Final Project

CSCI395 Web Development

Due: December, 20th 2024

1 Overview

In this project, students are expected to build a website using the Express/Node.js platform, with the Axios HTTP client, that integrates a chosen public API from the given list: Public API Lists. The website should interact with the chosen API, retrieve data, and present it in a user-friendly manner.

2 Objectives

- Develop an understanding of how to integrate public APIs into web projects.
- Gain practical experience using Express/Node.js for server-side programming.
- Enhance understanding of client-server communication using Axios.
- Demonstrate ability to manipulate, present, and work with data retrieved from APIs.
- Persist data using PostgreSQL database.

3 Example Idea: Weather Tracker Web Application

This web application allows users to monitor the current weather each time they log in. The application integrates with the OpenWeatherMap API to fetch real-time weather data.

Key Features

1. User Data Management

- Each user's email, password, and zip code are securely stored in a database.

- The zip code is used to determine the user's location and fetch corresponding weather details.

2. Weather History Tracking

- A dedicated **"History"** tab records and displays all instances when users accessed the weather information.
- This feature provides users with a complete log of their activity.

3. Activity Recommendations (Bonus Feature)

- The application includes an additional tab in the navigation bar that suggests random activities based on the current temperature.
- Activities are sourced from two tables in the database:
  - **"Warm Activities"** for higher temperatures.
  - **"Cool Activities"** for lower temperatures.
- These suggestions enhance user engagement by tailoring recommendations to the weather conditions.

4 Requirements

4.1 API Choice

- Browse through the provided list and choose an API of interest. This choice should be guided by the potential to retrieve, manipulate, and present data in a meaningful and interactive way. I recommend choosing an API that does not require authentication and is CORS enabled.(What is CORS?)

4.2 Project Planning

- Think through your project, researching the API documentation, project features, what data you will store, and how it will be used in your web application.

4.3 Project Setup

- Set up a new Node.js project using Express.js.
- Include Axios for making HTTP requests.
- Include EJS for templating.
- Ensure that the project has a structured directory and file organization.
- Include pg for working with your localhost PostgreSQL database

4.4 API Integration

- Implement at least a GET endpoint to interact with your chosen API.
- Use Axios to send HTTP requests to the API and handle responses.

4.5 Database integration

- Ensure users of your web application can create accounts to access certain information
- Create a table relevant to the topic of your website, and include a tab in the navigation bar of your web application to display this table. Your web application should also provide a means for adding data to this table.

4.6 Data Presentation

- Design the application to present the retrieved data in a user-friendly way. Use appropriate HTML, CSS(bootstrap if you want), and the templating engine EJS.

4.7 Error Handling

- Ensure that error handling is in place for both your application and any API requests. You can console log any errors, but you can also give users any user-relevant errors.

4.8 Documentation

- Include comments throughout your code to explain your logic.
- Include a Readme.md file that explains how to start your server, what commands are needed to run your code. e.g. **npm i and then nodemon index.js**