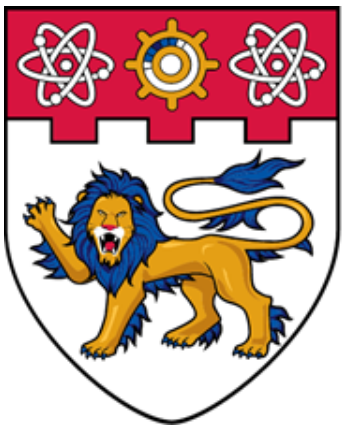


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

IM3080 Design and Innovation Project

(AY 2021/22 Semester 2)

Project Report

Title: FLOAT

Github: https://github.com/ljunqian/Float

Submitted by: Group 2

Supervisor: Chua Hock Chuan

Table of Contents

1. Background and Motivation

1.1 Background Information

In Asia, the majority of people do not understand the differences between mental wellness and mental illness, causing societal stigma when discussing one's mental health. Furthermore, this might lead to their withdrawal from society and discourage them from seeking help.

Mental wellness is the act of practicing healthy habits on a daily basis to attain better physical and mental health outcomes. It is more than the absence of mental illness. Being mentally well means that your mind is in order and functioning in your best interest. You can think, feel and act in ways that create a positive impact on your physical and social well-being.

Sustaining mental wellness requires time and effort. The more you invest in your mental wellness, the stronger it will become. We hope to lower the barrier of entry on mental wellness and make mental health more approachable.

Float is a wellness app with a focus on providing ways to develop a better you both mentally and physically as well as meaningful metrics for tracking your mental wellness journey. The app comprises short educational video clips on various topics such as stress, low self-esteem, emotional burnout, and sleep issues. Our target users will be youngsters and working adults who are consistently in a high pressure environment.

1.2 Motivation of our mobile application: Float

Our group's catalyst comes from our observation of the effects of the Covid-19 pandemic on the NTU student population's mental wellbeing. Our team collectively made a conscious decision to work on this specific project as we recognised the shortcomings of mental wellness applications available in the market. We try to tackle the low engagement and user incentives of current mental wellness apps by incorporating gamification.

2. Objective

2.1 Improve Coping Mechanism

Much of the student population lacks the correct coping mechanism to deal with daily stresses in school and are ill-equipped to take the correct steps in the direction of healthier mental wellbeing. Our app hopes to provide easy, step-by-step suggestions to get users to start on their mental wellbeing journey.

2.2 Monitoring and Tracking of User's Mental Well-Being

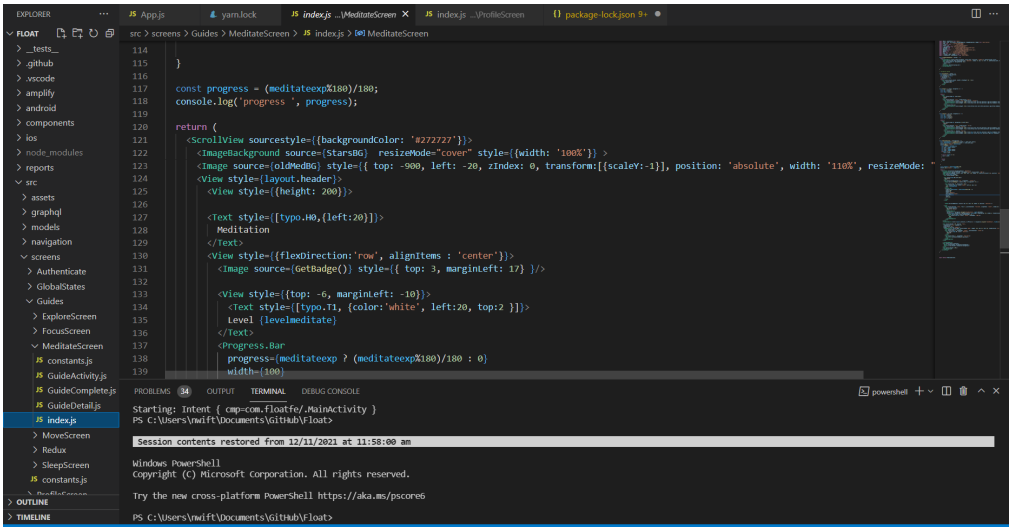
Many of the mental wellness apps in the market do not record users' mental wellness progress or display this information in a meaningful manner. Users can be disoriented on their mental wellness journey after spending the time and effort and seeing no noticeable results. Our app aims to rectify the problem by displaying the users' overall statistical journey to encourage users to maintain their good habits. At the same time, it allows users to accurately identify if their mental wellness issues require professional help.

3. Review of Literature/Technology

3.1 Frontend

3.1.1 React Native

"Learn once, write anywhere." Created by Facebook, React Native is a Javascript framework for natively rendering mobile applications. It supports cross-platform app development for both iOS and Android devices. React Native also has a large online community that provides a multitude of existing libraries and api codes to use, which encourages rapid development of Minimal Viable Product (MVP).



Written using Javascript XML (also known as JSX), it leverages the platform used to translate the developers' codes to native UI components easily, which is similar to that of HTML and CSS. Most crucially, developers can run changed features and see a direct preview of their code in real-time, which makes them less prone to making mistakes and time for debugging.

Basic core components such as button, image, touchableopacity, scrollview, etc. can be combined to create custom-made components, which go through life cycles via props and state changes in each re-render. React will trigger our custom actions for every information or condition change. For instance in the rewards store, the act of equipping an asset by clicking the asset component card, triggers a change in the colour of the card from white to purple. This also triggers a function that changes any other coloured card back to white, thus ensuring only one card is coloured.

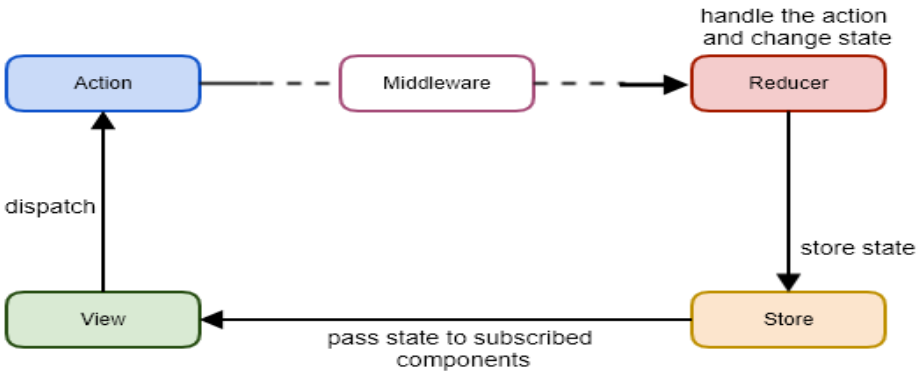
3.1.2 React Redux

React Redux is an open source javascript library used for managing and centralising application state. It is used together with React to build user interfaces. Redux was mainly used for state persistence when creating this app. State persistence means to save the user's data even after a process has been killed, i.e. when the user refreshes or navigates to another page.

The Redux architecture comprises 3 components - store, action and reducer. A store is the place which consists of the entire state of the application. An action is a plain javascript object that contains information about the user's event. It includes information such as type of action, time and location of occurrence, its coordinates and which state it wishes to change to. An action is sent from the view, which are payloads that can be read by the

Reducers. A reducer reads the payloads from the actions and updates the state in the store. It is a function which returns a new state from an initial state.

The Redux architecture follows a unidirectional data flow, as shown in the diagram below.



The overall flow of Redux is as such: an action is dispatched when a user interacts with the application. The root reducer function is then called to divide the task into smaller reducer functions, which then returns a new state to the store. The store notifies the view by calling their callback functions. The view then retrieves the updated state and re-renders again.

3.1.3 Android Studio

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA. Android Studio was used to set up an Android Emulator. The Android Emulator simulates Android devices on our computer so that we can test our application on a variety of devices and Android API levels without needing to have each physical device. Testing our app on the emulator is in some ways faster and easier than doing it on a physical device.

3.2 Backend

3.2.1 AWS Amplify

AWS Amplify is a set of purpose-built tools and services that makes it quick and easy for front-end web and mobile developers to build full-stack applications on AWS, with the flexibility to leverage the breadth of AWS services to further customize applications.

3.2.2 AWS AppSync

AWS AppSync is a fully managed service that makes it easy to develop GraphQL APIs by handling the heavy lifting of securely connecting to data sources like AWS DynamoDB, Lambda, and more.

3.2.3 AWS Cognito

Amazon Cognito lets developers add user sign-up, sign-in, and access control to your web and mobile apps quickly and easily.

3.2.4 AWS DynamoDB

Amazon DynamoDB is a fully managed, serverless, key-value NoSQL database designed to run high-performance applications at any scale.

3.2.5 AWS Lambda

AWS Lambda compute service lets developers run code without providing a server. AWS Lambda only executes code when it is needed and scales automatically. With AWS Lambda, developers can run code for any type of application or backend service.

3.2.6 Firebase

Cloud firestore is used to store user's messages of the chatroom feature. Cloud messaging enables users to chat with each other in real time.

3.3 Git

Git is a DevOps tool used for source code management. It is an open-source version control system used to handle small to very large projects efficiently. Git is also used to track changes in the source code, enabling multiple developers to work together on non-linear development. In the project, we use Github as the Git GUI.

4. Design and Implementation

4.1 Design Consideration / Choice of components

4.1.1 Thematic Choices — Colour scheme and font

Regarding our theme, we decided on outer space as our default background with mainly 4 different planets to represent the four different categories. We decided to take the approach of a monotone background to highlight the thematic colour difference on each category display. The dark grey background is designed to go easy on the user's eyes, but it also corresponds to our theme of space exploration.

One colour was specifically chosen for each category:
Meditation: Yellow-Orange – Since we hope to engage the user in mindfulness, orange was chosen in its association with "happy and uplifting", psychologically encouraging the user to meditate. The analogous yellow and orange is paired with light complementary blue for contrast while maintaining a warm and serene look.
Sleep: Dark Blue – Cool colours tend to calm and relax the user, easing the users to sleep. This is paired with subtle touches of its opposite colour, yellow, to add visual interest.
Move: Red – Red is a stimulating colour, and evokes strong feelings like excitement and energy. This is why red is chosen for the Move category, to induce passion and energise users to exercise. A saturated blue was used to support the red.
Focus: Green – Green promotes security and relieves stress, allowing users to relax and think from a better perspective. This is helpful for users when they need to concentrate. The touch of light yellow brings some warmth and diversity to the cooler green colour.

These four colours form the main colours of our mobile application. The colour palette is consistent throughout the different interfaces and activities. However for the rewards page, we used a light purple as a general theme to detach the rewards feature from the categories.

We chose green for our mascot. It is a colour strongly associated with nature, giving the feeling of growth and life. This fits the theme of the mascot: to accompany and grow along with the user through the journey of mindfulness.

For typography, we have chosen Fredoka One as the main font, a rounded, blockish sans serif font to relate to the gamified experience of the app. It has a clean but fun and engaging look, which is a great representation for the app. Montserrat was chosen as the supporting typeface. It is visually different from Fredoka One, flatter and much thinner in comparison. The two fonts share a rounded quality and thus complement each other.

4.1.2 Illustrations and icons

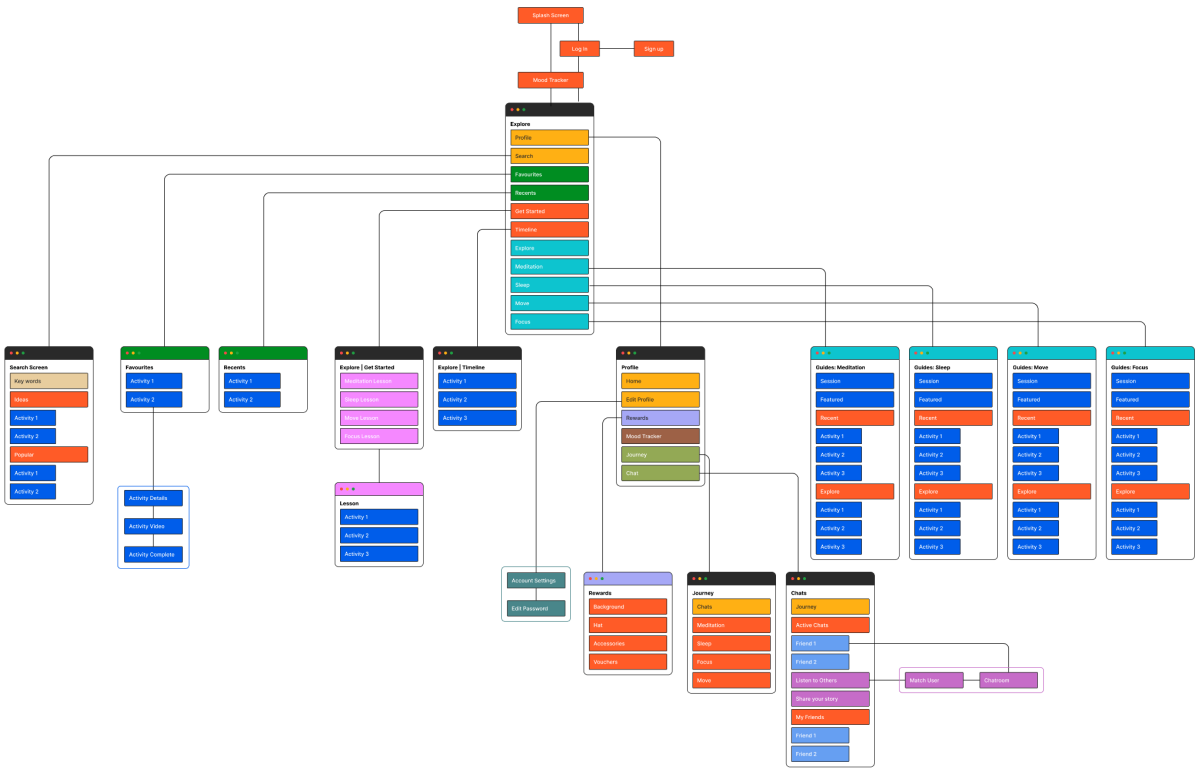
The app takes on a rounded and softer look for both UI and illustration. The UI components are long, rounded rectangles, with the icons being thickly-lined with rounded edges. The rounded quality is perpetuated throughout the app.

For our mascot design, we chose to stick to a cute design that is visually appealing to our audience. We decided on a non-humanoid mascot as it will be easier to work with. The simplicity of our mascot allows for an easier design and change of assets which users can then exchange for coins and customise their avatar.

The illustrations in the app adopts an organic and crayon-textured depiction of outer space to emphasise the gamification concept. Similar to colours, the 4 planets representing different categories are designed differently, which can be seen on their respective guides and journey page. The illustration of each activity depicts a scene or location on the “planet”, where the user can “explore”. The journey page is designed to look like an adventure map where the user can track his/her progress on the app.

4.2 Final Design

4.2.1 Sitemap

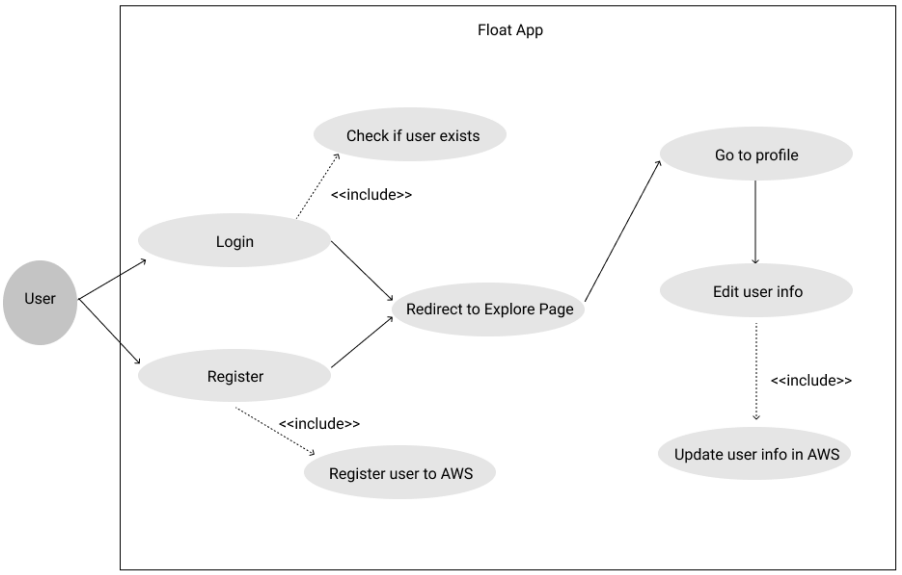


4.2.2 Features

Here are the features of Float along with the use case diagram. The final UI design can be found in Appendix D.

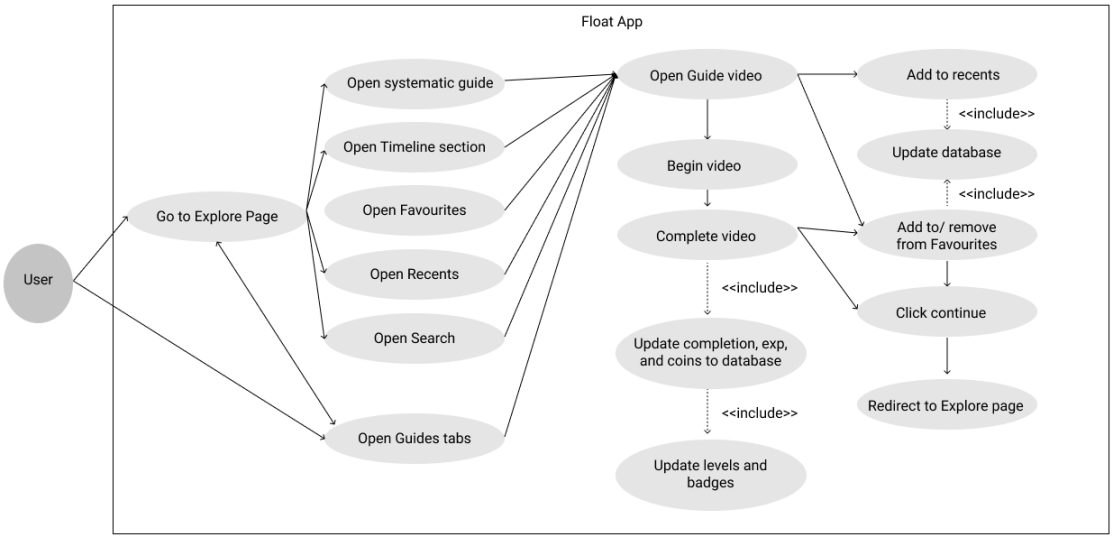
1. **Account**

This forms the basis of the user's account.



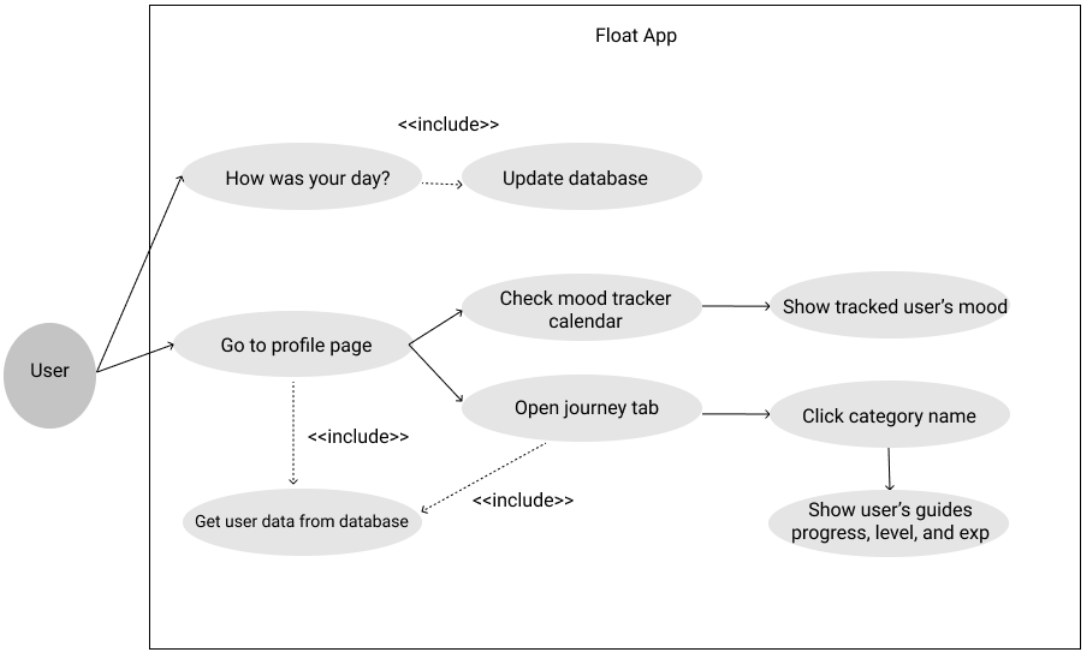
2. **Guides**

The guides will lead the user to complete an activity for mindfulness, which then updates the database upon completion.



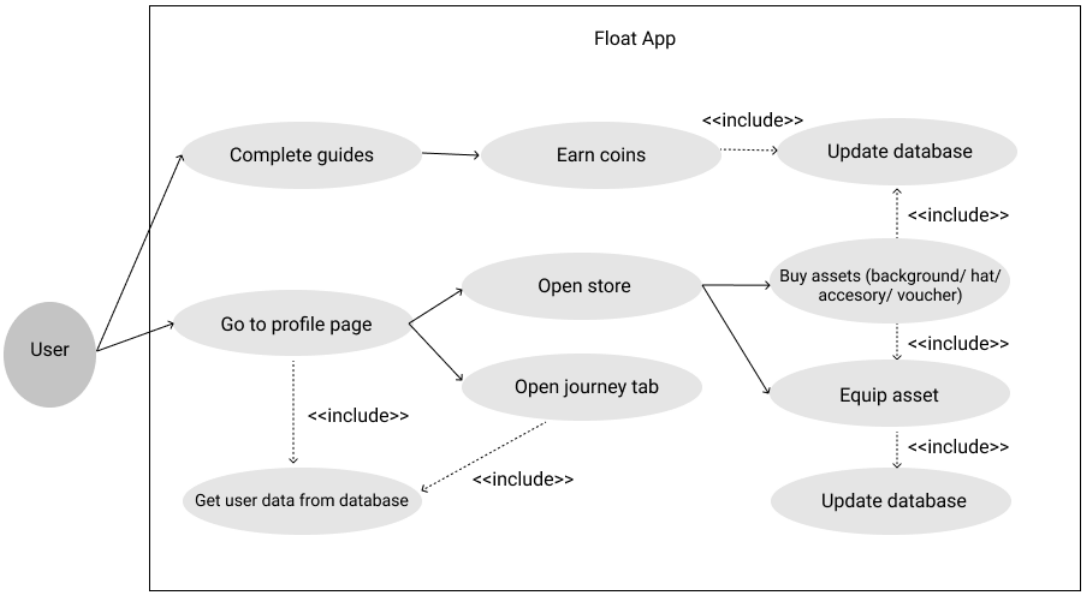
3. **Journey and Tracking**

The user can gain more insights into their wellbeing over time with the statistics displayed in the journey section and mood tracking calendar.



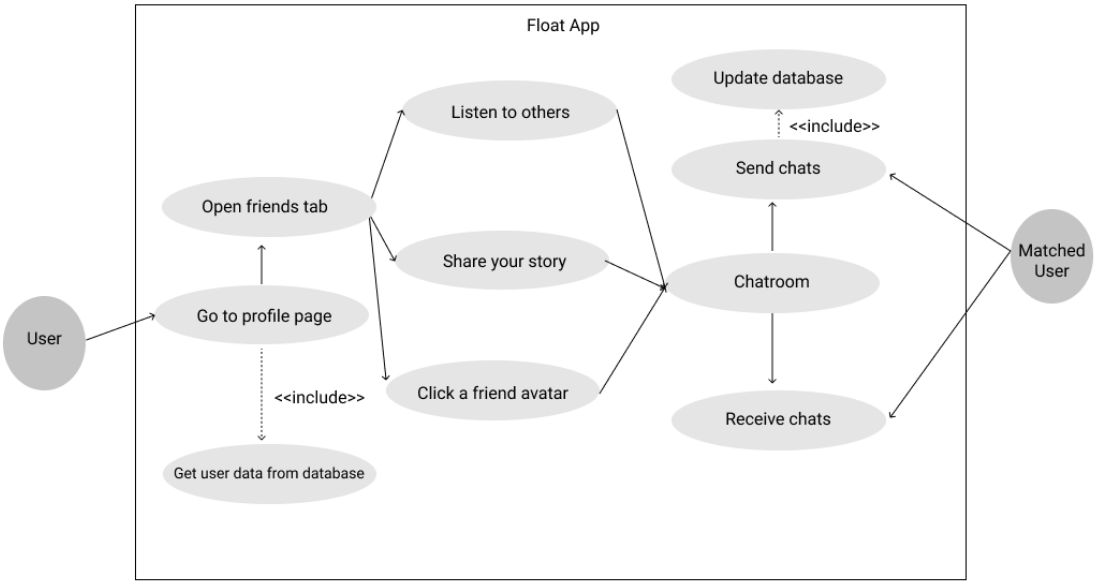
4. Rewards

The users can trade coins for in-app or voucher rewards as a motivation for completing activities.



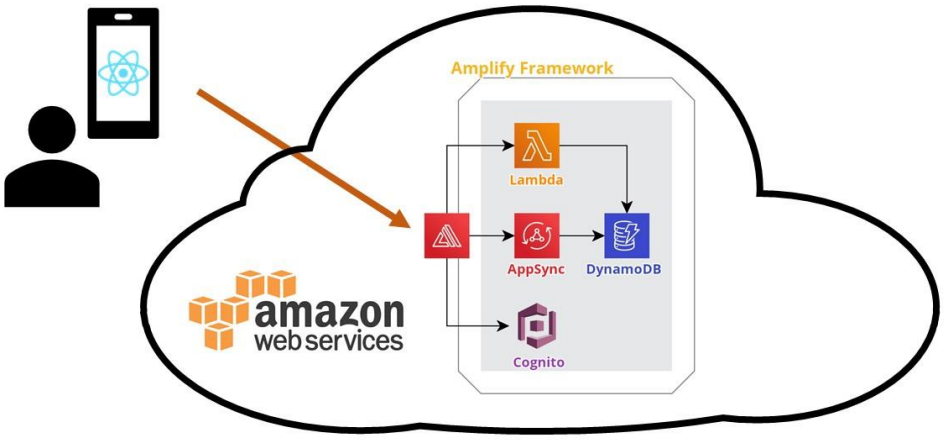
5. Chat room

This feature allows users to chat anonymously with others and add friends if they wish to. This is to allow users to have an outlet for expression, and gain friends along the way.



4.3 Implementation

4.3.1 System Architecture



The user will interact with our app which is based in React Native and Android. Our project uses AWS Amplify as a framework to implement other AWS services like AppSync, Cognito, DynamoDB and Lambda to be our mobile app's backend. This is a typical serverless architecture where developers build and run services without having to manage the underlying infrastructure. For many developers, serverless architectures offer scalability, more flexibility, and quicker time to release, all at a reduced cost.

4.3.2 Frontend

Float is built with React Native on top of the Android infrastructure. There are some libraries that we utilise in the process. This section will explain the different libraries in use.

The project is broken down into several components with React Native's components and props concepts. This makes creating repeated and reusable components easier. While for routing and render handling, we use React Navigation library to help us create our navigation and bottom bar. The navigation can be accessed directly in the App.js file.

In terms of storing data and state management, especially in our key features such as guides, rewards, and user details, we implemented React Redux framework to have persistent data as the user navigates across the app. Each feature is stored in different reducers. To communicate with reducer, action is required which can be used to access or update the respective data as the user progresses in the app. This lets the information shown across the app consistent.

One notable part of Float is the video player. Iframe facilitates direct access to Youtube videos. By using it, we are able to detect whether the user has completed the entire video. After a video guide is finished, the completed page will be triggered.

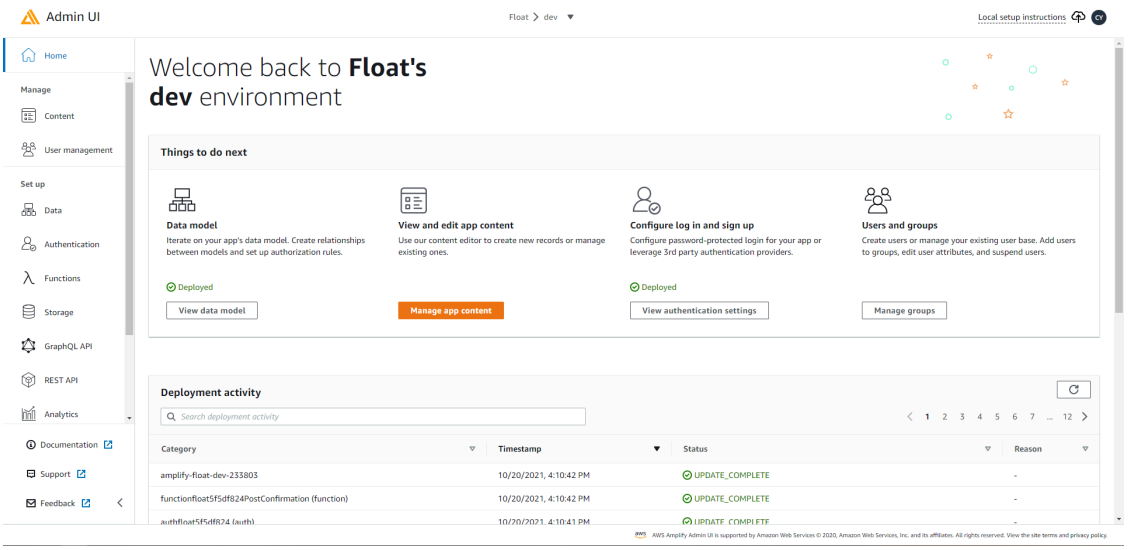
In the profile page, the Mood Tracker section occupies a large portion. The calendar used here is a combination between the React Native Calendar Picker library system and our styling with JSX styles. The conditional styling for each date is given according to the user's input in the 'How are you feeling?' section (refer to Appendix D). For each user input, the app will call a backend function to update the database. This database is then the reference point for the styling.

Another interesting component of our app is the pop-up which can be seen in the Journey section of the profile page, and when you purchase any assets in the Reward Shop. We use the React Native elements, Modal, in order to achieve this. With proper styling, the modal can be used for different purposes. Moreover, with the help of props passing, we can easily customise the content of the pop-up based on the user's input.

4.3.3 Backend

Admin UI

The Amplify Admin UI service provides a programmatic interface for configuring and managing app backends. It is a visual interface to develop app backends and manage app content outside the AWS Management Console.

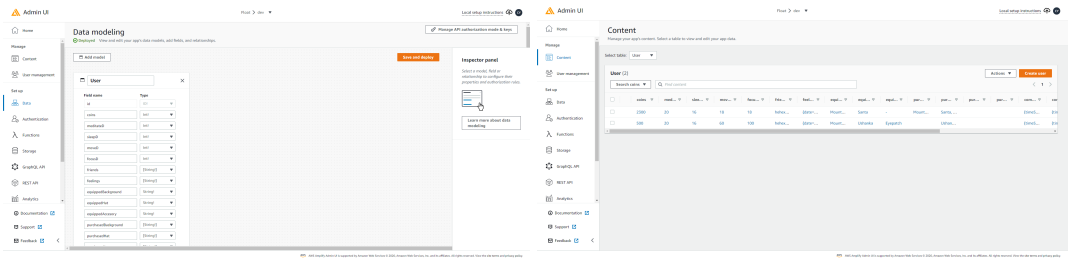


Amplify Admin UI Home

Using this visual interface of the backend, developers can easily set up the app's data model, authentication and authorisation rules, and manage users and content in a simple manner, and all in one place.

Visual data modelling allows developers to build the database focusing on the requirements of the app, rather than setting up database tables and API. AWS Amplify Admin UI handles the generation of schema and database tables, so developers don't have to worry about it.

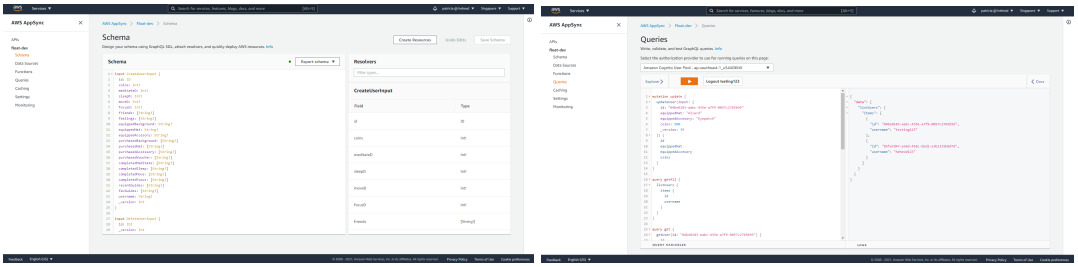
If there are updates in the backend data model, developers can simply update it using this visual interface. The schema can then be auto-generated using Amplify's CLI automated code generation feature. Database tables are automatically created in DynamoDB, a database service by AWS which we utilise. Any additional requirements in configuring the backend are handled by AWS.



Amplify Admin UI Visual Data Modelling, Amplify Admin UI Content Management

The Admin UI also makes it easy for developers to manage content of the app. The contents can be edited directly in this UI, rather than having to use functions to update the backend in the app. It also provides a clear view of the data content which is easily accessible, in comparison to querying the backend using the app and parsing it to check the contents.

AWS AppSync



AWS AppSync Schema, AWS AppSync Queries

AWS AppSync Queries makes it easy to run and test queries to the database in GraphQL before converting it to React Native code that can then be used by the frontend in the app to read or write data to the backend database.

To enable frontend to access the backend and query the database, we created an API requirements document to collaborate and share code snippets with frontend.

4.4 Development Phase

During the duration of the project, the team is split into 3 different sub groups with Clio, Violin and Jun Qian as sub-team leaders for UI/UX design, Frontend and backend respectively. In the first few weeks, the team is more focused on wireframing and designing UI to get started with the coding. Wireframing was finished on week 5 with 80% of the wireframe version being coded. The project has a working prototype on AWS Amplify with authentication and database on week 6. Afterwards, some design members shift to frontend and backend to support the development.

During the development phase of the project, our group implemented a loose communication between the sub-groups with weekly meetings. Every meeting was about weekly deliverables and updating of our progress. Implementing such a system allowed everyone to contribute to the project effectively.

To combine the individual progress, we use Figma for design and Github for code. With different branching names, version control, and features like pull requests from Github, we can review and merge the codes easily.

4.5 Discussion

This project requires a lot of effort in the process, from planning to development. This involves the collaboration of 3 aspects: design, frontend and backend. To achieve a fully working application, we need to spend an extensive duration of around 6-10 hours a week.

There are some challenges faced throughout the development such as information discrepancy, advanced custom design features like the sleep journey calendar, and code integrations. By continuously searching for resources, we managed to get to where we are now.

The choice of software that we use – VS Code, Figma, AWS, etc – allow us to expedite the development process. Not to mention, having Github makes the collaboration between developers easier, especially with the branching and version control.

5. Conclusion and Recommendation

5.1 Conclusion

The Covid-19 pandemic has exacerbated mental health issues faced by students. During the Float project, the mental wellness market has shown to the team many of their shortcomings. As of currently writing, AWS Amplify with react native has shown good reliability, flexibility and accessibility in developing a mobile application. With the accelerated advance of mobile application development tools. AWS Amplify has shown to be a capable tool in building mobile applications.

5.2 Recommendation

5.2.1 NoSQL Design for DynamoDB

Relational database systems (RDBMS) and NoSQL databases have different strengths and weaknesses. During the Float project, the AWS DynamoDB NoSQL database by many metrics is not implemented optimally. Which calls to the reason why DynamoDB was used as our project's backend? The predominant reason is the cost of development. To put into perspective, the cost of development on building on a serverless architecture for project Float was USD$2.11 for the month of october. Disregarding the cost, AWS Amplify provides a set of purpose-built tools to make development easier. AWS DynamoDB is also well suited for mobile applications because it is designed for high-scale use cases where consistent performance is critical as an application grows.

For future development, following AWS white paper on Modern Serverless Mobile/Web Application Architecture and best practises on designing schema for AWS DynamoDB. It would be best to attempt at designing a schema where the data can be scaled horizontally instead of vertically. In the AWS DynamoDB case, it would be as few tables as possible with as many rows of data as possible.

5.2.2 Functionalities

There are some functionalities that can be improved or explored further in the future:

- Explore Recommendation - algorithm to suggest users on new guides based on their activity.
- Systematic Guides - more lessons curated progressively for the user.
- Chat Room - better algorithm to match one person to another.
- Rewards - more assets and vouchers varieties.

5.2.3 Activities

As we are not experts in the field of mental wellness, we were unable to create better activities for our app. To fix this, we need to seek medical professionals to create better activities for our users.

Appendix A – User Guide

User Guide

- ➢ Create an account in our app
- ➢ After logging in, user will be asked to check-in via the mood tracker system
- ➢ Once user check-in, they will go through a journey map in their own float(tube)
- ➢ User can dock at different planets (categories) to explore/engage in the series of activities available at each planet (Move/Sleep/Focus/Meditation)
- ➢ Availability of customizable avatar settings through in-app purchase with coins
- ➢ Real-life vouchers accessible to users when they earned a certain amount of experience points and coins
- ➢ Chat room available for user to listen to story or share their story anonymously

Appendix B – Maintenance Guide

Please follow this guide

Required Knowledge

1. Backend – AWS cloud services
    a. AWS Amplify
    b. AWS AppSync
    c. AWS Cognito
    d. AWS DynamoDB
    e. AWS Lambda
2. Frontend – React Native
    a. React concepts
    b. React Navigation
    c. React Redux
3. Git

Application Needed

1. Visual Studio Code
2. Github

About the project

We are using:

- ➢ No*de ver 16.13.0 LTS*
- ➢ *jdk ver 14.0.1*
- ➢ *aws-amplify ver 6.3.1*

Setup

To setup the project on your computer, you should

- ➢ Set up React Native and Android Studio
- ➢ Create Github account and clone this project to your desired local directory
- ➢ Run *npm install* in the 'Float' folder to download the modules
- ➢ Run *npx react-native run-android* to run your app

To setup AWS amplify for your project

- ➢ AWS Amplify Documentation
- ➢ Run amplify init in the project directory
- ➢ Do you want to use an existing environment? Yes
- ➢ Default for all others
- ➢ Run npm install aws-amplify @aws-amplify/ui-react
- ➢ Run amplify push in the project directory

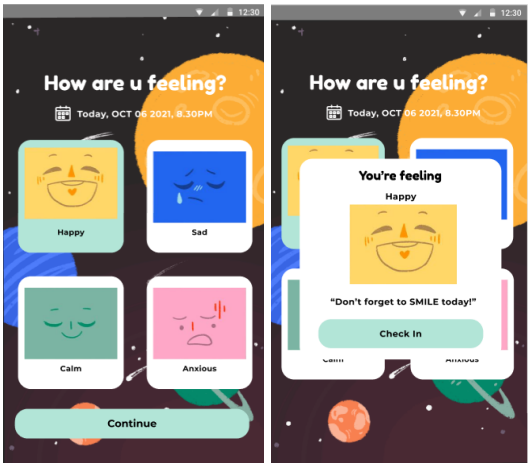
Appendix C – Source Code

All codes can be accessible via https://github.com/ljunqian/Float

Appendix D – Final Design Guide

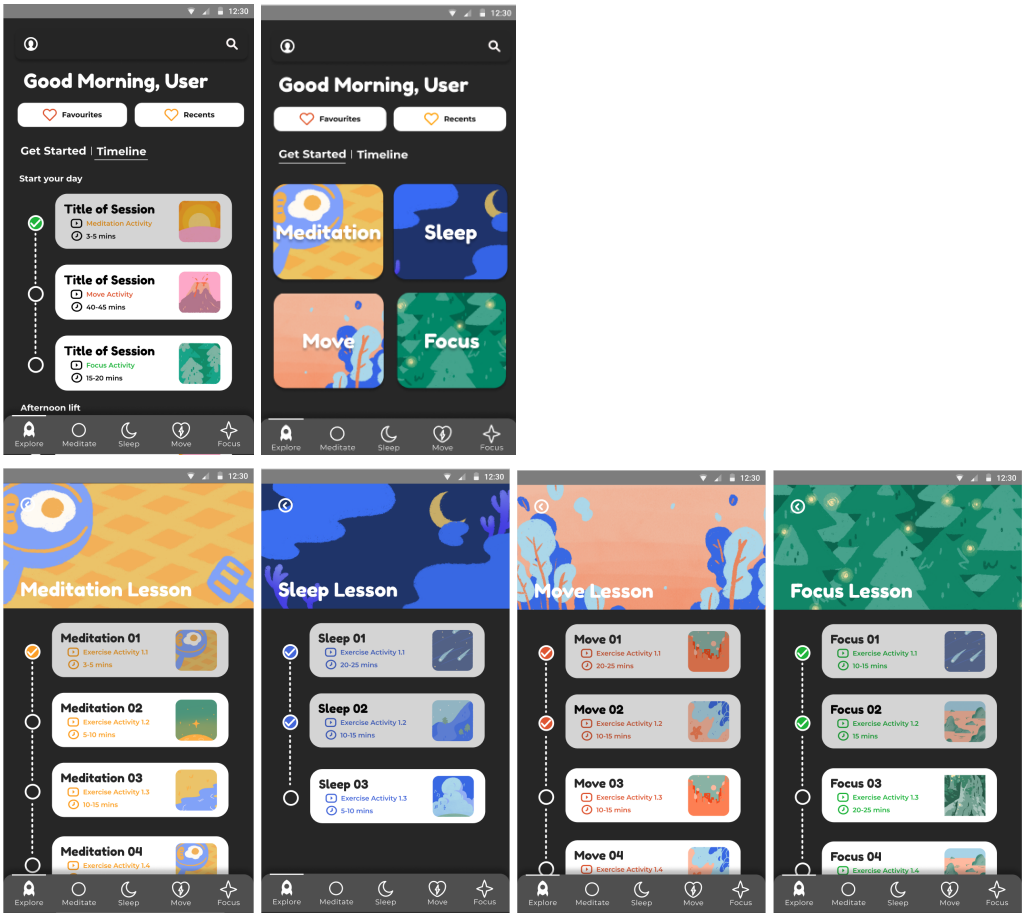
1. How's your day

Mood tracker system that stores selected mood via the check-in button to the calendar view displayed at Profile page.

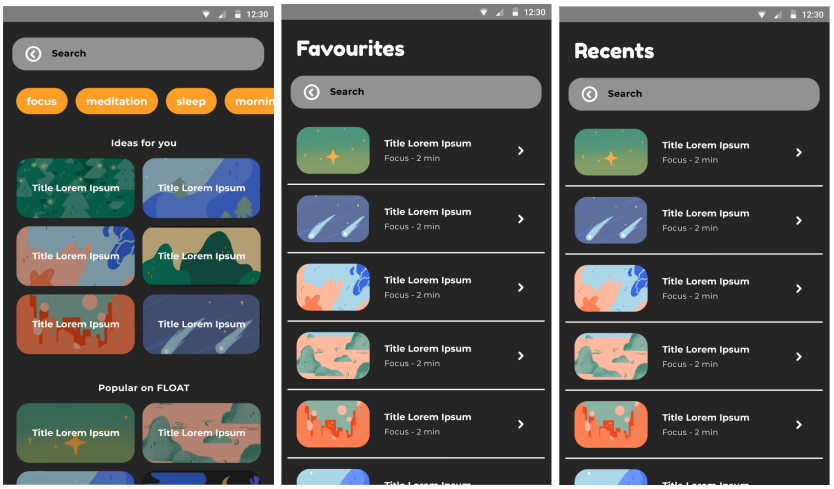


2. Explore

In the explore page, the default page would be the "Timeline" tab which lists out the suggested activities for each day for the user to follow. On the other hand, if a user clicks on the "Get Started" tab, he/she will be able to select one of the 4 categories and complete the activities according to the systemic approach.

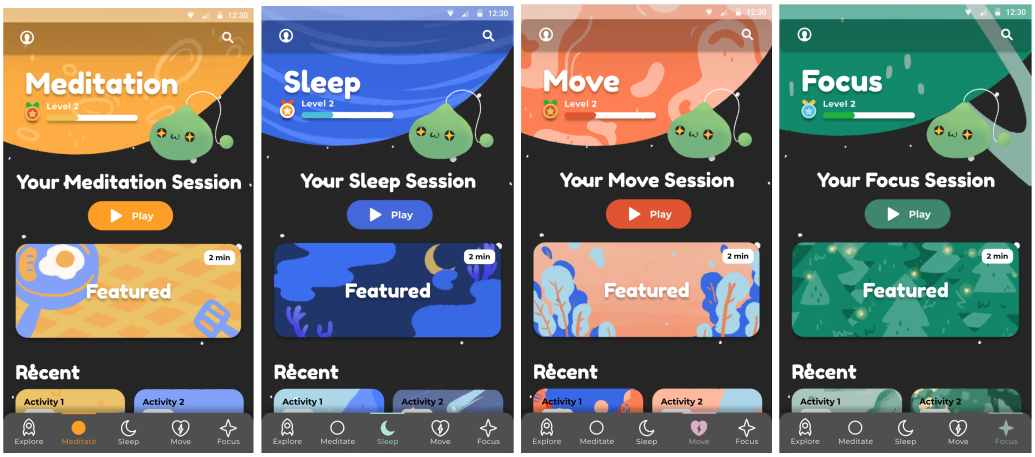


At the top right hand corner of the page there is also the Search button, where users can search for the activity that they want to do or relevant activities. It comes with suggested keywords and activities that predict user's searches. There is also the Favourites and Recents tab where users can find the activities they have favourited and did recently respectively.



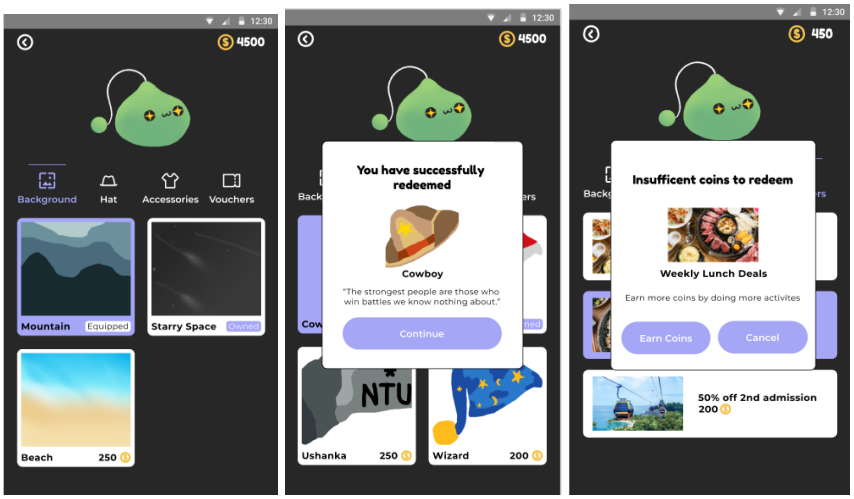
3. Guides

Each category displays a main session, featured activity, recent activities, and explore (more activities) so that users have a wide selection of activities they can engage in should they wish to work more on a particular category. Users will earn coins upon completing an activity, which they can use to redeem for rewards.



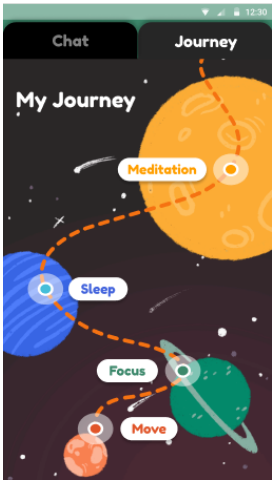
4. Rewards

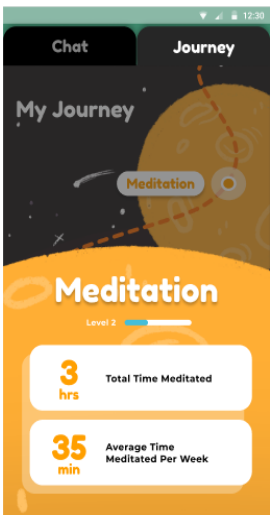
Redemption of coins for customizable avatar assets or real-life vouchers through the rewards page.

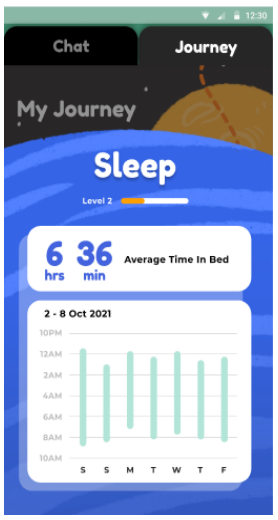


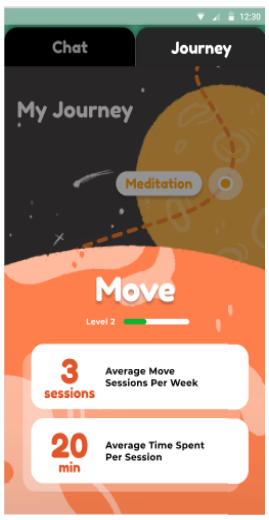
5. My Journey

In the My Journey page, it provides an overall statistical information displaying the total or average time spent every week or session for each category.







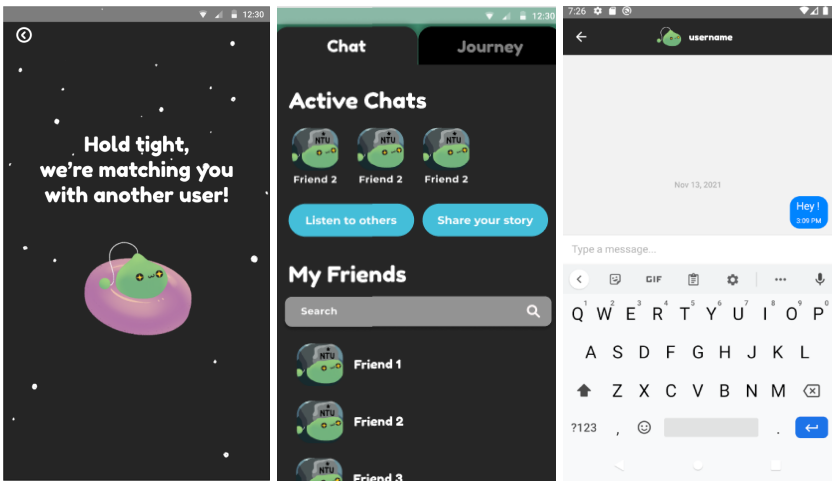


6. Chat Room

A feature which allows users to chat anonymously with each other. The purpose of this is to encourage people with problems, to be able to share their stories and problems to other people, as a tool to be able to express, and not just struggle alone.

The chatroom will connect two individuals (one willing to share a story, another one willing to listen and give feedback/response) based on Queue architecture (First In First Out).
Users can pick which category they want by clicking on any of the two buttons in the Chat section of the profile page, either “Listen to others” or “Share your story”.

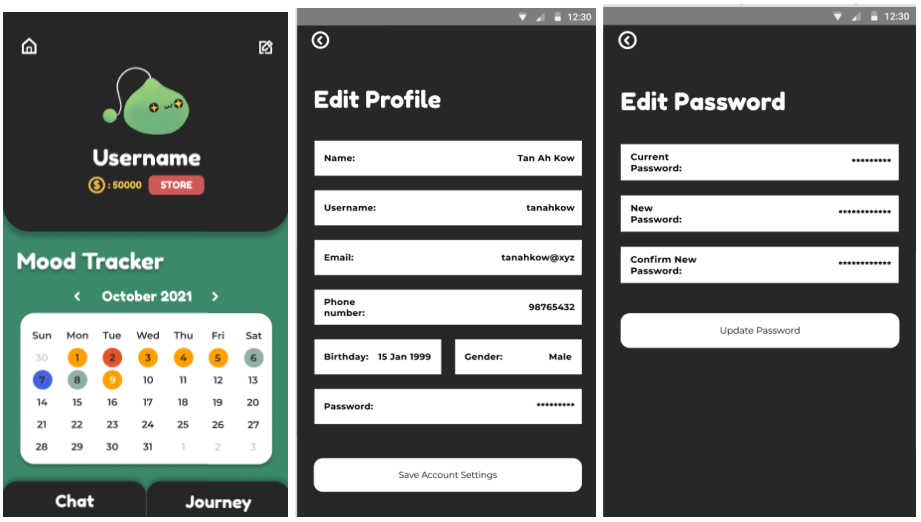
If both users agree, they would be able to “add friend” with each other.



7. Profile

The profile page contains a lot of sections which link to other features/pages.
The icon on the top right, would lead users to the "edit profile" and "edit password" section to allow users to update their personal information/credentials.
Mood tracker section which allows users to be able to view their daily mood throughout the entire month.

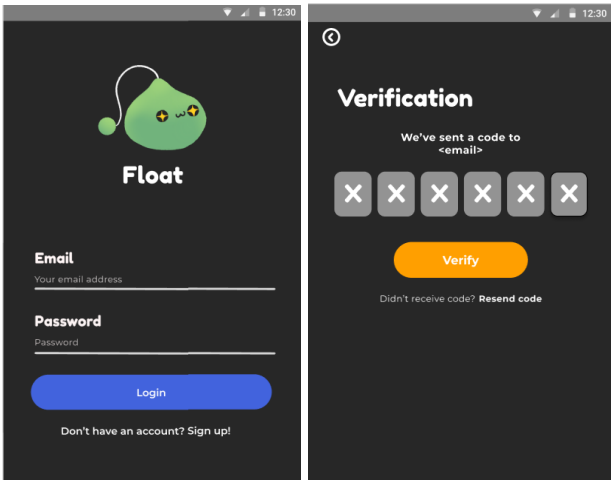
Users can switch tabs between "Chat" and "Journey" in chat, users can navigate to find someone to chat with using the chat room feature. Users may also view the entire list of their friends which has already been added previously.



8. Log-In

A simple log-in page to register or enter credentials before seeking for account verification that connects with the backend for authentication.
The verification feature is also used when creating the account for the first time, to verify the credibility of the email that is being inputted.

Final UI:


12:30
Float
Email
Your email address
Password
Password
Login
Don't have an account? Sign up!
12:30
Verification
We've sent a code to
<email>
Verify
Didn't receive code? Resend code