Task Management System with React

Depi Final Project

Group Code: m1e

Abdullah Goda	Leader,
Mohanad Khaled	schema, updating tasks function, api routes, integration between frontend & backend
Hania Elwakeel	Schema of users, function for creating tasks and deleting tasks.
Basel Mohamed	User profile, integration between frontend & backend, word doc, target audience.
Naiera Kamel	
Soussanah Magdy	Authentication function, set up server.js to handle server-side, manage application logic, user integration between frontend & backend, connected MongoDB with the system

In this Documentation, we will provide you with a quick brief of what's in our Task Management System with React project.
Explaining the main points and show some points of the code of making this project.

Schemas section:

Board Model (`board_model.js`):

- Defines the Board schema for a task board, which contains:
    - boardID: A unique identifier for the board.
    - users: An array of ObjectIds referencing the User model. This connects a board to the users who are part of it.
    - tasks: An array of ObjectIds referencing the Task model. This links the board to the tasks associated with it.

Task Model (`task_model.js`):

- Defines the **Task** schema for individual tasks, including:
    - title: The title of the task.
    - description: A description (optional).
    - status: The current status (e.g., "in-progress", "completed").
    - estimate: Estimated time or effort for completion.
    - tags: An array of strings for tagging tasks.
    - createdBy: An ObjectId referencing the **User** who created the task.
    - assigne: An ObjectId referencing the **User** assigned to the task.
    - board: An ObjectId referencing the **Board** to which the task belongs.

User Model (`user_model.js`):

- Defines the **User** schema for managing user data:
  - username: A unique username for each user.
  - email: A unique email address.
  - password: The user's password (encrypted).
  - boards: An array of ObjectIds referencing the **Board** model, allowing users to be part of multiple boards.

This setup allows for relationships between users, tasks, and boards, where:

- **Users** can be part of multiple **Boards**.
- Each **Board** can have multiple **Tasks** and **Users**.
- **Tasks** are linked to a **Board** and can be assigned to specific **Users**.

Controllers section:
Board Controller (board_controller.js):

- Manages operations related to task boards:
  - add_board: Creates a new board based on the request body and saves it to the database.
  - add_board_user: Adds a user to a board using the boardID and user ObjectId.
  - add_board_task: Adds a task to a board by updating the tasks array with a task ObjectId.
  - get_board: Retrieves a board with its users and tasks by performing aggregation, using lookup to reference the users and tasks collections, and projecting only the relevant fields (e.g., usernames, task details).

Task Controller (task_controller.js):

- Handles task-related actions:
  - create_task: Creates a new task in the database based on the provided data in the request.
  - get_all_tasks: Retrieves all tasks from the database and returns them.
  - update_task: Updates specific details (title, tags, estimate, assignee) of a task using its _id.

User Controller (user_controller.js):

- Manages user authentication and registration:
  - signup: Registers a new user, checks if the user already exists by email, hashes the password, and stores the user in the database.
  - signin: Authenticates a user by comparing the hashed password and returning the user if the credentials are valid.

Key Points:

- **Error Handling**: Every function uses try-catch blocks to catch and respond to any errors, ensuring a robust response system.
- **Aggregation and Lookup**: The get_board function uses MongoDB aggregation pipelines to join data from the users and tasks collections with the boards.
- **Password Security**: In user_controller.js, bcrypt is used to securely hash and verify passwords.

Routes Section:

Auth Routes (auth.js):

- Defines routes for user authentication.
    - /signup: Handles user registration by calling the signup function from user_controller.
    - /signin: Handles user login by calling the signin function from user_controller.

Board Routes (board_routes.js):

- Defines routes for board-related operations.
    - /getBoard: Retrieves a specific board, including users and tasks.
    - /addBoard: Creates a new board.
    - /addUser: Adds a user to a specific board.
    - /addTask: Adds a task to a board.

Task Routes (tasks_routes.js):

- Defines routes for task-related operations.
    - /create_task: Creates a new task.
    - /getTasks: Fetches all tasks.
    - /updateTask: Updates specific task details.

Key Points:

- **Modular Routing**: Each resource (auth, board, tasks) has its own route file, making the project structure clean and modular.

- **Express Router**: All routes use the Express router to define HTTP methods (GET, POST) and the respective controller functions that handle the requests.

Server Section:

index.js

- **Express Setup**:
  - Initializes an Express application with express().
  - Uses express.json() middleware to parse incoming JSON requests, allowing the server to handle data sent in JSON format.
- **Server Configuration**:
  - Sets the server to listen on port 3000.
  - Logs a message to the console indicating that the server is running and listening on the specified port.

Frontend Section:

UserProfile.js

- Displays user profile information and allows editing.
- Uses Redux to manage state:
  - handleSave: Dispatches an action to update the user profile.

Navbar.js

- Provides navigation with a button to access the profile if the user is authenticated.

profile.css

- Styles for the user profile component.

redux/userSlice.js

- Manages user profile state, allowing updates through actions.

redux/rootReducer.js

- Combines all reducers for the Redux store.

Target Audience

1. **Professionals and Teams:**
    a. Project Managers: Individuals overseeing projects can use the system to assign tasks, set deadlines, and monitor progress.

    b. Remote Teams: Teams working remotely need effective communication and task assignment tools to ensure collaboration.
2. **Small to Medium-sized Businesses (SMBs):**

    a. These businesses often require affordable, efficient task management solutions that can adapt to their unique workflows.

3. **Freelancers:**
    a. Freelancers managing multiple projects can benefit from organizing tasks, deadlines, and client communications in one place.
4. **Students and Educators:**

    a. Students can use the system to manage assignments and group projects, while educators can track student progress and assignments.
5. **Non-Profit Organizations:**

    a. Non-profits can benefit from better task allocation and project tracking to enhance productivity.

Benefits of the System:

6. **Increased Productivity:** Streamlined task management reduces time wasted on miscommunication and disorganization.

7. **Enhanced Collaboration**: Teams can work together more effectively, seeing who is responsible for what tasks.

8. **Improved Accountability**: Clear assignment of tasks ensures accountability among team members, which leads to better project outcomes.