Front-End Web Development II - Project Guidelines

In this project, you will create an **interactive web application** that uses **JavaScript**, **jQuery**, **AJAX**, and **external APIs** to fetch and display real-time data. Your application should provide dynamic content updates and a seamless user experience without requiring a full page reload.

Project Requirements:

1. **User Interaction**:
    - Use JavaScript to add interactive elements like **buttons**, **forms**, and **input fields** that respond to user actions.
2. **jQuery for Content Manipulation**:
    - Use **jQuery** for easier DOM manipulation (changing styles, adding/removing content).
    - Implement jQuery animations (e.g., **fade in/out**, **slide up/down**).
3. **API Integration**:
    - Integrate at least **one external API** to retrieve and display **real-time data**.
        - Examples: Weather API, Movie API, News API.
    - The data fetched from the API should be displayed dynamically on the page.
4. **AJAX for Data Fetching**:
    - Use **AJAX** (via **fetch()** or **jQuery's $.ajax()**) to **fetch data asynchronously**, ensuring the page doesn't need to reload to update the content.
    - Show a **loading indicator** (spinner or message) while the data is being fetched.
5. **Form Validation**:
    - Implement a form (e.g., feedback or contact form) and validate it before submission.

Steps for Completing the Project:

1. **Choose Your Project**:
    - Select the project ideas
    - Plan the features you want to include and sketch out the basic layout of your app.
2. **Set Up the Development Environment**:
    - Use **HTML**, **CSS**, **JavaScript**, and **jQuery** to create the app.
    - Create a simple folder structure for your project (e.g., index.html, styles.css, script.js).
3. **API Integration**:
    - Choose and integrate an API to fetch data for your app.
    - Use **AJAX** or **fetch()** to asynchronously retrieve data from the API.
    - Display the fetched data on the page dynamically.
4. **Develop Features**:
    - Add interactive elements like **forms** or **buttons**.
    - Use **jQuery** to manipulate the DOM and update content dynamically.
    - Add any necessary **animations** or **UI enhancements**.
5. **Testing**:
    - Test your app to ensure it works smoothly.
    - Ensure the **AJAX requests** return data correctly and that the content updates without page reloads.
    - Test **user interactions** like form submissions or searches.
6. **Deploy the App**:
    - Deploy your application using free hosting services like **GitHub.**

Deliverables:

1. **Project Files**:
    - Submit all the project files:
        - HTML files (e.g., index.html)
        - CSS files (e.g., style.css)
        - JavaScript files (e.g., app.js)
        - Any external libraries (e.g., jQuery)
        - Images or assets used in your app.

Project Report Structure

Your project report should follow the structure outlined below:

1. Title Page

- Project Title (Center alignment)
- Your Name
- Student ID
- Course Name

2. Table of Contents:

- Include all major sections and their corresponding page numbers for easy navigation.

3. Introduction:

- **Project Overview**:
  A brief explanation of the project's purpose, its main functionalities, and why you chose to build it.
- **Problem Statement**:
  Describe the problem the project is trying to solve or the value it provides. For example, "This project provides users with real-time weather information based on their location."
- **Technologies Used**:
  Provide a brief overview of the key technologies used to build the project (e.g., JavaScript, jQuery, AJAX, APIs, etc.).

4. Features and Functionality:

- **Main Features**:
  List and describe the core features of your application. State purpose and features of each functionality

5. Implementation Details:

Provide explanations for key parts of the code. You can include small code snippets and explain their functionalities.

6. Conclusion:

- Summarize the work completed, emphasizing the goals achieved and the project's outcome.
- Discuss any features you would add or improvements you would make if you had more time or resources (e.g., additional API features, UI improvements, better error handling).

7. References:

- List all references used in the development of your project. This may include:
    - **External APIs**: Links to documentation of any APIs integrated.
    - **Libraries and Frameworks**: (e.g., jQuery, Bootstrap, etc.).
    - **Tutorials and Guides**: Any online resources, blogs, or tutorials you consulted.
    - **Books**: Reference books or articles that were helpful.

Evaluation Criteria:

1. **Functionality (50%)**:
   - Does the app meet the requirements (e.g., user interaction, dynamic content updates)?
   - Does the app fetch data from an API and display it properly?
2. **Code Quality (20%)**:
   - Is the code well-organized, readable, and well-commented?
   - Does the code follow best practices for JavaScript, jQuery, and AJAX?
3. **User Experience (20%)**:
   - Is the app easy to use, with a clean and functional interface?
   - Are animations and interactions smooth and responsive?
4. **Deployment & Documentation (10%)**:
   - Is the app deployed and accessible online with a live demo?
   - Is the README clear and helpful?

Submission Instructions:

- **Final Submission**:
  - Submit the **project files** (code, images, etc.) through the course submission platform.

Please upload it to the course platform by the due date.

Formatting Guidelines

- **Font**: Times New Roman, 12 pt.
- **Spacing**: 1.5 or double line spacing.
- **Margins**: Standard 1-inch margins on all sides.
- **Page Numbers**: Include page numbers in the footer.
- **Headings**: Use consistent heading styles (e.g., Heading 1 for major sections, Heading 2 for subsections).