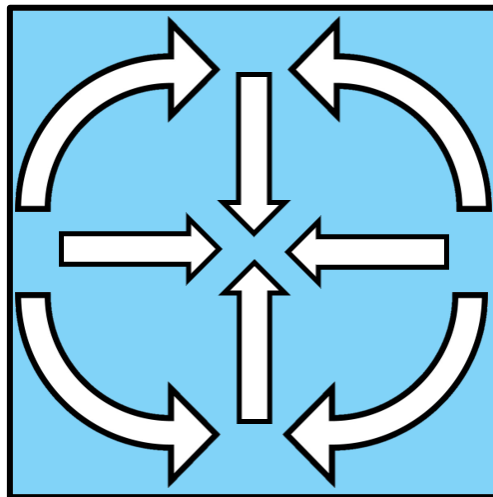




# **Final Report**

## **A Personal Diary App**



**Module: CM3203 One Semester Individual Project**

**Student Number: C1647731**

**Author: Jack Williams**

**Project Supervisor: Alia I Abdelmoty**

**Project Moderator: Philipp Reinecke**

## Abstract

Cognitive behavioural therapy (or CBT) is a talking therapy to help you manage how you think and behave. [1] A personal diary is one of the methods used to help a patient undergoing CBT. The purpose of a diary is to keep a record of different events or experiences that happen on a daily occurrence. This is used by the therapist to gain a better understanding of how a patient handles certain situations, and for patients to self-reflect on their behavioural patterns. In this modern era, traditional paper diaries can be troublesome due to the potential lack of confidentiality and impracticality of having one on hand at all times. In this report I will be showcasing how a simple diary application can be used to greatly improve the experience of a CBT patient.

One of the key advantages of a diary application is that, unlike paper diaries, the therapist would be able to instantly access all the patient's entries which will include any trouble they had that day with concurrent emotions felt during that time.

This report will outline the whole process of the development of the application, from the design, approaches and implementation to testing and evaluation.

## Acknowledgements

I would like to take this opportunity to thank all my friends and family for their support during the entire project. It has been a tough few months for everyone and still, they were there for me when I needed them to be.

I would also like to thank my amazing supervisor, Alia Abdelmoty, for guiding me through my project, picking me up when I was down and constantly providing me with external resources to guide me mentally and help with my project.

## Contents

Abstract.....	2
Acknowledgements.....	3
Table of Figures.....	6
Introduction .....	8
Project Overview.....	8
Project Aims .....	9
Personal Aims.....	9
Background .....	10
The Problem.....	10
Existing Solutions .....	10
Existing Solutions State Transition Networks .....	16
Therapist Diary Research .....	18
Personas.....	21
Project Constraints.....	23
Approach.....	24
Planning .....	24
Learning .....	24
Developer Guides.....	24
Data Storage.....	25
User Interface Design.....	25
User Interface .....	26
Specification and Design .....	27
Functional Requirements.....	27
Non-Functional Requirements.....	29
Use Cases .....	30
Prototype .....	38
First Design.....	40
Final Prototype Design .....	49
Implementation .....	51
UML Class Diagram .....	51
Database .....	52
User Schema .....	52
System.....	54
Postman .....	54
Create New User .....	54

Login.....	56
Retrieve User.....	59
Logout .....	61
Navigation .....	62
Navigation Overview.....	64
Add New Entry .....	65
Situation.....	67
Emotions .....	68
Worry .....	72
Final Entry Preview .....	75
My Entries .....	76
View an Entry and Publish .....	76
Refresh Entries .....	78
Delete an Entry .....	79
Search and Sort.....	80
Quick Add Entry .....	80
Implementation Conclusion.....	81
Results and Evaluation .....	83
Application Outcome Compared with Requirements.....	83
User Testing .....	85
User Test Results.....	87
Future Work.....	90
Conclusions .....	91
Reflection on Learning .....	92
References .....	93

## Table of Figures

Figure 1 - CBT Thought Diary Application: Add a new diary entry .....	17
Figure 2 - CBT Thought Diary Application: View a previous diary entry. ....	17
Figure 3 - 'Hot Cross Bun' found at <a href="http://psychologytools.com/cross-sectional-formulation.html">http://psychologytools.com/cross-sectional-formulation.html</a> ...	20
Figure 4 - Initial ideas for colour schemes .....	39
Figure 5 - Application Logo .....	40
Figure 6 - Login Page .....	40
Figure 7 - Home Page .....	41
Figure 8 - Menu Page .....	41
Figure 9 - Entries Screen .....	42
Figure 10 - Single Entry.....	42
Figure 11 - Situation Screen .....	43
Figure 12 - Emotions Screen .....	43
Figure 13 - Worry Screen .....	43
Figure 14 - Person Overlay .....	44
Figure 15 - Location Overlay.....	44
Figure 16 - Calendar Overlay's .....	45
Figure 17 - Application Information Screens .....	45
Figure 18 - Improved Home Screen .....	49
Figure 19 - Improved Entry Screens.....	50
Figure 20 - UML Class Diagram .....	51
Figure 21 - User Schema .....	52
Figure 22 - Example User and Entry.....	53
Figure 23 - Create User Route.....	54
Figure 24 - Mongoose 'pre' middleware.....	55
Figure 25 - User Hashed Password Example.....	55
Figure 26 - generateAuthToken function.....	56
Figure 27 - User Token Example .....	56
Figure 28 - New User Full Example .....	56
Figure 29 - Login Route .....	57
Figure 30 - findUserByCredentials Function .....	57
Figure 31 - User Log in Full Example .....	58
Figure 32 - Home Screen.....	58
Figure 33 - Get User Route.....	59
Figure 34 - Custom Authenticate Middleware.....	59
Figure 35 - Get User Postman Example .....	60
Figure 36 - Failed Logout Example .....	60
Figure 37 - Full Get User Postman Example.....	60
Figure 38 - Logout Route.....	61
Figure 39 - removeToken Function .....	61
Figure 40 - Logout Postman Example .....	62
Figure 41 - Navigation Container .....	62
Figure 42 - Home Screen Component.....	63
Figure 43 - HomeDrawerScreen.....	63
Figure 44 - Entry Screen Component.....	63
Figure 45 - EntryTabsScreen .....	64
Figure 46 - Navigation Overview.....	64
Figure 47 - Entry Help Screens .....	65
Figure 48 - Clear Entry Example.....	65
Figure 49 - Text Input Function Example .....	66
Figure 50 - AsyncStorage setItem .....	66

Figure 51 - Situation Screen .....	67
Figure 52 - DatePicker .....	68
Figure 53 - Emotions Screen .....	69
Figure 54 - AddEmotion Function .....	69
Figure 55 - SubmitHandler Function .....	70
Figure 56 - Duplicate Emotion Alert.....	70
Figure 57 - EmotionItem Display.....	71
Figure 58 - Remove Emotion Alert.....	71
Figure 59 - Remove Emotion Full Example .....	72
Figure 60 – Picker Component.....	72
Figure 61 - Full Picker Example .....	73
Figure 62 - onSubmit Full Entry.....	73
Figure 63 - Entry Validation Check.....	74
Figure 64 - Full Entry Validation Alerts .....	74
Figure 65 - Full Entry Preview .....	74
Figure 66 - Final Entry Preview useEffect Function .....	75
Figure 67 - Full Entry Example .....	75
Figure 68 - My Entries Screen .....	76
Figure 69 - Entries Flat List Display .....	76
Figure 70 - Custom Card Component.....	77
Figure 71 - Select Entry Button .....	77
Figure 72 - Route Values .....	77
Figure 73 - My Entries Entry Example .....	77
Figure 74 - Refresh Entries Button.....	78
Figure 75 - newEntryHandler Button .....	78
Figure 76 - newEntryHandler Function .....	78
Figure 77 - Delete Entry Icon .....	79
Figure 78 - Remove Entry Function.....	79
Figure 79 - Delete Entry Example.....	79
Figure 80 - Search and Sort Design .....	80
Figure 81 - Quick Add Entry Screen .....	80
Figure 82 - Hypothetical Quick Add Entry Example in My Entries.....	81

## Introduction

As we continue to learn how every human is different physically, mentally and emotionally, mental health becomes more and more prevalent in our society. This is an aspect of life that has previously had a stigma surrounding it. Thankfully, it is finally being accepted as a problem people face every day and is not something to be embarrassed or to hide away from anymore. The overall goal of my project is to help control the way someone may feel to lead a happier, healthier life mentally.

## Project Overview

Diaries are used by millions of people around the globe. The uses for a diary are extensive, ranging from jotting down a wish list to a simple calendar for work or pleasure – they can be used for pretty much everything. This has led to techniques being developed to use them from a medical standpoint.

Cognitive behavioural therapy (or CBT) is a talking therapy that is used to help change the current way you think and behave to manage your problems. [1] CBT succeeds by following the theory that your thoughts, feelings, physical sensations and actions are connected. Due to this, negative thoughts could leave you with recurring issues in the future.

CBT can be used to treat a number of disorders (a more extensive list is available via [1]):

- Bipolar disorder
- Borderline personality disorder
- Eating disorders – such as anorexia and bulimia
- Obsessive compulsive disorder (OCD)
- Panic disorder
- Phobias
- Post-traumatic stress disorder (PTSD)
- Psychosis
- Schizophrenia
- Sleep problems – such as insomnia

A popular method to treat these disorders is to have the patient fill out a diary every day throughout the course of their treatment. There are many different forms of diaries created to treat different disorders. These different forms contain a range of questions asked by the therapists for the patient to answer, depending on what disorder to treat. For example, an activity diary will ask a patient to note down an activity they have achieved, rate the success of that achievement and a pleasure rating from the activity. For my project, I will be focusing on the 'Worry Diary' which takes a situation and breaks it down into different sections.



## Project Aims

The aim of this project was to create a user-friendly diary application which patients can discreetly and effectively fill out for professional analysis from the therapist. This will allow the patient to get the most out of their treatment. If the app was not user-friendly, the information provided by the user may be tainted and ruin the whole focus of the process, potentially leaving a patient to feel alienated and misunderstood. The application should allow the patient to enter all aspects of the situation they are inputting into the diary, allowing the therapist to see exactly how the patient was feeling at that time.

For the project to be user-friendly and efficient, I adopted a user-focused design approach. This involved me using Nielsen's 10 heuristic evaluation principles to evaluate the layout and performance of the app's functionalities. [2]

These functionalities are:

- To allow the user to create a diary entry as straightforward as possible, without taking away from the overall goal of the entry.
- To allow the user to clearly view their previous entries and choose which ones to publish for the therapist to be able to view.
- To allow the user to express their thoughts and feelings through the different questions and options within the form by adding as many emotions and ratings as they feel necessary.
- To allow the user to quickly add any emotions they may be feeling without the need to type out a full entry. This is particularly useful for a patient to understand their range of feelings throughout the day, regardless of a situational trigger.

To keep the implementation part of the project manageable, I used an Agile based approach, breaking each functionality into different chunks to not confuse both myself and the overall flow of the application. [3]

## Personal Aims

Through my time at university, I have learnt various programming languages, such as Java and Python, and what they can be used for. One thing I have always wanted to be able to achieve is the creation of a mobile application. I had no previous mobile application development experience prior to this project and so used this opportunity to learn a relatively new mobile application framework, React Native.

## Background

### The Problem

Currently, therapists that work exclusively via phone call or video chat have to use awkward means to implement the diary method of treatment due to the lack of face to face contact. Firstly, patients can find the use of a diary uncomfortable so the therapist has to make sure the patient is actually filling the diary out which can prove difficult – this is for a few reasons. The physical size of a diary can be hard to keep on your person at all times, leaving patients no choice but to fill it out once they get home when the experience is no longer fresh in their mind. Another reason is because of the embarrassment and stigma surrounding filling a diary out whilst in public. When the patient is undergoing a situation that they feel their therapist should know about, it could prove quite troubling to take out their diary and fill it in. Another challenge is how the therapist sees the diary. This can range from being scanned in through a printer, sent pictures of the physical diary or even getting the patient to read it off over the phone whilst the therapist makes notes. This delay of information being received doesn't allow the therapist enough time to fully analyse and prepare for the session which has been paid for by the patient.

Because of this, patients need an easier way to communicate their diary entries prior to the appointment allowing their therapist time to analyse and work towards helping the patient recover – this is where my application would come in.

### Existing Solutions

To begin my project, I thought it best to research the current programs that are available and review them. I decided to download each application, highlight any features which stood out, how it interacts with the user and how the app performed data entry tasks e.g. writing an entry. Along with this, I will provide a personal rating out of 5 for the overall performance of each app compared to the average rating provided by the App Store.

All the applications I chose to look over were of the highest rated and so the quality of the application should be fairly high – containing good concepts and ideas for what congregates a successful app.

The applications I reviewed varied from CBT focused diary apps to standard diary apps. This allows me to get a feel of how different focusses vary in their technique to interact with the user, as well as how the designs differ. Alongside this, I was interested to find what features each app chose to implement i.e. adding a photograph, how it is presented to the user and if it works well as an implementation.

<b>Name:</b>	CBT Thought Diary
<b>Author:</b>	Eddie Liu
<b>Link:</b>	Available via: <a href="https://apps.apple.com/gb/app/cbt-thought-diary/id1010391170/?platform=ipad">https://apps.apple.com/gb/app/cbt-thought-diary/id1010391170/?platform=ipad</a>

<b>Description:</b>	<p>This application aims to teach the user how their thoughts impact their mood and behaviour. It mainly focuses on mood tracking via a range of smiley faces from 'Terrible' to 'Terrific'. Once a feeling has been inputted, the application asks for a title, what emotions do you feel and details of what exactly is happening/has happened. The user is then prompted to add any negative thoughts they're feeling, select a list of any cognitive distortions they may have been experiencing, how can the user challenge their negative thoughts and any alternative thoughts they may have felt. Finally, after filling the entry in, the user is asked how they feel after: Better than before, About the same or Worse than before. Once all this has been entered, it can be viewed in the 'Entries' list accessible on the home screen.</p>
<b>Stand out Features:</b>	<p>The app contains instructions on how to use each section within the app. On the home page, the user is presented with an 'i' icon, when pressed displays information on what a 'Thought Diary' is, along with when and how to use one.</p> <p>The step by step process of completing an entry with easy to follow instructions and explanation along the way - some sections only require a simple click of an emoji, emotion or feeling.</p> <p>These entries can be viewed from a simple 'Entries' list accessible from the home screen.</p> <p>Daily notifications can be set up to remind you at a specific time of the day, regardless if you've already accessed the app that day.</p> <p>A simple 'Contact Us' option is available through the menu which will take you to the mail app on your phone and auto fill all necessary details to send a query to the team.</p> <p>By upgrading to the pro version which is either £4.49 per month or £35.99 per year, you receive the options: 'Cloud Sync' to automatically secure your data to the cloud so it can be accessed by other devices, 'Passcode Protection' to add a passcode onto your entries and 'Export Entries' to share your entries with others or save them to a different platform.</p>
<b>My Experiences:</b>	<p>When using the main 'Add Entry' feature, instead of presenting me with a massive list of individual sections I was required to fill out, it presented them one at a time. This meant I wasn't overwhelmed with how much was required of me and I could express myself better. I was also presented with useful encouragement on why I should be filling out certain sections of the entry, for example, after entering a title and choosing a related emotion or emotions, you are asked for details – specifically 'What's going on?'. Beneath this was the following text: 'Sometimes, our thoughts are our worst enemies. It can be helpful to analyse negative thoughts and see if they match up with what really happened.'</p> <p>I feel that users would be pleased to see this and feel a sense of encouragement and comradery from within the app and be more willing and open when filling in sections.</p> <p>If I was stuck on a section, most of the time a small 'i' button was available to provide me with more information. This included descriptions</p>

	<p>of emotions and different examples of when you might feel them. This could be very useful to patients when you're really trying to pinpoint their exact thoughts and feelings at the time of the situation.</p> <p>Once the entries were saved, I was able to see them within the 'Entries' section of the application. Here I was shown the emoji I chose from the very first question presented in the new entry 'How are you feeling?'. Also, I was given the title I chose and the date of the entry. I feel as if more information could have been displayed in this section as it doesn't give the user much feedback on individual entries. Maybe allowing them to sort their entries into different categories e.g. Date (newest to oldest or vice versa) or grouped by emotion rating.</p> <p>Overall, there are a lack of features when it comes to customising your entries and visualising them. There aren't any options to add a location or image to the entries which are very important within a thought diary as it helps to remember the exact situation, allowing you to recall how you were feeling and how you were able to cope with it. When it comes to visualising the data, you are only given the 'Entries' section. It would have been useful to be given a chart or something similar to see the range of emotions you were feeling over the course of a specified time span. Once again, this would help the user to see if they are progressing at all and keep them on the right track – or even change the way they are currently dealing with their issues.</p>
<b>My Rating:</b>	4.5/5
<b>App Store Rating:</b>	4.8/5

<b>Name:</b>	MindShift CBT - Anxiety Canada
<b>Author:</b>	Anxiety Canada Association
<b>Link:</b>	Available via: <a href="https://apps.apple.com/gb/app/mindshift-cbt-anxiety-canada/id634684825">https://apps.apple.com/gb/app/mindshift-cbt-anxiety-canada/id634684825</a>
<b>Description:</b>	<p>This application is packed full of features that doesn't focus on any particular problem – just general anxiety. It offers features from guided meditation to helpful cue cards to the basic thought diary.</p> <p>Users can set goals, learn lots of frequency asked questions and even share your progress via email.</p> <p>The app is completely free with no in-app purchases.</p>
<b>Stand out Features:</b>	<p>When you are loading the application for the first time, you are asked if you would like 'MindShift' to send you notifications.</p> <p>An account is required to be able to access the app where you have to enter an email, password, nickname, birth year and the country you're in. There are many helpful tips scattered throughout the app. When filling in the diaries, you are given advice on what each section means and how it should be filled out. Along with this, for every type of anxiety that is available, you can revise lots of information regarding it and show you signs of the anxiety if you feel you could be experiencing it. These signs are: Body – what we feel physically and emotionally, Mind – what we</p>

	<p>think and Behaviours – what we do. Then any tips will help the user cope with their emotions.</p> <p>Users can make goals for themselves and schedule a completion date. For example, complete the thought journal 3 times this week, then set the date to Sunday that week.</p> <p>The centre button within the app is a ‘Quick Relief’ button which includes: Take a breath – a 2 minute guided meditation, Shift your thinking – small reminders to ease your mind, Ground yourself – little exercises to help if you’re currently experiencing a heightened attack, Take a small step – little tips to take the next step towards feeling better and Get help – a list of points that can help if you’re in a crisis and contact information. This app is designed for users living in the US and Canada, so the contacts are not useful for anyone living outside of this area.</p>
<b>My Experiences:</b>	<p>You can use the ‘check in’ feature which lets you explain what’s going on and rating how anxious you’re feeling. I feel this feature is a great addition as it is quick and easy to fill out with the added bonus of being able to track your mood throughout the day, week, month etc.</p> <p>It also contains general information about different types of anxiety by using the ‘My Anxiety’ section. In this, you get the introduction to that type, the signs revolving around your body, mind and behaviours and any tips to help cope. This is useful for all types of users. Ranging from people diagnosed with anxiety to people feeling they are experiencing a form of anxiety – all the information you need is within the app, broken up into manageable chunks to not lose interest.</p> <p>Being able to add your own goals from within the app makes it more personal to the user. You are told helpful tips and suggestions as to what goals you should aspire to achieve, but also given encouragement if you have failed to do it.</p> <p>Due to the app being free, all features are available right from the start. This is very appealing as there are lots of useful features within the app and there is no worry of any hidden fees or advertisements ruining your experience.</p>
<b>My Rating:</b>	4/5
<b>App Store Rating:</b>	4.6/5

<b>Name:</b>	Momento – Diary / Journal
<b>Author:</b>	d3i Ltd
<b>Link:</b>	Available via: <a href="https://apps.apple.com/us/app/momento-diary-journal/id980592846">https://apps.apple.com/us/app/momento-diary-journal/id980592846</a>
<b>Description:</b>	<p>A smart private journal which is used to capture and collect memories to explore, relive and share your life story.</p> <p>The app is free but offers a premium option for £2.49 a month, £14.99 a year or for a gold subscription, £34.99 for a year. With the standard subscription, you will receive: Export – Export data into plaintext, Unlimited Feeds – Connect as many feeds as possible, Hourly Feed</p>

	Updates – Feeds updated hourly, Lock, Multiple Photos, Themes and Formatting. With the gold subscription, support will be prioritised to you.
<b>Stand out Features:</b>	<p>It offers first time users a quick tour around the app, stating how the app works and what different sections can offer.</p> <p>Everything stored is either on your personal device or iCloud</p> <p>The application can be synced with other social media to automatically import any activities, photos, videos etc to the Momento app.</p> <p>You can select a date via the calendar icon and add a new 'moment'.</p> <p>Within this, you can write a note, add a place, people, tags and photos. If you allow the app to receive data about your location, this can be added automatically granted you are where the 'moment' happened.</p> <p>The 'Explore' page allows you to search within people, places, and tags that you have entered into the app via a 'moment'</p> <p>Can group memories together using the 'Events' feature, over a specific time period. This is useful for grouping holidays and other such events together.</p>
<b>My Experiences:</b>	<p>I found it was extremely easy and obvious what each button would achieve. The layout of the app was very simple but effective in terms of the interaction. This makes it much more enjoyable to use and visibly appealing to see your data in a well-structured manner.</p> <p>The search feature is useful for trying to find the required moments within your timeline. This could come in handy to check progress and compare similar situations.</p> <p>In the task bar, there are 4 easy to see and read icons with a title. These are the main pages you interact with within the application. In the middle of this bar is a noticeably large plus for adding a new moment. This makes it obvious to the user that it's the main function of the app.</p> <p>The feeds page works with Facebook, Instagram, Twitter, and many other social media apps. The purpose of this page is to link your account with the Momento app, which then automatically pulls information from your socials and straight into the app. With this new information, you can add it as a new moment within your diary. The only problem is you can only add 1 social media account using the free version of the app. With the premium version, you can add as many as you like. I think this is a fair any reasonable offer within the app as it is an uncommon feature which will appeal to most, if not all its users.</p>
<b>My Rating:</b>	4.5/5
<b>App Store Rating:</b>	4.2/5

<b>Name:</b>	What's Up? - A Mental Health App
<b>Author:</b>	Jackson Tempra
<b>Link:</b>	Available via: <a href="https://apps.apple.com/us/app/whats-up-a-mental-health-app/id968251160">https://apps.apple.com/us/app/whats-up-a-mental-health-app/id968251160</a>
<b>Description:</b>	This application uses methods from cognitive behavioural therapy and acceptance commitment therapy to help the user cope with mental

	health issues relating to these methods, such as anxiety and depression. This is a free app which accepts donations to maintain the functionality of the app. In exchange, you receive more customizability with the colours you're able to use.
<b>Stand out Features:</b>	<p>A large list of: Negative thinking patterns and ways to overcome them, metaphors to help cope with any negative feelings, different ways to manage worries and positive steps to take to start feeling better. You can protect your data using a passcode containing numbers and letters.</p> <p>You are able to synchronise your data onto other devices and back it up, so it's never lost.</p> <p>Taking part in a grounding game which has over 100 questions designed to take your mind off of the stress and anxiety you may be feeling.</p> <p>The ability to talk to other users by only using your nickname, so pretty much anonymous communication. This is via a built-in forum.</p>
<b>My Experiences:</b>	<p>The layout of the application stands out right away. Compared to the others, it is significantly less appealing to navigate around. That being said, it is still clear what each section of the app does and what it is trying to achieve.</p> <p>The app offers very customisable themes within the free version and even more once upgrading to the premium version. Being able to customise the colours used helps make the application feel more personal to the user, as well as not having to look at the same thing day in day out if you were to use this regularly.</p> <p>Having a vast amount of information from within the app makes the user not have to go scrolling through the internet to find the information they need – more than likely it will be on the app. This improves app retention time.</p> <p>The one thing that makes this app stand out over the others is their built-in forum. Just from the small time I spent looking through, there are many cases of users reporting such sensitive information, including self-harm and attempted suicide. The community within the app is strong and supportive. The comments on the posts involving those in tough periods of their lives are very loving and have a sense of comradery – very rare online.</p>
<b>My Rating:</b>	4/5
<b>App Store Rating:</b>	4.5/5

## Existing Solutions Conclusions

From the analysis of each application, they all follow a similar pattern in terms of the vocabulary used. The applications never try to overcomplicate any explanations of functionality and provide help on the more complex areas. These two points are important to keep in mind when designing my application to be able to cater for a wider audience and not deter anyone from using the app due to negligence of usability.

Another useful point to take away from these solutions is the colour schemes used. Most of the CBT focussed apps 'Mindshift' and 'What's Up?' use light, calming blue and white driven designs. This makes the application more appealing to use and welcoming to the user.

I then concluded that the 'CBT - Thought Diary App' contained the most relevant and fruitful source of information. Due to this, I decided to carry out a more extensive review of the important function, adding a new diary entry, to get an understanding of what I feel I should craft my application around and other features that are best left out. This can be found in Appendix [1]: CBT Thought Diary App Heuristic Evaluation - Adding a new entry. These conclusions will be used to help structure the functional requirements of my application, which will be discussed in more detail later in the report.

## Existing Solutions State Transition Networks

To gauge a feeling for how an app navigates through its different processes, I decided to also use the 'CBT – Thought Diary App' to create state transition networks. A state transition network, or state transition diagram, takes a set of data and displays the flow from different data points. [4]

I will be breaking down the diagrams into three sections:

- Total Main States
- Total Secondary States
- Total Operations.

Total main states are indicated by the rounded boxes and large hollow arrows. These are the screens which the user has to visit in order to complete the process.

Total secondary states are indicated by labels square boxes. These are either alerts presented to the user or tasks the user can choose to add to the form, for example, adding extra emotions.

Total operations are indicated by the numbered square boxes. The numbers are what state that operation would take the user to if they were to select to proceed.

By totalling these values, it is clearer for developers to see the usability provided to a user from what is seemingly a simple operation being performed. From the figures provided, it is important to take away how free the user is to choose what action to perform next. This way, they do not feel trapped on a screen and left dissatisfied with the process.



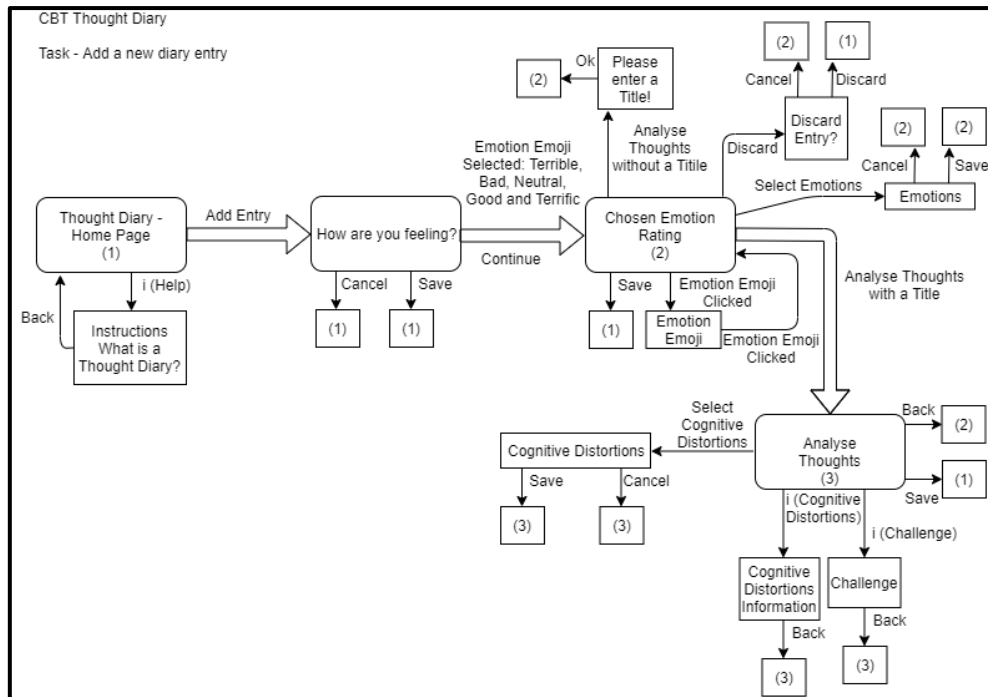


Figure 1 - CBT Thought Diary Application: Add a new diary entry

Total Main States: 4  
Total Secondary States: 8  
Total Operations: 14

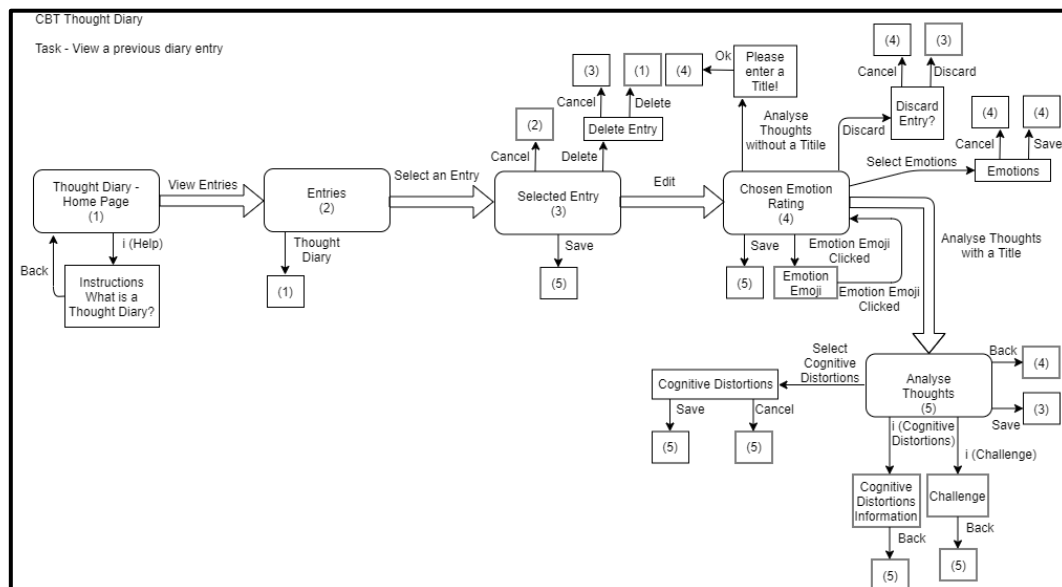


Figure 2 - CBT Thought Diary Application: View a previous diary entry.

Total Main States: 4  
Total Secondary States: 9  
Total Operations: 17

## STN Diagram Conclusions

As you can see from these mock-up STN's of the 'CBT Thought Diary Application' and the supporting figures for total number of states and operations, implementing functions can appear overwhelming for developers – more than one would think for such a seemingly simple aspect of the application. The options you have from even a single screen, for example 'Cancel' or 'Discard', can seem very complex, so it is important to have a clear pathway when first designing a mobile application. However, without these additions, the user would sometimes be left stuck on a certain screen – reducing the overall usability of the app. The number of different main states, indicated by the large arrows and rounded boxes, only take you to a maximum of five different states – with a few temporary screens presented when selecting an emotion, for example. If the application were to overuse the number of states needed to visit, the effectiveness of the important functionalities listed above would be drastically decreased. This is because of the time taken to complete certain actions, such as adding a new diary entry, within the application would be higher, leading to users potentially becoming impatient and not wanting to use the application.

Additionally, different routes to navigate the app are freely available for the user on each main screen. This will make the application more fluid. The users may be at a heightened emotional level, so catering to their current mental state is important for the app's likability.

## Therapist Diary Research

Near the beginning of the project, I was introduced to Dr Cheryl Jones, who is a private therapist treating CBT patients over video or voice calls. Dr Jones was kind enough to provide both myself and Dr Abdelmoty with important information regarding current therapy protocol that I have been using throughout the project. Within this information contained the current diaries she uses for her patients and any others available diaries that are in circulation. These diaries are the activity diary, anger thought record, health anxiety thought record, obsessive compulsive disorder diary, social anxiety diary, thought record sheet and worry diary. From each of these, I extracted the question titles and labelled the name of the data, data type, value expected and range of information that the patient is normally requested to fill out. A full list is available in Appendix [2]: Example Diaries - Questions and Variables.docx.

I discovered that the two most valuable locations for information came from the Situation and Trigger and Emotion and Rating sections. From these sections, I extracted this information:

Questions Asked from Diaries	Name of the data	Data Type	Value	Range (If Applicable)
<b>Situation and Trigger</b>				
Where?	Place	Location	Place name and/or Place Coordinates	

When?	Time	Moment in time	Date (dd/mm/yyyy) or Date and Time (dd/mm/yyyy hh:mm:ss)	An amount of time either in seconds, minutes and/or hours if it's time; Days or even weeks in terms of date
Who with?	Person	A Person or list of people  Plaintext	Persons Name or title	
<b>Emotion and Intensity of emotion</b>				
What emotion did I feel at that time?	Emotion	Plaintext or a list/wheel of emotions	List	List dependent on their initial emotion
What else?	Emotion	Plaintext or a more extensive list of emotions	List	List dependent on their initial emotion
How intense was it?	Percentage	Integer	xxx%	0 – 100 %

By extracting this information, I was better able to determine which questions, asked of the user, would be more complex to apply to the application - from a designing and developing point of view. For the other questions in the document, the input required from the user is just a simple plaintext and doesn't need to be developed any further.

Researching these points early on in the design process showcases essential aspects my application requires, making me better equipped to plan a user-friendly design. An example of this is providing the user with a simple calendar displayed when attempting to enter a date and time. By discovering these design ideas early on in development, I can then improve the efficiency of the implementation stage. Although all the questions from different diaries will be considered, the main and most relevant diary to use for the research of my project is the worry diary. The worry diary takes inspiration from the cross-sectional formulation, more commonly known as the 'Hot Cross Bun' diagram, which is shown below.

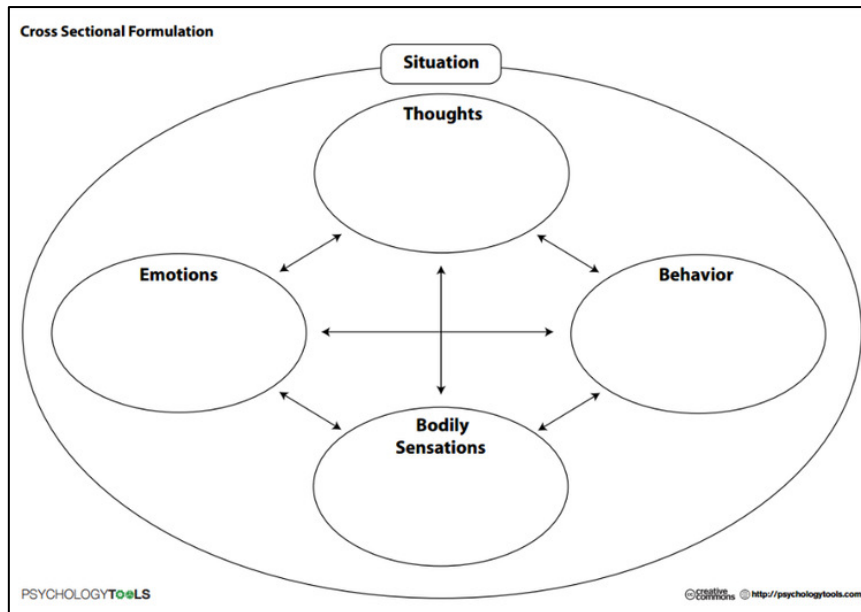


Figure 3 - 'Hot Cross Bun' found at <http://psychologytools.com/cross-sectional-formulation.html>

Using this diagram, the worry diary extracts information from each of the ovals to create an overall picture of what the patient went through and how they were feeling/what they were thinking. The diary focuses on everything experienced within a certain situation. To extract this information, certain questions need to be asked from within the diary.

Situation:

- Who were you with during the situation?
- What happened during the situation?
- When did this situation happen?
- Where did this situation happen?

Emotions:

- Emotion felt during the situation
- Rating of emotion out of 100. 0 being not at all, whilst 100 being the maximum feeling.

Worry:

- What went through your mind during the situation?
- Type of worry?
  - Hypothetical
  - Practical

Prediction:

- What is the worry predicting will happen?

For the application, I will use these questions to help structure the format of the input via the mobile screen. This is the simplest way to extract the most information without overloading the patient or making the diary a hassle to fill out.

## Personas

To be able to create a user-focused design application, it is important to consider the average user's needs - personas are an integral part of this. By creating personas, it allows you to think about different aspects of a design you may not think of when only considering your own personal experience of the application. Some users may be technically capable but struggle with other aspects my app offers, like the contents of a diary entry.

I will base these personas on the typical user of this application with the focus being on therapist patients and mental disorders.

### Primary Persona

**Name:** Owen Smith

**Age:** 19



Image source: [https://unsplash.com/photos/OK7VpKfbb\\_c](https://unsplash.com/photos/OK7VpKfbb_c)

**Quote:** "I need a place I can document all of my thoughts and feelings in the moment to reflect on them later"

**Description:** Owen is a full-time student that has been suffering with anxiety since he was 14. He has a diary at home and has been using it from time to time to roughly document some situations, which helps him understand how he could have responded differently. He is technically proficient and owns a smartphone which is kept on him throughout the day. He is looking to find an application where he can start documenting situations where he feels his anxiety is taking over more frequently and as they occur.

#### Goals:

- Increase the frequency of diary entries to help control his anxiety more
- Gain access to all previous entries to see any progress when dealing with similar situations
- Note down where each event occurred to see if there is a trend that triggers his anxiety

## Secondary Persona

**Name:** Samantha Jones

**Age:** 34



<https://unsplash.com/photos/6CgkUjUl4og>

**Quote:** “I’m very forgetful and would like an app I can use on the go to document my emotions throughout the day”

**Description:** Samantha is a busy social worker that goes through a lot of stress during the day. She discovered a method of dealing with stress is to document how you are feeling into a diary to get your thoughts out without having to rely on someone 24/7. She isn’t very technical and would benefit from an easy to use application with simple steps to follow in order to use it.

### Goals:

- Quickly note down emotions without having to have a triggering event relating to it
- Easy to use application that doesn’t overcomplicate a simple diary entry based off of emotions and thoughts

## Persona Conclusions

When considering both of the personas together, it’s evident that a simple, easy to use application suits all parties. This involves making the format for a new diary entry to be fluid from beginning to end, displaying to the user exactly what will be entered in the final entry. Therefore, the app needs to have clear progression on how to fully complete an entry without the user having to guess the next step or figure it out through trial and error.

It is also clear that all entries made need to be easy to view to allow for reflection. These entries need to be accessible 24/7 and clearly laid out, implying a constant local or cloud database should be used to accommodate this. The users would also benefit from being able to search for particular entries based on the information placed within them. For instance, the location or title of the entry. Along with a search feature, being able to sort the entries would be a good function for the app to have i.e. in order of date of entry, similar emotions grouped together or even the same locations.

## Project Constraints

Taking on a project with this magnitude will be the most difficult challenge so far in my entire academic history. It is important to not think of it as one large project, instead as little manageable chunks to aim towards completing bit by bit. There is a lot of software and documentation I have never used before, so planning enough time to research and learn this for the project is crucial. Having a strict plan set out beforehand will give me enough time to complete the main functionalities I planned to complete within the timeframe given.

An unexpected constraint has been the unfortunate global pandemic of COVID-19, better known as Coronavirus. This has hindered me both physically and emotionally. First off, it has made it impossible to perform my planned user testing where I wanted to get a population of around ten random people to take part in some simple tasks to perform on my application. I would then have made notes on the speed of completion and any complications they came across. Instead, I am going to perform these tests on the tenants I am currently residing with in Cardiff, which has significantly lowered the diversity of the users testing my app.

Emotionally, this has been a very tough time for me. Along with family complications unrelated to the virus, the Coronavirus has prevented me from seeing any of my family which led to a drop in my mood, motivation and overall mental health. This set me back a few weeks, but I managed to catch up on some work and hope to produce a final application and report I can look back and be proud of.

## Approach

### Planning

As previously mentioned, I decided to adopt an Agile design and development methodology. Normally, this methodology is for team management by breaking a project up into different stages, building onto and improving the software at each stage. Since this is an individual project, I had to adapt this methodology slightly to suit my situation better. To achieve this, I set out a week by week work plan stated in my initial report. This broke down my project into different chunks of achievable tasks which is good for gaining a sense of accomplishment when a certain task is completed, rather than feeling overwhelmed by the workload.

Using this work plan provided me with enough time to learn the basics of all the software I'm going to use and allowed me to progressively write my report so as not to leave anything in a rush at the end. As I'm still learning the software, concurrently I will be building my application, applying any knowledge learnt. Any more advanced functionalities will be added only if time permits. By doing this, I'm allowing myself to build a fully functioning application before trying to make it too complex and not grasping the core ideals of the app.

Along with a work plan, frequent communication was needed to be set up with the therapist, Dr Cheryl Jones, to get consistent feedback on the progression of the app. This is vital as Dr Jones is the closest connection to the interaction patients are currently dealing with in terms of diary entries and their success. Through Dr Jones, I collated a collection of both blank and example diaries to really get a grasp on the thoughts and feelings patients encounter every day and how my app will help them in their journey. This is very useful for organising where to place the different user inputs to match where patients input the most detail and where the most useful information is located within a normal entry.

### Learning

This project required me to take on challenges I had never experienced before. From learning about the design approaches for applications, to learning new JavaScript techniques and libraries. This was a vast amount of software development knowledge I hadn't previously touched upon in my life. Here are the main learning points I had to take on.

### Developer Guides

The developer guides for both of the main systems I am developing for are extremely important to consider when designing a mobile application. These contain information regarding every aspect of user interface design which I will use to create my user interface. These guides contain all the information needed to create the most coherent layout possible for an application. From the best colour schemes to use for certain interactions,



such as buttons, to form layouts and icons to use. By researching these integral details thoroughly, it gives me the best chance to create a design people will not only find easy to follow but more importantly an app people will want to use.

## Data Storage

This application relies on being able to store lots of information safely and securely. As the data that will be stored is very sensitive, data security is of the utmost importance. Along with security, a rich supply of documentation is required as my lack of experience in this field may hinder me later on in the application.

I have decided to use MongoDB to store my information. MongoDB is a NoSQL database which utilises JSON styled documents to structure the documents within the database. [5] MongoDB Atlas is a cloud database as a service which allows you to deploy MongoDB to the cloud for easier data retrieval. [6] I will be using this to make it easier to view information for testing.

To access this database, I will be using Nodejs to design routes to talk to the server. A route is how an application responds to a client's request. [7] For example, if the front end were to request to view all information from a certain user, a GET request could be set up in order to perform this request, retrieving the specified data from the database. will be set up using Mongoose. Mongoose is used to manage relationships between data, schema validation and translate the JSON objects from the objects in the code to the objects in MongoDB and vice versa. [8]

## User Interface Design

For user interface design, I will be using the software Axure. This software is for designing prototypes of many different applications, ranging from web to mobile to desktop. [9] The software allows you to shape a canvas into the size of any phone's dimensions currently available. This is useful to be able to get an idea of how your design will look on the majority of phones currently available.

Then, by using the available shapes and user input interactions, such as buttons and text fields, you are able to replicate an accurate skeleton of your desired application. Using this allows me to see how my app could look once implementation begins, and how the entry form flows from beginning to end – arguably the most important aspects of my whole application.

Along with studying the guidelines for both iOS and Android systems, I will also be heuristically evaluating my initial designs using Nielsen's 10 heuristic evaluation principles. [2] This allows me to design the best prototype possible before the development of my user interface.

## User Interface

My application's main goal is to allow patients of CBT therapy to have the convenience of a diary in the palm of their hand, to never miss an entry which could be analysed by the therapist and take a step closer towards recovery. Due to this, there needs to be no excuse not to fill in the application's form. Therefore, the app needs to be available for every available device possible. To solve this issue, I decided to use React Native. React Native is a mobile application framework which allows you to build and develop projects for Android, iOS and Web Applications. [10] The built-in native components for React, such as View and Text, are directly mapped to platform specific syntax so your code can all be in one place at a time and the software will convert it to whichever platform accesses the app.

## Specification and Design

### Functional Requirements

Must have		
Requirement Number	Requirement	Acceptance Criteria
1.	The system will allow the user to input information regarding an overview of the happenings within the situation: Who with, what happened, when and where.	<ul style="list-style-type: none"><li>The user will have a range of input formats to store this information e.g. Text and live location.</li></ul>
2.	The system will allow the user to input information regarding the worry within the situation: What if?	<ul style="list-style-type: none"><li>The user will have the chance to talk in as much or little detail as possible</li></ul>
3.	The system will allow the user to input information regarding their prediction in the situation: What is the worry predicting will happen?	<ul style="list-style-type: none"><li>The user will enter textual information regarding the prediction they feel might happen in a situation.</li></ul>
4.	The system will allow the user to input information regarding their emotions and intensity of emotions on a scale	<ul style="list-style-type: none"><li>The user will enter as many or as little emotions they were feeling at the time</li><li>This will then be rated on a mutually accepted scale</li></ul>
5.	The system will allow the user to input information regarding the type of worry: Hypothetical or practical	<ul style="list-style-type: none"><li>A hypothetical worry is regarding a situation which can't be fixed</li><li>A practical worry is about a situation where something can be done to deal with them</li></ul>
6.	The system will allow the user to enter the date of the situation	<ul style="list-style-type: none"><li>The default date when producing a new entry is the current date</li><li>The user can change the date to any day prior to the current day</li><li>Alongside the date of the situation, the date of the entry is also recorded and displayed</li></ul>
7.	The system will be able to track the users live location when using the application	<ul style="list-style-type: none"><li>When the user opens the application, their current location is available to them.</li><li>During the addition of a new entry, the patient could tag their current location to help identify any links to their triggers</li></ul>
8.	The system will highlight areas that aren't complete during each section of the entry	<ul style="list-style-type: none"><li>The user will be prompted that an area is incomplete and can decide whether they want to continue or cancel and fill that section in</li></ul>

9.	The system will require each user (patient and therapist) to set up a password protected profile	<ul style="list-style-type: none"> <li>The system will block a user from access to any content without the correct security credentials</li> <li>Profiles will be used to link the patient to the corresponding therapist, preventing unauthorised access to sensitive data</li> </ul>
10.	The user can view and edit previous entries	<ul style="list-style-type: none"> <li>The user can select a day on the calendar where all information of the entry will be displayed: Time of entry, text contents, emotion rating and any photographs attached.</li> <li>They will also have the option to edit any of the information.</li> </ul>
11.	The user can choose to publish certain entries	<ul style="list-style-type: none"> <li>The user will do a diary entry as normal and have the choice to make it accessible to the therapist to access</li> <li>To publish, make sure the entry is complete and select the 'Share' button</li> <li>The user will be notified if they wish to share the entry and when it has been published</li> <li>The user will be able to see which previous entries have been published via the calendar view</li> </ul>
12.	Via the therapist view, the system will be able to extract specific patient data	<ul style="list-style-type: none"> <li>The therapist will only be able to extract patient data they have the access to – other therapist's patients will not be available</li> <li>The therapist will navigate to their patient's page and will be shown days where at least one entry has been made. From there, they will be able to extract data for that day</li> </ul>
13.	The user will be able to add photographs to an entry	<ul style="list-style-type: none"> <li>The user can choose to add photographs straight from their camera or the camera roll already existing on the device</li> </ul>
<b>Should have</b>		
14	The system should allow the user to sort entries	<ul style="list-style-type: none"> <li>Each user should have the option to sort the entries in a range of ways including date</li> </ul>
15.	The system should allow the user to perform a text search	<ul style="list-style-type: none"> <li>The patient or therapist should enter a specific word – either a situation the patient has been in or an object etc. This allows the user to find a correlation for certain words</li> </ul>
<b>Could have</b>		

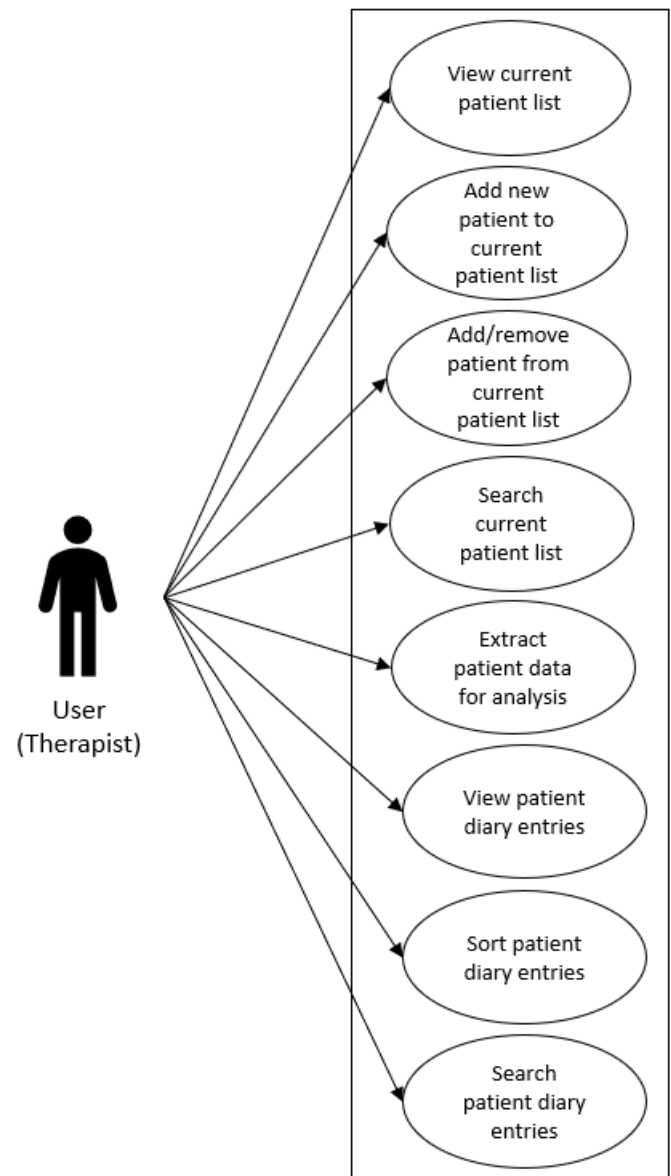
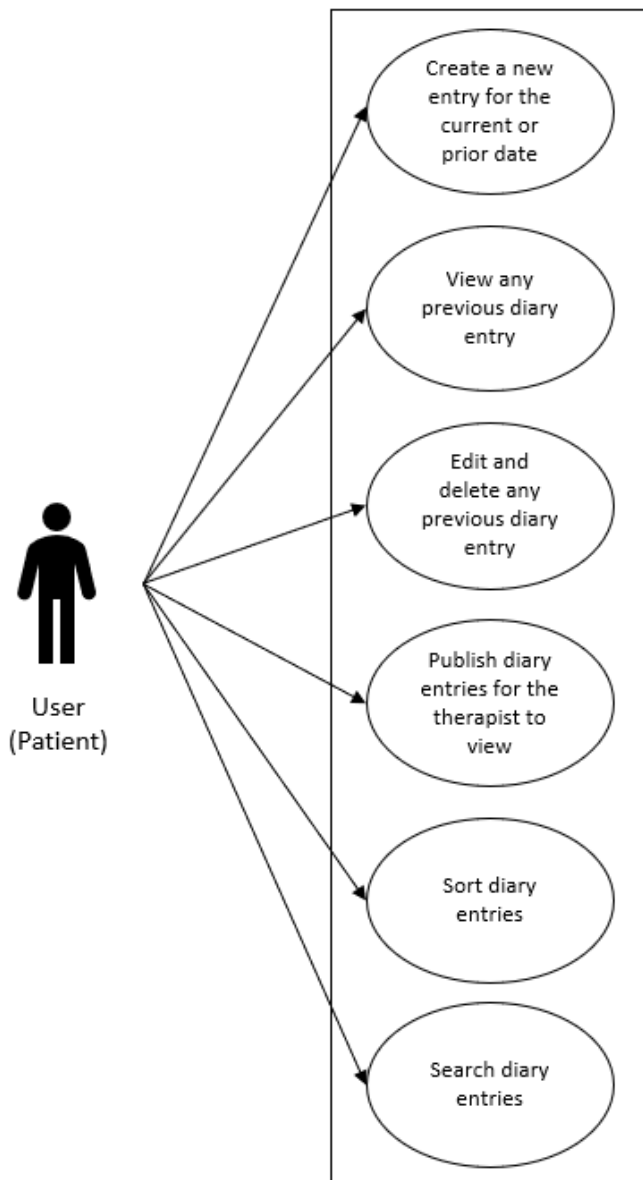
16.	The system could remind the user if they haven't yet filled out for that day	<ul style="list-style-type: none"> <li>The system could send a notification to the user's phone near the end of the day if no entry has been made yet</li> </ul>
<b>Will not</b>		
17.	The system will not allow you to view any other user's diary entries	<ul style="list-style-type: none"> <li>Either within the patient view or therapist view, no unauthorised access to other user's sensitive data will now be available</li> </ul>

## Non-Functional Requirements

<b>Must have</b>		
Requirement Number	Requirement	Acceptance Criteria
1.	The systems database is updated when a new entry is added	<ul style="list-style-type: none"> <li>Regardless if the user has decided to publish the entry for the therapist to see, the database will update and store the new entry</li> </ul>
2.	All entries, published or not, are always accessible to the user	<ul style="list-style-type: none"> <li>Via the calendar view, users are able to see all previous entries as it will be stored locally</li> <li>Therapists will be able to view all published entries</li> </ul>
3.	The application will provide help for each screen	<ul style="list-style-type: none"> <li>Support provided to the user to help navigate through the UI</li> </ul>
4.	The application will be easy to navigate	<ul style="list-style-type: none"> <li>Following heuristic evaluation, the usability of the UI will be reviewed</li> </ul>
<b>Should have</b>		
5.	The response time of the application should be a suitable standard	<ul style="list-style-type: none"> <li>This is mainly focussed on the publishing of the entries to the database as it is expected to be the longest process within the application</li> <li>Elsewhere, the application should appear to be seamless when performing actions</li> </ul>
<b>Could have</b>		
6.	The user could change the theme of the application	<ul style="list-style-type: none"> <li>Allowing the user to change the colour style of the application to their liking</li> <li>The colour change would make it more personal to the user and could make it easier for them to interact with certain features</li> </ul>

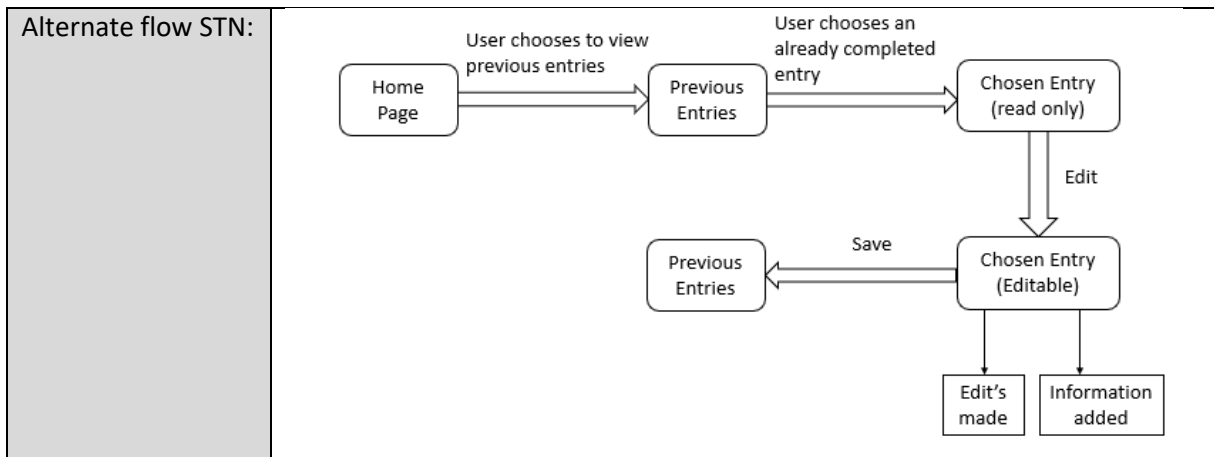
## Use Cases

The following diagram's display all the main tasks the user or therapist will be able to undertake. Beneath them are the corresponding use case tables which explore the functionality further.



## Patient Use Case's

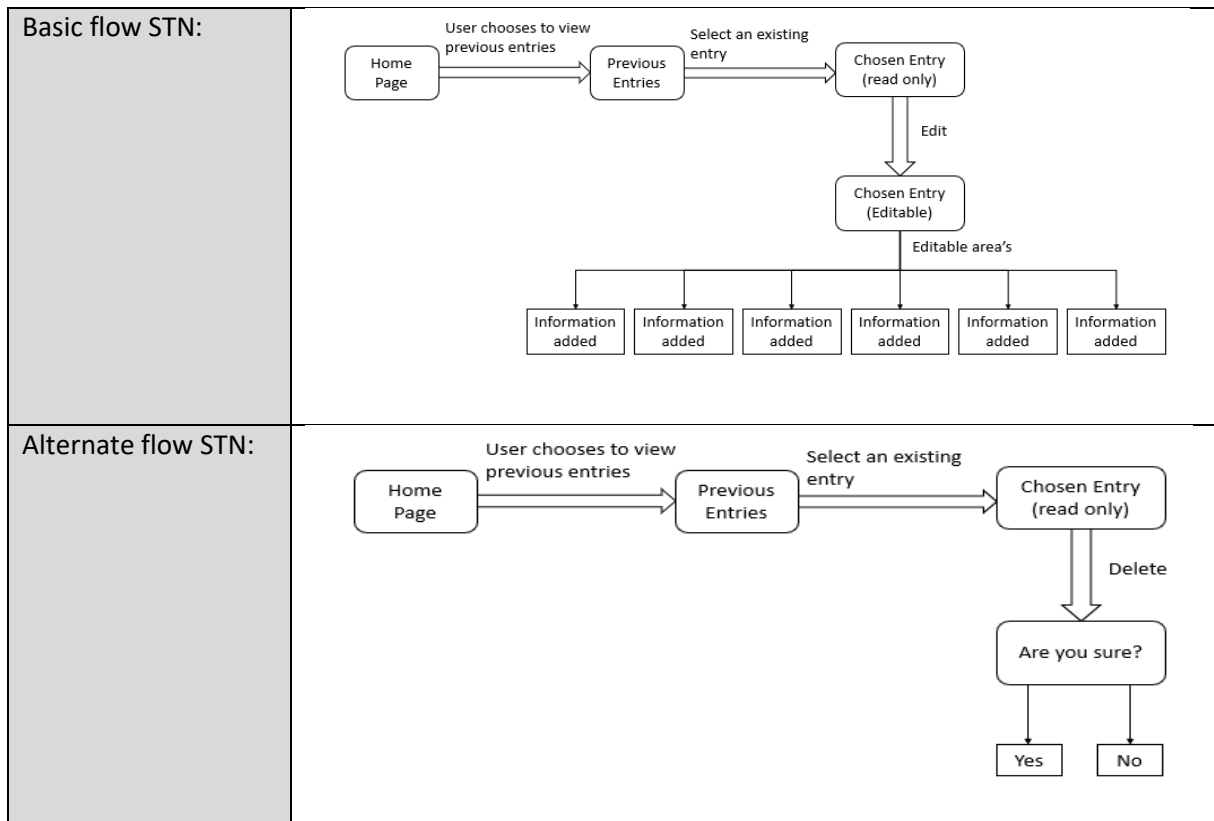
Use Case Name:	Create a new entry
Description:	The user can create a new diary entry for today's date or any date prior.
Pre-condition(s):	The date selected is not a future date.
Basic flow:	<ol style="list-style-type: none"> <li>1. The user opens the application on their preferred device</li> <li>2. The user logs onto their secure account</li> <li>3. The user chooses to create a new diary entry</li> <li>4. The user inputs information regarding the situation <ol style="list-style-type: none"> <li>a. Who</li> <li>b. What</li> <li>c. When</li> <li>d. Where</li> </ol> </li> <li>5. The user inputs information regarding the worry</li> <li>6. The user inputs information regarding their prediction</li> <li>7. The user inputs information regarding their emotions <ol style="list-style-type: none"> <li>a. User selects from a list</li> </ol> </li> <li>8. The user rates their emotion intensity <ol style="list-style-type: none"> <li>a. User rates their emotion using a percentage</li> </ol> </li> <li>9. The user inputs information regarding the type of worry experienced <ol style="list-style-type: none"> <li>a. Hypothetical or practical</li> </ol> </li> <li>10. The user can choose to add photographs to their entry</li> <li>11. The user indicates they are done, and the entry is saved</li> </ol>
Alternate flow:	<ol style="list-style-type: none"> <li>1. The user opens the application on their preferred device</li> <li>2. The user chooses to view previous entries</li> <li>3. The user selects to edit a previous entry</li> <li>4. The user adds new information to the entry; hence a new entry is made.</li> </ol>
Basic flow STN:	<pre> graph TD     LP[Login Page] -- "User securely logs onto account" --&gt; HP[Home page]     HP -- "User chooses to create a new diary entry" --&gt; S[Situation]     S -- "User inputs information regarding the situation" --&gt; W[Worry]     W -- "User inputs information regarding the worry" --&gt; P[Prediction]     P -- "User inputs information regarding the prediction" --&gt; E[Emotions]     E -- "Emotion and rating chosen" --&gt; TW[Type of worry]     TW -- "Type of worry chosen" --&gt; FR[Final Review]     FR -- "Option to attach a relevant photograph" --&gt; FR     FR -- "Save" --&gt; EA[Entry Added]     S --&gt; Who[Who]     S --&gt; What[What]     S --&gt; When[When]     S --&gt; Where[Where]     E --&gt; SE[Select an emotion]     E --&gt; RE[Rate the emotion]     TW --&gt; HOP[Hypothetical or Practical] </pre>



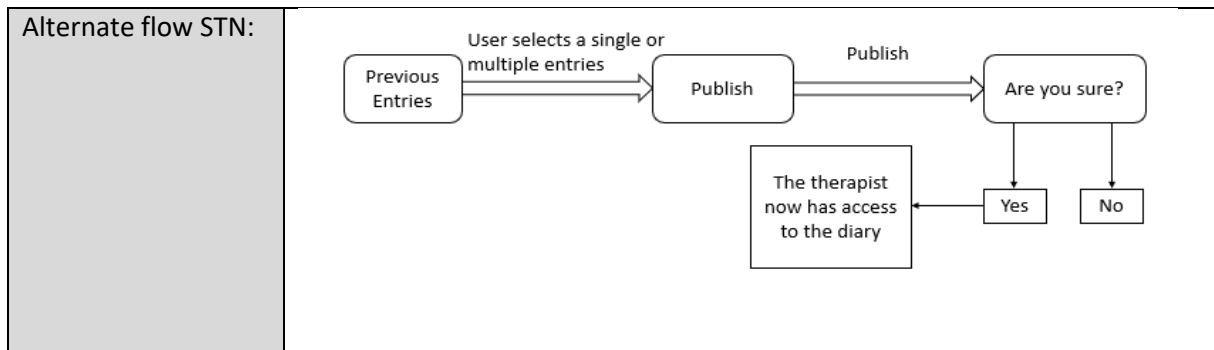
Use Case Name:	View previous entries
Description:	The user can visit their previous entries and view them.
Pre-condition(s):	An entry has previously been entered.
Basic flow:	<ol style="list-style-type: none"> <li>1. The user chooses to view previous entries</li> <li>2. The user selects their desired entry - the information contained within this entry will then be displayed to them</li> </ol>
Basic flow STN:	<pre> graph LR     HP[Home Page] -- "User chooses to view previous entries" --&gt; PE[Previous Entries]     PE -- "Select an existing entry" --&gt; CE[Chosen Entry (read only)]     CE --&gt; CV[Can view all information regarding entry] </pre>

Use Case Name:	Edit previous entries
Description:	The user will have an option to edit information from every section contained within the application. The user will also have the option to delete the whole entry.
Pre-condition(s):	An entry has previously been entered that can then be edited.
Basic flow:	<ul style="list-style-type: none"> <li>• The user chooses to view previous entries</li> <li>• The user selects their desired entry</li> <li>• The user selects to edit this entry</li> <li>• The user can then edit: <ol style="list-style-type: none"> <li>a. The situation</li> <li>b. The worry</li> <li>c. The prediction</li> <li>d. The emotion and emotion rating</li> <li>e. The type of worry</li> <li>f. The photograph</li> </ol> </li> </ul>
Alternate flow:	<ol style="list-style-type: none"> <li>1. The user can delete the whole entry</li> <li>2. The user is prompted to confirm deletion. Once this has been confirmed, this current entry will no longer exist</li> </ol>





Use Case Name:	Publish diary entries for the therapist to view
Description:	The user can publish a single entry or multiple entries for the therapist to be able to view. The therapist then uses these entries to analyse progress for the next appointment.
Pre-condition(s):	The user has finished their entry, filling it in as best as they can. The user is also happy for the therapist to view what they have written.
Basic flow:	<ol style="list-style-type: none"> <li>1. The user has just finished filling out their new diary entry and has saved it</li> <li>2. The user is then presented with their new entry where they can choose to publish the entry</li> <li>3. This then gives the therapist access to save it to their machine for analysis</li> </ol>
Alternate flow:	<ol style="list-style-type: none"> <li>1. The user navigates to the previous entries page</li> <li>2. The user selects a single entry or multiple entries they wish to share with the therapist</li> <li>3. The user chooses to publish their desired choice of entries</li> </ol>
Basic flow STN:	<pre> graph LR     NE1[New entry (Editable)] -- "Save" --&gt; NE2[New entry (Read only)]     NE2 -- "Publish" --&gt; AS[Are you sure?]     AS -- "Yes" --&gt; TA[The therapist now has access to the diary]     AS -- "No" --&gt; N[No] </pre>



Use Case Name:	Sort diary entries
Description:	The user can sort their diary entries using different sections of the form that isn't a text entry e.g. emotion rating.
Pre-condition(s):	The user has two or more entries for the sort to take effect.
Basic flow:	<ol style="list-style-type: none"> <li>The user navigates to the previous entries page</li> <li>The user selects the 'Sort' button</li> <li>The user chooses their sort by type               <ol style="list-style-type: none"> <li>Date – Newest first (default)</li> <li>Date – Oldest first</li> <li>Emotion tag chosen</li> </ol> </li> <li>The diary entries are then shown in the form of the sort by type chosen</li> </ol>
Basic flow STN:	<pre> graph LR     A[Previous Entries] -- "Sort" --&gt; B{What 'Sort By' type?}     B -- "Date - Newest first (default) Date - Oldest first Emotion tag chosen" --&gt; C[Entries sorted in desired order]           </pre>

Use Case Name:	Search diary entries
Description:	The user can perform a text search for words or phrases contained within their diary entries.
Pre-condition(s):	The user has at least one diary entry to be able to perform a search on.
Basic flow:	<ol style="list-style-type: none"> <li>The user navigates to the previous entries page</li> <li>The user selects the search option</li> <li>The user types a search term into the search bar supplied</li> <li>Matching entries are then displayed to the user</li> </ol>
Basic flow STN:	<pre> graph LR     A[Previous Entries] -- "Selects the search option" --&gt; B[Search]     B -- "Search term entered" --&gt; C[Corresponding Entries Displayed]           </pre>

## Therapist Use Case's

Use Case Name:	View current patient list
Description:	The therapist can view all patients they are currently caring for.
Pre-condition(s):	The therapist has at least one patient assigned to their care.
Basic flow:	<ol style="list-style-type: none"> <li>1. The therapist opens the application on their preferred device</li> <li>2. The therapist logs onto their secured account</li> <li>3. The therapist navigates to the current patient list found on the home page</li> <li>4. The therapist can now view every patient assigned to them</li> </ol>
Basic flow STN:	<pre> graph LR     LP(Login Page) -- "Therapist securely logs onto account" --&gt; HP(Home Page)     HP -- "Current patients" --&gt; CPL(Current Patient List)     CPL --&gt; F[The therapist can now view every patient assigned to them]           </pre>

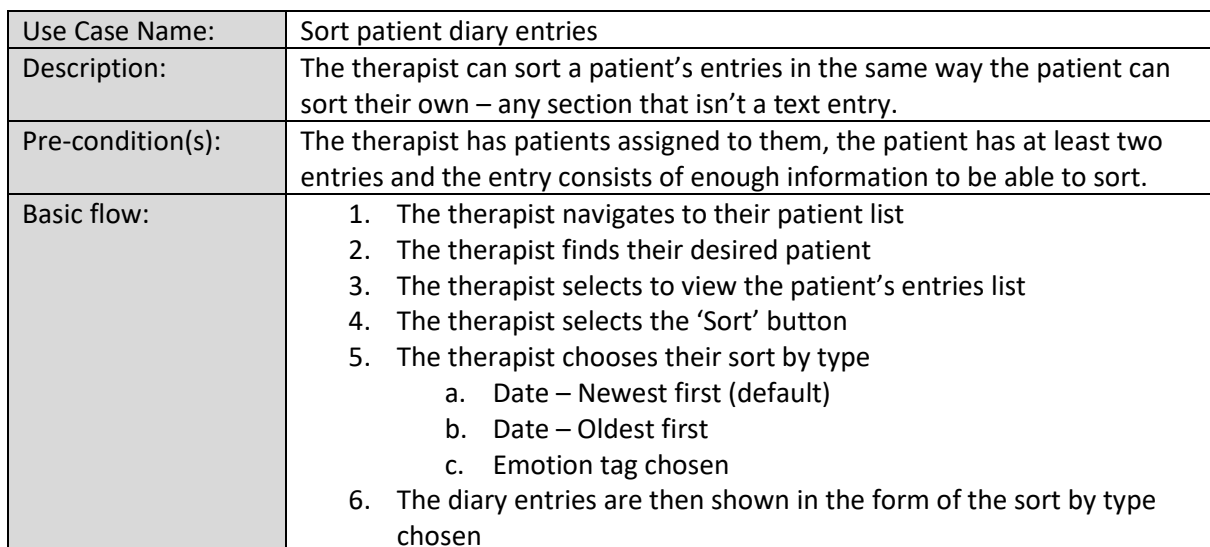
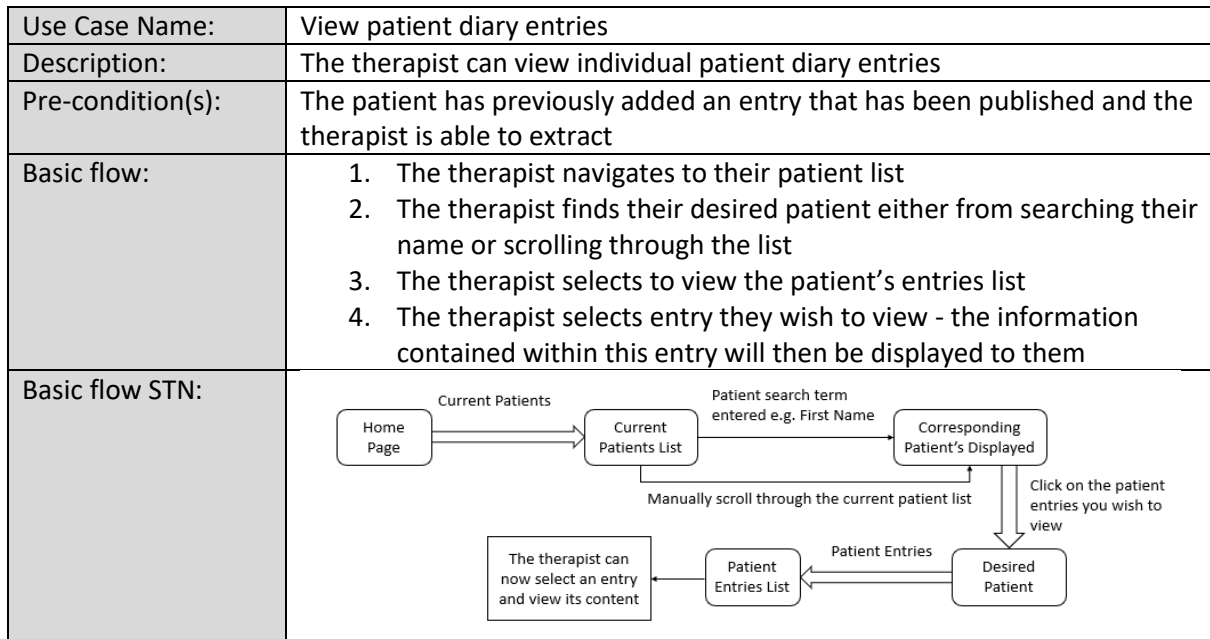
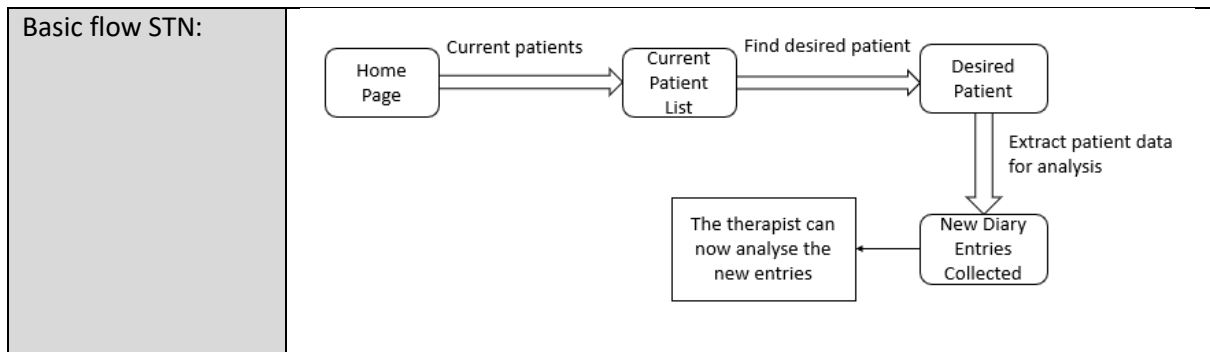
Use Case Name:	Add new patient to current patient list
Description:	The therapist can add more patients to their existing list
Pre-condition(s):	A patient is waiting to be assigned to a therapist and wants to participate with the diary application.
Basic flow:	<ol style="list-style-type: none"> <li>1. The therapist navigates to the current patient list found on the home page</li> <li>2. The therapist selects to add a new patient</li> <li>3. The therapist enters the details to add a new patient to their list</li> <li>4. The patient has now populated the therapist's list and is ready to be interacted with</li> </ol>
Basic flow STN:	<pre> graph LR     HP(Home Page) -- "Current patients" --&gt; CPL(Current Patient List)     CPL -- "Add new patient" --&gt; INPD(Insert New Patient Details)     INPD -- "Confirm new patient addition" --&gt; F[The user now populates the therapist's list]           </pre>

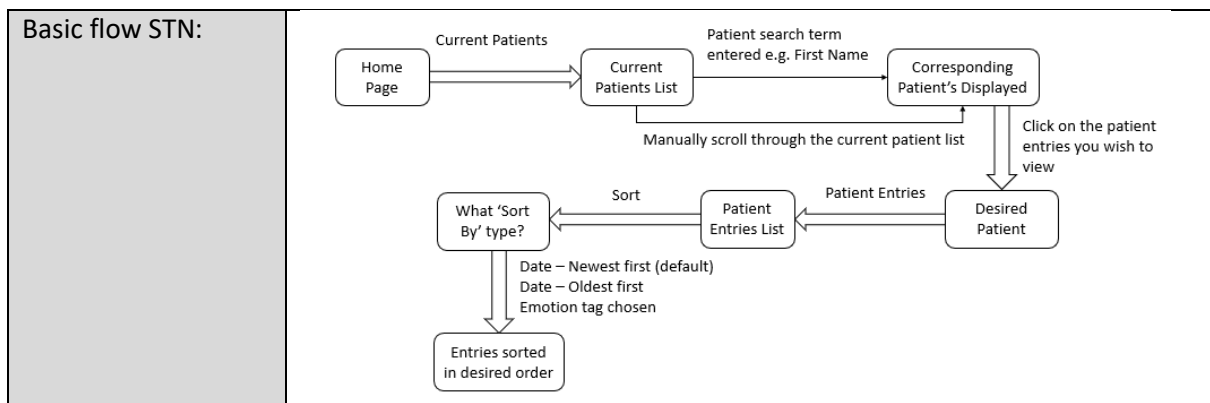
Use Case Name:	Remove patient from current patient list
Description:	The therapist can remove a patient from their current care list.
Pre-condition(s):	A patient has finished their course, decides they don't want to continue, or the therapist wants to terminate the patient's care, giving them a reason to be removed.
Basic flow:	<ol style="list-style-type: none"> <li>1. The therapist navigates to the current patient list found on the home page</li> </ol>

	<ol style="list-style-type: none"> <li>The therapist finds the desired patient to removed either by using the search function or scrolling through the list</li> <li>The therapist selects the patient and has the option to remove the patient from their list</li> <li>The patient has now been removed, along with any of their diaries saved on the therapist's computer. The patient is now available to be added to a different therapist's list if needed</li> </ol>
Basic flow STN:	<pre> graph LR     HP[Home Page] -- "Current patients" --&gt; CPL[Current Patient List]     CPL -- "Find desired patient" --&gt; DP[Desired Patient]     DP -- "Remove Patient" --&gt; CPR[Confirm Patient Removal]     CPR -- "Yes" --&gt; R[The patient has now been removed from the therapist's list] </pre>

Use Case Name:	Search current patient list
Description:	The therapist can search for a particular patient within an alphabetically ordered list
Pre-condition(s):	The therapist has at least one patient assigned to them
Basic flow:	<ol style="list-style-type: none"> <li>The therapist navigates to the current patient list found on the home page</li> <li>The therapist can choose how to find the patient <ol style="list-style-type: none"> <li>Using the search function and typing in their name to bring up corresponding patients</li> <li>Manually scrolling through the list to find the patient</li> </ol> </li> <li>If the patient exists within the list, the therapist has been able to find and retrieve this patient</li> </ol>
Basic flow STN:	<pre> graph LR     HP[Home Page] -- "Current Patients" --&gt; CPL[Current Patients List]     CPL -- "Patient search term entered e.g. First Name" --&gt; CPD[Corresponding Patients Displayed]     CPL -- "Manually scroll through the current patient list" --&gt; CPD </pre>

Use Case Name:	Extract patient data for analysis
Description:	The therapist can extract individual or multiple patient entries of which the patient has published.
Pre-condition(s):	The user is one of the therapist's patients and they have at least one diary entry.
Basic flow:	<ol style="list-style-type: none"> <li>The therapist navigates to the current patient list found on the home page</li> <li>The therapist selects the patient and chooses the synchronise patient data option</li> <li>The therapist is taken to all of the patient's entries, with the newest entries being displayed first</li> </ol>





Use Case Name:	Search patient diary entries
Description:	The therapist can perform a text search on the diary entries for patients
Pre-condition(s):	The therapist has patients assigned to them and at least one diary entry exists in their catalogue.
Basic flow:	<ol style="list-style-type: none"> <li>1. The therapist navigates to their patient list</li> <li>2. The therapist locates the desired patient</li> <li>3. The therapist selects the search option</li> <li>4. The therapist types a search term into the search bar supplied</li> <li>5. Matching entries are then displayed to the therapist</li> </ol>
Basic flow STN:	<pre> graph TD     HP[Home Page] -- Current Patients --&gt; CPL[Current Patients List]     CPL -- "Patient search term entered e.g. First Name" --&gt; CPD[Corresponding Patient's Displayed]     CPL -- "Manually scroll through the current patient list" --&gt; DP[Desired Patient]     CPD -- "Click on the patient entries you wish to view" --&gt; DP     DP -- Patient Entries --&gt; PEL[Patient Entries List]     PEL -- "Selects the search option" --&gt; S[Search]     S -- "Search term entered" --&gt; CED[Corresponding Entries Displayed] </pre>

## Prototype

As mentioned previously, I will be using Axure to create a prototype of how my application will navigate around to different screens and interact with the user to deliver the best results before I start with the implementation of the application.

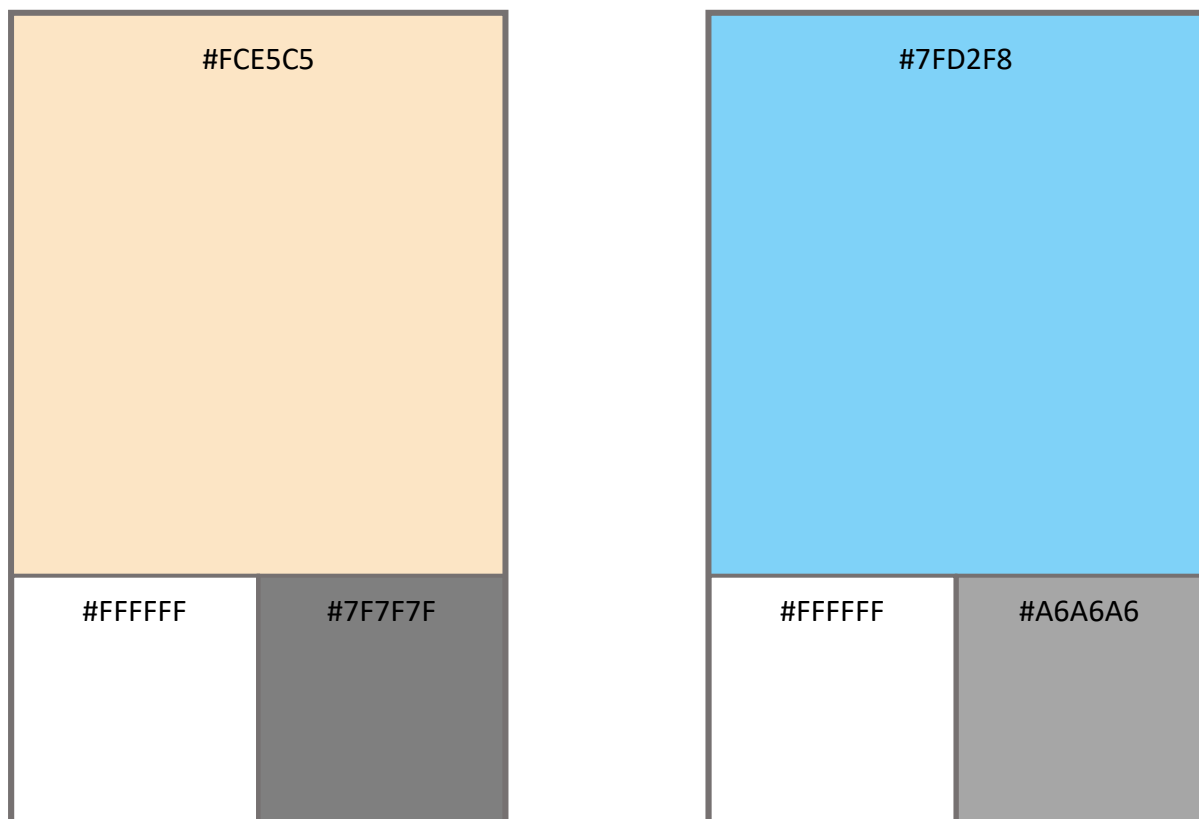
By using the use case's as the main functionality needed for my application, my prototype's goal is to achieve each use case in a simple and coherent manner. Because the use cases are using the functional requirements stripped down to the most basic form, it is a great starting point to explore different ways of displaying the information to the user.

## Font

The font chosen for this application is called Montserrat, a sans-serif font inspired by the Montserrat neighbourhood of Buenos Aires. [11] I wanted to find a font that was capable of being more interesting to look at compared to a standard font you can find on some

applications, whilst maintaining a professional look so users did not feel like the app was not catered to certain age groups.

### Colour Scheme



*Figure 4 - Initial ideas for colour schemes*

When choosing a colour scheme for my application, it's important to consider what the user will be accessing it for. They will most likely be currently in, or just coming out of a stressful situation which has triggered their disorder and made them feel a certain way. When they open the app, they should feel more relaxed and view the confinements of the application as a safe space to talk about their emotions. This is what led me to these colour schemes I created.

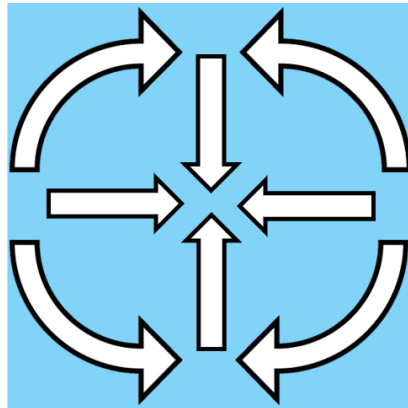
The colour scheme on the left is a light, tan colour with white and grey as the supporting colours to outline interactable objects. The colour scheme on the right is sky blue with similar supporting colours, which would also be used to highlight interactable objects. As previously discussed, whilst concluding the existing solutions, the majority of the CBT focussed applications opted for light blue and white as their primary colours - this was the inspiration behind this design.

Blue has been found to generate feelings of tranquillity and produce a calming effect, whilst tan gives off a warm, inviting feeling. [12] This means both of these would produce a positive effect on the user whilst using the application.

Although both of these would have worked for the application, I ended up opting for the sky-blue scheme. I feel as if it would have a more positive reaction as a main theme, whilst clearly indicating interactive objects using the opposing colours.

## Logo

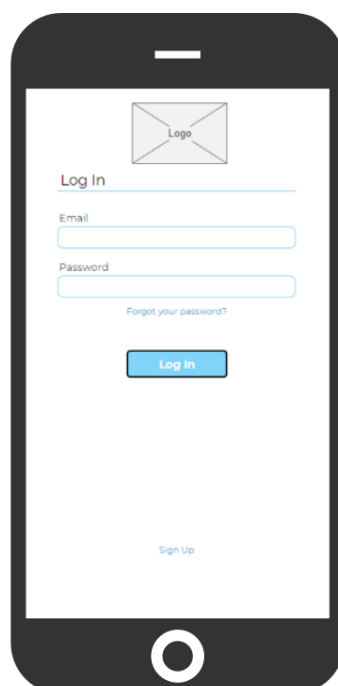
During the early stages of collecting information from Dr Jones, I was introduced to the 'Hot Cross Bun' diagram used for data collection from the patients - this can be found in the figure table as figure 3. I used this as inspiration, along with my chosen colour scheme to create the following logo:



*Figure 5 - Application Logo*

It is not 100% accurate in terms of how the 'Hot Cross Bun' diagram functions as the arrows are not all pointing in the correct direction or location. Even with this, it will be recognisable to patients of most CBT treatments. This is useful as people unfamiliar with this kind of treatment will not be able to recognise this pattern, maintaining the discreteness desirable for patients who may feel awkward or embarrassed by having this app on their phone.

## First Design



*Figure 6 - Login Page*



Screen:	Login Screen
Description:	This is the first screen the user will be presented with, so it has to be inviting. It clearly shows three options to choose from, enter your email and password and log in, forgot your password option or sign up to the application. By using this screen, the user will log into the database and have access to any previously entered entries, enter new entries to be stored and publish entries for their therapist to see.

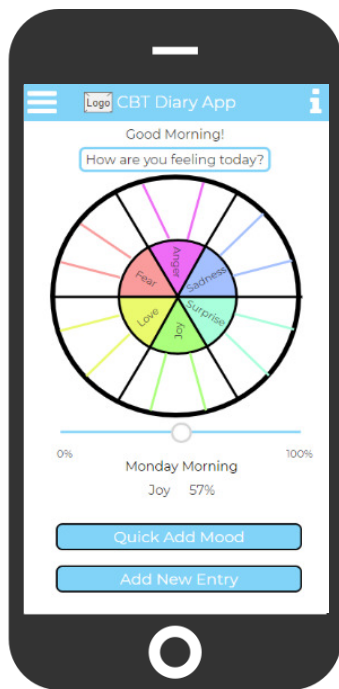


Figure 7 - Home Page

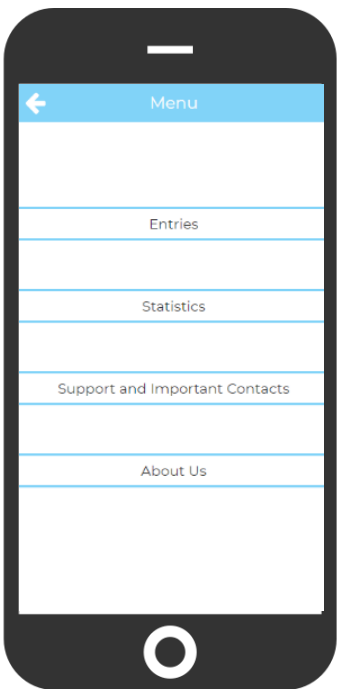


Figure 8 - Menu Page

Screen:	Home and Menu Screen
Description:	The screenshot on the left is the home screen. It contains two methods to enter a new entry. You can either use the emotions wheel to select how you're currently feeling and press the 'Quick Add Mood' button to add that mood to your entries list or click 'Add New Entry' to begin a whole new entry. The three lines in the top left bring up the menu which is shown on the right. From here, you have a wide range of options to choose from – the most important being the option to view all your previous entries.

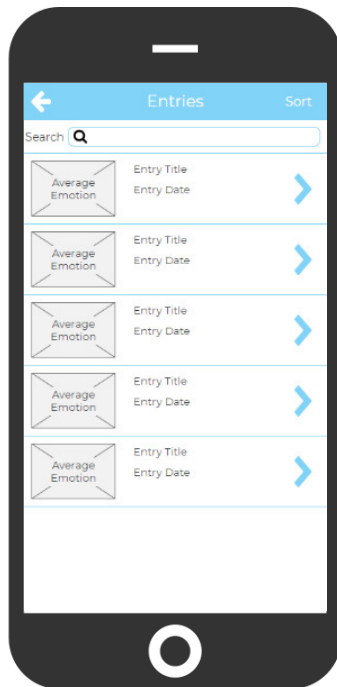


Figure 9 - Entries Screen

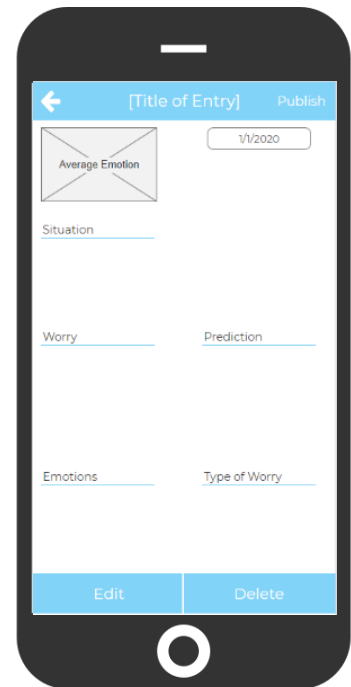


Figure 10 - Single Entry

Screen:	Entries screen
Description:	<p>These screenshots show the entries screen and the screen you are presented when you select an individual entry. Every entry entered up to a certain amount will be displayed in the entries screen. You will then have the option to either search for an entry title you may have previously entered or sort in terms of entry date – most recent is the default view when the screen is presented.</p> <p>Once you have clicked an entry, you will be taken to an entries screen where you can view every detail you entered. You have three options on this page: edit, delete and publish. Selecting edit will allow you to edit any details you may have gotten wrong from the current entry. Clicking delete will completely remove the entry from the database. Finally, pressing publish will then give the therapist access to view that entry and begin analysis on it.</p>

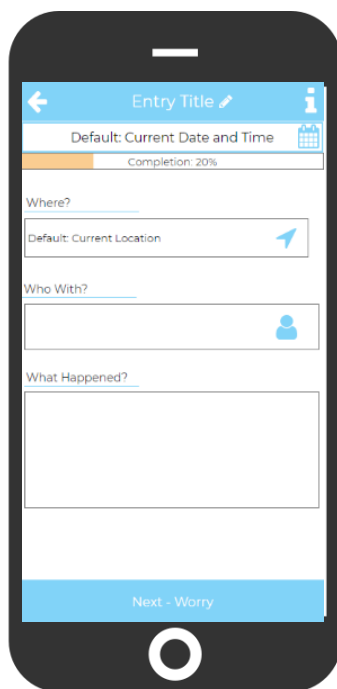


Figure 11 - Situation Screen

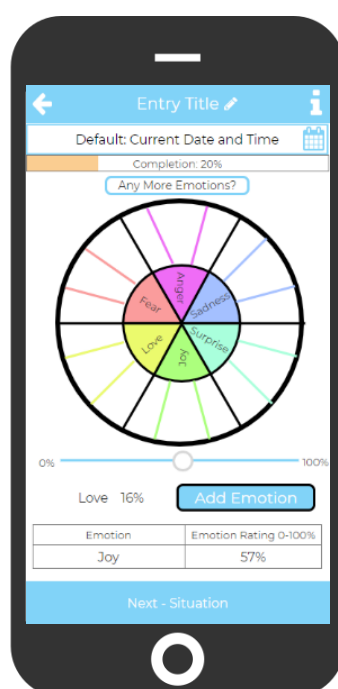


Figure 12 - Emotions Screen

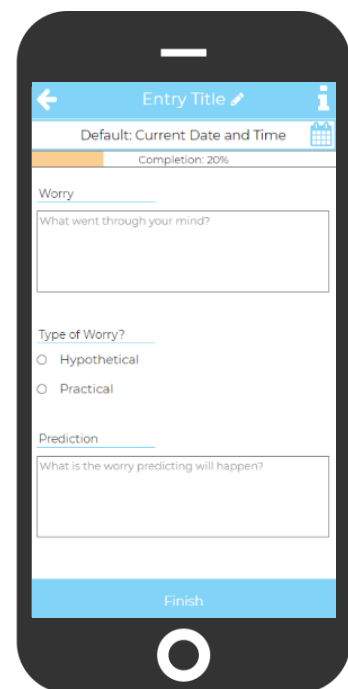


Figure 13 - Worry Screen

Screen:	Entry Form Screen's
Description:	<p>These screenshots contain the most important part of the application. These are the three screens that make up the diary entry form and are what the user will be spending most of their time on. Because of this, these screens need to be as clear and concise as possible. If the user struggles with conveying their thoughts and emotions through this format, it could tarnish their treatment and stall their recovery time.</p> <p>Each screen contains the opportunity to change the title and current date and time. This gives the user the opportunity to change both if they feel like they made a mistake or wish to change some details.</p> <p>The situation screen is really looking to answer three questions to set the scene: 'where did this happen?', 'who were you with?' and 'what happened?'. By answering these questions, the therapist can gain an understanding of what the situation was the patient was involved in without hearing about their reaction to the event yet. The therapist can use this to think of what their own reaction would be and if the patient's reaction was warranted.</p> <p>The emotions screen takes inspiration from the 'Emotion Wheel' that therapists often use to help their patients describe their emotions best. It takes the eight most basic emotions: anger, anticipation, disgust, fear, joy, sadness, surprise and trust. Then extends them to more precise emotions. For example, if the patient were to choose 'Sad', then this could be extended, for example: lonely, vulnerable, despair, guilty, depressed or hurt, and so on. The user is also asked to rate their emotion on a scale of 0-100%.</p>

	<p>The worry screen is where the user's reaction to the situation can be expanded on and explained. It is here where the therapist will see if the reaction is within normal limits of how someone should normally react, or is their disorder being triggered by this event.</p>
--	---



Figure 14 - Person Overlay

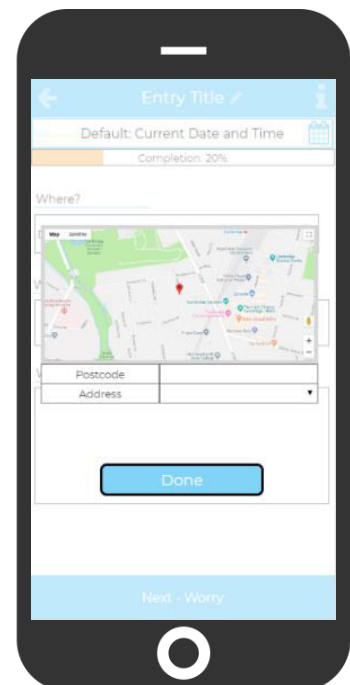


Figure 15 - Location Overlay

Screen:	Situation Overlay's
Description:	<p>These two screens are used to input data from the situation screen. When asked 'Who were you with?', the therapist can get a much better understanding from this if a title is also used alongside a name. Because of this, the patient will be asked for the relations of the person involved, with the name being optional to support the anonymity of the situation. The patient is also asked where they were during the situation. For this, a map will be brought up and the patient can pinpoint where they were or input the address if known using the postcode, which then brings up a list of properties under the same postcode.</p>



Figure 16 - Calendar Overlay's

Screen:	Entry Calendar Overlay
Description:	The purpose of these screens is to produce a calendar over the top of the current entry screen they are located in. This is so the user has constant access to change the date of which the situation at hand took place. The overlay is to not have to take the user off of the current entry screen and panic them as to what happened to all of their information they just entered.

Figure 17 - Application Information Screens



Screen:	Information Screens
Description:	Throughout the application, there are pages which introduce buttons with the functionality commonly used where I'd hope users would recognise and be able to deduce what it is trying to achieve in terms of my

	application. If they are confused about what anything does, there will be an 'i' icon in the top right of the screen to bring up the corresponding help screen explaining what each of the icons do. Along with help with the icons, there will be help displayed as to what each section of the form entails and how to make the most of your diary entries. This wouldn't tell the user exactly what to write in each section, as that could cause some forced information, but give the user a push in the right direction.
--	--

## Heuristic Evaluation

A heuristic evaluation is a method employed to inspect the usability for a piece of software, helping to identify any usability problems in the user interface. [2] As to the focus of this application is to be as efficient and user friendly as possible, performing this evaluation on my initial application design was very important.

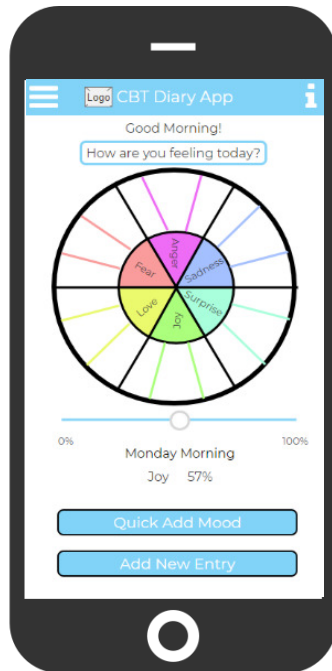
I will be using Jakob Nielsen's 10 heuristic principles to perform this evaluation. [2]

Principles:

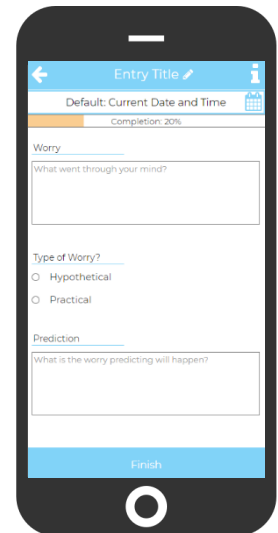
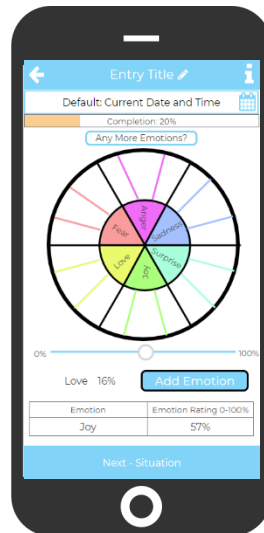
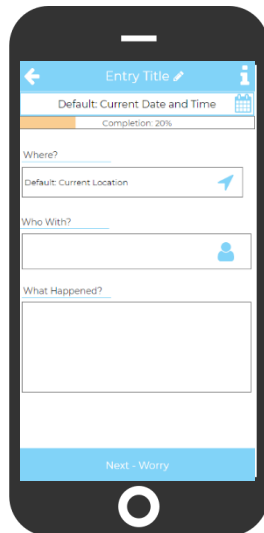
1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

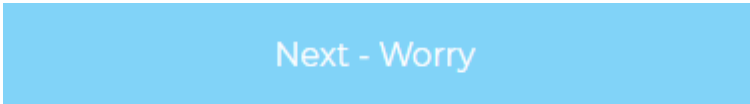
Severity:

1. Not a problem
2. Cosmetic – Only fix if time permits
3. Minor – Fix this issue at a low priority
4. Major – Fix this issue at a high priority
5. Catastrophic – Imperative to fix this issue



ID	Evaluation1
Screen(s):	Home Screen
Problem	Emotion's Wheel takes up too much room and the options are too vague
Heuristic(s) Violated	Flexibility and Efficiency, Aesthetic and Minimalist Design
Severity	3
Description	<p>This page could lead to an inefficient completion of an entry due to confusing the user. The user has no inclination that the emotion wheel has to be completed for the 'Quick Add Mood' or 'Add New Entry' or both.</p> <p>The screen is also crowded with multiple interactable elements. This could make it difficult for the user to accurately fill out this form.</p>
Solution	Remove the emotion wheel from the home screen design to make it clearer what options the user has once they log onto the application.



ID	Evaluation2
Screen(s):	Entry Screens: Situation, Emotion and Worry
Problem	The entry screens don't give enough feedback to the user regarding length of form
Heuristic(s) Violated	Show System Status
Severity	3
Description	A 'completion' percentage is displayed at the top of the screen as the user progresses through the form – this shows us a glimpse of system status. By using the buttons at the bottom to progress through the form, it could leave the user wondering 'how long is left?' and rushing the form. This would then not be an accurate depiction of their entry.
Solution	Replace the button navigation with highlightable tabs to show the user what page they're on and how long they have left to fill out before submitting.
Evidence	



## Final Prototype Design

Through the evaluation of the main screens of the application, here are the improved versions of the screens which failed a heuristic evaluation. The rest of the screens are to remain the same as they didn't fail a heuristic.

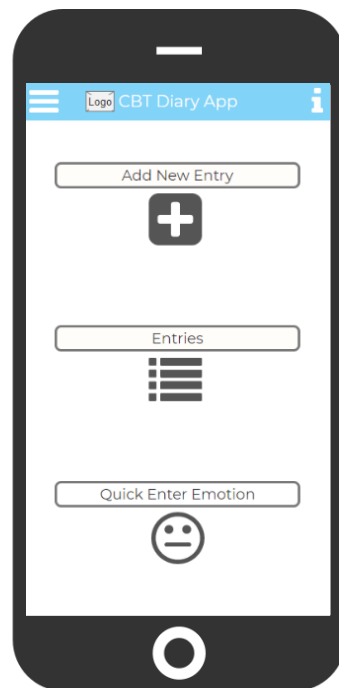


Figure 18 - Improved Home Screen

Related Evaluation ID	Evaluation1
Screen:	Home Page
Improvements:	The evaluation concluded that the emotion wheel took up too much room for the home screen and there wasn't enough clarity as to what the user can achieve or not. Because of this, I removed the emotion wheel and replaced it with three simple options: add new entry, entries and quick enter emotion. These will now be the three main functionalities of the application.

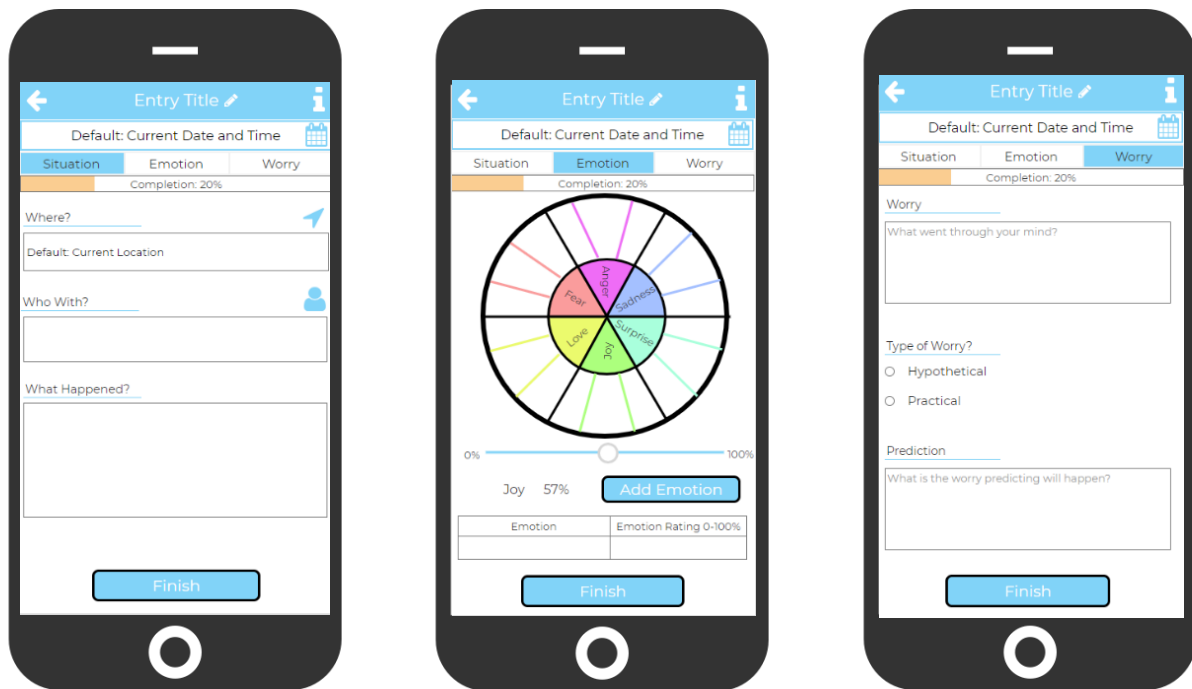


Figure 19 - Improved Entry Screens

Related Evaluation ID	Evaluation2
Screen:	Entry Page's
Improvements:	It was determined that the entry screens were too vague on the completeness of the form. Because of this, I have replaced the button at the bottom to progress the screen to top tabs. These tabs have the three entry screens available at all times. This shows the user they only have three sections to complete and can finish entering the form at any point by selecting 'Finish' at the bottom, which is available in every tab.

## Implementation

Unfortunately, due to time constraints, not everything spoken about within this section has been fully implemented into the front end of the application. I will now discuss the routes I have created and their intentions for the execution of the front end of the application. As previously mentioned, these routes are very important to allow the front end of the application to be able to gather information from the database. [7]

### UML Class Diagram

This diagram shows how the classes within my application interact with each other. To simplify this diagram, some methods and classes have been left out.

By providing a UML diagram, it is easier to picture the logical designs I discuss within this segment of the report.

Software used to design UML:  
[www.lucidchart.com](http://www.lucidchart.com)

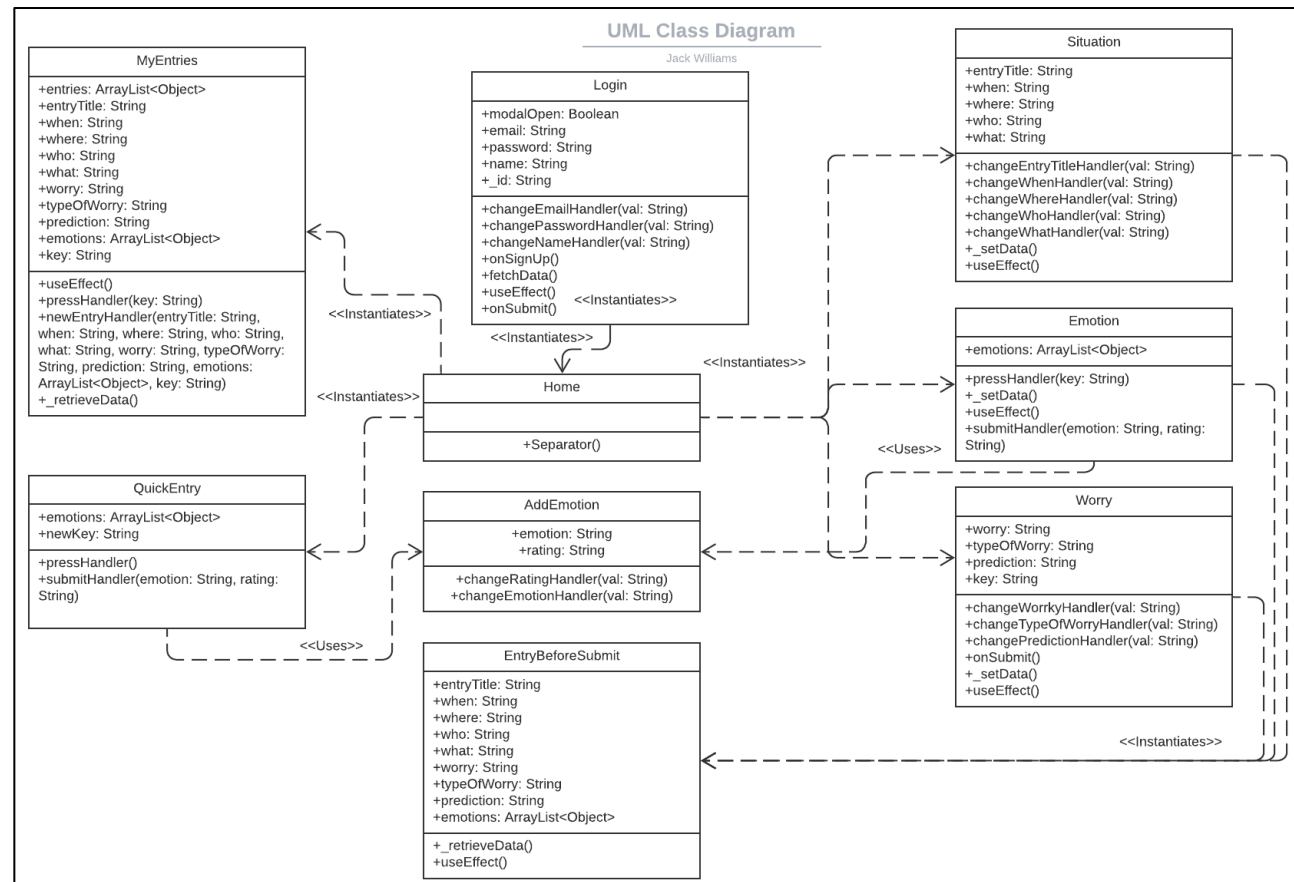


Figure 20 - UML Class Diagram

## Database

This database is using Node.js, MongoDB Atlas and the MongoDB object data modelling library, Mongoose. Node.js is needed to establish a connection to a database, query the database and manipulate the data by either inserting, deleting or updating. [13] MongoDB Atlas is used to handle the responsibility of hosting, patching, managing and securing a MongoDB cluster. [14] A MongoDB cluster is simply used to store a collection data in an efficient manner by breaking a collection into shards. Each shard then contains a subset of the sharded data. [15]

## User Schema

Schema's are used to organise data for the database to understand how data is going to be constructed within a collection, like a skeleton of the database. [16]

The User Schema I have created is designed to set a user up with the ability to create multiple entry objects within the 'entries' array. With the entries array being stored within a user's profile, it keeps these entries private and secure for only that user to see. At no point would a user be able to view an entry from another user's entries array stored on the database. This is a key component of the application as it ensures the users safety and trust in the application.

```
const UserSchema = new mongoose.Schema({
  name: {
    type: String,
    trim: true,
    required: true
  },
  email: {
    type: String,
    unique: true,
    required: true,
  },
  password: {
    type: String,
    required: true
  },
  tokens: [{
    access: {
      type: String,
      required: true
    },
    token: {
      type: String,
      require: true
    }
  }],
  entries: [{
    inputDate: {
      type: Date,
      default: Date.now
    },
    entryTitle: String,
    when: String,
    where: String,
    who: String,
    what: String,
    emotions: [{
      emotion: String,
      rating: String,
      key: String,
    }],
    worry: String,
    typeOfWorry: String,
    prediction: String
  }]
})
```

Figure 21 - User Schema

MongoDB Atlas automatically assigns each object inserted an object id, `_id`. This can be used to distinguish different entry objects within the entries array. An entry will consist of ten different sections where the user will be asked to fill in nine of these. The only section not editable by the user is the 'inputDate' section. This has been added to the schema to distinguish the therapist viewing the entry created by the user the time differential from a patient experiencing a situation to when they are creating a diary entry. To have the most accurate entry, it is most beneficial to both the patient and therapist to fill in the diary as close to the situation as possible. The therapist can then use this information and feedback to their patient if there is a concerning difference between 'inputDate' and 'when', which is the field inputted by the user. Another advantage to this schema design is to be able to input multiple emotions for each entry to describe how they are feeling. The user will also enter a rating between 0-99 for any emotion that they have input regarding a situation

Below is an example of a user that has been created within the MongoDB Atlas database. This user has entered a single entry.

```
{
  "_id": "ObjectId('5ed406988f13210e94a487d8')",
  "name": "Test User",
  "email": "TestUser@Cardiff.ac.uk",
  "password": "$2b$10$0ncDTb7DMHFJReH3/N70C.KKSgX7VA7XJ308VCwBnExvv2MXA5cK.",
  "entries": [
    {
      "_id": "ObjectId('5ed406988f13210e94a487d9')",
      "entryTitle": "Walk in the park",
      "when": "30-05-2020",
      "where": "Park",
      "who": "Mum and a stranger",
      "what": "We went to the park for a walk and there was a stranger behind us as w...",
      "emotions": [
        {
          "_id": "ObjectId('5ed406988f13210e94a487da')",
          "emotion": "Anxious",
          "rating": "50",
          "key": "1"
        },
        {
          "_id": "ObjectId('5ed406988f13210e94a487db')",
          "emotion": "Worried",
          "rating": "15",
          "key": "2"
        }
      ],
      "worry": "That the person behind us would get close enough to be able to cause e...",
      "typeOfWorry": "Practical",
      "prediction": "Either me, my mum or both of us will be attacked by the person behind ...",
      "inputDate": "2020-05-31T19:33:44.908+00:00"
    }
  ],
  "tokens": [
    {
      "_id": "ObjectId('5ed406988f13210e94a487dc')",
      "access": "auth",
      "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfcmFudCI6IiJZWQ0MDY5ODhmMTMyMTB1O..."
    }
  ],
  "__v": 1
}
```

Figure 22 - Example User and Entry

## System

This section will demonstrate the different ways that the system interacts with the database. I will talk about the routes I have created and what the intention was for the front end to be able to execute. To show the results of the code I have implemented in the app.js file located in the server folder, I will be using the Postman application.

## Postman

Postman is an API (application programming interface) client that allows a developer to send HTTP requests to a server and read the response. [17] This is beneficial because of its efficiency, enabling test routes without the need to write code. With Postman, you can see the server's response to your request straight away. Therefore, it is simple to see the information your front end will need to supply and the reply from the server that it will receive.

The HTTP methods I have implemented are:

- Get – Method used to perform a read operation
- Post – Method used to add a new resource
- Delete – Method used to remove a resource [18]

## Create New User

This route is a post route, meaning it's used to add a new resource to the database. In our case, used when signing up for a new account. In this route, the data being requested is the name, email, password and entries but the only required data at this stage is the name, email and password. This is requested from 'body' which is what the front end should send in a JSON format. This user's data is placed into a 'user' constant and a new user is created using the schema showcased earlier with the user data as the input. This new user is then processed by the Mongoose middleware 'save' which is used to insert the document into the database.

```
app.post('/create-user', (req, res) => {
  const userData = {
    name: req.body.name,
    email: req.body.email,
    password: req.body.password,
    entries: req.body.entries
  }
  const user = new User(userData)
  user.save()
    .then((user) => {
      if (user) {
        return user.generateAuthToken();
      } else {
        res.sendStatus(400)
      }
    })
    .then((token) => {
      res.header({ "x-auth": token }).send(user);
    })
    .catch((error) => {
      res.status(400).send(error)
    })
})
```

Figure 23 - Create User Route

In the user schema, the Mongoose middleware 'pre' is used. This takes an execution of a middleware method as its input and then, in my instance, calls the next function to call each middleware used one after another. In this case, before the 'save' middleware is executed, call the current user in question using 'this' and check if the password has been modified. This will be true in the case of changing the password or signing up for the first time. The password is then hashed using the bcrypt library. Bcrypt is a password hashing function based off of the Blowfish block cipher. [19] A salt, which is a piece of randomised data, is generated with a fixed number of rounds and is used as an additional input when hashing a password. [20]

```
UserSchema.pre("save", function (next) {
  const user = this;
  if (user.isModified("password")) {
    bcrypt.genSalt(saltRounds, function (err, salt) {
      bcrypt.hash(user.password, salt, function (err, hash) {
        // Store hashed password in database
        user.password = hash;
        next();
      });
    });
  } else {
    next();
  }
});
```

Figure 24 - Mongoose 'pre' middleware

Here is an example of a user's password presented in the database.

```
{
  _id: ObjectId("5ed406988f13210e94a487d8"),
  name: "Test User",
  email: "TestUser@Cardiff.ac.uk",
  password: "$2b$10$0NcDTb7DMHF3ReH3/N70C.KK5eX7VA7xJ308VcWbNExvv2MXA5ck."
}
```

Figure 25 - User Hashed Password Example

By doing this, the password entered by the user is encrypted before being stored in the database. This is the safest way to protect a user from an attack. If a hacker were to access the database, they will only find the plaintext of their email which without the password, would be useless in terms of retrieving information on the database. The hacker would have to attempt a brute force attack to find the data produced by the salt to hash the key in the first place, which would take a long time.

After the password has been hashed, the generateAuthToken function is called on the object produced by the outcome of user.save(). This is done by using the then() method which returns a Promise. A promise in terms of JavaScript is an object that will be in one of three states: fulfilled, rejected or pending. [21] If the user data exists, generate an authentication token for the user, else return a 400 error which indicates a bad request.

The generateAuthToken function takes the current user in question and creates an authentication token. It does this by using the JWT (JSON Web Tokens) library. This token string is created by signing the user object and the type of access the user has to the database, with a random secret key. The token and access level are then pushed into the tokens array of the user. This authentication token is used by later routes to identify different users safely and securely from each other.

```

UserSchema.methods.generateAuthToken = function () {
  const user = this;
  const access = "auth";

  const token = jwt.sign({ _id: user._id.toHexString(), access }, 'ASDFGHJKL123').toString();

  user.tokens.push({ access, token });

  return user.save().then(() => {
    return token;
  })
}

```

Figure 26 - generateAuthToken function

Here is an example of how the token and access level is presented within the database.

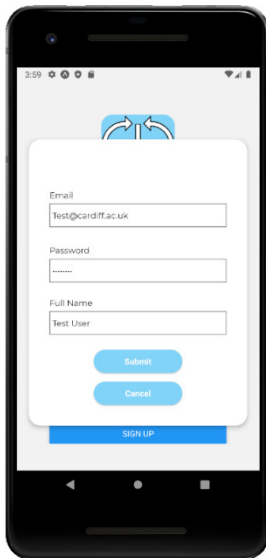
```

tokens: Array
  0: Object
    _id: ObjectId("5ed406998f13210e94a487dc")
    access: "auth"
    token: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfawQioiI1ZWQ0MDY5ODhmMTMyMTB10..."
    __v: 1

```

Figure 27 - User Token Example

Below is an example of how the user interface sends the data to the database and how all the different methods mentioned affect the data.



```

_id: ObjectId("5ed518580b2e2c6b609e9cbb")
name: "Test User"
email: "Test@cardiff.ac.uk"
password: "$2b$10$/xBbKxmUycOV0nQbCc4Q0FAR5Ggnv/Oa041e.RyI8YAcG5tP86Ri"
entries: Array
tokens: Array
  0: Object
    _id: ObjectId("5ed518590b2e2c6b609e9cbc")
    access: "auth"
    token: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfawQioiI1ZWQ1MTg1ODBiMTMyYzZiN..."
    __v: 2

```

Figure 28 - New User Full Example

## Login

This route is also a post route, adding an authentication token to the user that is being logged in. It does this by calling a function from within the UserSchema, once again – findUserByCredentials. This function requests the email and password from the front end in a JSON format.



```

app.post('/login', (req, res) => {
  User.findUserByCredentials(req.body.email, req.body.password)
    .then((user) => {
      if (!user) {
        console.log('No user found')
      } else {
        user.generateAuthToken()
          .then((token) => {
            res.header({ "x-auth": token }).send(user);
          });
      }
    });
});

```

Figure 29 - Login Route

The findUserByCredentials function uses the UserSchema blueprint to guide the search. The search in question is findOne, using the email object taken from the route. If a user doesn't exist, then the user argument being passed into the 'then' function will return null and so declaring the if statement as (!user) will be true and the Promise is rejected. If a user is found, a new Promise is created. Using the bcrypt library, the password entered in the front end is encrypted in the same way and compared to the hashed password already stored. If these hash values match, the response is true, and the user is resolved within the Promise and returned from the function. If the passwords don't match, the Promise is rejected. This is how my application authenticates user logins without compromising any of the user's data by having a point where two plaintext passwords are compared. It then uses the user retrieved to add the authentication token into the header.

```

UserSchema.statics.findUserByCredentials = function (email, password) {
  const User = this;
  return User.findOne({email}).then((user) => {
    if (!user) {
      Promise.reject(new Error('User Not Found'))
        .catch((err) => {
          console.log(err)
        });
    } else {
      return new Promise((resolve, reject) => {
        bcrypt.compare(password, user.password, (err, res) => {
          if (res) {
            resolve(user);
          } else {
            reject();
          }
        })
      })
    }
  });
}

```

Figure 30 - findUserByCredentials Function

This route has been fully implemented into the main application and can check if a user's information is correct based on the information provided on the log in screen. If the data

they have provided doesn't match any profile from the database, an error message is displayed informing the user that they have entered an incorrect email or password. If it is correct, they are taken to the home screen. To confirm the correct user has been logged in, I printed their details to the console and cross checked it with the database.



Figure 31 - User Log in Full Example

Afterwards, the user is navigated to the improved home screen, displaying all of the main functionalities the application has to offer.

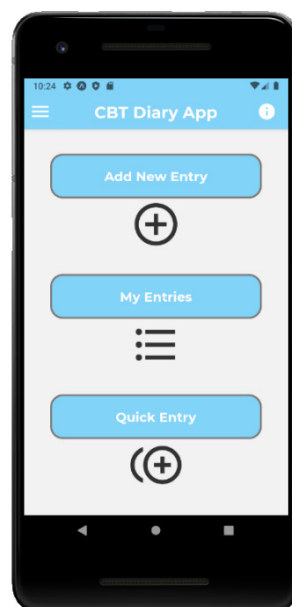


Figure 32 - Home Screen

## Retrieve User

Unfortunately, this route and the following route, logout, could not be implemented within the main system due to time constraints. Given this, the route was still set up and tested to be successful using Postman.

When a user logs in, they are given a new authentication token which can be used to retrieve unique users.

```
app.get('/get-user', authenticate, (req, res) => {
  res.send(req.user)
})
```

Figure 33 - Get User Route

This route is a get route, meaning it's used to perform a read operation. Read operations, also known as retrieve operations, are used to gather specific information. [22] The following route was first sent through a custom piece of middleware I created called 'authenticate'.

```
const mongoose = require('mongoose')
require('../User')
const User = mongoose.model("user")

const authenticate = (req, res, next) => {
  var token = req.header("x-auth");
  User.findUserByToken(token).then((user) => {
    if (!user) {
      Promise.reject().catch((err) => {
        res.status(401).send('User Not Recognised');
      })
    } else {
      req.user = user;
      req.token = token;
      next();
    }
  }).catch((err) => {
    res.status(401).send('Not Recognised');
  })
}

module.exports = authenticate;
```

Figure 34 - Custom Authenticate Middleware

In the authenticate middleware, it calls in the UserSchema blueprint and creates a new constant using that model. Later, we can use that empty blueprint to invoke a UserSchema function onto it, using a token as the argument input. This token is taken from the header supplied from the front end of the application. Since we were not able to implement this, here is how this information is being supplied to the backend through Postman, using the x-auth header as an active token for the input.



## Logout

The logout route works the complete opposite to the login route. Where the login route authenticates the user and generates an auth token, the logout route authenticates the user and removes the auth token – as seen in the image below.

```
app.delete('/logout', authenticate, (req, res) => {
  req.user.removeToken(req.token).then(() => {
    res.status(200).send("User logged out");
  }).catch(() => {
    res.status(401).send();
  })
})

//Check if server is running via console
app.listen(3000, () => {
  console.log('Server Running')
})
```

Figure 38 - Logout Route

This route is a delete route, meaning it's used to remove a resource. It first determines if the authentication token provided matches an existing auth token from within the database. If there's a match, the route requests the user and invokes the custom function, `removeToken` onto it – requesting the matching token as an argument, as per below.

```
UserSchema.methods.removeToken = function(token) {
  const user = this;

  return user.updateOne({
    $pull: {
      tokens: {token}
    }
  })
}
```

Figure 39 - removeToken Function

This function takes the current user in question, using 'this', and updates the user by pulling the token supplied from the route out of the token array stored within the user. This leaves the user without any tokens in the array, meaning they cannot access any application functionalities anymore.

Here is an example of a successful logout using Postman to showcase the code implemented.

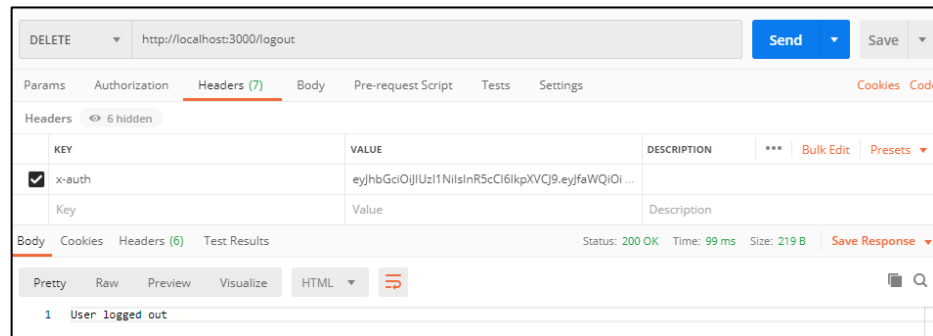


Figure 40 - Logout Postman Example

## Navigation

React native offers many different styles of navigation. I have implemented three ways to traverse around a mobile application - using stacks, drawers and tabs. A stack navigator allows you to transition between screens where each new screen is placed on top of a stack. A drawer navigator allows you to partially bring a screen on top of the current screen before actually transitioning away from the original screen – like pulling a drawer out from the side of the screen. The final navigation technique I have implemented is tabs, which is what most users will be accustomed using when on mobile applications. Tab navigation presents all the reachable screens either at the top or the bottom of the screen to press, highlighting what screen you're currently on.

```
export default function Navigator() {
  console.log('Navigator called from HomeStack');
  return (
    <NavigationContainer>
      <HomeStack.Navigator>
        <HomeStack.Screen
          name='Login' ...
        />
        <HomeStack.Screen
          name='Home' ...
        />
        <HomeStack.Screen
          name='HomeInformation' ...
        />
        <HomeStack.Screen
          name='Situation' ...
        />
        <HomeStack.Screen
          name='EntryBeforeSubmit' ...
        />
        <HomeStack.Screen
          name='MyEntries' ...
        />
        <HomeStack.Screen
          name='SelectedEntry' ...
        />
        <HomeStack.Screen
          name='QuickEntry' ...
        />
        <HomeStack.Screen
          name='QuickEntryInformation' ...
        />
      </HomeStack.Navigator>
    </NavigationContainer>
  );
}
```

This image shows the main navigator from mainNavigation.js called from the App.js. This contains the way that every screen is linked. The core idea of the navigation container is a stack from the login page to each page you see in the screenshot. When a button is pressed on these screens with the intention for navigation, the screen is added on to the top of the stack. Once the user has logged in and taken to the home screen, that becomes the origin of the stack. It is from the home screen where you can access every aspect of the application.

Figure 41 - Navigation Container

The component called from within the Home screen within the stack is the HomeDrawerScreen. This component is where the second form of navigation is stored – the drawer. Inside here is where the user can find their profile and settings screen.

```
<HomeStack.Screen
  name='Home'
  component={HomeDrawerScreen}
  options={{
    headerShown: false
  }}
/>
```

Figure 42 - Home Screen Component

The components called within this HomeDrawerScreen, HomeDrawScreen, ProfileStackScreen and SettingsStackScreen, are used to provide the navigator with information regarding what to display on the screen, what to place inside the header of the screens and to allow you to re-open the drawer from these screens.

```
const HomeDrawerScreen = () => (
  <Drawer.Navigator initialRouteName="Home">
    <Drawer.Screen
      name="Home"
      component={HomeDrawScreen}
    />
    <Drawer.Screen
      name="Profile"
      component={ProfileStackScreen}
    />
    <Drawer.Screen
      name='Settings'
      component={SettingsStackScreen}
    />
  </Drawer.Navigator>
)
```

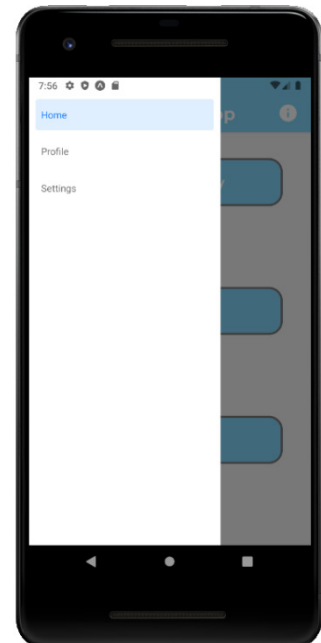


Figure 43 - HomeDrawerScreen

The final navigation implementation is the tabs section that has been implemented for the diary entry. This must be accessed initially from the HomeStack and uses the component EntryTabsScreen.

```
<HomeStack.Screen
  name='Situation'
  component={EntryTabsScreen}
  options={{
    headerShown: false
  }}
/>
```

Figure 44 - Entry Screen Component

The entry tabs screen holds information regarding all three of the entry screens: situation, emotions and worry. The tabs have all been assigned a suitable icon and distinguishable text colour to indicate what screen you are currently visiting.

```
const EntryTabsScreen = () => (
  <EntryTabs.Navigator
    > initialRouteName='Situation' ...
    </EntryTabs.Navigator>
  <EntryTabs.Screen
    > name='Situation' ...
    </EntryTabs.Screen>
  <EntryTabs.Screen
    > name='Emotions' ...
    </EntryTabs.Screen>
  <EntryTabs.Screen
    > name='Worry' ...
    </EntryTabs.Screen>
)

```



Figure 45 - EntryTabsScreen

## Navigation Overview

Here is an overview of how users can navigate around the app to access the three main focuses.

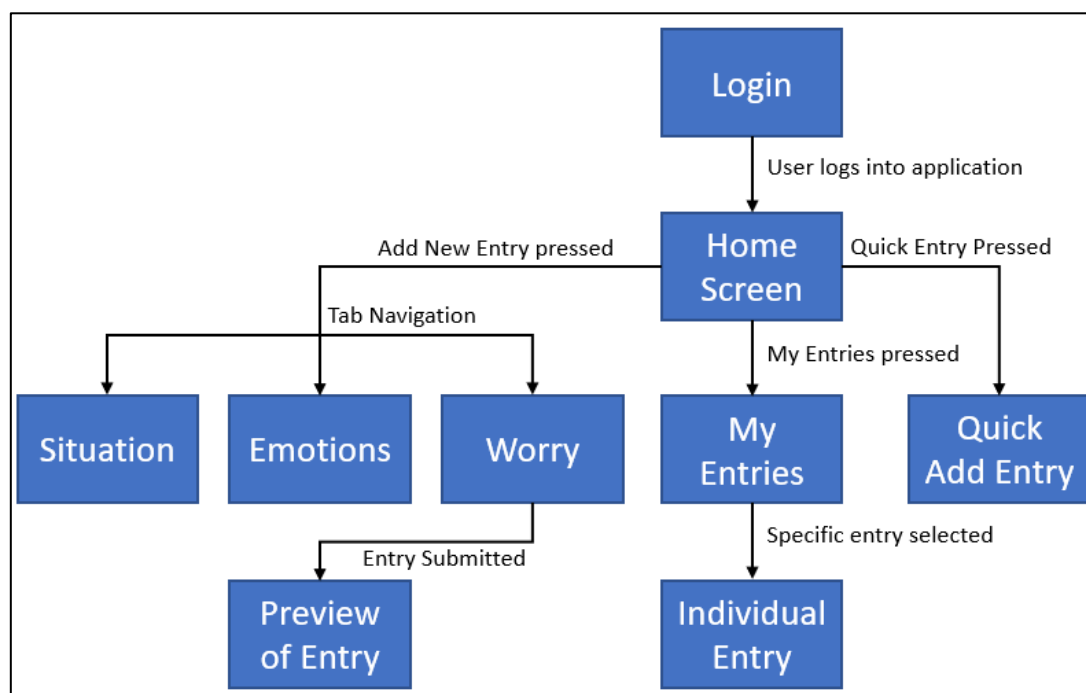


Figure 46 - Navigation Overview



## Add New Entry

From the home screen, you have the option to begin a new diary entry. If this is selected, the three tabs of 'situation', 'emotions', and 'worry' are called. It is in these screens where the user will spend most of their time on the application. Therefore, it is imperative to the core goal of this app that these screens are simple to fill in, user friendly and display clear instructions as to what the different inputs do and what they are for. It is because of this that I provided an information page for each of the three different screens. During my research, I found that the higher rated applications contained information pages on what each function did, so I adopted the same approach in my own application design. This improves the user friendliness of the app, catering to the users looking to gain more knowledge on what to put into an input box or just to clear up any confusion.

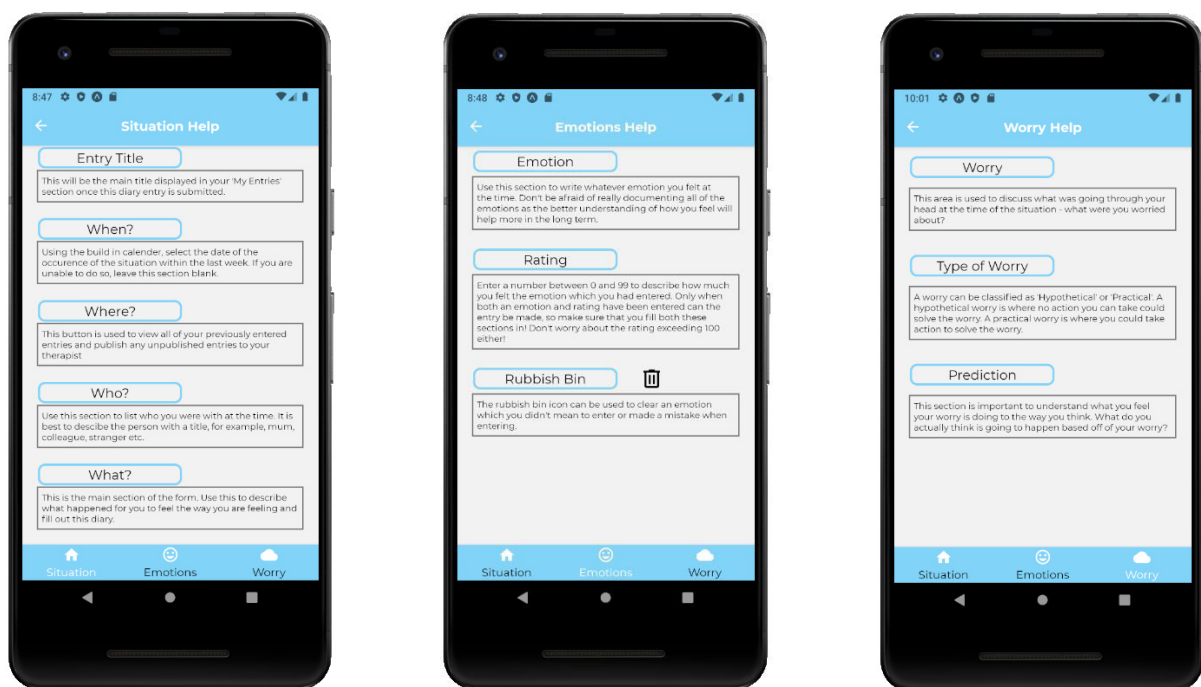


Figure 47 - Entry Help Screens

These three screens will help aid the user through what each input box is asking them to achieve and different icon meanings.

Another function that all three screens share is the ability to quit the process from any screen.

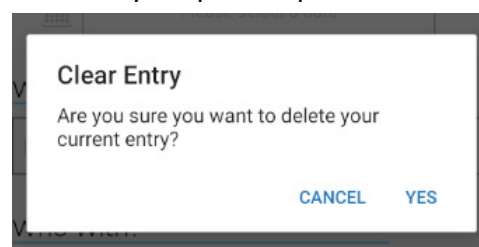


Figure 48 - Clear Entry Example

This small bin icon is located at the top left of each tabbed screen. By pressing this, you are presented with an alert box which asks you if you wish to abandon the current diary entry you are creating. Once this has been accepted, there is no way to retrieve the information you have just inputted.

Each text box on the entry screens processes information in the same way. It is gathered through one of the core components to React Native – `TextInput`. This component allows users to enter text into a field. `TextInput` contains an attribute called `'onChangeText'` which is invoked every time the text box is modified. I used this to create a function that processes that value every time it's changed.

```
const [entryTitle, setEntryTitle] = useState('');  
  
const changeEntryTitleHandler = (val) => {  
  setEntryTitle(val);  
}
```

Figure 49 - Text Input Function Example

This function takes advantage of the `useState` hook available within React Native. Hooks allows you to access core features of React without having to use classes. [23] The `useState` hook lets you declare a state variable. This means that whilst we're on a certain page, that data is preserved. The only argument taken by the `useState` is the initial value, in this case it's an empty string.

As the database couldn't be implemented in time, an alternative method was used to be able to process information that needs to be stored within the 'My Entries' section of the application. This method was using `AsyncStorage`. `AsyncStorage` is an unencrypted, key-value storage system that is global to the app. By this, information can be sent and stored using `AsyncStorage`, then this same data can be retrieved and displayed anywhere in the application. This method is purely used to demonstrate what the application could be able to achieve with the proper routes set up via the database. This could not be considered as a permanent solution as it doesn't offer continuous data storage, rendering diary entries useless. Once a new entry was created, it would replace the old data placed within the `AsyncStorage`.

```
const _setData = async () => {  
  try {  
    await AsyncStorage.setItem('entryTitle', entryTitle);  
    await AsyncStorage.setItem('when', when);  
    await AsyncStorage.setItem('where', where);  
    await AsyncStorage.setItem('who', who);  
    await AsyncStorage.setItem('what', what);  
    console.log('Information Stored')  
  } catch (error) {  
    console.log(error)  
  }  
}  
  
useEffect(() => {  
  _setData()  
})
```

Figure 50 - AsyncStorage setItem

Above is a screenshot of how data is being assigned within the situation screen. I am taking advantage of a different hook – `useEffect`. This hook runs after every screen render and update. Including the `onChangeText` attribute I spoke about earlier. Every time an entry box is updated, the `AsyncStorage` is setting each item to the most up to date value and storing it for later use.

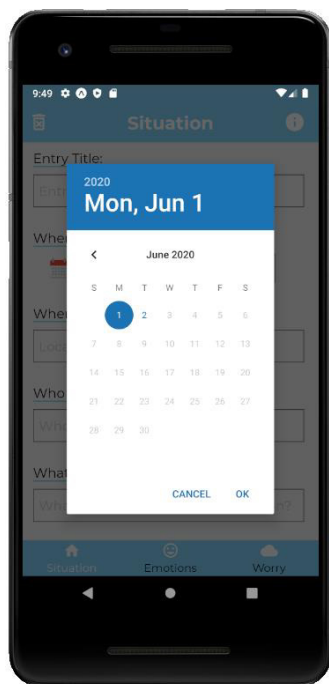
### *Situation*

The situation is the first screen you are presented with when starting a new diary entry.



*Figure 51 - Situation Screen*

The fields: `entryTitle`, `when`, `who` and `what` are the variables for the corresponding text boxes which are titled: entry title, when did this happen, where did this happen, who with and what happened. As previously mentioned, the text boxes will be using a combination `useState`, `useEffect` and `AsyncStorage` methods to update and store the information. The one entry field which differs on this screen is the 'when' variable.



```
<DatePicker
  style={styles.datePicker}
  date={when}
  mode="date"
  placeholder="Please select a date"
  format="DD-MM-YYYY"
  minDate={Moment().subtract(7, 'days').format('DD-MM-YYYY')}
  maxDate={Moment().add(0, 'days').format('DD-MM-YYYY')}
  confirmBtnText="Confirm"
  cancelBtnText="Cancel"
  customStyles={{
    dateIcon: {
      position: 'absolute',
      left: 0,
      top: 4,
      marginLeft: 0
    },
    dateInput: {
      marginLeft: 36
    }
  }}
  onChange={changeWhenHandler}
/>
```

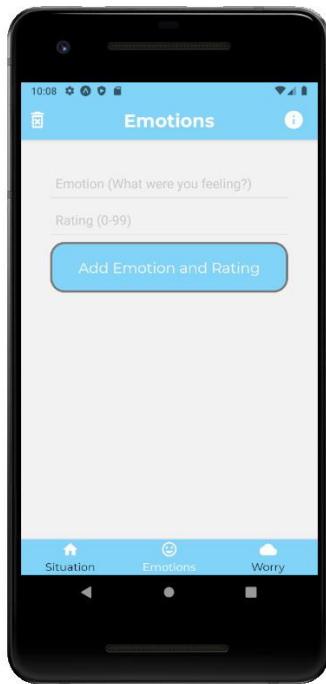
Figure 52 - DatePicker

When the date field is clicked, this calendar (as shown above) is presented. This is a component taken from a community library on GitHub called 'react-native-datepicker'. I have set the minimum date to be at most seven days prior to the current day. Encouraging the users to embrace the CBT process and input a diary entry as the situation occurs when their emotions are most fresh in their mind. Additionally, to stop users incorrectly entering a date from the future, the maximum date entered is the current date of use.

In the original design for the application, in the location field, the user was able to bring up a map to enter the location of the event. This would have given the user a pin to move around, defaulted at their current location or the option to enter a postcode and choose an address from a drop-down list. Once again, due to the complexity of the Google Maps API and the time constraint, this was removed from the final product. Instead, the user is presented with a text box and can enter whatever they wish in this section. This can range from an address to a vague description e.g. The Park. I was disappointed not to be able to achieve this functionality but am proud to have thought of an alternative method.

## Emotions

In the original prototype for this screen, I was going to use the emotions wheel to allow the user to input information regarding what they're feeling. This was a great idea in theory as utilising the wheel to pinpoint their feelings is already a part of their diary entry routine, as implemented by their therapist. When it came to develop the application, this proved too difficult to implement and so the following design was chosen instead.



```
<ScrollView style={globalStyles.container} >
  <View style={styles.content}>
    <AddEmotion submitHandler={submitHandler} />
    <View style={styles.list}>
      <FlatList
        data={emotions}
        renderItem={({ item }) => (
          <EmotionItem item={item} pressHandler={pressHandler} />
        )}
      />
    </View>
  </View>
</ScrollView>
```

Figure 53 - Emotions Screen

This design allows the user to enter any text they want in the 'Emotion' section and a number between 0-99 in the 'Rating' section. This has an element of freedom where the emotion wheel may constrict the user to a certain range of emotions. To add a new emotion to the list, I created a custom component called 'AddEmotion'.

```
export default function addEmotion({ submitHandler }) {

  const [emotion, setEmotion] = useState('');
  const [rating, setRating] = useState('');

  const changeEmotionHandler = (val) => {
    setEmotion(val);
  }

  const changeRatingHandler = (val) => {
    setRating(val);
  }

  return (
    <View>
      <TextInput
        style={styles.input}
        placeholder='Emotion (What were you feeling?)'
        onChangeText={changeEmotionHandler}
      />
      <TextInput
        style={styles.input}
        placeholder='Rating (0-99)'
        keyboardType='numeric'
        onChangeText={changeRatingHandler}
      />
      <TouchableOpacity onPress={() => submitHandler(emotion, rating)}>
        <View style={globalStyles.button}>
          <Text style={styles.buttonText}>
            Add Emotion and Rating
          </Text>
        </View>
      </TouchableOpacity>
    </View>
  )
}
```

Figure 54 - AddEmotion Function

This custom component works the same as the situation screen where every time either of the text boxes are modified, the corresponding variable state is updated to match the text inputted. When the user feels they are ready to submit their emotion, they press the 'Add Emotion and Rating' button. This invokes the submitHandler function, using both the emotion and rating as the argument.

```

const submitHandler = (emotion, rating) => {
  const newKey = Math.random().toString()

  if ((emotion.length > 2)
    && (rating.length > 0)
    && (rating.length < 3)
    && (emotion.key !== newKey)) {
    setEmotions((prevEmotions) => {
      const newEmotion = { emotion: emotion, rating: rating, key: newKey }
      let count = 0;
      console.log(newEmotion.emotion)
      for (let i = 0; i < prevEmotions.length; i++) {
        const previousEmotionsEntered = prevEmotions[i];
        console.log(previousEmotionsEntered.emotion)
        if (newEmotion.emotion === previousEmotionsEntered.emotion) {
          count += 1;
        }
      }

      if (count === 0) {
        return [
          newEmotion,
          ...prevEmotions
        ]
      } else {
        Alert.alert(
          "Duplicate Emotion",
          "Please enter a unique emotion",
          [
            { text: "OK", onPress: () => console.log("OK Pressed") }
          ],
          { cancelable: false }
        );
        return [
          ...prevEmotions
        ]
      }
    });
  } else {
    Alert.alert('Please enter a valid emotion and rating')
  }
}

```

Figure 55 - SubmitHandler Function

After this initial check, the emotion and rating are then attempted to be added to the emotions array. The new emotion, rating and unique key are made into an object and at the same time the current emotions in the array are called. I then created a loop to cycle through and add one to a count variable created, initialised at zero. When the new emotion is being compared to the current emotions, if the new emotion matches with any of the current emotions, the count is incremented by one. If the count remains at zero, it is a valid entry and is added to the beginning of the emotions array. If the count does not equal zero, then an alert is presented, explaining to the user why the entry is not valid i.e. that emotion has already been entered.

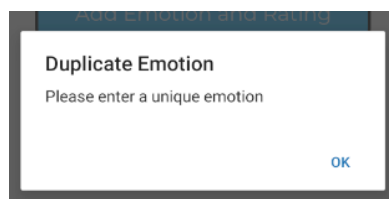


Figure 56 - Duplicate Emotion Alert

To display the emotions array list created, a flat list is used. A flat list takes an array of items and displays one after another with the stylings created. To style the emotions in an efficient way, I created another custom component – EmotionItem.

```
export default function EmotionItem({ item, pressHandler }) {
  return (
    <View style={styles.container}>
      <Text style={styles.item}>{item.emotion}</Text>
      <Text style={styles.item}>{item.rating}</Text>
      <View style={styles.icon}>
        <MaterialCommunityIcons
          name="trash-can-outline"
          size={24}
          color="black"
          onPress={() => pressHandler(item.key)}
        />
      </View>
    </View>
  )
}
```

Figure 57 - EmotionItem Display

This takes the array of emotions and displays the emotions and rating inside a container with a bin icon next to it. The significance of the bin icon is it invokes a custom function `pressHandler` using the emotion's unique key to identify which row has been selected.

```
Alert.alert(
  "Delete Emotion",
  "Are you sure you want to delete this emotion?",
  [
    {
      text: "Cancel", onPress: () => {
        console.log("Cancel Pressed")
      }
    },
    {
      text: "Yes", onPress: () => {
        setEmotions((prevEmotions) => {
          return prevEmotions.filter(emotion => emotion.key !== key)
        })
      }
    }
  ],
  { cancelable: false }
);
```

Figure 58 - Remove Emotion Alert

This function warns the user they're about to delete an emotion and if they want to proceed. If the user selects 'Ok', then the `filter` method is used on the current array, creating a new array, filtering out the emotion row that has been removed using the corresponding key.

Below is an example of how this would be displayed to the user.

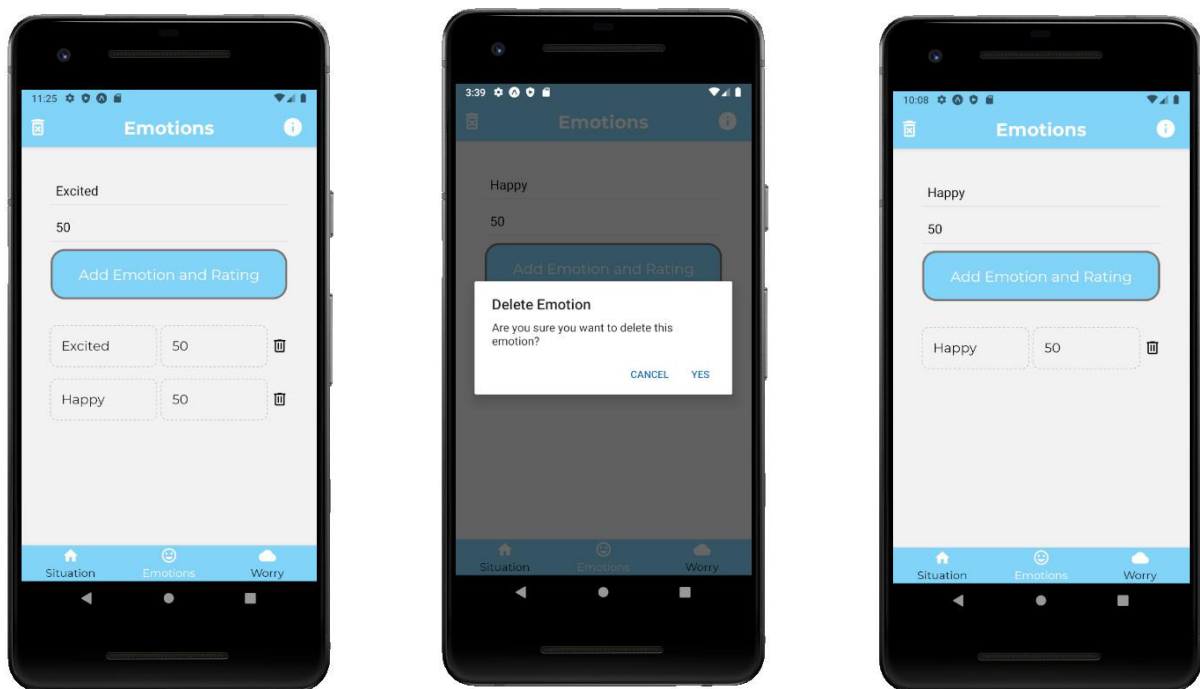


Figure 59 - Remove Emotion Full Example

Throughout this process, the `useEffect` hook has been working with `AsyncStorage` to keep the current information being inputted up to date. This is the same technique used on the situations page and the only purpose of this is for demonstration purposes only. Given more time, this information would have been processed into the 'emotions' section of the `UserSchema` blueprint.

### Worry

The worry screen is the final tab available to choose and the final entry point for the user. Similar to the situation page, this page is predominantly made up of text input boxes. There is one input section where the requirement is to recognise the type of worry and so the user can only choose from one of two options: hypothetical or practical. Originally, this was going to be done using radio buttons but due to the complexity of implementing this on React, I opted for the built in React Native component 'Picker'.

```
<Picker
  style={styles.input}
  selectedValue={typeofWorry}
  onValueChange={({itemValue, itemIndex}) =>
    |   changeTypeOfWorryHandler(itemValue)} >
  <Picker.Item label='Please select a type of worry...' value='' />
  <Picker.Item label='Hypothetical' value='Hypothetical' />
  <Picker.Item label='Practical' value='Practical' />
</Picker>
```

Figure 60 – Picker Component

This component displays the current value to the user, starting off with 'Please select a type of worry' which matches the `useState` value of an empty string, or `''`. When the value is



changed, the `changeTypeOfWorryHandler` is called and updates the value to the corresponding picker item value. As mentioned earlier, this is using the `useState` hook and taking advantage of the real time update on value changes from the `onValueChange` property of the `Picker` component. This value is now stored.

Below is how this information is displayed to the user.

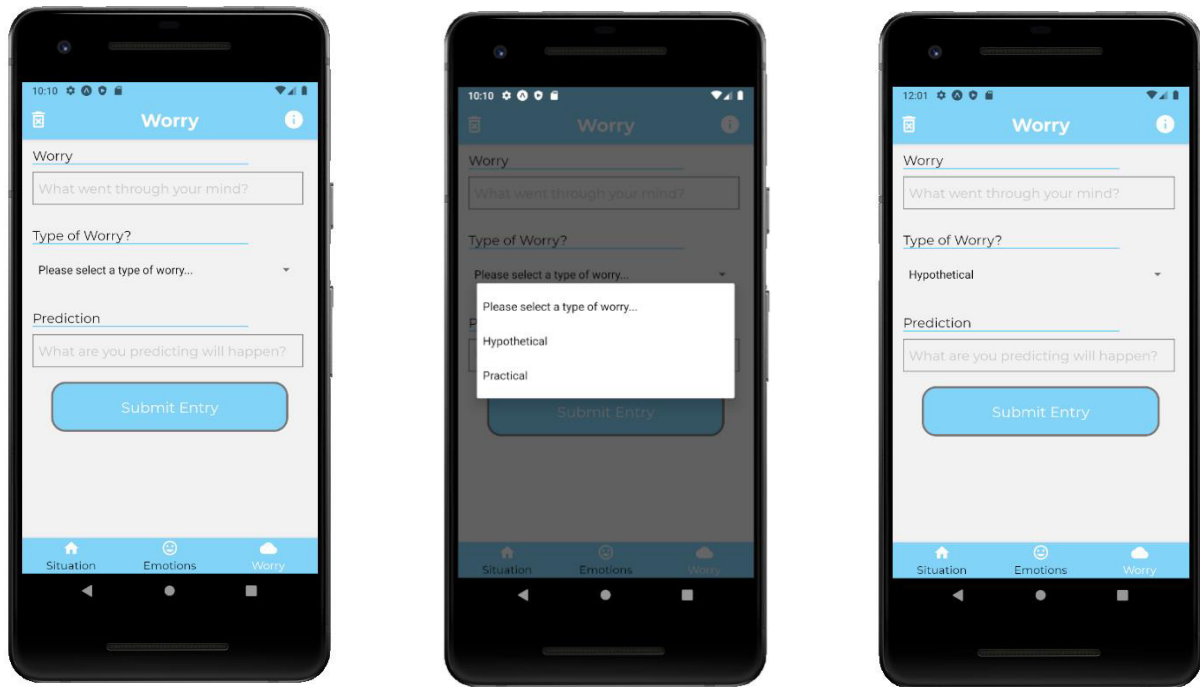


Figure 61 - Full Picker Example

Located at the bottom of the worry tab screen is the final submit button to commit the entry to be pushed into the diary entries array. This is a design choice which may confuse the user at first by causing them to wonder where they should go to locate the final entry submission. To first locate this, the user must click each tab of the form and explore the different screens. By not being able to submit the diary entry on every tab will prevent the user from rushing to complete an entry, missing some of the functions of the app. When this button is clicked, an `onSubmit` function is called.

```
const onSubmit = async () => {  
  try {  
    const entryTitle = await AsyncStorage.getItem('entryTitle');  
    const when = await AsyncStorage.getItem('when');  
    const where = await AsyncStorage.getItem('where');  
    const who = await AsyncStorage.getItem('who');  
    const what = await AsyncStorage.getItem('what');  
    const emotions = await AsyncStorage.getItem('emotions');  
    const valueEmotion = JSON.parse(emotions);  
    const worry = await AsyncStorage.getItem('worry');  
    const typeOfWorry = await AsyncStorage.getItem('typeOfWorry');  
    const prediction = await AsyncStorage.getItem('prediction');
```

Figure 62 - onSubmit Full Entry

The onSubmit function uses AsyncStorage to retrieve every item currently stored inside the local storage and performs a simple validation check on every value. This final validation check before submitting an entry first checks to see if either the entry title or the date is empty. If either of these two fields are empty, the user will not be able to submit the form as these are the minimum required input fields. If any other field is empty, a warning is displayed to the user asking them if they still wish to proceed with the submission of the entry.

```
if ((entryTitle == null)
    || (when == null)) {

else if
    ((where == null)
    || (who == null)
    || (what == null)
    || (valueEmotion.length == 0)
    || (worry == null)
    || (typeOfWorry == null)
    || (prediction == null)) {
```

Figure 63 - Entry Validation Check

This is how the information will be displayed to the user regarding any alerts sent from the validation.

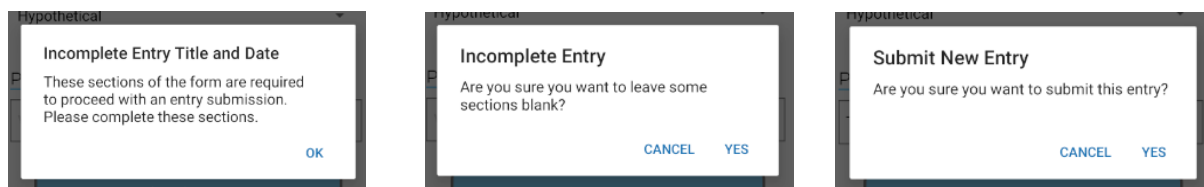


Figure 64 - Full Entry Validation Alerts

Once the user submitted an entry, they will be taken to a final preview screen which displays all the information they have inputted into the format of a single entry.

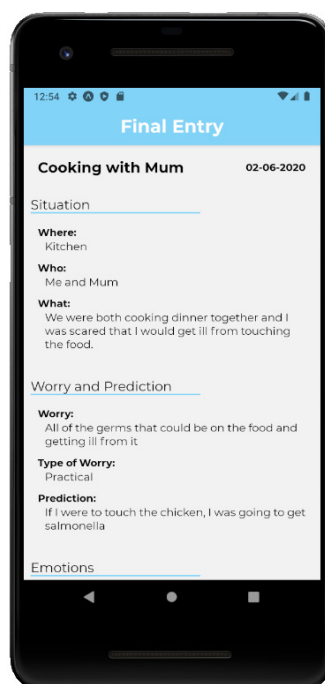


Figure 65 - Full Entry Preview

## Final Entry Preview

This screen retrieves all of the information from AsyncStorage via the getItem method, using the string value we assigned it to from the screen it was collected from. By using the same combination of useEffect, useState and AsyncStorage, I am able to demonstrate what the application could achieve given a functioning route to the database.

```
useEffect(() => {  
  _retrieveData()  
  setLoading(false)  
}, []);
```

Figure 66 - Final Entry Preview useEffect Function

In this instance, I have placed an empty array as the second parameter for useEffect. This stops the useEffect from constantly being called, taking up valuable memory resources whilst visiting this page. Below is an example layout of what a final entry would consist of through the whole form.

The figure displays five mobile app screens arranged in two rows. The top row shows the 'Situation', 'Emotions', and 'Worry' screens. The bottom row shows two views of the 'Final Entry' screen.

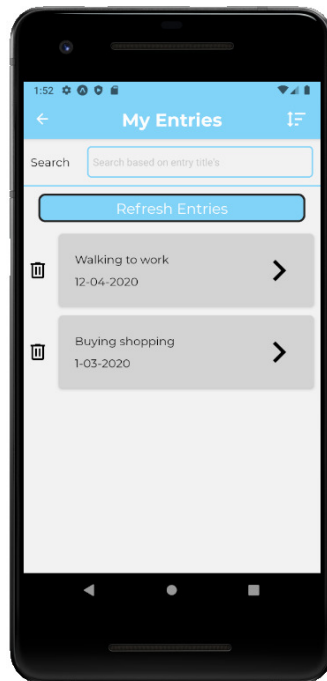
- Situation Screen:** Contains input fields for 'Cooking with Mum', 'When Did This Happen?' (02-06-2020), 'Where Did This Happen?' (Kitchen), 'Who With?' (Me and Mum), and 'What Happend?' (We were both cooking dinner together and I was scared that I would get ill from touching the food).
- Emotions Screen:** Contains a text input for 'Emotion (What were you feeling?)', a 'Rating (0-99)' field, an 'Add Emotion and Rating' button, and two rows of emotion/rating pairs: 'Cautious' with '20' and 'Scared' with '80'.
- Worry Screen:** Contains a 'Worry' text input (All of the germs that could be on the food and getting ill from it), a 'Type of Worry?' dropdown (Practical), a 'Prediction' text input (If I were to touch the chicken, I was going to get salmonella), and a 'Submit Entry' button.
- Final Entry Screen (Left View):** A summary screen titled 'Cooking with Mum' with date '02-06-2020'. It lists 'Situation', 'Where:', 'Who:', 'What:', 'Worry and Prediction', 'Worry:', 'Type of Worry:', 'Prediction:', and 'Emotions'.
- Final Entry Screen (Right View):** A summary screen titled 'Final Entry' showing a preview of the data entered, including 'Worry and Prediction', 'Worry:', 'Type of Worry:', 'Prediction:', and 'Emotions'.

Figure 67 - Full Entry Example

## My Entries

### *View an Entry and Publish*

From the home screen, there is an option to view 'My Entries'. Inside this section, you are presented with a full list of your entries. For demonstration purposes, I have added some data inside the array which is used to display the information already to show the desired outcome.



*Figure 68 - My Entries Screen*

To display the entries, I am using the same method used to display information from the emotions tab. This was achieved by taking an array of entries and using that as the data input of a flat list. Continuing to use the same method as emotions, I created a custom component called 'EntryItem' which takes in a single item from the entries array, a pressHandler function and the navigation object used to navigate to a new screen.

```
<FlatList
  data={entries}
  renderItem={({ item }) => (
    <EntryItem item={item} pressHandler={pressHandler} navigation={navigation} />
  )}
/>
```

*Figure 69 - Entries Flat List Display*

Inside the component, I have made a custom Card component to style the box and give the user the illusion that it is interactive, as well as displaying the corresponding entry's title and input date, using the items within the component to pass as props to the component itself.

```
export default function Card(props) {
  return (
    <View style={styles.card}>
      <View style={styles.cardContent}>
        { props.children }
      </View>
    </View>
  )
}
```

Figure 70 - Custom Card Component

The entry card has a TouchableOpacity applied to it so when it is pressed, a method is called. This is vital to be able to actually view the entry in its full form.

```
<TouchableOpacity onPress={() => navigation.navigate('SelectedEntry', item)}>
```

Figure 71 - Select Entry Button

When the Card component is pressed, it navigates the user to the 'SelectedEntry' page, passing the item object into the screen with it. This means that the page being navigated to will have access to all the properties within the item object by setting the parameters of the current route to their designated values and can display it.

```
const { entryTitle, when, where, who, what, emotions,
      worry, typeOfWorry, prediction } = route.params;
```

Figure 72 - Route Values

Once the user navigates to their entry of choice, they are shown the entry in full.

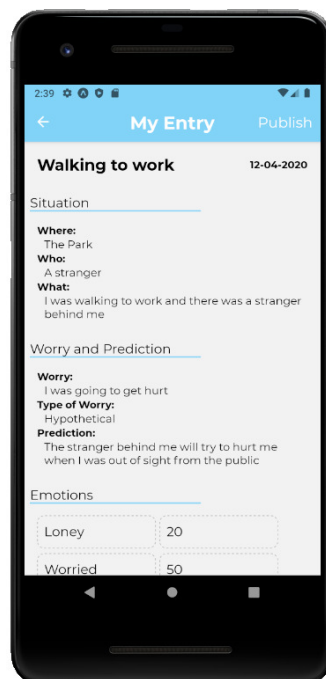


Figure 73 - My Entries Entry Example

Although this function has not been implemented within this project, the user would normally process any published entries through this screen by pressing 'Publish' in the top right of the header.

## Refresh Entries

This functionality would normally be used to process any changes made to the database. Since this was never fully implemented, I will be demonstrating the intended goal of this button using `useEffect`, `useState` and `AsyncStorage`.

If the user has entered a new entry, automatically retrieving it from the 'My Entries' page may cause some problems. During development, when the `useEffect` hook was used to retrieve the information automatically, empty cards would be placed onto the screen before the data had a chance to be received. To remedy this, a refresh button was put in place.



Figure 74 - Refresh Entries Button

`useEffect` is used as you access the 'My Entries' page to retrieve the entry information from `AsyncStorage`.

```
<TouchableOpacity onPress={() => newEntryHandler(entryTitle, when, where, who, what, worry, typeOfWorry, prediction, emotions, key)}>
```

Figure 75 - newEntryHandler Button

When the refresh button is pressed, the `AsyncStorage` will hold the values that are used for displaying the entries. The `newEntryHandler` function is called using these values as the input.

```
const newEntryHandler = (entryTitle, when, where, who, what, worry, typeOfWorry, prediction, emotions, key) => {  
  //Function for adding a new entry to the entries list  
  if (entryTitle !== null) {  
    setEntries((prevEntries) => {  
      const newEntry = {  
        entryTitle: entryTitle, when: when, where: where, who: who, what: what, worry: worry, typeOfWorry: typeOfWorry, prediction: prediction, emotions: emotions, key: key  
      }  
      return [  
        newEntry,  
        ...prevEntries  
      ]  
    })  
  } else {  
    Alert.alert('No new entries')  
  }  
}
```

Figure 76 - newEntryHandler Function

This function is used to first validate that there is a new entry within `AsyncStorage`. Due to the entry title being a required entry from the 'Add New Entry' functionality, checking if this value is null is the best way to check for a new entry waiting to be inserted. If there isn't a new entry, the user is presented with a simple alert saying 'No new entries'. If there is a new

entry available, the entry information is placed inside a new object and is placed at the top of the previous entries listed.

### *Delete an Entry*

To delete an entry, I am once again using the same technique used when deleting an emotion. The entry key is identified from within the EntryItem component and is used as the argument to call the pressHandler function.

```
<MaterialCommunityIcons
  name="trash-can-outline"
  size={30}
  color="black"
  onPress={() => pressHandler(item.key)}
/>
```

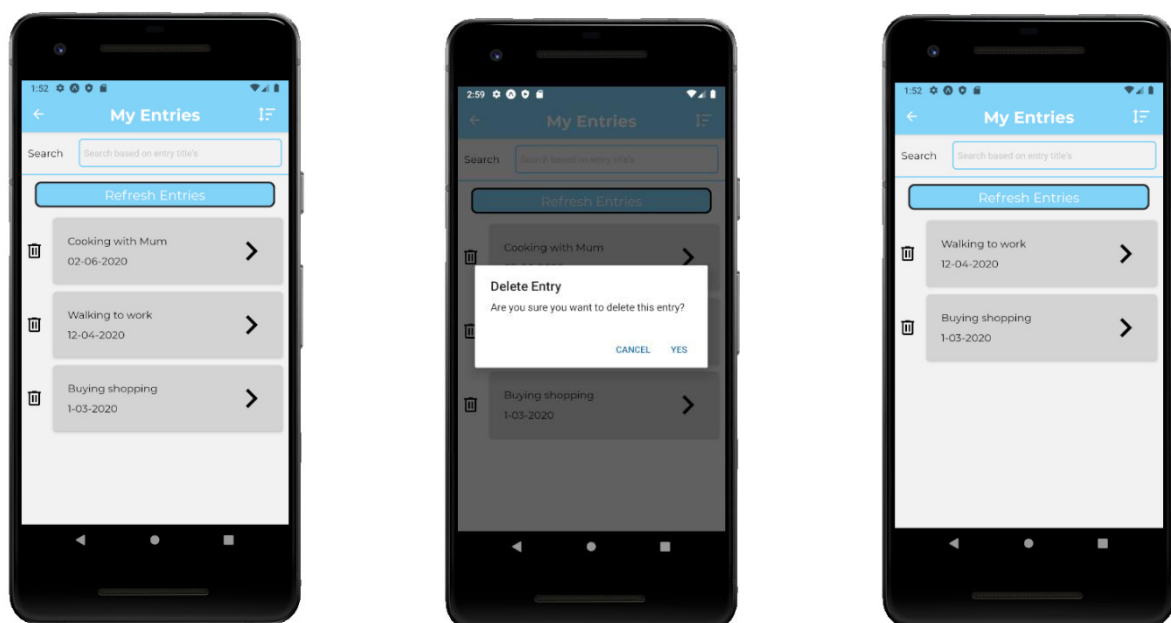
*Figure 77 - Delete Entry Icon*

The pressHandler function alerts the user that they are about to delete an entry and asks for their confirmation. Once the user has confirmed the entry to be deleted, the filter method determines which of the current entries in the list has the corresponding key and therefore needs to be removed.

```
onPress: () => {
  setEntries((prevEntries) => {
    return prevEntries.filter(entry => entry.key !== key)
  })
  AsyncStorage.clear();
}
```

*Figure 78 - Remove Entry Function*

Here's an example of how it would look to the user.



*Figure 79 - Delete Entry Example*

### Search and Sort

Once again due to time constraints, the original functional requirements of searching and sorting entries was not able to be implemented into the application, but it has still been designed into the screen.

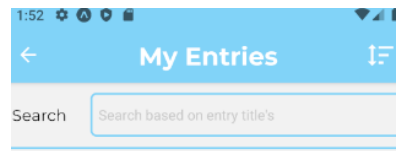


Figure 80 - Search and Sort Design

To search, the user would type text into the box provided and the entries will be proceeded to be filtered so the entry title matches the text. To sort, the user would have pressed the icon located in the top right of the screen and applied their desired sort. These sort options would have been to sort by the date of the entries i.e. latest to earliest or vice versa.

### Quick Add Entry

Currently, the Quick Add functionality does nothing apart from demonstrate how the idea of a quick entry could work by containing just the user's current emotions. This works in the same way that the emotions are inputted on the main 'Add New Entry' form, showing the users emotions as they input them in a table format and then press the 'Submit Quick Entry' button.

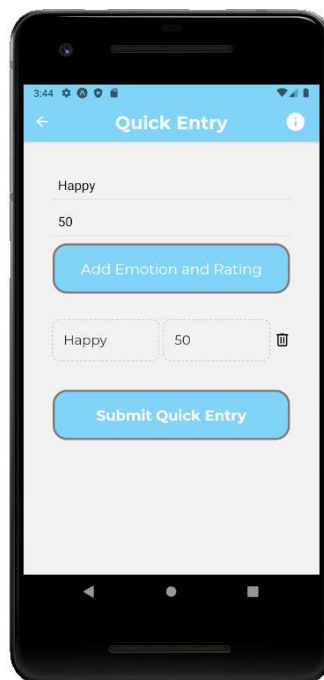
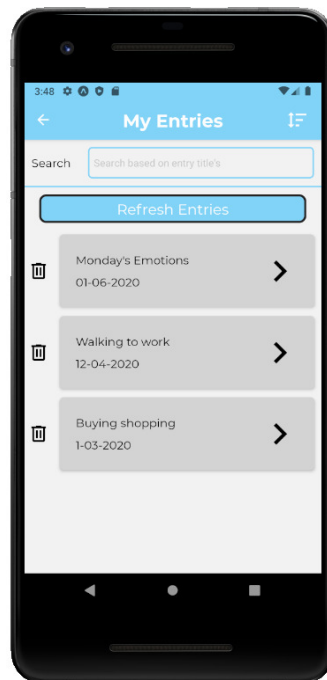


Figure 81 - Quick Add Entry Screen



The intended functionality of the 'Submit Quick Entry' button is to place it in the same location as normal entries – the 'My Entries' page. Instead of an entry title like a normal entry, it would just be a generic title of the day of the entry and the specific date underneath. Here is a mock-up as to how I would have liked it to look.



*Figure 82 - Hypothetical Quick Add Entry Example in My Entries*

## Implementation Conclusion

During the implementation processes, there were a number of struggles that I needed to overcome. This includes applying multiple navigation methods within the application i.e. drawer, stack and tabs. The documentation regarding the knowledge as to how I could incorporate this method together was lacking and so I needed to use trial and error to arrive at my desired outcome. Another struggle was the initial database setup. This proved more difficult than first imagined due to, once again, lack of documentation. React Native is a fairly new software and so help and guidance is limited at this point in time. Through tutorial videos and articles online, I managed to garner enough knowledge to get it up and running successfully.

Unfortunately, I couldn't fully achieve the database. However, through the application, the user is able to sign up a new account, adding their profile information into the database - hashing their password and assigning them a unique authentication token. To expand upon this, I would have liked to utilise the authentication token to identify users to help perform other tasks, such as adding a new entry or viewing a previous one.

Some other implementations that were originally planned but I wasn't able to fully achieve were a map visual for location and the use of an emotion wheel when the user is entering their emotion. Both of these ideas seemed to be good in the design phase but during

implementation, proved too difficult and were replaced by simpler means that achieved the same or similar results. For the location, a map and live user location was replaced by a generic description of a place. A map would have been useful to discover where the most common events occurred, but a straightforward location description does just as good of a job. For the emotion wheel, although the patients might have been able to recognise the idea of placing emotions into a diary using this technique, applying it within an application proved too laborious to use and implement. This was therefore replaced with a list and rating design that the user can freely edit, as discussed earlier.

## Results and Evaluation

### Application Outcome Compared with Requirements

The following table is taken from the requirements I put in place at the beginning of the project. I have taken the requirement number from the earlier requirements table, judged whether my solution passes the acceptance criteria I put in place and justified my decision in the final column.

Functional Requirements		
Must have		
Requirement Number	Pass/Fail	Justification
1.	Pass	The user is able to fill out a situation tab containing the ability to answer the questions: 'when?', 'where?', 'who with?' and 'what happened?'.
2.	Pass	The user is able to fill out a worry tab containing the ability to answer the question 'what if?'
3.	Pass	The user is able to fill out a worry tab containing the ability to answer the question 'what is the worry predicting will happen?'
4.	Pass	The user is able to fill out an emotions tab containing the ability to list any emotion with a corresponding rating.
5.	Pass	The user is able to fill out a worry tab containing the ability to answer the question 'What is the type of worry?' using a drop-down list.
6.	Pass	The user is able to enter the date of an entry using a pop-up calendar on the situation tab.
7.	Fail	The system has no capability to track the users live location.
8.	Pass	The user is prompted on submission if areas of the entry are left blank.
9.	Pass	The user is required to set up an account before gaining access to the application.
10.	Fail	The user is able to view previous entries but has no option to edit them.
11.	Fail	The user is able to choose to publish an entry but the functionality for this was never implemented.
13.	Fail	The user is unable to add photographs to their entry.
Should have		
14	Fail	This functionality was attempted but unfortunately could not be implemented in time.
15.	Fail	This functionality was attempted but unfortunately could not be implemented in time.
Could have		

16.	Fail	There is no system in place to remind the user to complete an entry
<b>Will not</b>		
17.	Pass	Due to the way the database has been implemented; the system would not allow other users to view separate user entries.

<b>Non-Functional Requirements</b>		
<b>Must have</b>		
Requirement Number	Pass/Fail	Justification
1.	Pass	The database has the ability to be updated when a new entry is entered, though the front end of the application doesn't offer this functionality.
2.	Pass	Entries are always accessible for the user through local storage.
3.	Pass	A help screen is provided for screens with lots of information or interactions which may be confusing for the user.
4.	Pass	Through heuristic evaluation and user testing, the application has been identified to be easy to navigate.
<b>Should have</b>		
5.	Pass	All actions performed on the application happen with a quick, seamless response time.
<b>Could have</b>		
6.	Fail	The user has no option to change the look or theme of the application.

### Breakdown of Results

Type of Requirement	Total Requirements	Successful Requirements	Requirement Success Percentage
<b>Functional</b>	17	9	53%
Must	13	8	62%
Should	2	0	0%
Could	1	0	0%
Will Not	1	1	100%
<b>Non-Functional</b>	6	5	83%
Must	4	4	100%
Should	1	1	100%
Could	1	0	0%
<b>Total</b>	23	14	61%

Above is a detailed table breaking down the individual sections which were in my original requirements table. Of the total 23 requirements listed, only 14 were successful. This is a 61% pass rate. This pass rate is a lot lower than expected prior to implementing this application. I believe this is due to the external pressures that were not accounted for during the planning of this project. This is discussed in greater detail further on in the report.

Upon reflection, I consider some of the 'must' requirements which I failed to pass were incorrectly assigned and were not crucial to the success of the project. Therefore, these should have been placed in different requirements sections, i.e. 'should' or 'could'.

## User Testing

To gather the most accurate results, I was hoping to test my application using a handful of anonymous clinical patients gathered by Dr Cheryl Jones. This data could be dummy data or real entries. Due to the anonymity, I would have never known if the information inputted was real, nor would I have known who entered it. These users would have informed me, as the researcher, how prolific they are with mobile applications and if they are avid diary users – aiming to find similarities to my previously created personas. This would have made my results more reliable, comparing the outcome of the tasks given the users previous experience with applications and diary concepts of this nature. Unfortunately, due to the current, unprecedented situation we find ourselves in, this was no longer possible. Instead, I have had to perform the test with only the two individuals that I am currently residing with. Before the testing begins, I will go through the briefing form which informs them of the overall goal of the test, as well as asking them to fill in a consent form.

The aim of the tasks I will be conducting are to showcase the most important aspects of my application. I will be reading through the task as well as having a physical prompt in front of them, explaining again what to do. Whenever data is asked to be entered into the application, dummy information will be supplied. This is to avoid any personal information being displayed to me as the researcher.

The tasks will be timed, and I will be making notes on anything the user does that may stray from the normal activities the task is designed to do. If the user begins to struggle on any of the tasks, I will try and hint to them what they could try to progress further. If they are still struggling, I will show them anything that they need to do to complete the task and it will be classified as a 'fail.'

Following the completion of the tasks, the users are asked to answer a quick questionnaire regarding the simplicity and efficiency of the application. Then they are asked for any other comments they have about the application, from the appearance to the ideas used. I haven't prepared any questions to ask the participants as I wanted unbiased and honest feedback on any aspect of the application and not just on areas, I would have been prompting them to discuss should I have used specific questions.

Task	Create a brand-new profile, login and view this information on the profile page
Task Number	T1
Description	<p>Your task is to create a new profile, login with this information and find the location of where to view this profile information.</p> <p>To avoid personal data being inputted, the following data will be used as dummy data for input: Email: <a href="mailto:TestUser@cardiff.ac.uk">TestUser@cardiff.ac.uk</a></p>

	Password: Password01 Full Name: Test User
--	--

Task	Create a new diary entry
Task Number	T2
Description	<p>Your task is to create a new diary entry, filling in every input box available.</p> <p>To avoid personal data being inputted, the following data will be used as dummy data for input:</p> <p><b>Entry Title:</b> A day at the park.  <b>When:</b> Current date of performing the task.  <b>Where:</b> The Park.  <b>Who:</b> Me, mum and a stranger.  <b>What:</b> We went to the park for a walk and there was a stranger behind us as we were walking. They got too close for comfort and it triggered my anxiety.  <b>Emotions and Rating:</b> Anxious - 55, Worried - 20 and Scared - 80.  <b>Worry:</b> That the person behind us would get close enough to be able to cause either of us harm.  <b>Type of Worry:</b> Practical.  <b>Prediction:</b> Either me, my mum or both of us will be attacked by the person behind us.</p>

Task	View the diary entry you created and publish it locally
Task Number	T3
Description	Your task is to locate where the previous diary entries are stored, locating the entry you created in the previous task. Once you have located your entry, publish it.

Task	Delete a diary one of the test entries within the 'My Entries' page
Task Number	T4
Description	Your task is to delete one of the entries that you didn't create within the 'My Entries' page.

Task	Add a new 'Quick Entry'
Task Number	T5
Description	<p>Your task is to locate where to add a quick entry, which consists of just emotions, and add them to your entries.</p> <p>To avoid using personal information, please use this dummy information provided:</p> <p><b>Emotions and Rating:</b> Anxious - 55, Worried - 20 and Scared - 80.</p>

## User Test Results

Here are the results from the two participants, performing the five tasks with little guidance to simulate a new user using the application for the first time.

<b>Task Number</b>	<b>T1</b>
<b>Participant Number</b>	<b>P1</b>
Task Success	Pass
Task Comments	User started the task by trying to log in before signing up. Maybe the application should make it more obvious that a new account needs to be created by suggesting they create a new profile. After this hiccup, the user was able to successfully log in and quickly discovered the profile page.
Time Elapsed (minutes)	2:03
<b>Participant Number</b>	<b>P2</b>
Task Success	Pass
Task Comments	The user found the sign-up form straight away and started inputting the information. Then they logged in and instantly recognised the three bars icon as a menu icon to find the profile page.
Time Elapsed (minutes)	0:49

<b>Task Number</b>	<b>T2</b>
<b>Participant Number</b>	<b>P1</b>
Task Success	Pass
Task Comments	The user discovered the Add New Entry function easily from the home page. User was confused with the tabs at first but easily became accustomed. The emotions button was pressed before any input was entered but the alert presented to the user told them exactly what went wrong. There were no issues regarding the submit entry button.
Time Elapsed (minutes)	4:23
<b>Participant Number</b>	<b>P2</b>
Task Success	Pass
Task Comments	The user asked about the difference between a quick entry and add a new entry. Once this was clarified, they clicked the correct button and proceeded onto inputting the information. This user was confused at the tabs on the bottom first but became quickly accustomed.
Time Elapsed (minutes)	2:22

<b>Task Number</b>	<b>T3</b>
<b>Participant Number</b>	<b>P1</b>
Task Success	Pass

Task Comments	This task was left vague on purpose to see what a completely new user would do, simulating a new patient. The user saw the dummy entries in the 'My Entries' section and was confused as to where the entry they just created was. After a few seconds passed, they noticed the refresh button and the entry appeared. They selected this and found the publish button to simulate the diary being made available to the therapist.
Time Elapsed (minutes)	0:22
<b>Participant Number</b>	<b>P2</b>
Task Success	Pass
Task Comments	The user clicked on a pre-existing entry and published that instead of the one they created previously in the task above. When asked if that was the correct entry, they then realised they had to refresh the entries before being able to retrieve theirs. After this, they were quickly able to understand what to do.
Time Elapsed (minutes)	0:26

<b>Task Number</b>	<b>T4</b>
<b>Participant Number</b>	<b>P1</b>
Task Success	Pass
Task Comments	The user located the 'My Entries' page and using recognition over recall, identified the bin icon meaning the entry would be deleted if this were pressed.
Time Elapsed (minutes)	0:07
<b>Participant Number</b>	<b>P2</b>
Task Success	Pass
Task Comments	The user easily recognised the bin icon as a way to remove the entry from the entries list.
Time Elapsed (minutes)	0:04

<b>Task Number</b>	<b>T5</b>
<b>Participant Number</b>	<b>P1</b>
Task Success (minutes)	Pass
Task Comments	The user asked if the 'Quick Entry' was a voice input due to the icon looking similar to the curved waves of a speaker icon. The user quickly found where to enter the quick add entry section.
Time Elapsed (minutes)	0:43
<b>Participant Number</b>	<b>P2</b>
Task Success	Pass
Task Comments	The user quickly found the 'Quick Entry' section and began entering the provided emotions.



Time Elapsed (minutes)	0:38
---------------------------	------

Task Questionnaire	Strongly Disagree	Disagree	Neither Agree nor Disagree	Agree	Strongly Agree
	1	2	3	4	5
I found signing up for a new profile and logging in simple and efficient				U2 U1	
I found that filling in the situation page using this application was easy to follow				U2	U1
I found that filling in the emotions page using this application was easy to follow and I could enter the correct information			U2		U1
I found that filling in the worry page using this application was easy to follow and I could enter the correct information				U2	U1
I found that finding the entry I created was simple and efficient			U2	U1	
I found that finding where to publish my entries efficient					U2 U1
I found that deleting an existing entry was easy to locate					U2 U1
I found that adding a 'Quick Entry' was easy to locate and efficient to use					U2 U1

Overall, the participants agreed that the application was simple and efficient, with the majority of the questionnaire being in the strongly agree section – 9/16 votes. The poorest task question was asking about the simplicity of finding the entry you just created within the 'My Entries' section. This is due to the fact that it is a manual refresh button, as opposed to an automatic refresh which would have been preferred for both users.

### Additional Comments

To summarise the general comments made from both users they found the application to be simple, yet effective. They weren't left confused by any icons or processes needed to achieve certain tasks. They also were impressed with the depth of the entry form. Both users didn't think a single occurrence of an event could have so much information to be taken from it and analysed.

A few improvements suggested included an instantaneous log in when you sign up for the first time to improve the efficiency of the initial sign up. Another improvement suggested is when a new entry is added on either the quick entry section or the emotions tab in the add new entry section, to remove the input from the text box to save time on having to remove the text already placed within the input box before entering the next emotion.

## Future Work

Currently, my application is in a position where it is very close to functioning as it was intended when originally designed. A large factor of the incomplete aspects of the project are due to time constraints caused by external issues that were out of my hands and caused a severe dip in my mental health.

The main regret is not being able to fully implement a functioning database to manipulate the data from the front end to the back end. If I were able to set up this aspect of my project, it would be much more presentable for users to test and to talk about within this report. This would greatly improve the overall appeal of the application by having the ability to perform more extensive tests over a longer period of time.

Some functionalities stated at the beginning of the project were unable to be satisfied. These include the user's being able to use their current live location from within a map feature and the ability to edit previous entries. Regarding error prevention, being able to edit a previous entry is important to adjust anything that was inputted as a mistake after it has already been committed to the database, thus improving a user's experience by providing an efficient way to rectify mistakes. A route was initially set up for this within my app.js file but could not be implemented due to complications previously discussed. With the user's live location, this is referencing the usability of the entry form. The ability for the user to pinpoint on a map the location of the event, rather than a broad description, is a massive help for the therapist to review and analyse behavioural patterns. As previously mentioned, I have provided a temporary alternative until I am able to fully implement this feature.

Another main aspect which I have alluded to throughout the report is a therapist view. Although this was never an intention to develop within this project, it was important to show what that would have done in terms of interacting with patient data – hence showing the functional requirements, non-functional requirements and use cases earlier within the report. For clinical tests to be a possible option in the future, this side of the application would need to be developed. Each therapist would have an assigned list of patients stored in a database, where the patient information is only accessible by accessing the application with the corresponding therapist credentials. It is important to stress this as if therapists had access to other patient's information there could be a breach in the patient, doctor confidentiality. This is even more important considering the latest controversy of GDPR (General Data Protection Regulation) and data being shared without consent.

A further future implementation that would improve the re-usability of this application would be the ability to customise the diary entry form layout. By making this section of the application fully customisable, it would cater to a much broader audience. From a user standpoint, default questions asked may not fully benefit the patient. In terms of the clinical aspect, a therapist could change the questions asked to mould the app for certain disorders without having to use a different software. This idea was discussed in meetings, however

since there was never an intention to develop a therapist view alongside this project, the idea never came to fruition.

Similarly, user evaluation and analysis of their own diary entries is an important factor in improving CBT process. A functionality to provide a close examination of a dataset from the user to discover details, such as the average emotion entered, average overall rating of emotions or even common words or phrases used throughout a diary entry. In my opinion, this would have greatly improved the self-reflection a user could provide themselves without having to wait to talk to their therapist. Also, the analytics of diary entries can help aid the therapist in guiding a patient's therapy.

## Conclusions

The main intention of this project was to create an application that a therapist can deploy for cognitive behavioural therapy patients to fill out in a simple diary format, utilising the cross-sectional formulation – also known as the ‘Hot Cross Bun’ diagram. This diary application is a convenient replacement to the paper diaries already in place, reducing the number of patients who fail to fill out the diaries given to them currently.

Throughout the project, a user driven approach was always the most important aspect at each stage to be more accessible and approachable for the average day to day user – along with the patients intended to use the app. From the usability of the application, making sure that it was simple to use and caused no confusion amongst users, to the style of the app. Making sure to use attractive, calming colours with fonts that made it welcoming to use, improving the overall experience more enjoyable for the user. By maintaining this focus, I was able to create a more efficient and appealing application that anyone would be able to use throughout their CBT and in life for self-reflection.

Although the app was targeted for CBT patients, given the social focus of maintaining one's mental health along with their physical health there is no reason anyone can not benefit from this app. Utilising the main features to reflect on situations throughout the day and taking advantage of the depth that the entry form allows you to go into is a great way for the public to be able to reflect on their day and wind down ready for a more restful night.

As mentioned in my user testing, both participants stated how going forward they will invest their time to start using diary applications in their day to day lives. This shows the flexibility my application offers and the growing interest from users outside of CBT patients that in the past may not have normally considered downloading these types of apps.

There were many issues I had to face during this project. The most prevalent issue being the outbreak of the global pandemic, COVID-19. Because of this, I was no longer able to receive the best support possible from the resources available at university to help guide my project. Another issue was during the months of March and April, I fell into a state of sadness due to numerous external happenings that were out of my control and therefore I didn't feel well in myself. Thankfully, through the support of my friends, family and mentor, I was able to pull myself out of that hole and progress effectively to produce the best

application I could. Given the new time constraint I set upon myself, and the challenge I had presented myself by choosing a format that I had very little knowledge of, implementing a MongoDB within a mobile application and setting up my database became an increasingly worrying task. Eventually, I managed to sort out a part solution showing what the potential that the application could achieve if I had more time.

Although some features were not implemented in a way that was originally intended, I still managed to meet most of the requirements throughout the application, and for that, I can be proud of the outcome I produced.

## Reflection on Learning

The main reflection to take away from this project is to try to fully understand the facilities that software can offer before diving in too deep with development. The choice to use React Native stemmed from the attractiveness of developing for both Android and iOS at the same time without having to re-write code for a specific operating system. Since this framework is very early in development in terms of its current competitors, the lack of support online for certain functionalities grew more and more present as I progressed through the project.

I have always been interested in learning about mobile app development and wanted to take on a new challenge with this project. My lack of experience was apparent at the beginning but grew more proficient as the days passed to the point where I am confident with how my application works. I am also hoping to continue this project, maintaining any software changes made from React Native and building it into a fully functioning application.

I also learnt that I have the capability to plan efficiently and accurately an extensive project to a high standard. When proposing the initial plan, I thought that I would veer off at some points, leaving the project to be a mad dash to the finish. Even with the stumble in the middle of development, I was able to perform many aspects very well:

- Steady learning with React Native to not have everything being implemented at once, confusing myself with what component achieves what outcome.
- Ensuring to produce many design iterations and critically evaluating them, taking the good points forward and removing the bad points.
- As I progressed through the application, concurrently progressing through the report. This would mean that my thoughts were fresh in my mind and I wouldn't have to try to recall my thoughts at the time from memory as I would have should I have chosen to write it later on.

Another personal learning reflection is to be less critical of myself. I repeatedly gave myself a hard time for the level of work I was achieving. This caused more unnecessary stress for myself, however, learning to understand these situations without getting overwhelmed has become a huge step forward in my personal progression as a soon to be graduate. Given the current situation the world finds itself in, and the dip in my mood, I should be proud of how my application ended up and where I find myself as a person. I am proud.

## References

- [1] NHS, "Cognitive behavioural therapy (CBT)," 16 July 2019. [Online]. Available: <https://www.nhs.uk/conditions/cognitive-behavioural-therapy-cbt/>. [Accessed 15 May 2020].
- [2] A. Jain, "10 Heuristic Principles – Jakob Nielsen's (Usability Heuristics)," UX NESS, 10 February 2015. [Online]. Available: <http://www.uxness.in/2015/02/10-heuristic-principles-jakob-nielsens.html>. [Accessed 31 May 2020].
- [3] I. Sacolick, "What is agile methodology? Modern software development explained," InfoWorld, 25 February 2020. [Online]. Available: <https://www.infoworld.com/article/3237508/what-is-agile-methodology-modern-software-development-explained.html>. [Accessed 5 June 2020].
- [4] J. R. a. S. M. Mike Barker, "Practical techniques in sequence control design," in *Practical Batch Process Management*, Newnes, 2005, pp. 70-85.
- [5] GURU99, "What is MongoDB? Introduction, Architecture, Features & Example," GURU99, 4 May 2020. [Online]. Available: <https://www.guru99.com/what-is-mongodb.html>. [Accessed 5 June 2020].
- [6] MongoDB, "MongoDB Atlas," 2020. [Online]. Available: <https://www.mongodb.com/cloud/atlas>. [Accessed 5 June 2020].
- [7] Express, "Routing," 4 May 2018. [Online]. Available: <https://expressjs.com/en/guide/routing>. [Accessed 4 June 2020].
- [8] N. Karnik, "Introduction to Mongoose for MongoDB," freeCodeCamp, 11 February 2018. [Online]. Available: <https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57/>. [Accessed 02 June 2020].
- [9] AXURE, "Axure RP 9.0.0.3696," 29 April 2020. [Online]. Available: <https://www.axure.com/>.
- [10] React Native, "React Native 0.62.0," 26 March 2020. [Online]. Available: <https://reactnative.dev/>.
- [11] J. Ulanovsky, Artist, *Montserrat*. [Art]. 2017.
- [12] L. Lubos, "The Role of Colors in Stress Reduction," Liceo Journal of Higher Education Research, July 2012. [Online]. Available: 10.7828/ljher.v5i2.39. [Accessed 20 May 2020].
- [13] Guru99, "Node.js MongoDB Tutorial with Examples," Guru99, 16 May 2020. [Online]. Available: <https://www.guru99.com/node-js-mongodb.html>. [Accessed 31 May 2020].
- [14] T. Gray, "MongoDB Atlas: What, Why?," OPTIMAL, 4 July 2018. [Online]. Available: <https://optimalbi.com/mongodb-atlas-what-why/>. [Accessed 3 June 2020].
- [15] mongoDB, "Sharded Cluster Components," 4 February 2020. [Online]. Available: <https://docs.mongodb.com/manual/core/sharded-cluster-components/>. [Accessed 3 June 2020].

- [16] A. Johari, "What is a Schema in SQL and how to create it?," EDUREKA, 24 October 2019. [Online]. Available: <https://www.edureka.co/blog/schema-in-sql/>. [Accessed 5 June 2020].
- [17] J. Sharir, "How to Use Postman to Manage and Execute Your APIs," BlazeMeter, 19 March 2019. [Online]. Available: <https://www.blazemeter.com/blog/how-use-postman-manage-and-execute-your-apis>. [Accessed 1 June 2020].
- [18] ProgrammerBlog, "How to create a nodejs mongodb rest api and test with postman," ProgrammerBlog, 19 August 2017. [Online]. Available: [https://programmerblog.net/nodejs-mongodb-rest-api/#disqus\\_thread](https://programmerblog.net/nodejs-mongodb-rest-api/#disqus_thread). [Accessed 1 June 2020].
- [19] N. P. a. D. Mazieres, "A Future-Adaptable Password Scheme," USENIX, 28 April 1999. [Online]. Available: [https://www.usenix.org/legacy/events/usenix99/provos/provos\\_html/node1.html](https://www.usenix.org/legacy/events/usenix99/provos/provos_html/node1.html). [Accessed 1 June 2020].
- [20] Defuse Security, "Salted Password Hashing - Doing it Right," 5 June 2019. [Online]. Available: <https://crackstation.net/hashing-security.htm#salt>. [Accessed 1 June 2020].
- [21] E. Elliott, "Master the JavaScript Interview: What is a Promise?," Medium, 23 January 2017. [Online]. Available: <https://medium.com/javascript-scene/master-the-javascript-interview-what-is-a-promise-27fc71e77261>. [Accessed 1 June 2020].
- [22] Stackify, "What are CRUD Operations: How CRUD Operations Work, Examples, Tutorials & More," 2 May 2017. [Online]. Available: <https://stackify.com/what-are-crud-operations/>. [Accessed 3 June 2020].
- [23] React, "Using the State Hook," 06 February 2019. [Online]. Available: <https://reactjs.org/docs/hooks-state.html>. [Accessed 1 June 2020].