**COLLEGE CODE : 9623**

**COLLEGE NAME : Amrita College of Engineering and Technology**

**DEPARTMENT : Computer Science and Engineering**

**STUDENT NM-ID : 3882526D227B55A46A95504A5BA2B8B3583**

**ROLL NO : 962323104033**

**DATE : 12/09/2025**

**Completed the project named as**

**Phase 2 TECHNOLOGY PROJECT**

**NAME : CHAT APPLICATION UI**

**SUBMITTED BY,**

**NAME : Giridhar P J**

**MOBILE NO : 6381055107**

📌 PROJECT PLAN

**Phase 2 – Solution Design & Architecture**

**2.1 Tech Stack Selection**

The chosen technology stack balances **simplicity, scalability, and cost-effectiveness**:

- **Frontend:** React.js with TailwindCSS
    - Reason: Component-based UI, responsive design, faster development.
- **Backend:** Node.js with Express.js
    - Reason: Lightweight, scalable, widely adopted for REST APIs.
- **Database:** MongoDB Atlas
    - Reason: Flexible schema, easy cloud hosting, strong community support.
- **Authentication:** JWT (JSON Web Token)
    - Reason: Stateless, secure, simple to implement.
- **Deployment:**
    - Frontend → Vercel (fast deployment, auto CI/CD).
    - Backend → Render/Heroku (simple Node.js hosting).
    - Database → MongoDB Atlas (scalable, reliable).

Each choice supports future scalability if the project expands from a club app to a larger audience.

---

**2.2 UI Structure (React Components)**

**Component Hierarchy:**

- App
- Login
- Dashboard
    1. TaskList
    2. TaskItem
    3. TaskForm
- **App:** Root component handling routes and authentication state.

- **Login:** Handles login/signup and token storage.
- **Dashboard:** Displays user tasks and provides a central hub.
- **TaskList:** Renders a list of tasks dynamically.
- **TaskItem:** Represents a single task with edit/delete options.
- **TaskForm:** Used to create or update tasks.

Each component communicates via props and uses React's state hooks for reactivity.

---

**2.3 API Schema Design**

**User Schema:**

```
{
  "username": "John Doe",
  "email": "john@example.com",
  "password": "hashed_password"
}
```

**Task Schema:**

```
{
  "title": "Prepare Event Banner",
  "description": "Design a poster for the tech fest",
  "assignee": "ObjectId(User)",
  "status": "In Progress",
  "dueDate": "2025-09-30",
  "createdBy": "ObjectId(User)"
}
```

This schema ensures task ownership and accountability. Status is validated against a predefined set (To Do, In Progress, Done).

---

### 2.4 Data Handling Approach

- **Frontend:**
    - Axios used for API calls.
    - JWT stored in localStorage or sessionStorage.
    - React Context or Redux for managing global state.
- **Backend:**
    - Middleware checks JWT token for protected routes.
    - Controllers handle logic, Models handle DB interactions.
    - Error handling middleware returns consistent JSON responses.
- **Database:**
    - MongoDB collections for users and tasks.
    - Reference keys (assignee, createdBy) link tasks to users.

---

### 2.5 Component/Module Diagram

**Frontend (React)**

App

- Login
- Dashboard
    1. TaskList
    2. TaskItem
    3. TaskForm

**Backend (Node.js)**

- Server
- Routes
    1. /auth
    2. /tasks
- Controllers
- Models (User, Task)
- Middleware (Auth, Error Handling)

---

**2.6 Basic Flow Diagram**

**Login Flow:**

1. User enters credentials on React login page.
2. React sends POST /auth/login to backend.
3. Backend validates credentials, generates JWT.
4. React stores JWT in localStorage.
5. User is redirected to Dashboard.

**Task Flow:**

1. User creates task in React TaskForm.
2. React sends POST /tasks with JWT in header.
3. Backend verifies token, stores task in MongoDB.
4. Backend responds with JSON object of created task.
5. React updates UI with new task in TaskList.

---

📌 **Phase 2 Summary :**

This phase covers the **system design, tech choices, UI components, schemas, data handling, and flow diagrams**. It bridges the gap between requirements and implementation, ensuring the project has a strong architectural foundation.