Code Documentation

This code documentation describes and explains the source code, the user manual with UI screenshots can be found here: https://github.com/lorifranke/RealEstateMobileApp/blob/master/User_Manual.pdf

Goal

Creating a mobile application that will be the interface to the existing backend component to connect to and extract data from the Realtor.com api is available here (https://rapidapi.com/apidojo/api/realtor). Display data of property, analyze investment and profit. A competing website is DealCheck (https://app.dealcheck.io/!/login/main).

Setup and Installation Instruction of the mobile App

This Real Estate App is written in TypeScript and JavaScript and built with the framework react native. It can be installed with Node.js using the node package manager. Download this folder and run `npm install` in the command line to get all packages. This will install the packages listed in the package.json file. Alternatively, you can also use `yarn` to install packages.

To run a mobile application, you will need to use `npm install -g expo-cli` for React-Native as well as a mobile phone emulator (unless you run the code as web application). For iOS, please use XCode with belonging phone simulator to run the app on a virtual phone. Another possibility is to use Expo CLI to run your React Native app on a physical device, therefore, just connect your phone with USB and run the app on your phone.

To inititalize an expo project, you will need to run `expo init YourProjectName` in the command line. This will link an expo project to the folder. Then, you can start the application with `npm start` and the browser will open the metro bundler where you are able to access different simulators. Another way to start the app is by directly launching the app on an Android Virtual Device by running `npm run android`, or on the iOS Simulator by running `npm run ios`. The simulator will open directly without using the metro bundler from the browser. Before opening the simulator make sure the backend is running (see link below to backend). The backend can be started in the corresponding folder as well with `npm start`.

A further detailed documentation on how to use and install React Native can be found here: https://reactnative.dev/docs/environment-setup and an introduction how to set up a first mobile app here: https://reactnative.dev/docs/getting-started.

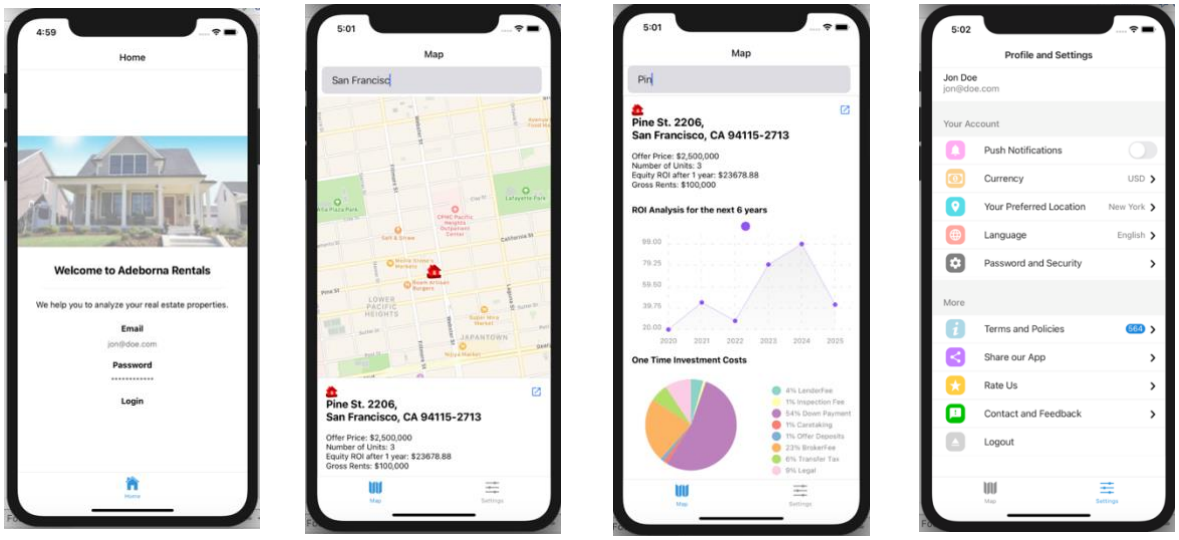
More resources on debugging the app while using React-native can be found here: https://reactnative.dev/docs/troubleshooting

See Readme.md as well for updates.

Source code and folder structure

React always requires importing the modules and packages, which is done in the top of each file. We can also import other files or components we have written in other files. The source code here is described by each of the different folders of this repository:

1. Screens:



This is the most important folder of the UI. It contains all main screens (or pages) of the mobile application. It contains different files, each belonging to one screen. The three screens we have in the app right now are called in the source code: tabone, tabtwo, tabthree. These are all linked in the navigation bar in the bottom of the UI. The UI screenshots and how the app's page/screen flow looks like can be found in the user manual of this repository. The login or home screen belongs to the TabOneScreen.tsx file. The profiles and settings are constructed in the TabTwoScreen.tsx file and the map is implemented with the TabThreeScreenPage. If we want to build additional pages that should be displayed in the app, we create a new screen file and put it in this Screens folder. In case we have some errors when displaying the app or call a screen is called that does not exist, there is the fallback screen NotFoundScreen.tsx for error handling.

1.1 TabOneScreen.tsx: This is the Login Screen, where the user enter his data and we call the backend database (MongoDB) within a fetch method and use the POST method for checking the users input (`fetch("http://localhost:5000/api/users/login",`

`{method: "POST",…..`). The server will send JSON content back as well as a token that we will use to identify the user that should exist in our database. We also catch possible errors with `.then((response) => {`… For example if there is a server error or if the user enter wrong input.

In react native it is important to understand how to work with states and props, these can be defined and given to single components. More details on this can be found here: https://reactnative.dev/docs/state.

On the bottom of this file, a few CSS-like styles are defined as a const. These are called in each component tag with style={styles.container} and enable styling in each of the screens.

1.2 TabTwoScreen: This is the settings page.

2. Assets:

Here we save all images and icons that we need (see image folder). We have for example the favicon, marker icons for the map, an icon for the waiting screen etc. The second folder in Assets contain files defining the fonts we use to make a nice design. These fonts can be found online and often exist for regular websites as well (e.g., Google Fonts), or can also be written by oneself.

3. Components:

This folder contains helper files that create little components of the app. Chevron.js, InfoText.js, index.js which are helper components for the Profile/Settings screen. Chevron.js creates the right icon that is displayed on each element to tab on in the Settings screen.

4. **Constants:**

   Here are files containing configurations for the color schemes and the overall layout of the app. React allows similar structure as CSS to style elements (see Colors.ts file).

5. **Navigation:**

   This folder contains all files that are necessary for the navigation in the mobile app, for example we have a bottom nav bar that navigates through the app (See BottomTabNavigator.tsx). All the screens (pages) we create *must* be linked in the navigation to be visible and usable. They have to be defined in a tag <BottomTab.Screen> otherwise they won't be displayed.

6. **Node-modules:** packages necessary for running the app. These modules which are in this folder are all listed in the package.json file as well and will be auto installed when running the setup part with `npm install` of this documentation.

property.json file: This file is created by the backend part of this project and is picked up by this UI. As soon as the backend is able to save multiple properties in a database, the UI can render this provided information from the property.json files. Depending on which information we have available from fetching data from the API connected to the backend, the UI will show the received information.

Backend:

This application is the User Interface (UI) part of a full software project. The backend belonging to this real estate mobile UI can be found here: https://github.com/MounikaChava304/CS682

Package.json, package-lock.json and yarn.lock are files that are necessary for installing and using the packages required for the app.