# Introduction

## Project Overview

This report outlines the approach that was taken when developing a mobile application which's design is to assist users to keep track of their mood over the month of January. The idea being that once a user downloads, installs and creates an account through the application they can then leave daily notes regarding their day .as well as rate them on a scale from “horrible” to “great”. By the end of each month they will have a graphic that is uniquely generated from the data that they have been inputting throughout the month.

## Aims and Objectives

With this application it is being targeted the identification of bad habits as well as the generation of a graphical form to display the user’s moods over the period of a month, for upcoming months, it is expected to continue the maintenance of the app by releasing updates with new monthly art to represent every new month.

However, for the aims and objectives of this project it was decided to stick with what is currently attainable and achievable by the given deadline, which was to deliver a working system with at least registration and login, graphic monthly art for the current month and finally the ability to enter data and dynamically change the graphic art based on the data entered by the user.

Beyond the deadline, some of the future aims of this project include the development of the platform and integration of subscription services to fund a specialist team that could provide feedback to the users where a given specialist would target each user individually and through in app messages or campaign emails would then provide them easily with digestible feedback that would aim to help improve their life.

## Justification of selection of Particular Technologies or Techniques

By looking at the job market it can be certainly observed that JavaScript is a widely sought after skill as a programming language that a software engineer should know. This is easily justifiable as its versatility allows for its use on web, console, desktop and mobile applications. Taking this into consideration, it seemed most useful to pick it as the main programming language for developing the mobile app as this would make learning the language, a fair amount more purposeful and enforcing.

However, JavaScript is not natively supported by Android or IOS smartphones, it so therefore requires a framework that enables this support, and for that purpose it was chosen to use React Native, which is written and maintained as an open sourced project by Facebook. The main advantages of using React Native are that, as a JavaScript framework, it facilitates the use of the same code for both Android and IOS, especially where it comes to graphical user interfaces, networking and a few other functions that do not require native support, in which case these may have to be written individually for both Android and/or IOS if the React Native framework does not support inter-compatibility right out of the gate, an example of this it could be to access the device’s storage media, Camera or GPS.

While these few functionalities that have to be written individually based on the operating system, using Java or Switft (i.e.) the vast majority of the code is shared, which makes the application easier to maintain and facilitates the scalability.

For the storage and authentication API solutions, it was chosen the use of Firebase, which is a platform developed by Google for creating web and mobile applications, one of its storage solutions is a real time NoSQL database, which is a service offered freely with a very reasonable limit, beyond this limit the costs are also fairly reasonable as they scale proportionally to the usage of the service, and on that note, since it is a service provided by Google, its scalability can accommodate just about any amount of unpredictable growth. This makes firebase the ideal candidate for start-up applications as opposed, for example, to self-hosting of services where it is required an advance investment in resources.

NoSQL databases are extremely easy to manage, maintain and scale as new attributes can be added at any time without affecting the existing data as opposed to SQL databases where in a lot of cases it would be necessary to completely modify the existing data model. This aspect is particularly important for the given project as during the inexperienced ongoing development of the app, it may be the case that new storage requirements might pop up which in a SQL database would translate into significant amounts of work and time being spent creating and optimizing the new structure of the model instead of working on more important aspects such as app features that would more likely impact the success of the app.

Lastly, the technology choices made were highly biased towards new high in-demand skills sought by employers, new open-sourced technologies that are highly regarded by the software engineering industry and towards delivering a project that in the future could easily also be adapted for iOS systems. This meant that, it was overlooked the use of old proprietary technologies such as Adobe PhoneGap as suggested by the assignment brief. Valid alternatives for the mobile framework included: Flutter, Ionic and NativeScript.

# Secondary Market Research

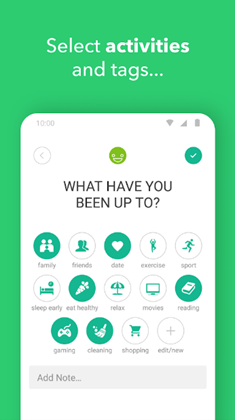
In this section it is going to be investigated what applications with similar aims have done right in order to get good user reviews and grow in popularity on the Google Play store.

Although there are a few similarities in between them, the end goal completely differs, as well as the way they are monetized. The majority of them ask for paid subscriptions or have to be bought in order to be used, whilst for ours we don’t. Insteal of doing that we provided with a donation functionality, highlighting the fact that this is a volunteer work and all the money obtained from the donations will futher be invested in the application itself.

## Daylio

Daylio is a mood/habit tracking app with a stylish modern design and very many features. The app has a mood tracker which asks the user to pick one of the five smiley faces representing moods. The mood data is then saved and over time as it is being collected, the user is shown a graphical representation of the days leading up until present, which enables them to see how their mood has progressed.

Another important feature is the habit tracker, where the users can pick from a pre-made selection of habits or tasks. Both the mood tracker and the habit tracker can then evaluate the data and display statistics, for example, on how many times the user went to sleep early in that given month and how it correlates with their moods. A diary and a journal are part of this app, however these functions are often found on pre-installed software on any mobile device and so are not considered as a big addition to the app and its features.



The monetizing strategy for this app is free to download and pay to customize. There is a large variety of customization, however, all of it is in line with the static modernist design of the app, so the user is only allowed to change icons, colours and add their own activities or habits to keep track of. The functionality of this app is its main selling point, but its design is rather overused by many other apps.

Pros:

* large variety of graphs and statistics
* easy to use interface

Cons:

* general and impersonal design
* very limited customization
* in app purchases

## Moodflow

As far as functionality goes, this app is almost identical to the previous. The only noticeable difference is its design, which uses more vivid colours and defined shapes. Additionally, this app also markets itself as a “self-improvement” app, emphasizing its planning and organizational utility.



Pros:

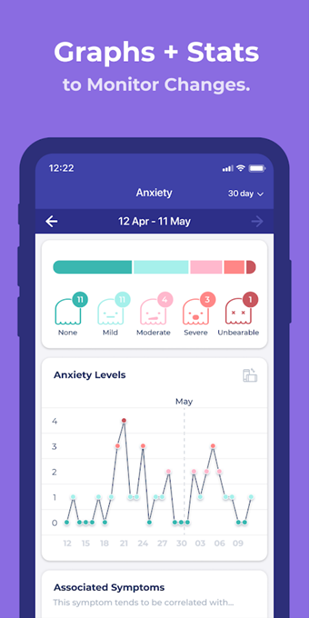
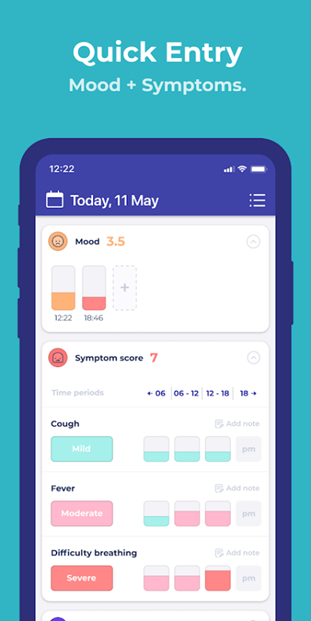
* Customizable charts and graphs

Cons:

* the interface is not very user-friendly
* the customization process is not well explained
* large amount of ads

## Bearable

In Bearable, users can keep track not only of their moods and habits, but also of symptoms when they are ill. This is done via a friendly-looking UI where the user can pick, for example, whether their fever is “moderate” or “severe”, and their current mood. This input is then processed into statistics and the user can then analyse them to find how the data collected may correlate, which for example, may enable them to see how impactful their symptoms are on their mood.



Pros:

* tracking symptoms, sleep, step count, etc
* correlates different data types into various forms of output, such as graphs

Cons:

* design is static and tied to the name/brand
* some features are not very useful (reminders or alarms)
* generic-looking graphs

# Project Constrains and Risks

Living in a technological era that facilitates human interactions means that now more than ever exist a variety of forms and means to communicate with each other, though the same technologies that bring us together have also made it tremendously more difficult for people to share their emotions.

Demographically speaking over the past years, it has been observed a great spike in cases of depression, anxiety and people that suffer of loneliness or that generally have a negative mind-set. (GANDER, 2019)  
The idea behind this application is one of collective effort, where people from a range of different industries are encouraged to collaborate and take action in helping to put a stop to negative mind-sets and doing so by bringing in their own ideas and/or programming to bring features to the app.

This is planned to be accomplished by open-sourcing this app project and opening discussions in online communities to therefore encourage any readers to join in and help deliver more feature packed updates to the app.

It is foreseen that despite the open-sourcing of the app it will still be necessary the creation of a small team composed by a few artists who are in charge of the monthly themes for the application, they would be responsible for bringing a completely new and unique design for every new month.

As previously mentioned, the initial features of the app would be developed by the group that was formed to complete this task as an assignment for the university, after that the project would be also maintained by the group but it would be open-sourced to benefit from the community and help deliver a better app for those that need it.

The main risk in this plan is whether or not the open-source community would accept to partake in helping the improvement of an app that requires a subscription to pay for the specialist team that provides the individually tailored advice for its users, however this may seem justifiable as it is otherwise very unlikely to get such a service freely, moreover, if the app starts growing in popularity too quickly it may be challenging to increase the size of the team in a short amount of time.

# Scope

For the given time it was decided to prioritize the basic features and the application itself, ensuring that an application some features fully implemented would be delivered and introduced on the Google Play store as the first build, which users could then download and check out.

This would consist of the login, register and main pages as well as some necessary subpages where the user can enter some data.

Once it is gathered a substantial of data the next step will be to look for volunteers who are willing to analyse it and write a few words for each individual, such as, on how they could improve their lifestyle and make the best out of it. Currently, the application uses a mountain graphic art to represent the month, but in the near future it is expected to have many UI improvements.

# Software Resources

As previously mentioned, the framework chosen for developing a mobile application was React and as defined in their documentation, there are multiple ways of setting up a React development environment (ReactJS, n.d.), some of the ways depend on the developer’s preference and some depend on how the capability of the developing machine itself.

Though, software wise, other than the React Native framework itself, for running the application being developed it was used an android emulator, installed with Android Studio.

To keep track of all the code iterations and to flexibly work as a team, it was used GitHub as a version control.

# Hardware Resources

For installing React Native dependencies and compiling, it is required, as a somewhat general rule of thumb, to have a semi-modern quad core CPU and almost equally important to have fast storage such as a good SSD or better yet an NVME SSD, alternatively, if this is not available, another option is to create a small RAM Disk using a utility such as “ImDisk” and temporarily store the project files on this drive in order to get fast node modules installations as well as code compilations, which otherwise may take a long time and delay the project. On a side note, if there is not enough RAM available on the system and if the system also possesses a dedicated GPU, a RAM drive can also be created on the dedicated memory, this grants speeds not typically as fast as the RAM but still significantly fast enough for the required purpose.

# Methodology

An agile approach was taken for this project, based on user reviews and the data collected and analysed, it was concluded that constant improvements would be required in order to ensure a good user experience.

The initial idea is to create a deliverable prototype and launch it as soon as possible acknowledging the fact that the very initial experience may be far from great and users might demand certain features which might take some time to implement. The development of the application will accommodate continuous integration as well as scalability.

# Stakeholders

The main stake holders are the group that developed the app, the contributors once, the app is open-sourced and the specialists team that are to be giving the users feedback once the app is complete, if the product delivered is not good or it fails in getting users, it might be that their time investment could go to waste.

# Functional Requirements

## Database connectivity

There must be some configurations done in a single place that allow for secure connections between the user app and the database.

## Login

The user must be able to log in using their registered credentials, the log in must be safely implemented and there must be error messages if something is not correct.

## Register

The user must be able to register, this data must be safely stored so that even if the authentication system gets compromised, the passwords stay protected, there must be error messages for if the email of the user has already been taken and/or if the password entered is not safe enough.

## Sign-out

The user session must be persistent and it must not force the user to sign-in every time they use the app, the app must also have the ability to sign-out.

## Persistent Storage

The data of the app must be persisted on a database and it must be retrieved and displayed when the user is using the app.

## Rate your day screen

There must be a screen that allows the user to enter data about their day.

## Auto populate the graphic art with user data

When the screen with the monthly graphic is loaded, it must be auto populated with the user data that they previously entered.

# Usability Requirements

The app was designed around the user in order for it to be user-centric, it was taken into account that the layout should feel natural and consistent and that there should be an overall colour scheme used by the whole app.

Any displayed text was sized according to what was judged to be the ideal size for the given screen and ability to read what the text says.

Forms were also intentionally kept short and concise so that the user could just fill them out and hop straight into using the app for its intended purpose.

Over the development of the app it was applied the use of on an MVC design pattern where possible, however the app is mostly composed of views (screens), and as it naturally applies to React, the logic is built straight into those views as separating it is going against the React principles for writing components.

# Identify target users

The application is addressed to the people that encounter difficulties coping with life. With the aid of the aforementioned specialists it is hoped that as long as the users are willing to try out the app as it is indented to be used and therefore follow along with the advice given, that they improve their lives.

Although they represent our main audience, the application can be used by anyone who simply wishes to keep track of their mood and habit evolution over time. Our specialists will only address the individuals that have paid for the subscription, those who haven’t will not receive any type of advice as the labour of the specialists needs to be paid for the service.

## Mobile app development

Since the assignment brief was extremely flexible in terms of what technologies can be used and what the application should be about it was started with a brainstorming session. After a number of ideas were set in stone, these were analyzed carefully and it was picked the one that was considered the one to have the highest chance of completion.

The next step was to break down the functional requirements as well as the tasks per individual, without merging conflicts. The database structure was sketched at the very beginning to ensure that all the group members had a rough idea about the plan to implement certain aspects of the project.

Once the tasks were assigned it was defined a set of rules which covered:

* the branching system
  + the use case for each developing branch
* when to merge to the main branch
* how to avoid overriding each other’s code

It was heavily relied on GitHub as a version control platform, having branches per developer minimizing the potential merge conflicts. Once a feature was marked as complete it would then be arranged to merge it to the main branch then pull it on the individual developer branches, ensuring that the codebase was majorly similar for everyone.

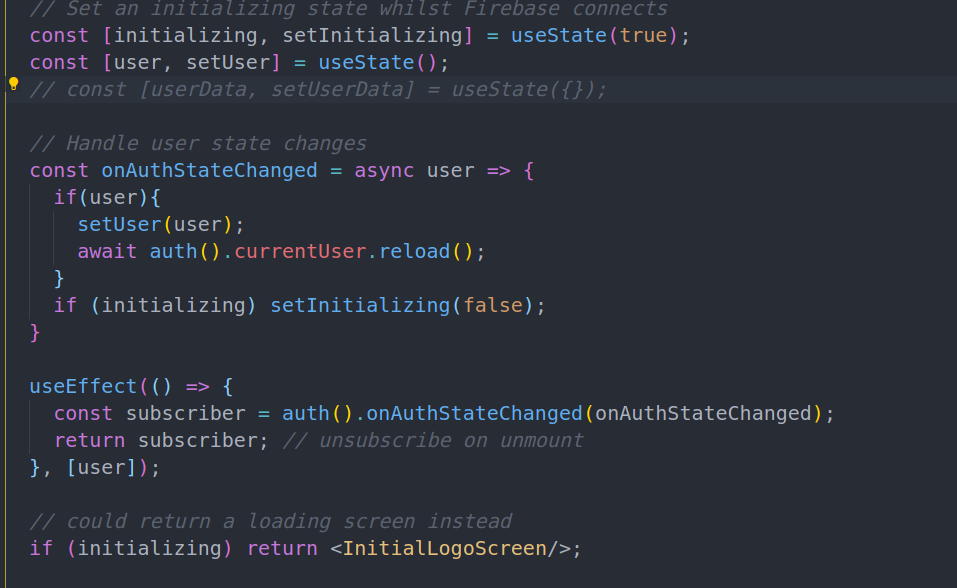
At the very start of the development process it was conducted a research on what node modules would be required for the necessary use cases.

One of the greatest problems encountered was the fact that the initial package manager used by the group was NPM, on which the firebase module was outdated at the time, which made it impossible to retrieve data from the database. To overcome that it was necessary to switch to the Yarn package manager, a different JavaScript node modules manager which had that library up to date. This solution also turned out to be faster at installing new packages and compiling code which was good news for the group as the whole process was quite long and meant that for some changes done to the codebase it would be necessary to wait a few minutes until it was known whether or not a change made has worked or has stopped the application from working.

The React Native framework also presented a few of its own issues too, such that the group would get a fair amount of odd behaviour where the changes to the codebase would sometimes not reflect themselves in the app and such would often be fixed by an application reload or by another lengthy recompilation altogether, though it was only guesswork that led the group to try these out when such queer issues occurred.

The firebase library was used to manages the data layer and keep the application and the database in sync. It also provided the project with a secure environment for user authentication and registration.

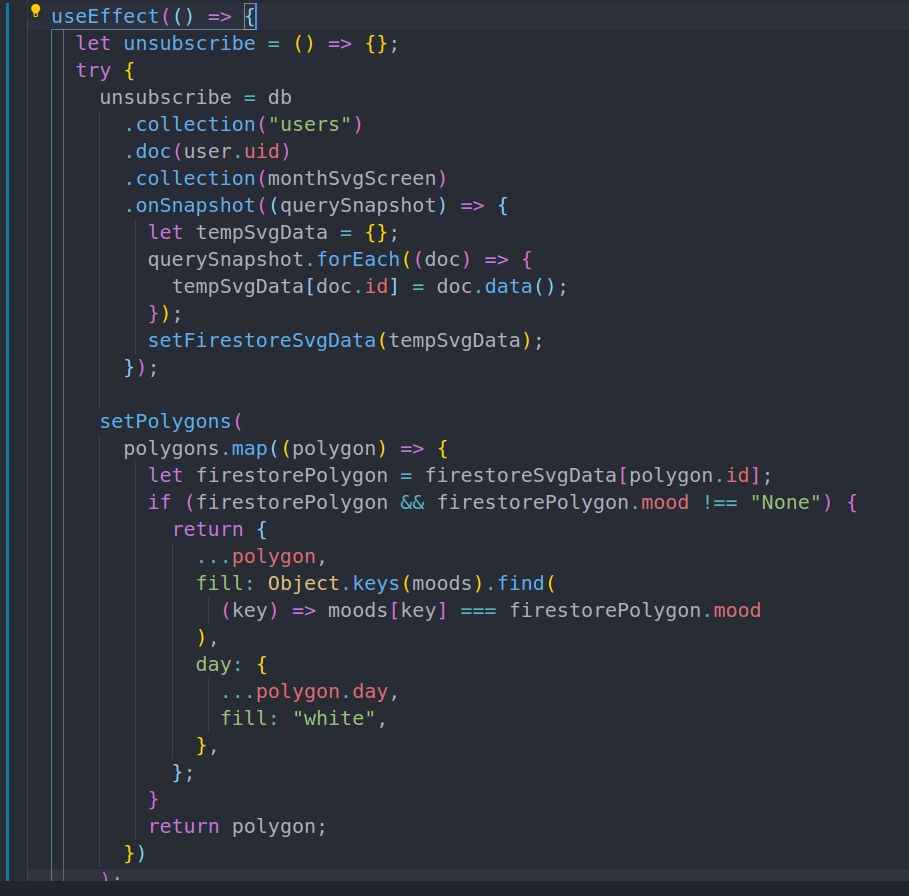
Regarding the react, it was chosen the use of functional components of class based ones simply for the fact that these are the new accepted standard and they make the code seem a lot more simple and easy to understand.



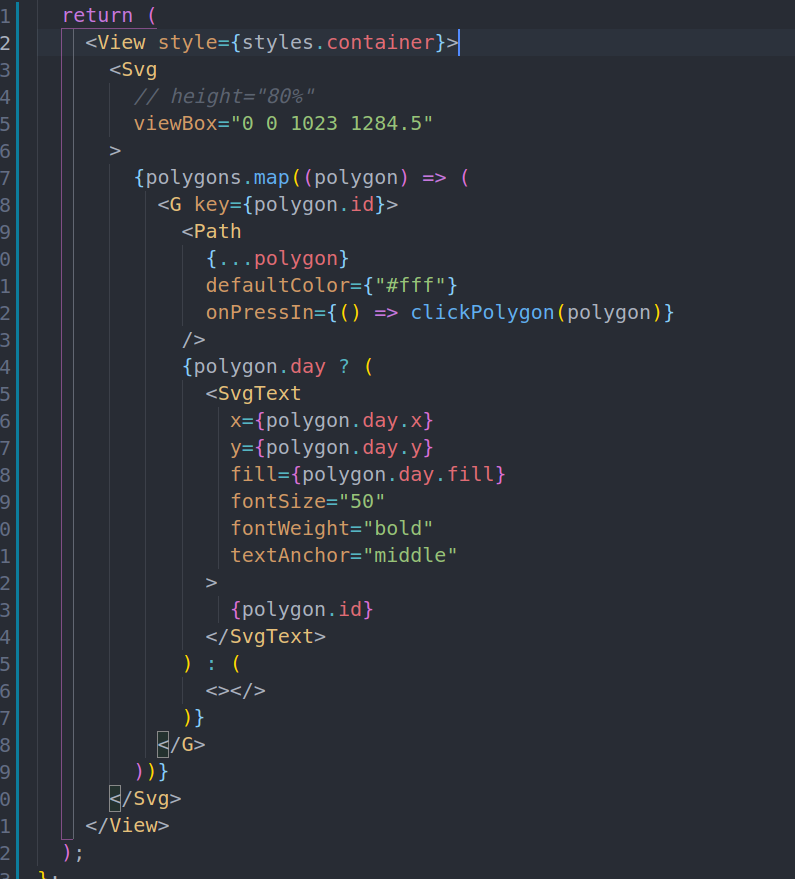
This code snippet attached above can be found in HomeScreen.js and verifies if the user is logged in or not. Based on that they will be prompted to the login/register screen or navigate to his personal page.



Above can be seen the data structure for each polygon rendered in the mood page. Where they have a unique id, the x, y coordinates which dictate where each should go in the view. The stroke, strokeLineCap and the other attributes are used for design purposes.



By default, all polygons come in a white filling, but if the user left notes or a review it is then retrieved the data from the database and render the page accordingly. That’s achieved by using useEffect, a react hook.

To render the polygons we interate through each one of them (the polygons array) using the map function, passing down the props mentioned above.

Once the user click on them they will be prompted to a new page, where they get to write a few words about their day and leave a rating.