### **Assignment: Task Management App**

#### **Objective:**

Build a task management app using Vite, React, Tailwind CSS, Shadcn, Express, Mongoose, and JWT. The app will allow users to register, log in, and manage their tasks. Each user will only see their own tasks. Use React Router for navigation and conditionally render a navbar based on the user's authentication status.

#### **Requirements:**

**Frontend:**

1. **Home Page:**
   * Accessible by all users.
   * Include general information about the app.
2. **Login and Register Pages:**
   * Allow users to login and register.
   * Logged in users can not view these pages.
   * Do not render the navbar on these pages.
3. **Tasks Pages:**
   * List Tasks Page: Display a list of tasks for the logged-in user.
   * Task Details Page: Show details for a specific task.
   * Accessible only to logged-in users.
   * Implement CRUD functionality for tasks (Create, Read, Update, Delete).
   * Tasks should be rendered as cards.
   * The user can pin to the top tasks (see implementation in google keep app).
   * **Bonus:** Provide an option to display the task list as a table using Shadcn.
4. **Navigation:**
   * Use React Router for routing.
   * Implement a navbar that renders on appropriate pages (not on login/register pages).
   * The navbar should include:
     + Logo
     + Links for Contact, About, and Tasks
     + An avatar for logged-in users or links to login/register if no user is logged in.
   * Use React Router child routes to conditionally render the navbar on specific pages. (Layout)
   * If a non logged in user tries to access a route that requires auth - redirect it to the login/register page.   
     see this video (ignore the TypeScript):  
     <https://www.youtube.com/watch?v=eFPvXGZETiY&t=428s>
5. **User Feedback:**
   * Show a toaster (using Shadcn) when a user logs in, registers, adds a task, removes a task, or updates a task.
   * Show an alert dialog (using Shadcn) to confirm task deletion.
6. **Bonus: Activity Page:**

implement an activity page to show the activities for the current user. Use shadcn Table component.

**Backend:**

1. **User Authentication:**
   * Implement user registration and login using JWT.
   * Store user data in MongoDB using Mongoose.
   * Ensure that only authenticated users can access tasks pages.
2. **Task Management:**
   * Each user should have their own tasks.
   * Task fields: title, description, body and todo list (with title and isComplete), isPinned, user.
   * Store task data in MongoDB using Mongoose.

### 

### **Instructions:**

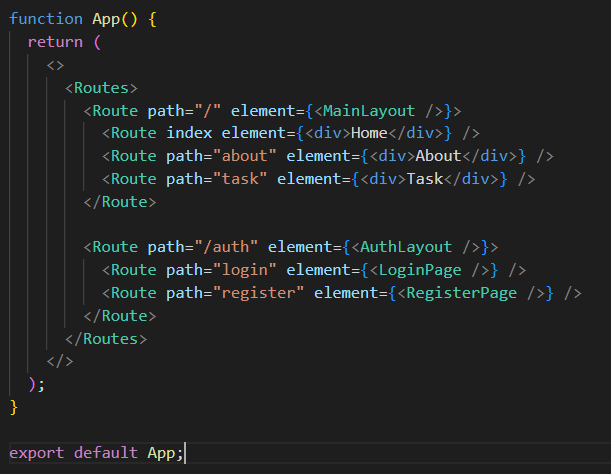
1. **Setup:**
   * Initialize a Vite project for the frontend.
   * Set up Tailwind CSS and Shadcn for styling.
   * Create an Express server for the backend.
   * Set up MongoDB and connect it to the backend using Mongoose.
2. **Frontend Development:**
   * Implement the home, login, register, tasks list, and task details pages using React and React Router.
   * Implement the navbar and conditionally render it based on the user's authentication status using React Router child routes.
   * Style the app using Tailwind CSS and Shadcn.
   * Render tasks as cards and optionally as a table (bonus).
   * Implement toasters (using Shadcn) for user feedback on login, register, adding, removing, and updating tasks.
   * Implement an alert dialog (using Shadcn) to confirm task deletion.
3. **Backend Development:**
   * Implement user authentication using JWT.
   * Create the User and Task Mongoose models.
   * Implement CRUD functionality for tasks, ensuring that each user can only access their own tasks.

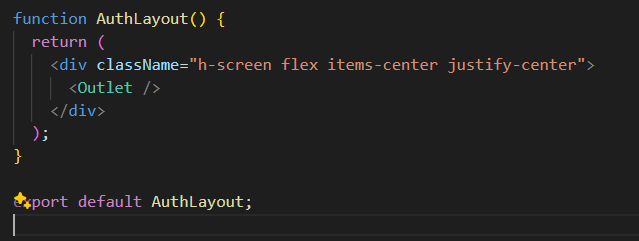
### 

### **Tips:**

* Use React Router's child routes to conditionally render the navbar.
* Use context to manage the authentication state.(AuthContext)
* Ensure proper error handling and validation on both the frontend and backend.
* Follow best practices for security, such as hashing passwords and securing JWTs.

This assignment will test your knowledge of full-stack development, including frontend frameworks, backend development, database management, and user authentication. Good luck!





**Mongoose Models:**

* **User Model Example:**

const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({

username: { type: String, required: true, unique: true },

email: { type: String, required: true, unique: true },

password: { type: String, required: true }

});

const User = mongoose.model('User', userSchema);

module.exports = User;

**Task Model Example:**

const mongoose = require('mongoose');

const todoSchema = new mongoose.Schema({

title: { type: String, required: true },

isComplete: { type: Boolean, default: false }

});

const taskSchema = new mongoose.Schema({

title: { type: String, required: true },

description: { type: String, required: true },

body: { type: String, required: true },

todoList: [todoSchema],

isPinned: { type: Boolean, default: false },

user: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true }

});

const Task = mongoose.model('Task', taskSchema);

module.exports = Task;

**Bonus: Possible Features**1. implement drag and drop for the card layout. hello-pangea/dnd

2. implement archive functionality. when a user Delete a task it first goes to the archive page. the user can permanently delete a task from the archive page (the alert modal should be used here instead of the “archive” option.  
3. Implement due date functionality for todos.  
4. implement image upload for task background