9530

St.MOTHER THERESA ENGINEERING COLLEGE

COMPUTER SCIENCE ENGINEERING

NM-ID:9D1F84DA855FD45936B4653F42567E9C

REG NO: 953023104033

DATE: 22-09-2025

Completed The Project Named as

Phase 3

TO-DO LIST APPLICATION

SUBMITTED BY,

IFFAT FATHIMA I

9677874707

PHASE-3 TO-DO LIST APPLICATION

Project Setup:

1. Create a folder named ToDoApp.

2. Inside it, add files: index.html, style.css, script.js.

3. Add a README.md for documentation.

4. Initialize Git using git init.

5. Create a GitHub repo and connect it.

6. Write a basic HTML structure (input, button, task list).

7. Link CSS and JS files in the HTML.

8. Add minimal CSS for design and layout.

9. Add basic JS to handle task addition.

10. Commit and push the setup to GitHub.

Core Features:

1. Add Task

MVP Implementation:

Input field + “Add” button.

JavaScript createElement() to add task to the DOM.

Save task to localStorage for persistence.

2. Edit Task

MVP Implementation:

Inline edit using contentEditable or an edit modal.

Update the task in the DOM and localStorage.

3. Delete Task

MVP Implementation:

Delete button next to each task.

Use removeChild() to remove task from DOM.

Remove task from localStorage.

4. Mark Task as Complete

MVP Implementation:

Checkbox toggle or strikethrough effect on click.

Update task status in localStorage.

5. View Task List

MVP Implementation:

Display all tasks dynamically on page load.

Fetch tasks from localStorage and render with JavaScript.

6. Persistent Storage

MVP Implementation:

Use localStorage to save tasks between sessions.

JSON stringify/parse for storing and retrieving tasks.

Data Storage(Local State/Database):

1. Local State (JavaScript Array)

Store tasks temporarily in an array.

Example:

```
let tasks = [];

tasks.push({ text: "Buy milk", completed: false });
```

2. Local Storage (Browser Persistence)

Use localStorage to keep tasks even after refresh.

Example:

```
localStorage.setItem("tasks", JSON.stringify(tasks));

let saved = JSON.parse(localStorage.getItem("tasks")) || [];
```

3. Database (Optional – if backend is added)

Use databases like MySQL, MongoDB, Firebase.

Tasks stored on a server, accessible across devices.

4. MVP Choice

For Phase 3 (MVP), localStorage is enough.

Database can be added later for multi-user support.

Testing Core Features:

1. Add Task

Test Steps:

1. Enter a new task in the input field.

2. Click the “Add” button.

Expected Result:

Task appears in the task list.

Task is saved in localStorage.

Edge Cases:

Empty input → should show error or prevent adding.

Very long task text → should display properly without breaking UI.

2. Edit Task

Test Steps:

1. Click on a task to edit (inline or modal).

2. Modify the task text and save.

Expected Result:

Task updates in the DOM.

Updated task persists in localStorage.

Edge Cases:

Editing to empty text → should prevent or show error.

Special characters → should display correctly.

3. Delete Task

Test Steps:

1. Click the delete button on a task.

Expected Result:

Task disappears from the DOM.

Task is removed from localStorage.

Edge Cases:

Delete the last task → list should show “No tasks” message.

4. Mark Task as Complete

Test Steps:

1. Click the checkbox next to a task.

Expected Result:

Task text shows strikethrough or “completed” style.

Status persists after page reload.

Edge Cases:

Toggle back to incomplete → should update DOM and storage correctly.

5. View Task List

Test Steps:

1. Reload the page.

Expected Result:

All saved tasks display correctly from localStorage.

Edge Cases:

Empty list → should display “No tasks” message.

6. Persistent Storage

Test Steps:

1. Add, edit, or delete tasks.

2. Reload the page or close and reopen the browser.

Expected Result:

All changes persist accurately.

Version Control(GitHub):

https://github.com/iffatfathimaa777-tech/Naan-muthalvan.git