Module 9 Challenge

External APIs allow developers to access their data and functionality by making requests with specific parameters to a URL. Developers are often tasked with retrieving data from another application's API and using it in their context, frequently consuming this data via a server.

Your Challenge this week is to build a weather dashboard application that calls the OpenWeather API and renders data in the browser.

Weather Dashboard

The application's front end has already been created. It's your job to build the back end, connect the two, and then deploy the entire application to Render.

- Use the 5-day weather forecast at https://openweathermap.org/forecast5 to retrieve weather data for cities.
- The base URL should look like the following:
  `https://api.openweathermap.org/data/2.5/forecast?lat={lat}&lon={lon}&appid={API key}`
- After registering for a new API key, you may need to wait up to 2 hours for that API key to activate.
- For more information on how to work with the OpenWeather API, refer to the Full-Stack Blog (https://coding-boot-camp.github.io/full-stack/apis/how-to-use-api-keys) on how to use API keys
- Before you start, make sure to download and unzip the starter code files and make your own repository with the starter code.

User Story & Acceptance Criteria

User Story

AS A traveler
I WANT to see the weather outlook for multiple cities
SO THAT I can plan a trip accordingly

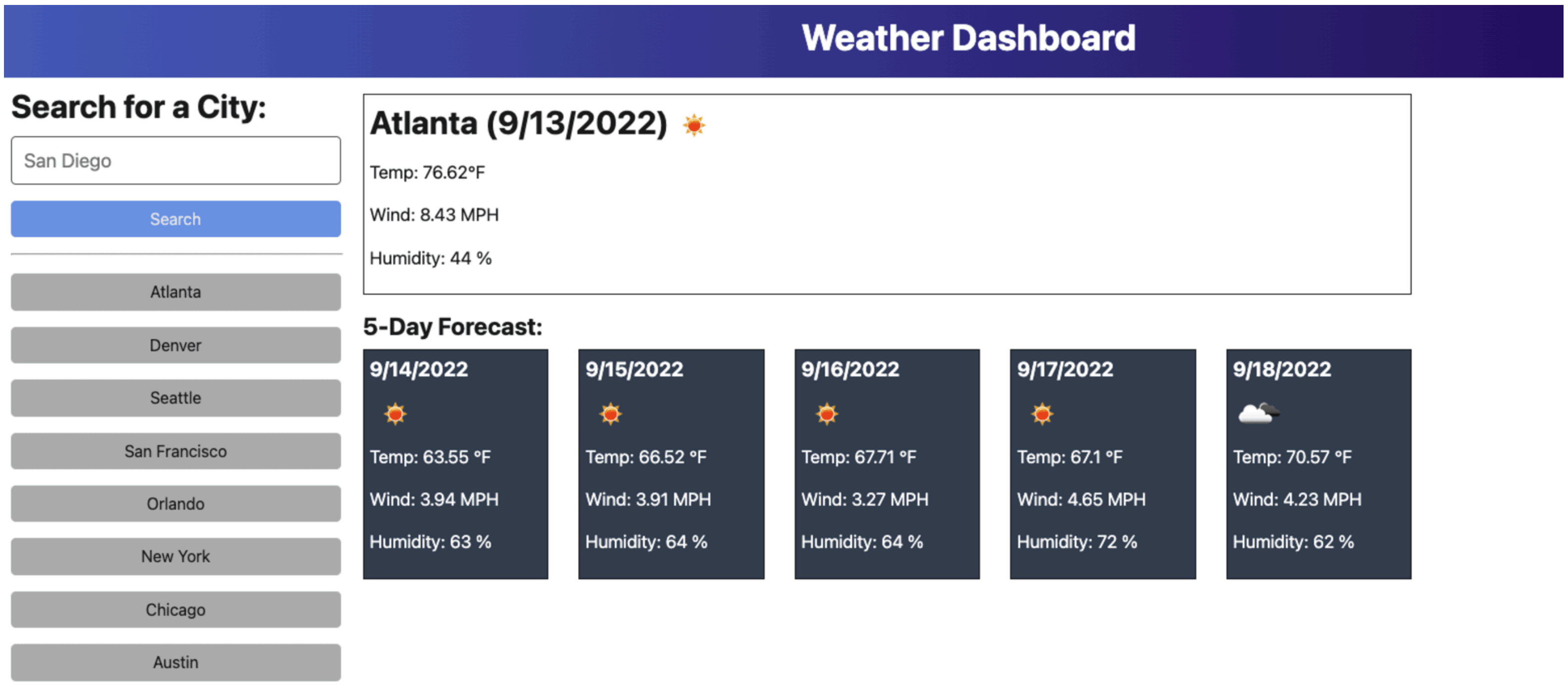
Acceptance Criteria

GIVEN a weather dashboard with form inputs
WHEN I search for a city
THEN I am presented with current and future conditions for that city, and that city is added to the search history
WHEN I view current weather conditions for that city
THEN I am presented with the city name, the date, an icon representation of weather conditions, a description of the weather for the icon's `alt` tag, the temperature, the humidity, and the wind speed
WHEN I view future weather conditions for that city
THEN I am presented with a 5-day forecast that displays the date, an icon representation of weather conditions, the temperature, the wind speed, and the humidity
WHEN I click on a city in the search history
THEN I am again presented with current and future conditions for that city

Mock-Up

Image

The following image shows the web application's appearance and functionality:



Weather Dashboard

Search for a City:

San Diego

Search

Atlanta

Denver

Seattle

San Francisco

Orlando

New York

Chicago

Austin

Atlanta (9/13/2022)

Temp: 76.62°F

Wind: 8.43 MPH

Humidity: 44 %

5-Day Forecast:

9/14/2022	9/15/2022	9/16/2022	9/17/2022	9/18/2022
Temp: 63.55 °F	Temp: 66.52 °F	Temp: 67.71 °F	Temp: 67.1 °F	Temp: 70.57 °F
Wind: 3.94 MPH	Wind: 3.91 MPH	Wind: 3.27 MPH	Wind: 4.65 MPH	Wind: 4.23 MPH
Humidity: 63 %	Humidity: 64 %	Humidity: 64 %	Humidity: 72 %	Humidity: 62 %

Mock Up Image Description

This weather dashboard mock-up displays weather information retrieved from an external API, specifically the OpenWeather API, using a clean and structured user interface. The application's purpose is to allow users to search for cities and view their current weather conditions along with a five-day forecast.

Key Features and Functionalities

1. Search for a City

- A text input field where users can type the name of a city.
- A "Search" button that triggers an API call when clicked.
- The city searched is added to a search history list.

2. Search History

- A list of previously searched cities displayed as buttons.
- Clicking a button retrieves and displays the weather for that city.
- Cities are stored persistently using a JSON file on the backend.

3. Current Weather Display

- Shows the selected city and date.
- Displays an icon representing the current weather condition.
- Provides:
  - Temperature
  - Wind Speed
  - Humidity

4. Five-Day Forecast

- A horizontal row of forecast cards displaying:
  - Date
  - Weather icon
  - Temperature
  - Wind Speed
  - Humidity

5. API Integration

- The app uses OpenWeather API to fetch current and forecast weather.
- Requires converting city names to latitude and longitude coordinates.
- Data retrieved is processed and displayed dynamically.

6. Data Storage and Retrieval

- A searchHistory.json file is used to store previously searched cities.
- Provides options for retrieving and deleting stored searches.

Getting Started

On the back end, the application should include a `searchHistory.json` file that will be used to store and retrieve cities using the `fs` module.

The following HTML route should be created:

- `GET *` should return the index.html file.

The following API routes should be created:

- `GET /api/weather/history` should read the searchHistory.json file and return all saved cities as `JSON`.
- `POST /api/weather` should receive a city name to save on the request body, add it to the `searchHistory.json file`, and then return associated weather data to the client. You'll need to find a way to give each city name a unique id when it's saved (look into npm packages that could do this for you).

Refer to the Full-Stack Blog on deploying to Render at https://coding-boot-camp.github.io/full-stack/render/render-deployment-guide and the Render documentation on setting environment variables at https://render.com/docs/configure-environment-variables.

Hints

- Using the 5-day weather forecast API, you'll notice that you'll need to pass in coordinates instead of just a city name. Using the OpenWeatherMap APIs, how could we retrieve geographical coordinates given a city name?
- How could we make the OpenWeather API calls server-side, parse the data, and then send the parsed data client-side?

Bonus

This application offers the DELETE functionality on the front end. As a bonus, try to add the DELETE route to the application using the following guideline:

`DELETE /api/weather/history/:id` should receive a route parameter that contains the id of a city name to delete. To delete a city, you'll need to read all the cities from the `searchHistory.json file`, remove the city with the given id property, and then rewrite the cities to the `searchHistory.json file`.

Grading Requirements

If a Challenge assignment submission is marked as “0”, it is considered incomplete and will not count towards your graduation requirements. Examples of incomplete submissions include the following:

- A repository that has no code
- A repository that includes a unique name but nothing else
- A repository that includes only a README file but nothing else
- A repository that only includes starter code

This Challenge is graded based on the following criteria:

Technical Acceptance Criteria: 40%

- Satisfies all of the preceding acceptance criteria plus the following:
- Application uses the OpenWeather API to retrieve weather data.
- Application back end must store cities that have a unique id in a JSON file.
- Application must be deployed to Render.

Deployment: 32%

- Application deployed at live URL.
- Application loads with no errors.
- Application GitHub URL submitted.
- GitHub repository that contains application code.

Application Quality: 15%

- Application user experience is intuitive and easy to navigate.
- Application user interface style is clean and polished.
- Application resembles the mock-up functionality provided in the Challenge instructions.

Repository Quality: 13%

- Repository has a unique name.
- Repository follows best practices for file structure and naming conventions.
- Repository follows best practices for class/id naming conventions, indentation, quality comments, etc.
- Repository contains multiple descriptive commit messages.
- Repository contains a quality README file with description, screenshot, and link to deployed application.

Bonus

Fulfilling the following can add 10 points to your grade. Note that the highest grade you can achieve is still 100:

- Application allows users to delete cities.

How to Submit the Challenge

You are required to submit BOTH of the following for review:

- The URL of the functional, deployed application.
- The URL of the GitHub repository. Give the repository a unique name and include a README describing the project.