**Project Proposal: Task Management and Collaboration App**

## **1. Project Overview**

The **Task Management and Collaboration App** is a web-based or mobile application designed to help individuals and teams efficiently manage tasks, collaborate, and track progress. The application will include features such as user authentication, task creation and assignment, collaboration tools, and visual dashboards for progress tracking. The goal is to provide an intuitive and scalable platform for students, professionals, and small businesses.

## **2. Objectives**

* Develop a modern **task management system** that enables users to create, update, and assign tasks.
* Implement **collaboration tools** to facilitate team discussions and file sharing.
* Integrate a **relational database** for structured data storage and retrieval.
* Design an **intuitive GUI** for seamless user interaction.
* Utilize **modern development tools** to ensure scalability and maintainability.

## **3. Key Features**

Here are the **user stories with descriptions** for implementation:

## **3.1 User Authentication**

### **1. Secure User Sign-Up**

* **User Story:** As a user, I want to sign up securely so that I can create an account and access the system.
* **Implementation:**
  + Create a **registration form** with fields for name, email, and password.
  + Implement **password encryption** using bcrypt or Argon2.
  + Use email verification for account activation.
  + Store user credentials securely in the database (MySQL/PostgreSQL).

### **3. Profile Management**

* **User Story:** As a user, I want to manage my profile so that I can update my personal information.
* **Implementation:**
  + Provide an **edit profile** section where users can update their details.
  + Allow users to change passwords securely.
  + Store profile updates in the database while ensuring **data validation**.

### **4. Role-Based Access Control (Admin)**

* **User Story:** As an admin, I want to assign roles to users so that I can control their level of access.
* **Implementation:**
  + Add **role-based access control (RBAC)** logic in the backend.
  + Define roles such as **Admin, Team Member** in the database.
  + Restrict actions based on roles using middleware/auth guards.

### **5. Restricted Access Based on Role**

* **User Story:** As a team member, I want to have restricted access based on my role so that I can only perform actions relevant to my responsibilities.
* **Implementation:**
  + Ensure team members can only see and edit **tasks assigned to them**.
  + Prevent unauthorized access to **admin-only actions**.
  + Implement API **role-based validation** to enforce these restrictions.

## **3.2 Task Management**

### **6. Task Creation**

* **User Story:** As a user, I want to create tasks so that I can assign work to myself or my team.
* **Implementation:**
  + Develop a **task creation form** with fields: title, description, assignee, priority, and deadline.
  + Store tasks in the **tasks table** of the database.
  + Implement **form validation** to prevent empty or invalid entries.

### **7. Task Updating**

* **User Story:** As a user, I want to update tasks so that I can keep information up to date.
* **Implementation:**
  + Allow users to **edit task details** (title, description, assignee, etc.).
  + Update changes in the database and notify affected team members.
  + Implement **audit logs** for task changes.

### **8. Task Deletion**

* **User Story:** As a user, I want to delete tasks so that I can remove unnecessary or completed work items.
* **Implementation:**
  + Provide a **delete button** with a confirmation prompt.
  + Soft-delete tasks (mark as archived) instead of permanently deleting them.
  + Allow **admins to restore deleted tasks** if needed.

### **9. Assigning Tasks**

* **User Story:** As a user, I want to assign tasks to team members so that everyone knows their responsibilities.
* **Implementation:**
  + Add an **"Assignee" field** during task creation.
  + Show assigned tasks in **team members’ dashboards**.
  + Notify assigned members through **email or in-app notifications**.

### **10. Task Deadlines**

* **User Story:** As a user, I want to set deadlines for tasks so that I can manage time effectively.
* **Implementation:**
  + Include a **date picker field** for setting deadlines.
  + Send **reminders for upcoming deadlines**.
  + Display overdue tasks in **red or a separate section**.

### **11. Task Priorities**

* **User Story:** As a user, I want to define task priorities so that I can indicate which tasks need urgent attention.
* **Implementation:**
  + Add **priority levels** (Low, Medium, High, Urgent).
  + Color-code tasks based on priority.
  + Enable **sorting and filtering by priority** in the dashboard.

### **12. Task Status Updates**

* **User Story:** As a user, I want to change the status of tasks so that I can track progress.
* **Implementation:**
  + Allow users to update task status (**To Do, In Progress, Completed**).
  + Implement **drag-and-drop Kanban** board for easy status updates.
  + Automatically notify the team when a task's status changes.

## **3.3 Collaboration Tools**

### **13. Task Comments**

* **User Story:** As a user, I want to comment on tasks so that I can discuss them with my team.
* **Implementation:**
  + Implement a **comment system** under each task.
  + Allow @mentions to notify specific users.
  + Store comments securely in the database.

### **14. Real-Time Notifications**

* **User Story:** As a user, I want real-time notifications for task updates and mentions so that I stay informed.
* **Implementation:**
  + Use **WebSockets (Socket.io) or Firebase** for real-time notifications.
  + Show notifications for **task assignments, updates, and comments**.
  + Allow **email or push notifications** for important updates.

### **15. File Upload & Sharing**

* **User Story:** As a user, I want to upload and share files related to tasks so that I can keep everything organized.
* **Implementation:**
  + Enable **drag-and-drop file uploads** (images, PDFs, docs).
  + Use **cloud storage (AWS S3, Firebase Storage, etc.)** for file management.
  + Provide a preview and download option.

## **3.4 Project Dashboard**

### **16. Visual Task Representation**

* **User Story:** As a user, I want a dashboard with charts to visualize project progress.
* **Implementation:**
  + Use **React Chart.js or D3.js** to display task completion trends.
  + Integrate a **Kanban board** for task movement.
  + Include a **Gantt chart** for timeline visualization.

### **17. Task Filtering**

* **User Story:** As a user, I want to filter tasks by status, priority, and assignee so that I can quickly find relevant information.
* **Implementation:**
  + Implement **dropdown filters** in the task list.
  + Allow **search by keywords** for faster navigation.

## **3.5 Database Integration**

### **18. Secure Database Storage**

* **User Story:** As a developer, I want to store user data, tasks, and comments securely in MySQL/PostgreSQL.
* **Implementation:**
  + Use **ORMs (Sequelize, Prisma)** for database management.
  + Implement **encryption for sensitive data**.
  + Use **database indexing** for faster queries.

## **3.6 Modern GUI**

### **19. Responsive User Interface**

* **User Story:** As a user, I want a modern and responsive interface so that I can easily navigate the system.
* **Implementation:**
  + Use **React.js for web and Flutter for mobile**.
  + Ensure **mobile-friendly design** with CSS Grid/Flexbox.

## **3.7 Advanced Features**

### **20. Third-Party Integrations**

* **User Story:** As a user, I want integrations with Google Calendar and Slack so that I can sync tasks with my workflow.
* **Implementation:**
  + Use **Google Calendar API** for event syncing.
  + Implement **Slack bot notifications** for task updates.

### **21. AI-Based Task Prioritization**

* **User Story:** As a user, I want AI-powered recommendations for task prioritization so that I can focus on important work.
* **Implementation:**
  + Use **machine learning models (Python or TensorFlow.js)** to analyze deadlines and workload.
  + Suggest **priority adjustments based on urgency and dependencies**.

## **4. Technology Stack**

| **Component** | **Technology** |
| --- | --- |
| **Frontend** | React.js (Web) or Flutter (Mobile) |
| **Backend** | Node.js with Express or Django |
| **Database** | MySQL or PostgreSQL |
| **Hosting** | AWS, Heroku, or Firebase |
| **Version Control** | Git with GitHub/GitLab |

## **5. Justification**

This project aligns with the **CP3407 Project** requirements by incorporating:

* **Software development** (Frontend, Backend, Database).
* **Modern Database** (MySQL/PostgreSQL).
* **GUI Development** (React.js or Flutter).
* **Modern Tools** (Git, Cloud Hosting, APIs).

Additionally, the project is **scalable, innovative, and practical**, making it an excellent learning opportunity with real-world applications.

## **6. User Stories**

1. As a **user**, I want to create a task so that I can track my work.
2. As a **team member**, I want to comment on a task so that I can collaborate with my team.
3. As an **admin**, I want to view a dashboard so that I can monitor project progress.

## **7. Timeline**

| Phase | Tasks | Duration |
| --- | --- | --- |
| **Week 1-2** | Research & planning, wireframes | 2 weeks |
| **Week 3-5** | Backend & database development | 3 weeks |
| **Week 6-7** | Frontend development | 2 weeks |
| **Week 8** | Integration & testing | 1 week |
| **Week 9-10** | Final testing & deployment | 2 weeks |

## **8. Expected Outcome**

* A **functional task management application** with authentication, task tracking, and collaboration features.
* Opportunities for further improvements, such as AI-based task prioritization and third-party integrations.

## **9. Conclusion**

The **Task Management and Collaboration App** is an excellent project that meets academic and real-world requirements. It incorporates modern software development practices and tools while offering valuable functionality for users.