# React Router Theory & My Academy Trainers App Code

## Explain the need and benefits of React Router

React is a library for building user interfaces. However, it does not include built-in support for routing. In traditional websites, each page load triggers a full request to the server. In contrast, React-based Single Page Applications (SPAs) dynamically update the content on the page without refreshing the browser.  
  
To handle navigation in SPAs, React Router is needed. It is a third-party library that enables routing between different views or pages in a React application.  
  
Benefits:  
- Client-side Navigation: Allows navigation without reloading the page.  
- Dynamic Routing: Routes can change based on the user’s input or logic.  
- Bookmarkable URLs: Users can bookmark specific views or data points.  
- History Support: Supports browser navigation (back/forward buttons).  
- Modular Design: Helps in separating different views into components.

## Identify the Components in React Router

React Router consists of several components that help define and control navigation:  
  
- BrowserRouter: Enables routing in a web app using HTML5 history API.  
- Routes: A container for one or more Route definitions.  
- Route: Maps a URL path to a specific React component.  
- Link: Renders a hyperlink to navigate between routes without reloading the page.  
- NavLink: Similar to Link, but allows styling for active routes.  
- Navigate: Used for programmatic redirection.  
- useParams(): Retrieves dynamic values from the URL.  
- useNavigate(): Used to navigate programmatically within the application.

## List the types of Router Components

React Router provides different router components based on the platform or use case:  
  
- BrowserRouter: Uses the HTML5 history API. Suitable for modern web apps.  
- HashRouter: Uses the URL hash (#) to keep UI in sync. Useful for static hosting.  
- MemoryRouter: Keeps the navigation state in memory. Used in testing.  
- StaticRouter: Used for server-side rendering. It does not change the URL in the browser.

## Parameter passing via URL (Theoretical)

Parameter passing allows you to send values via the URL. In React Router, this is done using dynamic segments.  
  
Example: /trainer/T101  
  
Define route: <Route path="/trainer/:id" element={<TrainerDetails />} />  
Access param: const { id } = useParams();  
  
Advantages:  
- Clean and semantic URLs  
- Supports deep linking and bookmarking  
- Useful for displaying or modifying specific records based on the URL

# My Academy Trainer App

## App.js

import React from 'react';  
import { BrowserRouter as Router, Routes, Route, Link } from 'react-router-dom';  
import Home from './Home';  
import TrainersList from './TrainerList';  
import TrainerDetails from './TrainerDetails';  
import trainers from './TrainersMock';  
  
function App() {  
 return (  
 <Router>  
 <div>  
 <h1>Trainers Portal</h1>  
 <nav>  
 <Link to="/">Home</Link> | <Link to="/trainers">Trainers</Link>  
 </nav>  
 <hr />  
 <Routes>  
 <Route path="/" element={<Home />} />  
 <Route path="/trainers" element={<TrainersList trainers={trainers} />} />  
 <Route path="/trainer/:id" element={<TrainerDetails />} />  
 </Routes>  
 </div>  
 </Router>  
 );  
}  
  
export default App;

## Home.js

import React from 'react';  
  
function Home() {  
 return (  
 <div>  
 <h2>Welcome to the Cognizant Academy Trainer Portal</h2>  
 <p>This portal helps you manage trainer details efficiently.</p>  
 </div>  
 );  
}  
  
export default Home;

## Trainer.js

class Trainer {  
 constructor(id, name, email, phone, technology, skills) {  
 this.TrainerId = id;  
 this.Name = name;  
 this.Email = email;  
 this.Phone = phone;  
 this.Technology = technology;  
 this.Skills = skills;  
 }  
}  
  
export default Trainer;

## TrainersMock.js

import Trainer from './Trainer';  
  
const trainers = [  
 new Trainer("T101", "Amit", "amit@example.com", "9876543210", "Java", ["Spring", "Hibernate"]),  
 new Trainer("T102", "Neha", "neha@example.com", "9876543211", "Python", ["Django", "Flask"]),  
 new Trainer("T103", "Rahul", "rahul@example.com", "9876543212", "React", ["Redux", "Hooks"]),  
 new Trainer("T104", "Sara", "sara@example.com", "9876543213", "Data Science", ["Pandas", "NumPy"]),  
];  
  
export default trainers;

## TrainerList.js

import React from 'react';  
import { Link } from 'react-router-dom';  
  
function TrainersList({ trainers }) {  
 return (  
 <div>  
 <h2>Trainers List</h2>  
 <ul>  
 {trainers.map(trainer => (  
 <li key={trainer.TrainerId}>  
 <Link to={`/trainer/${trainer.TrainerId}`}>{trainer.Name}</Link>  
 </li>  
 ))}  
 </ul>  
 </div>  
 );  
}  
  
export default TrainersList;

## TrainerDetails.js

import React from 'react';  
import { useParams } from 'react-router-dom';  
import trainers from './TrainersMock';  
  
function TrainerDetails() {  
 const { id } = useParams();  
 const trainer = trainers.find(t => t.TrainerId === id);  
  
 if (!trainer) return <p>Trainer not found</p>;  
  
 return (  
 <div>  
 <h2>Trainer Details</h2>  
 <p><strong>ID:</strong> {trainer.TrainerId}</p>  
 <p><strong>Name:</strong> {trainer.Name}</p>  
 <p><strong>Email:</strong> {trainer.Email}</p>  
 <p><strong>Phone:</strong> {trainer.Phone}</p>  
 <p><strong>Technology:</strong> {trainer.Technology}</p>  
 <p><strong>Skills:</strong> {trainer.Skills.join(", ")}</p>  
 </div>  
 );  
}  
  
export default TrainerDetails;  
**Output:**







