📖 README

Table of contents

Project Background

This app is a networking mobile app. It was created to connect mentors and mentees in a fun and casual way! The app simply focuses on providing the first point of contact - the mentoring sessions should be one-on-one and in-person. The app also aims to be a welcoming place for shy and first-generation students to expand their professional network.

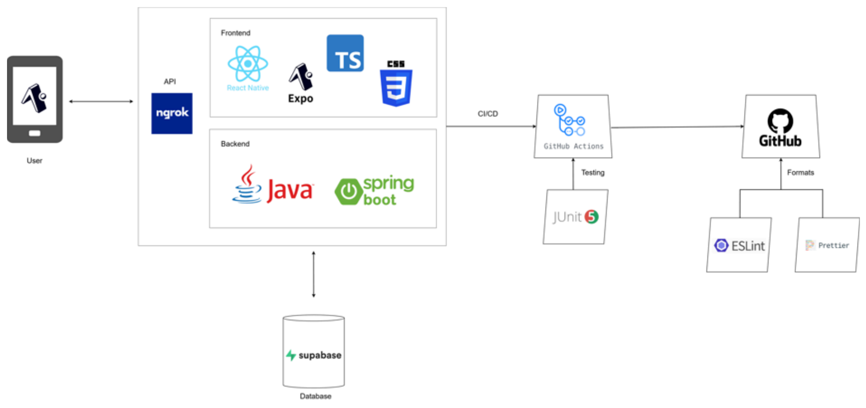
Documentation

Note: The diagrams have been simplified for demonstration purposes in this README. You can find the full diagrams here: PrandaArchitectureModels

Features (User Stories)

See here: Features (User Stories)

System



User

- Will access the networking app via the Expo Go app

Frontend

- UI is built with the React Native framework via Expo Go, to ensure the app can be used on both iOS and Android devices. React Native uses Typescript and CSS as the coding languages

Backend

- Using the Springboot framework as it systematically and robustly handles database queries from the frontend. Springboot uses Java as its coding language

API

- ngrok hosts a gateway for the frontend to call the backend when in need of any database functions

Database

- Using Supabase to store user data

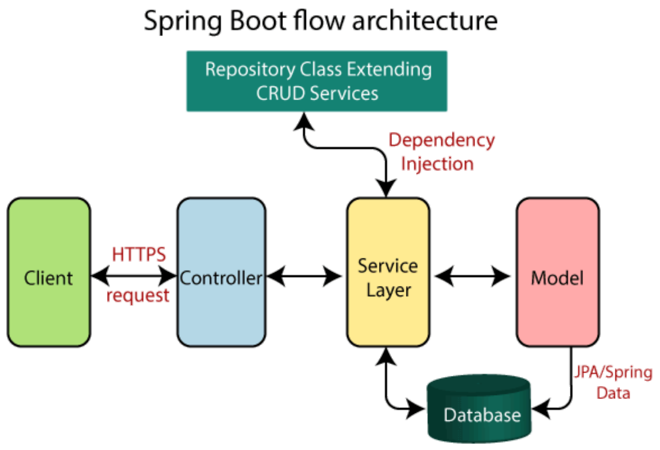
CI/CD

- Using GitHub Actions to automatically run tests whenever new changes in the code are made. Tests are written in Java's JUnit framework

Formatting

- Using ESLint and Prettier to format the frontend code. The backend code is automatically formatted with IntelliJ (a code editor for Java)

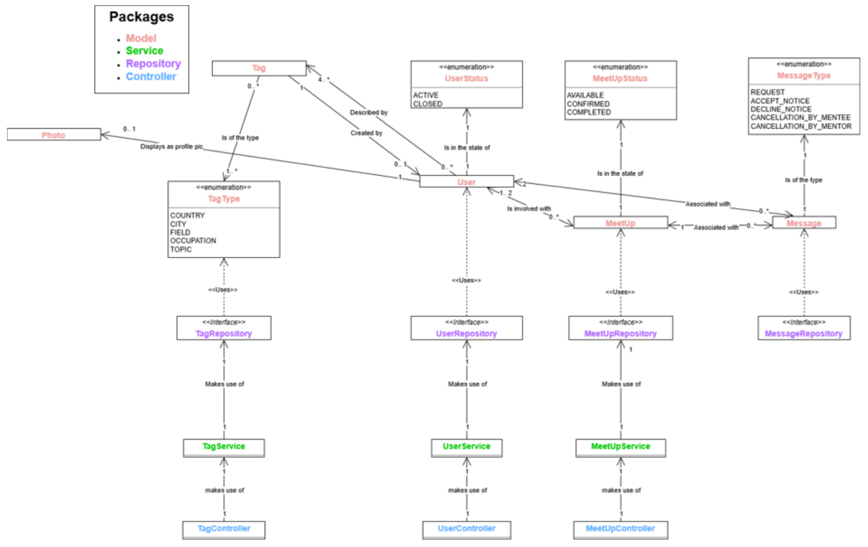
Architecture



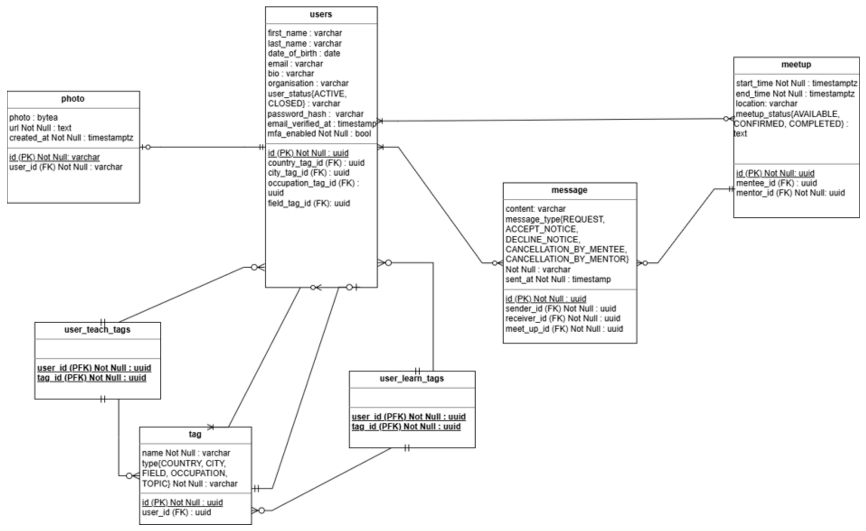
Since the project uses the Springboot framework for the backend, it follows the Spring Boot architecture, which has the following components:

- **Database:** where the user data is stored
- **Model:** represents the database tables in code
- **Repository**: allows the Service classes to access the Model classes (and by extension, the database)
- **Service:** implements the key functionality of the app. Queries the database via the Repository class when needed
- **Controller:** accessed by the Client to access the backend functionality. Calls methods in the Service class to perform the tasks required by the Client. Provides a safe interface so the frontend doesn't interact directly with the backend logic (prone to errors if modified)
- **Client:** the app's frontend

Each of the key database tables (users, meetup, message, tag) have a Model, Repository, Service, and Controller class in the backend:



Database


users
first_name : varchar
last_name : varchar
date_of_birth : date
email : varchar
bio : varchar
organisation : varchar
user_status{ACTIVE, CLOSED} : varchar
password_hash : varchar
email_verified_at : timestamp
mfa_enabled Not Null : bool
id (PK) Not Null : uuid
country_tag_id (FK) : uuid
city_tag_id (FK) : uuid
occupation_tag_id (FK) : uuid
field_tag_id (FK): uuid
photo
photo : bytea
url Not Null : text
created_at Not Null : timestamptz
id (PK) Not Null: varchar
user_id (FK) Not Null : varchar
meetup
start_time Not Null : timestamptz
end_time Not Null : timestamptz
location: varchar
meetup_status{AVAILABLE, CONFIRMED, COMPLETED} : text
id (PK) Not Null: uuid
mentee_id (FK) : uuid
mentor_id (FK) Not Null: uuid
message
content: varchar
message_type{REQUEST, ACCEPT_NOTICE, DECLINE_NOTICE, CANCELLATION_BY_MENTEE, CANCELLATION_BY_MENTOR} Not Null : varchar
sent_at Not Null : timestamp
id (PK) Not Null : uuid
sender_id (FK) Not Null : uuid
receiver_id (FK) Not Null : uuid
meet_up_id (FK) Not Null : uuid
user_teach_tags
user_id (PFK) Not Null : uuid
tag_id (PFK) Not Null : uuid
user_learn_tags
user_id (PFK) Not Null : uuid
tag_id (PFK) Not Null : uuid
tag
name Not Null : varchar
type{COUNTRY, CITY, FIELD, OCCUPATION, TOPIC} Not Null : varchar
id (PK) Not Null : uuid
user_id (FK) : uuid

users

- Represent the users of the app (mentees and mentors)
- Associations:
  - One of each of these tags: COUNTRY, CITY, FIELD, OCCUPATION
  - Many teach tags
  - Many learn tags
  - Many meetups
  - Many messages
  - One photo

*Note: Supabase has its own users for authentication which is used for sign up, sign in, sign out, etc. The app will create an associated entry in the **users** table for each authentication user, so that the custom fields (e.g., tags) can be stored.

tag

- Represent the tags used to describe users. Also used as filters when searching profiles
- the user_id represents the id of the user who created it (if any)

user_teach_tags

- Represents a user being associated with a teach tag. Needs to be a separate table as a user can have many teach tags and a teach tag can describe many users
- Associations:
  - One user
  - One teach tag

user_learn_tags

- Represents a user being associated with a learn tag. Needs to be a separate table as a user can have many learn tags and a learn tag can describe many users
- Associations:
  - One user
  - One learn tag

meetup

- Represent the mentoring sessions between mentees and mentors
- Associations:
  - One mentor
  - One mentee (if confirmed as booked)
  - Many messages

message

- Represents the messages that a user can send when performing the following tasks:
  - REQUEST: when a mentee requests to book a mentor's meetup
  - CONFIRMATION: when a mentor accepts a mentee's request
  - CANCELLATION: when a mentee cancels a meetup that was confirmed as booked
  - DELETION: when a mentor deletes a meetup that was confirmed as booked
- Associations:
  - One sender
  - One receiver
  - One meetup

What's currently in the database

We've got some test users, meetups, messages, and tags which we can delete. The only data that is recommended to be kept in the database are the default tags (user_id is null), as these will be used by users when signing up, editing their profile, and searching for mentees.

Key Algorithms

User

- Browse profiles: gets a list of recommended mentor profiles according to how well the mentor's teach tags match the user's learn tags, and filters (CITY, FIELD, OCCUPATION, TOPIC)
  - The recommendations automatically filter by mentors in the same city as the user, but this can also be changed by the user
  - Only mentors who have future available meetups will show up
- Search profiles: gets a list of users according to how well the mentor's profile matches and search and filters (CITY, FIELD, OCCUPATION, TOPIC)

  - Automatically filter by mentors in the same city as the user, but this can also be changed by the user
  - Only mentors who have future available meetups show up

Tag

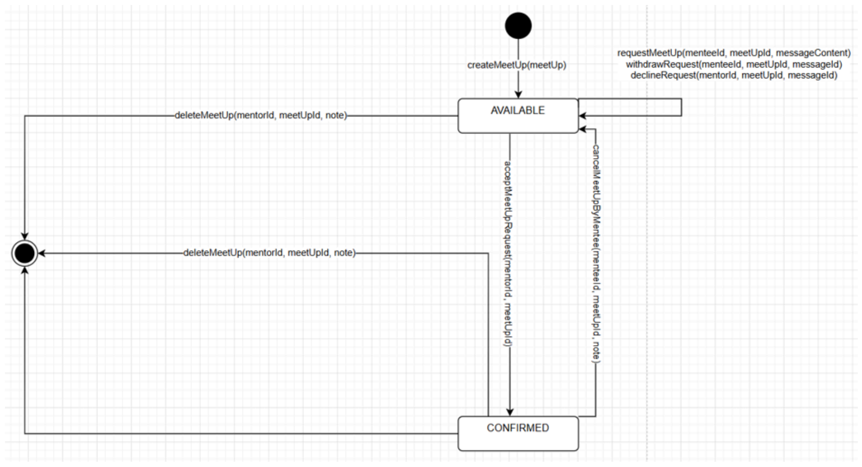
We create tags in database that all users can choose from (default options). For default tags, the user_id will be null as it was created by the database admins in Supabase.

- Get tags: retrieves tags from the database based on an optional tag type and optional search
  - If no tag type or search: returns all default tags
  - If only tag type: returns all default tags of the tag type
  - If tag type and search: returns all default tags matching the tag type and the search
- Create tags: creates a tag when a user wants to select a tag that is not already in the database
  - The ‘user_id’ field in the tag table specifies the user who created the tag
  - Users can only create tags of the type: FIELD, OCCUPATION, and TOPIC. They cannot create tags of the types: COUNTRY and CITY
  - Users can only see the tags they’ve created and no one else’s. This is to avoid users selecting a tag owned by another user that the tag owner may delete. A possible solution to this was to prevent users from deleting tags, but this could potentially clog up the database with misspelt tags and such
- Delete tags: deletes a tag when a user deselects a tag they’ve created as no one else will be using the tag

Meetup

- Create meetup: only mentors (users with teach tags) will be able to create a meetup
  - A pop up will show in the frontend when a user with no teach tags tries to make one
- Get incoming meetup requests: will show incoming requests for a mentor’s AVAILABLE meetups (not CONFIRMED or COMPLETED)
- Get outgoing meetup requests: will show outgoing requests from a mentee where the meetup is AVAILABLE (not CONFIRMED or COMPLETED)
- Cancel meetup by mentee: the meetup with go back to AVAILABLE status. Requests that were made before the meetup was CONFIRMED will once again be visible for the mentor and mentees

Here is the lifecycle for a meetup:


createMeetUp(meetUp)
requestMeetUp(menteeId, meetUpId, messageContent)
withdrawRequest(menteeId, meetUpId, messageId)
declineRequest(mentorId, meetUpId, messageId)
AVAILABLE
deleteMeetUp(mentorId, meetUpId, note)
acceptMeetUpRequest(mentorId, meetUpId)
cancelMeetUpByMentee(menteeId, meetUpId, note)
deleteMeetUp(mentorId, meetUpId, note)
CONFIRMED

Tests

See here: 🙌 Test Reports

Set-Up Guide

1. Required accounts

1. Create a GitHub account - to access the code repository
2. Create a Supabase account - to access the database

Send the emails associated with your GitHub and Supabase accounts to the current admins and they can give you access

2. Install dependencies

1. Install Node.js. You can check if it's already installed by heading to your computer's Terminal application and running:

```
$ node --version
```

   If it outputs a version number, Node.js is already installed. If it comes up with an error, it is not yet installed.

2. Install the Expo Go app on your device of choice. You can install the app on Android and iOS mobile devices, Android Studio Emulator, and iOS Simulator (only on Mac computers)
   a. If you have an Android device, we recommend using Android Studio Emulator instead as there have been issues with connection in the past

3. ngrok - for the API calls
   a. Install ngrok:
      i. For Windows devices, download ngrok via the Microsoft Store
      ii. For Mac devices, run this in your Terminal:

```
brew install ngrok
```

iii. For Linux devices, run this in your Terminal:

```
sudo snap install ngrok
```

b. Create an ngrok account

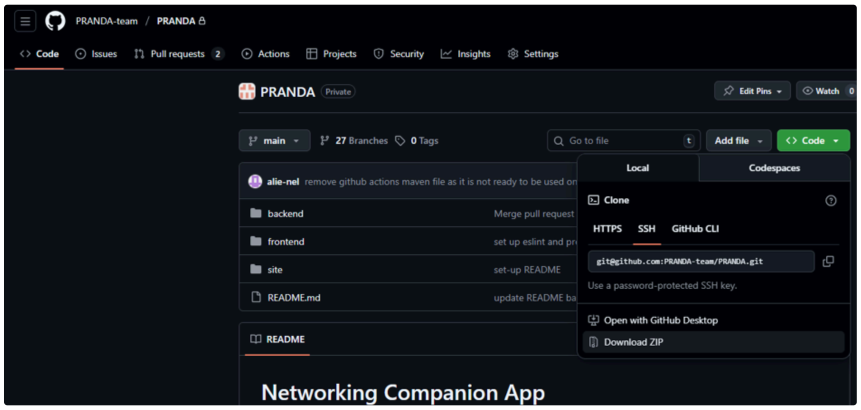
c. Copy an AuthToken from the dashboard page

d. In your Terminal, run:

```
ngrok config add-authtoken <Token you got from dashboard>
```

3. Clone the repository

1. Click on the '<> Code' button and press 'Download ZIP'



alie-nel remove github actions maven file as it is not ready to be used on

backend Merge pull request

frontend set up eslint and pr

site set-up README

README.md update README ba

git@github.com:PRANDA-team/PRANDA.git

Use a password-protected SSH key.

Networking Companion App

2. Extract the ZIP file
3. env files - these store private information needed for the app to run. It protects the data as env files will not be uploaded to the GitHub repository

   You should have been sent a folder called `env-files` containing some env files

   a. Move the .env file from the `env-files/backend` folder to the `PRANDA-main/PRANDA-main/backend` folder (or whichever folder the code repository is stored in)

   b. Move the .env file from the `env-files/frontend` folder to the `PRANDA-main/PRANDA-main/frontend` folder (or whichever folder the code repository is stored in)

4. Start the backend

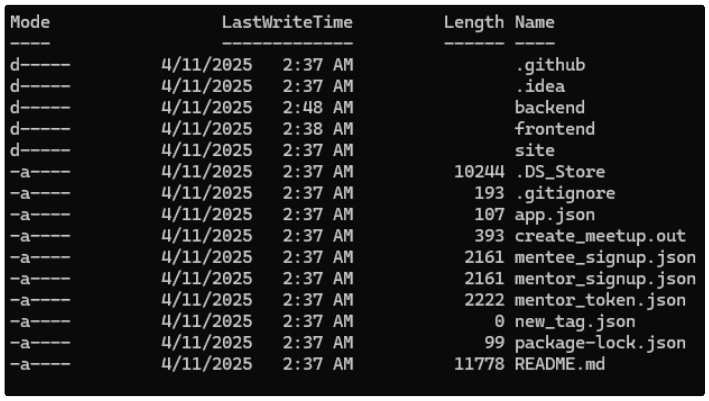
1. Open your computer's Terminal application
2. Change directory to the folder where the code is stored. For example:

```
cd ~/Downloads/PRANDA-main/PRANDA-main
```

   When you run:

```
ls
```

You should see files like this:



```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         4/11/2025   2:37 AM                .github
d-----         4/11/2025   2:37 AM                .idea
d-----         4/11/2025   2:48 AM                backend
d-----         4/11/2025   2:38 AM                frontend
d-----         4/11/2025   2:37 AM                site
-a----         4/11/2025   2:37 AM          10244 .DS_Store
-a----         4/11/2025   2:37 AM            193 .gitignore
-a----         4/11/2025   2:37 AM            107 app.json
-a----         4/11/2025   2:37 AM            393 create_meetup.out
-a----         4/11/2025   2:37 AM           2161 mentee_signup.json
-a----         4/11/2025   2:37 AM           2161 mentor_signup.json
-a----         4/11/2025   2:37 AM           2222 mentor_token.json
-a----         4/11/2025   2:37 AM              0 new_tag.json
-a----         4/11/2025   2:37 AM             99 package-lock.json
-a----         4/11/2025   2:37 AM          11778 README.md
```

4. Change directory to the `backend` folder:

```
cd backend
```

5. Set the env variables
   a. For Mac, run:

```
set -a; . ./.env; set +a
```

   b. For Windows, for each env variable, run:

```
$env:<VARIABLE>="<VALUE>"
```

For example:

```
$env:SUPABASE_DB_NAME="name"
```

You can also run them all at once by separating the variables with `;`. For example:

```
$env:SUPABASE_DB_NAME="name"; $env:SUPABASE_DB_USER="user"
```

6. Run the backend with:

```
mvn -DskipTests spring-boot:run
```

Note: this will only work if your device and your internet provider supports IPv6. You can check if your internet provider does so here. If you cannot connect via IPv6, you may consider purchasing Supabase Pro as it supports IPv4.

5. Start the frontend

1. Open a new Terminal window
2. Run:

```
ngrok http 8080
```

3. The above command should give you a link which localhost is being forwarded to. It should look something like:

```
Forwarding      <link> -> http://localhost:8080
```

   Copy that link and edit the `PRANDA-main/PRANDA-main/frontend/.env` file so that the `EXPO_PUBLIC_API_URL` variable is set to this link

4. Open a new Terminal window and change directory to the `frontend` folder:

```
cd frontend
```

5. Install dependencies

```
npm install
```

6. Run the frontend with:

```
npx expo start
```

7. Scan the QR code or input the link starting with `exp://` into the Expo Go app. You should now be able to use the app via Expo Go!

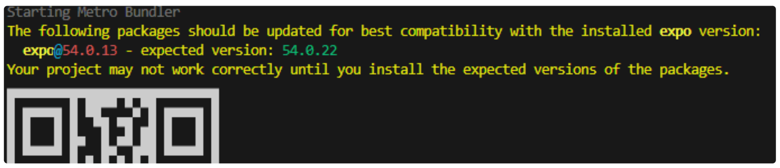
Whenever you want to use the app, repeat sections 4 and 5.

6. Updating packages

After running

```
npx expo start
```

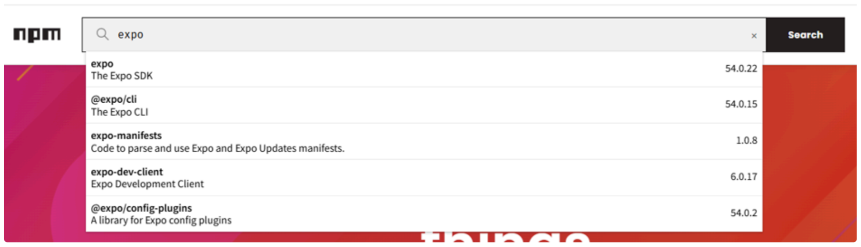
you may be prompted to update some packages. For example,



```
Starting Metro Bundler
The following packages should be updated for best compatibility with the installed expo version:
  expo@54.0.13 - expected version: 54.0.22
Your project may not work correctly until you install the expected versions of the packages.
```

To update them:

1. Stop the frontend by running `^C`
2. Head to https://www.npmjs.com/ and search for the package you need. For example,



expo
The Expo SDK 54.0.22

@expo/cli
The Expo CLI 54.0.15

expo-manifests
Code to parse and use Expo and Expo Updates manifests. 1.0.8

expo-dev-client
Expo Development Client 6.0.17

@expo/config-plugins
A library for Expo config plugins 54.0.2

3. Run the command provided to install the latest version of the package. In this case, it would be

```
npm i expo
```

   To install a specific version, you can use `@` to specify this. For example,

```
npm i expo@54.0.22
```

4. Once you've installed all the packages you need, make sure you're still in the `frontend` folder, and run

```
npx expo start
```

   The errors should be absent:

7. Expo Go version

Sometimes, the app may require you to download a specific version of the Expo Go app, which may may be an older version that is unavailable in the app store. You can download a specific version of the Expo Go app here: Expo Go - Expo

Potential Costs

Supabase

- We are currently using the Free tier
- Their price packages:
  - Free
    - 50,000 MAU (Monthly Active Users)
    - Database size: 500 MB
    - Storage size: 1 GB
    - Realtime Peak Connections: 200
  - Pro - $25 a month
    - 100,000 MAU included, then $0.00325 per MAU
    - Database size: 8 GB disk per project included, then $0.125 per GB
    - Storage size: 100 GB included, then $0.021 per GB
    - Realtime Peak Connections: 500 included, then $10 per 1000
  - More information:

    - Pricing & Fees | Supabase
    - About billing on Supabase | Supabase Docs
    - Pricing | Supabase Docs

GitHub Actions

- We are currently using the Free tier
- We have a relatively small number of changes to the repository, so it should be able to stay within the Free tier limits, however, here's the pricing information if needed:
  - GitHub Actions billing - GitHub Enterprise Cloud Docs

Future Improvements

Profile picture

- When researching how to implement this feature, we realised that photos would take up a lot of storage in the database. Since we are restricted by Supabase's free tier, we were unable to implement an effective way to achieve this feature
- To implement this in the future, consider upgrading to Supabase Pro or seeking alternative storage options (e.g., AWS)
- Since no profiles have profile pictures at the moment, the profiles currently display the user's initials as per the client's request

Deployment

- Unfortunately, we were unsuccessful in deploying the app on an external platform so the general public can use it. We tried deploying on free platforms such as Railway.com and Render.com, but they had issues with connecting to Supabase
- To deploy this app in the future, we recommend using a paid platform that is compatible with Supabase

Licensing

The PRANDA team (Amy Sun, Natalie Lam, Rishi Mukherjee, Patrida Anankathan, and Dylan Tran) gives the client (MJ Wei) full rights to this project. This project uses third party services including but not limited to: React Native, Expo, Supabase, Springboot, ngrok, etc., which may be subject to separate licensing agreements.