COLLEGE CODE: 9504

COLLEGE NAME: DR G U pope college of engineering

DEPARTMENT: CSE

STUDENT NM-ID: au90423104041

ROLL NO:41

DATE:15-08-2025

Completed the project named as phase:TODOLIST

TECHNOLOGY PROJECT NAME

TODOLIST Applications

SUBMITTED BY,

NAME:Subashini M

MOBILE NO:9688549368

Tech Stack Selection

For the To-Do List Application, the following technologies will be used:

Frontend: HTML, CSS, JavaScript – to design the user interface and handle user interactions.

Backend: Node.js with Express (optional) – to manage tasks and connect frontend with the database.

Database: MongoDB / MySQL (or LocalStorage for simple version) – to store, update, and retrieve tasks.

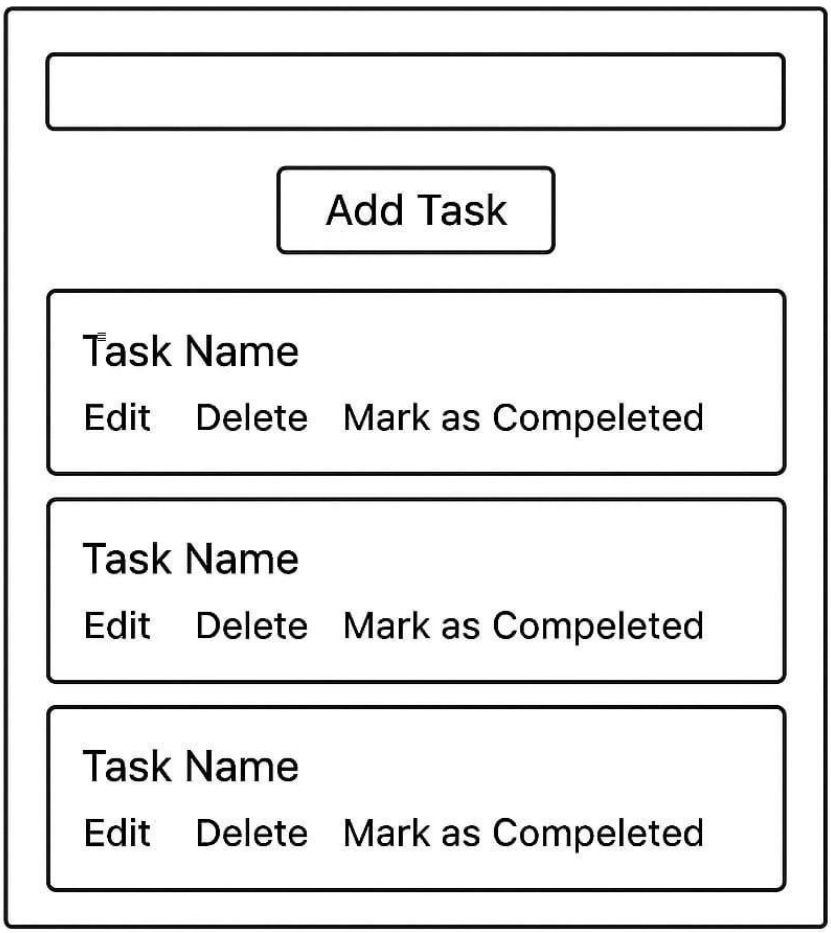
UI Structure / API Schema Design

UI Structure (Frontend Layout):

- Input box to enter tasks
- "Add Task" button
- Task list display with each task showing Edit, Delete, and Mark as Completed buttons

API Schema (Backend Endpoints, optional):

- POST /task → Add a new task
- GET /tasks → Retrieve all tasks
- PUT /task/:id → Update task (edit/mark complete)
- DELETE /task/:id → Delete task



DataHandling Approach

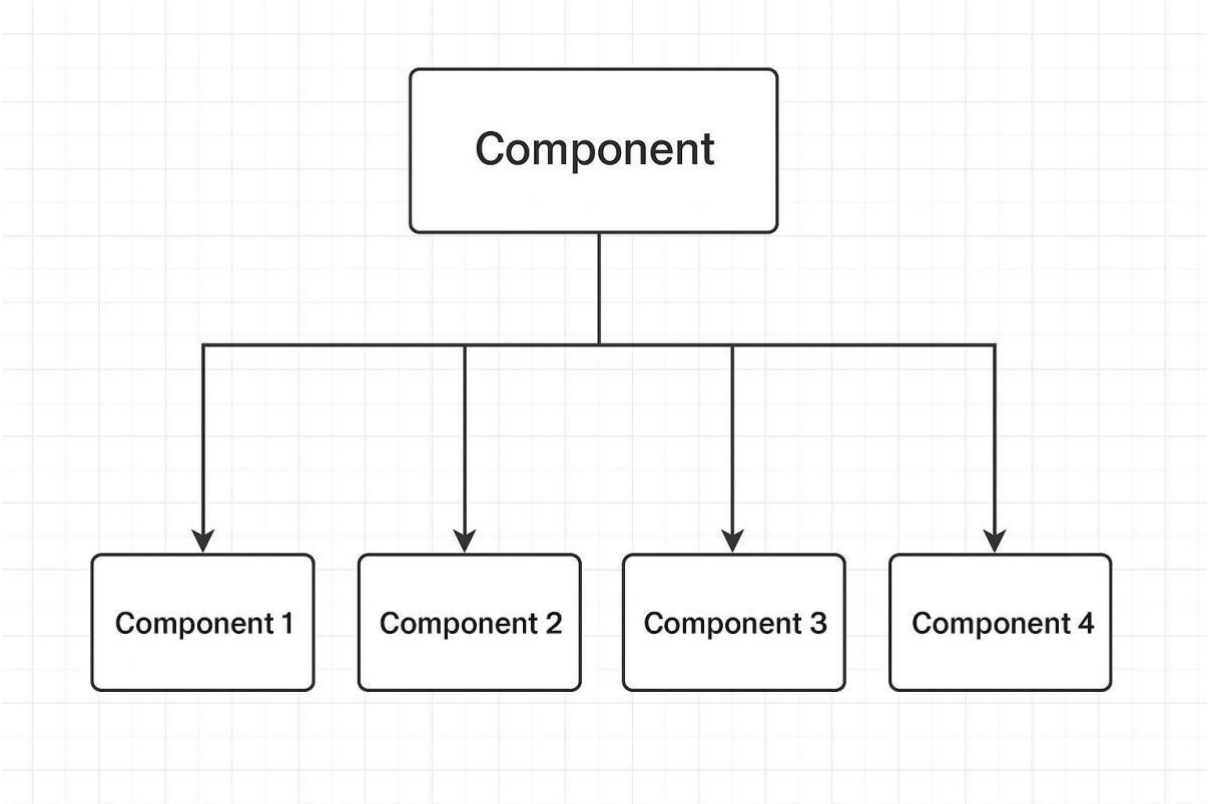
- CRUD operations (Create, Read, Update, Delete)
- Data stored in LocalStorage (basic) or DB (advanced)
- When user adds task → data saved → displayed immediately

Component / Module Diagram

1. Draw boxes with arrows:
   - User → Frontend (UI) → Backend (API) → Database

2. Label each box:

- Frontend handles input & display
- Backend handles logic
- Database stores tasks

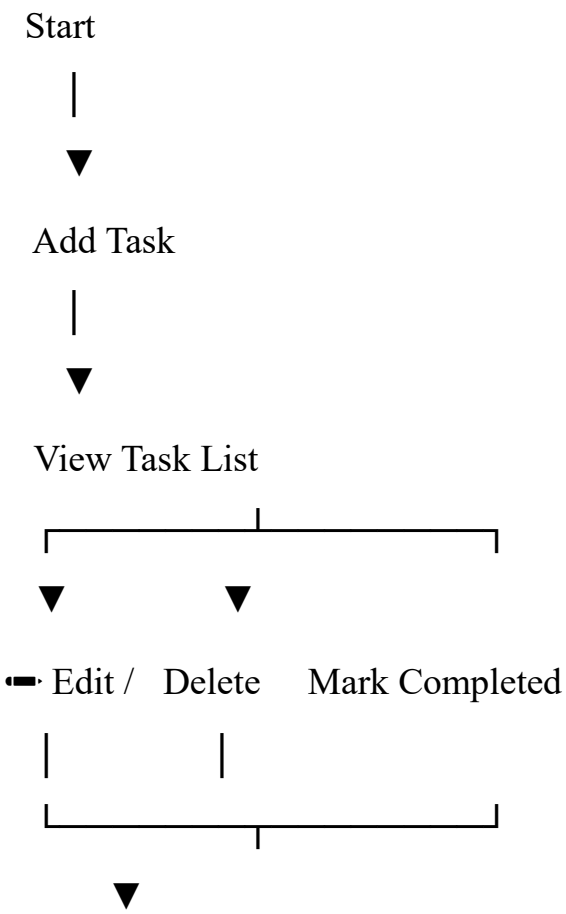


Basic Flow Diagram

1. Use flowchart symbols (Word shapes also enough):

- Start → Add Task → Task Saved → Task Displayed
- If Edit/Delete/Complete → Update → Show Updated List

2. Connect with arrows → easy flow.


Start
Add Task
View Task List
Edit / Delete
Mark Completed
View Updated Task List
End