**Week – 06**

**React**

**Objectives**

* **Define SPA and its benefits**
* **Define React and identify its working**
* **Identify the differences between SPA and MPA**
* **Explain Pros & Cons of Single-Page Application**
* **Explain about React**
* **Define virtual DOM**
* **Explain Features of React**

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

* **Set up a react environment**
* **Use create-react-app**

**Prerequisites**

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

* **Node.js**
* **NPM**
* **Visual Studio Code**

**Notes**

**Estimated time to complete this lab: 30 minutes.**

**Create a new React Application with the name “myfirstreact”, Run the application to print “welcome to the first session of React” as heading of that page.**

1. **To create a new React app, Install Nodejs and Npm from the following link:**

[**https://nodejs.org/en/download/**](https://nodejs.org/en/download/)

1. **Install Create-react-app by running the following command in the command prompt:**

****

1. **To create a React Application with the name of “myfirstreact”, type the following command:**

****

1. **Once the App is created, navigate into the folder of myfirstreact by typing the following command:**

****

1. **Open the folder of myfirstreact in Visual Studio Code**
2. **Open the App.js file in Src Folder of myfirstreact**
3. **Remove the current content of “App.js”**
4. **Replace it with the following:**

****

1. **Run the following command to execute the React application:**

****

1. **Open a new browser window and type “localhost:3000” in the address bar**

****

Sol.

import logo from './logo.svg';

import './App.css';

function App() {

  return (

    <div className="App">

      <h1>Welcome the first session of React</h1>

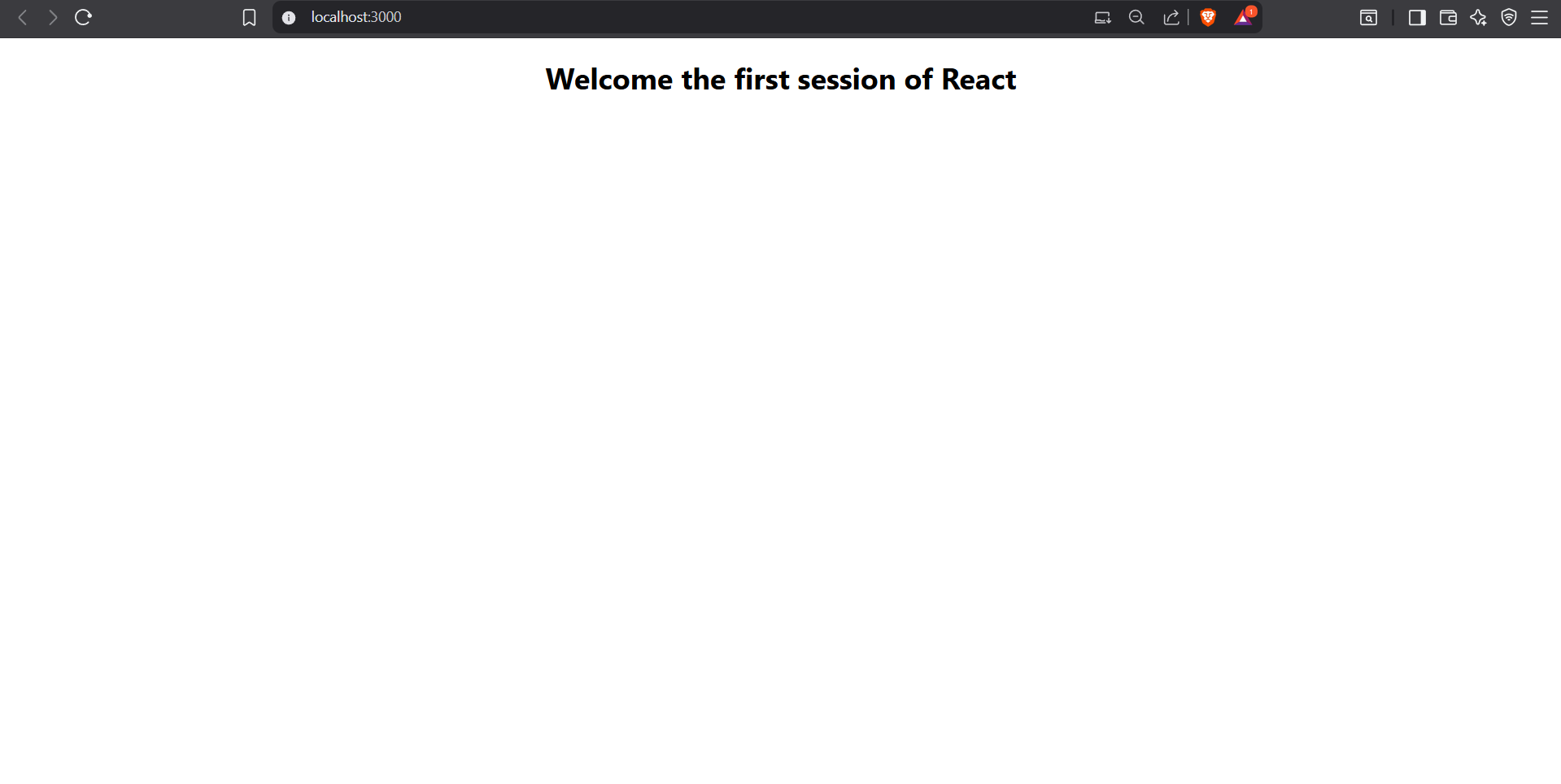
    </div>

  );

}

export default App;

output:



**Objectives**

* Explain React components
* Identify the differences between components and JavaScript functions
* Identify the types of components
* Explain class component
* Explain function component
* Define component constructor
* Define render() function

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

* Create a class component
* Create multiple components
* Render a component

**Prerequisites**

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

* Node.js
* NPM
* Visual Studio Code

**Notes**

Estimated time to complete this lab: **30 minutes.**

Create a react app for Student Management Portal named StudentApp and create a component named Home which will display the Message “Welcome to the Home page of Student Management Portal”. Create another component named About and display the Message “Welcome to the About page of the Student Management Portal”. Create a third component named Contact and display the Message “Welcome to the Contact page of the Student Management Portal”. Call all the three components.

1. Create a React project named “StudentApp” type the following command in terminal of Visual studio:



1. Create a new folder under Src folder with the name “Components”. Add a new file named “Home.js”
2. Type the following code in Home.js



1. Under Src folder add another file named “About.js”
2. Repeat the same steps for Creating “About” and “Contact” component by adding a new file as ”About.js”, “Contact.js” under “Src” folder and edit the code as mentioned for “Home” Component.
3. Edit the App.js to invoke the Home, About and Contact component as follows:



1. In command Prompt, navigate into StudentApp and execute the code by typing the following command:



1. Open browser and type “localhost:3000” in the address bar:



Sol.

App.js

import logo from './logo.svg';

import './App.css';

import Home from './Components/Home';

import About from './Components/About';

import Contact from './Components/Contact';

function App() {

  return (

    <div className="App">

      <Home/>

      <About/>

      <Contact/>

    </div>

  );

}

export default App;

Home.js

import React, { Component } from 'react'

class Home extends Component {

  render() {

    return (

      <div>

        <h1>Welcome to the Home Page of the Student Management Portal</h1>

      </div>

    )

  }

}

export default Home;

About.js

import React, { Component } from 'react'

class About extends Component {

  render() {

    return (

      <div>

        <h1>Welcome to the About page of the Student Mangement Portal</h1>

      </div>

    )

  }

}

export default About;

Contact.js

import React, { Component } from 'react'

class Contact extends Component {

  render() {

    return (

      <div>

        <h1>Welcome to the Contact page of the Student Management Portal</h1>

      </div>

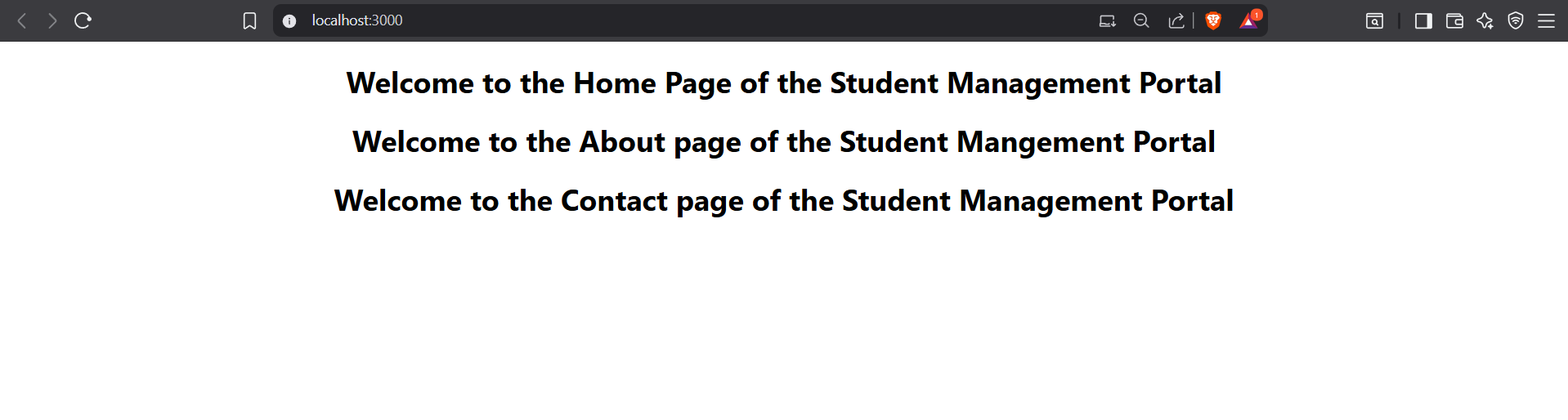
    )

  }

}

export default Contact;

Output:



**Objectives**

* Explain React components
* Identify the differences between components and JavaScript functions
* Identify the types of components
* Explain class component
* Explain function component
* Define component constructor
* Define render() function

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

* Create a function component
* Apply style to components
* Render a component

**Prerequisites**

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

* Node.js
* NPM
* Visual Studio Code

**Notes**

Estimated time to complete this lab: **30 minutes.**

Create a react app for Student Management Portal named scorecalculatorapp and create a function component named “CalculateScore” which will accept Name, School, Total and goal in order to calculate the average score of a student and display the same.

1. Create a React project named “scorecalculatorapp” type the following command in terminal of Visual studio:



1. Create a new folder under Src folder with the name “Components”. Add a new file named “CalculateScore.js”
2. Type the following code in CalculateScore.js





1. Create a Folder named Stylesheets and add a file named “mystyle.css” in order to add some styles to the components:



1. Edit the App.js to invoke the CalculateScore functional component as follows:



1. In command Prompt, navigate into scorecalculatorapp and execute the code by typing the following command:



1. Open browser and type “localhost:3000” in the address bar:



Sol.

CalculateScore.js

import '../Stylesheets/mystyle.css'

const percentToDecimal = (decimal)=>{

    return (decimal.toFixed(2) + '%')

}

const calcScore = (total, goal) =>{

    return percentToDecimal(total/goal);

}

export const CalculateScore = ({Name, School, total, goal}) => (

    <div className='formatstyle'>

        <h1><font color = "Brown"></font></h1>

        <div className='Name'>

            <b><span>Name: </span></b>

            <span>{Name}</span>

        </div>

        <div className='School'>

            <b><span>School: </span></b>

            <span>{School}</span>

        </div>

        <div className='Total'>

            <b><span>Total: </span></b>

            <span>{total}</span>

            <span>Marks</span>

        </div>

        <div className='Score'>

            <b>Score</b>

            <span>

                {

                    calcScore(total, goal)

                }

            </span>

        </div>

    </div>

)

Mystyle.css

.Name{

    font-weight: 300;

    color: blue;

}

.School{

    color: crimson;

}

.Total{

    color: darkmagenta;

}

.formatstyle{

    text-align: center;

    font-size: large;

}

.Score{

    color: forestgreen;

}

App.js

import logo from './logo.svg';

import './App.css';

import { CalculateScore } from './Components/CalculateScore';

function App() {

  return (

    <div className="App">

      <CalculateScore Name={"Steeve"}

      School={"DNV Public School"}

      total={284}

      goal={3}/>

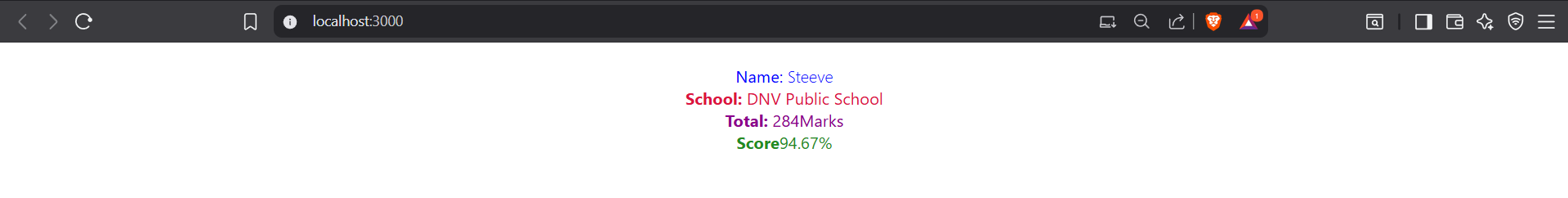
    </div>

  );

}

export default App;

Output:



**Objectives**

* Explain the need and Benefits of component life cycle
* Identify various life cycle hook methods
* List the sequence of steps in rendering a component

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

* Implement componentDidMount() hook
* Implementing componentDidCatch() life cycle hook.

**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.**

1. Create a new react application using *create-react-app* tool with the name as “blogapp”
2. Open the application using VS Code
3. Create a new file named as **Post.js** in **src folder** with following properties



*Figure 2: Post class*

1. Create a new class based component named as **Posts** inside **Posts.js** file



*Figure 3: Posts Component*

1. Initialize the component with a list of Post in state of the component using the constructor
2. Create a new method in component with the name as **loadPosts()** which will be responsible for using Fetch API and assign it to the component state created earlier. To get the posts use the url (<https://jsonplaceholder.typicode.com/posts>)



*Figure 4: loadPosts() method*

1. Implement the **componentDidMount()** hook to make calls to **loadPosts()** which will fetch the posts



*Figure 5: componentDidMount() hook*

1. Implement the **render()** which will display the title and post of posts in html page using heading and paragraphs respectively.



*Figure 6: render() method*

1. Define a **componentDidCatch()** method which will be responsible for displaying any error happing in the component as alert messages.



*Figure 7: componentDidCatch() hook*

1. Add the Posts component to App component.
2. Build and Run the application using *npm start* command.

Sol.

Post.js

import React, { Component } from 'react';

class PostModel {

  constructor(id, title, body) {

    this.id = id;

    this.title = title;

    this.body = body;

  }

}

class Post extends Component {

  constructor(props) {

    super(props);

    this.state = {

      posts: [

        new PostModel(1, 'Hello React', 'This is the first blog post.'),

        new PostModel(2, 'Component Lifecycle', 'Let’s learn about lifecycle methods.'),

        new PostModel(3, 'Fetching Data', 'We can fetch API data using fetch.'),

      ],

      hasError: false,

      errorMsg: '',

    };

  }

  loadPosts() {

    fetch('https://jsonplaceholder.typicode.com/posts')

      .then((res) => {

        if (!res.ok) throw new Error('Network response was not ok');

        return res.json();

      })

      .then((data) => {

        const loadedPosts = data.map(

          (item) => new PostModel(item.id, item.title, item.body)

        );

        this.setState({ posts: loadedPosts });

      })

      .catch((error) => {

        this.setState({ hasError: true, errorMsg: error.message });

      });

  }

  componentDidMount() {

    this.loadPosts();

  }

  componentDidCatch(error, info) {

    alert(`Rendering error: ${error}`);

  }

  render() {

    const { posts, hasError, errorMsg } = this.state;

    if (hasError) {

      return <div>Error: {errorMsg}</div>;

    }

    return (

      <div>

        <h1>Blog Posts</h1>

        {posts.map((post) => (

          <div key={post.id}>

            <h2>{post.title}</h2>

            <p>{post.body}</p>

            <hr />

          </div>

        ))}

      </div>

    );

  }

}

export default Post;

Output:



**Objectives**

* Understanding the need for styling react component
* Working with CSS Module and inline styles

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

* Style a react component
* Define styles using the CSS Module
* Apply styles to components using className and style properties

## **Prerequisites**

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

* Node.js
* NPM
* Visual Studio Code

## **Notes**

Estimated time to complete this lab: **30 minutes.**

My Academy team at Cognizant want to create a dashboard containing the details of ongoing and completed cohorts. A react application is created which displays the detail of the cohorts using react component. You are assigned the task of styling these react components.

Download and build the attached react application.



1. Unzip the react application in a folder
2. Open command prompt and switch to the react application folder
3. Restore the node packages using the following commands



Figure 1: Restore packages

1. Open the application using VS Code
2. Create a new CSS Module in a file called “CohortDetails.module.css”
3. Define a css class with the name as “box” with following properties

*Width = 300px;*

*Display = inline block;*

*Overall 10px margin*

*Top and bottom padding as 10px*

*Left and right padding as 20px*

*1 px border in black color*

*A border radius of 10px*

1. Define a css style for html <dt> element using tag selector. Set the font weight to 500.
2. Open the cohort details component and import the CSS Module
3. Apply the box class to the container div
4. Define the style for <h3> element to use “green” color font when cohort status is “ongoing” and “blue” color in all other scenarios.
5. Final result should look similar to the below image



Figure 2: Final Result

Sol.

CohortDetails.css

.box{

    width: 300px;

    display: inline-block;

    margin: 10px;

    padding-top: 10px;

    padding-bottom: 10px;

    padding-left: 20px;

    padding-right: 20px;

    border: 1px solid black;

    border-radius: 10px;

}

dt{

    font-weight: 500;

}

CohortDetails.js

import './CohortDetails.css'

function CohortDetails(props) {

    return (

        <div className='box'>

            <h3 style={{color:"green"}}>

                {props.cohort.cohortCode} -

                <span>{props.cohort.technology}</span>

            </h3>

            <dl>

                <dt>Started On</dt>

                <dd>{props.cohort.startDate}</dd>

                <dt>Current Status</dt>

                <dd>{props.cohort.currentStatus}</dd>

                <dt>Coach</dt>

                <dd>{props.cohort.coachName}</dd>

                <dt>Trainer</dt>

                <dd>{props.cohort.trainerName}</dd>

            </dl>

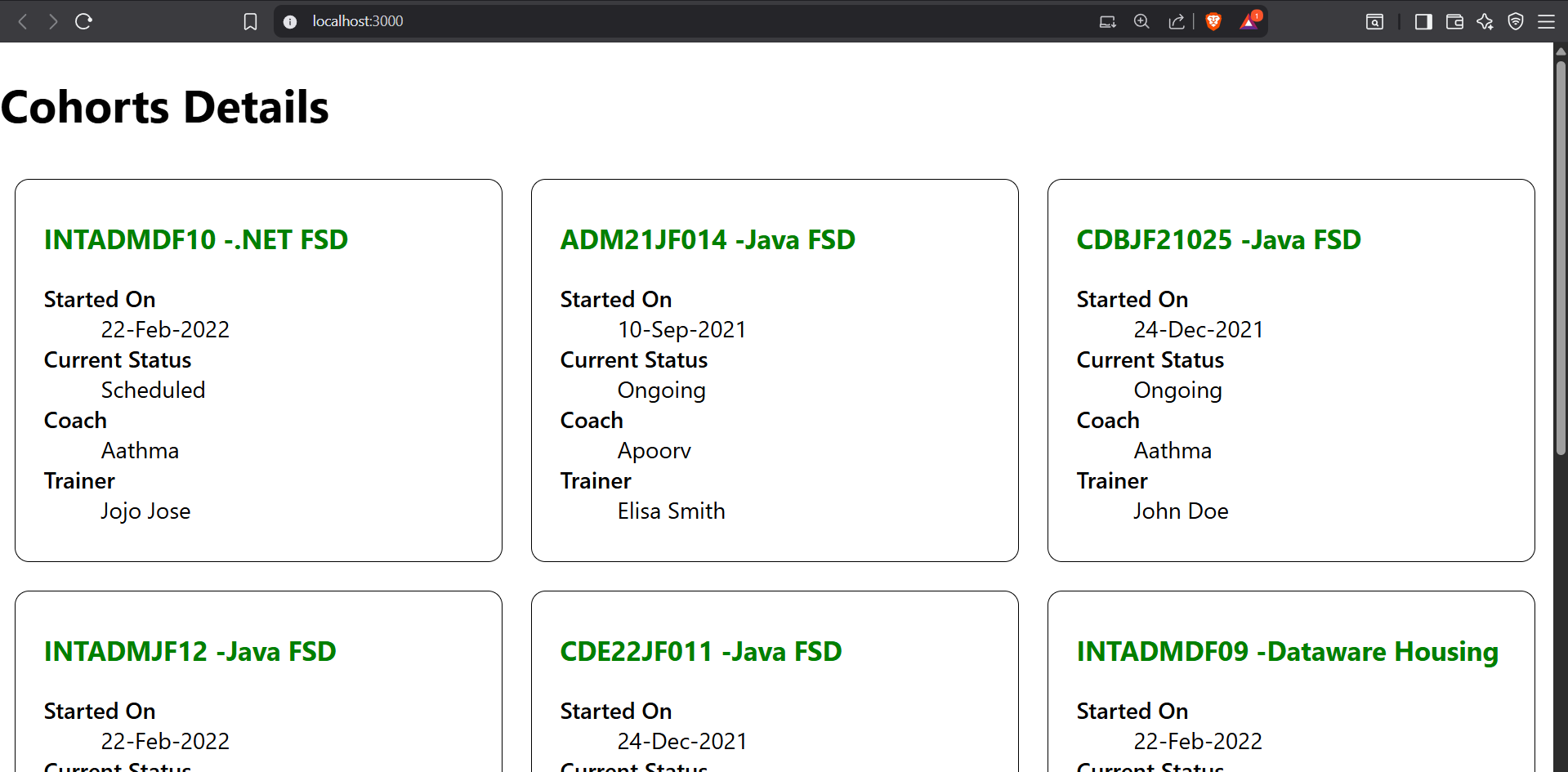
        </div>

    );

}

export default CohortDetails;

Output:



**Objectives**

* Explain the need and benefits of React Router
* Identify the Components in React Router
* List the types of Router Components
* Parameter passing via url

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

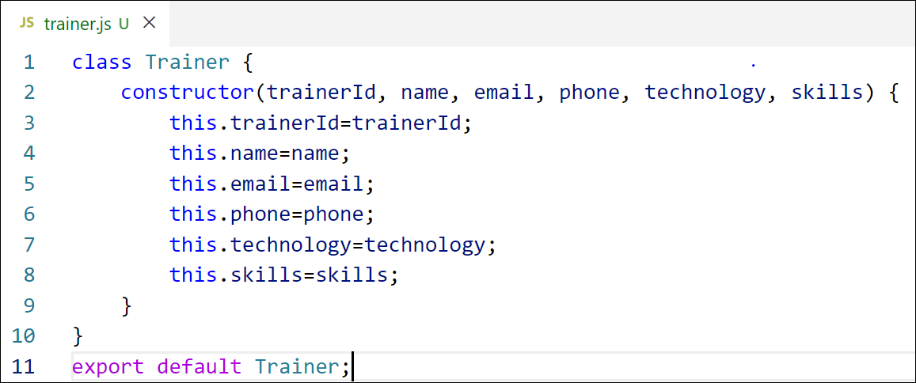
**Notes**

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.

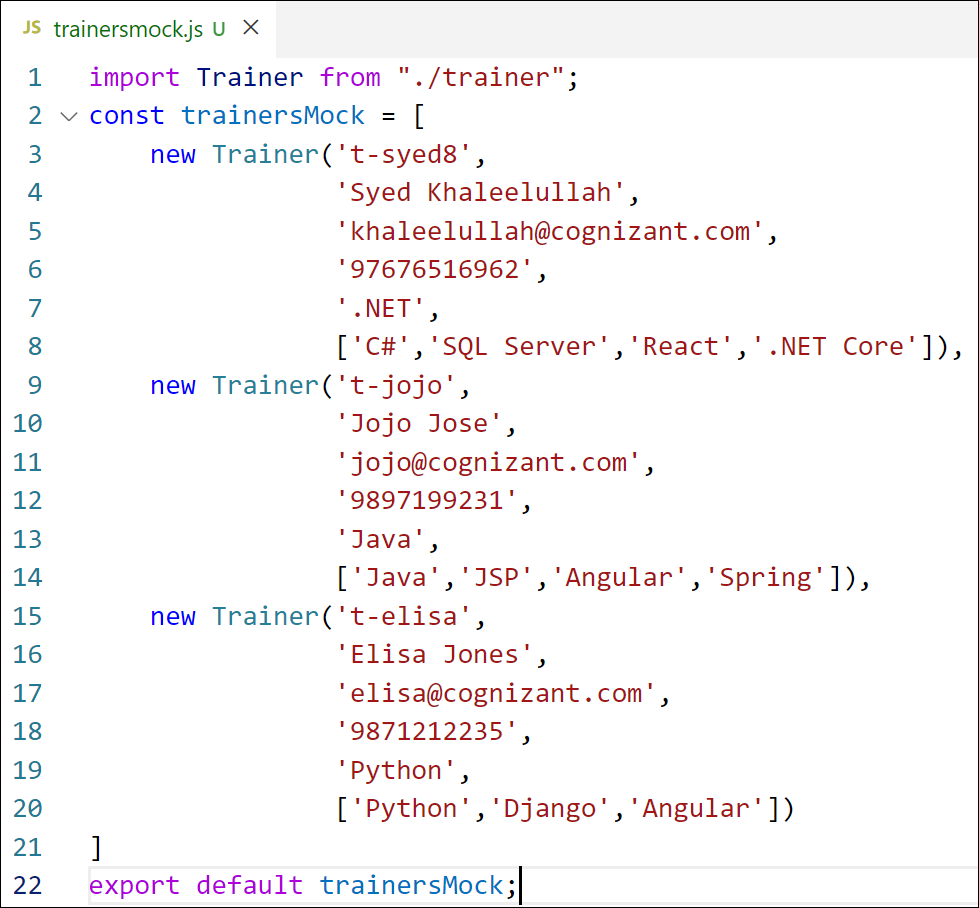
The following trainers’ data application will deal.

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



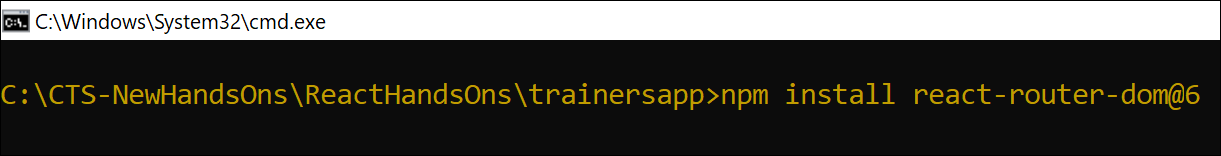
*Figure 2: Trainer.js*

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



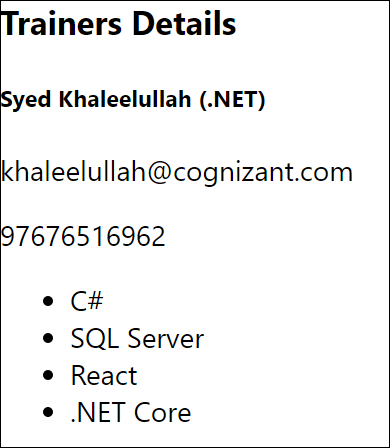
*Figure 7: App Component*

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.

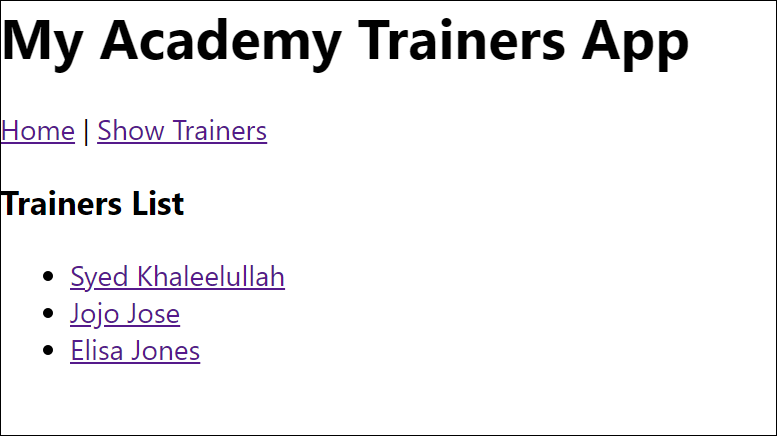


*Figure 8: Trainers Detail Component*

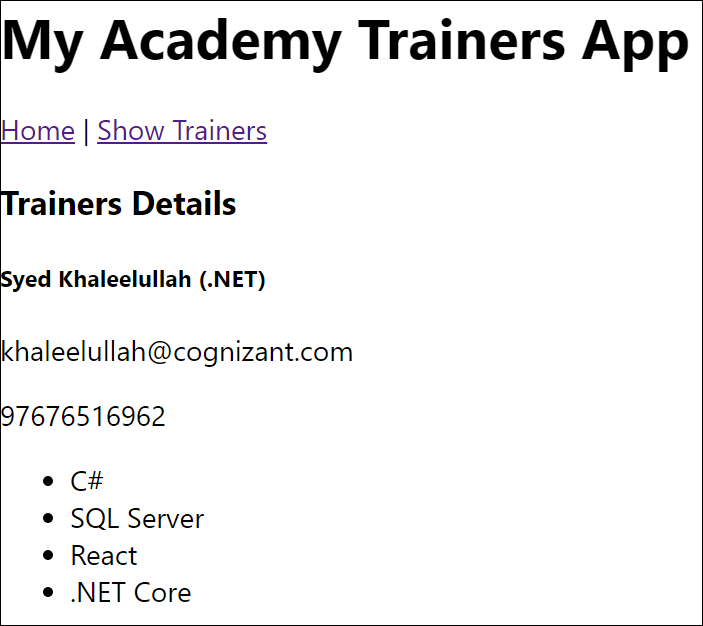
1. Build and run the application. The complete layout of the application will look as follows.



*Figure 9: Home*



*Figure 10: Trainers List*



*Figure 11: Trainer Details*