COLLEGE CODE : 8203

COLLEGE NAME : A.V.C College of Engineering

DEPARTMENT : B.tech – Information Technology

STUDENTNM-ID:C9C912D4BD1EB53B5A04BF0E7A44DD4E

ROLL NO : 23IT84

DATE : 08-09-2025

Completed the project named as Phase 1

TECHNOLOGY PROJECT NAME : To-Do App with React Hooks

SUBMITTED BY,

Name :Ragulkumar A

MOBILE NO :9360457452

Problem Statement

In today's fast-paced lifestyle, individuals often struggle to manage and organize their personal and professional tasks efficiently. Many existing task management tools are either too complex, overloaded with features, or require paid subscriptions. There is a need for a **lightweight, userfriendly To-Do application** that provides the core functionalities of creating, editing, viewing, and deleting tasks with real-time responsiveness.

The application should be simple, reliable, and accessible across devices. By leveraging modern technologies like **ReactJS for the frontend** and **Node.js with MongoDB for the backend**, this app ensures both scalability and smooth user experience.

Users & Stakeholders

Primary Users:

Students – to track assignments, exams, and deadlines.

Working professionals – to manage meetings, project tasks, and reminders.

General users – to keep track of daily errands and personal goals.

Stakeholders:

End Users → Expect a seamless, bug-free, and responsive app.

Developers → Responsible for maintaining scalable, efficient code.

Product Owner / Business Owner → Defines roadmap, ensures value delivery.

Admin (future scope) → Monitors system performance, manages large user base.

User & Stakeholder Stories

User Stories:

1. As a user, I want to **view all my tasks** in a clean interface so that I can stay organized.
2. As a user, I want to **add new tasks quickly** so that I don't forget important things.
3. As a user, I want to **edit a task** so I can update information when plans change.
4. As a user, I want to **delete tasks** so I can remove unnecessary or completed items.
5. As a user, I want to **mark tasks as complete** so I can track my progress.
6. As a user, I want to **see real-time updates** without refreshing the page.

Stakeholder Stories:

1. As a product owner, I want a **minimum viable product (MVP)** that can be tested quickly to gather user feedback.
2. As a developer, I want **clear API endpoints and database schema** so I can build features efficiently.
3. As a product owner, I want the app to be **scalable** so it can handle more users in the future.

Minimum Viable Product (MVP)

The MVP includes:

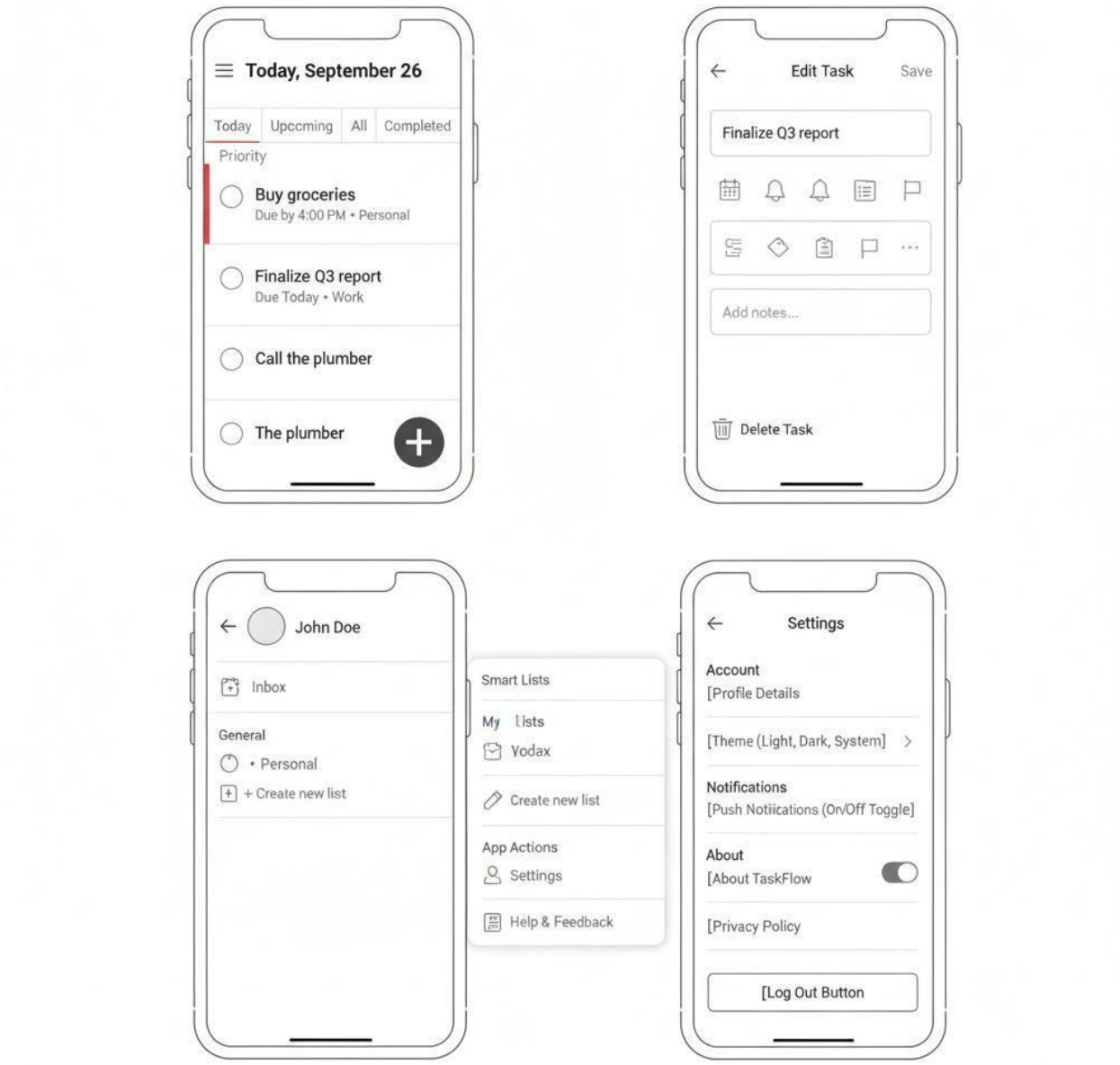
- **Frontend (React + Hooks)**: Task creation, viewing, editing, and deletion with dynamic updates.
- **Backend (Node.js + Express)**: REST API to handle CRUD operations.
- **Database (MongoDB)**: Store tasks persistently.
- **Validation**: Prevents empty tasks and duplicate entries.
- **Responsive UI**: Works on both desktop and mobile.
- **Real-time feedback**: Instant updates upon task changes.

Future scope:

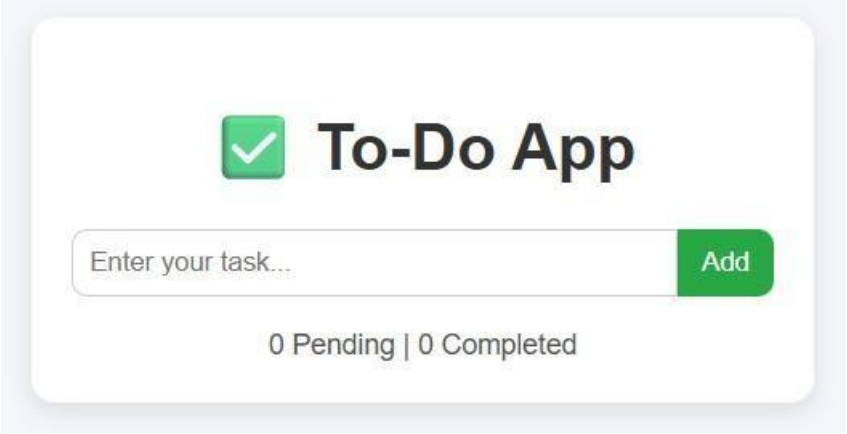
- User authentication (login/signup).
- Due dates and reminders.
- Categories and priorities.
- Cloud deployment (e.g., Vercel + Render/Atlas).

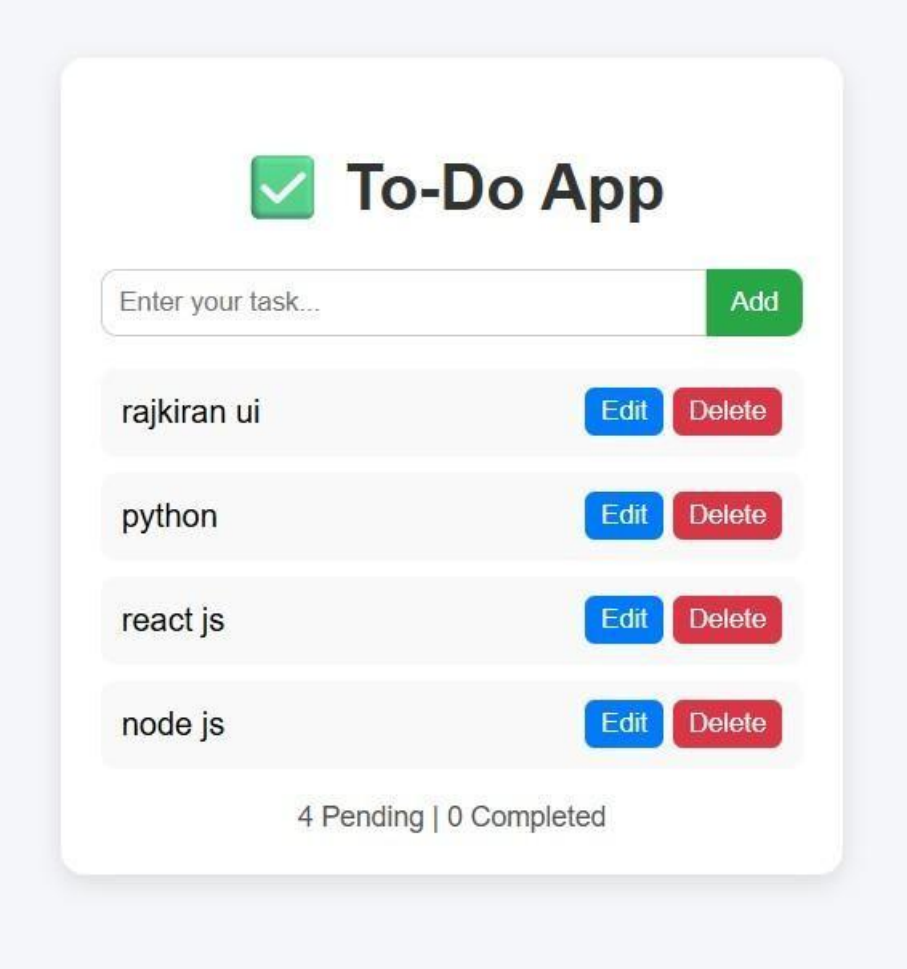
Wireframes

Task Dashboard



Edit Task Modal (popup)





API Endpoint List

Method	Endpoint	Description	Request Body	Response Example
GET	/api/task s	Fetch all tasks	None	[{ "_id": "1", "title": "Task one", "completed": false }]

POST	/api/tasks	Create a ne task	{ "title": w"New Task" }	{ "_id": "2", "title": "New Task", "completed": false }
PUT	/api/tasks/:id	Update a task	{ "title": "Updated" }	{ "_id": "2", "title": "Updated", "completed": false }
DELETE	/api/tasks/:id	Delete a task	None	{ "message": "Task deleted successfully" }
PATCH	/api/tasks/:id	Mark as complete/incomplete	{ "completed": true }	{ "_id": "2", "title": "Updated", "completed": true }

Acceptance Criteria

The app should allow users to add, view, edit, and delete tasks seamlessly.
API should handle CRUD operations with proper status codes (200, 201, 400, 404).
No empty or duplicate tasks should be allowed.
UI should update instantly after CRUD operations without manual refresh.
Tasks must persist in MongoDB even after page reload.
Responsive design for desktop and mobile.
Error messages displayed for failed actions.

Performance: Task list should load within 2 seconds.