9530

St.MOTHER THERESA ENGINEERING COLLEGE

COMPUTER SCIENCE ENGINEERING

NM-ID: 9D1F84DA855FD45936B4653F42567E9C

REG NO: 953023104033

DATE:15-09-2025

Completed the project named as

Phase 2

TO-DO LIST APPLICATION

SUBMITTED BY,

IFFAT FATHIMA I

96778 74707

Phase 2 - To-Do List Application

Tech Stack Selection

Frontend

HTML5 → Structure of the application (input, buttons, task list).

CSS3 → Styling and layout (colors, fonts, responsive design).

JavaScript (Vanilla JS) → Handles DOM manipulation (add/edit/delete/mark tasks).

Backend

Node.js + Express.js → REST API for handling tasks (CRUD operations).

Database (Optional)

MongoDB (NoSQL) → To store tasks in a collection (fields: id, taskName, status).

Or LocalStorage (for frontend-only version) → To persist tasks in the browser without a backend.

Development & Tools

IDE: VS Code (or any preferred editor)

Version Control: Git + GitHub

Browser: Chrome/Firefox for testing

UI STRUCTURE

1. Header Section

Title: “To-Do List Application”

2. Input Section

Text input box → for entering a new task

“Add Task” button → to add the task to the list

3. Task List Section

Each task displayed in a list format

Options for each task:

Edit → update task details

Delete → remove task

Mark as Completed → checkbox or strike-through styling

API SCHEMA DESIGN

Each task will have the following fields:

```
 {"id": "number",        // unique identifier for the task

   "title": "string",     // task description

   "completed": "boolean" // true = completed, false = pending }
```

API Endpoints

POST /tasks → Add a new task

GET /tasks → Retrieve all tasks

PUT /tasks/:id → Edit/Update a task

DELETE /tasks/:id → Delete a task

PATCH /tasks/:id/complete → Mark task as complete.

DATA HANDLING APPROACH

Frontend-Only Version

Data is managed directly in the browser.

JavaScript Arrays → Used to temporarily store tasks during the session.

localStorage → Used for persistence so that tasks remain even after the page is refreshed or browser is closed.

Example:

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

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

Frontend + Backend Version (With Persistence)

Tasks are stored in a database (e.g., MongoDB) and managed using a REST API.

Data Flow:

1. User action (Add/Edit/Delete) on the UI.

2. JavaScript sends an API request (fetch or axios).

3. Backend (Node.js + Express) processes the request.

4. Database (MongoDB) stores/retrieves tasks.

5. Response returned → UI updates dynamically.

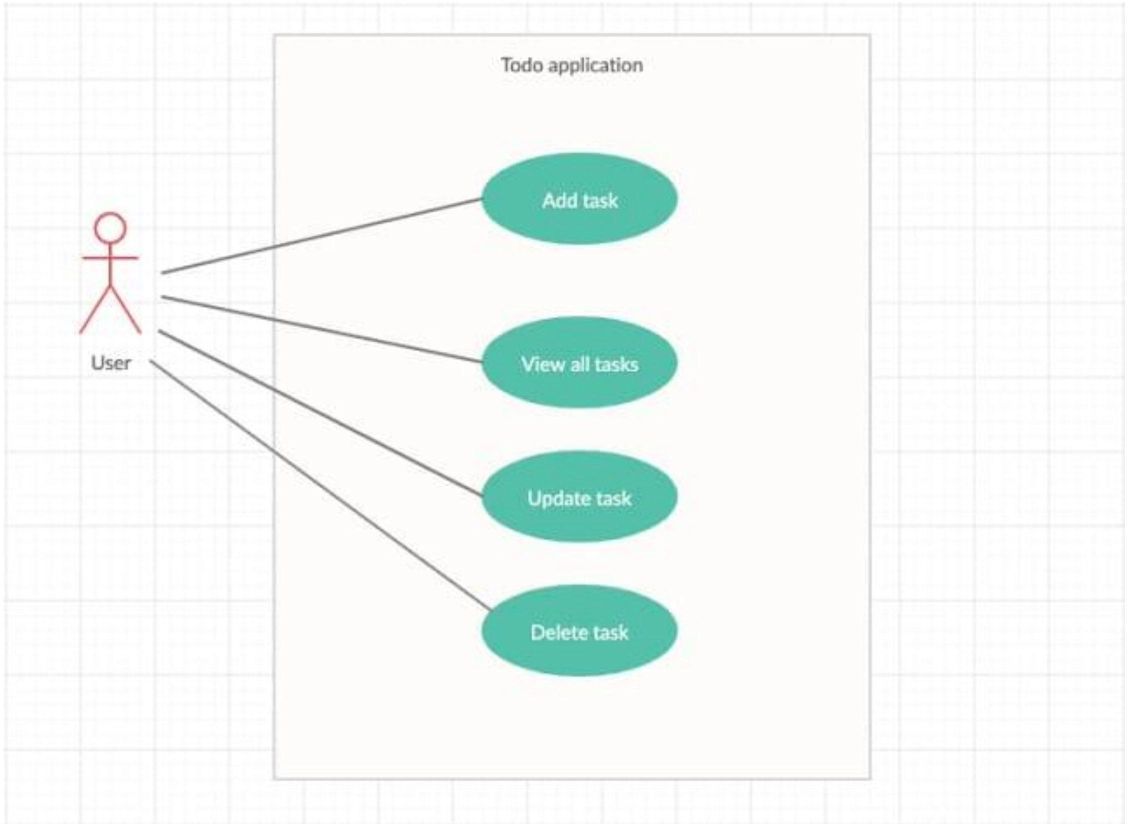
Error Handling & Validation

Prevent adding empty tasks.

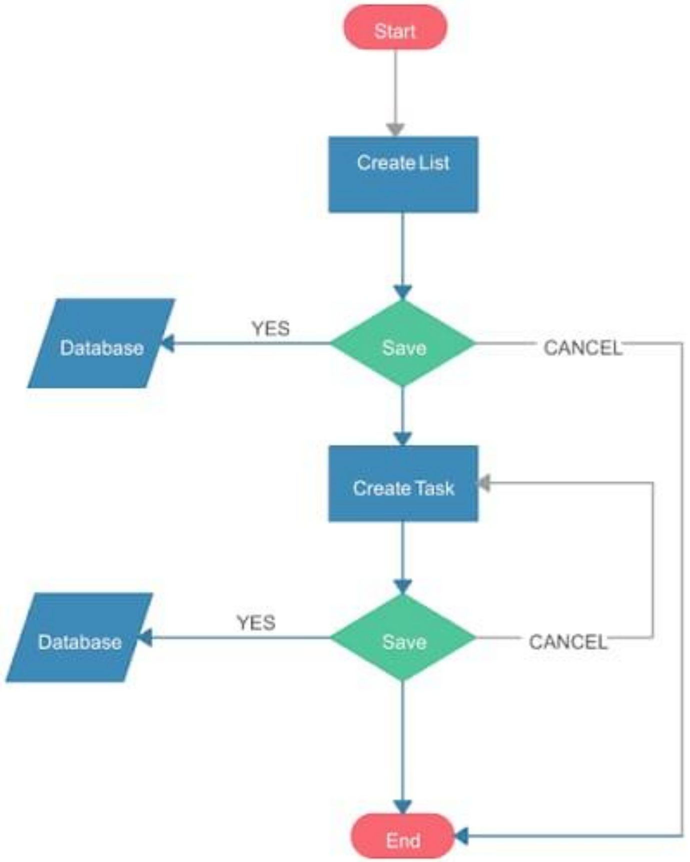
Handle duplicate task entries (optional).

Show user-friendly messages (e.g., “Task added successfully”, “Task deleted”)

Component/module diagram



Basic flow diagram


Start
Create List
Save
YES
Database
CANCEL
Create Task
Save
YES
Database
CANCEL
End