Protected Route

- Protected routes are sections of a web application that require users to be authenticated or logged in to access.
- These routes often contain sensitive or private information that should only be accessible to authorised users.
- To implement protected routes, developers use conditional rendering based on the user's authentication status.

Example Implementation:

```
const PrivateRoute = ({ children }) => {
  if (!isLoggedIn) return <Navigate to="/login"
replace={true} />;
  return children;
};
```

In the provided example, a PrivateRoute component is defined to protect specific routes.

React Router Configuration:

```
const router = createBrowserRouter([
  {
    path: "/",
    errorElement: <NotFound />,
    element: <Header />,
    children: [
      // ... other routes
      {
        path: "/profile",
        element: (
          <PrivateRoute>
            <Profile />
          </PrivateRoute>
        )
      },
      // ... other routes
    ]
  }
]);
```

- The createBrowserRouter function is used to set up routes within the React Router.
- The /profile route is protected using the PrivateRoute component.

Usage in App Component:

```
export default function App() {
  const [isLoggedIn, setIsLoggedIn] = useState(true);

  return (
    <div className="App">
      <RouterProvider router={router} />
    </div>
  );
}
```

- The App component renders the app structure and provides the router configuration using RouterProvider.

PrivateRoute Functionality:

- The PrivateRoute component checks if the user is logged in (isLoggedIn state).
- If the user is not logged in, it uses the Navigate component to redirect to the login page (/login).
- If the user is logged in, it renders the children components.

Summary:

- Protected routes, as demonstrated in this example, ensure that certain sections of the app are only accessible to authenticated users.
- The PrivateRoute component serves as a gatekeeper, controlling access based on the user's authentication status.
- React Router provides a powerful mechanism for defining and navigating between different parts of the application, including protected routes.