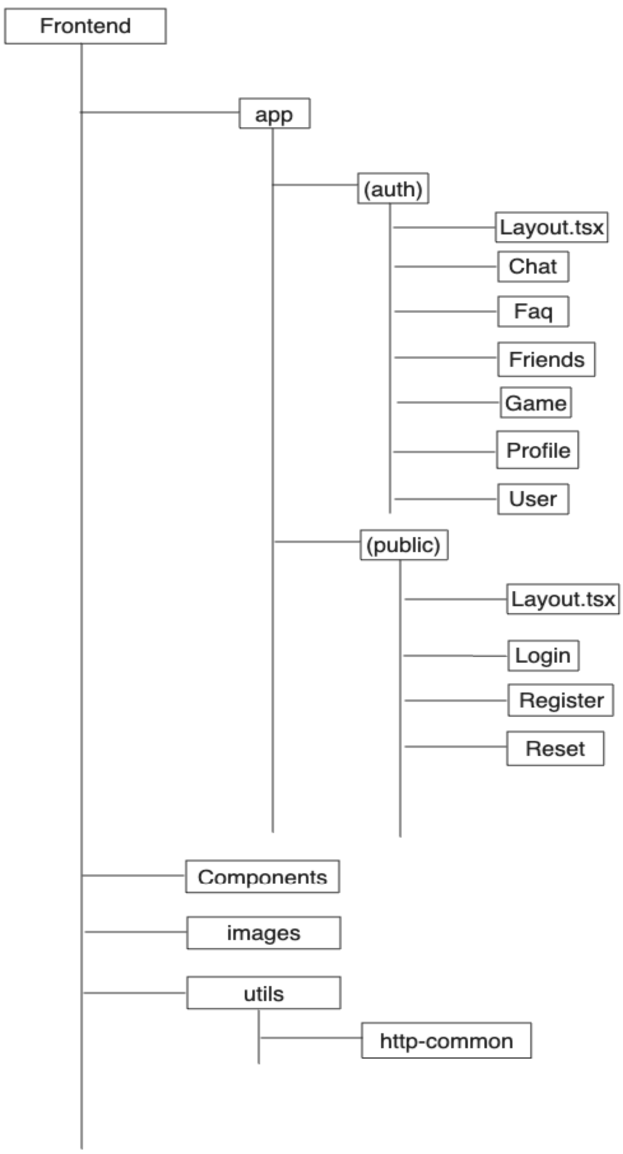
Technologies Used

- React Native
- Expo Router
- Clerk
- Nativewind + Tailwind.css
- Google Maps API
- Geolib

Project Structure

The folder structure for the frontend is structured in folders that ties to the functionality of the components encapsulated within the folders (EX: Folder named Chat contains all functionality and components related to chat).

(auth) folder - Core functionalities of the app, components accessible by user after authentication.
(public) folder - Contains components responsible for logging in and authentication
Components folder - contains reusable components for the application (barButtons, etc.)
Utils folder - Contains http-common, a file used to configure routes to connect to the backend



Setup and Installation

1. Node js , typescript should be installed on the developer’s machine.

2. Once the repository for the frontend is downloaded onto the developer's machine, run the “npm install” command to install all the necessary packages tied to the frontend.

3. Create a .env folder specifying two constants (refer to API keys setup for more details)
    a. **EXPO_PUBLIC_CLERK_PUBLISHABLE_KEY**
    b. **EXPO_PUBLIC_GOOGLE_MAPS_API_KEY**

4. Configure connection route to the backend by navigating to the utils/http-common.js file and configure the baseURL based on your development needs.

API keys setup

Before running the application, 2 API keys must be specified
Instructions for setting up API keys are shown below.

1. Create a .env file for storing environment variables

2. Paste the following lines into the .env file
   EXPO_PUBLIC_CLERK_PUBLISHABLE_KEY = <Clerk secret key>
   EXPO_PUBLIC_GOOGLE_MAPS_API_KEY = <Google Maps api key>

3. Create a Clerk account and create an API key for development and replace the **<Clerk Secret key>** with the key.
   Refer to the video below for more information about how clerk authentication can be implemented : https://galaxies.dev/react-native-authentication-clerk

4. Create a google maps API key and replace the **<Google Maps api key>** with the google api key.

Running the application

1. Run the backend application on your local machine / remote server.
2. For Mac Users, make sure Xcode is installed, then run an application named “simulator” to open up an Iphone Emulator. Refer to **step 3** if you’d like to run the application on an android emulator.
   Refer to video for more details : https://www.youtube.com/watch?v=DIoY4tyzKDA
3. For Windows Users , the application can only be run via an android emulator. Install android studio on your local machine and run an instance of an android emulator.
4. Once an emulator is running on your local machine, run the frontend application via the “npm start” command, then press “a” to run on your android emulator or “i” on your ios emulator.

API integration / API Calls

The general endpoint to connect to the backend can be configured in the http-common.js file.

Http requests are made through services. Under the each folder pertaining to a core functionality (EX: Chat folder), a folder named with the convention <functionality>Service.ts can be found. All api configurations to the backend are configured in a Service.ts file.