**Objectives**

* Explain the need and benefits of React Router

React is a Single Page Application (SPA) framework. This means the page doesn't reload when users navigate between views. But SPAs still need a way to simulate multi-page behavior. That’s where React Router comes in — it allows developers to create dynamic routing in React apps.

**Benefits:**

* Enables navigation without full page reloads.
* Supports nested routes and layouts.
* Allows URL parameterization and state-based navigation.
* Provides route protection and redirects.
* Enhances user experience with faster transitions.
* Identify the Components in React Router

|  |  |
| --- | --- |
| Component | Purpose |
| <BrowserRouter> | Wraps the app and enables routing functionality using HTML5 history API |
| <Routes> | A container for all <Route> elements |
| <Route> | Defines a path and its corresponding component |
| <Link> | Creates navigation links without reloading the page |
| <NavLink> | Like <Link>, but allows styling of active links |
| <Outlet> | Used for nested routes (renders child route component) |
| <useParams> | React hook to access URL parameters |
| <useNavigate> | React hook to programmatically navigate |
| <useLocation> | Access current location object |

* List the types of Router Components

|  |  |
| --- | --- |
| Router Component | Description |
| <BrowserRouter> | Most common; uses HTML5 history API for clean URLs |
| <HashRouter> | Uses hash (#) in the URL (e.g., /home#section1) — good for static file hosting |
| <MemoryRouter> | Keeps history in memory — often used for testing or non-browser environments |
| <StaticRouter> | Used for server-side rendering |

* Parameter passing via url

**Example use case: Display user profile by ID**

**Step 1: Define Route with parameter**

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

**Step 2: Create Component and Access Parameter**

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

function User() {

const { id } = useParams();

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

}

**Step 3: Navigate using a Link**

<Link to="/user/42">View User 42</Link>

## **Notes**

Estimated time to complete this lab: **60 minutes.**

Cognizant Academy teams want to maintain a list of trainers along with their expertise in a SPA using React as the technology. You are assigned the task of creating this React app.

The following trainers’ data application will deal.

1. T-ID
2. Name
3. Phone
4. Email
5. Stream
6. Skills
7. Create a new React app using *create-react-app* tool with the as “TrainersApp”
8. Open the application using the VS Code
9. Add a new file called *trainer.js* inside the **src folder** and define a class named as “Trainer” with the following properties
   1. TrainerId
   2. Name
   3. Email
   4. Phone
   5. Technology
   6. Skills

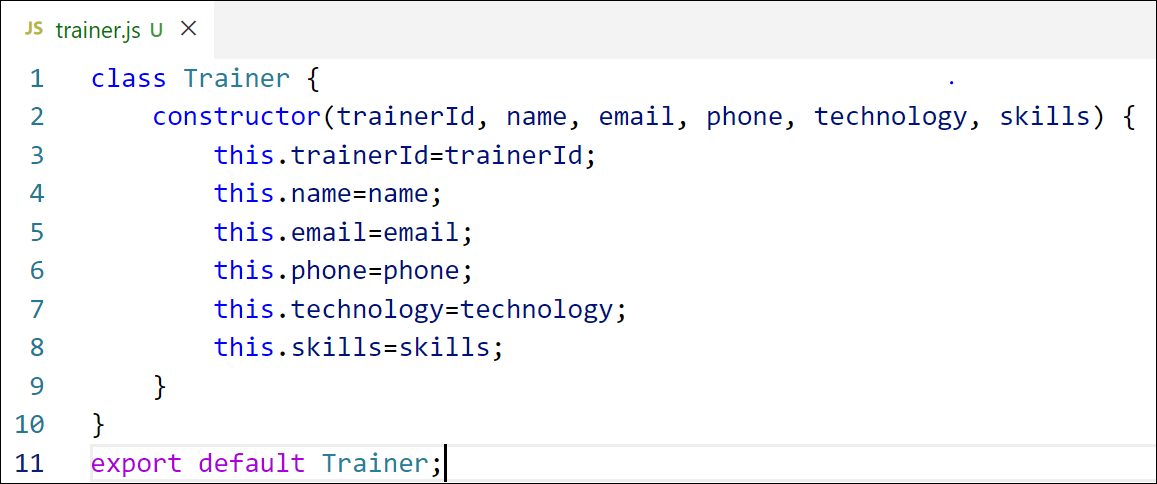


Figure 2: Trainer.js

1. Create a new TrainersMock.js file which will contain the mock trainer data. Refer the following screenshot for mock data

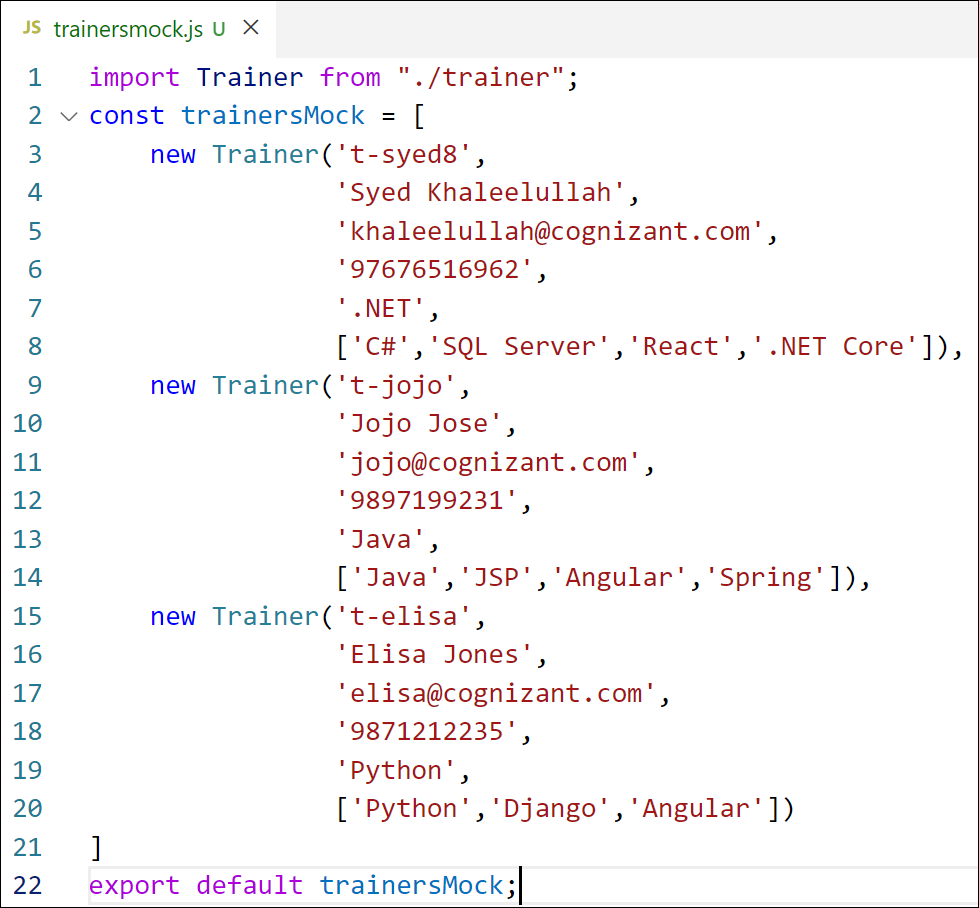


Figure 3: TrainersMock.js

1. Install the support for React router for the dom. Execute the following command.

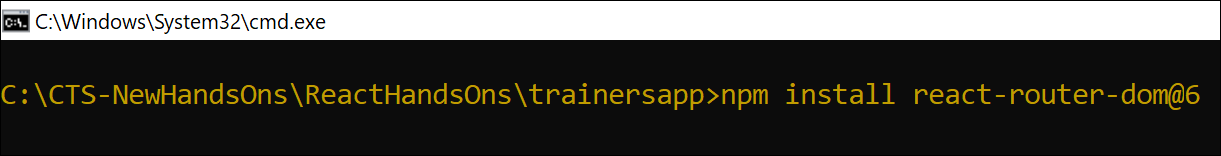


Figure 4: Install React Router

1. Create new component named as **TrainersList** inside *Trainerlist.js* file. The component should accept the trainer’s data as parameter and render it as a list. The list should display names of each trainers which must be clickable like a hyper link. Refer the following screenshot for the component layout.



Figure 5: TrainersList Component

1. Create a new component named as Home inside Home.js which will be responsible for displaying the following



Figure 6: Home Component

1. Modify the App component to add support for routing and defining the navigation links to Home component and TrainersList component. Use BrowserRouter, Routes, Route and Link components from the react-router-dom library.

Define the following URL

1. / - must redirect to home component
2. /trainers – must redirect to trainers list component.

The layout of the page must be similar to the following



Figure 7: App Component

1. Create a new component named **TrainerDetail** in *TrainerDetails.js* file.

The component should retrieve a parameter named id from the URL with the help of “useParams” hook from the React router DOM library.

It should query the mock trainer data using the id and display the trainer details as show in screenshot.

Modify the TrainersList component to add Links to TrainerDetail component while passing the ID. Define a route in App component for the same.

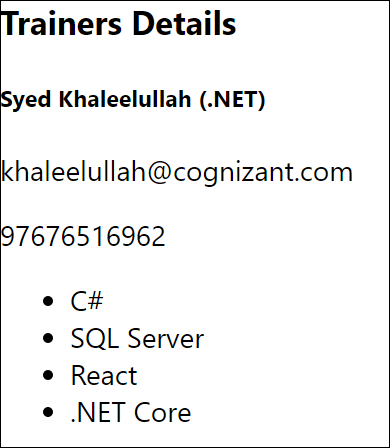


Figure 8: Trainers Detail Component

1. Build and run the application. The complete layout of the application will look as follows.



Figure 9: Home

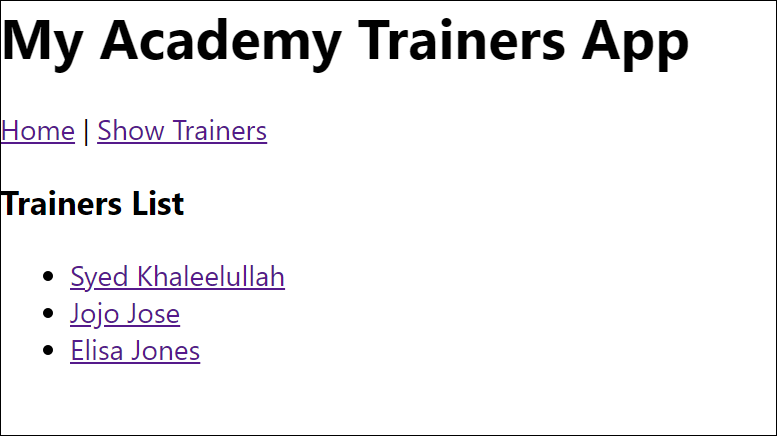


Figure 10: Trainers List

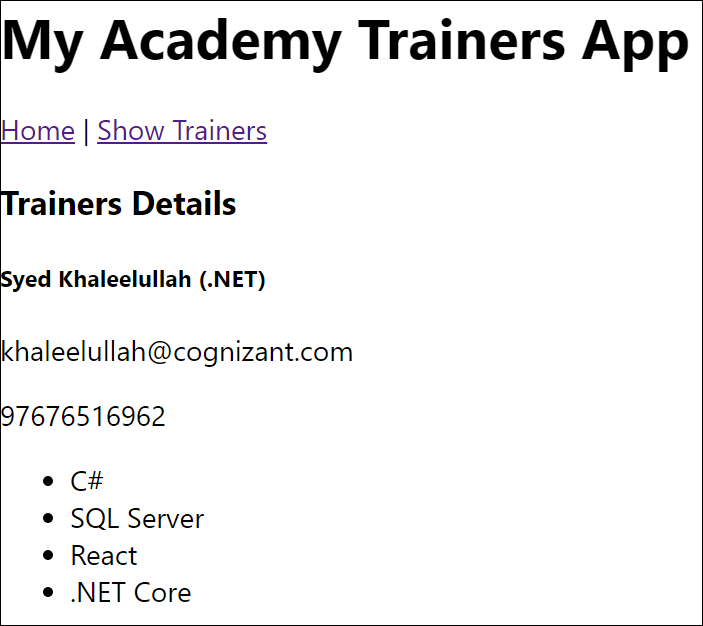


Figure 11: Trainer Details

HandsOn:

