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

**Why React Router is Needed:**

* React is a **Single-Page Application (SPA)** library, meaning it loads one HTML page and dynamically updates the UI.
* By default, React doesn’t support multiple pages or navigation between views.
* **React Router** allows navigation between different “pages” (components) **without refreshing the page**, simulating multi-page behavior in an SPA.

**Benefits of React Router:**

* Enables client-side routing in React apps.
* Improves user experience with fast, seamless navigation.
* Allows for **nested routing** and **dynamic URLs**.
* Supports **route-based rendering** (renders only what matches the route).
* Integrates well with browser history, enabling back/forward navigation.

**2. Identify the Components in React Router**

React Router provides several core components to manage navigation and routing:

| **Component** | **Purpose** |
| --- | --- |
| <BrowserRouter> | Wraps the app and enables routing using browser’s history API |
| <Routes> | Wraps multiple <Route> components |
| <Route> | Defines the path and the component to render |
| <Link> | Navigation link that prevents full-page reload |
| <NavLink> | Like <Link> but can apply an "active" style automatically |
| useParams() | Hook to access URL parameters |
| useNavigate() | Hook to programmatically navigate to another route |

**3. List the Types of Router Components**

React Router supports a few types of routers, depending on the environment:

| **Router Component** | **Use Case** |
| --- | --- |
| **<BrowserRouter>** | Commonly used in modern web apps; uses browser history |
| **<HashRouter>** | Uses hash (#) in URLs; useful when server doesn’t support pushState |
| **<MemoryRouter>** | Keeps URL state in memory; used in testing or non-browser environments |
| **<StaticRouter>** | Used for server-side rendering (SSR) |

In most client-side React apps, **<BrowserRouter>** is the standard choice.

**4. Parameter Passing via URL**

React Router allows passing **dynamic parameters** through the URL.

**Example:**

<Route path="/user/:id" element={<User />} />

If the URL is /user/42, React will render the User component.

**Inside the User component:**

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

function User() {

const { id } = useParams();

return <h2>User ID: {id}</h2>;

}

* :id is a **route parameter**
* useParams() lets you access it inside the component

You can also pass multiple parameters like:

<Route path="/product/:category/:productId" element={<Product />} />

And access them using:

const { category, productId } = useParams();