Assignment Overview:

This assignment will focus on building a secure multi-page React application using **React Router**, managing component states and lifecycle, handling events, passing props between components, and implementing **role-based access control** and **security best practices**.

Learning Objectives:

- Understand and implement key features of React, including **state management**, **component lifecycle**, **event handling**, and **routing**.
- Build reusable **functional** and **class components**.
- Use **props** for passing data between components and manage **state** using `useState` and `useEffect` hooks.
- Create a **multi-page application** using **React Router** with protected routes and role-based access control.
- Implement security measures like **input validation**, **CSRF** and **XSS** prevention, and **secure session management**.

Part 1: React Basics and Component Lifecycle (2 hours)

Task 1: Setting up a React Project

- Create a new React project using `create-react-app` or `Vite`.
- Install **React Router** using `npm` or `yarn`.

Task 2: Creating Components

- Create two types of components in the project:
  - **Functional Component** for displaying a user profile page.
  - **Class Component** for handling user authentication (login page).
- Add the following features to each component:
  - **User Profile**: Display user information, such as name and role (admin/user).
  - **Login**: Create a simple form for username and password.

Task 3: Managing Component Lifecycle

- In the **Class Component** (Login page):
  - Implement `componentDidMount` to initialize form state.
  - Use `componentDidUpdate` to monitor state changes (e.g., successful login).
  - Utilize `componentWillUnmount` to clear session data upon logout.

Deliverables:

- Functional and class components with lifecycle methods.
- Display a user profile using props passed from the parent component.

Part 2: Components, States, and Props (2 hours)

Task 4: State Management in Functional Components

- In the **User Profile** component, use the `useState` hook to manage the user's status (e.g., "Logged In" or "Logged Out").
- Use the `useEffect` hook to fetch user data from a mock API (use `setTimeout` to simulate fetching).

Task 5: Passing Data with Props

- Create a **Parent Component** (Dashboard) that manages the logged-in user's state and passes the user data to the **User Profile** component via props.
- Implement **prop drilling** to pass data from the parent (Dashboard) to deeper child components.

Task 6: Event Handling

- Add an **onClick** event to a Logout button in the **User Profile** component to trigger a state change (log the user out).
- Pass a function from the **Parent Component** to the child component to handle this logout event.

Deliverables:

- A dashboard managing user state and passing data to child components using props.
- Event handling for logging the user out.

Part 3: React Router and Role-based Access Control (2 hours)

Task 7: Multi-page Application using React Router

- Implement **React Router** to navigate between the following pages:
  1. **Login Page**
  2. **User Profile Page**
  3. **Admin Dashboard** (visible only to users with the 'admin' role)

Task 8: Role-based Access Control

- Use the `useState` hook to track the logged-in user's role (e.g., 'admin', 'user').
- Implement **Protected Routes** for the **Admin Dashboard**. Only allow access to users with the 'admin' role. Redirect unauthorized users to the login page.

Task 9: Input Validation and Security

- Add input validation on the **Login Page** (ensure that username and password are not empty).
- Implement security measures:
  - Protect against **XSS** by sanitizing user input.
  - Add a basic **CSRF** protection mechanism by generating and validating a CSRF token upon login.
  - Use **session storage** to manage the user's login state securely.

Deliverables:

- A multi-page React application with protected routes and role-based access control.
- Input validation and basic security measures (CSRF, XSS prevention).

Quiz and Coding Challenge (2 Hours)

Quiz (1 hour)

- The quiz will cover:
  - Key React concepts such as **component lifecycle methods**, **state management**, **props**, and **React Router**.
  - **React vs. Angular**: Compare React's component architecture with Angular's.
  - Questions on **event handling** and differences between **functional** and **class components**.

Coding Challenge (1 hour)

Challenge: Enhance your multi-page React application by integrating:

- **Role-based access control** (admin/user).
- **Input validation** and secure **session management**.
- Implement protection against **common vulnerabilities** (CSRF, XSS).

Scenario:

Build a secure multi-page React application for an organization where:

- Admins have access to an **Admin Dashboard** where they can manage users.
- Regular users can view their **profile** and update basic information.
- Secure sensitive pages using **role-based access control**.
- Implement **client-side navigation** using **React Router** and ensure that only authorized users can access certain routes.

Bonus:

- Implement a **Logout** button that clears the session and redirects to the login page.
- Handle invalid route access by displaying a **404 Not Found** page.

Deliverables:

- Final React project with routing, protected routes, role-based access control, and security implementations.

This assignment should help learners understand core React concepts while practicing real-world development scenarios.