Application Overview

Demo link: https://expo.io/@urmajesty516/projects/freedom_travel

Wireframe: https://www.figma.com/file/bdwUrot2T35P6WiTiPh3OZ/Wireframes?node-id=0%3A1

Mockups: https://www.figma.com/file/xFbaDlYsEdGzDphYK6W6sX/Mockups?node-id=0%3A1

Platform: Android.

Phone model used for developing: Google Pixel 5.

Description:

This app allows User to browse different travel options of cities. Travel options can be filtered by a particular city or a category. User has to login or sign up to access the app features. User is able to view, add, edit, or delete any travel option under a particular city. Each travel option also includes add, edit, and delete photos feature from local gallery.

Data Structure:

City (Name)

Category (Name)

Place (User.fk, City.fk, Category.fk, name, introduction, address, photos, phone, email, website)

User (email, password, name)

Side Note:

Please run the app via the demo link. If you would like to run it locally, please make sure the local environment mode is switched to production.

Extra features

- I have made a NodeJS backend API service for the app and host it on https://freedom-travel-backend.herokuapp.com/.
- The app stores and fetches data from a separated backend service via `axios`.
- The app stores image data statically on the remote server. However, due to the ephemeral feature of the server, newly uploaded images will be deleted in the next cycle when the dyno restarts.

- The app has a splash screen as an image.
- The app loads custom font style that is not included in the expo package when the app starts.
- `useEffect` has been used multiple times in the app to initialize the data to display on the screen.
- Hide bottom tab button for specific screens via screen options: `tabBarButton: () => null`.
- Handle form validation manually by passing a validation handler to a form's validate property via `Formik`. This provides more flexibility on form validation, instead of integrating with `Yup`.
- Authenticates user's login via JWT token and saves the token in mobile's local storage, so it keeps the User's login state next time he/she accesses the app. This is achieved via `AsyncStorage`.
- After User has logged in, the app prevents User from going back to login screen by pressing back button on their phone. This is achieved by popping up an alert box asking User if he/she wants to stay on the screen or logout.
- Custom SVG icon components on Single City screen. This is achieved by `react-native-svg` package.
- Custom category slider where User can tap the arrows to scroll to view on Single City Screen.
- Google map on Single Place screen. However, it hasn't been integrated to show input address on the map. This is achieved by react-native-maps.
- Custom `ImageSwiper` integrated with gesture handler on Single Place screen.
- User is able to upload or edit or delete photos from their local camera gallery to the app on Edit/Add Listing screen.
- Custom City and Category filter with AND logic on My Travel screen.
- Pull up to fresh or initialize the screen's state after filtering on My Travel screen and Single City Screen. This is achieved by `ScrollView`.
- I used `widthPercentageToDP` and `heightPercentageToDP` from `react-native-responsive-screen` to covert width and height percentage to device-independent pixels for some layout design purposes.