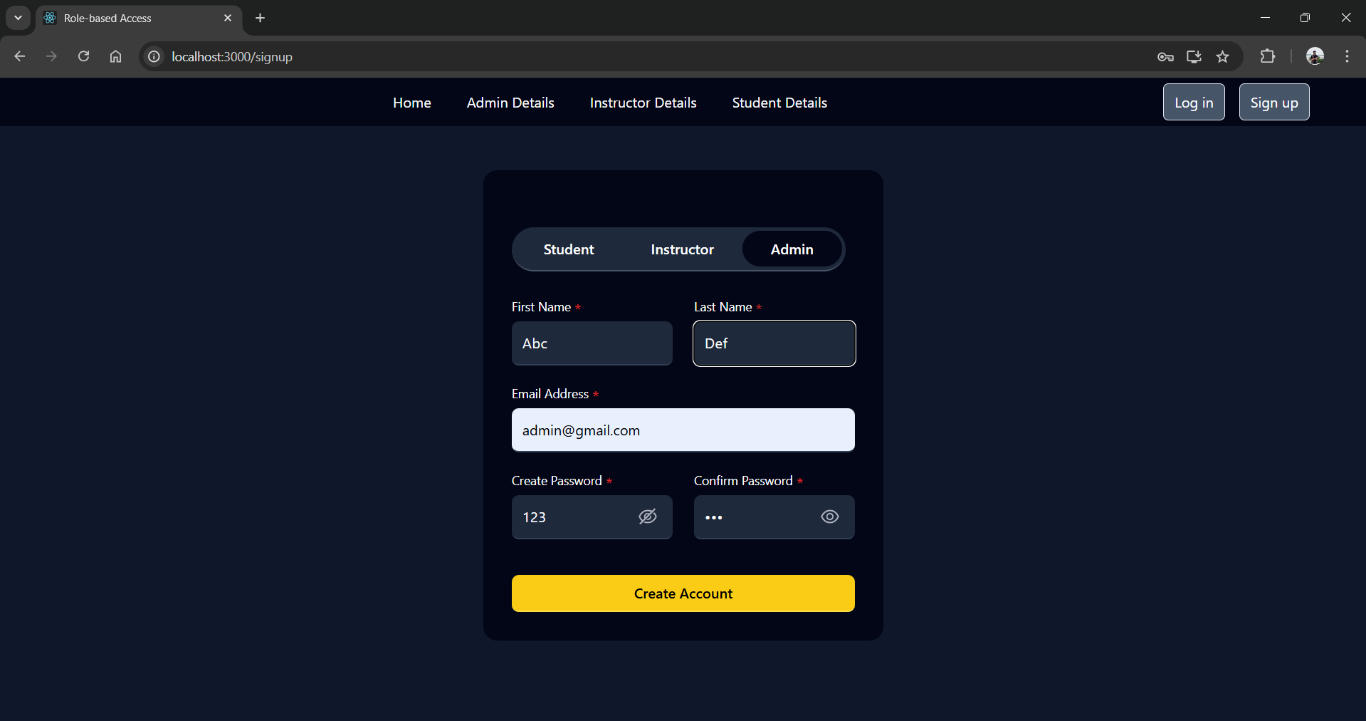
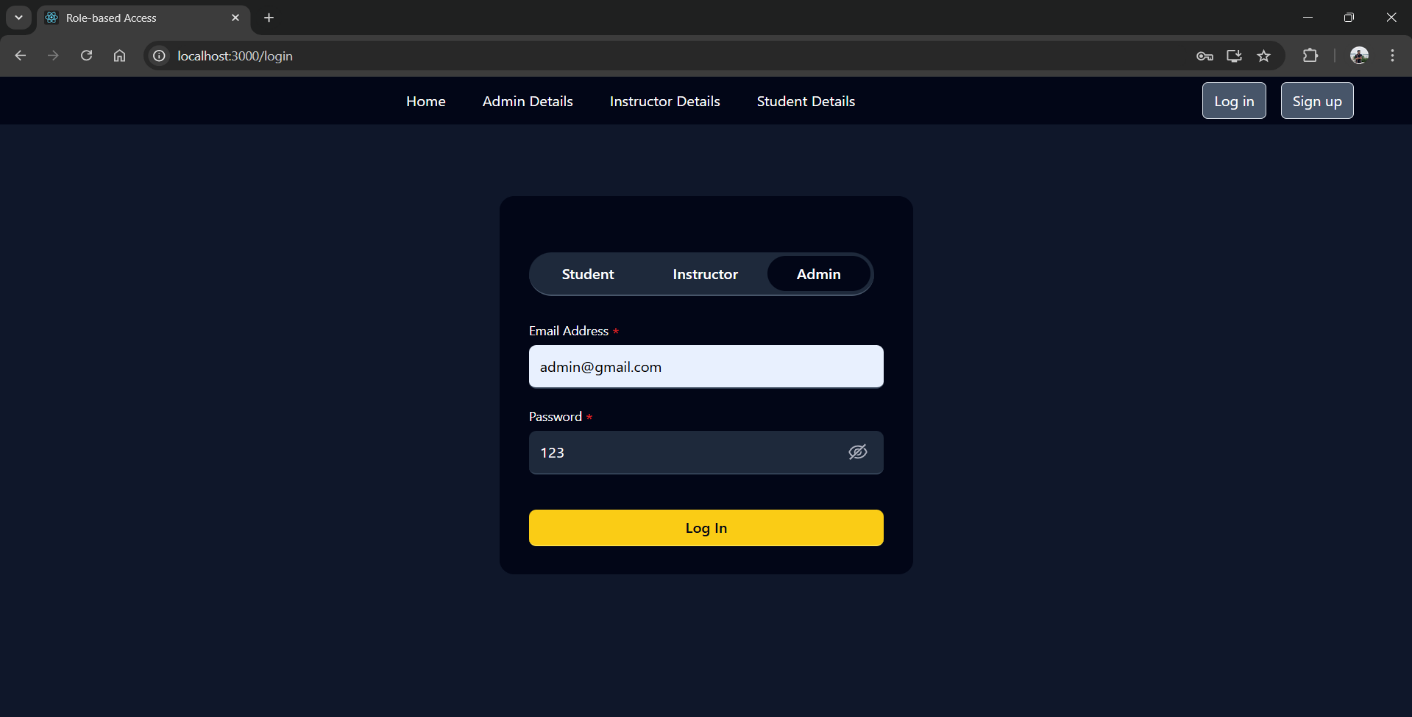
**Task : Role-based Access**

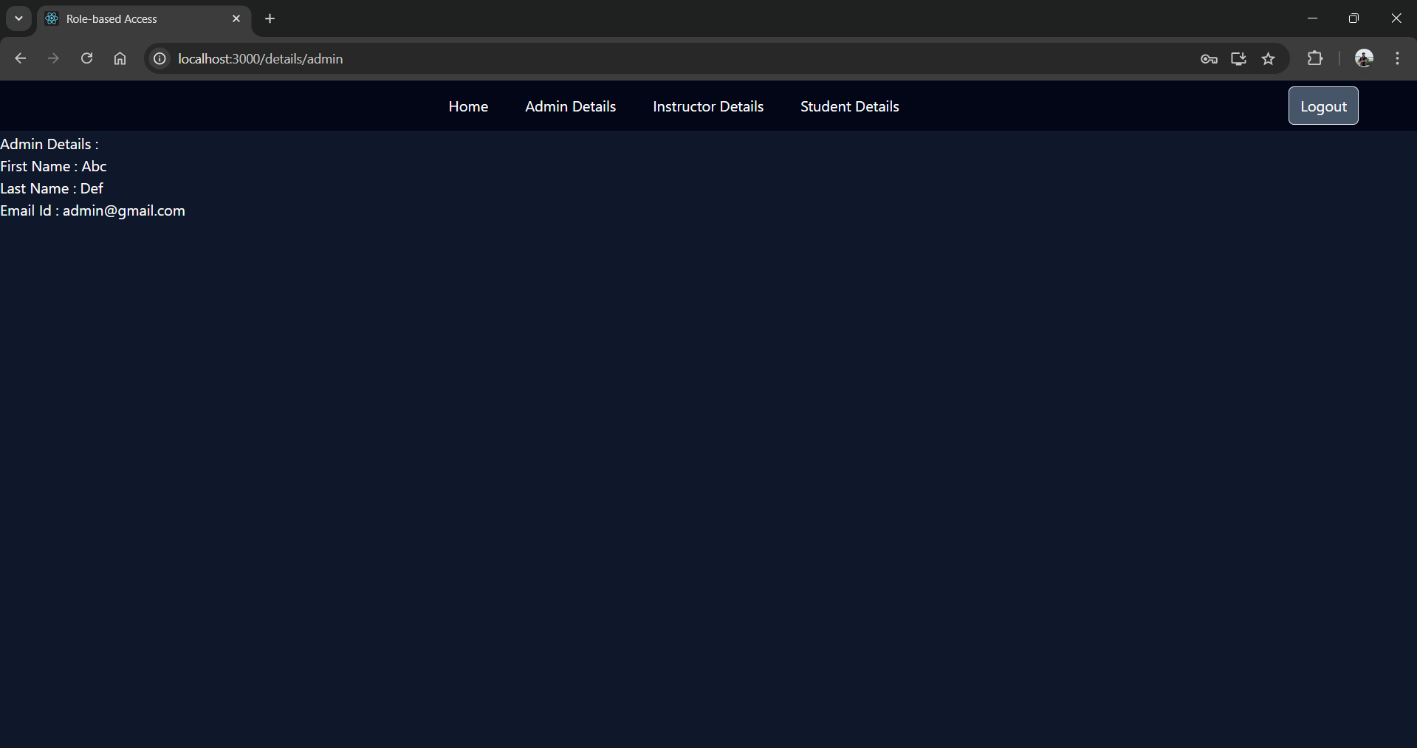
**Task Description :** Develop a role-based access control (RBAC) system that restricts access to specific routes or components based on the user’s role.

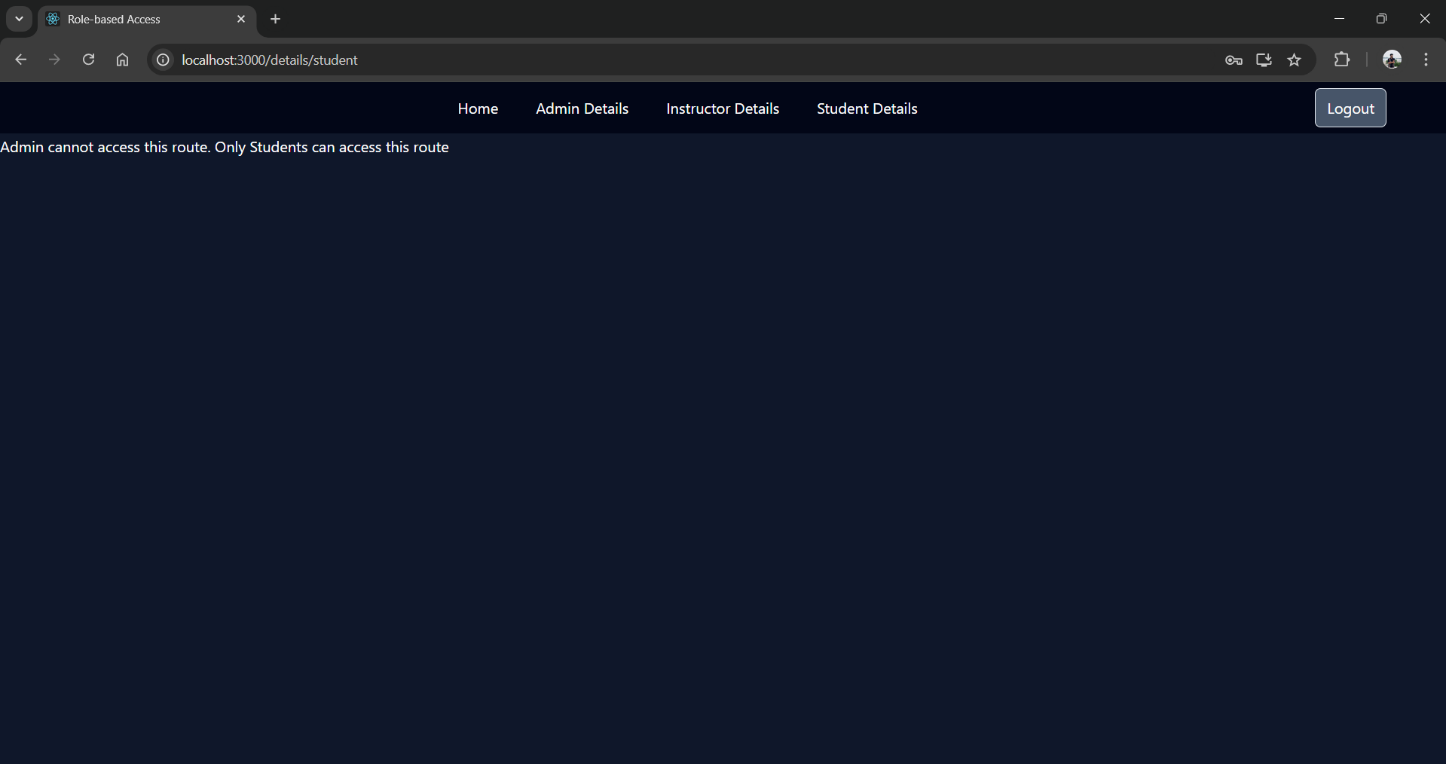
Technology :- React.js, Node.js, Express.js, MongoDB, Redux, Tailwind CSS

**Task Output Screenshot :**









**Widget/Algorithm used in task :**

1. React Router for Navigation :

* Used React Router to handle routing between different components/pages, including public and protected routes.

1. JWT for Authentication :

* Implemented JWT (JSON Web Tokens) for user authentication. After logging in, a token is stored and used to verify the user’s identity and role.

1. Role-Based Access Logic :

* Implemented conditional rendering based on the user’s role(e.g., “Admin”, “Instructor”,”Student”). Protected routes are only accessible to user with the correct role.

1. Private Routes :

* Created a PrivateRoute component that checks if a user is authenticated and has the right role before allowing access to certain routes.

1. Redirects for Unauthorized Users :

* User without the required role are redirected to a login page using the Navigate from React Router.

1. bcrypt for Password Hashing :

* Used bcrypt to securely hash passwords before storing them in the background, ensuring passwords are never stored in plain text.

1. Toasters :

* Integrated react-hot-toast to display toast notifications for events like successful login, unauthorized access attempts or errors.

1. Backend Role Management :

* Role-based access is enforced server-side by checking the role stored in the database.

1. State Management :

* Manages the authentication state and passed it down through components using React Redux.