

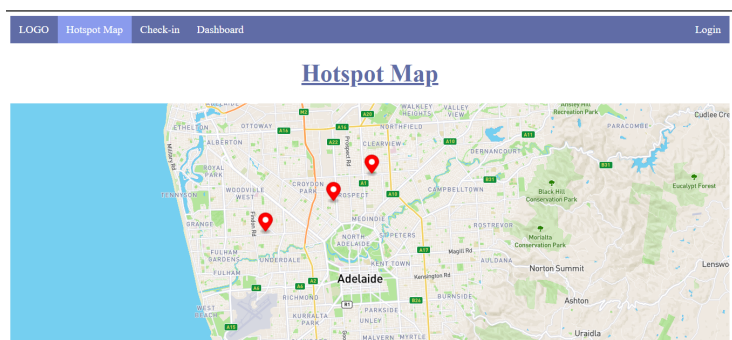
Current State of the web app: check-in.html



This page will be the main page of the website. GET “/” will redirect to this page to reduce kinematic load since this is the feature that will be most used in our web app. This page has been made adaptive using Vue.js. It has two states, signed in and not

signed in. If the user is not logged in, they will see the extra input fields where they can put their information. If they are logged in, they only see the field for check-in code. Allowing users who aren’t signed in to check-in was a feature we decided to implement when doing research on other contact tracing websites. We also designed our database to work with this approach.

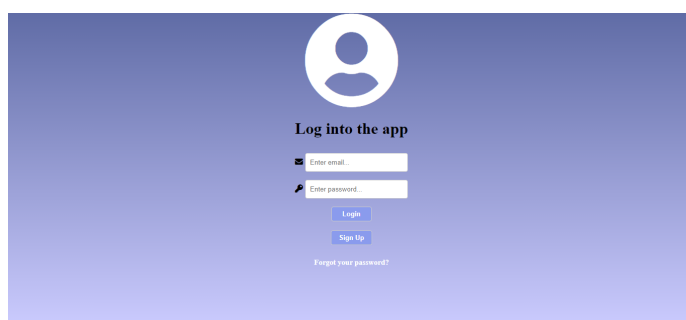
hotspot-map.html



In this page we display an interactive map, originating at Adelaide, which has a couple of marlings on the map where hotspots were created by admins. We require admins to give the address of the hotspot but to put it on the map we need longitude and latitude of

the location. To make it simple on health officials creating the hotspots, we will be using a public API that turns addresses to longitude and latitudes with sufficient accuracy when we tested it.

login.html



This is the page where users can sign in or choose to sign up to our website. It functions the same as many other websites.

user-signup.html & Business_signup.html

[Create a new business account?](#)

Create a new account

First name* :

Last name* :

Phone number* :

Passport/IC* :

Email* :

Password* :

Confirm Password* :

Sign Up

Password must contain the following:

✗ A lowercase letter

✗ A capital (uppercase) letter

✗ A number

Create a new business account

First name* :

Last name* :

Company name* :

Phone number* :

Building Name* :

Street* :

Zip Code* :

City* :

Country* :

Email* :

Password* :

Confirm Password* :

Sign Up

Password must contain the following:

✗ A lowercase letter

✗ A capital (uppercase) letter

✗ A number

Both are sign up pages that allow the respective user to create an account of their desired type. Both pages allow sign-ups using various social media platforms.

dashboard.html

LOGO

Hotspot Map

Check-in

Dashboard

Switch Account Type (Now: admin) #DEMO ONLY#

Talhah

Profile

Hotspots

Venues

Users

Signup an Admin

Profile

First Name

Last Name

Edit

Talhah

Zubayer

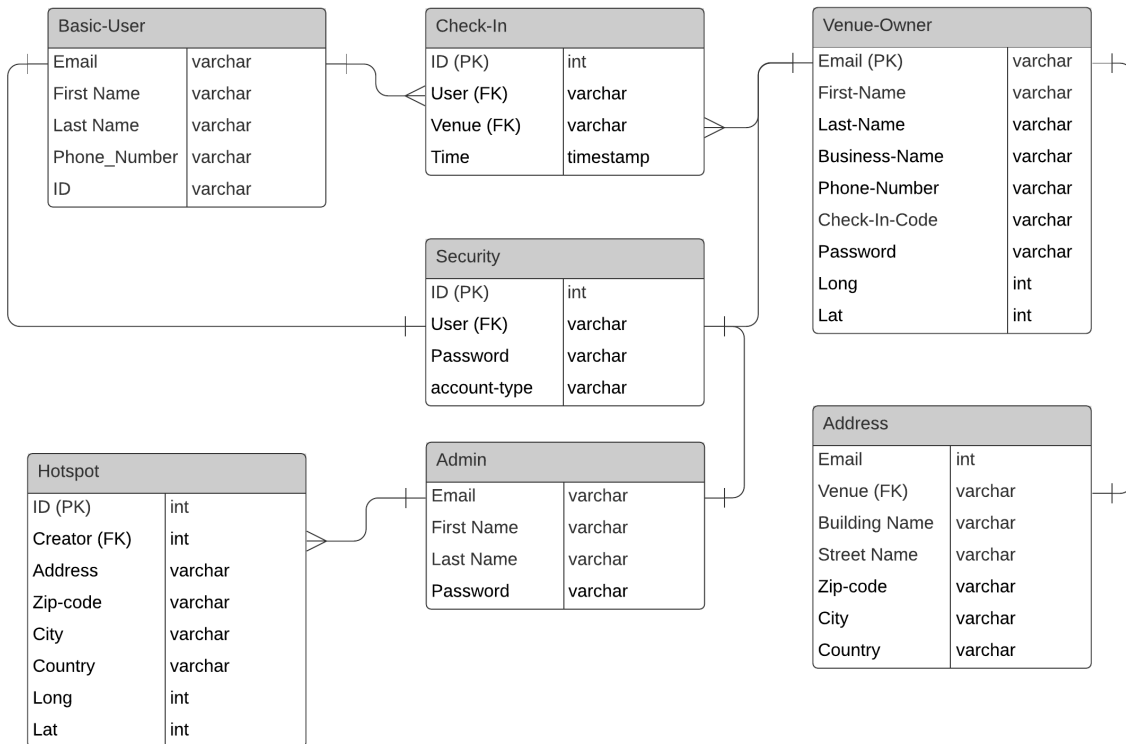
Email:

abc@gmail.com

Reset Password

This page is by far the most technically challenging page on the website. It houses the dashboard for 3 different account types (user,venue,admin/health-official). Each user type has their own functionalities associated with them which are represented by a vertical list of tabs. The page heavily uses vue to adapt to the account type and the required functionalities. Since we do not have server side code at the moment, we decided to implement a button on the top right of the screen that lets you change the account type for this page. This way you can see the different functionalities for each account type displayed using fake data.

Database Schema:



This database schema represents what we thought to be the best data structure for our web application.

Routes

- GET /hotspots.ajax
 - This route will return a JSON list of hotspots from the database. This route is crucial for hotspot-map.html
- POST /check-in.ajax
 - Headers: check-in-code / first-name / last-name / phone-num / id
 - This route will be called by hotspot-map.html and will add a row to the check-in table of the database. It will also use the user's cookies to see if they were logged in or not. If they were, it would take the needed details from the database. If they are not signed in, it will look for the optional headers. If it encounters any issues such as invalid check-in code or no optional headers when not signed in, it will respond with a 4xx response code.
- POST /login.ajax
 - The route that will be called for logins. It will handle session tokens which will be stored in cookies so that the user remains logged in.
- POST /user-signup.ajax
 - The route will handle sign-ups. Once the user is successfully signed up and added to the database, the route will return a 2xx response code and redirect the client to the log-in page.
- POST /business-signup.ajax
 - The route will handle business sign-ups.
- POST /admin-signup.ajax
 - The route will handle admin and health official sign-ups
- POST /reset-password-code.ajax
 - The route will email the user a unique verification code.
- POST /reset-password.ajax
 - The route will compare the given verification code and change the user's password to the given password.
- GET /dashboard
 - This route will be used any time the user is redirected to the dashboard. Since the dashboard requires the user to be signed in, this route will make sure they are signed in before sending them dashboard.html. If they are not signed in they will be redirected to login.html.
- GET /profile.ajax
 - This route will send the client a JSON object of information based on the user's sign in information. Data will vary based on the type of the user. The data that will be sent is the same as the fields shown in the profile tab of the dashboard for the given account type.
 - User: ID, F-Name, L-Name, Number, Email, Email Preferences
 - Venue: Check-In Code, F-Name, L-Name, B-Name, Building, Street, Zip Code, City, Country
 - Admin: F-Name, L-Name, Email
- GET /user-history.ajax

- This route will use the client's session details to identify the user. Then it will send all the rows from the check-in table that belong to the user. This will be in the form of objects in a JSON formatted array.
- GET /venue-history.ajax
 - This route will do the same task as user-history.ajax but for a venue.
- POST /email-pref.ajax
 - This route will take the email preferences and change them in the database for the user who is identified using the session details.
- GET /venue-details.ajax
 - This route will return the details of every venue in the database in a JSON formatted array of objects. Only allowed for admins so the session details will be used to verify them.
- GET /user-details.ajax
 - This route will do the same as venue-details.ajax but for users.
- POST /update-info.ajax
 - This route will take in the information of a user,venue,admin or hotspot and update the database. The data sent to the route must specify what type of account it is updating and all the necessary information. If it cannot update information it will respond with a 4xx code. Admins can freely use this route in through the dashboard to update users and venues. This route will also be used for the profile tab of the dashboard in which case, the session details will be used to ensure the account they are updating is their own.
- POST /add-hotspot.ajax
 - This route will be used to create new hotspots in the database.