Mini Project 2: Full-Stack Web Application

Due Date: December 19th, 2024

Objective

Build a fully functional web application that integrates a React frontend with an Express backend. The goal is to create a seamless interaction between the client (frontend) and server (backend), applying the concepts learned in Modules 5 and 6.

Project Requirements

1. **Frontend (React):**
   - Use React to create a dynamic and responsive user interface.
   - Employ React features like components, props, state management, event handling, and conditional rendering.
   - Design a form in React for user input (e.g., data submission or queries).
2. **Backend (Express):**
   - Set up an Express server that handles at least 3 routes (GET, POST, PUT/DELETE).
   - Implement CRUD operations to interact with a simple in-memory or file-based dataset.
   - Use Express middleware to handle JSON requests.
3. **Integration:**
   - Use `fetch` or `axios` to send requests from React to your Express server.
   - Display responses dynamically in the frontend (e.g., render lists, tables, or alerts based on API responses).

Example Application Ideas

- **Task Manager:** Users can create, view, update, and delete tasks. Use React forms for task creation and editing, and display tasks in a list or table.
- **Movie Library:** Create a collection of movies. Allow users to add new movies, fetch details of a movie, and delete movies from the list.
- **Simple Calculator:** Perform server-side operations like addition, subtraction, multiplication, and division, displaying results dynamically on the frontend.
- **Weather Dashboard:** Use an external API for data (optional). Allow users to enter a city name and display the weather details.

Submission Checklist

1. **Codebase:** A GitHub repository with a clear README explaining the project, its structure, and how to run it.
2. **Demonstration:** A short recorded demo (or live presentation) of your app, showing both backend and frontend interactions.

Evaluation Criteria

1. **Functionality**: Does the app meet the outlined requirements and perform the intended actions?
2. **Code Structure**: Is the code well-organized with a clear separation of concerns? Are React components and Express routes modular and reusable?
3. **User Experience**: Is the UI intuitive and responsive? Is data presented in a clear and meaningful way?
4. **Creativity**: Is the idea unique or implemented with an interesting twist?