**6**. **ReactJS – HOL**

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

**Need for React Router:**

* React is a Single Page Application (SPA) framework — there's only one HTML page.
* To navigate between different views (like Home, About, Contact), we use React Router instead of reloading pages.

**Benefits:**

* No full page reload – faster and smoother navigation
* Supports URL-based routing (like /about, /contact)
* Easy to create multiple views in a single app
* Helps in dynamic routing with parameters (like /user/123)
* Breaks app into reusable components with routes

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

Some important components provided by React Router:

|  |  |
| --- | --- |
| **Component** | **Purpose** |
| <BrowserRouter> | Wraps the app and enables routing using the browser's URL |
| <Routes> | Holds all the route definitions |
| <Route> | Maps a path (URL) to a component (e.g., /home → HomeComponent) |
| <Link> | Creates a clickable link (like <a>) without page refresh |
| <Navigate> | Used for redirecting to another route programmatically |
| useParams() | Hook to read URL parameters like user ID or product name |

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

React Router offers different types of routers depending on how the app runs:

|  |  |
| --- | --- |
| **Type** | **Description** |
| BrowserRouter | Uses browser history API (ideal for web apps) |
| HashRouter | Uses URL hash (#) for routing (useful when server setup is limited) |
| MemoryRouter | Stores history in memory (used in testing or non-browser environments) |

**4. Parameter Passing via URL**

You can pass values like IDs, names, or categories through the URL.

**Step 1: Define a Route with a parameter**

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

**Step 2: Access the parameter inside the component using useParams()**

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

function User() {

const { id } = useParams(); // 'id' comes from the URL

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

}

**Example URL:**http://localhost:3000/user/101 → It will show: User ID: 101

**React Application : trainersapp**

1. Create a new React app using *create-react-app* tool with the as “TrainersApp”
2. Open the application using the VS Code
3. 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

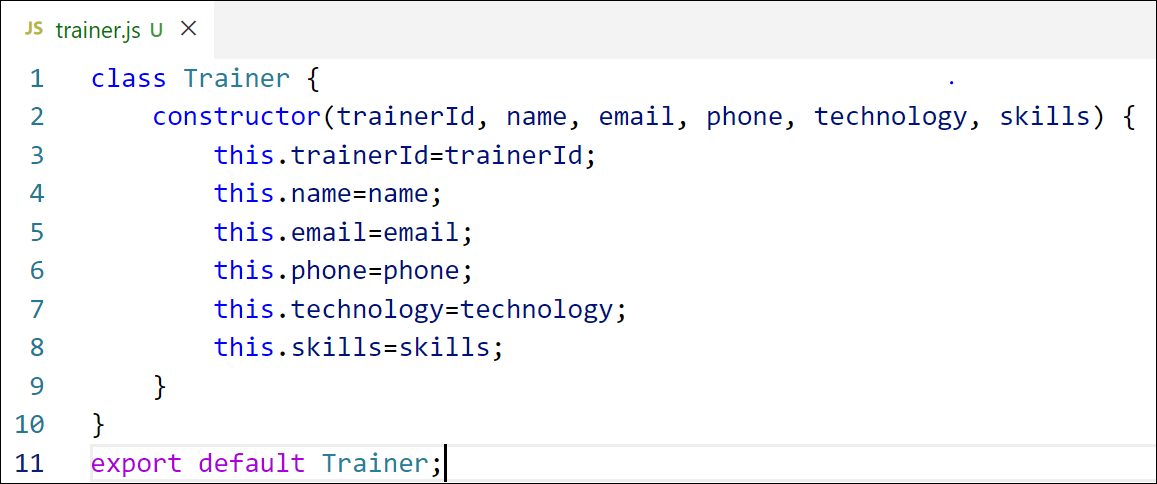


Figure 2: Trainer.js

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

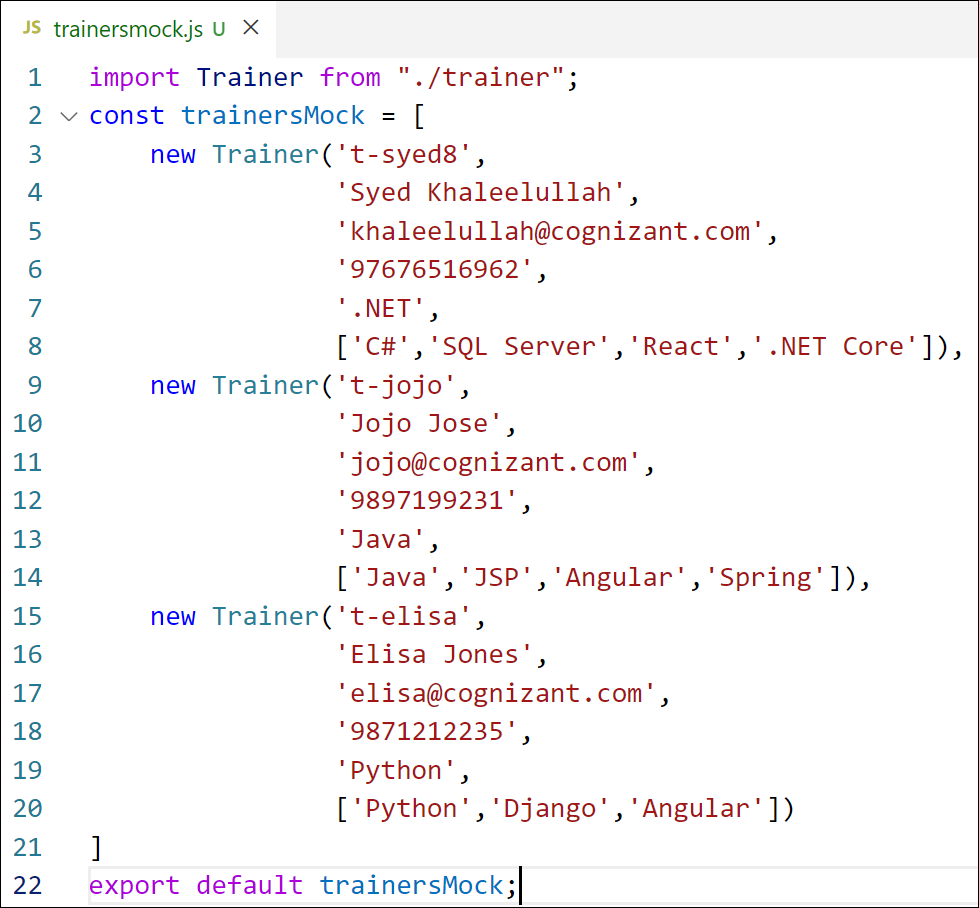


Figure 3: TrainersMock.js

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

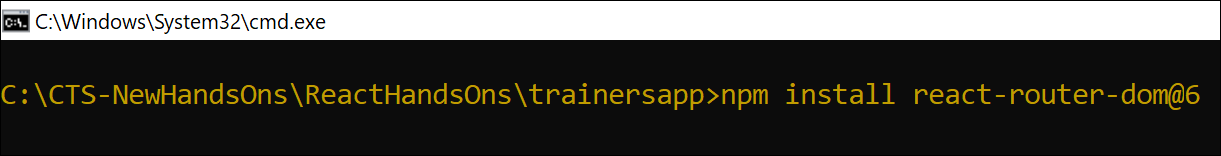


Figure 4: Install React Router

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.
2. Create a new component named as Home inside Home.js which will be responsible for displaying the following
3. 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

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

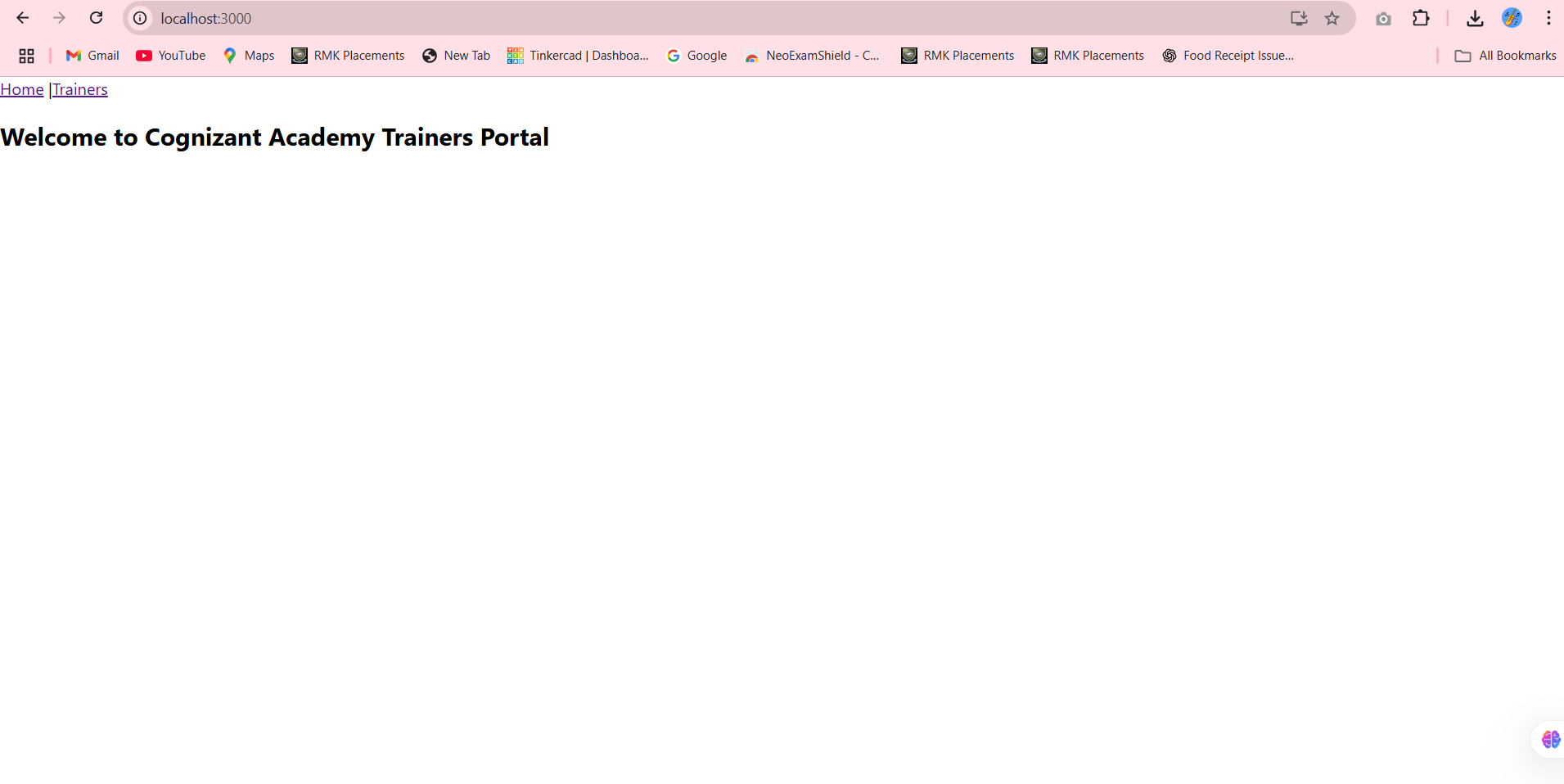
The component should retrieve a parameter named id from the URL with the help of “useParams” hook from the React router DOM library.

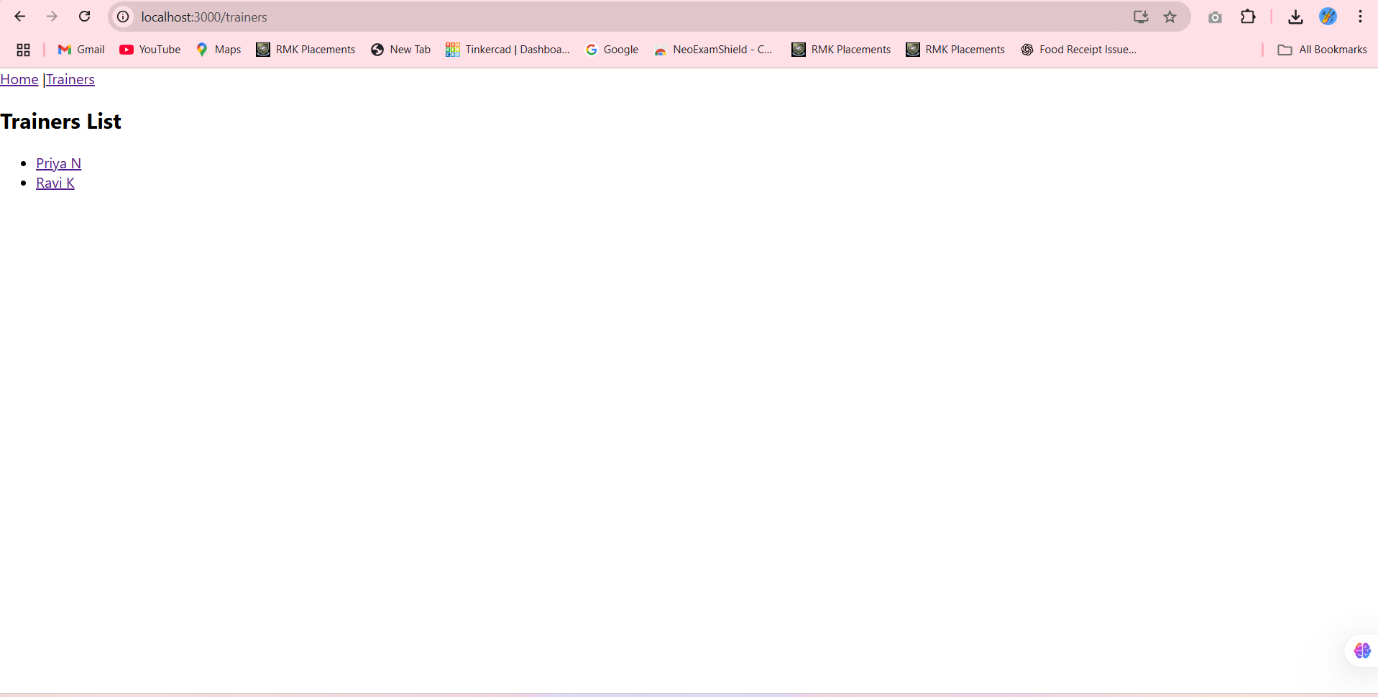
It should query the mock trainer data using the id and display the trainer details as show in screenshot.

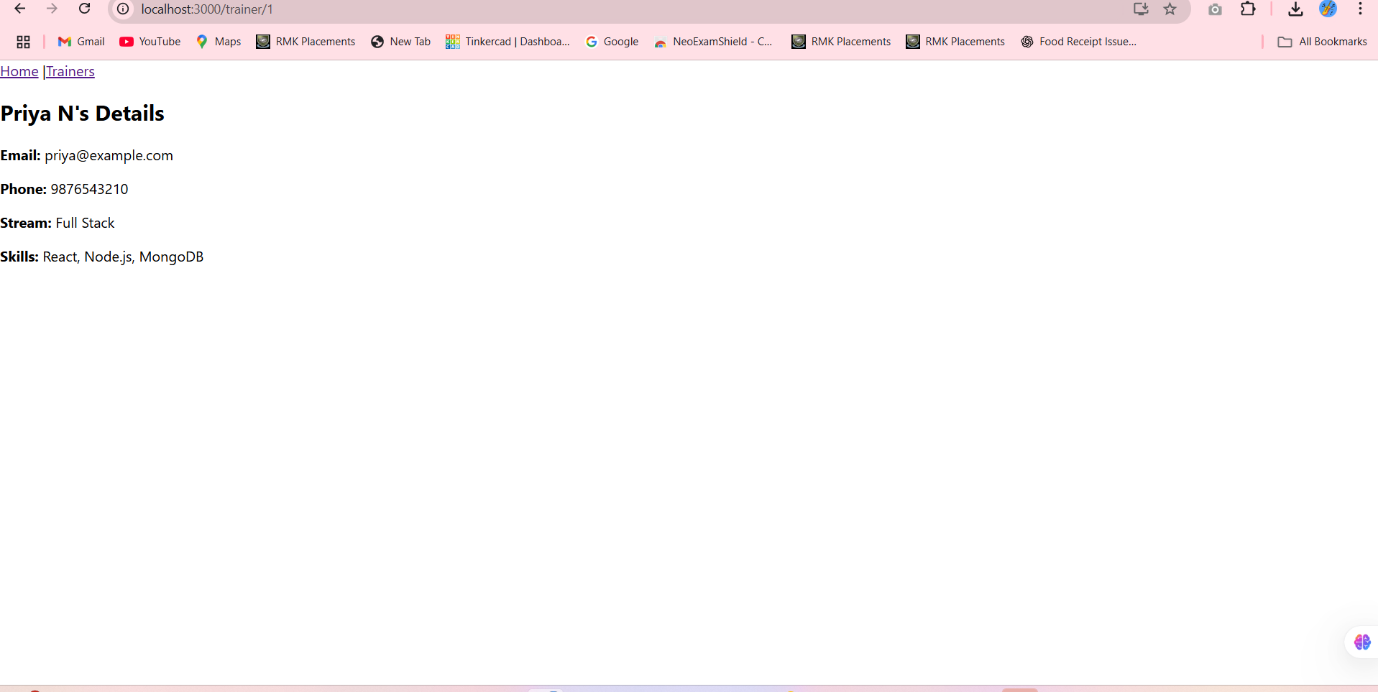
Modify the TrainersList component to add Links to TrainerDetail component while passing the ID. Define a route in App component for the same.

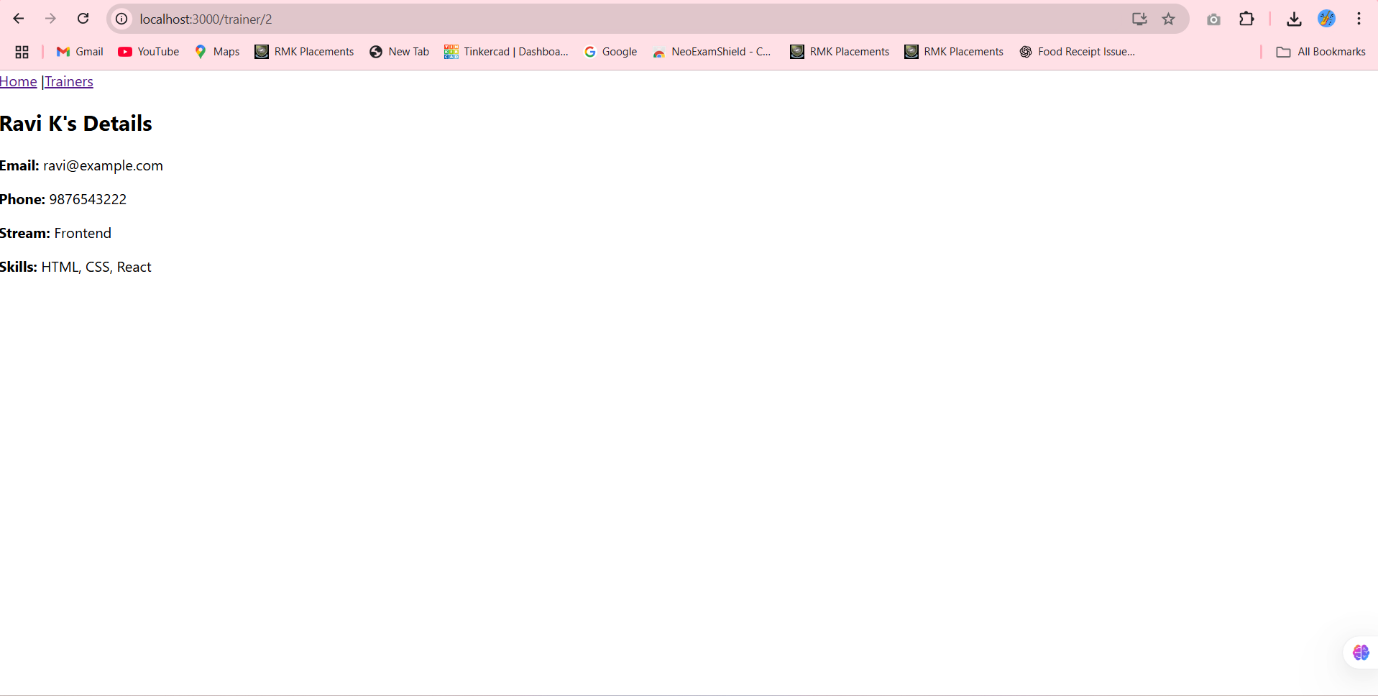
1. Build and run the application. The complete layout of the application will look as follows.

**Final Output:**

****

****

****

****