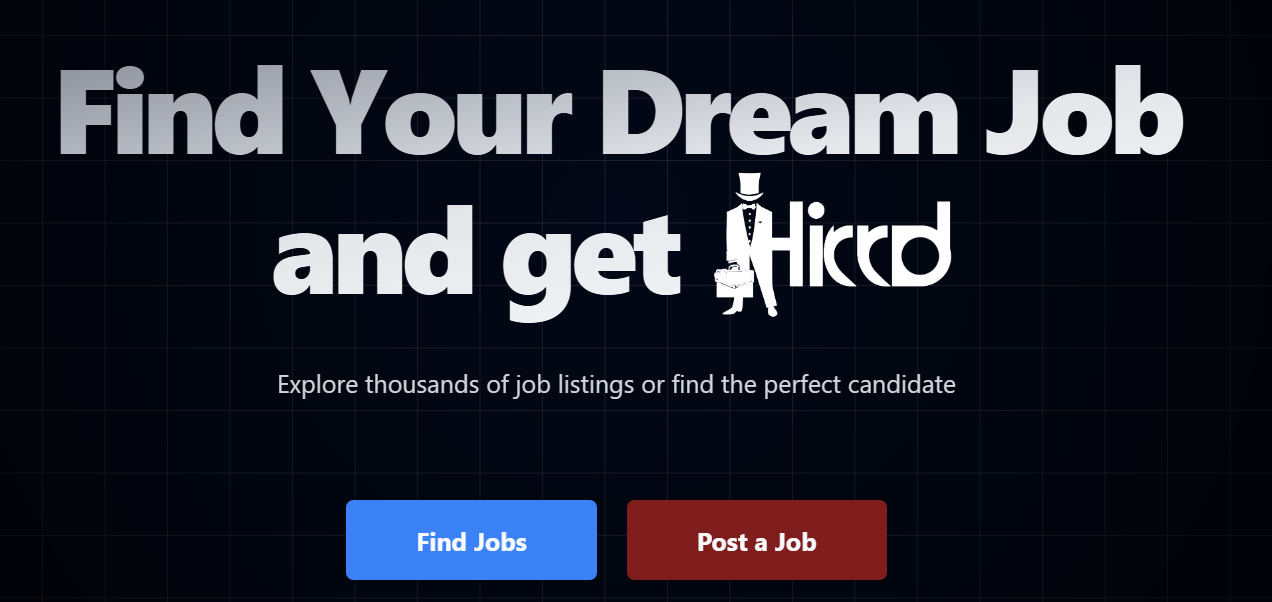
**Report**

1. **Task Description**

The task involves implementing a role-based authorization system in a React application to control access to specific routes and components. This ensures that users can only access features and pages based on their assigned roles, enhancing security and user experience.

1. **Task Output Screenshot**



1. **Widget/Algorithm Used In Task**
2. **Role-Based Access Control (RBAC) Logic:**  
   The project uses a role-based access control system to define user roles (e.g., Admin, User, Guest) and assign permissions. Roles are checked at runtime to determine access to specific routes or components.
3. **ProtectedRoute Component:**  
   A custom ProtectedRoute wrapper ensures secure access by verifying user roles before rendering child components. If the role does not match, the user is redirected to an appropriate fallback page.
4. **React Context API for Global State Management:**  
   The React Context API is used to store and manage authentication and user role data globally. This enables efficient access to user roles across components without prop drilling.
5. **Dynamic Route Matching with react-router-dom:**  
   Dynamic routing with react-router-dom allows route parameters (e.g., /job/:id) and supports fine-grained access control for individual pages based on user roles.
6. **Custom ThemeProvider:**  
   The application integrates a custom ThemeProvider to maintain consistency in UI design while supporting a light and dark mode. This enhances usability and user engagement across roles.