**OBJECTIVES**

**1. Need and Benefits of React Router**

**Need:**  
React is a library for building single-page applications (SPAs). In SPAs, navigation between views should happen without full page reloads. React Router enables this by synchronizing the UI with the browser's URL, allowing users to bookmark pages, use the back/forward buttons, and share links—just like in traditional multi-page applications.

**Benefits:**

* **Client-Side Routing:** No full page reloads, leading to faster transitions.
* **Dynamic Routing:** Load components based on the URL.
* **Nested Routes:** Organize routes hierarchically for complex layouts.
* **History Management:** Supports browser history navigation (back/forward).
* **URL Parameters:** Pass data via URLs (e.g., /trainers/123).
* **SEO-Friendly:** Enables server-side rendering compatibility.

**2. Components in React Router**

React Router provides several key components:

* <BrowserRouter>  
  Wraps the app and provides routing functionality using HTML5 history API.
* <Routes>  
  Container for defining route mappings (replaces <Switch> in v6).
* <Route>  
  Defines a route path and the component to render. Example:

jsx

<Route path="/trainers" element={<TrainersList />} />

* <Link>  
  Replaces <a> tags for navigation without page reloads. Example:

jsx

<Link to="/trainers">Trainers</Link>

* <Navigate>  
  Programmatically redirects users (e.g., after login).
* <Outlet>  
  Renders child routes in nested layouts.

**3. Types of Router Components**

React Router supports different types of routers based on the environment:

* BrowserRouter  
  For web apps with clean URLs (e.g., /trainers). Uses history.pushState.
* HashRouter  
  For legacy browsers or static hosting. Uses URLs with # (e.g., /#/trainers).
* MemoryRouter  
  For testing or non-browser environments (e.g., React Native).
* StaticRouter  
  For server-side rendering (SSR).

**4. Parameter Passing via URL**

React Router allows passing parameters in URLs:

* **Dynamic Segments**  
  Define a route with a parameter (e.g., id):

jsx

<Route path="/trainers/:id" element={<TrainerDetail />} />

* **Accessing Parameters**  
  Use the useParams hook in the target component:

jsx

import { useParams } from 'react-router-dom';

function TrainerDetail() {

const { id } = useParams();

// Fetch trainer data using `id`

}

* **Query Parameters**  
  For optional parameters (e.g., ?filter=active), use useSearchParams:

jsx

const [searchParams] = useSearchParams();

const filter = searchParams.get('filter');