1.Project name : To-Do List

2.Project name : Personal Budget Tracker

1.Project name : To-Do List

https://to-do-list-topaz-omega-47.vercel.app/

Project Title: To-Do List Application

Overview: This is a **To-Do List Web Application** that allows users to easily manage their daily tasks. The app provides an intuitive interface for users to add, edit, and delete tasks, as well as track their progress by marking tasks as completed. It's a simple yet powerful tool to help individuals stay organized and improve productivity.

Technologies Used:

- **Frontend**: Built using **HTML**, **CSS**, and **JavaScript** in **Visual Studio Code**. This was chosen for its efficiency in developing clean and responsive web interfaces.
- **Deployment**: Hosted using **Vercel**, which provides fast and reliable deployment for static websites.

Key Features:

1. **Task Creation**: Users can add new tasks easily with descriptions, and the task list updates instantly.
2. **Task Management**: Each task can be marked as **Complete** or **Pending**, providing clear progress tracking.
3. **User-Friendly Interface**: The app has a simple, responsive design that adapts well to different devices, ensuring ease of use across both mobile and desktop.
4. **Dynamic Task List**: Tasks are displayed dynamically as they are added, edited, or completed, creating a seamless user experience.

Project Development:

- **Development Environment**: Developed using **Visual Studio Code**, which provided a powerful and efficient coding environment for HTML, CSS, and JavaScript.
- **Features**: While the app is lightweight and focuses on task management, it can be expanded further with additional features such as categories or task deadlines in the future.

Challenges and Learning: This project allowed me to improve my front-end development skills, especially in designing responsive layouts and interactive features using JavaScript. I also learned how to streamline the development process using **Vercel** for deployment.

Test Case 1: Adding a Task

- **Objective**: Ensure that the user can add a new task to the list.
- **Steps**:
    1. Open the To-Do List application.
    2. Enter a task description in the input field (e.g., "Buy groceries").
    3. Click the "Add Task" button.
- **Expected Result**: The new task appears in the task list.
- **Pass/Fail**: Pass if the task is displayed in the list.

Test Case 2: Marking a Task as Complete

- **Objective**: Ensure that users can mark a task as complete.
- **Steps**:
    1. Add a new task (e.g., "Finish homework").
    2. Click the checkbox next to the task to mark it as complete.
- **Expected Result**: The task's appearance changes (e.g., it may be crossed out or highlighted as "Completed").
- **Pass/Fail**: Pass if the task is marked as completed.

Test Case 3: Editing a Task

- **Objective**: Ensure that users can edit an existing task.
- **Steps**:
    1. Add a new task (e.g., "Prepare dinner").
    2. Click the "Edit" button (if available).
    3. Modify the task description (e.g., change "Prepare dinner" to "Prepare dinner for guests").
    4. Save the changes.
- **Expected Result**: The updated task description is shown in the list.
- **Pass/Fail**: Pass if the task is edited correctly.

Test Case 4: Deleting a Task

- **Objective**: Ensure that users can delete a task.
- **Steps**:
    1. Add a new task (e.g., "Clean the house").
    2. Click the "Delete" button (if available) next to the task.
- **Expected Result**: The task is removed from the list.
- **Pass/Fail**: Pass if the task is deleted and no longer appears in the list.

Test Case 5: Task List Persistence (Page Refresh)

- **Objective**: Ensure that tasks persist after refreshing the page.
- **Steps**:
    1. Add a new task (e.g., "Write report").
    2. Refresh the page.
- **Expected Result**: The newly added task should still appear in the task list after the page reloads.
- **Pass/Fail**: Pass if the task persists after refreshing the page.

Test Case 6: Empty Task Input

- **Objective**: Ensure that the user cannot add an empty task.
- **Steps**:
    1. Try to add a new task with an empty description.
    2. Click the "Add Task" button.
- **Expected Result**: The task should not be added, and the app should provide a warning message (e.g., "Task description cannot be empty").
- **Pass/Fail**: Pass if the app prevents adding empty tasks.

Test Case 7: Responsive Layout (Mobile View)

- **Objective**: Ensure the app works correctly on mobile devices.
- **Steps**:
    1. Open the app in a browser and resize the window to a mobile viewport size or use a mobile device.
    2. Check the layout and functionality.
- **Expected Result**: The app should be fully functional, with all elements adjusting to the mobile screen (e.g., buttons, input fields).
- **Pass/Fail**: Pass if the app is fully responsive and usable on mobile.

Test Case 8: Task Sorting or Filtering (Optional)

- **Objective**: Ensure that tasks can be sorted or filtered (if this feature is implemented).
- **Steps**:
    1. Add multiple tasks (e.g., "Buy groceries", "Finish homework").
    2. Sort or filter the tasks (e.g., by completion status).

- **Expected Result**: The tasks should appear in the correct order based on the selected criteria (e.g., completed tasks at the bottom).
- **Pass/Fail**: Pass if sorting or filtering works as expected.

Test Case 9: Visual Indication of Task Status (Completed vs. Pending)

- **Objective**: Ensure the completed tasks are visually distinct from pending tasks.
- **Steps**:
    1. Add a task (e.g., "Write email").
    2. Mark the task as completed.
- **Expected Result**: The task should visually change (e.g., a line through it or a color change) to indicate that it is completed.
- **Pass/Fail**: Pass if the completed task is visually distinct from the pending task.

https://to-do-list-topaz-omega-47.vercel.app/

2.Project name : Personal Budget Tracker

Project Title: Personal Budget Tracker

Overview: This project is a **Personal Budget Tracker** web application, designed to simplify personal finance management by allowing users to record income and expenses. The goal was to create a straightforward and accessible tool that helps users monitor their spending patterns and manage their budget.

Project Details:

- **Development Environment**: The project was built in **Visual Studio Code**, leveraging its powerful tools and extensions to streamline development.
- **Data Storage**: Data is stored in **Excel** files, which are used to log entries for income and expenses.
- **Assistance**: I developed this project with guidance from ChatGPT, making it a collaborative learning experience and allowing me to tackle challenges in real-time.

Key Features:

1. **Income and Expense Entry**: Users can log details of their income and various expenses manually, making it easy to keep track of all transactions.
2. **Categorization**: Each expense can be assigned a category (like food, transportation, entertainment) to help users see where they are spending the most.
3. **Summary Dashboard**: The app displays a summary of expenses and income to give users a clear financial overview.
4. **Excel Integration**: All data entries are stored in an Excel file, making it easy for users to export, analyze, or share data.
5. **Responsive Design**: The interface is designed to be user-friendly and accessible on both desktop and mobile devices.

Technologies Used:

- **Frontend**: HTML, CSS, and JavaScript for structuring, styling, and functionality.
- **Backend**: The project doesn't utilize a traditional backend database; instead, it's designed to store data locally in an Excel sheet.

For quality assurance, I wrote and executed several **manual test cases**.

- **Smoke Testing**: To ensure the core features were functioning as expected.
- **Functionality Testing**: To verify that all functionalities were working in line with the requirements.

- **Integration Testing**: To test interactions between different parts of the system.
- **System Testing**: To ensure the entire system worked as intended.
- **Adhoc Testing**: To identify potential issues that weren't caught by predefined test cases.
- **Regression Testing**: To ensure new updates didn't affect the existing functionalities.

Challenges and Learning: Working on this project allowed me to gain experience in handling data integration and in building responsive web applications. The guidance from ChatGPT helped me to understand concepts quickly and apply them effectively in my code.

https://personal-budget-tracker-one.vercel.app/