06 Server-Side APIs: Weather Dashboard

Your Task

Third-party 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 the context of their own. Your challenge is to build a weather dashboard that will run in the browser and feature dynamically updated HTML and CSS.

Use the 5 Day Weather Forecast 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.

Hint: Using the 5 Day Weather Forecast API, you'll notice that you will 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?

You will use `localStorage` to store any persistent data. For more information on how to work with the OpenWeather API, refer to the Full-Stack Blog on how to use API keys.

User Story

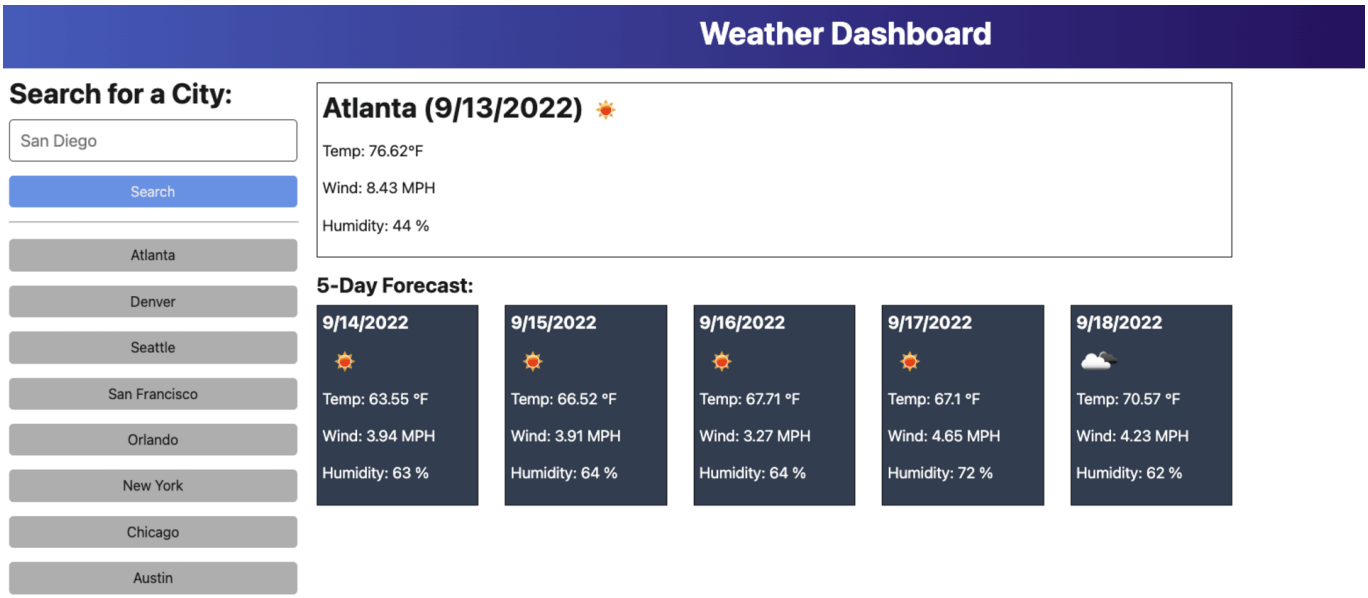
```
AS A traveler
I WANT to see the weather outlook for multiple cities
SO THAT I can plan a trip accordingly
```

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, 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

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



Grading Requirements

> **Note**: 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 above acceptance criteria plus the following:
  - Uses the OpenWeather API to retrieve weather data.
  - Uses `localStorage` to store persistent data.

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 quality readme file with description, screenshot, and link to deployed application.

Review

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.
