Clay Carpenter
GitHub Link: ClayCarp/Web-Framework-1

High-Level Purpose Statement

The purpose of this project is to build a simple, functional web application using Node.js and JavaScript that helps users decide where to eat by randomly suggesting restaurants from a chosen food category. Users can also mark restaurants as favorites, which are saved locally for future reference. A key feature is the embedded Google Map that visually displays restaurant locations, enhancing usability. The application is containerized with Docker to ensure consistent development and deployment environments. Docker eliminates the “it works on my machine” issue by encapsulating the application and its dependencies in a portable container, streamlining collaboration and deployment across systems. This is particularly useful as the app expands in complexity.

Experimental Design

The core functionality is a restaurant recommendation system that utilizes JavaScript classes to generate suggestions based on a selected category. Users can view restaurant names and save favorites, which are reflected in a dedicated favorites page (favorites.html). Front-End: Built with HTML, CSS, and vanilla JavaScript for simplicity and clarity. Back-End: Powered by Node.js to serve static files and manage routing. Map Integration: Google Maps embeds show restaurant locations, enhancing decision-making. Containerization: The app runs inside a Docker container using a Node.js base image, ensuring uniform behavior across devices. Live development is supported through volume mounts, making it easy to test updates without rebuilding. While the current restaurant data is hardcoded for simplicity, the design anticipates future integration with APIs like Yelp for real-time, location-based suggestions.

Resources Available

- JavaScript Documentation
- Node.js Documentation
- YouTube video tutorials for Express.js and Docker
- Code from previous web framework assignments
- Docker Documentation, including guides on Dockerfile creation, docker-compose, and volume mounting

Time Estimate

The estimated time for this part of the project was around 2 hours. Since the base functionality was completed in a prior assignment, this phase focused on implementing the favorites page and Dockerizing the entire application. Docker added about 30 minutes of setup time, including writing the Dockerfile, creating the docker-compose.yml, and adjusting the project structure. However, this setup greatly simplified future development and testing, especially across different systems.

Experiment Notes

Developing this project using Node.js and JavaScript was straightforward due to prior experience. The front-end logic for displaying restaurants and saving favorites was implemented in script.js, while the back-end handled routing and file serving. Docker was used to encapsulate the Node.js app using a custom Dockerfile. A docker-compose.yml file simplified startup and port mapping. Volume mounts (volumes: in docker-compose.yml) allowed real-time editing of HTML and JS files during development without rebuilding the image. Avoiding a database helped keep things simple; favorite selections are stored using localStorage. Overall, Docker enabled smoother collaboration and ensured the app behaved the same regardless of where it was deployed.

Results

The web application was completed and works as expected: Users can select a food category and receive random restaurant suggestions. Favorites can be saved and viewed on a separate page. Google Maps is embedded for each selected restaurant. The application runs reliably across machines due to Docker containerization. Dockerization was a key strength—it simplified setup, provided predictable behavior, and streamlined deployment, especially useful for future API integration and testing.

Consequences for the Future

This project lays a solid foundation for future improvements: Dynamic Data: Replace hard coded restaurant lists with API calls to services like Yelp or Google Places. Persistent Storage: Use a database to store user favorites across sessions. UI Improvements: Add search filters, animations, and real-time map updates for better UX. The current Docker setup ensures any future additions can be tested and deployed with minimal hassle, and new contributors can spin up the app without worrying about setup inconsistencies.

Steps for Setup

1. Run npm install to install dependencies.
2. Install Axios with npm install axios for making API requests.
3. Run the server with Node index.js
4. Click the hyperlink in the terminal to access the web app.