A Basic To-Do Web App

📌 **Task ID :** OIBSIP_WebDevelopment2_Task3

👤 **Name :** N Akshitha

🎓 **Domain:** Web Development Internship

🗓️ **Date:** 14 July 2025

🎯 Objective:

Build a responsive, user-friendly To-Do list web app where users can add, edit, complete, and delete tasks with timestamps.

📋 Steps Performed:

1. Created HTML form and task tables for "Pending" and "Completed".

2. Styled UI using CSS (background image, flex layout, styled buttons).

3. Built JavaScript logic:

- `tasks` array to store task objects
- `renderTasks()` to refresh the UI
- Functions: add, delete, edit, complete tasks

4. Fixed container layout with `box-sizing` and `align-items: flex-start`.

5. Tested all functions: add/edit/delete/complete tasks with timestamps.

6. Prepared documentation and this PDF summary.

🛠️ Tools Used:

- HTML5 & CSS3
- JavaScript (Vanilla)
- VS Code (IDE)
- Chrome DevTools
- Git & GitHub

TO-DO WEBAPP

Html

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Professional To-Do App</title>
  <link rel="stylesheet" href="style2.css" />
</head>
<body>

  <h1>To-Do List Web App</h1>
  <div class="main">
    <div class="form-box">
      <form id="todoForm">
        <input type="text" id="title" placeholder="Title" required />
        <textarea id="description" placeholder="Description" required></textarea>
```

```
        <button type="submit">Save</button>
      </form>
    </div>

    <div class="task-list">
      <div class="section-title">📌 Pending Tasks</div>
      <table class="task-table" id="pendingTable">
        <thead>
          <tr>
            <th>Title</th>
            <th>Description</th>
            <th>Time</th>
            <th>Action</th>
          </tr>
        </thead>
        <tbody></tbody>
      </table>

      <div class="section-title">☑ Completed Tasks</div>
      <table class="task-table" id="completedTable">
        <thead>
          <tr>
            <th>Title</th>
            <th>Description</th>
            <th>Completed At</th>
            <th>Action</th>
          </tr>
        </thead>
        <tbody></tbody>
      </table>
    </div>
  </div>

  <script src="script2.js"></script>
</body>
</html>
```

CSS

```
body {
```

```
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 20px;
  background: url('https://images.unsplash.com/photo-1506748686214-e9df14d4d9d0')
no-repeat center center fixed;
  background-size: cover;
}

h1 {
  text-align: center;
  color: #333;
  margin-bottom: 30px;
  background: white;
  padding: 10px;
  border-radius: 10px;
}

.main {
  display: flex;
  gap: 30px;
  justify-content: center;
  align-items: flex-start; /* ☑ Add this line */
  flex-wrap: wrap;
}



.form-box {
  background: white;
  padding: 20px;
  border-radius: 12px;
  width: 300px;
  box-shadow: 0 4px 10px rgba(0,0,0,0.2);
}

.form-box input,
.form-box textarea {
  width: 100%;
  padding: 10px;
  margin: 10px 0;
  font-size: 14px;
}

.form-box button {
```

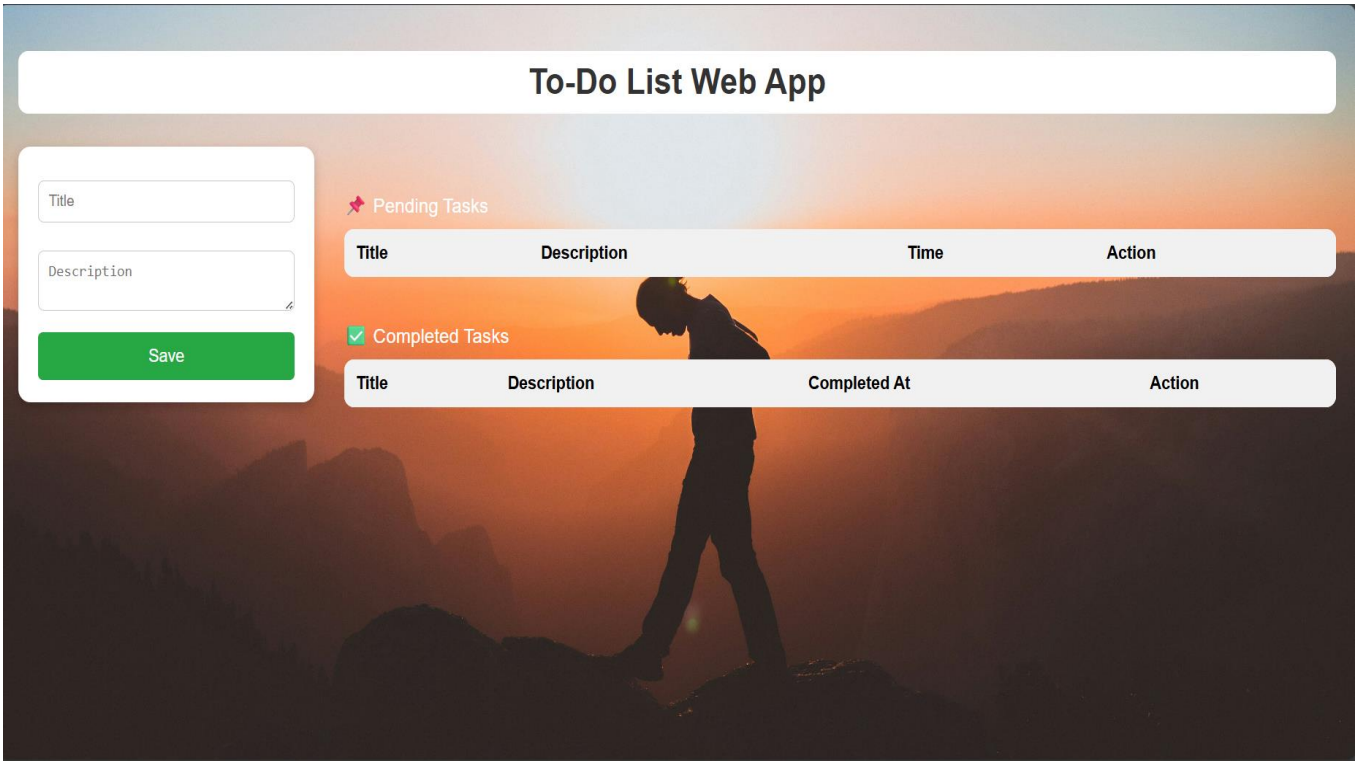
```
  background-color: #28a745;
  color: white;
  border: none;
  padding: 12px;
  width: 100%;
  border-radius: 5px;
  font-size: 16px;
  cursor: pointer;
}

.form-box button:hover {
  background-color: #218838;
}

.task-list {
  flex: 1;
  min-width: 350px;
}

.task-table {
  width: 100%;
  border-collapse: collapse;
  background: white;
  border-radius: 10px;
  overflow: hidden;
}

.task-table th, .task-table td {
  padding: 12px;
  text-align: left;
}

.task-table th {
  background: #f1f1f1;
}

.task-table td {
  border-top: 1px solid #eee;
}

.btn {
  padding: 5px 10px;
  border: none;
```

```
  border-radius: 5px;
  cursor: pointer;
  font-size: 14px;
}

.btn-delete {
  background-color: #dc3545;
  color: white;
}

.btn-edit {
  background-color: #ffc107;
}

.btn-complete {
  background-color: #007bff;
  color: white;
}

.section-title {
  margin-top: 40px;
  margin-bottom: 10px;
  font-size: 18px;
  color: white;
}

* {
  box-sizing: border-box;
}

.form-box input,
.form-box textarea {
  box-sizing: border-box;
  width: 100%;
  padding: 10px;
  margin-top: 10px;
  margin-bottom: 15px;
  border: 1px solid #ccc;
  border-radius: 6px;
  resize: vertical;
}
```

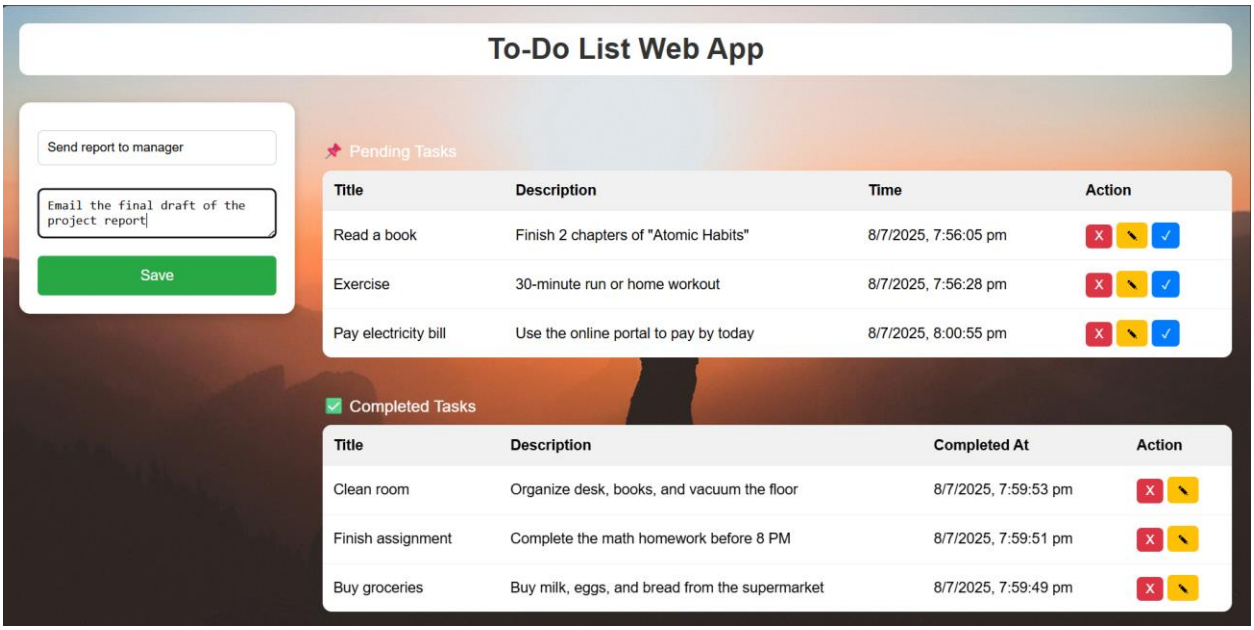
JS

```
let tasks = [];

document.getElementById("todoForm").addEventListener("submit", function (e) {
  e.preventDefault();
  const title = document.getElementById("title").value.trim();
  const description = document.getElementById("description").value.trim();

  if (title && description) {
    tasks.push({
      id: Date.now(),
      title,
      description,
      time: new Date(),
      isCompleted: false,
      completedAt: null
    });

    this.reset();
    renderTasks();
  }
});

function renderTasks() {
  const pendingBody = document.querySelector("#pendingTable tbody");
  const completedBody = document.querySelector("#completedTable tbody");
  pendingBody.innerHTML = "";
  completedBody.innerHTML = "";

  tasks.forEach(task => {
    const tr = document.createElement("tr");

    tr.innerHTML = `
      <td>${task.title}</td>
      <td>${task.description}</td>
      <td>${task.isCompleted ? formatDate(task.completedAt) :
formatDate(task.time)}</td>
      <td>
        <button class="btn btn-delete"
onclick="deleteTask(${task.id})">X</button>
        <button class="btn btn-edit" onclick="editTask(${task.id})">✎</button>
```

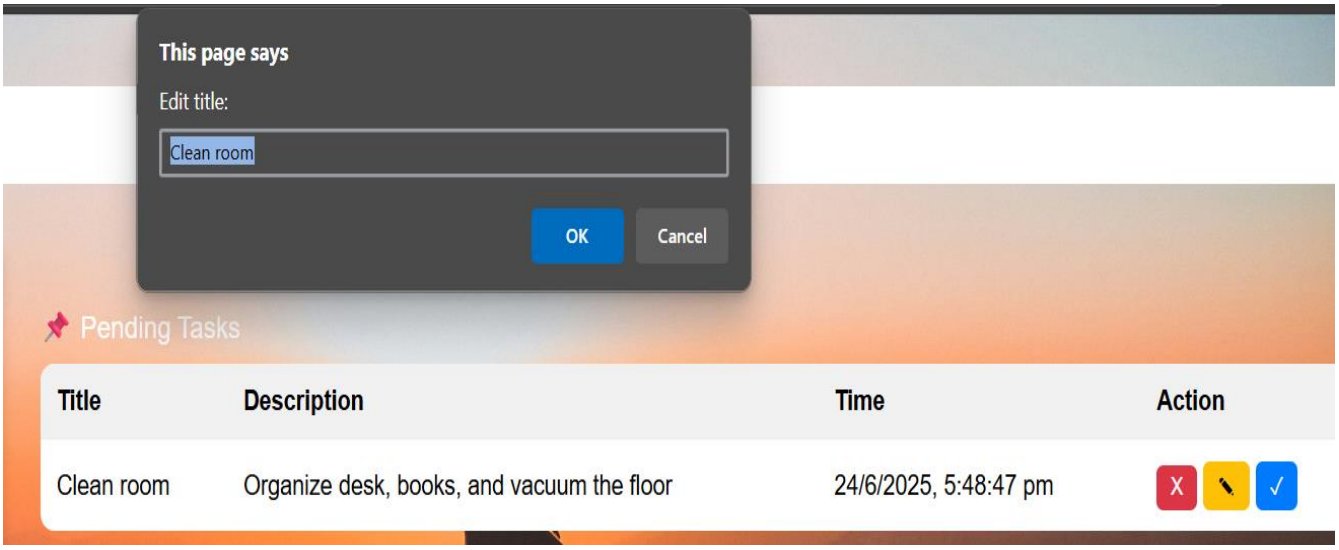
```
        ${!task.isCompleted ? `<button class="btn btn-complete"
onclick="completeTask(${task.id})">✓</button>` : ""}
      </td>
    `;

    task.isCompleted ? completedBody.appendChild(tr) :
pendingBody.appendChild(tr);
  });
}

function deleteTask(id) {
  tasks = tasks.filter(task => task.id !== id);
  renderTasks();
}

function completeTask(id) {
  tasks = tasks.map(task => {
    if (task.id === id) {
      task.isCompleted = true;
      task.completedAt = new Date();
    }
    return task;
  });
  renderTasks();
}

function editTask(id) {
  const task = tasks.find(t => t.id === id);
  const newTitle = prompt("Edit title:", task.title);
  const newDesc = prompt("Edit description:", task.description);

  if (newTitle && newDesc) {
    task.title = newTitle;
    task.description = newDesc;
    renderTasks();
  }
}

function formatDate(date) {
  return new Date(date).toLocaleString();
}
```



Adding Contents



Editing text or content



Summary:

- File Structure: index.html, style.css, script.js

- Run Locally: Open index.html in any browser

- Limitation: No persistent storage; tasks reset on reload

Outcome:

The app works end-to-end: users can add, edit, complete, and delete tasks.

The interface is clean, easy to use, and responsive.