User Guide

Sign up and sign in

1. Open “My Account”
2. If you are not currently signed in, you may sign in using email/password, or one of the social sign in methods.
3. If you use multiple sign in methods that share the same email, your account will merge these sign in methods

Known issues:
Once signed up using Facebook, the user cannot sign in with Google

Add campsites

1. Select “Add a site”
2. If you are already located in a park, the “Park Name” field will be filled in. (i.e. your location is already determined by the device location)
3. Specify a name for the campsite by filling in the “Site Name” field.
4. Make sure the type is “Campsite”

Add spotlight site

1. Follow steps 1-3 in Add campsites
2. Set the type to “point of interest”

Add post

1. Select the guestbook on which you wish to add your post.
2. Open “Add an entry”, you will be prompted to select a post type (video, image, and audio), as well as enter your notes. You may omit uploading any media and create a text-only post.

Admin

1. The test user provided during project handoff is configured as admin.
2. At this time, the only way to add new admins is by inserting entries into the `permissions` MongoDB collection manually.

Deployment and Setup Instructions

Google Cloud

1. Billing account: Add a new billing account if needed, ensure that billing is enabled for the `my-campsite` project. Check permissions to the billing account if needed.
2. IAM:
    2.1. Service accounts: development and production accounts are named with prefixes `devel-` and `prod-dx-` respectively. These service accounts have access to resources with the same prefixes.
3. Cloud storage: objects for storing media are publicly readable.
4. Identity Platform: configure OAuth client IDs and/or secrets that Firebase will use to sign in users. Since social sign in flows are handled by `expo-auth-session` and its proxy, redirect URLs shown here are not used.
5. API credentials: configure Sign in with Google OAuth clients and redirect URLs as needed. Currently, the clients and URLs are configured for the `expo-auth-session` proxy.
6. Secret Manager: keystores for signing deliverable 3 Android applications are found here.

Other Third-party Services

1. Heroku: production Express instances. Configure environment variables as required in the “Express (production setup)” section.
2. Facebook Developer: OAuth clients configuration
3. MongoDB Atlas: production database instances
4. OpenWeather: API key

Express (development setup)

1. Download your service account key from the development service account on Google Cloud.
2. Create a text file named `.env` in the root directory of the express app, and add the following variables:
    2.1. GOOGLE_APPLICATION_CREDENTIALS: path to your service account key.
    2.2. OPEN_WEATHER_MAP_KEY: API key for the weather API.
3. Configure and start a local MongoDB instance:
    3.1. Default port (`27017`)
    3.2. Database: `301project`
    3.3. No authentication

Express (production setup)

1. Download your service account key from the production service account on Google Cloud.
2. Save the service account .json content as a base64 string.
3. Configure the environment variables:
    3.1. GCP_SERVICE_ACCOUNT_KEY_ASC: base64 value of your service account key.
    3.2. OPEN_WEATHER_MAP_KEY: API key for the weather API.
    3.3. MONGODB_URI: URI to the production MongoDB instance.
4. Configure the production MongoDB instance to match the configuration in MONGODB_URI.

Expo (development on frontend)

1. Obtain access to an **Expo** account @ https://expo.dev/
2. Install the latest version of `expo-cli` on your machine.
3. Navigate to the frontend directory (i.e. /expo) of the My Campsite project
4. Run commands (login when prompted)

    ```
    expo login
    expo publish
    ```

    If using a release channel use: `expo publish --release-channel <channel>`
5. View your app on **Expo Go**. (Download this from the iOS App Store/ Google Play)
    a. **Note**: We recommend testing on a **physical device** as Three.js insists.

Feature Documentation

- This section explains the high-level functionality of each component (i.e. its interconnected parts in frontend and backend) and small caveats of working with the app
- **Note: Feature documentation below is only accurate since CSC301's December 2021 offering.**

Maps

- Using `react-native-maps` as a cross-platform option for displaying maps on Android and iOS.
    - Relevant components: MapScreen,
    - We are using `expo-location` to get latitude and longitude coordinates of the user's device. These coordinates are then passed into the MapView component to render the map.
        - Toggle the boolean flag passed into the component during navigation with `react-navigation` to switch between **demo mode** and **live mode**.

Three.js

- We use a ported version of Three.js called `expo-three` that allows for loading 3d models (i.e `robot.glb` under the `/assets/models` folder.) in Expo.
    - Relevant components: MapScreen
    - 3d model loading is independent from Map (i.e. Google Maps, Apple Maps) functionality and is purposely synced up to match map movements by storing the camera as a ref with `useRef`.
        i. We use a ref to store a state of the camera instead of useState to get work around component refresh.
    - The functionality of the 3d model is that it is always centered and adjusts to `react-native-maps` MapView camera rotation, tilt, and zoom using estimation by spherical coordinates.
- Note: Although it may be possible, we were not able to get `.obj` and `.mtl` files loading properly through Expo Three. (so unless it is resolved later, stick to using `.glb` models)