**AGILE TRACK SYSTEM**

**Project Description:**

The **Agile Track System** is a **Single Page Application (SPA)** built with **React** to streamline **task management**. It enables **users (employees)** to log in, view their assigned tasks, update task statuses, and track progress. **Admins** can manage users, create scrum teams, assign tasks, and monitor project activities.

The system uses **React Router** for navigation, **User Context** for authentication, and interacts with a **Fake REST API** for data management. It ensures a smooth workflow for Agile teams by providing an intuitive dashboard, role-based access, and real-time updates.

### **Level 0 (Context Diagram)**

At the highest level, the Agile Track System consists of three key components:

1. **Users/Admins** (External Entities) interact with the system.
2. **Agile Track System** (Process 1.0) processes all user actions.
3. **External API** (Fake API for Users, Tasks, Scrum Teams) provides backend data.

+-------------------------+

| External Entities |

| - User |

| - Admin |

+-------------------------+

|

v

+-----------------------------+

| Agile Task System |

| (Process: 1.0) |

+----------------------------- +

|

v

+-----------------------------------------+

| External API (Fake API) |

| (Users, Tasks, Scrum Teams) |

+------------------------------------------+

**Explanation**:

* **External Entities (Users & Admins):** They access the system through the frontend (React App).
* **Agile Track System (SPA):** This is the main system that handles navigation, authentication, task tracking, and team management.
* **External API:** The system communicates with the backend API (Fake REST API) for fetching/storing data.

### **Level 1 DFD (Decomposition of Process)**

Now, let's break down the **Agile Task System** **Application** process (Process 1.0) into more detailed steps.

+-----------------------+ +---------------------------------+

| External Entity | | External API |

| (User) | | (Users, Tasks, Teams) |

+-----------------------+ +---------------------------------+

| |

| |

v v

+---------------------------+ +----------------------------------+

| Process: 1.1 - Login | | Process: 1.2 - Fetch Data |

+---------------------------+ +----------------------------------+

| |

| |

v v

+----------------------------------+ +----------------------------------------------+

| Process: 1.3 - Dashboard | | Process: 1.4 - View Scrum Teams |

+----------------------------------+ +----------------------------------------------+

| |

| |

v v

+-----------------------------------+ +--------------------------------------------+

| Process: 1.5 - View Tasks | | Process: 1.6 - Add New Scrum |

+----------------------------------- + +-------------------------------------------+

| |

| |

v v

+--------------------------------------+ +-----------------------------------------+

| Process: 1.7 - Update Status | | Process: 1.8 - Manage Users |

+--------------------------------------+ +------------------------------------------+

**Explanation:**

**Process 1.1: Login**

* Users and Admins log in using the **Login Component** (Login.js).
* The credentials are verified using an **External API**.
* If authentication is successful:
  + The user is redirected to their respective **Dashboard**.
  + The **User Context** is updated with the user’s role and session data.

**Process 1.2: Fetch Data**

* Once logged in, the system fetches data from the backend:
  + **Users & Roles**: Determines if the user is an employee or an admin.
  + **Scrum Teams & Tasks**: Loads assigned tasks and scrum team details.

**Process 1.3: Dashboard**

* The **Dashboard Component** (Dashboard.js) provides an overview of scrum teams.
* The user sees options based on their role:
  + Employees see their **assigned tasks**.
  + Admins see options to **manage scrum teams and users**.

**Process 1.4: View Scrum Teams**

* Users can view their assigned scrum teams.
* Clicking **Get Details** fetches:
  + **Team Name**
  + **List of Tasks**
  + **Users in the Team**
* Admins can **edit tasks** and **assign users**.

**Process 1.5: View Tasks**

* Employees can view tasks assigned to them.
* The **User Profile Component** (UserProfile.js) displays:
  + **Task Name**
  + **Task Description**
  + **Current Task Status**
* Admins see an **overview of all users and their tasks**.

**Process 1.6: Add New Scrum (Admins Only)**

* The **Admin Dashboard** allows the creation of new Scrum teams.
* Admins enter:
  + **Scrum Name**
  + **Task Title, Description, and Status**
  + **Assigned User**
* Data is stored using***.***

### **Data Flow**

**Data Flow for React Application**

1. **User Logs In**
   * The Login.js component captures credentials.
   * The User Context updates the state.
   * The user is redirected to the **Dashboard**.
2. **Data Fetching**
   * The Dashboard.js component fetches **tasks, teams, and users**.
   * The data is stored in the **React state** for easy access.
3. **User Interaction**
   * Employees **view and update tasks**.
   * Admins **manage scrum teams and users**.
4. **State Management**
   * React' s **use Context** handles authentication and session storage.
   * Changes are reflected in the **User Profile** and **Dashboard** dynamically.
5. **Navigation**
   * react-router-dom manages navigation between pages.
   * Nav.js dynamically shows menu options based on login state.
6. **User Logout**
   * Clicking **Logout** clears the user session.
   * The system redirects the user to the **Login Page**

### **React- Specific Implementations**

* **Navigation**
  + react-router-dom is used to manage routes (/, /login, /signup, /profiles).
  + Conditional rendering in Nav.js ensures users see correct links.
* **State Management**
  + User Context maintains user authentication state across the application.
* **API Calls**
  + Fake REST API simulates authentication, task updates, and user management

### **NOTES:**

* **Agile Track System** is a **React-based SPA** designed for Agile project management.
* Users interact through **Login, Dashboard, and Profile Pages**, ensuring easy access to tasks and teams.
* The system **fetches and updates data** from a **Fake REST API**, handling user authentication, tasks, and scrum teams.
* **State management (User Context API)** maintains authentication and ensures smooth navigation.
* **React Router** handles page transitions, ensuring a seamless Single Page Application experience.
* **Admins can manage users, assign tasks, and update statuses**, while employees track their assigned work.