutter for the purpose of improving development experience and allowing for cross-platform code reuse. During development, Flutter apps run in a virtual machine that offers stateful hot reload of changes without needing a full recompile. For release, Flutter apps are compiled directly to machine code.

The lib folder is the area of most interest when discussing this project in a high-level manner. The models folder sets out blueprints for the core classes, user and entry, which can be seen in Figure 5. These files declare a structure that ensure compatibility of data between the architectural components of the application. For example, the structure of a user object within the app must be transferrable to the structure of a user object in the database. The same is to be said for each of the entry objects.

Flutter emphasizes widgets as a unit of composition. Widgets are the building blocks of a Flutter app's user interface, and each widget is an immutable declaration of part of the user interface. Widgets are created in a nested fashion which lends itself to a hierarchical structure. This structure carries all the way up to the root widget, in this case 'main.dart'. The screens folder simply sets out parent widgets for each of the screens in the view layer. The parent widgets invoke calls to render child widgets, which are contained within the widgets folder. Flutter allows for the creation of Stateful and Stateless widgets where appropriate that can update as data changes.

The services folder makes available both core and secondary functionality. The primary service here is the ApiService that handles sending data via a specified HTTP verb to a certain endpoint of the API. Also found in the services folder are functionalities that allow for converting the user's list of entries to CSV and emailing the generated file, opening links in the user's web browser, and receiving a mood's corresponding name, colour and emote given an integer value.

'Pubspec.yaml' is a file required by DART to attain metadata that defines information such as required assets and dependencies. A list of this project's dependencies can be seen earlier in Figure

4. This file is written in the YAML language and also sets out SDK and build versions. ‘Pubspec.lock’ simply nails down concrete versions, where the other file allows for ranges.

High-level design work included the creation of sketched low-fidelity wireframes. These were created after the user research stage consisting of the user stories and personas generated during this project. These sketches can be seen included in Appendix 4. The sketches provided a way of brainstorming ideas and a reference point for the lower-level creation of high-fidelity wireframes that get closer to the final design and showcase more detail.

As mentioned in chapter 2.3, the relation between design and technological constraints is often symbiotic. It is common for the choice of technology to influence the design decisions made as it would be counter-beneficial to spend a great deal of effort designing a solution that is not possible within the constraints of the chosen technology. Thankfully, Flutter does offer a distinguishable degree of creative freedom in this regard. For example, any RGB colours are permitted as well as any choice of font family. Although, as will be discussed in chapter 5.4, the Material Design widgets available through Flutter caused differences to exist between the final design and the final build of the app, the choice of technology created little to no design constraints.

4.4 Low Level

4.4.1 Low Level - Design

Low level design included carrying on from the sketch work to progress into high-fidelity, interactable wireframes to give a sense of what a functioning prototype would look and feel like. Figma, the online design tool, was used for this as it allows for remote access to working files and an extensive plug-in library for no cost.

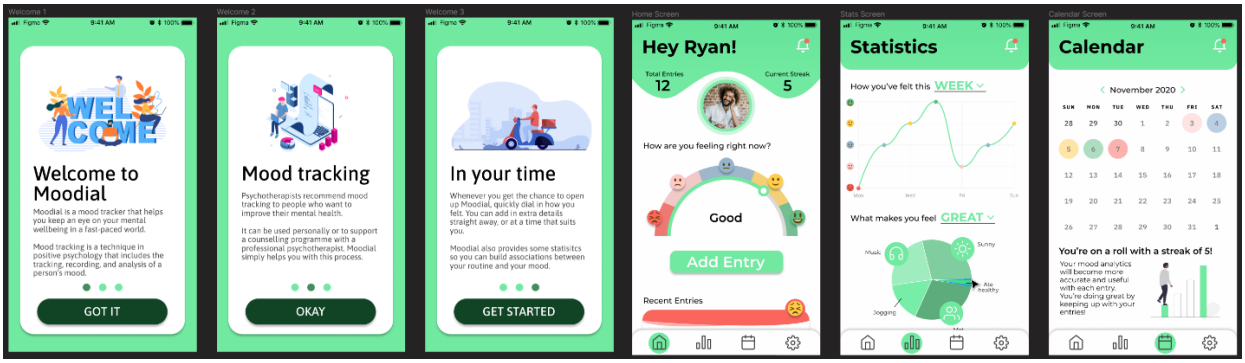


Figure 9 - High-Fidelity Prototype

Figure 9 shows a selection of some the screens designed within Figma. The product of this stage was an interactable view of the design, that enabled scrolling and navigation. This can be exceptionally effective when there are separate design and development teams as there is a greater understanding of the vision for the product.

4.4.2 Low Level - Development

```
@override
Widget build(BuildContext context) {
  return Stack(
    alignment: Alignment.center,
    children: [
      Container(
        width: 120,
        height: 120,
        decoration: BoxDecoration(
          color: Theme.of(context).primaryColor.withAlpha(100),
          shape: BoxShape.circle,
        ), // BoxDecoration
      ), // Container
      Container(
        width: 105,
        height: 105,
        decoration: BoxDecoration(
          color: Theme.of(context).primaryColor.withAlpha(150),
          shape: BoxShape.circle,
        ), // BoxDecoration
      ), // Container
      user.avatarLink == null
          ? CircleAvatar(
              radius: 45.0,
              backgroundImage: AssetImage('assets/images/avatar.png'),
            ) // CircleAvatar
          : CircleAvatar(
              radius: 45.0,
              backgroundImage: NetworkImage(
                user.avatarLink,
              ), // NetworkImage
            ), // CircleAvatar
    ],
  ); // Stack
```

Figure 10 - Build Method

Figure 10 is an example of the build method required by each component to be rendered to the screen and makes clear the nested nature of Flutter's widgets. The method receives the context of the parent widget which called the build method and returns a single widget containing any amount of nested child widgets. The given example is the widget for the user's avatar as shown in Home Screen design in Figure 9. A stack widget is the return value which contains children that align themselves on top of each other as you would expect of a stack. A ternary operator is utilised to render different widgets based on a Boolean statement. The ability to conditionally render is a powerful tool that was used several times throughout the source code of this project.

lib > models > user.dart > ...

```
1   import 'package:Moodial/models/entry.dart';
2
3   class User {
4     final String loginStatus;
5     final String userToken;
6     String username;
7     Entry lastEntry;
8     String avatarLink;
9
10    User({
11      this.loginStatus,
12      this.userToken,
13      this.username,
14      this.lastEntry,
15      this.avatarLink,
16    });
17
18    // ignore: empty_constructor_bodies
19    factory User.fromJSON(Map<String, dynamic> json) {
20      return User(
21        loginStatus: json['message'],
22        userToken: json['JWT'],
23      );
24    }
25  }
```

Figure 11 - User Class

Figure 11 is the User class that is instantiated upon logging in to the app and is an example of how classes are defined in the Dart language. As in most object-oriented languages, the class declares properties and methods. The properties for the user object as depicted in Figure 11 include the user's login status (used for conditional rendering as previously mentioned), userToken (used for authentication when making API calls) and other self-explanatory information.

The properties are followed by a constructor and a factory method, which takes in a map derived from the JSON response of an API call and returns a User object containing the response values.

4.5 Implementation

4.5.1 Implementation – Design

On first boot after downloading the app, a user is presented with introductory information shown the Preparation Screens included in Appendix 3, that describes mood tracking and how using this app fits into this practice. These screens were implemented to fulfil the Preparation stage of the mood tracking process. After this tutorial of sorts, they can register and login to the app.

The implementation of the Home Screen, also included in Appendix 3, allows the collection of data. There is some contrast with the Home Screen prototype design as depicted in Figure 9, mainly around the slider for dialling in a mood value. Here, a single emote relating to the current slider value is displayed stacked over the user's avatar. This contributes to the user experience by building a connotation between the user and the selected mood, strengthening the idea that this is *their* mood.

Below the slider is a display of the user's five most recent entries, in reverse chronological order. The functionality provided here is that the user can get a quick overview of their past entries. Tapping the card for a specific entry will display all the information associated with it and gives the user the opportunity to update any data. In this way, should the user be busy, they can quickly add a mood entry and come back at a later time to fill in more details. It was identified through the user research and problem analysis stages that mood tracking should be flexible to the user's lifestyle. This feature makes tracking mood a less obstructive practice with the aim that a user would be less likely to give up on it and would therefore have a more rewarding experience.

The implementation of the Statistics Screen is depicted in Appendix 3. The data visualisation provided here is based on the data inputted by the user and becomes available after seven mood entries have been made. The screen enables the user to reflect on how they have felt and gain insights into trends in their mood. Graphs making comparisons between tracked parameters give the user opportunities to identify associations between their actions and how they have felt. For example, a bar chart contrasting they amount of sleep they got against how irritable they felt, and a pie chart outlining their dietary choices that may have an impact on their mood.

The Calendar Screen, as shown in Appendix 3, also fulfils the Reflection stage of the mood tracking process. Here a user can identify what mood value they inputted on any date. The Calendar can handle multiple mood entries being made on the same day by stacking the emote icons. It can be difficult to remember exactly how one felt at a specific date and having access to that information can assist both users of the app and their clinician should they avail of one.

More secondary functionality is provided to the user via the Settings screen depicted in Appendix 3. A user can add a custom avatar by inputting a valid URL to an image hosted on the web. An entire history of the user's mood diary data can be easily converted to a CSV file format and then emailed. This feature opens the device's native mailing app with a draft email that contains the CSV file attached and some text explaining what is being sent. The user has the option to log out of the app, but also has the option to delete their user and the associated data from the database should they wish. The export feature along with the presented resources as shown in the Resources implementation seen in Appendix 3, fulfil the Action stage of the mood tracking process. These

features provide an ‘aftercare’ of sorts, in that the aim of the app is to further help users self-manage their mood past the acts of simply logging data and having it visualised.

4.5.2 Implementation – Development

To communicate with the API, a service was created so that API calls could be easily accessed throughout the source code via method calls. To implement this, an ApiService class was created that contained a method for each of the usable HTTP verbs for each endpoint as set out by the API.

lib > services > api.dart > ApiService

```dart
import 'dart:convert';
import 'package:Moodial/models/entry.dart';

import '../models/user.dart';
import 'package:http/http.dart' as http;

class URLS {
  static const String BASE_URL = 'https://moodial-server.herokuapp.com/';
}

class ApiService {
  static Future<User> login(username, password) async {
    final response = await http.post(
      Uri.parse('${URLS.BASE_URL}/login'),
      headers: <String, String>{
        'Content-Type': 'application/json; charset=UTF-8',
      },
      body: jsonEncode(<String, String>{
        'username': username,
        'password': password,
      }),
    );
    if (response.statusCode == 200) {
      //Successful log in
      return User.fromJSON(jsonDecode(response.body));
    } else if (response.statusCode == 401) {
      //Invalid username/password
      return User.fromJSON(jsonDecode(response.body));
    } else {
      throw Exception('Failed to log in due to server error.');
    }
  }
```

Figure 12 - ApiService

Figure 12 shows the implementation of the ApiService class and includes the login() method. This asynchronous method is called when the login form is submitted by a user and takes the field values for input. This example calls the post method as defined by Dart’s http package. The call to the post method declares the URI, and request headers and body, which are encoded into JSON so that they will be accepted by the API. The ‘await’ keyword ensures the request is completed before running any more code. After a successful response, a User object is returned by decoding the JSON response body. Each of the other methods follow a very similar process.

4.6 Summary

This chapter took a multi-level approach to discussing the solution that formed the practical element of this project. The solution was produced in response to the problem established in chapter 3 and by finding its foundation in the research carried out in chapter 2. To verify the solution, it must be evaluated through a process of testing that will be discussed in the next chapter.

Chapter five: Evaluation

5.1 Software Design Verification

This project followed Test-Driven Development (TTD) to verify the software design as the development process progressed. Initial testing was carried out on the API through the Insomnia client which allowed for the identification of correctly functioning endpoints and endpoints that required repair. Using Insomnia as a tool, HTTP requests could be quickly sent to the server and the response could be examined – an example of which can be seen in Appendix 4.

The sequence diagrams, such as that seen in Figure 6, were referenced to verify that each interaction a user could have with the app was available and followed the predetermined sequence of events as set out in the design. Sticking to agile development practices, as code was written in sprints, the code was subsequently manually tested so that each function was verified before progressing to the next sprint. This allowed for the application to be developed incrementally with confidence that what has been developed, has also been verified.

Manual tests were carried out during development using an Android emulator running on a Windows machine. Through Android Studio's AVD manager, the app could be tested on several android devices, covering a range of screen sizes. This was more complicated for iOS, as it is only possible to run an iOS emulator from a MacOS machine. However, after development but before building the app, an iOS emulator was accessed through a MacOS virtual machine hosted on the web - enabling for some verification for iOS devices.

5.2 Software Verification

5.2.1 Test Approach

The test approach taken as part of the software verification of this project was that of user testing. Given that the approach to this UI/UX project was of a very user-centred nature, it was logical to utilise user testing as a testing method. As such, the focus group was quite small. A group of 13 UI/UX/IxD students volunteered to join the tests. The data collected was of a mostly qualitative nature and was gathered by means of a survey to be completed after attempting a set number of tasks that would test each function of the application.

A dedicated Microsoft Teams channel was created for the purpose of the test where all the relevant information was posted to ensure that each of the testers were well informed. Sub-channels were created for the purposes of announcements, bug reports, tasks to be completed by testers, and a channel for general chat or introductions among the testers.

The test was limited to Android users due to the high cost involved in testing for iOS devices. A build version of the app was uploaded to the Google Play store and through Google's developer console, each of the testers could be whitelisted for internal testing. The testers could then download the app to their Android device and begin tests.

5.2.2 The Tests

There were six specific tasks set out to be attempted by the testers, each task testing a specific function of the application. The tasks were created by referencing the use cases as set out in chapter 3.1 to confirm the implementation of these key activities. The tasks were made available through the Teams channel and each included a title, description and small set of instructions. These instructions were to give some direction to the tester but were left vague enough so that it would become apparent if the app is confusing to use.

The tasks to be completed were as follows:

1. Log your mood by adding an entry.
2. Update your previous entry with more information.
3. Find previous mood via Calendar.
4. Change from the default avatar.
5. Get analysis on your entries (Available after 7 entries).
6. Email your entries to yourself in CSV format.

5.3.3 Test Results

The full test results can be seen included in Appendix 4.

5.3.4 Interpretation of Results

Interpretating the results as seen in Appendix 4 suggests that the functionalities a user would expect in a mood tracker are present, and that the activities set out by the use cases have been implemented and are functioning. There were no issues interacting with the API to perform CRUD operations on the database, indicating that core functionalities were operating as intended. However, rather than adding new features, future efforts should be spent on refining existing features by improving the UX and eliminating bugs. This is further demonstrated by the gathered quantitative data showing a majority of the testers finding the app to have good usability and be worth recommending to a peer, while the gathered qualitative data found that there was still much to improve on.

5.3 Validation/Measurements

5.3.1 Results

The results gathered from user testing realised both strengths and weakness in the app's design and development. The traits are most easily evident in the quantitative data. Examples of such strengths include 75% of users finding the app 'Easy' to use, with the remaining 25% finding it 'Very easy' to use. 100% of the testers would recommend the app to a peer. All the testers agreed they could find the features they needed with ease, while 87.5% agreed that the UI was not too complex.

The positive results noted above are contrasted by data such as 25% of testers agreeing that ‘there were a lot of bugs present’. The same number stated that they would be ‘somewhat unlikely’ to use the app. The room to improve is mostly found within the qualitative data, where testers could put into words their frustrations with the app. 3 of the 8 responses to the least favourite feature question pointed to the Avatar feature, with one citing that ‘it is far too complex for what it offers’. In the same question, one response stated that the entry form ‘is tedious to fill an entry’ and it is not ‘appealing enough for doing it every day’. Testers also noted that more options for certain fields would be helpful, such as an ‘indifferent’ or ‘neutral’ option for mood, or the ability to fill in more dietary information.

5.3.2 Explanation of Results

User testing is a powerful tool as it challenges the assumptions that were made by the designer and offers an insight into how the design is viewed from a range of perspectives. The results gathered as part of this project fulfilled this description. Each of the questions gathered data on an individual tester’s experience of using the app, indicating the broader picture of how the app’s UX would function with a more general userbase. The responses to these questions described some reasons to be optimistic about the project meeting its aims, while also discovering some concerns that would need to be addressed before another round of testing. The purpose of these results is to ensure that future work would be focused in the right areas and that decisions made from this point on would be further informed, allowing for improved user-centred design and the optimisation of UX.

5.3.3 Analysis of Results

Analysing the results paints a picture of an application that is in its early stages with potential to develop but many areas to improve upon. Rather than a focus on new or secondary features, the app’s core functionality should be improved on. Less important features, such as the Avatar, may risk weighing down the overall UX more than the availability of the functionality improves it. Core functionality should receive concentration going forward. The collection of mood data is central to the app and making this process pleasurable should be paramount. Some responses to the survey point to a refactoring of the model for mood as set out by the API being worthwhile for achieving this.

The purpose of this build version was to fulfil the requirements set by the use cases and to ensure the interactions, as seen in the UML sequence diagrams in chapter 3.2, were possible and the correct sequence of events was triggered and completed. This round of user testing confirmed that, although the UX could be much improved, this milestone was achieved.

The responses to why testers would recommend the app to a peer showcase the perceived benefits of a mood tracker, solidifying the purpose of an app such as this being developed. One tester stated, ‘Seems like an interesting app that would help people to stay well mentally, which I believe is important especially in the current times’.

5.4 Summary

This chapter discussed the methods used to evaluate the effectiveness of the solution as produced in this project that was discussed in chapter 4, and how it was verified whether the problem as identified in chapter 3 was solved. Test-driven development was used throughout the code sprints and a process of user testing was employed to evaluate the built app. It was discovered that the app had good usability without overcomplicating the UI, although refining the UX would much enhance the overall experience of using the app. It is worth noting that the requirements set out by use cases were satisfied by the solution and this was confirmed by the testers. Receiving contributions from many perspectives gave great insights into how the app could be further developed and improved upon.

Chapter six: Conclusion

6.1 Introduction

The aim of this project was to design and develop a cross-platform mobile application which facilitated the practice of mood tracking – a practice shown to have positive implications for a person's mental health. A solution was delivered that fulfilled this brief and underwent a process of user testing that verified its functionality and usability. User research and an investigation into the concept of mood and previous solutions for modelling and tracking mood provided a guide up to the delivery of this solution.

6.2 Contribution to the state-of-the-art

The application designed and developed for this project contributes to the existing ecosystem of mood tracking apps by providing more user support in the Preparation and Action stages of mood tracking, as identified by Caldeira et al. in their study. The front-end built upon the API opens previously completed work to a much broader audience, facilitating ease of use without the technical knowledge required to interact with an API without a dedicated user interface. The user-centred design adopted by this project with a focus on optimizing UI/UX, should fulfil the aim of this project as set out by the brief – enabling the practice of mood tracking for users in the contexts of both personal use and use as part of a program set out by a psychotherapist.

6.3 Results discussion

Although the testing group of 13 people was of a relatively small number, it was sufficient in gaining valuable qualitative data that gave an indication of how a more general group of users would react to the app. The qualitative nature of the response data does require more interpretation than quantitative data that could be sourced from more statistical testing methods. The insights gathered on where users found conflict in their use of the app steers the direction of future work to be taken to further improve the UI and UX. Speaking broadly, the results of the testing process indicated that the app is a mostly user-friendly experience that functions as a mood tracker should, but in its immature state, has much room to be developed and improved further.

The approach taken which handed much of the responsibility to the testers to carry out each of the tasks and then fill out the corresponding survey was threatened by a significant amount of those who joined the test not responding. In this way the amount of data gathered was limited and not as generalizable as it could have been should there have been more responses from different perspectives. A large factor in this was likely due to the remote nature of the tests because of social distancing guidelines and therefore there was a reliance on testers having the ability to set aside time for the tests.

6.4 Project Approach

This project was first approached with extensive research into existing solutions to the problem. Existing apps were reviewed, and it was considered how this project could fit in among that environment. The approach taken placed equal emphasis on the design and development stage, placing the user at the core of decision making. User research formed the theoretical foundations that allowed the development of this idea of the end user. Care was taken to ensure accessibility issues were overcome.

From a technological perspective, Flutter, developed by Google and built on the Dart language, was chosen as a framework for cross-platform mobile application development. The front-end that was developed built upon an existing API that was created as part of a previous project. Code was tested following each sprint as an agile approach was taken in this project. A build version of the app underwent a process of user testing to gain insights and identify bugs.

6.5 Future Work

The design of certain UI components in the functioning app have a different appearance to the prototype design. This is due to the Material Design widgets available as part of Flutter. As Flutter and Material Design are products by Google, there is great support for material widgets within the framework. Using these allows for an app to quickly get functional, visually appealing UI components, such as buttons. Future work would include building custom widgets to give the app a more unique look and feel. Unfortunately, this was not feasible within the time frame of this project – hence the use of material widgets.

The reflection stage of mood tracking could be improved by further developing the Statistics and Calendar screens of the app. Input from a professional psychotherapist could be insightful to provide more useful data visualization. It would also be good to develop the calendar so that all the information from past entries could be reflected on and/or updated, rather than just being able to see the mood values as is all the component offers as it stands. A notification system could also be included in future work, giving the user the ability to set reminders to log their mood so that they do not fall out of the habit.

The contributions by testers provided valuable insights into what could be done as part of future work to further improve the app. This showed the benefit of the user testing approach as ideas outside of the developer's mindset can come to fruition. The results as discussed in the previous chapter show that this app has the foundations for being a useful assistant in a user's self-management of mood and that further work with a concentrated focus on eliminating existing bugs and refining UI/UX is required to see it to that goal.

6.6 Summary

Given that this project was an introduction to mobile application development, the processes of user research, following UI/UX best practices, seeing through the development of an app through coding and user testing, were all immensely educational. Having not worked on a project of this scale and thoroughness before, it was greatly rewarding to work alongside testers and receive feedback on a project that received so much time and effort. Reflecting in this way also allows for realisation of how things could have been done differently that may have positively impacted the produced solution. If this project were to be done again, more time would have been invested into researching techniques to optimize UX and learning how to make typically tedious application components such as forms, more pleasurable to use. It is a positive outcome for the first version of an app to fulfil the brief and have the potential to have positive implications towards a wide audience's emotional wellbeing.

References

"Colour Blindness - Colour Blind Awareness". Colour Blind Awareness, 2021, https://www.colourblindawareness.org/colour-blindness/. [Accessed 15 Mar 2021].

"Vision Impairment And Blindness". Who.Int, 2021, https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment#:~:text=Prevalence,has%20yet%20to%20be%20addressed [Accessed 15 Mar 2021].

Broome, M R et al. "Mood instability: significance, definition and measurement." *The British journal of psychiatry: the journal of mental science* vol. 207,4, 2015, pp. 283-5. doi:10.1192/bjp.bp.114.158543 [Accessed 12 Mar 2021].

Caldeira, Clara et al. "Mobile apps for mood tracking: an analysis of features and user reviews." *AMIA ... Annual Symposium proceedings. AMIA Symposium* vol. 2017 495-504. 16 Apr. 2018 [Accessed 12 Mar 2021].

Desmet, Pieter. "Design For Mood: Twenty Activity-Based Opportunities To Design For Mood Regulation". International Journal Of Design, vol 9, 2015, pp. 1-19., https://www.researchgate.net/publication/281457299_Design_for_Mood_Twenty_Activity-Based_Opportunities_to_Design_for_Mood_Regulation. [Accessed 13 Mar 2021].

Eiseman, Leatrice. Color-messages & meanings: A PANTONE color resource. North Light Books, 2006.

Kurt, Sevinc, and Kelechi Kingsley Osueke. "The Effects of Color on the Moods of College Students." SAGE Open, Jan. 2014, doi:10.1177/2158244014525423 [Accessed 15 Mar 2021].

Mohsin, Manshad Abbasi, and Anatoly Beltiukov. "Summarizing Emotions From Text Using Plutchik'S Wheel Of Emotions". Proceedings Of The 7Th Scientific Conference On Information Technologies For Intelligent Decision Making Support (ITIDS 2019), 2019. Atlantis Press, doi:10.2991/itids-19.2019.52. [Accessed 13 Mar 2021].

Panchal, Nirmita et al. "The Implications Of COVID-19 For Mental Health And Substance Use". KFF, 2021, https://www.kff.org/coronavirus-covid-19/issue-brief/the-implications-of-covid-19-for-mental-health-and-substance-use/. [Accessed 12 Mar 2021].

Satcharoen, Kleddao. "The Influence Of Colour On Intention To Adopt Food Delivery Service Mobile App". Proceedings Of The 3Rd International Conference On Communication And Information Processing - ICCIP '17, 2017, pp. 87-91. ACM Press, doi:10.1145/3162957.3163018. [Accessed 15 Mar 2021].

Totterdell, Peter et al. "Forecasting Feelings: The Accuracy And Effects Of Self-Predictions Of Mood". Journal Of Social Behavior And Personality, vol 12, 1997, pp. 631-650., https://www.researchgate.net/publication/234066174_Forecasting_feelings_The_accuracy_and_effects_of_self-predictions_of_mood. [Accessed 13 Mar 2021].

Van Rheenen, Tamsyn E et al. "Mental health status of individuals with a mood-disorder during the COVID-19 pandemic in Australia: Initial results from the COLLATE project." Journal of affective disorders vol. 275, 2020, pp. 69-77. doi:10.1016/j.jad.2020.06.037 [Accessed 12 Mar 2021]

Web Content Accessibility Guidelines 2.0, W3C World Wide Web Consortium Recommendation (http://www.w3.org/TR/200X/REC-WCAG20-20081211/, Latest version at

http://www.w3.org/TR/WCAG20/) [Accessed 13 Mar 2021].

Willcox, Gloria. “The Feeling Wheel: A Tool for Expanding Awareness of Emotions and Increasing Spontaneity and Intimacy.” Transactional Analysis Journal, vol. 12, no. 4, Oct. 1982, pp. 274–276, doi:10.1177/036215378201200411 [Accessed 15 Mar 2021].

World Health Organization. Investing in Mental Health. World Health Organization, 2003, pp. 7-8.

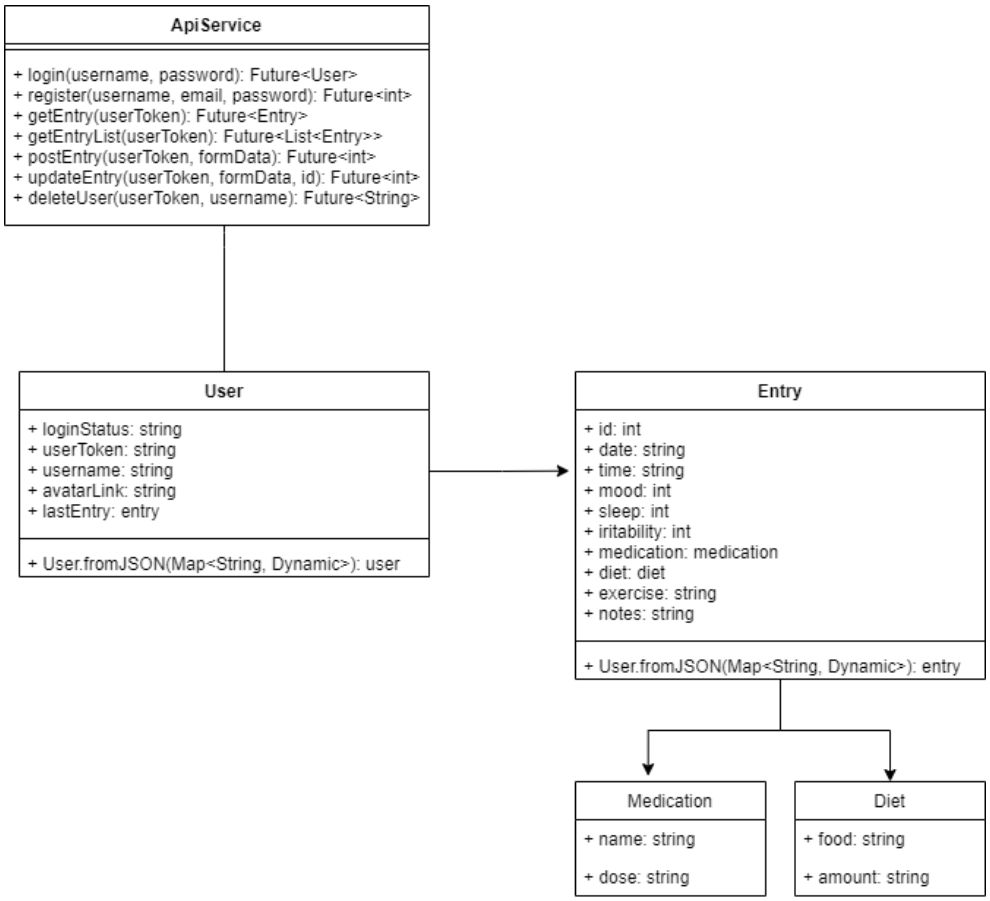
Appendices

Appendix 1 Code developed for this project.

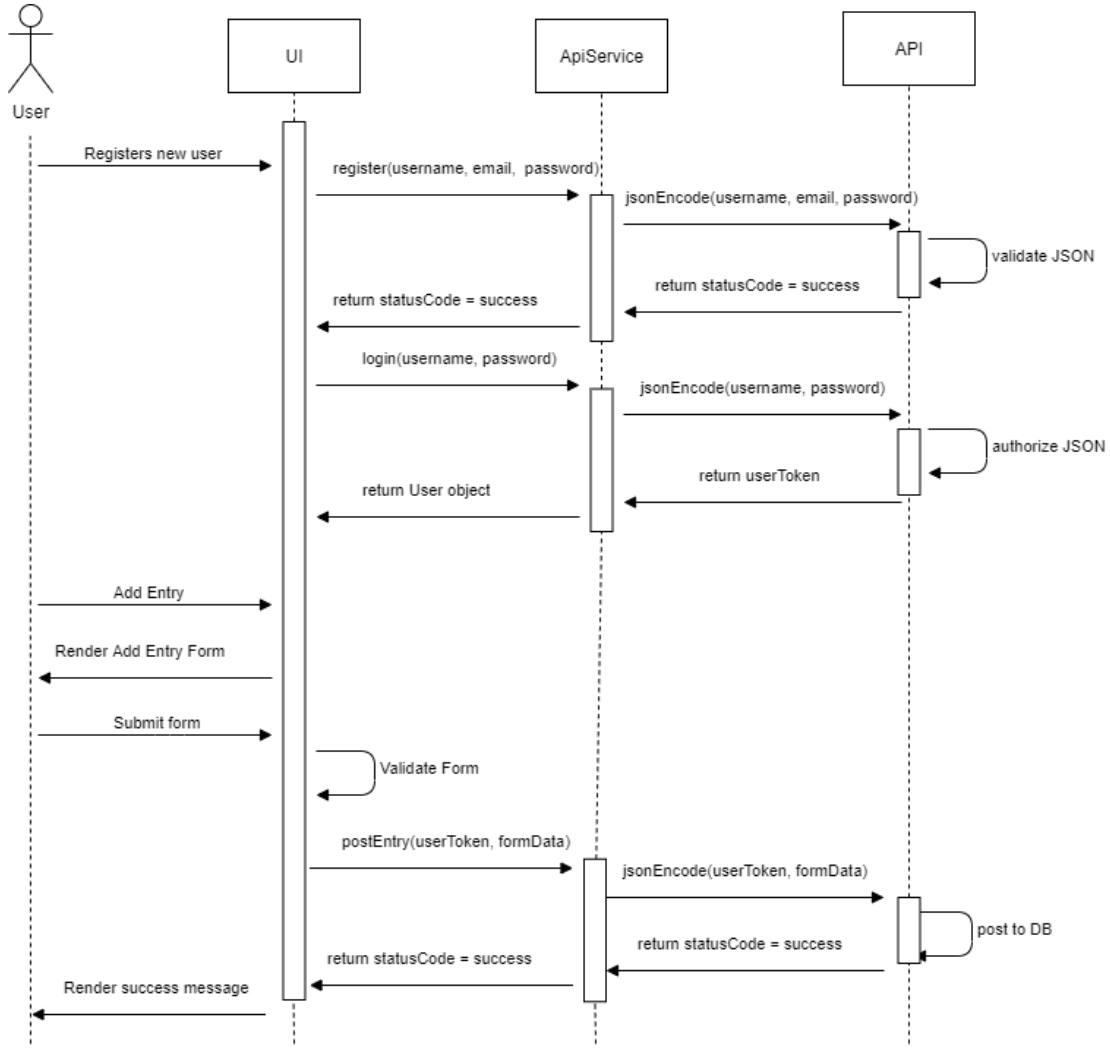
Due to the number of files, code developed for this project is available in the support folder and is accessible via Github: https://github.com/davidryan-mu/moodial.

Appendix 2 UML Class, Use Case and sequence diagrams for this project.

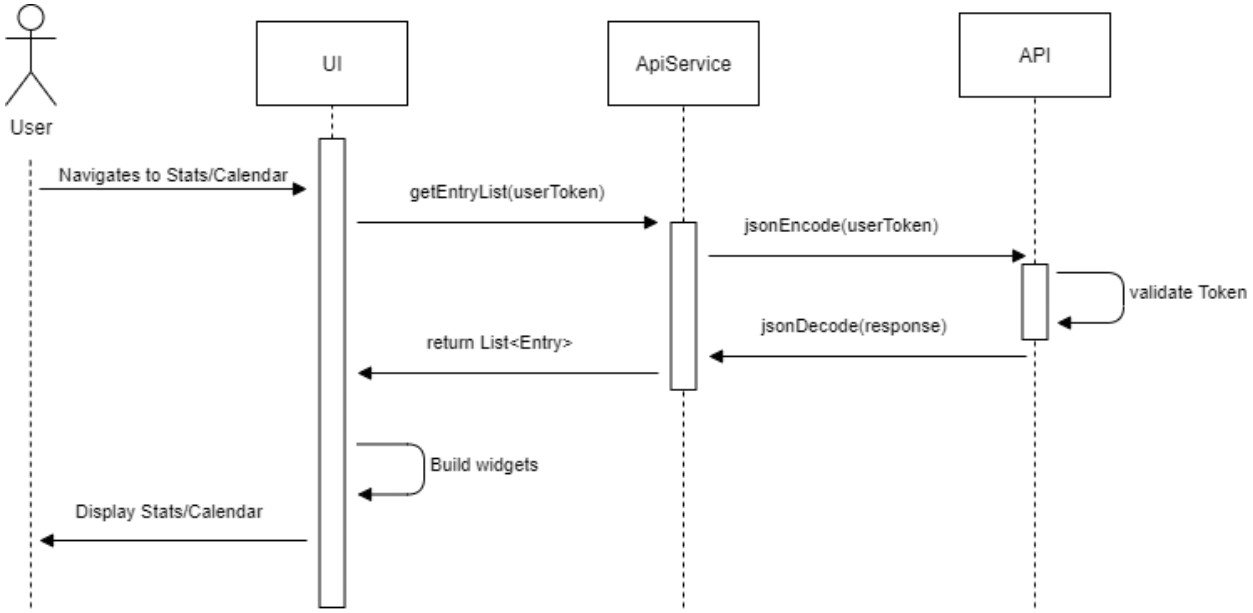
UML Class Diagram for this project



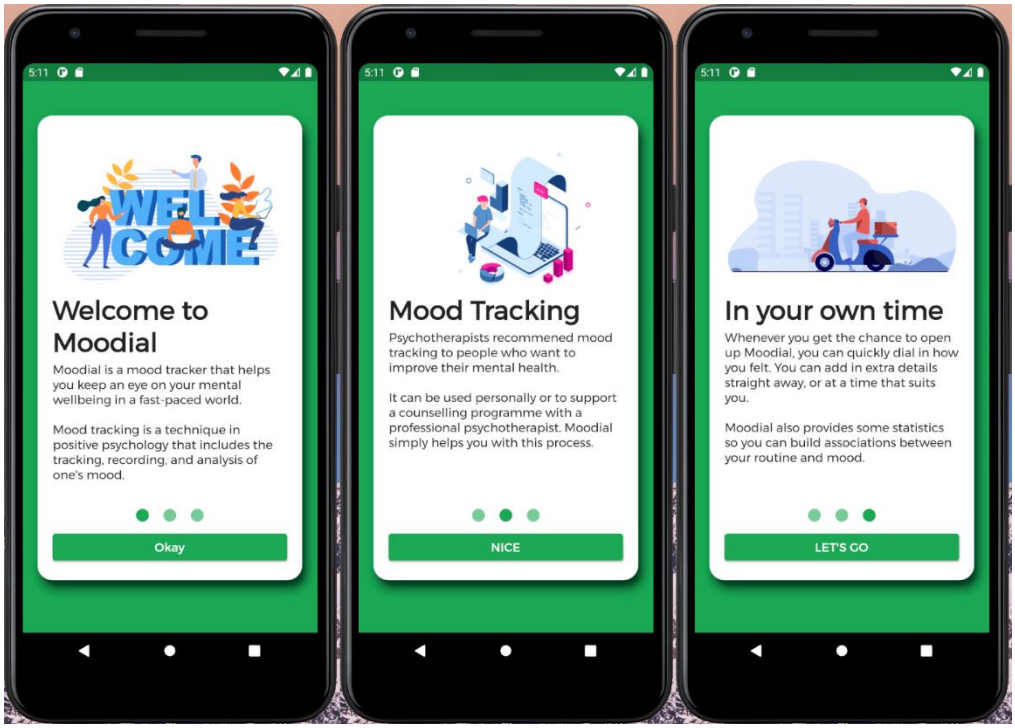
UML Sequence Diagram for Registration, Login & Add Entry



UML Sequence Diagram for building Stats & Calendar Screens

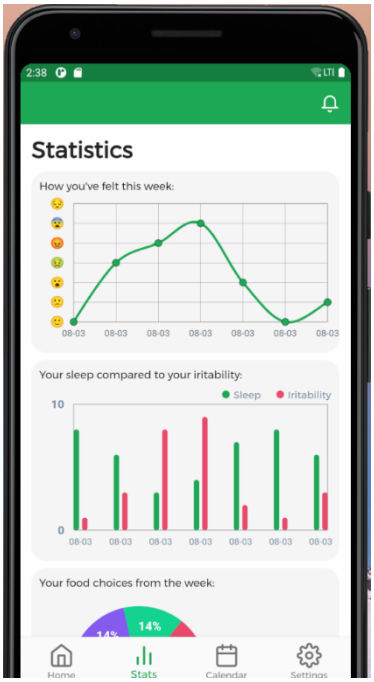


Appendix 3 Screen shots of the project implementation

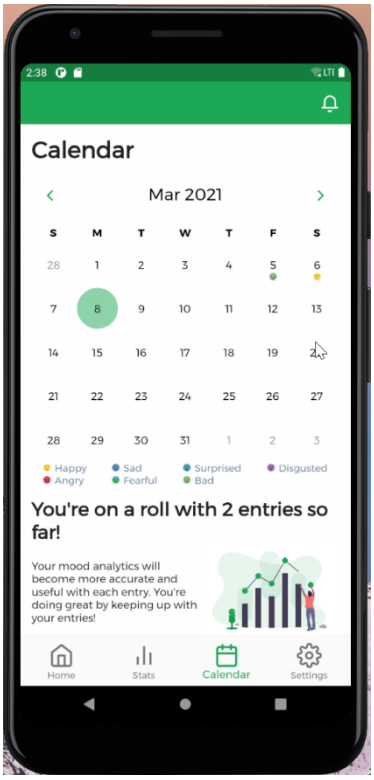


Preparation Screens Implementation

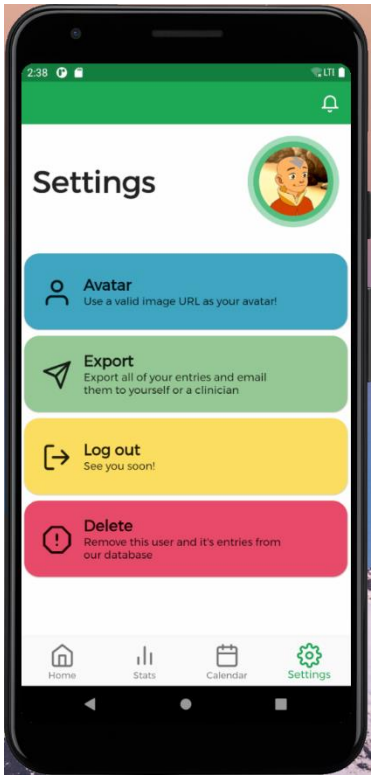
Home Screen Implementation



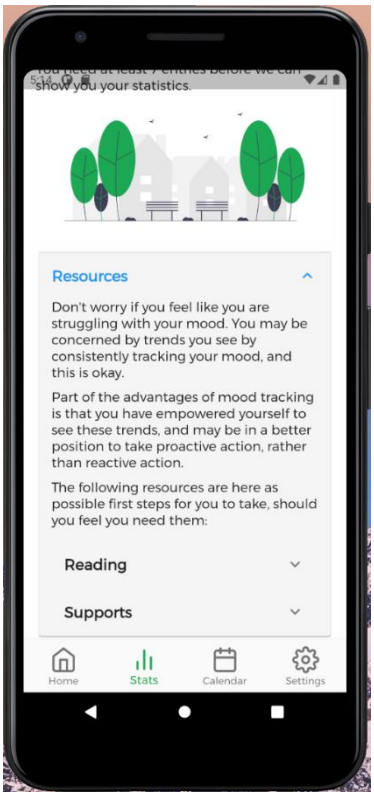
Statistics Screen Implementation



Calendar Screen Implementation



Settings Screen Implementation

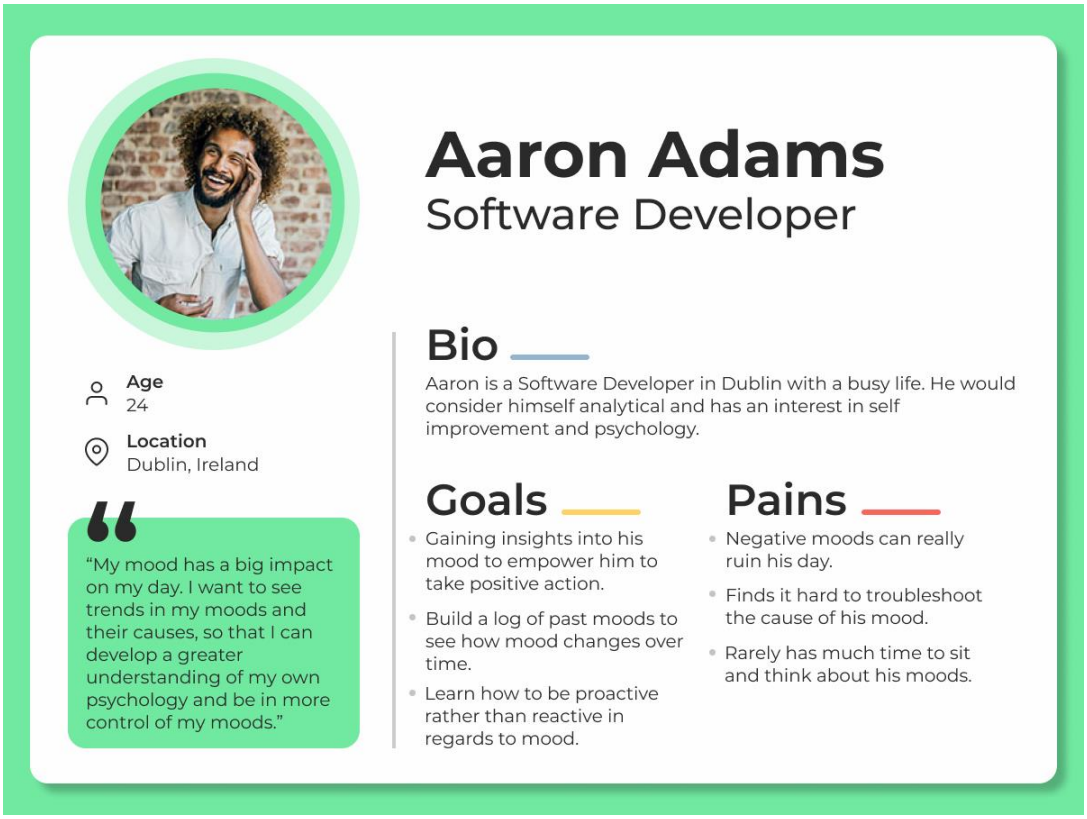


Action Stage Resources Implementation

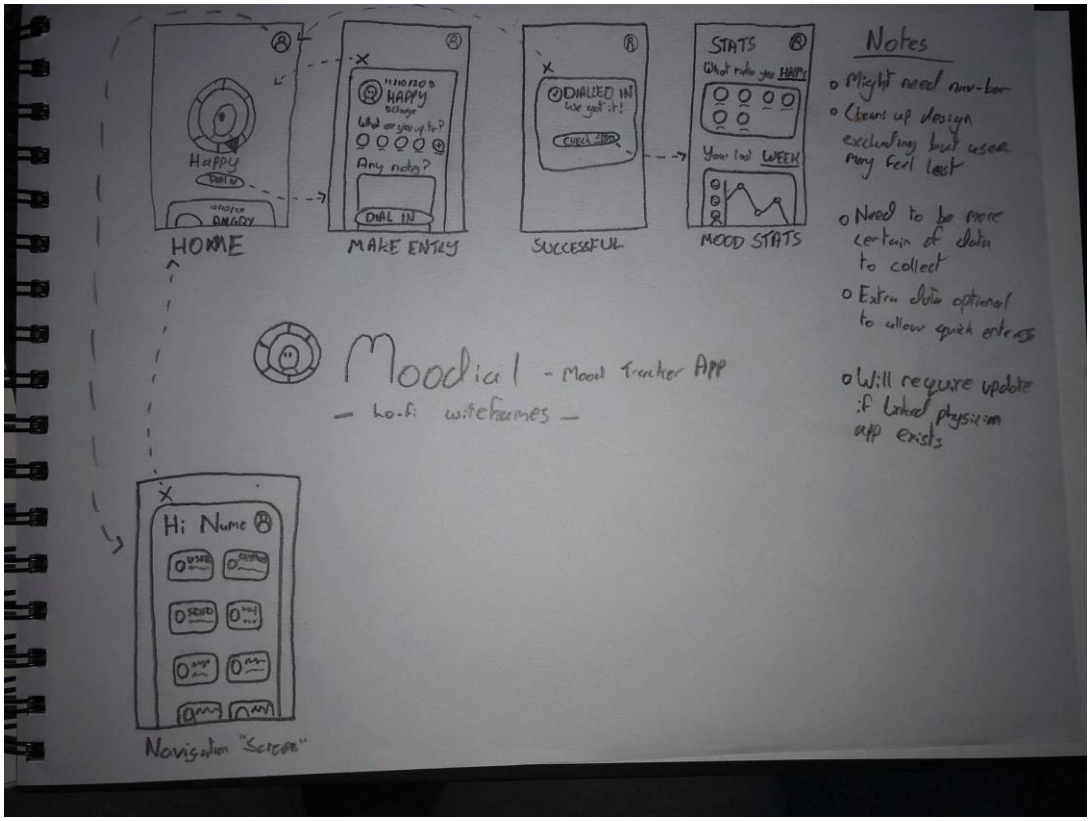
Appendix 4 Extra Material

Design Process Material

User Stories

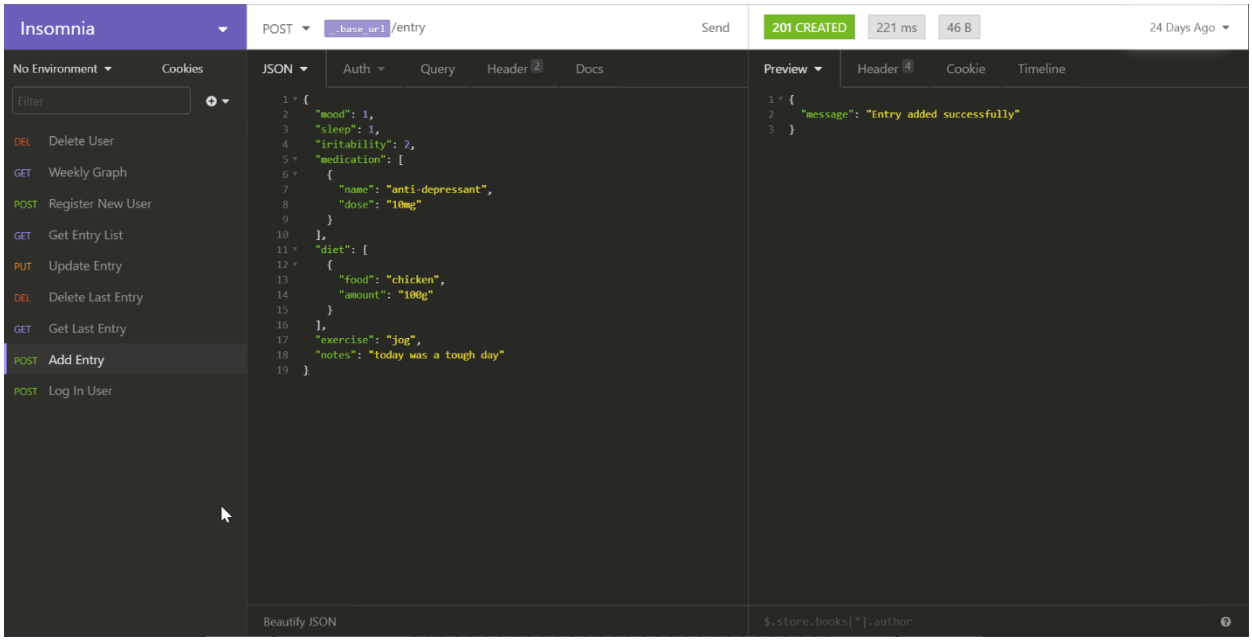


Low-Fidelity Wireframe Sketches



Testing

Insomnia API Test

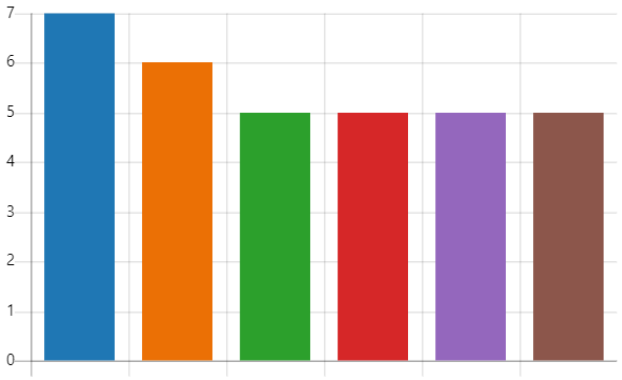


User Testing Survey Results

1. What tasks(s) did you attempt?

More Details

Add Entry	7
Update Entry	6
Mood History	5
Update Avatar	5
Get Statistics	5
Export Entries	5



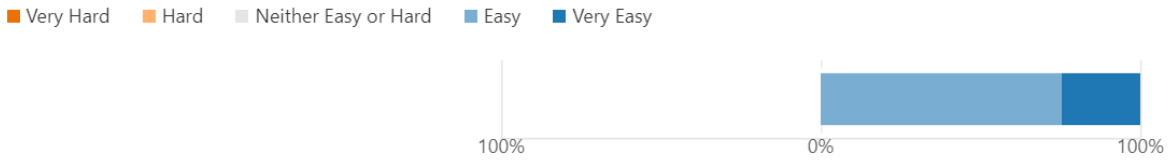
2. Did you complete your task(s)?

More Details

Yes	7
No	0
Some but not all	1

3. How easy was Moodial to use?

More Details



4. Was there sufficient feedback on your actions?

More Details

Yes	7
No	1

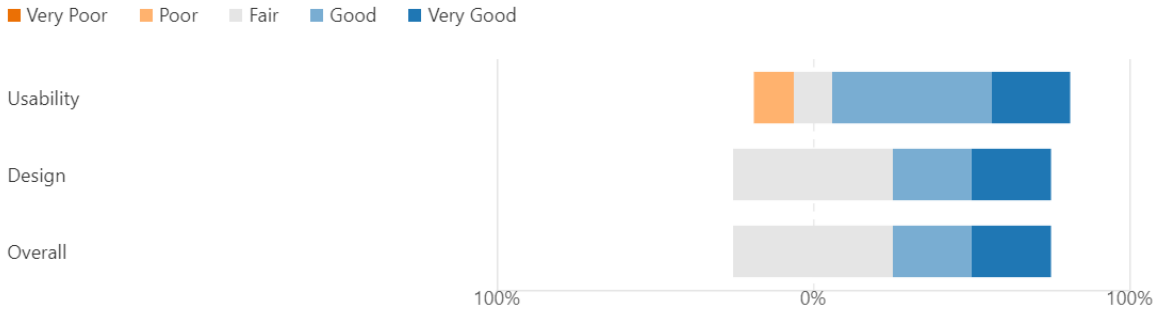
5. If not, please explain using a specific example?

1 Responses

ID ↑	Name	Responses
1	anonymous	When logging in (I must have mistyped my password), the interface responded 'Processing' with nothing else happening. It would be useful to have an indication of what went wrong.

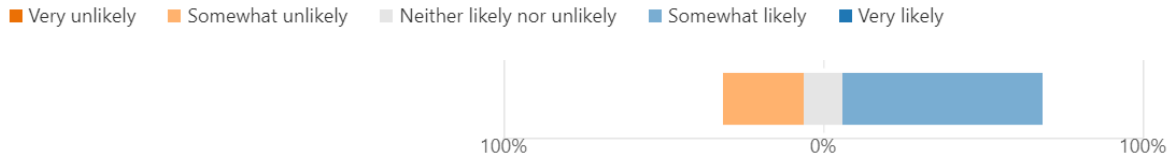
6. How would you rate the following?

More Details



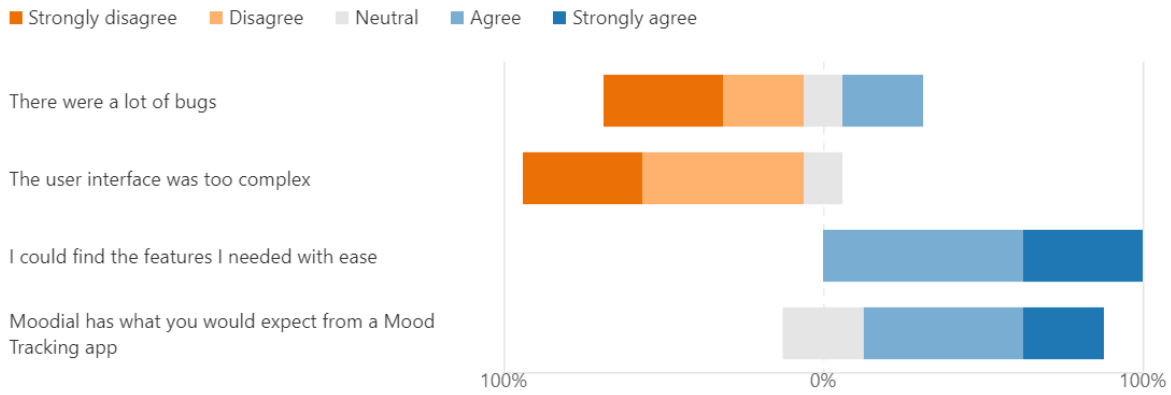
7. How likely would you be to use Moodial?

More Details



8. Would you agree with the following?

More Details



9. Would you recommend Moodial to a peer?

More Details

Yes	8
No	0

10. Why to the above?

6 Responses

ID ↑	Name	Responses
1		Seems like an easy way to develop a wellbeing habit
2	anonymous	Seems like an interesting app that would help people to stay well mentally, which I believe is important especially in the current times we're living with the pandemic ongoing.
3	anonymous	Simple design
4	anonymous	I think it is useful to keep an overview. We tend to forget our mood landscape over the course of time and this would be an easy tool to keep track of what matters most. I would however have serious privacy concerns.
5	anonymous	I think it will help to develop skill of self awareness
6	anonymous	I can show you graphs according to your moods which is helpful for diagnosing illnesses either mental or physical.

11. What was your favourite feature in Moodial? Why?

8 Responses

ID ↑	Name	Responses
1		Entries summary
2	anonymous	I like the idea of the statistics and the calendar to track actual changes on the mood which are important to keep a healthy mind, both need refinements though so that they actually reflect the relevant information to the users.
3	anonymous	Stat tracker - Nice to visualise day by day
4	anonymous	i liked the drag motion with the animation for choosing the mood
5	anonymous	very easy to edit feelings, well done.
6	anonymous	Mood Entry, although very buggy. Also liked the calendar feature, although it did not respond when clicking on a day.
7	anonymous	Statistics. Because it shows a bigger picture of changes in the mood, food, sleep, etc
8	anonymous	Statistics, because graphing helps see a trend in moods.

12. What feature did you like the least in Moodial? Why?

7 Responses

ID ↑	Name	Responses
1		The input of the fields for an entry, I would like not to have to type everything
2	anonymous	The "New Entry editor" can be improved in several ways, right now is tedious to fill an entry manually and it is not really appealing enough for doing it everyday. As this is a core functionality of the app I'll suggest work on it before adding or improving other features.
3	anonymous	Entry for meals was a bit too vague potentially.
4	anonymous	the drag part was above the emotions. so when im dragging, the emotions are blocked
5	anonymous	The Avatar. It is far too complex for what is offers.
6	anonymous	Insertion of avatar, because it's much easier to choose an image from your device
7	anonymous	Avatar. That's because it's hard to use and find a picture online according to your tastes. There should be an upload option for uploading a personal avatar.

13. What features weren't present that you would want or expect from a mood tracking app?

6 Responses

ID ↑	Name	Responses
1		None
2	anonymous	I'll add a reminder feature for adding a daily entry and perhaps a monthly summary of the entries. I'll also hide previous entries and keep on the main screen the one's from today and maybe some sort of day's summary of just the entries that were created on a day.
3	anonymous	I think maybe more options for mood could be useful. A lot of days where I'd describe my mood as indifferent or neutral, rather than happy or sad etc.
4	anonymous	N/A
5	anonymous	Synchronization with other devices like fitness trackers to get info about sleep
6	anonymous	Uploading picture as avatar. Uploading a picture in the description on adding a mood. There should be an option for adding social contact hours - Family / Friends.

14. Any other comments?

6 Responses

1		None
2	anonymous	Here's some general feedback which could be useful for the developer(s): - Update avatar test: I wasn't able to change to my avatar using a public link to a SVG image, if it constraint to some formats it may be useful to specify which ones. Another option could be using a third-party service like Gravatar if you're going to use only avatars through web links. -Bug: I ran into a bug where all my entries went missing from the app, the were restore after two restarts of the app. -Statistics page: I don't understand the need for the graph of "emotions" but if you're going to keep it consider inverting the order of the emotions so that the Y axis resembles the decay on the mood (so that Happy is on the top and not the other way around) otherwise it becomes very confusion to understand the mood changes as they seem to be inverted. -Consider adding multiple entries for the fields "Medication", "Diet" and predefined units for "Exercise", such as hours, minutes) - Resources: I would create a separate section for the resources section as this seems to be very relevant and seems quite hidden in the current design (at the bottom of the Statistics page)
3	anonymous	Really well done, especially seeing its only in testing stages.
4	anonymous	it just says "iritability" with a number scale. what are these numbers? a little explanation would be good. also, ive probably eaten more than one thing in the day. maybe an option for more food?

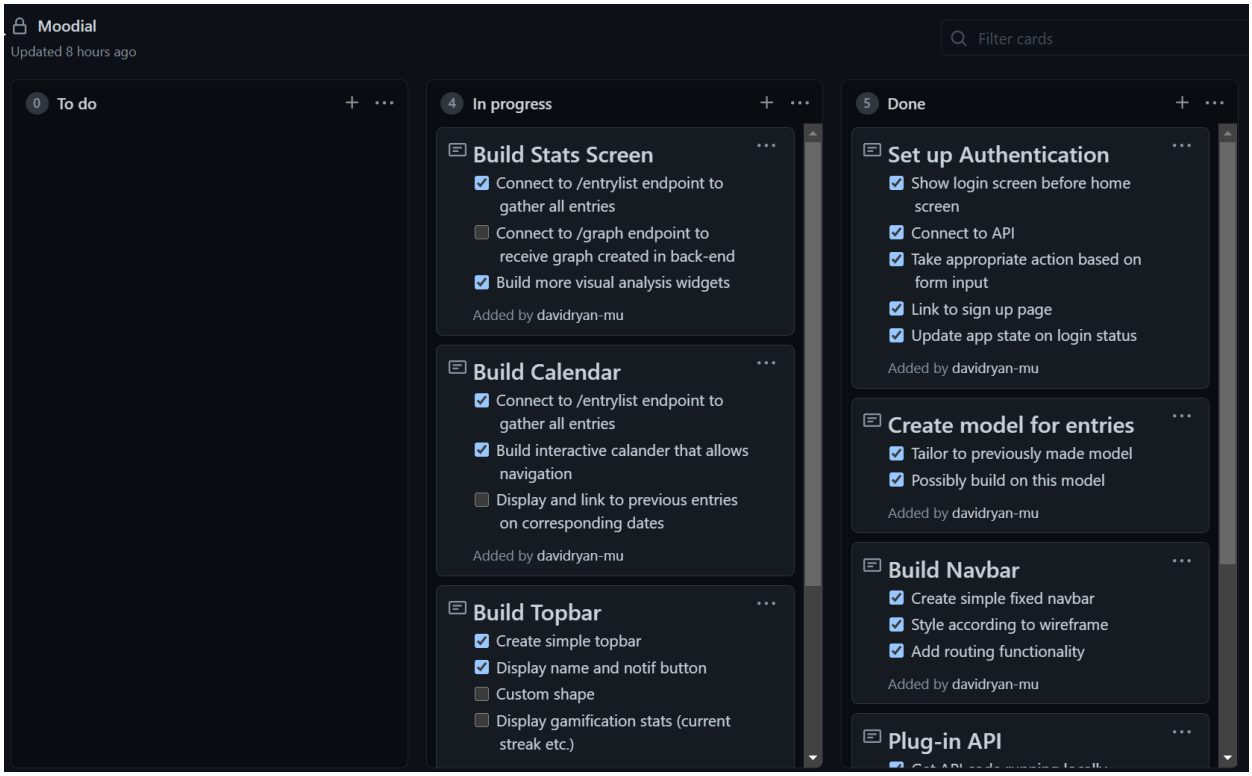
14. Any other comments?

6 Responses

5	anonymous	very good, very easy to do!
6	anonymous	1) When entering a mood (e.g. Happy), the eye is drawn to 'Iritability'. When selecting a value for this featurer (which is not explained, so I would have to guesst that a higher number means that I do feel more iritable) I get an error because I did not first enter the hours that I have slept. When selecting iritablity again, the cursor jumps back to sleep (even though I have already entered it). This may be a bug. 2) Medication and dose seem very generic. Would it be possible to have common measures (eg. mg. ml, etc.) to help peope expressing this part better? 3) Food seems very generic. I was not sure what is expected here. If I eat dinner, would I need to enlist everything that was in it? Would I need to enter an entry every time I slept or ate or would it not matter? Does it matter that my food and my sleep is disaccociated from my emotion (e.g. hours earlier)? The mental model behind the data is unclear to me, but I may be overthinking here :) 4) When clicking on a mood in the calendar, nothing happens. I noticed that there seems to be text stuck below the calendar that I cannot reach. 5) When I want to update an avatar, the last thing I would expect is text-entry for a link. Even though you provided a link on Teams, I consider this a rather weak (and complex) interaction. People will not really copy and paste links for that (as people tend to be lazy in this regard). Instead, I would more likely expect picking an avatar from a selection, be able to upload a picture, or choose a picture that is then converted into an avatar with a service (e.g. turing pictures into cartoon version for pricacy). 6) Somehow I broke the entry feature by entering a range of false

Project Management

Kanban Chart



Daily Work Diary

Mood Diary App Front-end - Daily Diary

Foreword

The previous semester's daily diary was accidentally lost in a wipe of my Google Drive. I have submitted a recovery form to Google in the hopes of getting it back but could be waiting a while for a response due to Covid. I will use this document going forward after picking up progress again now that exam season is over.

The current progress summarised is that the design of the User Interface has been completed in the form of a high-fidelity prototype along with preliminary user personas etc. that were produced as part of the design process. This was preceded by plenty of research into the areas such as mood/emotion, mood journaling and design aspects such as use of colour and accessibility issues.

There is currently a running app in the early stages of development using the mobile development framework, Flutter, written in the Dart programming language. The app is connected to the API running locally, which was created in a previous student's project. There are some issues with certain endpoints of the API, however the app currently allows a user to log in as a verified user, which will provide the foundation for working with the other available endpoints.

Wednesday 3rd Feb 2021
Returning back for semester two, I had my first meeting with John today since early December. We discussed the progress of the project so far and he seemed to be happy with the work that has been done, which put some of my concerns at ease. We are now planning a weekly meeting to ensure the development of the project stays on track. The next steps until the next meeting are to properly test the API in a tool called Insomnia and determine what features are available and working, so we can decide what features will be present in the front-end and if the API will need some work to add new features to that. From what John said in the meeting, I think he wants to do this to solidify how the API handles data and what it returns - effectively unit testing this portion of the project.

Thursday 4th Feb 2021
Connected the existing API running locally to Insomnia which enables me to test the available endpoints with different data. The /entrylist endpoint, which provides a list of all a given user's entries, and the /weeklygraph endpoint, which provides a generated graph based on those entries, both return errors from the API code. All other endpoints work fine, allowing for CRUD operations, which would suggest the issue doesn't lie with the database.
Friday 5th Feb 2021
Got an assignment for another module so worked on that to get it completed and out of the way.

Monday 08/02/2021
Starting the process of building interfaces for each of the currently available API endpoints. Today I fixed up the login screen and added a sign up feature. Toggling between these screens and executing the corresponding API calls is working quite smoothly. Signing up also automatically logs the user in to save them typing in their details twice. Next I'll work on building a display of the data contained in the user's last entry.

Tuesday 09/02/2021
Didn't hear back from John about the meeting that was being planned for today. Continued trying to build the front-end for the API's available endpoints. Currently have the app displaying data from the user's last entry through a get call to the /entry end point. There was a bit of fiddling around with data types due to the API returning nested objects but all of the returned data is now accessible. The data that the API collects for a mood diary entry is conflicting with my UI design as I received the API code after the design process. I'll have to discuss this with John to decide on the best way to deal with this going forward. In the process of building and styling a UI component in the form of a card to display this data.

Thursday 11/02/2021
The work carried out to grab data from the /entry endpoint needed to be restructured to eliminate unnecessary API calls which would have caused extra network traffic. Spent the day going about this and further developing the card component for data display. Had a meeting with John in the morning. It was decided that the API endpoints throwing errors could be replaced with hardcoded versions for now as the focus of the project is to build the app around the API. If there's time at the end the API can be actually fixed, but I'm still concerned about getting the project done with the time left. I'm going to work

with the functioning aspects of the API for now to get as much done as I can before diving into any of the API's code.

Friday 12/02/2021
Building a component to allow for adding entries via a post message to the /entry endpoint of the API. Passing data around widgets and getting widgets to rebuild is the main recurring issue as it can be quite temperamental getting Flutter to do what you want. Relying quite heavily on material design components, however, the design isn't too far off the wireframes in terms of what has been worked on so far.
Monday 15/02/2021
The form for adding entries is built and functioning with the corresponding API call. There was some tinkering required to ensure each of the data types being sent across were what was desired, mainly due to the nested objects. Spent the day reusing these components to build a form for updating entries. This is functioning but requires some refining to ensure the last entry displayed is updated with the new details.

Tuesday 16/02/2021
Refined yesterday's work as there were issues with the app being aware of the data change of the updated entry at a level high enough to trigger a rebuild of relevant widgets. Also began progress on building screens past the landing and home screens. Implemented navigation that conditionally renders the correct widget based on nav bar value. The next steps are structuring these screens with dummy content for now. I'll then have to add finer details to the UI such as dynamic colours, before going and fixing the /entrylist endpoint in the API as the data it should return is fairly essential to the app's functionality.

Thursday 18/02/2021
Remodelled the dial to match the model for mood set out in the API. Also reworked the UI to incorporate this change and look more visually appealing and dynamic to what it previously was. From research into moods, I don't know if the seven possible values are a way of capturing mood but rather the 7 core emotions. Mood seems to be better modelled more on a 2-dimensional scale that can go either side of the axes, as it's a fairly complex concept to capture. However, at this stage in the process I think the better option is to go with what is set out in the API as I believe my project is focused on building on that.

Friday 19/02/2021
Built the settings screen which integrates with the /deleteuser endpoint of the API, allowing the user to remove their account and associated entries from the database. That's all of the functioning API endpoints incorporated into the app now. I also add options for the user to log out or change the avatar with a valid image URL. Going into next week I will build the remaining two screens (stats & calendar), using dummy data for now before trying to hardcode/fix the API code.

Wednesday 24/02/2021
Created randomly generating dummy data that allows me to build the stats and calendar pages using fake entries before I have to fix the API. Currently working on the stats page. Spent some time looking for and trying out chart building libraries for Flutter, settling on Fl_Charts. Have a line chart built that displays the mood values entered over the last seven entries. In an ideal situation the user would add an entry once a day so

this would show them the change in their mood over the last week. However limiting to one post a day would conflict with the fact that the update feature of the API updates the date of an entry to the the date of updating rather than maintaining the earlier date. So if a user was to add an entry and also wanted to update yesterday's entry, this would cause issues if there was a one day limit. Currently building a bar chart that compares the sleep to irritability values.

Thursday 25/02/2021
Finished off the previously mentioned chart and also created and finished a pie chart displaying the user's dietary choices over the past week. All of the charts/graphs are responsive to new data. This is conveyed by the fake entry data being instantiated with newly (somewhat) randomized data each time the graph widgets are built to the UI.

Friday 26/02/2021
Added functional components to the calendar screen. Using a nice calendar carousel library which allows for marking dates with event objects which I am using for entries given their date value. Aside from making use of a legend-type widget to show a user the mood value they entered on a date, there isn't much more functionality to the calendar screen as it stands. I'd like for a user to be able to be able to tap on a date to access the corresponding entry and view all of the inputted data, with the possibility of updating any of the fields. Regarding the previously mentioned issue, allowing for a user to make more than one entry on a date, and updated entries being appended with a new timestamp rather than maintaining their original date - the calendar responds nicely to this by allowing the marked dates to have a stack of events represented in the UI. Future work might involve creating a pop-up modal after tapping on a date, showing options for all entries on that date, then linking to update forms after selecting an individual entry.

Saturday 27/02/2021
Haven't had a meeting with John so I recorded a video to outline the progress made so far and the app's current state. I think the video format might work well as he can watch the video in his own time and hopefully give some feedback without the limited timeframe of a midday meeting.

Monday 01/03/2021
Sent the video to John in the morning, he said get on to me about a meeting in the afternoon after he'd watched the video. Didn't hear from him in the end so worked on an assignment for another module.

Tuesday 02/03/2021
Fixed the issues within the API without having to take advantage of any hardcoding. The issue was that the return type of the EntryList.get method wasn't correct. I imported Flask's make_response method to pack the json and status code into the correct type. The API now returns a list of entries as JSON objects. Next I'll update my app to build the Stats and Calendar widgets after grabbing this data. There was a deprecated method in use for the insert call within the API so I updated that.

Wednesday 03/03/2021
The Stats and Calendar screens now grab data from the API and build based on that, handling loading times, null returns etc. This allowed me to get rid of the fake entries.

This marks the core functionality of the app being present so any work now will be improvements based on that. Had a meeting with John this afternoon. He gave great feedback and suggestions on how to improve elements of the UI/UX. The dial could be much improved on it's current (very) alpha state by making better use of the available real estate and utilisation of building colour associations through the emotes. He also mentioned that it would be worth looking into the analytics provided in the Stats screen to ensure they coincide with a user's expectations for what should be present there.

Thursday 04/03/2021
Redesigned the dial to have better UX and be more visually appealing. Made these changes consistent throughout the app. Also implemented an export feature which converts the user's list of entries to a CSV file and opens an email in their native mail application with the file attached and an email ready to send. John is happy with the changes and the plan going forward - which is to tidy up the app, learn how to bundle it up and get it running on mobile devices for testing purposes, and then get to writing the thesis.

Friday 05/03/2021
Added a resources section to the Stats page to give a user information on useful links and phone numbers should they be struggling with mood or feel worried based on their mood tracking results. Also reformatted the recent entries section on the homepage to build off of the /entrylist endpoint rather than the /entry endpoint. Altered the API to withhold the original date and time when updating an entry.

Monday 08/03/2021
Got Sean's API deployed to Heroku as a web application. This took a fair bit of playing around with reworking code to be consumed by Heroku. Rebuilt the dependencies of the server code in this process which got the file size down by ~20MB. The API can now be accessed from any device with a network connection. The first request tends to be a bit slow as it's being hosted for free but it more than does the job for this part of the project.

Tuesday 09/03/2021
Got a build version of the app available to Android testers through Google play. Getting a successful build was a bit of work but the process is quite simple once you get a working bundle uploaded. This allowed me to test out the app on all the Android devices I have in my house and send the app out to people I know with Android mobiles. I found the Android OS UI was covering the Save buttons so I could amend this with some padding and roll out an update fairly easily. Doing the same for iOS would mean I would have to spend €100 on their developer subscription and then wouldn't be able to test it properly without a mac. I got an iOS build version of the app through Codemagic and sent it on to John. Hopefully he can get it running on his machine. Had a meeting with John then to discuss the way we'll test the app.