Task Calendar

Introduction

The goal of this project is to create a task management system similar to Jira using the MERN stack. Users will be able to create parent tasks and subtasks, track their status, and manage tasks via a calendar interface. The system will include user authentication and role-based access to ensure that only authorized users can manage tasks.The system will have a frontend interface for user interaction and a backend system to handle requests, manage data.

Architecture Overview

The system will be divided into three main components:

1. Frontend: A React application for user interaction.
2. Backend: An express.js , Nodejs server for handling API requests.
3. Database: MongoDB (Atlas) for data storage.

Features

Here are some key features expected in the design:

1. **User Registration and Authentication**
   - Users can register and log in using their email and password.
   - JWT tokens will be used for session management.
   - Verifies JWTs on protected routes to ensure only authenticated users can access certain endpoints.

2. **Task Management**
   - User should be able to create parent task
   - Subtasks can be created under parent task which will have delete and edit functionality
   - Subtasks have different states: To Do, In Progress, Completed, Postponed, Suspended.
   - List down all the subtasks along with their status for the particular parent task when selected

3. **Calendar Integration**
   - Users can create new tasks directly for the parent task by selecting a date on the calendar.
   - Display all subtasks for the selected date in a calendar
   - Sort all the subtasks for a specific date based on their completion status.

Non-Functional Requirements

1. Scalability: The system should handle multiple users and a large number of tasks without performance degradation.
2. Security: Implement JWT-based authentication and HTTPS for secure data transmission.
3. Performance: Ensure the application loads quickly and responds to user actions in real-time.

High Level Design :

1. Database Designing

Collection 1 : Login/SignUp (Stores user credentials and details.)

- **name**: String
- **email**: String
- **password**: String (hashed)
- **Tasks :** List of tasks associated with this user (foreign key)

Collection 2 : Tasks (Stores parent tasks with references to subtasks)

- **name**: String
- **description**: String
- **Created Date**: Date
- **Subtasks**: List of subtasks associated with this task (foregin key)

Collection 3 : Subtasks (Stores subtasks with references to their parent tasks)

- **name**: String
- **description**: String
- **Created Date:** date and time (Auto DateTime Field)
- **status:** Array
- **ParentTask:** Reference to parent task

2. Web Server Designing

UI Screen 1:
- Login/Signup

UI Screen 2:
- Task List View (Displays all tasks and subtasks in a list format)
- Parent task Creation
- Filter task view based on status of parent card

UI Screen 3:
- Calendar View (A monthly calendar with clickable dates to view tasks and create new ones)

3. API Endpoints

● Login/Signup

Frontend: Create a simple list UI in React that allows for new user to register and login for existing users
Backend: Define an API endpoint in Express.js that handles a POST request to store/fetch the details from/in the database and return them to frontend

● Listing all the parent task

Frontend: Create a simple list UI in React that displays all the parent task and links them to their subtask pages.
Backend: Define an API endpoint in Express.js that handles a GET request to fetch all parent task from the database (MongoDB) and return them to the frontend.

● Show Individual Task and their subtask

Frontend: Design a template in React to select a particular parent card and shows all its subtask associated with it along with its status which can be edit and delete
Backend: Define an API endpoint in Express.js to handle a GET request to fetch parent task details from the database and return them to the frontend.

● New task creation

Frontend: Include a create task button in the calendar integartion page where user can select an particular parent task and create a subtask for it.
Backend: Define an API endpoint in Express.js to handle a POST request from the frontend. This endpoint should execute the following :
1. Create a new subtask for the parent task for a particular date

● Listing down all subtasks based on date selection

Frontend: List down all the subtasks for the associated date and have filter option for view based on status of tasks
Backend: Define an API endpoint in Express.js to handle a GET request from the frontend to get al the subtasks for the particular date