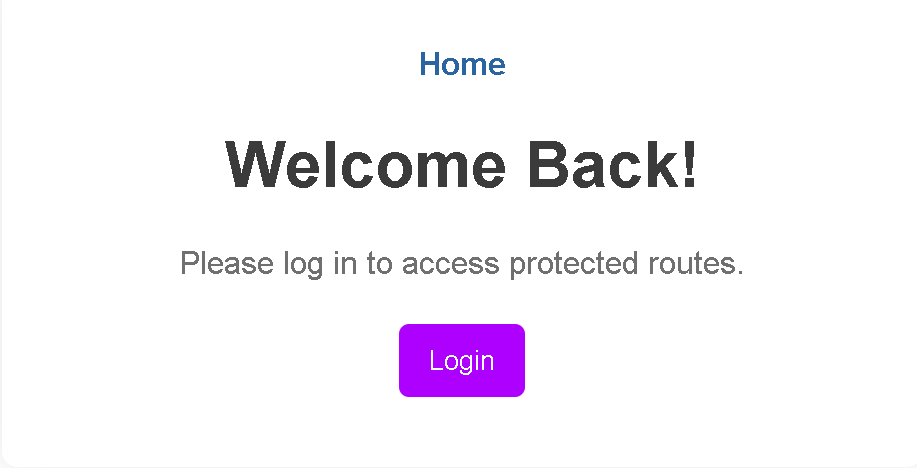
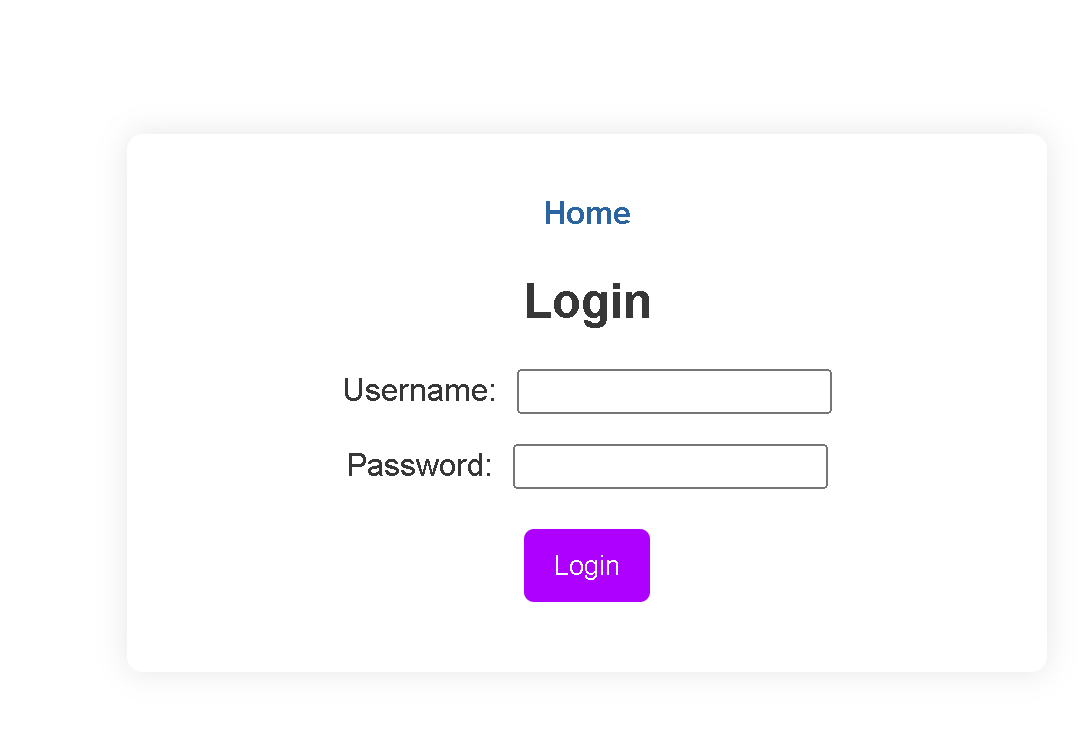
**Report**

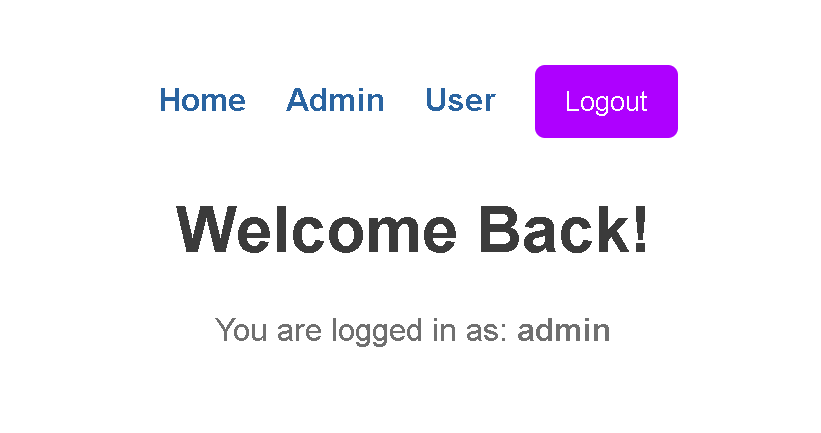
1. **Task Description**

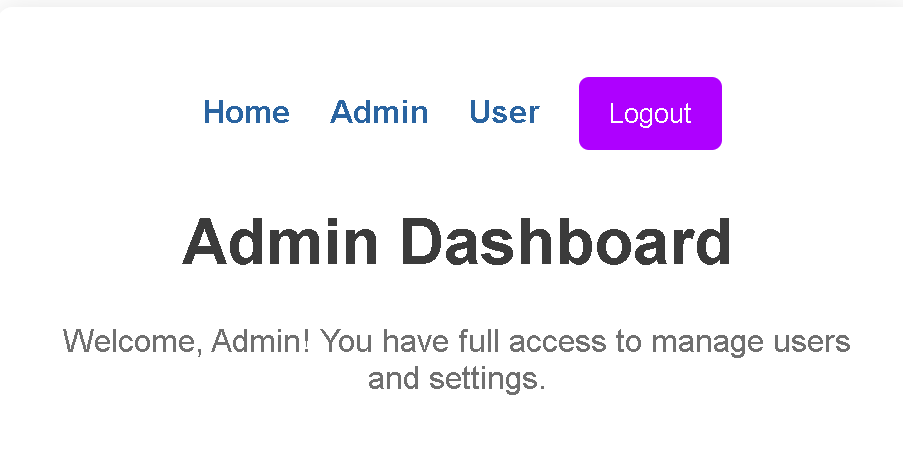
Develop a role-based authorization system to control access to specific routes or components.

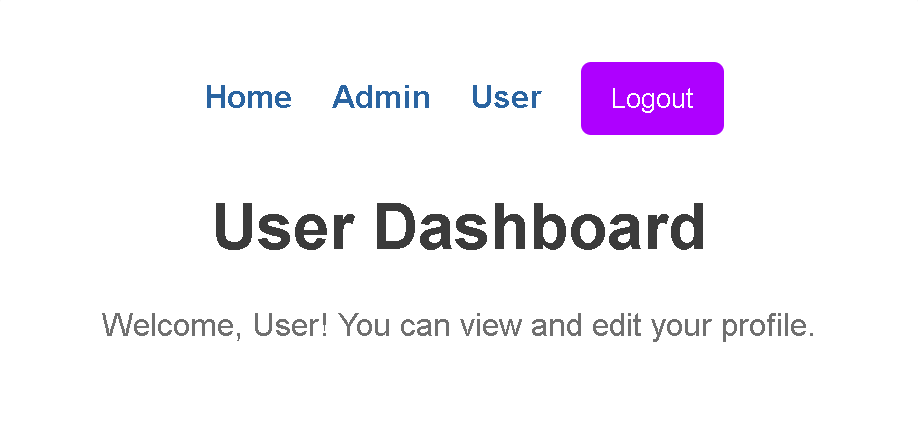
1. **Task Output Screenshots**











1. **Widget/Algorithm Used In Task**

* ***Router and Navigation***

1. Router, Routes, Route (from react-router-dom): It is used to handle multiple routes and page navigation in an application.
2. Link (from react-router-dom): It offers navigation links to different routes such as Home, Admin, and User.
3. useNavigate (from react-router-dom): It is used for navigating between pages.

* ***State Management***

1. useState: It manages state of user, isAuthenticated, and form data, like login credentials.
2. setUser: It saves the role of an authenticated user for role-based access control.

* ***Protected Route (ProtectedRoute Component)***

1. Role-Based Access Control: Only users with 'admin' or 'user' roles are allowed to certain routes like the Admin and User dashboards.
2. Nav (from react-router-dom): redirect to an unauthorized page when the user doesn't have the appropriate role.

* ***Conditional Rendering***

1. User && Link : links and logout button according to whether someone is logged in or not
2. Login Check: if a user is not authenticated, call login page.

* ***Login Form (Login Component)***

1. Input Fields: inputs for the user's name and password, with states managed using useState.
2. Submit Button: triggers the submitting of the form and validates the user
3. Error Message: if the submission action failed

* ***Logout Button***

1. Logout functionality that resets the user's state. It then redirects the user to the home page.

* ***Dashboard Pages***

1. Admin and User Components: pages showing a message based on the type of the user 8.

* ***Styling***

1. Basic CSS applies for styles such as layout, buttons, navigation links, and error messages.