ROUTE GUARD IN REACT

WHY WE NEED ROUTE GUARDS ?

- **Protect sensitive information:** Route guards prevent unauthorized users from accessing restricted areas of your application that contain sensitive data or functionality.
- **Prevent confusion:** Route guards redirect unauthenticated users to appropriate pages (like login).
- **Role-based access control**: Route guards can be extended to implement role-based access, allowing different user types (e.g., admin, regular user) to access different parts of the application.

STEPS:

1. Implement public and private page component.
2. Create a Context Provider to manage auth in your app.
3. Create a custom hook to get and set auth.
4. Create protected route component which will be HOC.
5. Create Routing to secure public and protected components.

STEP1: PUBLIC & PRIVATE COMPONENT

```
// Home.jsx (Public Component)
const Home = () => {
  return <p>This is public route</p>;
};

export default Home;
```

```
// Profile.jsx (Protected Component)
cons Profile = () => {
  return <div>This is protected route</div>;
}

export default Profile;
```

STEP2: AUTH PROVIDER

```
// AuthProvider.jsx

export const AuthContext = createContext({});

export function AuthProvider({ children }) {
  const [auth, setAuth] = useState({});

  return (
    <AuthContext.Provider value={{ auth, setAuth }}>
      {children}
    </AuthContext.Provider>
  );
}

export default AuthProvider;
```

// Global Context which can manage and pass auth object to all the components in the app

STEP3: CUSTOM HOOK USEAUTHO

```
// useAuth.jsx

import { AuthContext } from "./AuthProvider";

const useAuth = () => {
  return useContext(AuthContext);
};

export default useAuth;

// Custom hook to access auth and setAuth from
anywhere in the app
```

STEP4: GUARD COMPONENT

```
// RequireAuth.jsx

import useAuth from "./useAuth";
import { Navigate, Outlet } from "react-router-dom";

const RequireAuth = () => {
  const { auth } = useAuth();

  return auth?.user ? <Outlet /> : <Navigate to="/" />;
};

export default RequireAuth;

// Parent component to secure child components
```

STEP5: IMPLEMETING ROUTES

```
// App.jsx

import React, { useEffect, useState } from "react";
import { BrowserRouter, Route, Routes } from "react-router-dom";

function App() {
  return (
    <div className="App">
      <BrowserRouter>
        <AuthProvider>
          <Routes>
            <Route path="/" element={<Home />} />
            <Route element={<RequireAuth />}>
              <Route path="/profile" element={<Profile />} />
            </Route>
          </Routes>
        </AuthProvider>
      </BrowserRouter>
    </div>
  ); // Public and protected routes of the of the app
}
```

SAHU-HIMANSHU

THANK YOU

PLEASE SHARE YOUR FEEDBACK

@GITHUB/HSAHU615