## **🏗 Architecture Overview – TeamHub**

### **1. Tech Stack**

#### **✅ Backend:**

* **Node.js + Express.js**
* **Prisma ORM** (with PostgreSQL)
* **PostgreSQL** (as the database)
* **JWT** for authentication
* **bcryptjs** for password hashing
* **Docker** for containerization (backend, frontend, DB)
* **Swagger (optional)** for API documentation

#### **✅ Frontend:**

* **React.js (Vite)**
* **MUI (Material-UI)** + **Styled Components** for UI design
* **React Router DOM** for routing
* **Axios** for API calls
* **React-Toastify** for notifications

## 

## 

## 

## 

## **⚙ Key Architectural Components**

### **1. Modular File Structure**

The project is broken into modules for **separation of concerns**, scalability, and maintainability:

#### Backend Folder Structure

teamhub-backend/

├── prisma/

│ ├── migrations/

│ └── schema.prisma

├── src/

│ ├── controllers/

│ ├── middlewares/

│ ├── prisma/

│ ├── routes/

│ └── services/

├── .Dockerignore

├── .env.example

├── Dockerfile

├── index.js

├── package-lock.json

└── package.json

#### Frontend Folder Structure

teamhub-frontend/

├── public/

├── src/

│ ├── assets/

│ ├── components/

│ │ └── layout/

│ ├── features/

│ │ ├── auth/

│ │ ├── dashboard/

│ │ ├── projects/

│ │ ├── tasks/

│ │ └── teams/

│ ├── hooks/

│ ├── layouts/

│ ├── routes/

│ ├── services/

│ ├── theme/

│ └── utils/

├── App.jsx

├── index.css

├── main.jsx

├── .Dockerignore

├── .env.example

├── Dockerfile

├── eslint.config.js

├── index.html

└── package.json

### **2. Prisma as the ORM**

* Prisma provides **type safety**, **auto-generated client**, and **declarative schema modeling**.
* Used **migrations** for schema changes.
* Centralized Prisma client for DB access.

### 

### **3. JWT-Based Auth (Stateless)**

* Token includes only the userId.
* Secure routes with middleware: authenticate.

### **4. Role/Ownership-Based Access Control**

* Only **project/task creators** or **assigned users** can update/delete certain resources.
* **Tasks** can be assigned to the **Team Members** associated with the Project.
* Team members must be **ACCEPTED** to interact with team-based data.

### **5. Team Invites and Membership Status**

* Used InvitationStatus ENUM: PENDING, ACCEPTED, REJECTED.
* Invitees can accept/reject invitations.
* The inviter is now tracked via invitedBy.

### **6. Task Status & History Tracking**

* Tasks maintain a **status enum** (TODO, INPROGRESS, DONE).
* A TaskHistory model keeps track of every status change and comment.
* Only **creator/assignee** can change status.

### **7. Responsive, Clean UI with MUI**

* Charts implemented using **Recharts**.
* All tables support sorting, searching, filtering, and inline status updates.

**8. API-first Design**

* RESTful API structure using modular controllers/services.

Clearly defined endpoints, e.g.:  
  
POST /auth/login

POST /projects/create

GET /teams/my-teams

POST /tasks/:id/comment

### **9. Dockerized Environment**

* Separate containers for frontend, backend, and DB.
* .env used for environment-specific configs.
* docker-compose.yaml orchestrates services together.

## **🔐 Design Decisions**

| **Decision** | **Reason** |
| --- | --- |
| JWT with short expiry | To allow stateless authentication and control session lifetime |
| Prisma ORM | Type-safe, modern ORM |
| MUI + Styled Components | Combines customization with responsiveness and consistent design |
| Modular routes/services | Improves scalability and maintainability |
| Role-based access control | Ensures only authorized users perform sensitive actions |
| API-first backend | Keeps frontend and backend decoupled |
| Dockerized setup | Simplifies deployment and environment replication |
| Charts from backend data | Real-time representation of task data trends |

## 

## **🗂 Additional Features**

* Real-time Dashboard statistics (tasks, status distribution, weekly trends)
* Commenting system on tasks
* Invite suggestions with email autocomplete