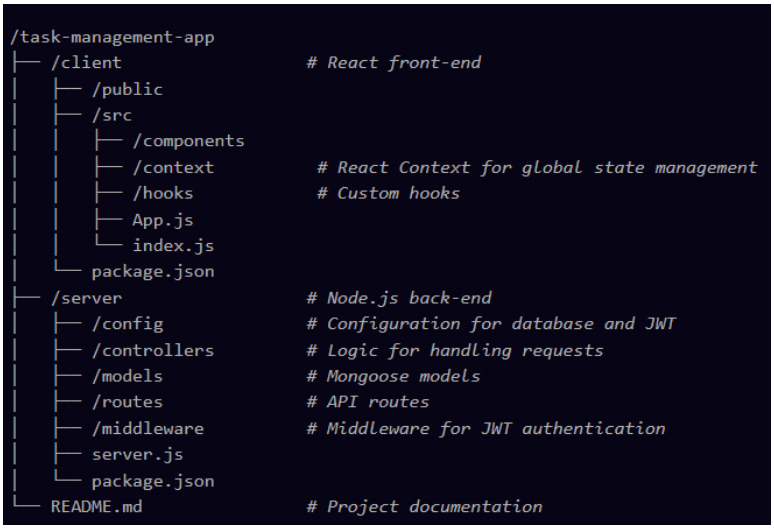
Project Title: Task Manager Application

Project Description

Designed a complete full-stack task management application that will enable users to add, update, and delete tasks with much ease. The application is developed with usability in mind, hence allowing the users to keep track of their tasks effectively. This project integrates a backend for data persistence using MongoDB. It also includes user authentication using JSON Web Token (JWT) to ensure secure access to the application.

Project Description



Technologies Used

1. **Frontend:**

- React.js: To create the user interface and dynamic content of the app.
- React Router: Will help deal with routing and navigation inside the app.
- Axios: To perform HTTP requests against the backend.
- CSS3: Styling the UI components and layout.

- Material-UI or Bootstrap: Prebuilt, responsive UI components.

2. **Backend:**

- Node.js: Runtime of the backend for request handling and running the server.
- Express.js: Web framework to create API endpoints and handle server-side logic.
- MongoDB: NoSQL database to persist data; it stores information about the users and also task data.
- Mongoose: ORM, Object Relational Mapping to MongoDB, for defining the schema and interaction with the database.
- JWT - JSON Web Token: To perform secure authentication for users.

Project Objectives:

- ✓ To implement a Task Manager Application where users will be allowed to handle everyday tasks.
- ✓ To develop an authentication system securely regarding users with JWT.
- ✓ To design a clean and responsive user interface using React.
- ✓ Using MongoDB, tasks and user information are persisted for data retrieval.

Key Features

1. **User authentication**

- ✓ Registration - Based on user's email and password.
- ✓ Login - An authenticated user can log in to their account using JWT to securely access their tasks.
- ✓ Authorization - Only an authenticated user can create/update/delete tasks. JWT is used to verify the user's identity for the routes that are protected.

2. **Task management**

- ✓ Create Tasks: The user can insert a new task with information like title, description, due date, and priority.
- ✓ View Tasks: A dashboard will show all the tasks created based on their status, like To Do, In Progress, and Completed.
- ✓ Update Tasks: The user can modify the information of the task or update the status of any particular task.
- ✓ Delete Tasks: The user will be allowed to delete tasks that are no longer needed.

3. **Data Persistence**

- ✓ All tasks and information regarding the users are maintained using MongoDB so that data is persisted and can be accessed across sessions.
- ✓ Mongoose for the definition of schemas and interaction with MongoDB.

4. **Responsive Design**

- ✓ Application is completely responsive and will work on different screen sizes and devices using CSS3 or any frameworks such as Material-UI/Bootstrap.
- ✓ Mobile-friendly interface. Easy to navigate.

5. **Task Filtering and Sorting**

- ✓ Filter tasks by status, due date, or priority.
- ✓ The tasks can be sorted to make it easier for a user to manage by showing high-priority or urgent tasks.

6. **User-Friendly Interface**

- ✓ The application should be designed with a neat and clean UI so that users can easily interact with it and manage their tasks accordingly.
- ✓ React components are used to reuse them, and efficiency can be achieved for dynamically rendering content.

7. **Notifications (Optional)**

- ✓ Provide notification features that shall remind the user of deadlines or delayed tasks.

8. **Error Handling**

- ✓ Error handling on both the front and back end of typical issues: network errors, authentication failures, and form validation.

Step-by-Step Development

1. **Setup of Project**

Backend Setup

- Setup a new directory for the project and further initialize Node.js for backend-related purposes.

```
mkdir task-management-app
cd task-management-app
mkdir server
cd server
npm init -y
```

- Install packages:

```
npm install express mongoose cors dotenv jsonwebtoken bcryptjs body-parser
```

- In the server directory, create: config, controllers, models, routes, middleware

Frontend Setup

- In the task-management-app, using Create React App, create the React frontend.

```
npx create-react-app client
```

- Switch to the client directory:

```
cd client
npm install axios react-router-dom
```

2. **Back-End Development**

Environment Variables

- In the server directory, create a .env file:

```
MONGODB_URI=your_mongodb_connection_string
JWT_SECRET=your_jwt_secret
```

- **Database Configuration config/db.js:**
- **User Model models/User.js:**
- **User Model models/User.js:**
- **Task Model models/Task.js:**
- **User Authentication Middleware (middleware/auth.js):**
- **Controllers for Handling Requests (controllers/userController.js):**
- **Task Controller (controllers/taskController.js):**
- **Routes (routes/userRoutes.js):**
- **Routes (routes/taskRoutes.js):**
- **Setting Up the Server (server.js):**

3. **Front-End Development**

Create Context for Global State Management

In the src/context directory, add AuthContext.js:

Create Components

- Create components for tasks, forms, and task listing in the src/components directory.

TaskForm Component (components/TaskForm.js):

TaskList Component (components/TaskList.js):

App Component (src/App.js):

4. Deployment

✓ **Backend:** You can deploy the Node.js backend using platforms like Heroku or Render. Don't forget to set up environment variables for MongoDB connection and JWT secret.

✓ **Frontend**: Deploy the React app using services such as Netlify or Vercel. Also, update API URLs of your React app to point to the deployed backend.

Challenges Faced

- Secure authentication of users and handling the JWT token appropriately on the front and back end.

- Management of application state in React, paying a particular emphasis on user interaction and API responses.
- Responsive and fluid design of UI for good use on both desktop and mobile devices.

Future Enhancements

- Real-time collaboration: Allow the collaboration of several users on a single task in real time, using WebSockets or similar technologies.
- Task reminders: it shall integrate with some notification system, like email or SMS, to remind the users about upcoming deadlines.
- Task categories/tags: Provide a tagging system so that users can categorize tasks by project or context.
- Dark mode: Provide users with a dark mode toggle, since most users prefer changing their theme.
- Progress tracking: Implement the feature to provide the user with progress tracking on tasks, such as completion rates or streaks.

Conclusion

The Task Management Application is a vast project, fine-tuning my full-stack development skills. It enables smooth interaction between the backend and frontend, manages user authentication, and efficiently handles task data. Building the project enriched my understanding of how to work with modern web technologies and the best practices in software development.