FSDI 106 JavaScript and jQuery Solutions Competency Report

Name: Sequoyah Dozier

Project: Task Manager - Web Application

1. Project Overview

The **Task Manager Web Application** is designed to allow users to create, display, and manage tasks efficiently. The application provides a structured approach to task management by incorporating input validation, dynamic UI updates, and API interactions for data persistence.

This project demonstrates my proficiency in **JavaScript, jQuery, HTML, CSS, AJAX, API Integration, and Git** while also improving my ability to debug issues and optimize functionality.

2. Technologies Used

- **HTML5** – For structuring the web page and form elements.
- **CSS3 & Bootstrap** – For styling, layout, and responsive design.
- **JavaScript & jQuery** – For handling event-driven interactions and AJAX requests.
- **AJAX & API Integration** – For interacting with a remote server to store and retrieve tasks.
- **Git & GitHub** – For version control and project management.

3. Features Implemented

3.1 Task Registration Form

Users can enter the following task details:

- **Title**
- **Description**
- **Color (for task categorization)**

    - **Start Date**
    - **Status (New, In Progress, Completed, Canceled)**
    - **Budget**

✅ **Validation:** Ensures that all required fields are filled before allowing submission.

✅ **Dynamic UI Updates:** Newly created tasks appear instantly in the task list.

3.2 Task Display (List View)

- Tasks are displayed dynamically in a well-structured container.
- The task list shows:
    - **Title**
    - **Description**
    - **Start Date**
    - **Status**
    - **Budget**
- Tasks are visually differentiated using color indicators.

3.3 Task Persistence Using an API

- **Tasks are stored on a remote server** using an HTTP POST request.
- **Tasks are retrieved and displayed** when the page loads using an HTTP GET request.

3.4 Task Deletion

- The application includes a **“Clear All Tasks”** button to remove all tasks from the server and UI.
- The deletion request is sent via an HTTP DELETE request.

3.5 User Interface Enhancements

- **Responsive Design:** The UI is optimized for different screen sizes.
- **Improved Styling:** Task cards have structured layouts and color-coded labels for better readability.

4. Challenges & Solutions

During development, I encountered several issues that required debugging and improvements. Here's a breakdown of the problems and how I solved them:

🔹 4.1 Tasks Were Not Being Added to the List After Clicking "Add Task"

❌ **Issue:** When clicking the "Add Task" button, the task was not appearing in the task list.
✅ **Solution:**

- Implemented the displayTask() function correctly to append tasks dynamically.
- Ensured the saveTask() function executed **before** making the AJAX request to store the task.
- Debugged the function calls using console.log() to verify task objects were created properly.

🔹 4.2 API Requests Not Persisting Tasks

❌ **Issue:** Tasks were not being stored on the server correctly, leading to data loss on page refresh.
✅ **Solution:**

- Fixed an issue in the **AJAX POST request** where the request payload was incorrectly formatted.
- Verified the server response using console.log(response) to confirm successful storage.
- Implemented a loadTask() function to fetch stored tasks on page load.

🔹 4.3 Tasks Not Loading on Page Refresh

❌ **Issue:** Even after saving tasks, they were not appearing after a page refresh.
✅ **Solution:**

- Implemented an **AJAX GET request** in the loadTask() function to fetch and display all tasks.
- Ensured only **tasks belonging to the user** were displayed by checking the task.name field.

🔹 4.4 Task Budget Was Not Displaying Properly

❌ **Issue:** The budget amount was displaying as undefined or with too many decimal places.
✅ **Solution:**

- Used parseFloat(task.budget).toFixed(2) to ensure proper numeric formatting.
- Ensured budget input values were correctly parsed and stored.

🔹 4.5 The "Clear All Tasks" Button Was Not Deleting Tasks from Server

❌ **Issue:** Clicking the "Clear All Tasks" button removed tasks from the UI but not from the server.
✅ **Solution:**

- Implemented a **DELETE** request to remove all tasks from the API.
- Ensured the UI cleared successfully after receiving a response from the server.

🔹 4.6 Task Start Date Was Not Being Captured Correctly

❌ **Issue:** The startDate field was coming in as undefined.
✅ **Solution:**

- Fixed a typo in the constructor function (startDate vs startdate).
- Verified that the correct **input field ID** was used in the JavaScript function.

🔹 4.7 jQuery Event Listeners Not Triggering

❌ **Issue:** The event handlers for the **Save** and **Clear** buttons were not working.
✅ **Solution:**

- Used **$(document).ready(function() {...})** to ensure the DOM was fully loaded before attaching event listeners.
- Verified button IDs were correctly referenced in $("#btnSave").click(saveTask);.

5. Final Result

After implementing and debugging the application, the **Task Manager Web System now successfully:**
✅ **Allows users to add new tasks.**
✅ **Saves tasks to the server using AJAX POST requests.**
✅ **Fetches and displays tasks dynamically on page load.**
✅ **Clears all tasks when requested via a DELETE API call.**
✅ **Ensures proper validation, formatting, and UI responsiveness.**

6. GitHub Repository

🔗 **Repository:** FSDI 105 - Competency Report
🔗 **Branch:** main

7. Next Steps

Although the project meets the competency requirements, I plan to **enhance the system further** with additional features:

1. **Edit Feature:** Allow users to edit task details instead of just adding/deleting tasks.
2. **Advanced Notifications:** Improve user experience with success/error notifications.
3. **Drag-and-Drop Sorting:** Let users reorder tasks visually.
4. **Task Categories & Filtering:** Enable filtering tasks by status (New, In Progress, Completed).
5. **Improved UI Design:** Refine the styling for better aesthetics and readability.

8. Conclusion

This project significantly strengthened my skills in:
✅ **JavaScript & jQuery** – Implementing dynamic UI updates and event handling.
✅ **AJAX & API Integration** – Making HTTP requests for data storage and retrieval.
✅ **Debugging** – Solving issues with event handling, API requests, and data persistence.
✅ **Version Control (Git/GitHub)** – Managing my project workflow efficiently.

This experience has given me a deeper understanding of **full-stack web development**, and I look forward to building upon this foundation in future projects.

✅ **Final Verdict:**

🚀 My Task Manager Web System meets all competency requirements and demonstrates mastery of the course skills. 🚀