Day 13: React Router - In-Depth Guide 🚀

Today, you'll **master React Router**, which allows seamless navigation between pages in a **single-page application (SPA)** without full-page reloads.

1️⃣ What is React Router?

React Router is a **client-side routing** library for React that:
✅ **Manages page navigation** without reloading the page.
✅ **Supports URL parameters**, query strings, and nested routing.
✅ **Enhances user experience** with a fast and dynamic interface.
✅ **Mimics multi-page behavior** in a single-page React app.

2️⃣ Installing React Router

To use React Router, install the package:

```
npm install react-router-dom
```

✅ **`react-router-dom` is specifically for web applications**

3️⃣ Setting Up Basic Routing

How Routing Works in React

1. **Wrap the entire application in `<BrowserRouter>`**
2. **Define routes using `<Routes>` and `<Route>`**
3. **Use `<Link>` for navigation instead of `<a>`**

Example: Basic Routing

```
import { BrowserRouter as Router, Routes, Route, Link } from "react-router-dom";

function Home() {
  return <h2>Home Page</h2>;
}

function About() {
  return <h2>About Page</h2>;
}

function App() {
  return (
    <Router>
      <nav>
```

```
        <Link to="/">Home</Link>
        <Link to="/about">About</Link>
      </nav>

      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
      </Routes>
    </Router>
  );
}

export default App;
```

How This Works:

✅ `<BrowserRouter>` enables routing.
✅ `<Routes>` groups multiple `<Route>` components.
✅ `<Route path="/" element={<Home />} />` renders the **Home** component at `/`.
✅ `<Link>` replaces `<a>` to navigate **without refreshing the page**.

4️⃣ Navigating Between Pages

Apart from `<Link>`, you can also navigate programmatically using `useNavigate`.

Example: Navigating with useNavigate

```
import { useNavigate } from "react-router-dom";

function Home() {
  const navigate = useNavigate();

  return (
    <div>
      <h2>Home Page</h2>
      <button onClick={() => navigate("/about")}>Go to About</button>
    </div>
  );
}

export default Home;
```

How This Works:

✅ `useNavigate()` gives access to the **navigate function**.
✅ Calling `navigate("/about")` **redirects the user dynamically**.

5️⃣ Dynamic Routes & URL Parameters

React Router supports **dynamic parameters** in URLs using `:paramName`.

Example: User Profile Route

```
import { useParams } from "react-router-dom";

function UserProfile() {
  const { username } = useParams();
  return <h2>Profile of {username}</h2>;
}
```

Define Route with a Parameter

```
<Route path="/user/:username" element={<UserProfile />} />
```

Navigate to a Dynamic Route

```
<Link to="/user/Danish">Go to Danish's Profile</Link>
```

How This Works:

✅ `useParams()` extracts `:username` from the URL.
✅ Visiting `/user/Danish` will display **"Profile of Danish"**.
✅ Works for **any username**, e.g., `/user/John`.

6️⃣ Nested Routes (Child Routes)

React Router allows **nested components** using `<Outlet>`.

Example: Dashboard with Nested Pages

```
import { Routes, Route, Outlet, Link } from "react-router-dom";

function Dashboard() {
  return (
    <div>
      <h2>Dashboard</h2>
      <nav>
        <Link to="settings">Settings</Link>
        <Link to="profile">Profile</Link>
      </nav>
      <Outlet />
    </div>
  );
}

function Settings() {
  return <h3>Settings Page</h3>;
}

function Profile() {
  return <h3>Profile Page</h3>;
}

function App() {
  return (
    <Routes>
      <Route path="/dashboard" element={<Dashboard />}>
        <Route path="settings" element={<Settings />} />
        <Route path="profile" element={<Profile />} />
      </Route>
    </Routes>
  );
}

export default App;
```

How This Works:

✅ `<Outlet>` acts as a placeholder for nested routes.
✅ `/dashboard/settings` loads **Dashboard + Settings Page**.
✅ `/dashboard/profile` loads **Dashboard + Profile Page**.

7️⃣ Redirecting Users (`Navigate` Component)

To **redirect users** automatically, use the `<Navigate>` component.

Example: Redirect to Home

```
import { Navigate } from "react-router-dom";

function NotFound() {
  return <Navigate to="/" />;
}
```

✅ If the user visits a **non-existent route**, they are **redirected to the home page**.

8️⃣ Protected Routes (Authentication)

To **restrict access** to certain pages, create a **Protected Route component**.

Example: Protecting the Dashboard

```
import { Navigate } from "react-router-dom";

function ProtectedRoute({ children }) {
  const isAuthenticated = false; // Replace with actual authentication logic
  return isAuthenticated ? children : <Navigate to="/" />;
}

function Dashboard() {
  return <h2>Dashboard (Protected)</h2>;
}

function App() {
  return (
    <Routes>
      <Route
```

```
        path="/dashboard"
        element={
          <ProtectedRoute>
            <Dashboard />
          </ProtectedRoute>
        }
      />
    </Routes>
  );
}

export default App;
```

How This Works:

✅ `isAuthenticated` determines if the user is logged in.
✅ If **not logged in**, user is **redirected to `/`**.
✅ If **logged in**, they can **access the dashboard**.

9️⃣ Handling 404 Pages (Not Found Routes)

To handle **non-existing routes**, create a 404 page.

Example: Displaying a 404 Page

```
function NotFound() {
  return <h2>404 - Page Not Found</h2>;
}

function App() {
  return (
    <Routes>
      <Route path="/" element={<Home />} />
      <Route path="/about" element={<About />} />
      <Route path="*" element={<NotFound />} />
    </Routes>
  );
}
```

✅ The * path matches **any undefined route.**
✅ Visiting `/random-page` will show **"404 - Page Not Found"**.

🔟 Summary of Day 13

✔ **React Router enables navigation in SPAs**
✔ **Basic routing with `<Routes>` and `<Route>`**
✔ **Navigation using `<Link>` and `useNavigate()`**
✔ **Dynamic routes (`useParams`) extract URL parameters**
✔ **Nested routes with `<Outlet>` structure pages better**
✔ **Protected routes restrict access based on authentication**

✔ **`<Navigate>` is used for automatic redirection**
✔ **404 pages handle invalid routes gracefully**

🚀 Next Step: Day 14 - API Integration in React