Project 3: To-Do List

Web-Based To-Do List Application (HTML, CSS & JavaScript)

Student Name: Shravani

College: Thakur Polytechnic College, Kandivali (E)

Course: TY BCA

Submission Date: 29 October 2025

Objective

The primary objective of this project is to design and implement a simple, user-friendly web-based To-Do List application using standard front-end technologies — HTML, CSS, and JavaScript. The application should allow users to add, edit, delete, and mark tasks as completed, while demonstrating clear code structure, accessibility considerations, responsive layout, and persistent storage using the browser's localStorage. The project aims to showcase practical skills in DOM manipulation, event handling, and application state management in the browser environment.

Description

A To-Do List is a practical, lightweight application that helps users organize daily tasks, manage priorities, and track progress. This web-based implementation focuses on clean UI and straightforward functionality. Users can input task titles, mark tasks as completed, edit their content, and remove tasks. To retain data between sessions, tasks are stored in the browser's localStorage so that returning users find their task list preserved. The interface is intentionally minimal to reduce cognitive load: a text input field with a clear 'Add' button, a visually distinct list of tasks, and simple icons for editing and deleting. Completed tasks are styled differently (e.g., strikethrough) so users can see their accomplishments at a glance. From a development perspective, the project demonstrates separation of concerns: semantic HTML for structure, modular CSS for layout and visual design, and vanilla JavaScript for logic and events. The code is commented and organized for readability and maintainability, making it suitable as a learning resource or a foundation for extensions like categories, due dates, or synchronization with a backend. Overall, the project provides a compact yet complete example of a CRUD interface (Create, Read, Update, Delete) in the browser, reinforcing essential front-end development concepts.

Code Implementation (HTML, CSS & JavaScript)

index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>To-Do List</title>
  <link rel="stylesheet" href="styles.css" />
</head>
<body>
  <main class="container">
    <h1>To-Do List</h1>
    <section class="add-task">
      <input id="task-input" type="text" placeholder="Add a new task..." />
      <button id="add-btn">Add</button>
    </section>
    <ul id="task-list" class="task-list"></ul>
  </main>
  <script src="app.js"></script>
</body>
</html>
```

styles.css

```
/* styles.css */
* { box-sizing: border-box; }
body { font-family: Arial, sans-serif; margin: 0; padding: 20px; background: #f6f8fa; color:
```

```
.container { max-width: 700px; margin: 0 auto; background: #fff; padding: 20px; border-radiu
h1 { margin-top: 0; }
.add-task { display: flex; gap: 8px; margin-bottom: 12px; }
#task-input { flex: 1; padding: 10px; border: 1px solid #ddd; border-radius: 4px; }
#add-btn { padding: 10px 14px; border: none; cursor: pointer; border-radius: 4px; backgrounc
.task-list { list-style: none; padding: 0; margin: 0; }
.task-item { display: flex; align-items: center; justify-content: space-between; padding: 8p
.task-item .left { display:flex; align-items:center; gap:10px; }
.task-item.completed .title { text-decoration: line-through; color: #888; }
.task-actions button { margin-left: 6px; padding: 6px 8px; border-radius: 4px; border: none;
.task-actions .edit { background: #ffc107; color:#111; }
.task-actions .delete { background: #dc3545; color: #fff; }
```

app.js

```
// app.js
document.addEventListener('DOMContentLoaded', () => {
  const input = document.getElementById('task-input');
  const addBtn = document.getElementById('add-btn');
  const list = document.getElementById('task-list');
  const STORAGE_KEY = 'todo_tasks_v1';

  let tasks = loadTasks();

  renderTasks();

  addBtn.addEventListener('click', () => {
    const text = input.value.trim();
    if (!text) return;
    const task = { id: Date.now(), title: text, completed: false };
    tasks.push(task);
    saveTasks();
    renderTasks();
    input.value = '';
  });

  input.addEventListener('keydown', (e) => {
    if (e.key === 'Enter') addBtn.click();
  });

  function renderTasks() {
    list.innerHTML = '';
    tasks.forEach(task => {
      const li = document.createElement('li');
      li.className = 'task-item' + (task.completed ? ' completed' : '');
      li.dataset.id = task.id;

      const left = document.createElement('div');
      left.className = 'left';

      const checkbox = document.createElement('input');
      checkbox.type = 'checkbox';
      checkbox.checked = task.completed;
      checkbox.addEventListener('change', () => {
        task.completed = checkbox.checked;
        saveTasks();
        renderTasks();
      });
      const title = document.createElement('span');
```

```
      title.className = 'title';
      title.textContent = task.title;

      left.appendChild(checkbox);
      left.appendChild(title);

      const actions = document.createElement('div');
      actions.className = 'task-actions';

      const editBtn = document.createElement('button');
      editBtn.className = 'edit';
      editBtn.textContent = 'Edit';
      editBtn.addEventListener('click', () => {
        const newTitle = prompt('Edit task', task.title);
        if (newTitle === null) return;
        task.title = newTitle.trim();
        saveTasks();
        renderTasks();
      });

      const deleteBtn = document.createElement('button');
      deleteBtn.className = 'delete';
      deleteBtn.textContent = 'Delete';
      deleteBtn.addEventListener('click', () => {
        if (!confirm('Delete this task?')) return;
        tasks = tasks.filter(t => t.id !== task.id);
        saveTasks();
        renderTasks();
      });

      actions.appendChild(editBtn);
      actions.appendChild(deleteBtn);

      li.appendChild(left);
      li.appendChild(actions);

      list.appendChild(li);
    });
  }

  function saveTasks() {
    localStorage.setItem(STORAGE_KEY, JSON.stringify(tasks));
  }

  function loadTasks() {
    try {
      const raw = localStorage.getItem(STORAGE_KEY);
      return raw ? JSON.parse(raw) : [];
    } catch (e) {
      console.error(e);
      return [];
    }
  }
});
```

Output

When the application is opened in a web browser, the user sees a clean card-style interface containing an input field and an 'Add' button. Adding a task appends it to the list below with a checkbox, title, and action buttons for editing and deleting. Checking the box marks the task as completed and updates the visual style. Tasks persist between sessions via localStorage, so reloading the page or closing the browser retains the user's tasks. The interface is responsive and works on mobile and desktop; basic accessibility is supported by using semantic elements and allowing keyboard submission via Enter.

Conclusion

This To-Do List project demonstrates how a compact web application can provide essential task management features using vanilla HTML, CSS, and JavaScript. It reinforces key front-end concepts such as DOM manipulation, event handling, state persistence, and modest UI design. The codebase is intentionally small and readable, making it easy to extend—possible next steps include adding categories, due dates, search/filter functionality, and synchronization with a backend API for cross-device syncing.