Assignment 9 Report

Team: runtime-terror

Author(s): Kia Afzali, Victoria Delk, Rohit Ghosh, Yanyu Wang, Charles Zhang

Objective

We are implementing a small web application that allows users to input a city in the world and retrieve current and/or future weather information for that city. We are doing this so that users have a simple way of obtaining weather information. Current weather apps can be very confusing and often inundate users with too much data. Our app would be very easy to use and allow users to efficiently gain access to weather information. Users should be able to access an HTML form from a variety of browsers that allows them to submit a location, a nearby date, and a unit of temperature, submit the form, and have the projected temperature returned to them. Users should then be able to save notes on the weather and access them later. This application will not allow users to pick and choose what information they see or display any trends over periods of time. This application is not intended to have a complex frontend or be used on the command line.

Specific Contribution

Our two handlers (`WeatherHandler` and NotesHandler), along with the HTML forms/JavaScript they serve, were developed by us. To support our idea, we made use of the Open-Meteo weather API to geocode human-readable locations into latitude/longitude and to fetch the weather data that we're serving.

Tools

In this assignment, our team made use of multiple technologies for the first time. This was our first experience serving HTML forms to the user with the intention of interacting with them. This meant we had to learn to use fetch and query strings to communicate information between the JavaScript and our server, and had to think about the separation of frontend from backend. We also had the new experience of interacting with a third-party API (Open-Meteo), which gave us more experience in reading documentation and integrating new code with our existing codebase.

Methodology

(links to new handlers/classes and testing)

Weather Handler:

https://code.cs130.org/plugins/gitiles/runtime-terror/+/refs/heads/main/src/weather_handler.cc

This is a new handler class called `WeatherHandler` that listens on the `/weather` endpoint and is used to serve our HTML markup and make our third-party API calls.

The handler begins by serving a form `weather_form.html`, which allows the user to specify a location, a unit of temperature, and a date range. This form then sends the input back to our server through a query string. The server will then parse this query string to extract the information needed to make calls to the Open-Meteo API. It will then verify that the information submitted was valid before making the API calls (geocode → forecast).

After receiving a response from Open-Meteo, the handler then serves a form `notes_form.html`. This is used to display the response to the user's weather query, while also providing them with an HTML form to save a note about the weather response. This form submits its data in JSON format in the body of a POST request to our `NotesHandler`.

Notes Handler

https://code.cs130.org/plugins/gitiles/runtime-terror/+/refs/heads/main/src/notes_handler.cc

To handle the user's ability to save and access notes, we made a `NotesHandler` class mounted at `/notes` that saves notes as files in the `/notes` directory.

This handler is essentially a modified version of `CRUDHandler` that accepts `GET`, `POST`, `DELETE`, and `LIST` requests. It handles these requests by checking for a `username`, `title`, and `note` content in HTTP requests it receives and then saves/accesses them in the underlying filesystem. To make it easy for users to access their notes, it also serves an HTML markup at `/notes?username=${USERNAME}` that displays a list of notes saved under that username.

Evaluation

On immediate observation, we've seen that our weather application seems to function about as well as we expected. We're able to access the landing page through the `/weather` route, we're able to make well-formed requests to the Open-Meteo API, and we're able to view the response to the call. In addition, our notes seem to work well, with them saving with no problems and being easy to access at `/notes?username=${USERNAME}`. Ideally, we would improve our styling and shore up our error handling so that our server is more resilient to malicious inputs that may crash our server. In addition, it would be nice to have real user authentication instead of simply associating notes with usernames that anyone can access.