**WEEK 6 ADDITIONAL HANDS ON**

**6. React JS - HOL**

**Objectives**

* **Explain the need and benefits of React Router**

React Router is a powerful routing library used in React applications to handle navigation between different components or pages without reloading the entire application. Since React is a single-page application (SPA) framework, traditional page reloads are not ideal. React Router enables dynamic routing that updates the URL and renders the correct component without refreshing the page. This improves performance and provides a smoother user experience. Its benefits include easy navigation, nested routing, parameterized routes, route guards, and code-splitting, making it an essential tool for building scalable and interactive React apps.

* **Identify the Components in React Router**

React Router provides several core components that help in defining and managing routes in a React application. The most commonly used components are:

* BrowserRouter: Wraps the entire application and enables routing using the browser's history API.
* Routes: A container for all the route definitions; replaces the older Switch component.
* Route: Defines a path and the component that should be rendered when the path matches.
* Link: Used to navigate between routes without reloading the page, similar to an anchor tag.
* useParams: A hook used to access route parameters from the URL inside functional components.
* Navigate: Used to programmatically redirect users to another route.
* **List the types of Router Components**

React Router provides different types of routers to suit various use cases:

1. BrowserRouter: Uses the HTML5 history API (pushState, popState) to manage routing. Best for modern web apps.
2. HashRouter: Uses the hash portion of the URL (e.g., example.com/#/about) for routing. Useful when server-side configuration is limited.
3. MemoryRouter: Keeps the history of your “URL” in memory (does not read or write to the address bar). It's mainly used for testing or non-browser environments like React Native.  
   Each router type has its own use case depending on deployment and environment.

* **Parameter passing via url**

React Router allows passing parameters via the URL using route parameters. For example, a route path like /user/:id captures id as a route parameter. This allows you to build dynamic routes, such as showing details for a specific user or product based on the id passed in the URL. It's a clean and efficient way to pass data between routes.

In this hands-on lab, you will learn how to:

* Implement a Simple Navigation Menu
* Add Basic Routes (install, configure)
* Use Routes in React Applications

## **Prerequisites**

The following is required to complete this hands-on lab:

* Node.js
* NPM
* Visual Studio Code

## **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

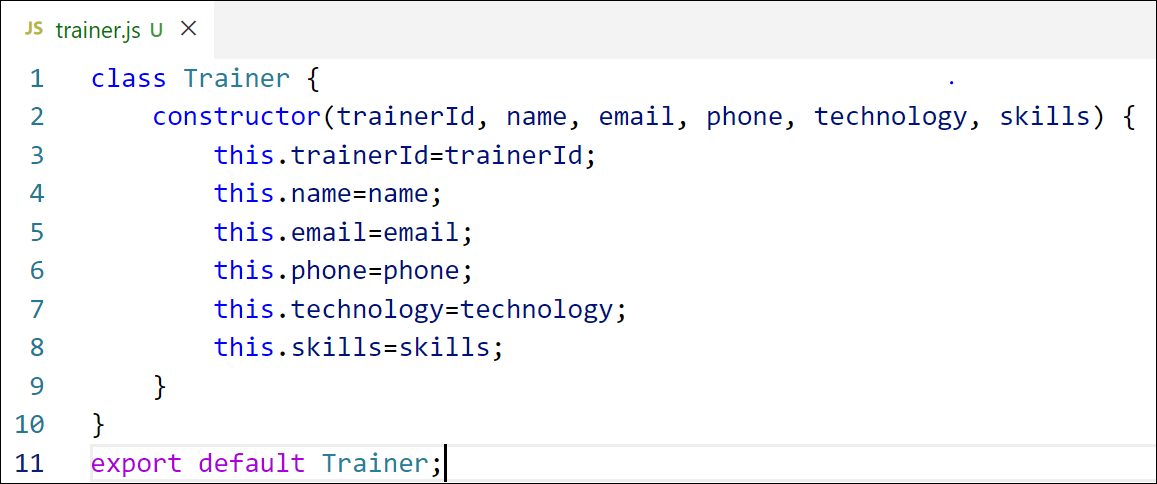


Figure 2: Trainer.js

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

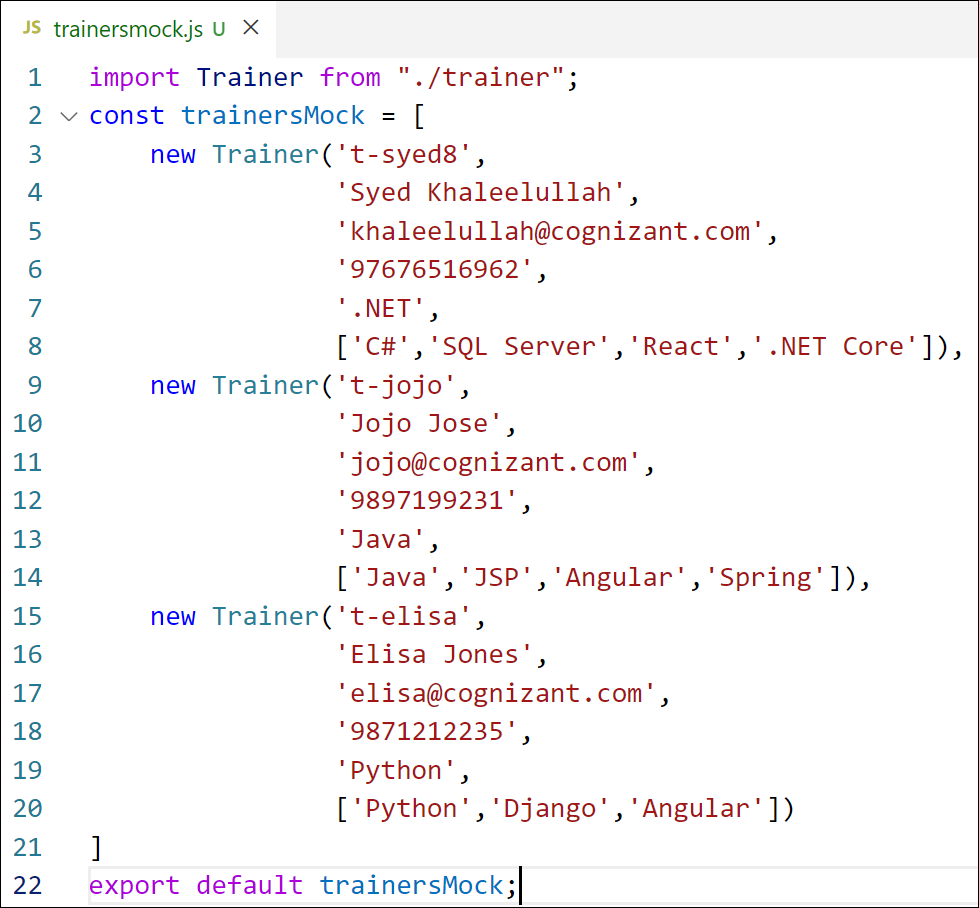


Figure 3: TrainersMock.js

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

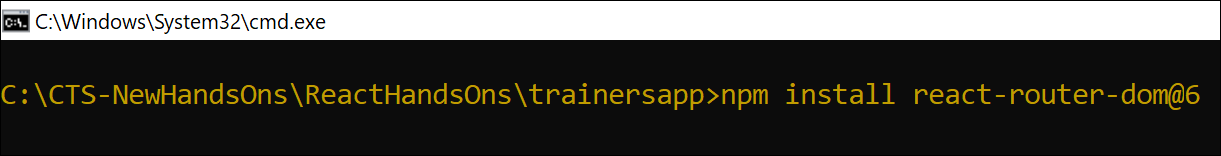


Figure 4: Install React Router

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.

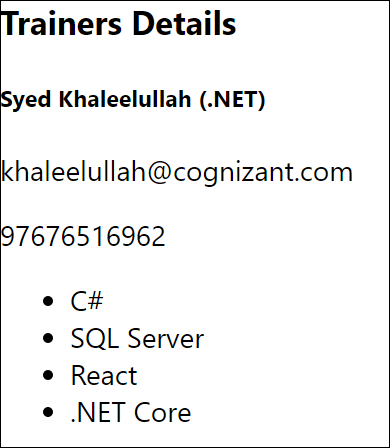


Figure 8: Trainers Detail Component

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



Figure 9: Home

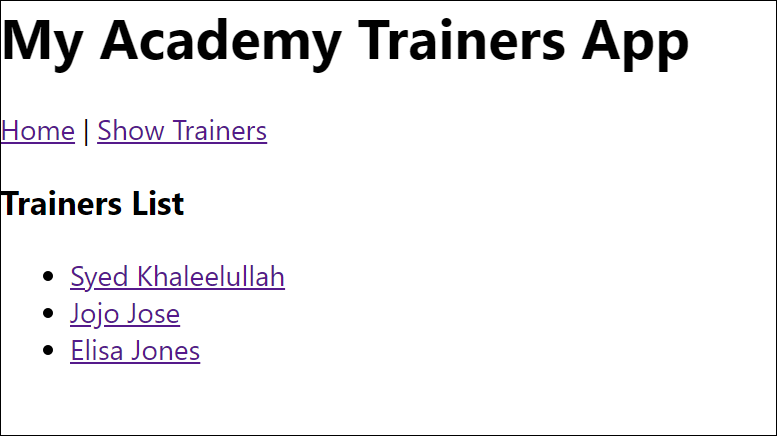


Figure 10: Trainers List

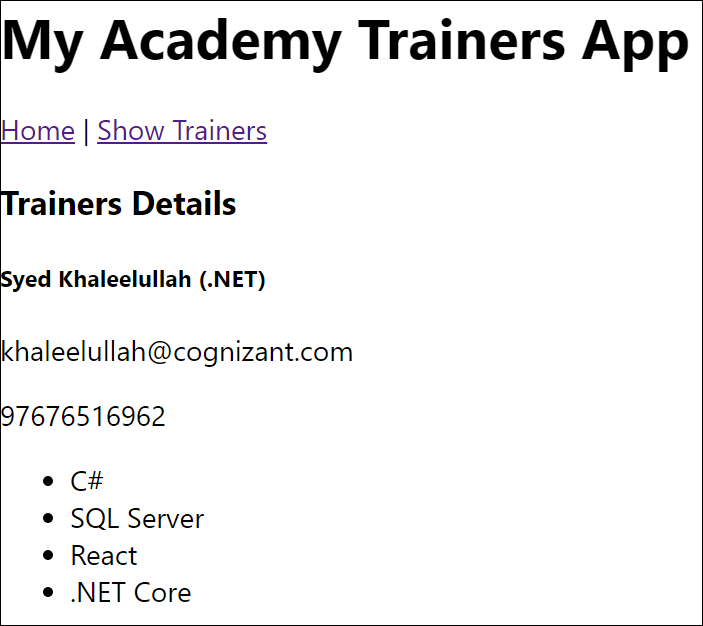
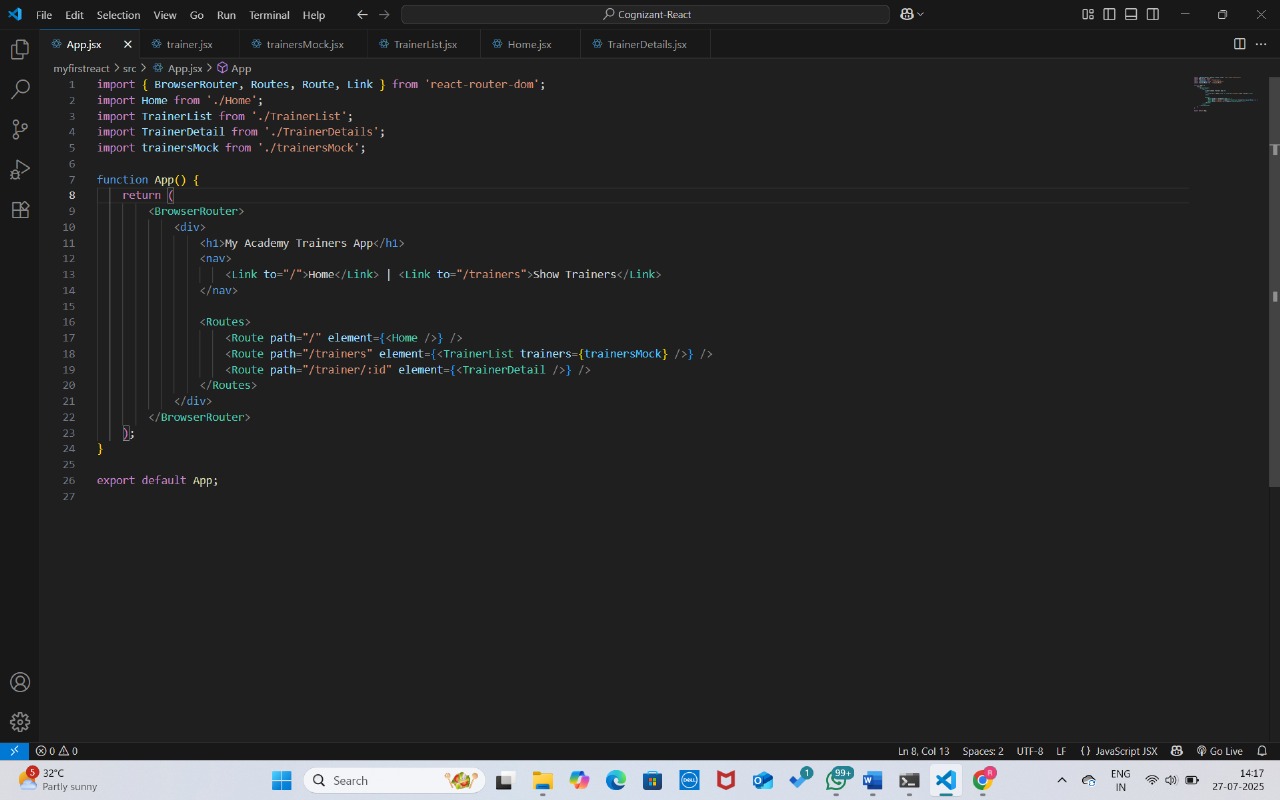
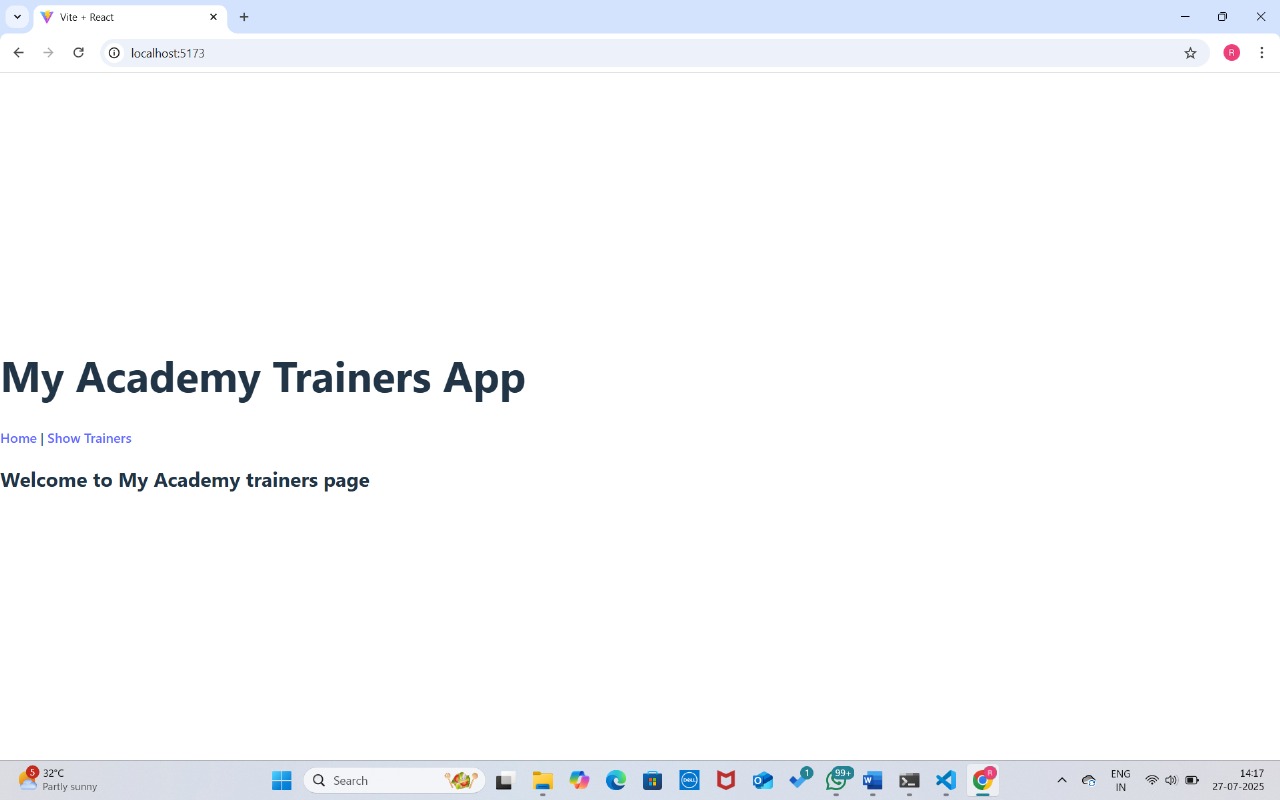
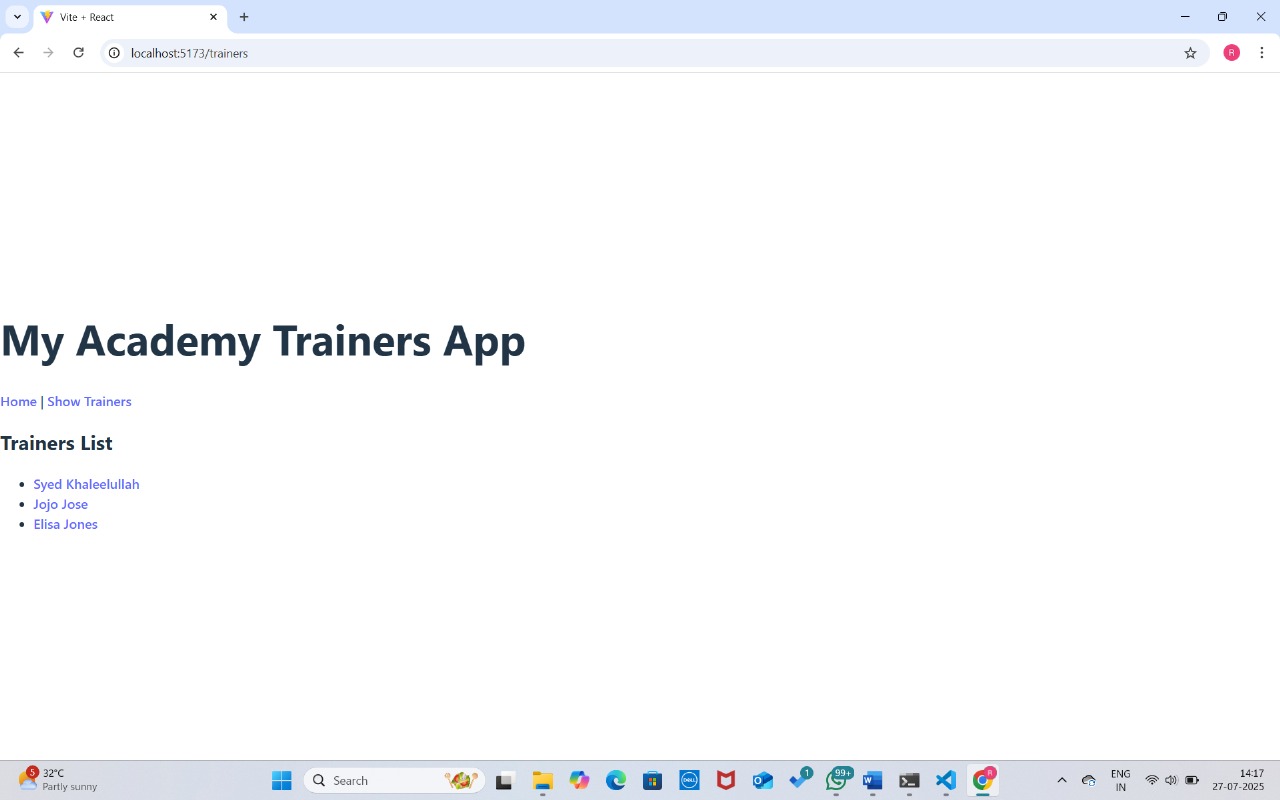


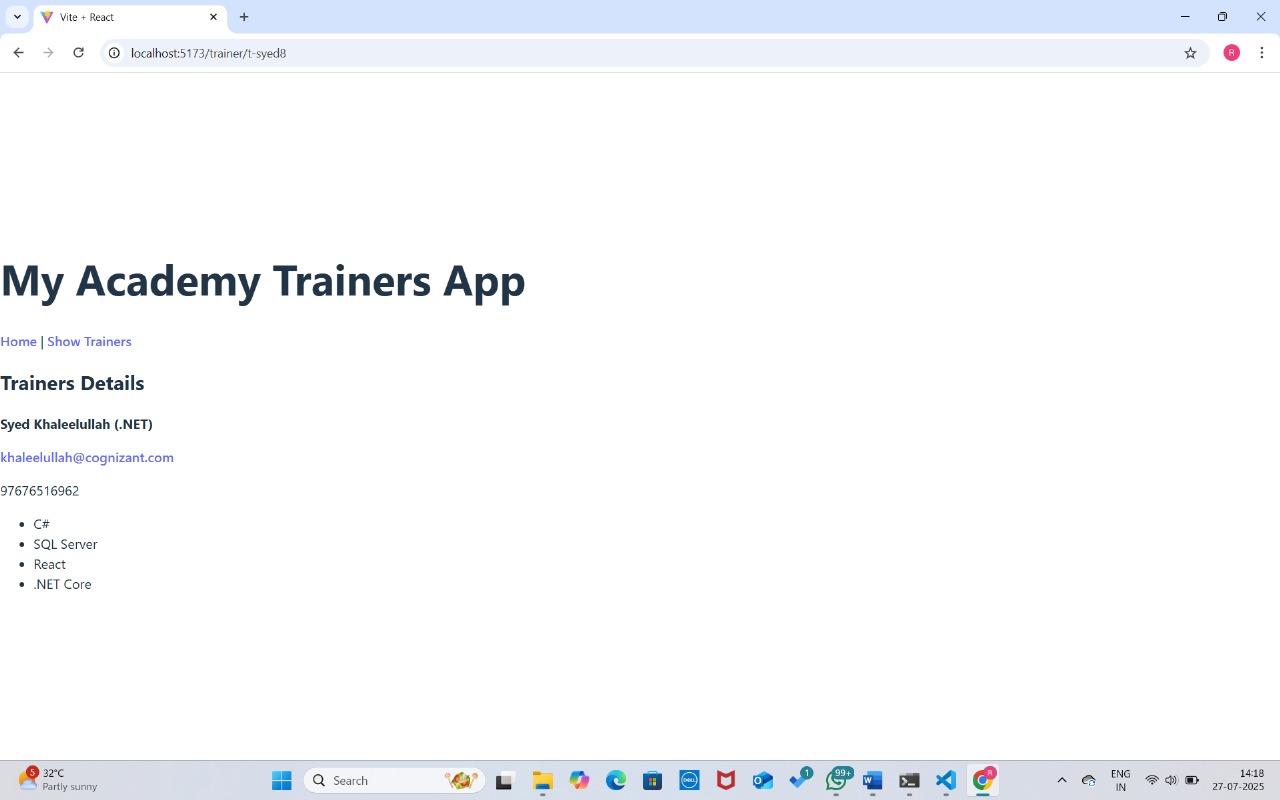
Figure 11: Trainer Details

**OUTPUT:**









**7. React JS – HOL**

**Objectives**

* **Define Props**

Props (short for "properties") in React are read-only inputs passed from a parent component to a child component. They are used to customize and control the behavior or appearance of a component. Props help make components reusable by allowing different values to be passed each time the component is used. Inside a component, props are accessed using props.propertyName or destructured directly in the function parameters. Since props are immutable, they cannot be modified inside the child component — making them ideal for passing static or external data.

* **Explain Default Props**

Default props are a way to define fallback values for props in case no value is provided by the parent component. This ensures that the component still behaves correctly even when some props are missing. In React, you can define default props by assigning them to the component’s defaultProps property.

* **Identify the differences between State and Props**

The main difference between state and props in React is in their purpose and mutability.

* Props are used to pass data from parent to child components. They are read-only and cannot be modified by the receiving component.
* State is used to store and manage data that can change over time within a component. It is mutable and controlled using hooks like useState() or this.setState() in class components.  
  In short, props are used for external configuration, while state is for internal data management.
* **Explain reactDOM.render()’**

The ReactDOM.render() function is used to render a React component (or a React element) into the DOM. It takes two arguments: the first is the JSX or component you want to render, and the second is the DOM element where it should be mounted.

In this hands-on lab, you will learn how to:

* Use Props
* Apply reactDOM.render()

**Prerequisites**

The following is required to complete this hands-on lab:

* Node.js
* NPM
* Visual Studio Code

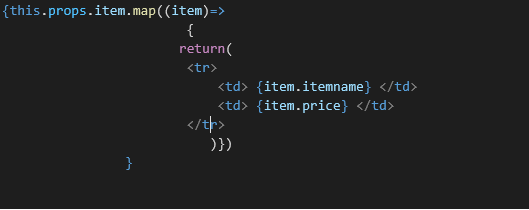
**Notes**

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

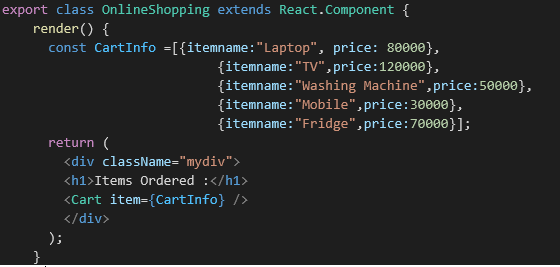
Create a React Application named “shoppingapp” with a class component named “OnlineShopping” and “Cart”.

1. In Cart class, create 2 properties as mentioned below:

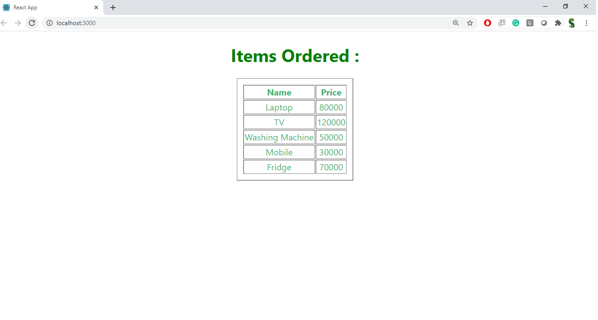
* Itemname
* Price



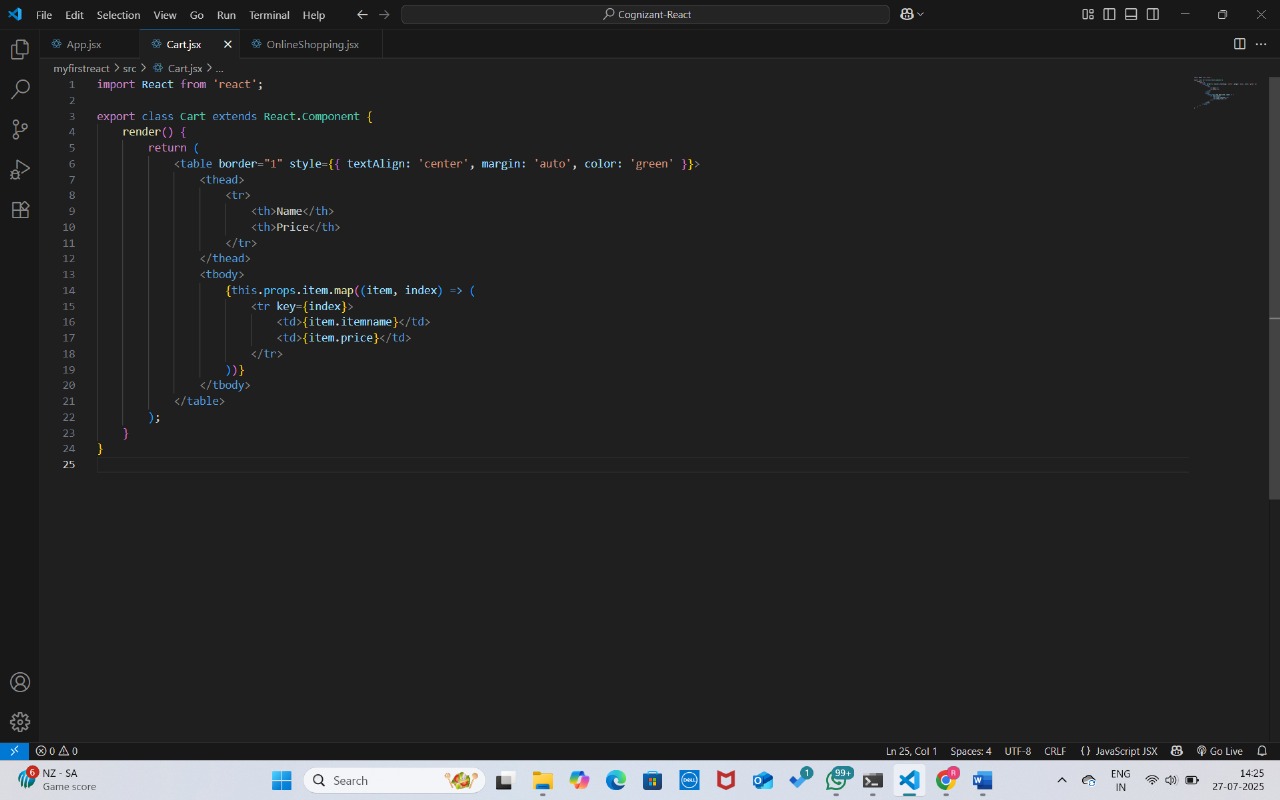
1. In OnlineShopping class, create an array of Cart and initialize 5 items.

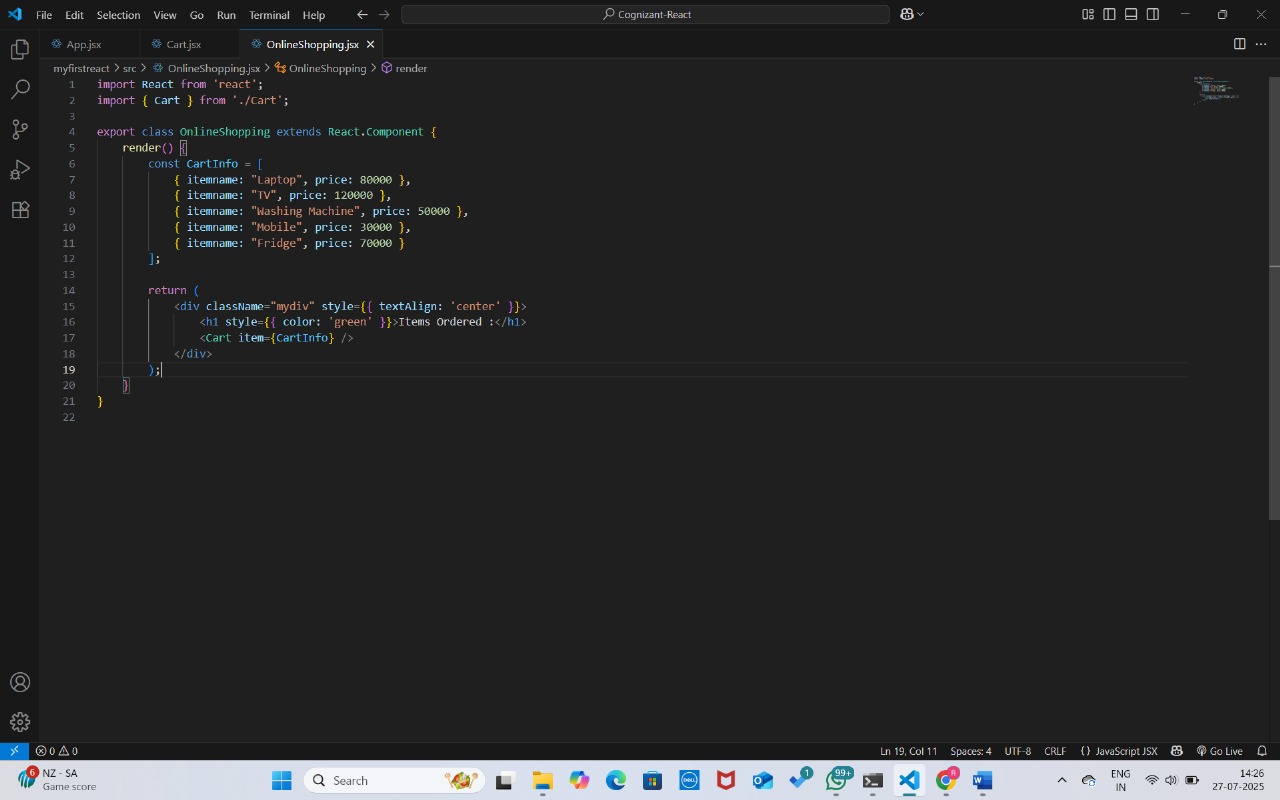


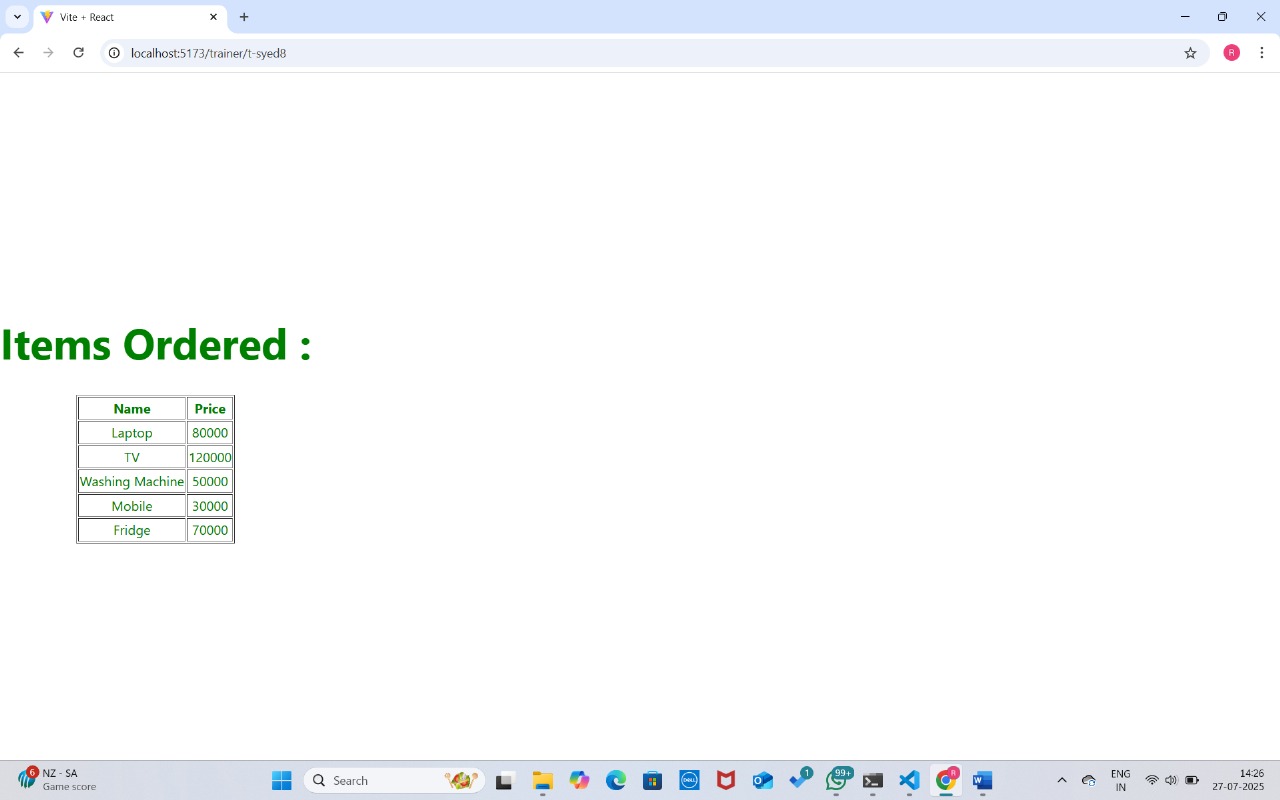
1. Loop through these items and display the data as shown below:



**OUTPUT :**







**8. React JS – HOL**

**Objectives**

* **Explain React State**

React State is a built-in object used to store dynamic data in a component. It allows a component to keep track of information that can change over time, such as user input, API responses, or interactive UI elements. In class components, state is initialized in the constructor using this.state and updated with this.setState(). In function components, state is managed using the useState() hook. Whenever the state changes, React automatically re-renders the component to reflect the updated data in the UI. Unlike props, which are passed from parent to child and are read-only, state is local to the component and can be updated internally to manage interactivity and dynamic behavior.

In this hands-on lab, you will learn how to:

* Use React State object

**Prerequisites**

The following is required to complete this hands-on lab:

* Node.js
* NPM
* Visual Studio Code

**Notes**

Estimated time to complete this lab: 60 minutes.

Create a React App “counterapp” which will have a component named “CountPeople” which will have 2 methods.

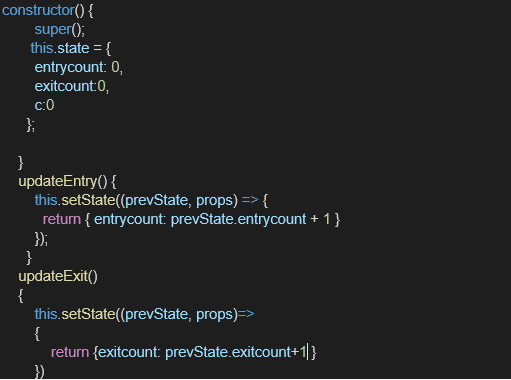
UpdateEntry() 🡪 which will display the number of people who entered the mall.

UpdateExit() 🡪 which will display the number of people who exited the mall.

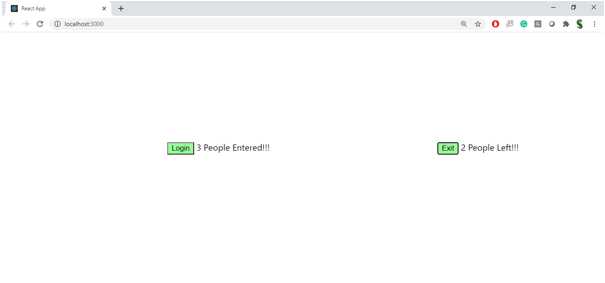
Use Constructor and state to Store the entrycount and exitcount.

The component has 2 buttons

1. Login 🡪 when clicked, the entrycount should get incremented by 1
2. Exit 🡪 when clicked, the exitcount should get incremented by 1



The output should be as follows:



**OUTPUT :**

