COLLEGE CODE :- 9509

COLLEGE NAME :- Holy Cross Engineering College

DEPARTMENT :- CSE

STUDENT NM-ID:-

DAE074EB4A8886B8B224E01DC0C33A55

DATE :-22/09/2025

Completed the project named as

Phase__TECHNOLOGY PROJECT NAME :

TO DO LIST APPLICATION

SUBMITTED BY,

NAME :- K.Sanjay
MOBILE NO :- 9342667280

Todo List MVP Implementation

I'll create a comprehensive todo list application with all the required features in a single HTML file that you can run directly in a browser.

html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Todo List MVP</title>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
        }

        body {
            background: linear-gradient(135deg, #6a11cb 0%, #2575fc 100%);
            min-height: 100vh;
            display: flex;
            flex-direction: column;
            align-items: center;
            padding: 40px 20px;
            color: #333;
        }

        .container {
            width: 100%;
            max-width: 600px;
            background: white;
            border-radius: 16px;
```

```
    box-shadow: 0 10px 30px rgba(0, 0, 0, 0.2);
    overflow: hidden;
    margin-bottom: 30px;
}

header {
    background: #4a6fc7;
    color: white;
    padding: 25px;
    text-align: center;
}

h1 {
    font-size: 2.5rem;
    margin-bottom: 10px;
}

.subtitle {
    font-size: 1rem;
    opacity: 0.9;
}

.app-controls {
    padding: 20px;
    display: flex;
    gap: 10px;
    border-bottom: 1px solid #eee;
}

#new-task {
    flex: 1;
    padding: 15px;
    border: 2px solid #ddd;
    border-radius: 8px;
    font-size: 1rem;
    transition: border-color 0.3s;
}

#new-task:focus {
    border-color: #4a6fc7;
    outline: none;
}

#add-btn {
```

```
    background: #4a6fc7;
    color: white;
    border: none;
    border-radius: 8px;
    padding: 0 20px;
    cursor: pointer;
    font-size: 1.5rem;
    transition: background 0.3s;
}

#add-btn:hover {
    background: #3a5bb0;
}

.filter-options {
    display: flex;
    justify-content: center;
    padding: 15px;
    background: #f8f9fa;
    border-bottom: 1px solid #eee;
}

.filter-btn {
    background: none;
    border: none;
    padding: 8px 16px;
    margin: 0 5px;
    border-radius: 20px;
    cursor: pointer;
    font-weight: 500;
    transition: all 0.3s;
}

.filter-btn.active {
    background: #4a6fc7;
    color: white;
}

#tasks-container {
    padding: 20px;
    max-height: 400px;
    overflow-y: auto;
}
```

```
.task {
    display: flex;
    align-items: center;
    padding: 15px;
    margin-bottom: 12px;
    background: #f8f9fa;
    border-radius: 8px;
    transition: transform 0.2s, box-shadow 0.2s;
}

.task:hover {
    transform: translateY(-2px);
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

.task.completed {
    opacity: 0.7;
}

.task.completed .task-text {
    text-decoration: line-through;
}

.task-checkbox {
    margin-right: 15px;
    width: 22px;
    height: 22px;
    cursor: pointer;
}

.task-text {
    flex: 1;
    font-size: 1.1rem;
    word-break: break-word;
}

.task-actions {
    display: flex;
    gap: 10px;
}

.task-actions button {
    background: none;
    border: none;
```

```
    cursor: pointer;
    font-size: 1.1rem;
    color: #6c757d;
    transition: color 0.3s;
}

.task-actions button.edit-btn:hover {
    color: #4a6fc7;
}

.task-actions button.delete-btn:hover {
    color: #dc3545;
}

.empty-state {
    text-align: center;
    padding: 40px 20px;
    color: #6c757d;
}

.empty-state i {
    font-size: 3rem;
    margin-bottom: 15px;
    color: #ced4da;
}

.stats {
    padding: 20px;
    text-align: center;
    background: #f8f9fa;
    border-top: 1px solid #eee;
    display: flex;
    justify-content: space-around;
}

.stat-item {
    display: flex;
    flex-direction: column;
    align-items: center;
}

.stat-value {
    font-size: 1.8rem;
    font-weight: 700;
```

```
            color: #4a6fc7;
        }

        .stat-label {
            font-size: 0.9rem;
            color: #6c757d;
        }

        footer {
            text-align: center;
            color: white;
            margin-top: auto;
            padding: 20px;
            font-size: 0.9rem;
        }

        @media (max-width: 600px) {
            .app-controls {
                flex-direction: column;
            }

            #add-btn {
                padding: 12px;
            }

            .filter-options {
                flex-wrap: wrap;
            }

            .stats {
                flex-direction: column;
                gap: 15px;
            }
        }
    </style>
</head>
<body>
    <div class="container">
        <header>
            <h1>Todo List MVP</h1>
            <p class="subtitle">Manage your tasks
effectively</p>
        </header>
```

```
        <div class="app-controls">
            <input type="text" id="new-task"
placeholder="Enter a new task...">
            <button id="add-btn"><i class="fas fa-
plus"></i></button>
        </div>

        <div class="filter-options">
            <button class="filter-btn active" data-
filter="all">All</button>
            <button class="filter-btn" data-
filter="active">Active</button>
            <button class="filter-btn" data-
filter="completed">Completed</button>
        </div>

        <div id="tasks-container">
            <!-- Tasks will be added here dynamically -->
            <div class="empty-state">
                <i class="fas fa-clipboard-list"></i>
                <h3>No tasks yet</h3>
                <p>Add a task to get started</p>
            </div>
        </div>

        <div class="stats">
            <div class="stat-item">
                <span class="stat-value" id="total-
tasks">0</span>
                <span class="stat-label">Total
Tasks</span>
            </div>
            <div class="stat-item">
                <span class="stat-value" id="completed-
tasks">0</span>
                <span class="stat-label">Completed</span>
            </div>
            <div class="stat-item">
                <span class="stat-value" id="remaining-
tasks">0</span>
                <span class="stat-label">Remaining</span>
            </div>
        </div>
    </div>
```

```
    <footer>
        <p>Todo List MVP Implementation | Phase 3 -
Week 8 Deadline</p>
        <p>Data is stored in your browser's local
storage</p>
    </footer>

    <script>

document.addEventListener('DOMContentLoaded',
function() {
            // DOM Elements
            const newTaskInput =
document.getElementById('new-task');
            const addBtn =
document.getElementById('add-btn');
            const tasksContainer =
document.getElementById('tasks-container');
            const filterBtns =
document.querySelectorAll('.filter-btn');
            const totalTasksEl =
document.getElementById('total-tasks');
            const completedTasksEl =
document.getElementById('completed-tasks');
            const remainingTasksEl =
document.getElementById('remaining-tasks');

            // State
            let tasks =
JSON.parse(localStorage.getItem('tasks')) || [];
            let currentFilter = 'all';

            // Initialize the app
            function init() {
                renderTasks();
                updateStats();

                // Event Listeners
                addBtn.addEventListener('click', addTask);
                newTaskInput.addEventListener('keypress',
function(e) {
                    if (e.key === 'Enter') addTask();
                });
```

```
    filterBtns.forEach(btn => {
        btn.addEventListener('click', function() {
            setFilter(this.dataset.filter);
        });
    });
}

// Add a new task
function addTask() {
    const taskText = newTaskInput.value.trim();
    if (taskText === '') return;

    const newTask = {
        id: Date.now(),
        text: taskText,
        completed: false,
        createdAt: new Date().toISOString()
    };

    tasks.push(newTask);
    saveTasks();
    renderTasks();
    updateStats();

    newTaskInput.value = '';
    newTaskInput.focus();
}

// Set current filter
function setFilter(filter) {
    currentFilter = filter;

    filterBtns.forEach(btn => {
        if (btn.dataset.filter === filter) {
            btn.classList.add('active');
        } else {
            btn.classList.remove('active');
        }
    });

    renderTasks();
}
```

```
        // Render tasks based on current filter
        function renderTasks() {
            // Clear container
            tasksContainer.innerHTML = '';

            // Filter tasks
            let filteredTasks = tasks;
            if (currentFilter === 'active') {
                filteredTasks = tasks.filter(task =>
!task.completed);
            } else if (currentFilter === 'completed') {
                filteredTasks = tasks.filter(task =>
task.completed);
            }

            // Show empty state if no tasks
            if (filteredTasks.length === 0) {
                const emptyState =
document.createElement('div');
                emptyState.className = 'empty-state';
                emptyState.innerHTML = `
                    <i class="fas fa-clipboard-list"></i>
                    <h3>No ${currentFilter !== 'all' ?
currentFilter : ''} tasks</h3>
                    <p>${currentFilter === 'all' ? 'Add a
task to get started' : 'Try changing your filters'}</p>
                `;
                tasksContainer.appendChild(emptyState);
                return;
            }

            // Render tasks
            filteredTasks.forEach(task => {
                const taskEl =
document.createElement('div');
                taskEl.className = `task
${task.completed ? 'completed' : ''}`;
                taskEl.innerHTML = `
                    <input type="checkbox" class="task-
checkbox" ${task.completed ? 'checked' : ''}>
                    <span class="task-
text">${task.text}</span>
                    <div class="task-actions">
                        <button class="edit-btn"><i
```

```
class="fas fa-edit"></i></button>
                    <button class="delete-btn"><i
class="fas fa-trash-alt"></i></button>
                </div>
            `;

            // Add event listeners to task elements
            const checkbox =
taskEl.querySelector('.task-checkbox');
            checkbox.addEventListener('change', ()
=> toggleTaskCompleted(task.id));

            const deleteBtn =
taskEl.querySelector('.delete-btn');
            deleteBtn.addEventListener('click', () =>
deleteTask(task.id));

            const editBtn =
taskEl.querySelector('.edit-btn');
            editBtn.addEventListener('click', () =>
editTask(task.id));

            tasksContainer.appendChild(taskEl);
        });
    }

    // Toggle task completed status
    function toggleTaskCompleted(id) {
        tasks = tasks.map(task => {
            if (task.id === id) {
                return { ...task, completed:
!task.completed };
            }
            return task;
        });

        saveTasks();
        renderTasks();
        updateStats();
    }

    // Delete a task
    function deleteTask(id) {
        if (!confirm('Are you sure you want to delete
```

```
this task?')) return;

            tasks = tasks.filter(task => task.id !== id);
            saveTasks();
            renderTasks();
            updateStats();
        }

        // Edit a task
        function editTask(id) {
            const task = tasks.find(task => task.id ===
id);
            const newText = prompt('Edit your task:',
task.text);

            if (newText !== null && newText.trim()
!== '') {
                tasks = tasks.map(t => {
                    if (t.id === id) {
                        return { ...t, text: newText.trim() };
                    }
                    return t;
                });

                saveTasks();
                renderTasks();
            }
        }

        // Update statistics
        function updateStats() {
            const total = tasks.length;
            const completed = tasks.filter(task =>
task.completed).length;
            const remaining = total - completed;

            totalTasksEl.textContent = total;
            completedTasksEl.textContent = completed;
            remainingTasksEl.textContent = remaining;
        }

        // Save tasks to localStorage
        function saveTasks() {
            localStorage.setItem('tasks',
```

```
    JSON.stringify(tasks));
            }

            // Initialize the app
            init();
        });
    </script>
</body>
</html>
```

Features Implemented

1. **Project Setup** - Complete HTML/CSS/JS structure with responsive design
2. **Core Features**:
   - Add new tasks
   - Mark tasks as complete/incomplete
   - Edit existing tasks
   - Delete tasks
   - Filter tasks (All, Active, Completed)
   - Task statistics
3. **Data Storage**:
   - Uses browser's localStorage to persist tasks between sessions
   - Maintains task state (text, completion status, ID, creation date)
4. **Testing**:
   - The application includes error handling and user confirmation for destructive actions
   - All functionality can be tested directly in the browser
5. **Version Control Ready**:
   - The code is well-structured and commented for easy maintenance
   - Ready to be committed to GitHub