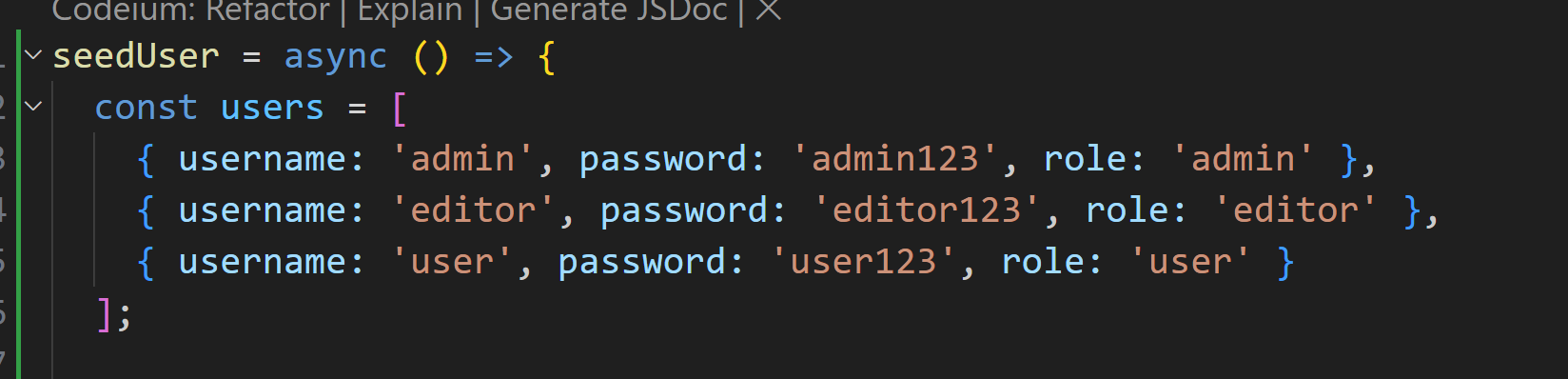
**ROLE BASED ACCESS:**

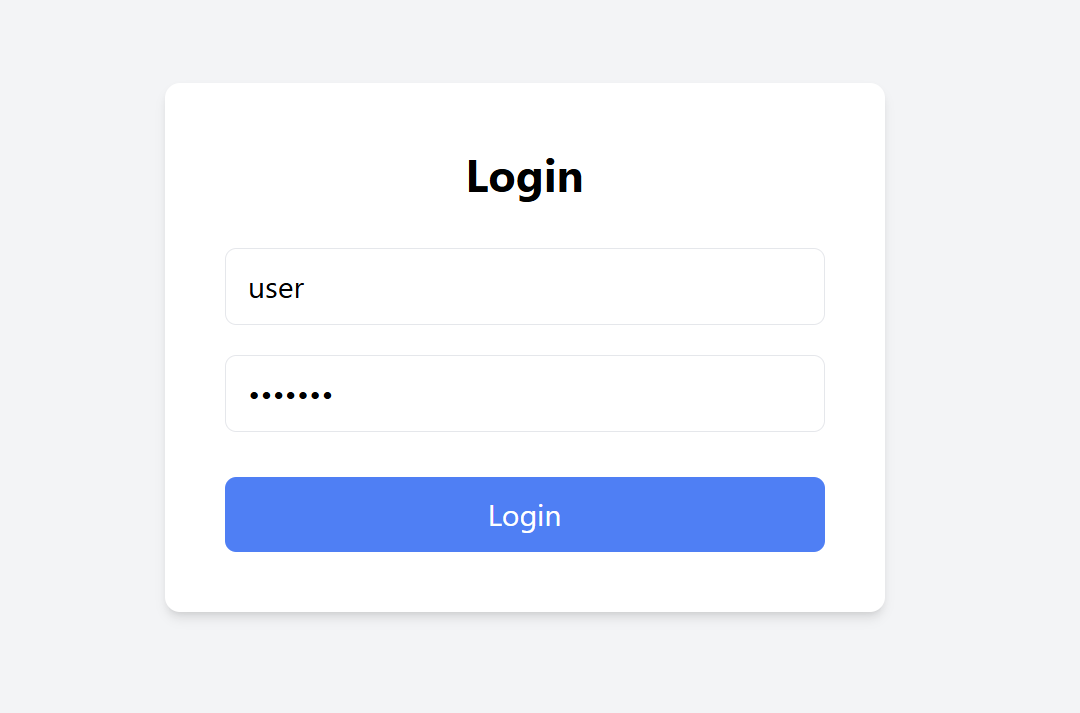
1. **Task Description**

Develop a role-based authorization system to control access to specific routes or components. The system should have three roles: **Admin**, **Editor**, and **User**.

1. **Backend**: Implement APIs for login and role-based access control. Validate user credentials, assign roles, and restrict access to unauthorized routes.
2. **Frontend**: Create a login page using React (with Vite) and Tailwind CSS. After successful login, redirect users based on their roles to role-specific pages (Admin, Editor, User). Unauthorized access to restricted routes should display an error message or redirect to a "403 Unauthorized" page.
3. **Task Output Screenshot**

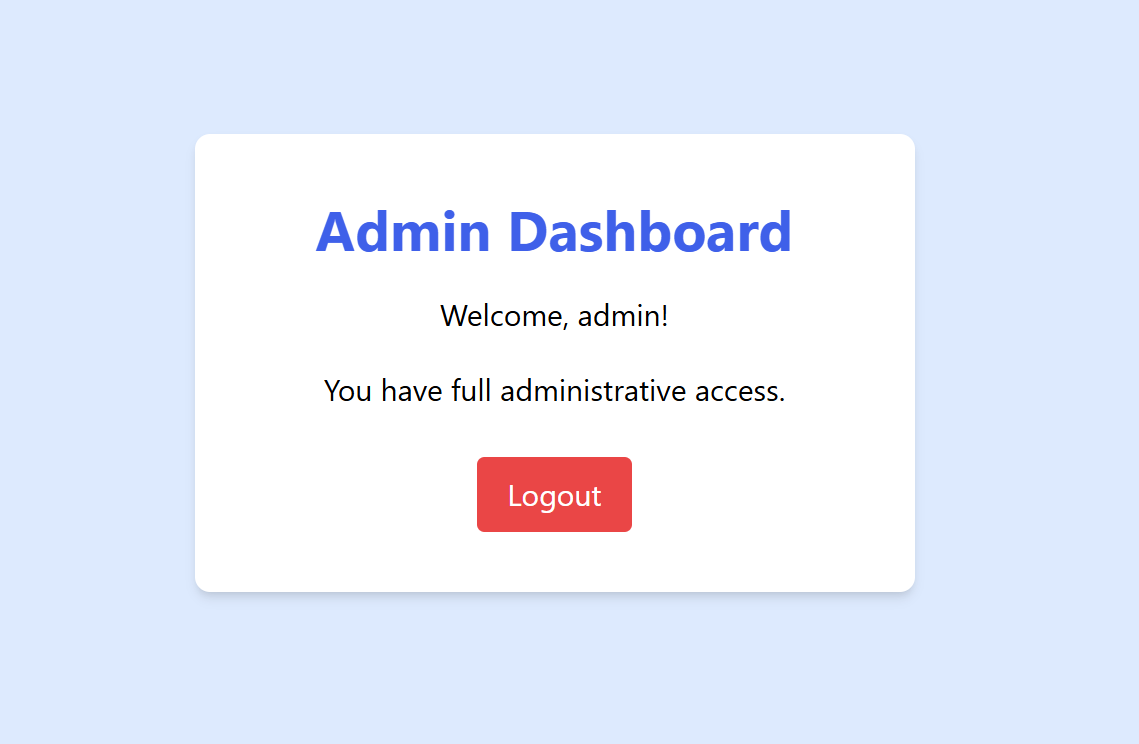
**Login Page: A form where users enter their credentials.**



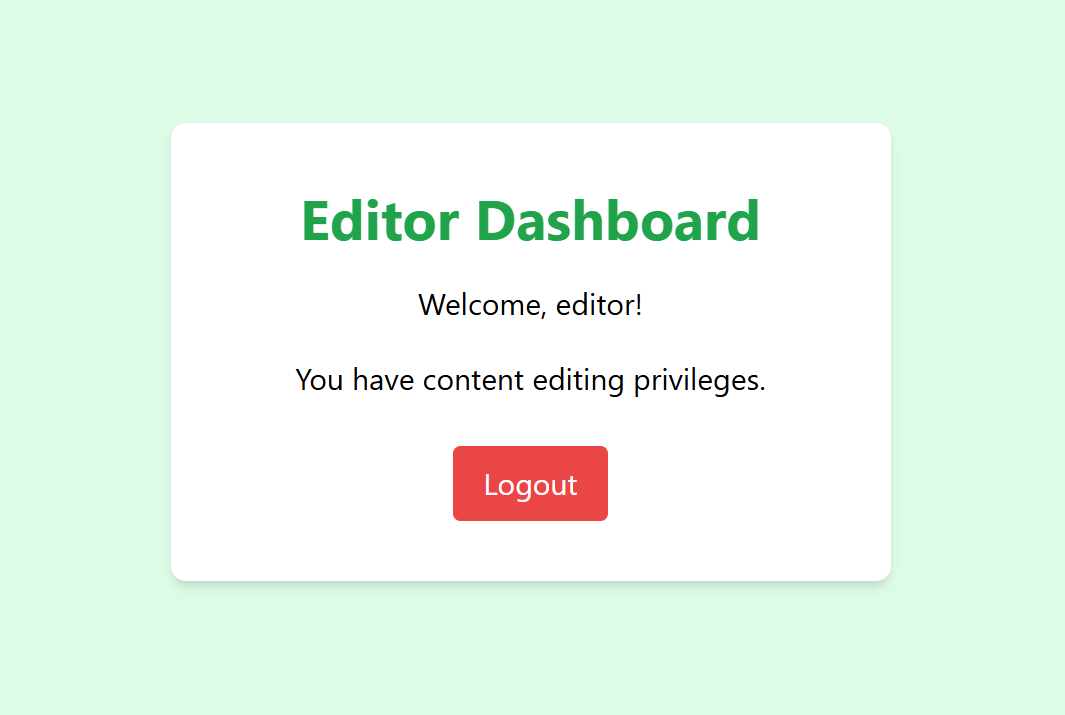
****

**Role-Based Pages:**

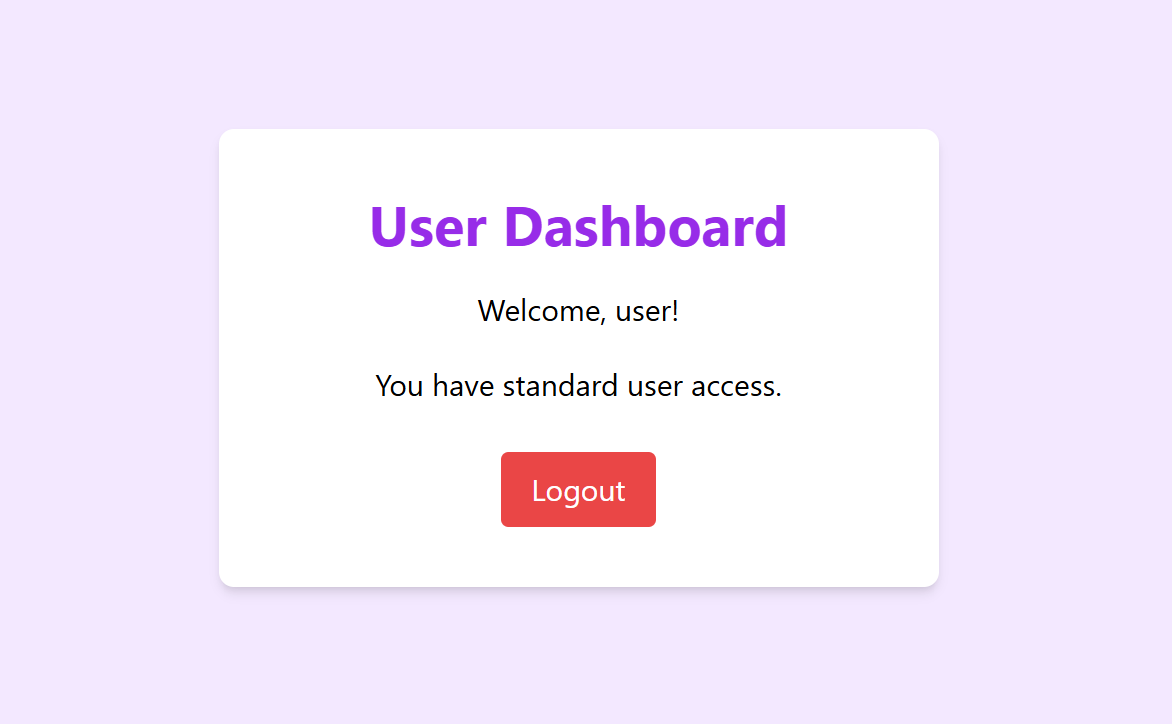
* **Admin Page: "You are an Admin."**



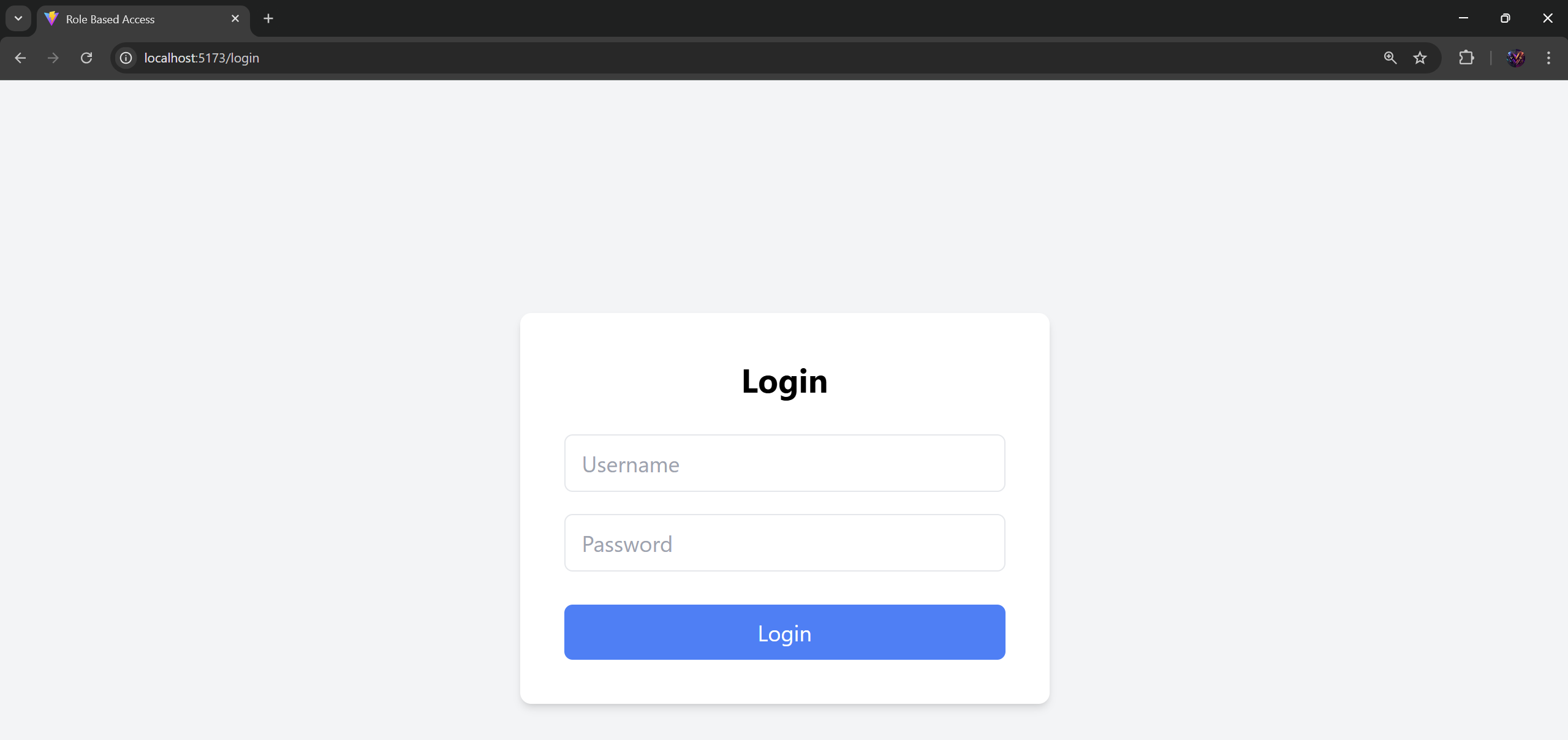
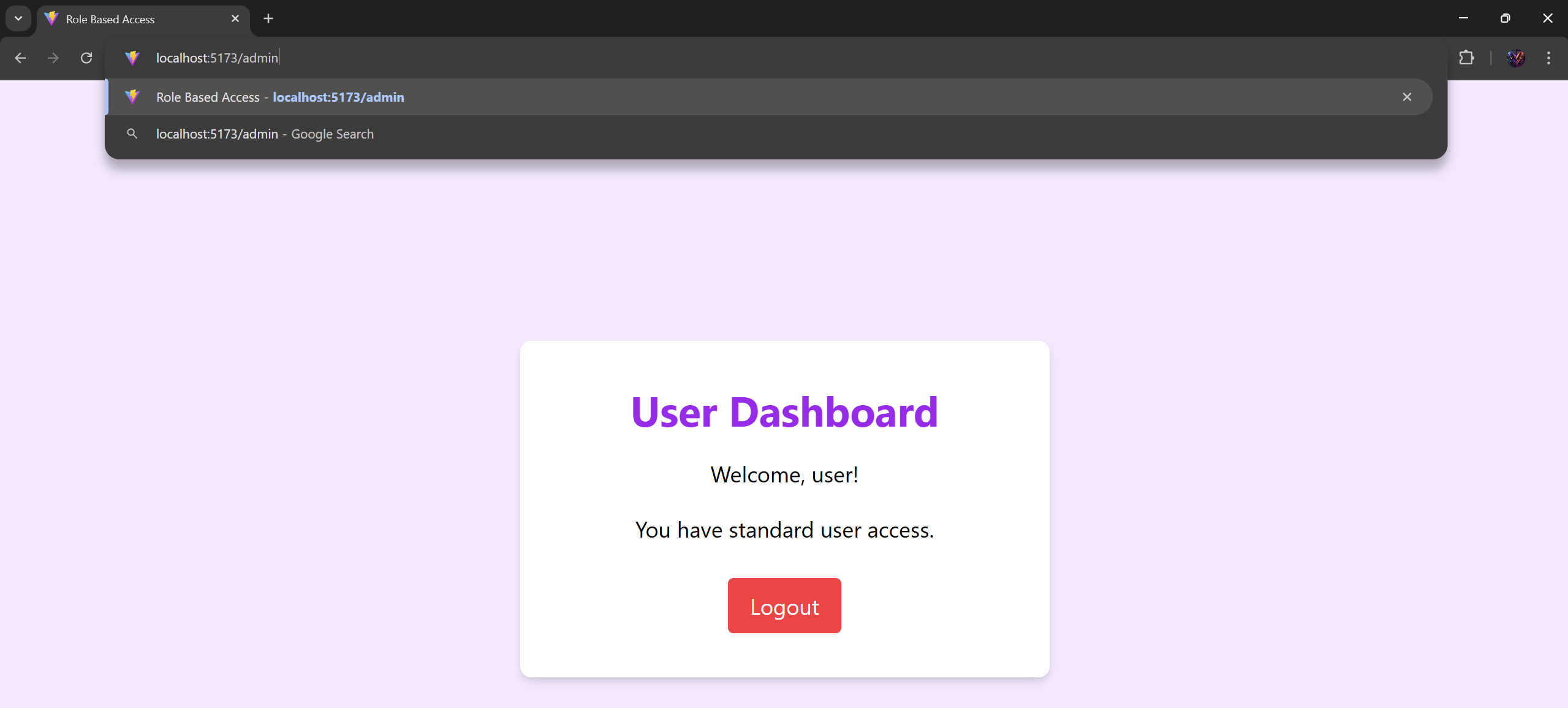
* **Editor Page: "You are an Editor."**



* **User Page: "You are a User."**



**Unauthorized Access: Example of a user trying to access a restricted route (e.g., /admin) and being redirected or shown an error message.**



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

**Frontend Widgets**

1. **TextInput Field: Used for username and password input.**
2. **Button: Submit button for login form.**
3. **Navigation: Route handling and redirection using React Router.**
4. **Error Messages: Conditional rendering for displaying unauthorized access messages.**

**Backend Algorithms**

1. **JWT Token Generation: Authenticate users and generate role-specific JWT tokens using jsonwebtoken.**
2. **Middleware:**
   1. **Role-based middleware to check if the user has access to a specific route.**
   2. **Error handling middleware to handle unauthorized access.**
3. **Database Operations: MongoDB used for storing user credentials and roles.**