**PROJECT REPORT**

**To-Do List Web Application**

**1. Title of the Project**

**Project Name:** To-Do List Web Application

**2. Introduction**

The **To-Do List Web Application** is a simple yet efficient tool that allows users to **add, manage, and track daily tasks**. Built using **HTML, CSS, and JavaScript**, this application helps users stay productive and organized. The interface is intuitive, making it easy for users to add tasks, mark them as completed, and delete them when needed.

**3. Objectives of the Project**

* **Develop a user-friendly to-do list** that helps users manage tasks efficiently.
* **Ensure smooth interactivity** using JavaScript for task management.
* **Implement responsive design** using CSS so the application works on multiple devices.
* **Provide functionality** for adding, deleting, and marking tasks as completed.

**4. Technologies Used**

* **HTML5** (Structuring the web page)
* **CSS3** (Styling and responsiveness)
* **JavaScript** (Handling interactivity)

**5. Project Design and Development**

**5.1 Project Workflow**

1. **User enters a task** in the input field and clicks "Add Task".
2. **The task appears in the list** with two buttons: **✔ (Mark as Complete)** and **✖ (Delete Task)**.
3. Clicking **✔** crosses out the task, indicating completion.
4. Clicking **✖** removes the task from the list.
5. The application ensures **responsive design**, allowing tasks to be managed on both **desktop and mobile**.

**5.2 HTML Structure**

The HTML file consists of:

* An input box for entering tasks.
* A button to add tasks.
* A list (<ul>) to display added tasks. Here is the code snippet:

html

<div class="container">

<h1>To-Do List</h1>

<input type="text" id="task-input" placeholder="Enter a task...">

<button id="add-task">Add Task</button>

<ul id="task-list"></ul>

</div>

**5.3 CSS Styling**

The **CSS** enhances the application's visual appeal by:

* Styling the **task list** with background colors.
* Adding **hover effects** to buttons.
* Making the application **responsive**. Here’s a snippet:

css

li {

padding: 10px;

background: #e0e0e0;

display: flex;

justify-content: space-between;

border-radius: 5px;

}

.completed {

text-decoration: line-through;

color: gray;

}

**5.4 JavaScript Functionality**

The JavaScript code is **event-driven** and listens for user actions:

* **Adding tasks** using addEventListener().
* **Deleting tasks** using remove().
* **Marking tasks as completed** using toggle(). Here’s a key snippet:

js

document.getElementById("add-task").addEventListener("click", () => {

const taskText = document.getElementById("task-input").value.trim();

if (taskText === "") return;

const taskItem = document.createElement("li");

taskItem.innerHTML = `${taskText} <button class="complete-task">✔</button> <button class="delete-task">✖</button>`;

document.getElementById("task-list").appendChild(taskItem);

taskItem.querySelector(".complete-task").addEventListener("click", () => {

taskItem.classList.toggle("completed");

});

taskItem.querySelector(".delete-task").addEventListener("click", () => {

taskItem.remove();

});

});

**6. Features Implemented**

**User Functionality**

✅ **Add Task** – Enter a task and click “Add Task.” ✅ **Mark as Completed** – Click ✔ to cross out tasks. ✅ **Delete Task** – Click ✖ to remove a task.

**Responsive Design**

* **Works across all screen sizes** (Desktop, Tablet, Mobile).
* **Adaptable layout** with flexible styling.

**Interactivity**

* **JavaScript enhances the experience** by detecting clicks and updating tasks dynamically.

**7. Testing and Debugging**

**Test Cases**

1. **Test task addition** → Ensure tasks display correctly.
2. **Test task deletion** → Verify tasks disappear from the list.
3. **Test marking completion** → Ensure tasks toggle between complete/incomplete.
4. **Responsiveness check** → Test functionality on mobile devices.

**Debugging Techniques**

* Used console.log() to **trace errors in JavaScript functions**.
* Ensured CSS selectors were properly applied.

**8. Future Enhancements**

**Possible upgrades to improve the project:**

* **Local Storage Integration** – Save tasks even after closing the browser.
* **Due Date Feature** – Add a deadline for tasks.
* **Sorting & Filtering** – Categorize tasks into priorities.

**9. Conclusion**

The To-Do List application successfully demonstrates **front-end development skills** in HTML, CSS, and JavaScript. It provides an **intuitive**, **interactive**, and **responsive** experience for users who need a simple task management system. Future improvements can be made to enhance usability and functionality.