Assignment 2

Convert static site to dynamic site.

Problem Statement: Dynamic Task Management Website

Context:
Convert static website created in Assignment 1 to dynamic site using plain javascript

Requirements:
The website must have the following features:

1. **Home Page**
    - Display a list of tasks in a table format.
    - The table should include the following columns:
        - Task ID
        - Task Description
        - Assigned To
        - Due Date
    - Data should not be hardcoded in the HTML but stored in a JavaScript object and dynamically rendered into the table.
    - The `Task ID` column should be clickable and function as a hyperlink.
2. **Task Detail Page**
    - Clicking on a `Task ID` in the table should navigate to a **Task Detail Page**.
    - The page should show details of the selected task in a form with the following fields:
        - Task ID (read-only)
        - Task Description
        - Assigned To
        - Due Date
    - Task details should be populated dynamically based on the clicked `Task ID`. Pass the `Task ID` as a query string parameter and use JavaScript to retrieve the corresponding data.
3. **Additional Guidelines**
    - Use **plain HTML, CSS, and JavaScript** (no libraries or frameworks like React or Angular).
    - Organize your code into multiple files:
        - `index.html`: The home page.
        - `tr_detail.html`: The task detail page.
        - `data.js`: JavaScript file to store task data.
        - `script.js` and `detail.js`: JavaScript files to handle dynamic content on respective pages.
        - `styles.css`: CSS file for styling the website.
    - Use the **Live Server** extension in Visual Studio Code to test your solution.

Example Data:
Use the following example tasks to populate the JavaScript object:

```
const tasks = [
    { id: 1, description: "Fix Navbar", assignedTo: "John Doe", dueDate: "2024-11-20" },
    { id: 2, description: "Update Table", assignedTo: "Jane Smith", dueDate: "2024-11-25" },
    { id: 3, description: "Test Functionality", assignedTo: "Alice Brown", dueDate: "2024-11-30" },
];
```

Deliverables:

1. A fully functional dynamic website that meets the requirements.
2. Clean and well-commented HTML, CSS, and JavaScript code.
3. Screenshots or a video demonstrating:
    - The task list displayed on the home page.
    - Navigation to the task detail page by clicking a `Task ID`.
    - Correct data being displayed in the detail page form.

Evaluation Criteria:

- Correct implementation of dynamic table and form population using JavaScript.
- Code readability and maintainability.
- Proper navigation between pages and handling of query string parameters.
- Styling of the website for readability and usability.

Solution

Part 1: Modify `index.html` to Use JavaScript for Table Data

1. **Store Task Data in a JavaScript Object**
    - Create a `data.js` file to store task data in a JSON-like structure.

```
// data.js
const tasks = [
    { id: 1, description: "Fix Navbar", assignedTo: "John Doe", dueDate: "2024-11-20" },
    { id: 2, description: "Update Table", assignedTo: "Jane Smith", dueDate: "2024-11-25" },
    { id: 3, description: "Test Functionality", assignedTo: "Alice Brown", dueDate: "2024-11-30" },
];
```

2. **Modify `index.html` to Include JavaScript**

- Remove the static table rows in `index.html` and include `data.js` and a custom script file, `script.js`.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="styles.css">
    <title>Home</title>
</head>
<body>
    <header>
        <nav>
            <a href="index.html">Home</a> |
            <a href="tr_detail.html">TR Detail</a>
        </nav>
    </header>
    <main>
        <h1>Task List</h1>
        <table>
            <thead>
                <tr>
                    <th>Task ID</th>
                    <th>Task Description</th>
                    <th>Assigned To</th>
                    <th>Due Date</th>
                </tr>
            </thead>
            <tbody id="taskTableBody">
                <!-- Rows will be populated dynamically -->
            </tbody>
        </table>
    </main>
    <script src="data.js"></script>
    <script src="script.js"></script>
</body>
</html>
```

3. **Create `script.js` to Populate the Table**

- Use JavaScript to dynamically generate the table rows based on the `tasks` array.

```
const taskTableBody = document.getElementById('taskTableBody');

tasks.forEach(task => {
    const row = document.createElement('tr');

    row.innerHTML = `
        <td><a href="tr_detail.html?taskId=${task.id}">${task.id}</a></td>
```

```
            <td>${task.description}</td>
            <td>${task.assignedTo}</td>
            <td>${task.dueDate}</td>
        `;

        taskTableBody.appendChild(row);
    });
```

Part 2: Modify `tr_detail.html` to Populate Form Fields Dynamically

1. **Update `tr_detail.html` to Include JavaScript**
    - Include `data.js` and a new script for the detail page, `detail.js`.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="styles.css">
    <title>Task Detail</title>
</head>
<body>
    <header>
        <nav>
            <a href="index.html">Home</a> |
            <a href="tr_detail.html">TR Detail</a>
        </nav>
    </header>
    <main>
        <h1>Task Detail</h1>
        <form>
            <label for="taskId">Task ID:</label>
            <input type="text" id="taskId" readonly><br>
            <label for="description">Description:</label>
            <input type="text" id="description"><br>
            <label for="assignedTo">Assigned To:</label>
            <input type="text" id="assignedTo"><br>
            <label for="dueDate">Due Date:</label>
            <input type="date" id="dueDate"><br>
        </form>
    </main>
    <script src="data.js"></script>
    <script src="detail.js"></script>
</body>
</html>
```

2. **Create `detail.js` to Populate the Form**

- Parse the query string to get the `taskId` and use it to find the task in the `tasks` array.

```
// Parse query string to get the task ID
const urlParams = new URLSearchParams(window.location.search);
const taskId = urlParams.get('taskId');

// Find the task details
const task = tasks.find(t => t.id == taskId);

// Populate the form fields if the task is found
if (task) {
    document.getElementById('taskId').value = task.id;
    document.getElementById('description').value = task.description;
    document.getElementById('assignedTo').value = task.assignedTo;
    document.getElementById('dueDate').value = task.dueDate;
} else {
    alert("Task not found.");
}
```

Testing the Dynamic Site

1. **Launch the Site**
   - Open `index.html` with the "Live Server" extension in VS Code.
2. **Test the Functionality**
   - Verify that the `Task ID` column in the table is a hyperlink.
   - Clicking a `Task ID` should navigate to `tr_detail.html` with the corresponding task ID in the query string.
   - The task details page should display the correct task information in the form.
