MEAN Stack Application Design Document

1. Introduction

Application Name: MyMeanApp

Version: 1.0.0

Description: A modern web application built using MongoDB, Express.js, Angular, and Node.js.

---

2. System Overview

Purpose:

To provide a comprehensive platform for users to manage tasks, collaborate on projects, and track progress through an intuitive web interface.

Scope:

This document outlines the design and architecture of the MyMeanApp, including the system architecture, database schema, API endpoints, and user interface design.

Stakeholders:

- Product Owner: John Doe (johndoe@mymeanapp.com)

- Development Team: Jane Smith (janesmith@mymeanapp.com)

- Operations Team: DevOps Team (devops@mymeanapp.com)

---

3. System Architecture

Architecture Diagram:

![Architecture Diagram](https://via.placeholder.com/800x400.png?text=Architecture+Diagram)

Components:

- **Frontend:** Angular application
- **Backend:** Node.js with Express.js
- **Database:** MongoDB
- **Web Server:** Nginx
- **Authentication:** JWT (JSON Web Tokens)

3.1. Frontend

- **Framework:** Angular
- **Key Features:**
  - Responsive design
  - User authentication and authorization
  - Real-time updates with WebSocket

3.2. Backend

- **Framework:** Express.js
- **Key Features:**
  - RESTful API
  - Middleware for error handling and logging
  - JWT authentication and authorization

3.3. Database

- **Database:** MongoDB
- **Data Storage:**
  - User profiles

  - Tasks
  - Projects
  - Comments
- **Backup:** Daily backups to a secure storage location

3.4. Web Server

- **Server:** Nginx
- **Configuration:** Load balancing, reverse proxy for API, SSL termination

---

4. Database Design

4.1. Schema Diagram

![Database Schema](https://via.placeholder.com/800x400.png?text=Database+Schema)

4.2. Collections and Fields

Users Collection:
- **userId**: ObjectId (Primary Key)
- **username**: String (Unique)
- **email**: String (Unique)
- **passwordHash**: String
- **createdAt**: Date
- **updatedAt**: Date

Projects Collection:
- **projectId**: ObjectId (Primary Key)
- **name**: String

- **description**: String

- **ownerId**: ObjectId (Reference to Users)

- **members**: [ObjectId] (References to Users)

- **createdAt**: Date

- **updatedAt**: Date

Tasks Collection:

- **taskId**: ObjectId (Primary Key)

- **title**: String

- **description**: String

- **status**: String (e.g., 'To Do', 'In Progress', 'Done')

- **assigneeId**: ObjectId (Reference to Users)

- **projectId**: ObjectId (Reference to Projects)

- **createdAt**: Date

- **updatedAt**: Date

Comments Collection:

- **commentId**: ObjectId (Primary Key)

- **text**: String

- **authorId**: ObjectId (Reference to Users)

- **taskId**: ObjectId (Reference to Tasks)

- **createdAt**: Date

---

5. API Design

5.1. Authentication

POST /api/auth/login

- **Request Body:**
  ```json
  {
    "email": "user@example.com",
    "password": "password123"
  }
  ```
- **Response:**
  ```json
  {
    "token": "jwt-token"
  }
  ```

POST /api/auth/register

- **Request Body:**
  ```json
  {
    "username": "newuser",
    "email": "newuser@example.com",
    "password": "password123"
  }
  ```
- **Response:**
  ```json
  {
    "message": "User registered successfully"
  }
  ```

5.2. Projects

GET /api/projects

- **Response:**
  ```json
  [
    {
      "projectId": "60c72b2f9b1e8f001c8b4567",
      "name": "Project A",
      "description": "Description of Project A",
      "ownerId": "60c72b2f9b1e8f001c8b4568",
      "members": ["60c72b2f9b1e8f001c8b4569"],
      "createdAt": "2024-08-22T12:34:56Z",
      "updatedAt": "2024-08-22T12:34:56Z"
    }
  ]
  ```

POST /api/projects

- **Request Body:**
  ```json
  {
    "name": "New Project",
    "description": "Description of the new project",
    "ownerId": "60c72b2f9b1e8f001c8b4568",
    "members": ["60c72b2f9b1e8f001c8b4569"]
  }
  ```
- **Response:**

```json
  {
    "projectId": "60c72b2f9b1e8f001c8b456b",
    "message": "Project created successfully"
  }
  ```

5.3. Tasks

GET /api/tasks

- **Response:**
  ```json
  [
    {
      "taskId": "60c72b2f9b1e8f001c8b456c",
      "title": "Task 1",
      "description": "Description of Task 1",
      "status": "To Do",
      "assigneeId": "60c72b2f9b1e8f001c8b4569",
      "projectId": "60c72b2f9b1e8f001c8b456a",
      "createdAt": "2024-08-22T12:34:56Z",
      "updatedAt": "2024-08-22T12:34:56Z"
    }
  ]
  ```

POST /api/tasks

- **Request Body:**
  ```json
  ```

  {
    "title": "New Task",
    "description": "Description of the new task",
    "status": "To Do",
    "assigneeId": "60c72b2f9b1e8f001c8b4569",
    "projectId": "60c72b2f9b1e8f001c8b456a"
  }
  ~~~
- **Response:**
  ```json
  {
    "taskId": "60c72b2f9b1e8f001c8b456d",
    "message": "Task created successfully"
  }
  ~~~

5.4. Comments

POST /api/comments

- **Request Body:**
  ```json
  {
    "text": "This is a comment",
    "authorId": "60c72b2f9b1e8f001c8b4569",
    "taskId": "60c72b2f9b1e8f001c8b456c"
  }
  ```
- **Response:**
  ```json
  {
  ```

```
    "commentId": "60c72b2f9b1e8f001c8b456e",
    "message": "Comment added successfully"
  }
```

---

6. User Interface Design

6.1. Login Page

- **Features:**
  - Email and Password fields
  - Login button
  - Link to registration page

6.2. Dashboard

- **Features:**
  - Overview of Projects and Tasks
  - Navigation menu
  - User profile and settings

6.3. Project Management

- **Features:**
  - List of Projects
  - Create, Edit, and Delete Project
  - View Project Details and Members

6.4. Task Management

- **Features:**
  - List of Tasks
  - Create, Edit, and Delete Task
  - Task Status and Assignment
  - Comments section

---

7. Security Considerations

- **Authentication:** JWT tokens for secure user sessions
- **Authorization:** Role-based access control for different user roles
- **Data Protection:** Encryption for sensitive data and secure API endpoints
- **Compliance:** Adherence to GDPR and other relevant data protection regulations

---

8. Performance and Scalability

- **Load Balancing:** Use Nginx for distributing traffic
- **Caching:** Implement caching strategies for API responses
- **Scalability:** Horizontal scaling for Node.js and MongoDB as needed

---

9. Testing Strategy

- **Unit Testing:** Use Jasmine and Karma for Angular components, Mocha and Chai for Node.js
- **Integration Testing:** Postman for API testing
- **End