Introduction to the Agile Track System (Project)

The **Agile Track System** is a **React-based project management tool** designed to streamline task management and team collaboration in an agile development environment. This system allows **teams to create, track, and manage tasks efficiently**, ensuring smooth project execution while following agile principles.

Key Features:

1. **User Authentication & Role-Based Access:**
   - Users can sign up, log in, and access role-based functionalities.
   - Admins can manage users, assign tasks, and track progress.
2. **Scrum Team Management:**
   - Admins can create Scrum teams and assign users to them.
   - Each team has its own tasks and progress tracking.
3. **Task Management:**
   - Employees can view their assigned tasks and update their status.
   - Admins can modify task details, assign deadlines, and track completion.
4. **Progress Tracking & Status Updates:**
   - Tasks have statuses like **"To Do," "In Progress," and "Completed."**
   - Admins can view the status history of each task.
5. **Context API for Global State Management:**
   - Stores user authentication details.
   - Ensures persistence of login sessions.
6. **API Integration with Backend:**
   - Uses **Axios** for sending API requests.
   - Supports CRUD operations for users, tasks, and Scrum teams.
7. **Navigation & Redirection:**
   - React Router ensures seamless navigation between login, profiles, and task management pages.

1️⃣ User Authentication (Login & Signup)

📌 Functionality:

- Users can **sign up** with their **email & password**.
- Login authentication using **JWT (JSON Web Token) or session storage**.
- Role-based access control (**Admin & Employee**).

📌 Key Code Components:

- **Signup Component:** Handles user registration.
- **Login Component:** Verifies credentials and redirects users.

2️⃣ Dashboard Module

📌 Functionality:

- Provides an **overview of active tasks, sprints, and user performance**.
- Displays **scrum details, sprint progress, and assigned tasks**.

📌 **Key Code Components:**
- **Dashboard.js**: Fetches and displays **task details**.
- **Charts/Stats (if implemented)** to **visualize project progress**.

3️⃣ Scrum Details Module
📌 **Functionality:**
- Allows admins to **create, update, and track scrum meetings**.
- Employees can **view their assigned tasks and deadlines**.

📌 **Key Code Components:**
- **ScrumDetails.js**: Fetches and updates scrum meeting details.
- **API Integration**: Fetches data from the backend.

4️⃣ User Profile Module
📌 **Functionality:**
- Users can **update their profile information**.
- Admins can **view all users and their assigned tasks**.

📌 **Key Code Components:**
- **UserProfile.js**: Fetches user details and task history.
- **Role-Based Access Control (RBAC)**.

5️⃣ User Contest Module (If Implemented)
📌 **Functionality:**
- Users can participate in **contests for skill improvement**.
- Track progress and leaderboard rankings.

📌 **Key Code Components:**
- **Contest.js**: Fetches contest details and leaderboard.

———————————————————————————

1️⃣ What are Components in React?

Components are **reusable, independent UI blocks** in React. They help build complex UIs by breaking them into smaller, manageable pieces.
There are **two types** of components:
✔ **Functional Components** - Uses functions and React Hooks (useState, useEffect).
✔ **Class Components** - Uses ES6 classes and this.state (older method).

2️⃣ Four Important Things Inside a Component
1. **State (useState)** - Stores and manages component-specific data.
2. **Props (Properties)** - Allows passing data from parent to child components.
3. **JSX (JavaScript XML)** - Defines the component's UI using a syntax similar to HTML.

4. **Lifecycle Methods / Hooks** - Manages component behavior (e.g., useEffect for side effects).

3️⃣ What is the Function of useState?
useState is a React Hook that allows a **functional component** to manage **state** (dynamic data).

const [state, setState] = useState(initialValue);

4️⃣ Which Components Did I Use in My Project?
In my **Agile Track System** project, I used the following components:
✅ **Login Component** - Handles user authentication.
✅ **Signup Component** - Registers new users.
✅ **Dashboard Component** - Displays project and task details.
✅ **User Profile Component** - Shows user-specific details and history.
✅ **Scrum Details Component** - Manages Agile Scrum data.
✅ **User Context Component** - Manages global user state (authentication & roles).
✅ **App Component** - Handles routing and navigation.

1. Did you use any frameworks for form validation
I didn't use any, I did validation using logic.

2. What are the ways components use to communicate with each other
Props and context

3. What is context and why use context instead of props
Context is a global state management and all the states and functions can be used across all the components using usecontext
whereas props can only send data between components so when all components requires same functionality in that case context is used, in this project user authentication is achieved through context

4. How did you connect frontend and backend
Using axios library

whether expectation handling is used?
yes while connecting with backend, try-catch block is used to send and retrieve data

🔷 Why Use Hooks?
✅ Allows **state management** in functional components.
✅ Enables **side effects** like fetching data (useEffect).
✅ Simplifies **code structure** compared to class components.

useState,useEffect,useContext,useRef,useRef

🔷 Why Use Props?
✅ Allows **data sharing** between components.
✅ Helps in **reusability** of components.
✅ Makes components **dynamic** by passing values.

🔷 2. Context API (Global State Management)
📌 **Definition:**
- **Context** is used when **many components** need access to the **same data** without passing props at every level.
- It helps avoid **"prop drilling"**, where data is passed down through multiple components unnecessarily.

1️⃣ What is try and catch in JavaScript?

try and catch are used for **error handling** in JavaScript. They help prevent the application from crashing when an error occurs.

🔷 MVC Components
1️⃣ **Model** (🔷 Data & Business Logic)
- Manages data, business logic, and rules of the application.
- Interacts with the database.
- Example: Fetching user details from a database.

2️⃣ **View** (🔷 User Interface)
- Handles what the user sees (UI/Frontend).
- Displays data from the model.
- Example: React components rendering user profiles.

3️⃣ **Controller** (🔷 Request Handling & Logic)
- Handles user input and updates the model & view.

- Example: A function that processes login requests.