Week 1:

Create a simple static website and use JavaScript to make API requests + manage version control using Git.

**Problem Statement**

Design and develop a simple static website with multiple HTML pages linked together, ensuring a consistent layout using CSS for styling. Write a JavaScript program that fetches data from a public API (e.g., JSON Placeholder) and displays the results on a web page, using asynchronous JavaScript (Promises or async/await) to handle the API requests and responses, with error handling and user-friendly error messages. Incorporate basic HTML and CSS to create a user interface for displaying the fetched data. Utilize Git for version control to track changes.

**Website Project Documentation**

**Overview**

This project showcases a functional website built with HTML, CSS, and JavaScript, focusing on API integration and user interaction. It includes a multi-page structure to simulate real-world web applications, with seamless navigation between pages. The project primarily demonstrates fetching and displaying data from a dummy JSON API and includes user-centric features like login and contact forms.

**Pages and Functionalities**

**1. Login Page**

* **Purpose**: Acts as the entry point for users to access the website.
* **Features**:
  + Login form with input fields for username and password.
  + Validation ensures that both fields are filled before proceeding (basic front-end validation).
  + On successful login, redirects the user to the homepage.
  + Clean and user-friendly UI designed with CSS to provide a professional look.
* **Navigation**:
  + Redirects to the homepage upon logging in.
  + A logout button on other pages redirects back to this login page.

**2. Homepage**

* **Purpose**: Introduces the website's purpose and educates users about APIs.
* **Content**:
  + Explanation of **What is an API?**
    - Definition: APIs (Application Programming Interfaces) enable communication between software applications.
    - Example: Fetching user data from a server to display on a website.
  + **Project Workflow Description**:
    - Step-by-step explanation of how the API data is fetched and displayed on the site.
    - Highlighting the data categories fetched (users, recipes, and products).
    - Mentioning the absence of asynchronous handling (async/await) for educational clarity.

**3. API Working Page**

* **Purpose**: Demonstrates dynamic data fetching and rendering.
* **Details**:
  + Fetches data from a dummy JSON API endpoint.
  + Data categories:
    - **Users**: Names, email addresses, and other user-related information.
    - **Recipes**: Recipe titles, descriptions, and ingredients.
    - **Products**: Product names, prices, and descriptions.
  + Uses the fetch method to make API calls:
    - Simple JavaScript code fetches data synchronously (without async/await).
    - Data is dynamically displayed using HTML and JavaScript DOM manipulation.

**4. Contact Page**

* **Purpose**: Allows users to reach out for feedback or queries.
* **Features**:
  + A simple form with the following fields:
    - **Name**: Text input.
    - **Email**: Email input with basic validation for proper formatting.
    - **Message**: Textarea for longer user messages.
  + A **Submit button**:
    - Currently non-functional, as no backend integration exists.
    - Could be extended to send data to a server using a future backend implementation.
* **Design**:
  + Styled with CSS for a modern and responsive look.
  + Placeholder text in each field to guide user input.

**5. Logout Button**

* **Purpose**: Ensures users can safely exit the website.
* **Features**:
  + Redirects users back to the login page.

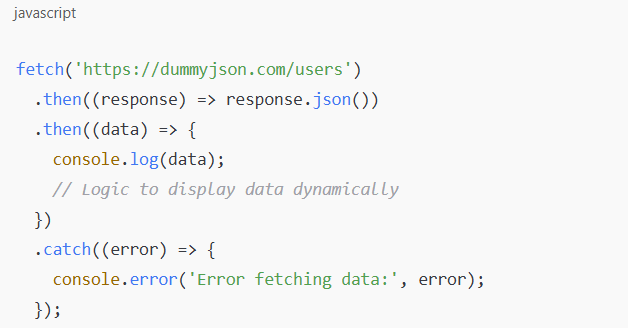
**Technical Details**

**Technologies Used**

* **HTML**: For structuring the pages.
* **CSS**: For designing and styling the website with responsiveness.
* **JavaScript**: For:
  + Fetching data from the dummy API.
  + Navigation logic and interactivity (e.g., login validation, data rendering).

**API Fetching Implementation**

* **Fetch Method**:
  + The fetch method is used to make API calls.
  + Data from the dummy JSON API includes users, recipes, and products.
  + Example code snippet:
  + No asynchronous syntax (async/await) is used, emphasizing basic promise chaining.



**Navigation Workflow**

* **Login Page** → **Homepage** → **API Page** → **Contact Page** → **Logout Page**.
* Navigation links are included on each page for easy access to other sections.

**Future Enhancements**

* **Asynchronous Handling**:
  + Implement async/await for more efficient and readable API calls.
* **Backend Integration**:
  + Add functionality to the Contact Page submit button to save or send data to a server.
* **Enhanced UI/UX**:
  + Add animations and improved responsiveness.
* **Security Features**:
  + Implement user authentication and session management.
* **Error Handling**:
  + Display user-friendly error messages for API call failures.

**Conclusion**

This project effectively demonstrates API integration and website navigation using fundamental web development technologies. Its modular approach and clear structure provide a solid foundation for further enhancements.