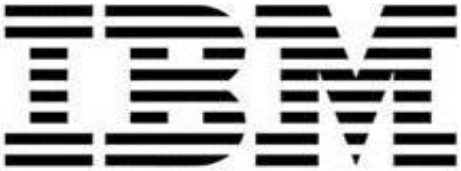




**COLLEGE CODE : 8203**

**COLLEGE NAME : A.V.C College of Engineering**

**DEPARTMENT : B.tech – Information Technology**

**STUDENT NM-ID : C9C912D4BD1EB53B5A04BF0E7A44DD4E**

**ROLL NO : 23IT84**

**DATE : 15-09-2025**

**Completed the project named as Phase 2**

**TECHNOLOGY PROJECT NAME : To-Do App with React Hooks**

**SUBMITTED BY,**

**Name :Ragulkumar A**

**MOBILE NO :9360457452**

## Tech Stack Selection

The choice of technologies ensures scalability, maintainability, and performance.

### Frontend (ReactJS + Hooks):

- React provides a component-based architecture for reusable UI.
- Hooks (useState, useEffect) simplify state management and lifecycle handling.
- TailwindCSS for responsive and modern UI styling.

### Backend (Node.js + Express):

- Node.js handles asynchronous, non-blocking operations.
- Express.js simplifies REST API creation, middleware usage, and routing.

### Database (MongoDB):

- NoSQL document-oriented storage for flexibility.
- JSON-like documents match API responses directly.
- Indexed by timestamps for faster retrieval of recent tasks.

### Additional Tools:

- **Postman** – API testing and debugging.
- **GitHub** – version control and collaboration.
- **Vercel/Netlify** – frontend hosting. • **MongoDB Atlas** – managed cloud database.

## UI Structure

The UI is designed to be simple, responsive, and intuitive.

### Main Screens:

**Dashboard:** Shows list of tasks.

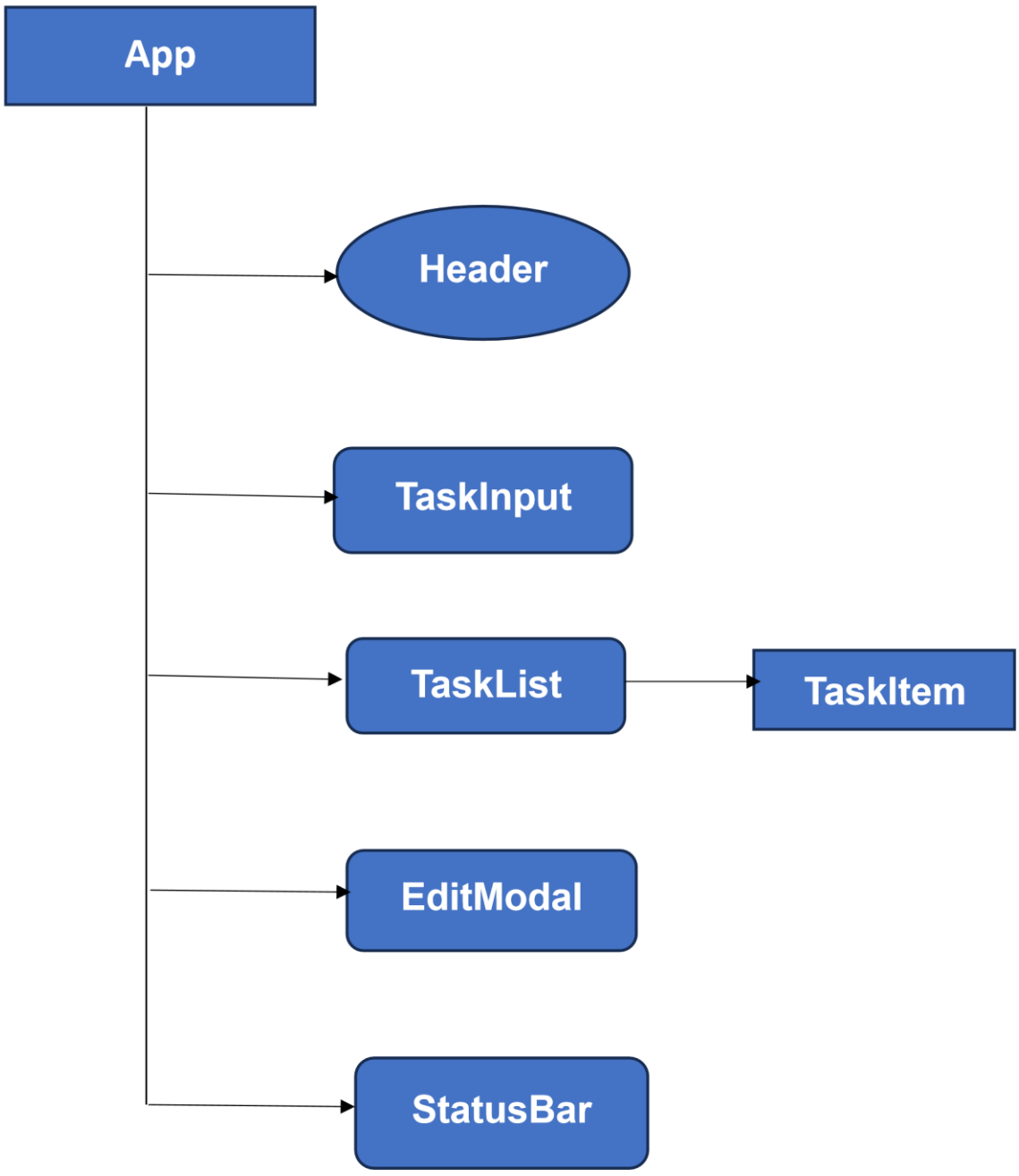
**Task Input Section:** Add new tasks.

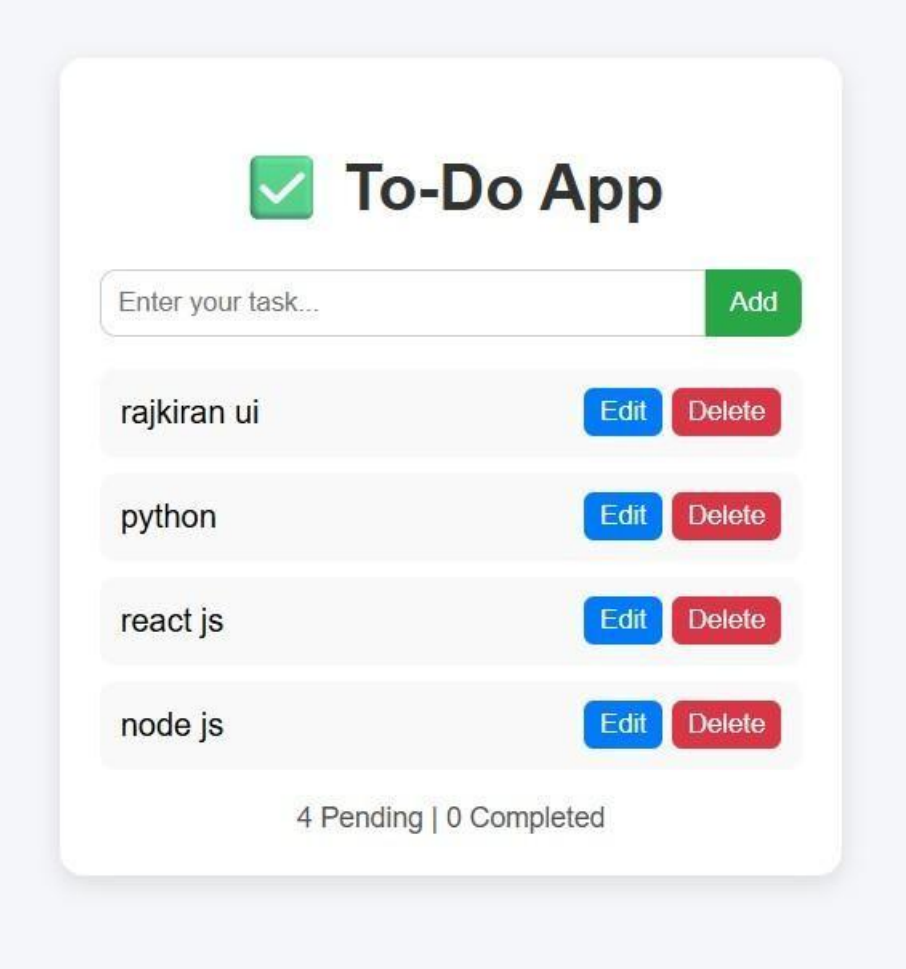
**Edit Modal:** Update task details.

**Delete Confirmation Popup:** Prevents accidental deletion.

**Status Bar:** Shows pending vs completed tasks.

## Component Hierarchy (React):

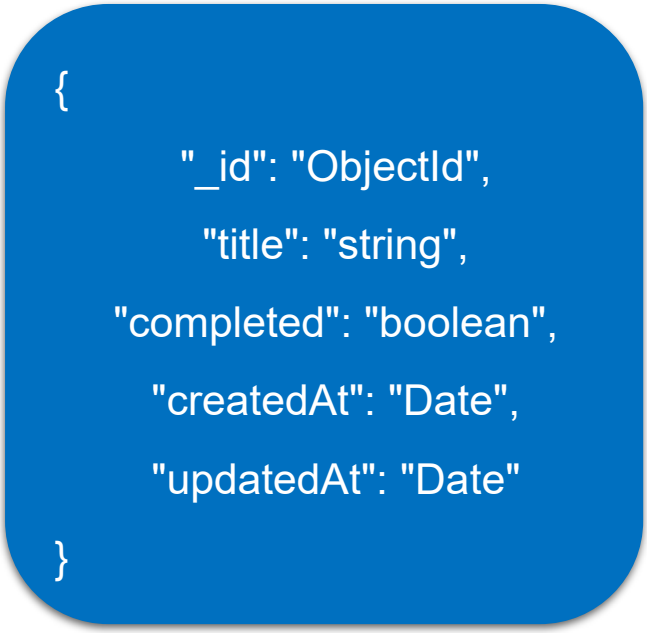





## API Schema Design

The **API** is built using REST principles with clear endpoints.

MongoDB Schema (Task Model):




API Endpoints:

- **GET /api/tasks** → Fetch all tasks
- **POST /api/tasks** → Add a new task
- **PUT /api/tasks/:id** → Update task details
- **PATCH /api/tasks/:id** → Mark complete/incomplete
- **DELETE /api/tasks/:id** → Delete a task

## Data Handling Approach Frontend (React):

- Uses useState for local state management.
- Uses useEffect to fetch tasks on page load.
- Axios/Fetch API for CRUD operations.
- Optimistic updates for real-time feedback.

## Backend (Node + Express):

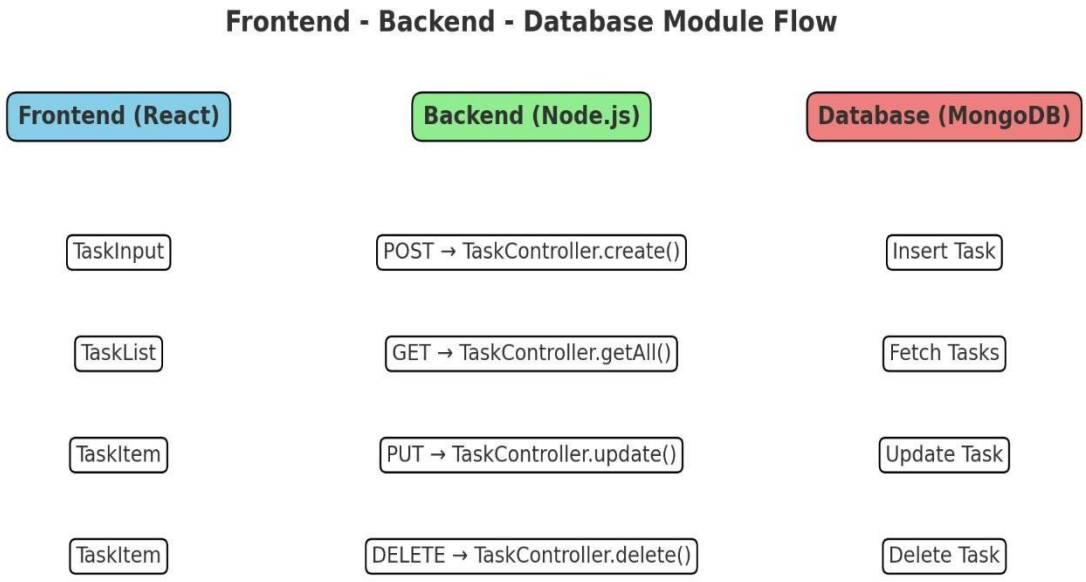
- Request validation with middleware.
- Controller functions handle CRUD.

- Error handling with HTTP status codes.

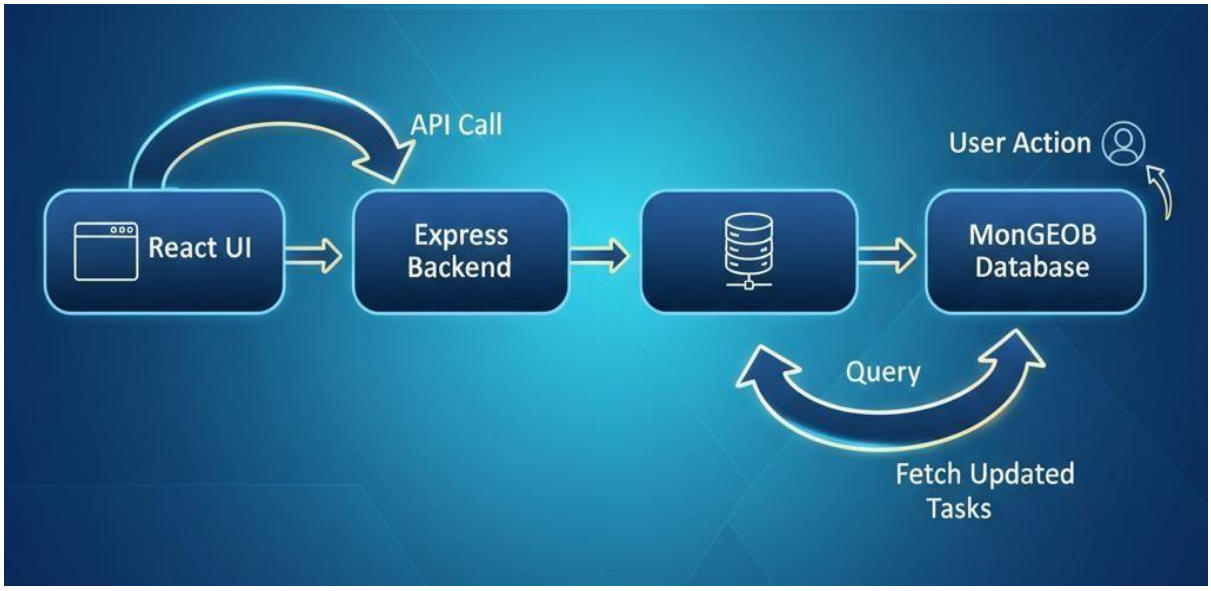
## Database (MongoDB):

- Tasks stored in collections.
- Indexed by createdAt for fast retrieval.
- Auto-managed _id for unique identification.

# Component / Module Diagram




# Flow Diagram




## Expected Outcomes

A **clear architecture blueprint** for frontend, backend, and database.

**API contracts** well-defined for integration.

**UI layout & wireframes** aligned with user needs.

**Data handling strategy** ensures smooth task updates.

**Flow diagrams & component diagrams** created for team clarity.