Project Title: To – Do list Application

By

R. Sakthi Manikandan

To

Unified Mentors.

Duration: 3 Months.

Date: 15-05-2025 to 15-08-2025.

Contents

Abstract

The To-Do List App is a simple yet effective productivity tool developed using HTML, Tailwind CSS, and JavaScript. The main goal of this project is to help users manage their daily tasks in an organized and efficient way. Users can easily add, edit, and delete tasks, and can also mark them as completed using a toggle button. The interface is clean, responsive, and built using Tailwind CSS to ensure a consistent user experience across devices.

This project focuses heavily on core JavaScript concepts like DOM manipulation, array handling, and dynamic updates. Each task is stored temporarily in memory, and the list updates in real time based on user actions. It helped me gain hands-on experience in building interactive web pages and improved my understanding of front-end logic. Overall, this project demonstrates how even a small tool can improve productivity and how basic web technologies can be combined to create something practical and user-friendly.

Objective

- To build a simple and user-friendly task management tool using HTML, Tailwind CSS, and JavaScript.
- To help users organize their daily tasks by allowing them to add, edit, delete, and mark tasks as completed.
- To practice and demonstrate core JavaScript skills such as DOM manipulation and dynamic event handling.
- To create a responsive UI that works smoothly across different devices using Tailwind CSS.
- To improve productivity by offering a digital solution for managing to-do lists efficiently.
- To understand the logic behind updating and displaying user-generated data in real time.

Technology Stack

3.1. HTM

- HTML is the standard language of the web, ensuring that your web pages are displayed consistently across all major browsers like Chrome, Firefox, Safari, and Opera.
- It works in conjunction with CSS (for styling) and JavaScript (for interactivity) to create visually appealing and dynamic web experiences.
- HTML uses various tags to structure and organize content logically, making it easier to read and understand for both human users and search engines.
- Semantic HTML tags like <header>, <nav>, <main>, <article>, <section>, and <footer> provide clear meaning and context to different parts of your webpage, improving readability, accessibility, and SEO.
- HTML integrates seamlessly with other web technologies like CSS for styling, JavaScript for interactivity, and backend languages like PHP, Python, and Ruby for dynamically generated content.

3.2. CSS

- CSS is the language we use to style an HTML document.
- CSS describes how HTML elements should be displayed.
- In this project, we have used the Css framework **Tailwind Css.**
- Faster UI building: Tailwind allows you to apply styles directly within HTML using predefined utility classes, eliminating the need to write custom CSS from scratch.
- Centralized configuration: The tailwind.config.js file allows you to define your design system (colors, spacing, typography, etc.) in one place, ensuring adherence to design guidelines.
- Reduced CSS file size: Tailwind's PurgeCSS integration automatically removes unused styles from your production build, resulting in a smaller CSS file and faster load times.

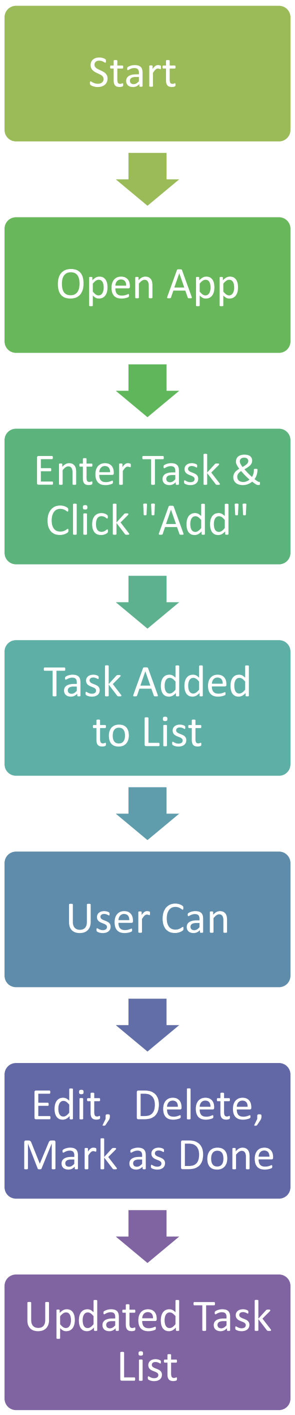
.

3.3. JAVASCRIPT

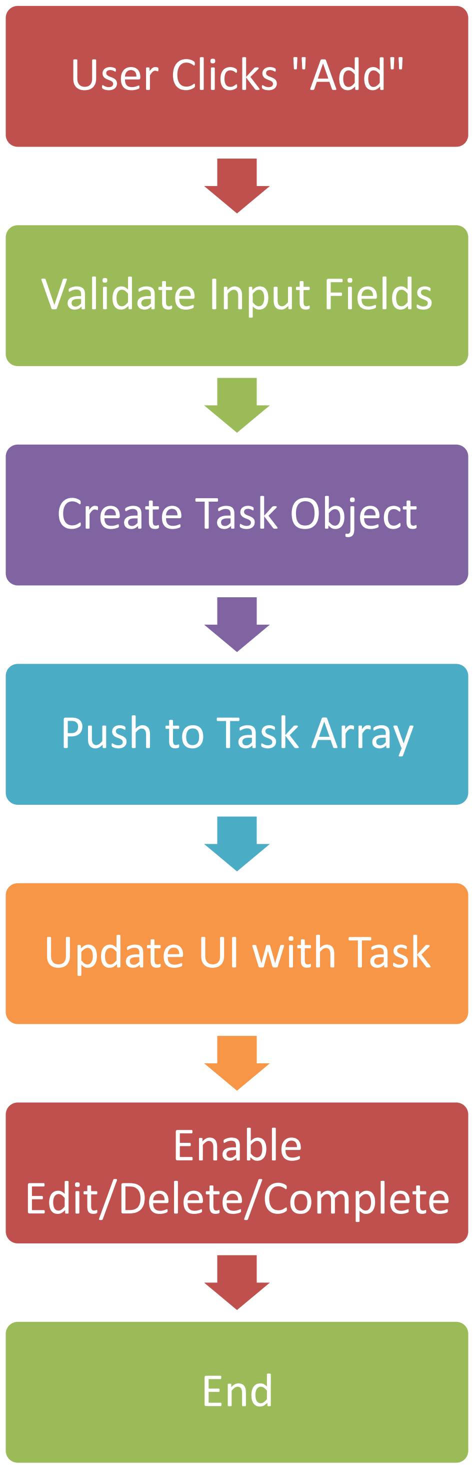
- JavaScript is a versatile and powerful programming language that forms a core part of web development alongside HTML and CSS. It's used to add interactivity, functionality, and dynamic behavior to web pages and applications.
- JavaScript breathes life into static HTML pages by allowing you to add dynamic content, animations, and interactive elements.
- JavaScript is primarily known for client-side (front-end) web development, where it runs directly in the user's browser, handling user interactions and manipulating the Document Object Model (DOM).
- JavaScript has a vast and vibrant ecosystem of libraries and frameworks like React, Angular, Vue.js, and Express.js.
- These tools simplify development, provide reusable components, and offer a structured approach to building complex web applications.
- JavaScript runs seamlessly across all major web browsers and operating systems, making it a reliable and versatile choice for web development
- Node.js – Server-side environment to handle real-time messaging.
- Express.js – Lightweight web framework used to serve HTML and manage routes.
- Socket.IO – Enables real-time, bi-directional communication between client and server.
- WebSockets – Underlying technology powering the chat feature.

4. Flow charts

4.1. User flow chart



4.2. System flow chart



5. Features Implemented

The To-Do List App is packed with essential and practical features that improve task management for users. The application was designed with simplicity and usability in mind. It focuses on reducing distractions while offering full control over task tracking.

- ❖ **Add New Tasks:** Users can easily add tasks by entering a description and clicking the "Add" button. Each task is added dynamically without refreshing the page.

- ❖ **Edit Tasks:** Users can update an existing task. When the "Edit" button is clicked, the task's content is prefilled into the input field, allowing the user to modify and resubmit.

- ❖ **Delete Tasks:** Any task can be removed from the list with a single click on the delete button.

- ❖ **Mark as Completed:** A checkbox or toggle allows users to mark tasks as completed. Completed tasks are visually distinguished (e.g., strikethrough or faded).

- ❖ **Responsive Design:** The layout is built using Tailwind CSS, ensuring the interface works smoothly across mobile, tablet, and desktop.

- ❖ **Clean UI:** The UI is intentionally minimal to keep the user focused on their tasks.

- ❖ **Dynamic Task Rendering:** Tasks are dynamically inserted into the DOM and updated live without page reloads.

- ❖ **Local Storage (optional):** To preserve tasks between sessions, the app can use local storage to save data in the browser.

6. Code Structure

The project's code is written using HTML, Tailwind CSS, and JavaScript. It follows a modular and clean structure for maintainability and scalability.

HTML Structure:

- The index.html file contains:
- A task input section
- Add button
- A task display section
- Optional filters or status indicators

HTML is minimal and semantic, with appropriate div, input, and button tags.

CSS (Tailwind):

- Instead of custom styles, Tailwind utility classes are used.
- Styling is applied directly to HTML elements for:
- Layout (e.g., flex, justify-between)
- Spacing (p-4, m-2)

- Colors (bg-blue-500, text-gray-700)
- Typography and borders.

JavaScript:

- Code is organized to separate logic into functions:
- **addTask():** Adds new task to an array and updates the DOM.
- **editTask(id):** Prefills input fields for editing.
- **deleteTask(id):** Removes the task from the array and re-renders.
- **toggleComplete(id):** Changes task completion status.
- **renderTasks():** Updates the task display on the UI.

- Event listeners are attached to buttons dynamically.
- Optionally uses localStorage.setItem() and getItem() to persist data.

This structure allows for clean readability and future extensions like login or due-date features.

3. Challenges Faced

During the development of the To-Do List App, several challenges were encountered, especially when balancing design simplicity with interactivity.

- **Dynamic DOM Manipulation:** Initially, updating the DOM with JavaScript every time a task was added or deleted caused glitches, especially when dealing with task IDs. This was solved by implementing a consistent rendering logic using a loop and an id property.

- **Edit vs. Add Confusion:** A common problem occurred when switching between editing and adding tasks. Without proper state management, tasks were getting duplicated. This was solved by adding logic to detect whether the task was being edited or newly added.

- **Maintaining Completion Status:** It was tricky to preserve the "completed" status across edits or deletions. The solution involved storing a completed: true/false property for each task object.

- **Layout on Small Screens:** Though Tailwind helped with responsiveness, fine-tuning spacing and alignment on smaller mobile screens required several UI adjustments.

- **Task Order & Persistence:** Sorting tasks by creation time or priority was not a built-in feature. Although not implemented in the first version, the code structure was built with future sorting in mind.

- **Local Storage Handling:** Reading from and writing to localStorage required proper parsing (JSON.parse and JSON.stringify), which caused issues when tasks were not properly formatted.

4. Conclusion

The To-Do List App was a foundational project that helped me solidify my understanding of core web technologies. It provided hands-on experience with dynamic JavaScript, responsive design with Tailwind CSS, and basic data handling logic. I now have a clear idea of how simple tools can be incredibly useful in real life. This app, though small in scope, opened up ideas for scaling into larger productivity apps with login, task categorization, reminders, and more.