# MERN Stack Training

**V.SRI NITHI**

**717821L351**

Week 3 mini project

## HTML FILE

<!DOCTYPE html>

<html>

<head>

    <title>Task Scheduler</title>

    <link rel=&quot;stylesheet&quot; href=&quot;ts.css&quot>

</head>

<body>

    <div class=&quot;task-form&quot;>

        <input type=&quot;text&quot; id=&quot;taskName&quot; placeholder=&quot;Task Name&quot;>

        <input type=&quot;date&quot; id=&quot;dueDate&quot>

        <select id=&quot;priority&quot;>

            <option value=&quot;low&quot;>Low</option>

            <option value=&quot;medium&quot;>Medium</option>

            <option value=&quot;high&quot;>High</option>

        </select>

        <button id=&quot;addTaskBtn&quot;>Add Task</button>

    </div>

    <div id=&quot;taskList&quot; class=&quot;task-list&quot;>

    </div>

    <script src=&quot;ts.js&quot;></script>

</body>

</html>

# CSS FILE:

\* {

    margin: 0;

    padding: 0;

    box-sizing: border-box;

}

body {

    font-family: Arial, sans-serif;

    background-color: #edbdfc;

    display: flex;

    flex-direction: column;

    align-items: center;

    justify-content: center;

    height: 90vh;

    margin: 0;

}

.container {

    background-color: #fff;

    border-radius: 50px;

    box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);

    padding: 30px;

    text-align: center;

    max-width: 600px;

    width: 100%;

}

h1 {

    font-size: 24px;

    color: #333;

    margin-bottom: 20px;

}

.task-form {

    display: flex;

    justify-content: center;

    align-items: center;

    margin-bottom: 20px;

}

.task-form input[type=&quot;text&quot;],

.task-form input[type=&quot;date&quot;],

.task-form select {

    padding: 10px;

    margin-right: 10px;

    border: 1px solid #f4c7ff;

    border-radius: 4px;

}

.task-form button {

    padding: 10px 20px;

    background-color: #b80cec;

    color: #fff;

    border: none;

    border-radius: 4px;

    cursor: pointer;

}

.task-list {

    max-width: 400px;

    width: 100%;

    background-color: #fff;

    padding: 10px;

    border-radius: 4px;

    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

}

.task {

    display: flex;

    justify-content: space-between;

    align-items: center;

    padding: 10px;

    border: 1px solid #ccc;

    margin-bottom: 5px;

    border-radius: 4px;

    background-color: #e2b7f7;

    transition: background-color 0.2s ease-in-out;

}

.task.completed {

    background-color: #d4edda;

}

.deleteTaskBtn {

    background-color: #dc3545;

    color: #fff;

    border: none;

    border-radius: 4px;

    padding: 4px 8px;

    cursor: pointer;

    transition: background-color 0.2s ease-in-out;

}

.deleteTaskBtn:hover {

    background-color: #c82333;

}

@media (max-width: 600px) {

    .task-form input[type=&quot;text&quot;],

    .task-form input[type=&quot;date&quot;],

    .task-form select,

    .task-form button {

        width: 100%;

        margin-right: 0;

        margin-bottom: 10px;

    }

}

## JAVASCRIPT FILE:

class Task {

    constructor(taskName, dueDate, priority) {

        this.taskName = taskName;

        this.dueDate = dueDate;

        this.priority = priority;

        this.completed = false;

    }

    getTaskDetail() {

        return `${this.taskName} (Due: ${this.dueDate}, Priority:

${this.priority})`;

    }

    toggleCompletion() {

        this.completed = !this.completed;

    }

}

let taskList = [];

function addTask(...tasks) {

    taskList.push(...tasks);

}

function deleteLastTask() {

    taskList.pop();

}

function addTaskToFront(...tasks) {

    taskList.unshift(...tasks);

}

function deleteFirstTask() {

    taskList.shift();

}

function TaskOperations() {

    let totalTasks = 0;

    return {

        getTotalTasks: () => totalTasks,

        addTask: (task) => {

            totalTasks++;

            addTask(task);

        },

        deleteTask: (taskName) => {

            const index = taskList.findIndex(task => task.taskName ===

taskName);

            if (index !== -1) {

                totalTasks--;

                taskList.splice(index, 1);

            }

        }

    };

}

const taskOperations = TaskOperations();

function saveTasks() {

    return new Promise((resolve, reject) => {

        const serializedTasks = JSON.stringify(taskList);

        localStorage.setItem(&#39;tasks&#39;, serializedTasks);

        resolve();

    });

}

async function loadTasks() {

    return new Promise((resolve, reject) => {

        const serializedTasks = localStorage.getItem(&#39;tasks&#39;);

        if (serializedTasks) {

            taskList = JSON.parse(serializedTasks);

            resolve();

        } else {

            reject();

        }

    });

}

const taskNameInput = document.getElementById(&#39;taskName&#39;);

const dueDateInput = document.getElementById(&#39;dueDate&#39;);

const priorityInput = document.getElementById(&#39;priority&#39;);

const addTaskBtn = document.getElementById(&#39;addTaskBtn&#39;);

const taskListContainer = document.getElementById(&#39;taskList&#39;);

function renderTasks() {

    taskListContainer.innerHTML = &#39;&#39;;

    taskList.forEach(task => {

        const taskItem = document.createElement(&#39;div&#39;);

        taskItem.className = `task ${task.completed ? &#39;completed&#39; : &#39;&#39;}`;

        taskItem.innerHTML = `

            <span>${task.getTaskDetail()}</span>

            <button class=&quot;deleteTaskBtn&quot;>Delete</button>

        `;

        const deleteTaskBtn = taskItem.querySelector(&#39;.deleteTaskBtn&#39;);

        deleteTaskBtn.addEventListener(&#39;click&#39;, () => {

            taskOperations.deleteTask(task.taskName);

            renderTasks();

        });

        taskItem.addEventListener(&#39;click&#39;, () => {

            task.toggleCompletion();

            renderTasks();

        });

        taskListContainer.appendChild(taskItem);

    });

}

function addTaskUI() {

    const taskName = taskNameInput.value;

    const dueDate = dueDateInput.value;

    const priority = priorityInput.value;

    if (taskName &amp;&amp; dueDate &amp;&amp; priority) {

        const task = new Task(taskName, dueDate, priority);

        taskOperations.addTask(task);

        taskNameInput.value = &#39;&#39;;

        dueDateInput.value = &#39;&#39;;

        priorityInput.value = &#39;low&#39;;

        saveTasks().then(() => {

            renderTasks();

        });

    }

}

addTaskBtn.addEventListener(&#39;click&#39;, addTaskUI);

loadTasks().then(() => {

    renderTasks();

});

## OUTPUT:

