Day 4 coding assessment

Code:

index.html code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Blog & Todo Manager</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <header>
    <h1>📝 Blogging & Todo Platform</h1>
    <p>Using Fetch API + Async/Await + JS Module Pattern</p>
  </header>

  <div class="container">
    <div class="section" id="postsSection">
      <h2>📄 Blog Posts</h2>
      <button id="fetchPostsBtn">Fetch Posts</button>
      <div id="postsContainer"></div>
      <p id="postError" class="error"></p>
    </div>

    <div class="section" id="todosSection">
      <h2>☑️ Todo List</h2>
      <button id="fetchTodosBtn">Fetch Todos</button>
      <div id="todosContainer"></div>
      <p id="todoError" class="error"></p>
    </div>
  </div>

  <script src="script.js"></script>
</body>
</html>
```

style.css code:

```
body {
  font-family: "Segoe UI", Arial, sans-serif;
  background: #f9fafc;
  margin: 0;
```

```
  padding: 20px;
}

header {
  background-color: #a7ceee;
  color: rgb(30, 26, 26);
  padding: 15px;
  border-radius: 8px;
  text-align: center;
}

.container {
  display: flex;
  justify-content: space-around;
  flex-wrap: wrap;
  margin-top: 20px;
}

.section {
  background-color: #f1f0f1;
  border: 1px solid #787474;
  border-radius: 8px;
  padding: 20px;
  width: 45%;
  box-shadow: 0 2px 6px rgba(0,0,0,0.1);
  margin-bottom: 20px;
}

h2 {
  border-bottom: 2px solid #264259;
  padding-bottom: 5px;
}

.post, .todo {
  background-color: #ebeff4;
  border: 1px solid #33373c;
  border-radius: 5px;
  padding: 10px;
  margin: 10px 0;
}

.todo.completed {
  background-color: #d1ddd3;
  text-decoration: line-through;
}

button {
  background-color: #1b1f22;
```

```css
  color: white;
  border: none;
  padding: 10px 15px;
  border-radius: 5px;
  cursor: pointer;
  margin-top: 10px;
}

button:hover {
  background-color: #272eb7;
}

.error {
  color: rgb(214, 17, 17);
  margin-top: 10px;
  font-weight: bold;
}

.loading {
  color: #685c5c;
  font-style: italic;
}

@media (max-width: 800px) {
  .section {
    width: 90%;
  }
}
```

script.js code:

```js
// 🧩 Using the Module Pattern to encapsulate logic and avoid global
pollution
const AppModule = (() => {

  // API Endpoints
  const POSTS_API = "https://jsonplaceholder.typicode.com/posts";
  const TODOS_API = "https://jsonplaceholder.typicode.com/todos";

  // DOM Elements
  const postsContainer = document.getElementById("postsContainer");
  const todosContainer = document.getElementById("todosContainer");
  const postError = document.getElementById("postError");
  const todoError = document.getElementById("todoError");

  // ---- Utility: Fetch Wrapper with Error Handling ----
  async function fetchData(url) {
    try {
```

```
      const response = await fetch(url);
      if (!response.ok) throw new Error(`Error ${response.status}: Failed to
fetch data`);
      return await response.json();
    } catch (err) {
      console.error("Fetch Error:", err);
      throw err;
    }
  }

  // ---- UI Rendering Functions ----
  function renderPosts(posts) {
    postsContainer.innerHTML = "";
    posts.slice(0, 10).forEach(post => {
      const div = document.createElement("div");
      div.classList.add("post");
      div.innerHTML = `
        <h3>${post.title}</h3>
        <p>${post.body}</p>
      `;
      postsContainer.appendChild(div);
    });
  }

  function renderTodos(todos) {
    todosContainer.innerHTML = "";
    todos.slice(0, 10).forEach(todo => {
      const div = document.createElement("div");
      div.classList.add("todo");
      if (todo.completed) div.classList.add("completed");
      div.textContent = todo.title;
      todosContainer.appendChild(div);
    });
  }

  // ---- Event Handlers ----
  async function handleFetchPosts() {
    postsContainer.innerHTML = `<p class="loading">Loading posts...</p>`;
    postError.textContent = "";
    try {
      const posts = await fetchData(POSTS_API);
      renderPosts(posts);
    } catch (error) {
      postError.textContent = "Failed to load posts. Please try again later.";
      postsContainer.innerHTML = "";
    }
  }
```

```
  async function handleFetchTodos() {
    todosContainer.innerHTML = `<p class="loading">Loading todos...</p>`;
    todoError.textContent = "";
    try {
      const todos = await fetchData(TODOS_API);
      renderTodos(todos);
    } catch (error) {
      todoError.textContent = "Failed to load todos. Please try again later.";
      todosContainer.innerHTML = "";
    }
  }

  // ---- Public Methods (Revealing Module Pattern) ----
  return {
    init: () => {
      document.getElementById("fetchPostsBtn").addEventListener("click",
handleFetchPosts);
      document.getElementById("fetchTodosBtn").addEventListener("click",
handleFetchTodos);
    }
  };
})();

// Initialize the app when DOM is ready
document.addEventListener("DOMContentLoaded", AppModule.init);
```

Documentation:

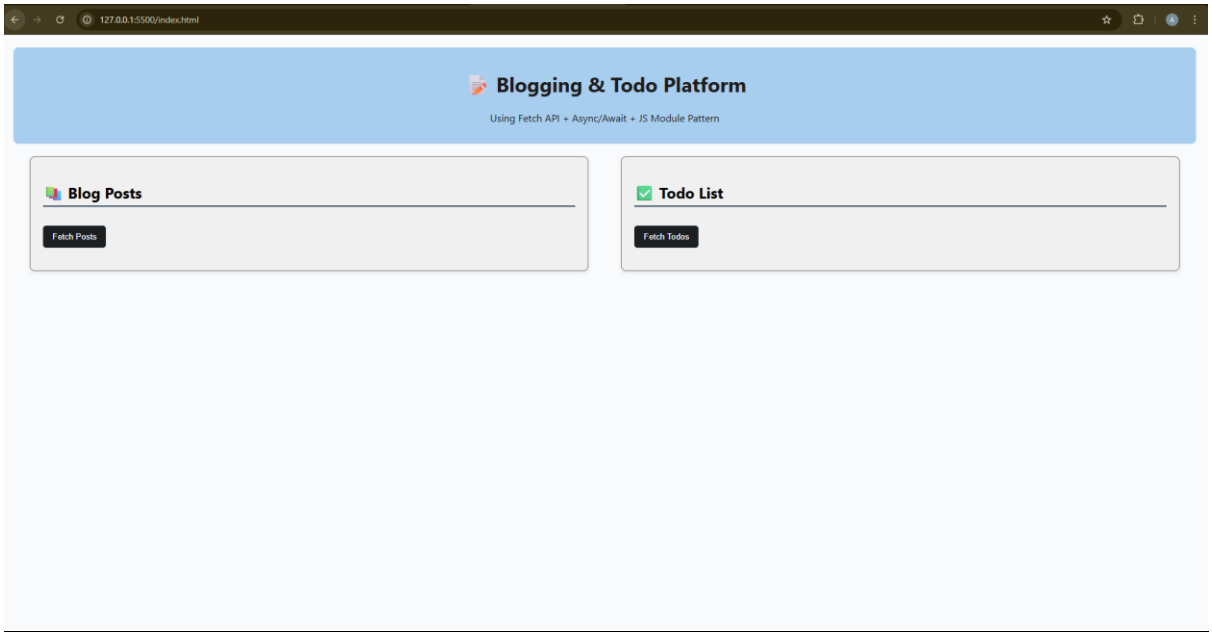
For this assignment, my approach was to design a simple web application using the Fetch API with async/await to retrieve data from a public API and display it dynamically. I divided the interface into two sections — one for blog posts and another for todos — and structured the JavaScript using the module pattern to keep the code organized and prevent global variable conflicts.

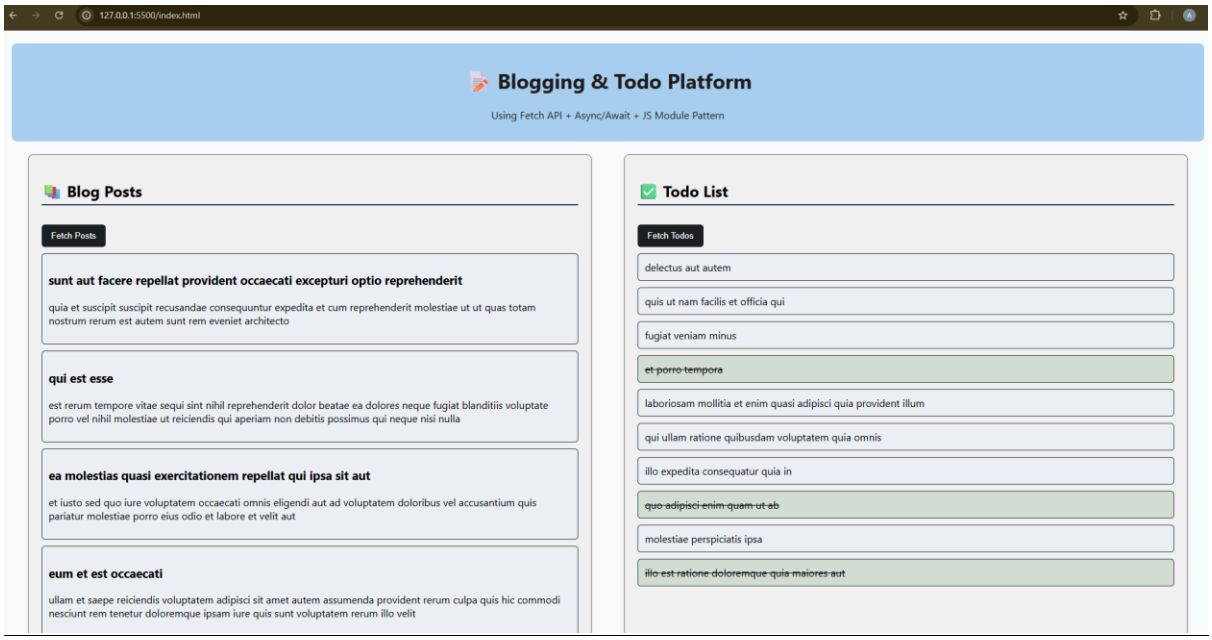
During implementation, I focused on keeping the code modular and readable, adding comments to explain each function's purpose and maintaining consistent naming conventions. The main challenge I faced was handling API errors and loading states, which I solved by using proper try...catch blocks and conditional rendering in the UI.

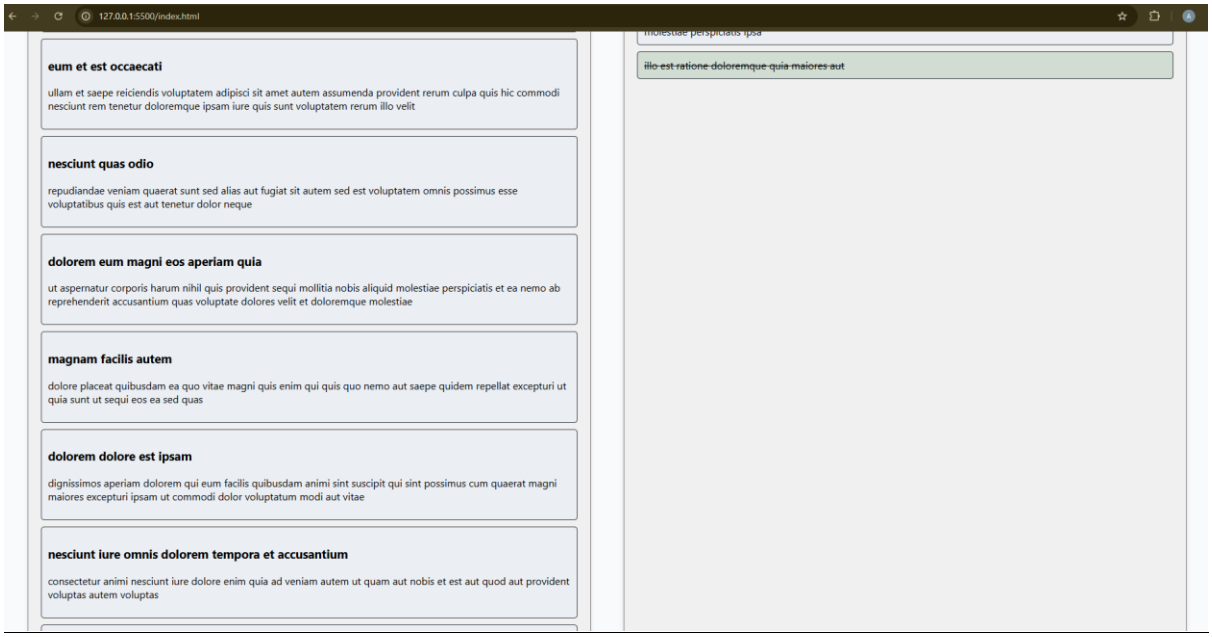
I submitted the completed assignment within the given time frame, ensuring all files (HTML, CSS, JS) were properly linked and functional.

Overall, this task helped me strengthen my understanding of asynchronous JavaScript, API handling, and modular design patterns. I also learned the importance of clear structure, error handling, and documentation in building maintainable web applications.

Screenshots:






127.0.0.1:5500/index.html
eum et est occaecati
ullam et saepe reiciendis voluptatem adipisci sit amet autem assumenda provident rerum culpa quis hic commodi nesciunt rem tenetur doloremque ipsam iure quis sunt voluptatem rerum illo velit
nesciunt quas odio
repudiandae veniam quaerat sunt sed alias aut fugiat sit autem sed est voluptatem omnis possimus esse voluptatibus quis est aut tenetur dolor neque
dolorem eum magni eos aperiam quia
ut aspernatur corporis harum nihil quis provident sequi mollitia nobis aliquid molestiae perspiciatis et ea nemo ab reprehenderit accusantium quas voluptate dolores velit et doloremque molestiae
magnam facilis autem
dolore placeat quibusdam ea quo vitae magni quis enim qui quis quo nemo aut saepe quidem repellat excepturi ut quia sunt ut sequi eos ea sed quas
dolorem dolore est ipsam
dignissimos aperiam dolorem qui eum facilis quibusdam animi sint suscipit qui sint possimus cum quaerat magni maiores excepturi ipsam ut commodi dolor voluptatum modi aut vitae
nesciunt iure omnis dolorem tempora et accusantium
consectetur animi nesciunt iure dolore enim quia ad veniam autem ut quam aut nobis et est aut quod aut provident voluptas autem voluptas
illo est ratione doloremque quia maiores aut