Project Instructions: Developing a Dynamic Single-Page Website

Overview

In this project, you will develop a dynamic single-page website that incorporates JavaScript, Web APIs, and WebSockets. You are free to choose the topic of your web app, but it should be engaging and useful.

Requirements

JavaScript Functionality

- Use JavaScript to manipulate the DOM and dynamically update content.
- Make use of ES6+ features such as let/const, arrow functions, and template literals.

Web API Integration

- Select a suitable Web API that fits your chosen topic.
- Fetch data from the API using the Fetch API or Axios.
- Display the fetched data dynamically on your web page.
- Implement error handling to manage failed API requests gracefully.
- Utilize at least 2 public Web APIs

WebSockets

- Implement WebSocket functionality to enable real-time updates.
- Connect to a WebSocket server (set-up your own).
- Update the web page dynamically based on real-time data received through WebSockets.
- Implement proper event handling to manage Websocket events on both clients and servers.
- Remember that Websockets have a wide-range of application.

Dynamic Content

- Ensure that your website updates dynamically based on user interactions or real-time data.

Documentation

- Create a document showcasing the main objective, functionalities and features of your project.
- Program structure and code documentation should also be included,

Submission

- Push your project to a public GitHub repository.
- Include a README.md file that describes your project, its features, and how to set it up locally.
- Submit the GitHub repository link. (Submission Link will be given)

Evaluation Criteria

- **Functionality**: Does the web app meet all the functional requirements specified?
- **Code Quality**: Is the code well-organized, readable, and follows best practices?
- **UI/UX Design**: Is the user interface intuitive and visually appealing?
- **Responsiveness**: Does the website work well on different devices and screen sizes?
- **Error Handling**: Does the web app handle errors gracefully, especially with API requests and WebSocket connections?
- **Real-Time Updates**: Are real-time updates via WebSockets implemented effectively?
- **Deployment**: Is the web app deployed correctly and accessible online?

Tips

- Choose an API that provides clear and concise documentation.
- Test your WebSocket implementation thoroughly to ensure it handles connections and data updates smoothly.
- Plan your project structure before you start coding to ensure a clean and maintainable codebase.
- Use GitHub Issues or a similar tool to manage tasks and track progress.