

1.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/>

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

```
C:>npm install -g create-react-app
```

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

```
C:>npx create-react-app myfirstreact
```

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

```
C:>cd myfirstreact
```

5. Open the folder of myfirstreact in Visual Studio Code
6. Open the App.js file in Src Folder of myfirstreact
7. Remove the current content of "App.js"
8. Replace it with the following:

```
function App() {  
  return (  
    <h1> Welcome the first session of React </h1>  
  );  
}
```

9. Run the following command to execute the React application:

```
C:\myfirstreact>npm start
```

10. Open a new browser window and type "localhost:3000" in the address bar

Answer:

Objectives:

1. Define SPA (Single Page Application) and its benefits

- SPA is a web app that loads a single HTML page and dynamically updates the content as the user interacts.
- Benefits:
 - Faster navigation (no full page reloads)
 - Improved user experience
 - Efficient data loading via APIs

2. Define React and identify its working

- React is a JavaScript library developed by Facebook for building user interfaces.
- It works by using components, a virtual DOM, and one-way data binding to build fast and scalable apps.

3. Identify the differences between SPA and MPA (Multi-Page Applications)

Feature	SPA	MPA
Page Loading	Loads once, then updates views	Reloads a new page every time
Speed	Faster after initial load	Slower due to page reloads
URL Structure	Usually one base URL	Multiple URLs per page
Maintenance	Easier with component-based UI	Harder due to page duplication

4. Explain Pros & Cons of Single Page Applications

- **Pros:**
 - Fast and responsive
 - Seamless user experience
 - Easy to debug using browser tools
- **Cons:**
 - SEO can be harder
 - Initial load time can be higher
 - Requires JavaScript to function

5. Explain about React

- React is used to create reusable UI components.
- It enables the creation of large web applications that can update data without reloading the page.
- React uses JSX (JavaScript + XML).

6. Define Virtual DOM

- A lightweight in-memory representation of the real DOM.
- React uses it to detect what has changed and efficiently update only that part of the actual DOM.

7. Explain Features of React

- Component-Based Architecture
- Virtual DOM
- Unidirectional Data Flow
- JSX Syntax
- Reusability
- Strong Community Support

Answer:

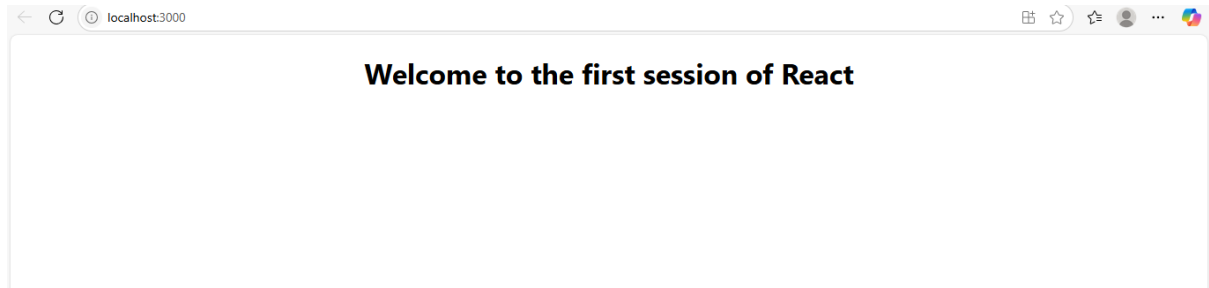
App.js:

```
function App() {  
  return (  
    <div style={{ textAlign: "center" }}>  
      <h1>Welcome to the first session of React</h1>  
    </div>  
  );  
}
```

```
}
```

```
export default App;
```

Output:



2.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:

```
C:>npx create-react-app StudentApp
```

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

```
import React, {Component} from 'react';

import class Home extends Component{
  render(){
    <div>
      <h3> Welcome to the Home Page of Student Management Portal </h3>
    </div>
  }
}
```

4. Under Src folder add another file named “About.js”

5. 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.
6. Edit the App.js to invoke the Home, About and Contact component as follows:

```
import logo from './logo.svg';
import './App.css';
import {Home} from './Components/Home';
import {About} from './Components/About';
import {Contact} from './Components/Contact';

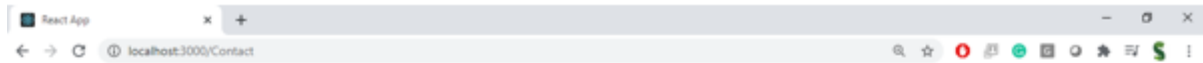
function App() {
  return (
    <div className="container">
      <Home/>
      <About/>
      <Contact/>
    </div>
  );
}

export default App;
```

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

```
C:\studentapp>npm start
```

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



Welcome to the Home Page of Student Management Portal

Welcome to the About Page of Student Management Portal

Welcome to the Contact Page of Student Management Portal

Answer:

Explain React Components

React components are the core building blocks of a React application. They encapsulate structure, behavior, and styling, and are used to render parts of the user interface.

2. Identify the Differences Between Components and JavaScript Functions

- Components are used to build UI; JavaScript functions perform logic.
- Components return JSX; functions return values.
- Components can manage state; functions do not manage UI state (unless enhanced with hooks).
- Components follow React's lifecycle; functions do not.

3. Identify the Types of Components

- **Class Components** – Use ES6 classes and lifecycle methods.
- **Function Components** – Use plain functions and React hooks for state and effects.

4. Explain Class Component

A class component is a React component defined using a JavaScript ES6 class. It can hold and manage internal state and use lifecycle methods.

5. Explain Function Component

A function component is a simpler type of React component defined as a JavaScript function. It can use React hooks to manage state and lifecycle behavior.

6. Define Component Constructor

The constructor is a special method used in class components to initialize state and bind methods. It runs once when the component is created.

7. Define render() Function

The render() function in a class component is responsible for describing the UI. It returns the JSX that should be displayed on the screen.

Output:

About.js:

```
import React, { Component } from 'react';
```

```
class About extends Component {  
  render() {  
    return (  
      <div>  
        <h3>Welcome to the About Page of Student Management Portal</h3>  
      </div>  
    );  
  }  
}
```

```
export default About;
```

Contact.js:

```
import React, { Component } from 'react';
```



```
class Contact extends Component {  
  render() {  
    return (  
      <div>  
        <h3>Welcome to the Contact Page of Student Management Portal</h3>  
      </div>  
    );  
  }  
}
```

```
export default Contact;
```

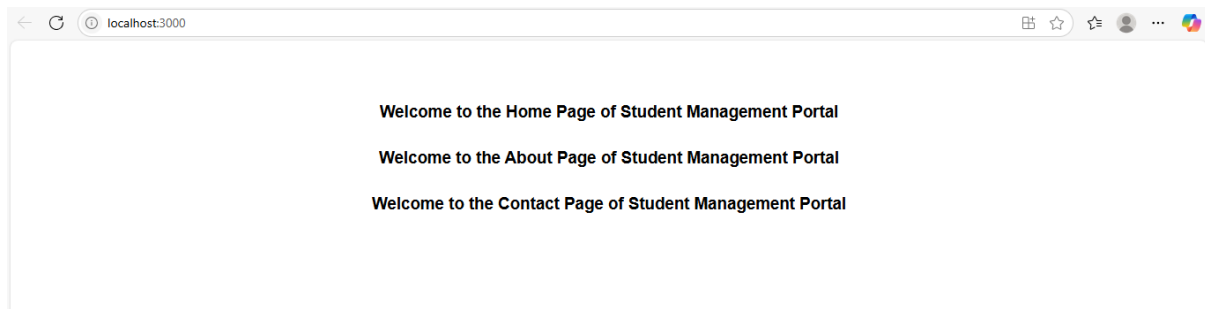
Home.js:

```
import React, { Component } from 'react';
```

```
class Home extends Component {  
  render() {  
    return (  
      <div>  
        <h3>Welcome to the Home Page of Student Management Portal</h3>  
      </div>  
    );  
  }  
}
```

```
export default Home;
```

Output:



3.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.

9. Create a React project named "scorecalculatorapp" type the following command in terminal of Visual studio:

```
C:>npx create-react-app scorecalculatorapp
```

10. Create a new folder under Src folder with the name "Components". Add a new file named "CalculateScore.js"

11. Type the following code in 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">Student Details:</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>
)

```

12. Create a Folder named Stylesheets and add a file named "mystyle.css" in order to add some styles to the components:

```

.Name
{
  font-weight:300;
  color:blue;
}
.School
{
  color:crimson;
}
.Total
{
  color:darkmagenta;
}
.formatstyle
{
  text-align:center;
  font-size:large;
}
.Score
{
  color:forestgreen;
}

```

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

```

import {CalculateScore} from '../src/components/CalculateScore';

function App()
{
  return(
    <div>
      <CalculateScore Name={"Steeve"}
        School={"DNV Public School"}
        total={284}
        goal={3}
      />
    </div>
  )
}

export default App;

```

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

```
C:\scorecalculatorapp>npm start
```

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

Answer:

Explain React Components

React components are **independent, reusable building blocks** of a user interface in a React application. Each component defines how a part of the UI should appear and behave.

• Identify the Differences Between Components and JavaScript Functions

Aspect	React Components	JavaScript Functions
Purpose	Build and manage UI	Perform general logic or computation
Return Type	JSX (used to define UI)	Primitive values or objects
State & Lifecycle	Can manage state & lifecycle (in React)	No state or lifecycle
React-specific	Designed for React framework	Generic programming use

• Identify the Types of Components

React provides two main types of components:

- **Class Components:** Use ES6 class syntax, support state and lifecycle methods.
- **Function Components:** Simpler and use hooks to manage state and effects.

• Explain Class Component

A class component is a React component defined using a class. It can have its own state, lifecycle methods, and uses a render() method to return JSX.

• Explain Function Component

A function component is a React component written as a plain JavaScript function. It returns JSX and can use React Hooks (like useState) to manage state and side effects.

- **Define Component Constructor**

The constructor is a special method in a class component. It is used to initialize component state and bind methods. It is called once when the component is created.

- **Define render() Function**

The render() function is a method in class components. It returns JSX that describes what should be displayed on the screen. React calls this method during the rendering phase.

Output:

CalculateScore.js

```
import "../Stylesheets/mystyle.css";

const percentToDecimal = (decimal) => {
  return (decimal * 100).toFixed(2) + '%';
};

const calcScore = (total, goal) => {
  return percentToDecimal(total / goal);
};

export const CalculateScore = ({ Name, School, total, goal }) => {
  return (
    <div className="formatstyle">
      <h1><font color="brown">Student Details:</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;
}
```

```
.Score {
  color: forestgreen;
}
```



```
.formatstyle {  
  text-align: center;  
  font-size: large;  
}
```

App.js:

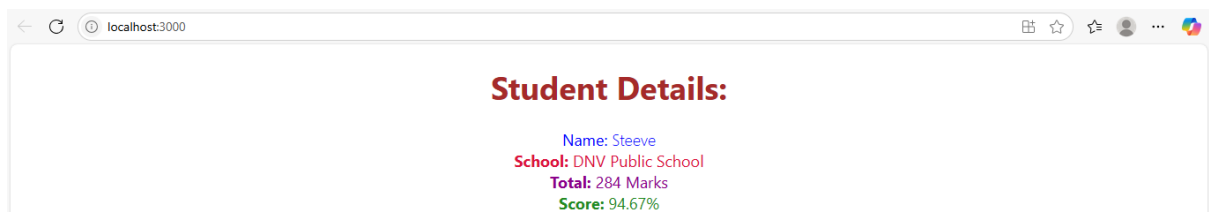
```
import { CalculateScore } from './Components/CalculateScore';
```

```
function App() {  
  return (  
    <div>  
      <CalculateScore  
        Name="Steeve"  
        School="DNV Public School"  
        total={284}  
        goal={300}  
      />  
    </div>  
  );  
}
```

```
export default App;
```

```
App;
```

Output:



4. 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

```

JS Post.js U X
1 class Post {
2     constructor(id, title, body){
3         this.id=id;
4         this.title=title;
5         this.body=body;
6     }
7 }
8 export default Post;

```

Figure 1: Post class

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

```

JS Posts.js U X
1 class Posts extends React.Component {
2     constructor(props){
3         super(props);
4     }
5 }

```

Figure 2: Posts Component

5. Initialize the component with a list of Post in state of the component using the constructor
6. 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>)

```

JS Posts.js U X
1 class Posts extends React.Component {
2     constructor(props){
3         super(props);
4         //code
5     }
6     loadPosts() {
7         //code
8     }
9 }

```

Figure 3: loadPosts() method

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

```

JS Posts.js U X
1  class Posts extends React.Component {
2    constructor(props){
3      super(props);
4      //code
5    }
6    loadPosts() {
7      //code
8    }
9    componentDidMount() {
10     //code
11   }
12 }

```

Figure 4: componentDidMount() hook

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

```

JS Posts.js U X
1  class Posts extends React.Component {
2    > constructor(props) { ...
5    }
6    > loadPosts() { ...
8    }
9    > componentDidMount() { ...
11   }
12   render() {
13     //code
14   }
15 }

```

Figure 5: render() method

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

```

JS Posts.js U X
1  class Posts extends React.Component {
2  >    constructor(props) { ...
5      }
6  >    loadPosts() { ...
8      }
9  >    componentDidMount() { ...
11     }
12 >    render() { ...
14     }
15     componentDidCatch(error, info) {
16         //code
17     }
18 }

```

Figure 6: componentDidCatch() hook

10. Add the Posts component to App component.

11. Build and Run the application using *npm start* command.

Answer:

Objectives:

Explain the Need and Benefits of Component Lifecycle:

React components go through a lifecycle of mounting, updating, and unmounting. Lifecycle methods allow you to:

- Perform side effects like API calls (componentDidMount)
- Handle errors (componentDidCatch)
- Control rendering performance
- Clean up resources

Identify Various Lifecycle Hook Methods

- | | | |
|------------------|-------|---|
| 1.Phase | ----> | Lifecycle Methods |
| 2.Mounting | ----> | constructor, render, componentDidMount |
| 3.Updating | ----> | shouldComponentUpdate, componentDidUpdate |
| 4.Unmounting | ----> | componentWillUnmount |
| 5.Error Handling | ----> | componentDidCatch, getDerivedStateFromError |

Sequence of Steps in Rendering a Component

1. constructor()

2. render()
3. componentDidMount()
4. On update: shouldComponentUpdate() → render() → componentDidUpdate()
5. On error: componentDidCatch()

Post.js:

```
class Post {  
  constructor(id, title, body) {  
    this.id = id;  
    this.title = title;  
    this.body = body;  
  }  
}
```

export default Post;

Posts.js:

```
// src/Posts.js  
import React from 'react';  
import Post from './Post';
```

```
class Posts extends React.Component {  
  constructor(props) {  
    super(props);  
    this.state = {  
      posts: [],  
      hasError: false  
    };  
  }  
}
```

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

```
.then(response => response.json())
.then(data => {
  const postList = data.slice(0, 10).map(post => new Post(post.id, post.title, post.body));
  this.setState({ posts: postList });
})
.catch(error => {
  console.error("Fetch error:", error);
  this.setState({ hasError: true });
});
}
```

```
componentDidMount() {
  this.loadPosts();
}
```

```
componentDidCatch(error, info) {
  console.error("Error in Posts component:", error, info);
  alert("An error occurred in Posts component.");
  this.setState({ hasError: true });
}
```

```
render() {
  if (this.state.hasError) {
    return <h2>Something went wrong while loading posts.</h2>;
  }
}
```

```
return (
  <div>
    <h1>Blog Posts</h1>
    {this.state.posts.map(post => (
      <div key={post.id} style={{ marginBottom: '20px' }}>
```

```
        <h3>{post.title}</h3>
        <p>{post.body}</p>
      </div>
    )}
  </div>
);
}
```

```
export default Posts;
```

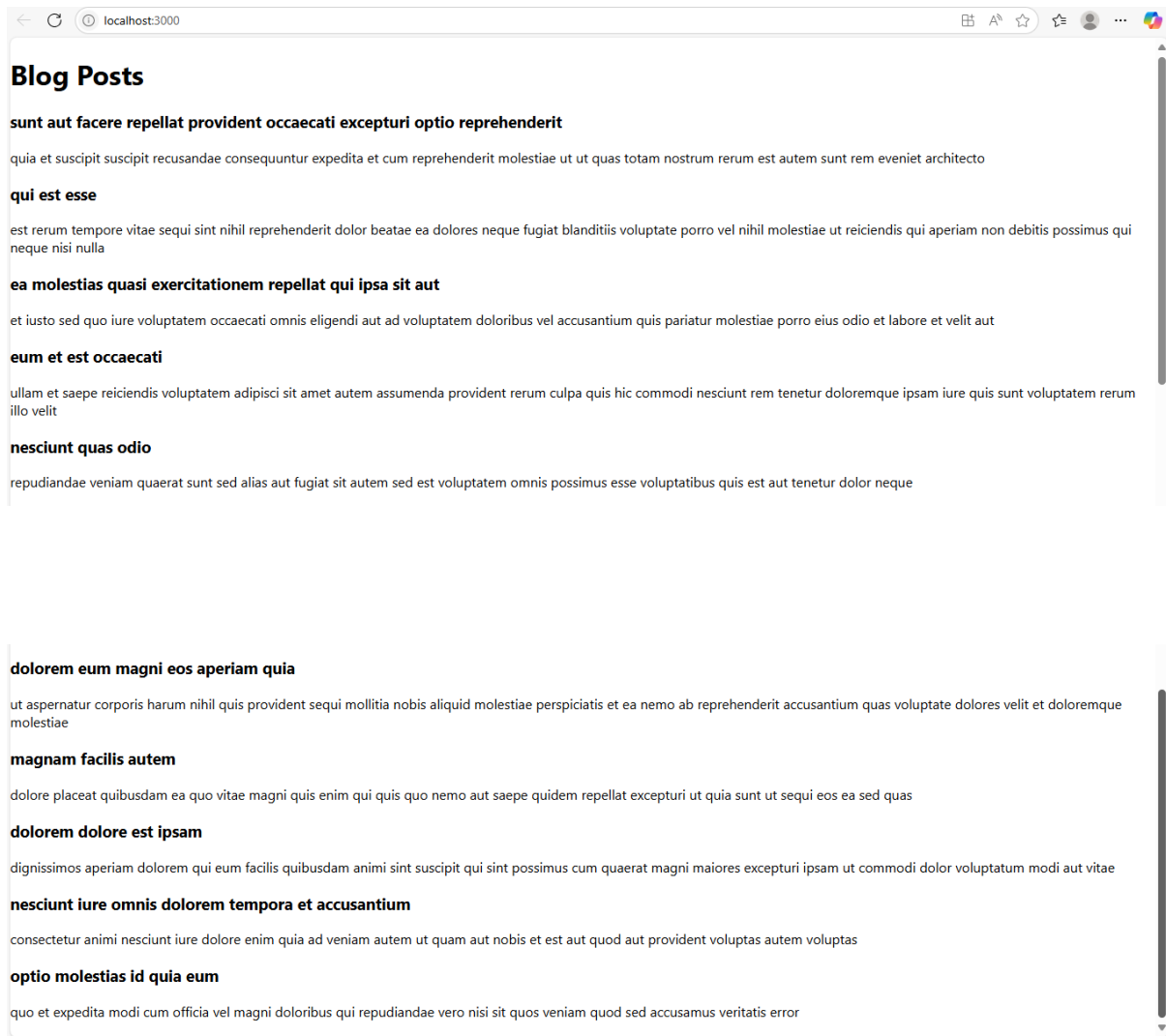
App.js:

```
import React from 'react';
import Posts from './Posts';
```

```
function App() {
  return (
    <div className="App">
      <Posts />
    </div>
  );
}
```

```
export default App;
```

Output:



5. 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

```

C:\Windows\System32\cmd.exe

C:\CTS-NewHandsOns\ReactHandsOns\cohortstracker>npm install
  
```

Figure 7: Restore packages

4. Open the application using VS Code
5. Create a new CSS Module in a file called "CohortDetails.module.css"
6. 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

7. Define a css style for html <dt> element using tag selector. Set the font weight to 500.
8. Open the cohort details component and import the CSS Module

9. Apply the box class to the container div
10. Define the style for <h3> element to use “green” color font when cohort status is “ongoing” and “blue” color in all other scenarios.
11. Final result should look similar to the below image

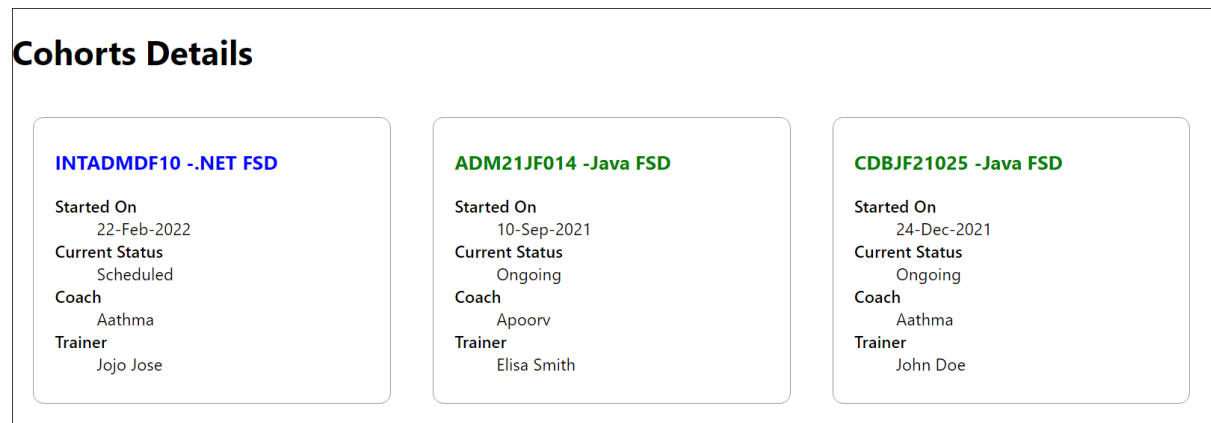


Figure 8: Final Result

Answer:

CohortDetails.js:

```
import React from 'react';
import styles from './CohortDetails.css';

const CohortDetails = ({ cohort }) => {
  const headingStyle = {
    color: cohort.status.toLowerCase() === 'ongoing' ? 'green' :
      cohort.status.toLowerCase() === 'scheduled' ? 'blue' : 'black',
  };

  return (
    <div className={styles.box}>
      <h3 style={headingStyle}>{cohort.name}</h3>
      <dl>
        <dt>Started On</dt>
        <dd>{cohort.startDate}</dd>
```

```
    <dt>Current Status</dt>
    <dd>{cohort.status}</dd>

    <dt>Coach</dt>
    <dd>{cohort.coach}</dd>

    <dt>Trainer</dt>
    <dd>{cohort.trainer}</dd>
  </dl>
</div>
);
};
```

export default CohortDetails;

CohortDetails.css:

```
.outer-box {
  border: 0.5px solid #ccc;
  padding: 30px;
  margin: 40px auto;
  background-color: #fff;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
  display: inline-block;
}
```

```
.cohort-container {
  display: flex;
  flex-direction: row;
  justify-content: center;
  gap: 20px;
  flex-wrap: wrap;
```

```
}
```

```
.box {  
  width: 300px;  
  padding: 15px 20px;  
  border: 1px solid #ccc;  
  background-color: #fefefe;  
  border-radius: 10px;  
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);  
}
```

```
dt {  
  font-weight: 500;  
}
```

```
h3 {  
  margin-top: 0;  
  margin-bottom: 10px;  
}
```

App.css:

```
.outer-box {  
  border: 1px solid #070707;  
  padding: 0.1px 15px 20px 4px;  
  margin: 40px auto;  
  background-color: #fff;  
  box-shadow: 0 0 10px rgba(95, 95, 95, 0.1);  
  display: inline-block;  
}
```

```
.cohort-wrapper {
```

```
display: flex;
flex-direction: row;
justify-content: center;
gap: 40px;
flex-wrap: wrap;
}
```

```
.box {
width: 300px;
padding: 15px 20px;
border: 1px solid #ccc;
background-color: #fefefe;
box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}
```

```
.cohort-wrapper .box:first-child {
margin-left: 10px;
margin-top: 10px;
}
```

```
.cohort-container {
display: flex;
flex-direction: row;
justify-content: center;
gap: 20px;
}
```

App.js:

```
import React from 'react';
import CohortDetails from './Components/CohortDetails';
import './App.css';
```

```
function App() {  
  const cohorts = [  
    {  
      name: 'INTADMDF10 - .NET FSD',  
      startDate: '22-Feb-2022',  
      status: 'Scheduled',  
      coach: 'Aathma',  
      trainer: 'Jojo Jose',  
    },  
    {  
      name: 'ADM21JF014 - Java FSD',  
      startDate: '10-Sep-2021',  
      status: 'Ongoing',  
      coach: 'Apoorv',  
      trainer: 'Elisa Smith',  
    },  
    {  
      name: 'CDBJF21025 - Java FSD',  
      startDate: '24-Dec-2021',  
      status: 'Ongoing',  
      coach: 'Aathma',  
      trainer: 'John Doe',  
    },  
  ];  
  
  return (  
    <div className="outer-box">  
      <h2>Cohort Details</h2>  
  
      <div className="cohort-wrapper">
```

```

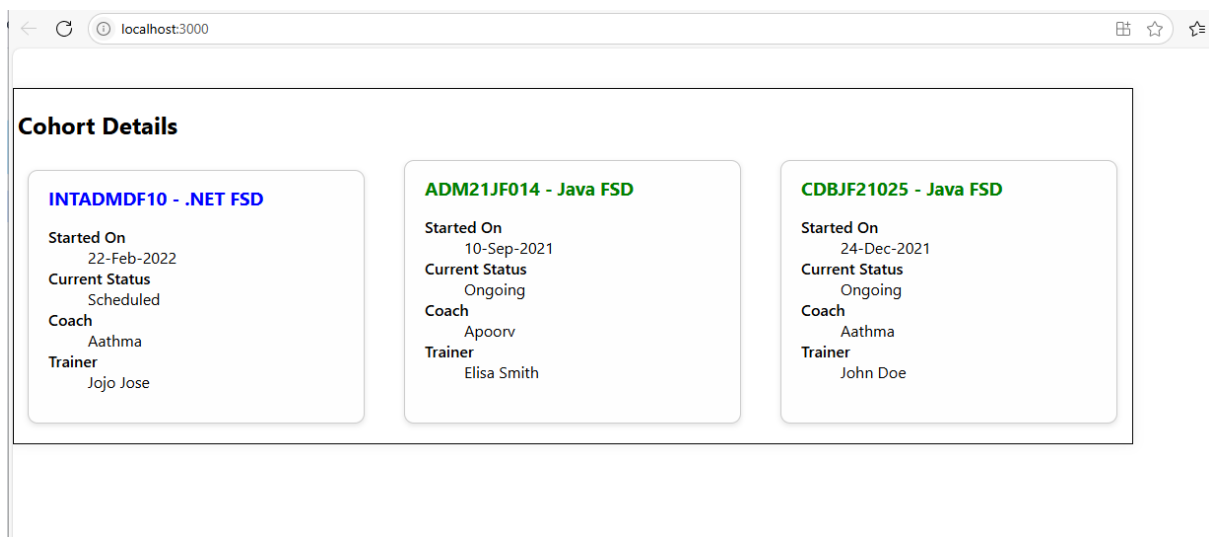
{cohorts.map((cohort, index) => (
  <div className="box" key={index}>
    <CohortDetails cohort={cohort} />
  </div>
)}}
</div>
</div>

);
}

```

export default App;

Output:



6. 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

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
 - a. TrainerId

- b. Name
- c. Email
- d. Phone
- e. Technology
- f. Skills

```
JS trainer.js U X
1 class Trainer {
2   constructor(trainerId, name, email, phone, technology, skills) {
3     this.trainerId=trainerId;
4     this.name=name;
5     this.email=email;
6     this.phone=phone;
7     this.technology=technology;
8     this.skills=skills;
9   }
10 }
11 export default Trainer;
```

Figure 9: Trainer.js

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

```
JS trainersmock.js U X
1 import Trainer from "../trainer";
2 const trainersMock = [
3   new Trainer('t-syed8',
4     'Syed Khaleelullah',
5     'khaleelullah@cognizant.com',
6     '97676516962',
7     '.NET',
8     ['C#', 'SQL Server', 'React', '.NET Core']),
9   new Trainer('t-jojo',
10     'Jojo Jose',
11     'jojo@cognizant.com',
12     '9897199231',
13     'Java',
14     ['Java', 'JSP', 'Angular', 'Spring']),
15   new Trainer('t-elisa',
16     'Elisa Jones',
17     'elisa@cognizant.com',
18     '9871212235',
19     'Python',
20     ['Python', 'Django', 'Angular'])
21 ]
22 export default trainersMock;
```

Figure 10: TrainersMock.js

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

```
C:\Windows\System32\cmd.exe
C:\CTS-NewHandsOns\ReactHandsOns\trainersapp>npm install react-router-dom@6
```

Figure 11: Install React Router

6. 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.

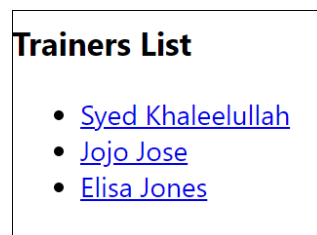


Figure 12: TrainersList Component

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



Figure 13: Home Component

8. 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 14: App Component

9. 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 15: Trainers Detail Component

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

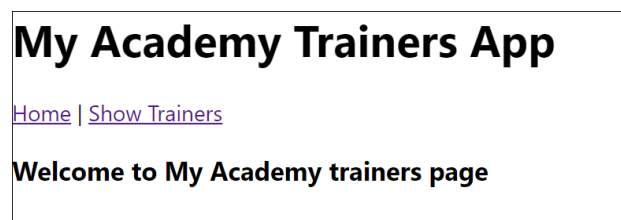


Figure 16: Home

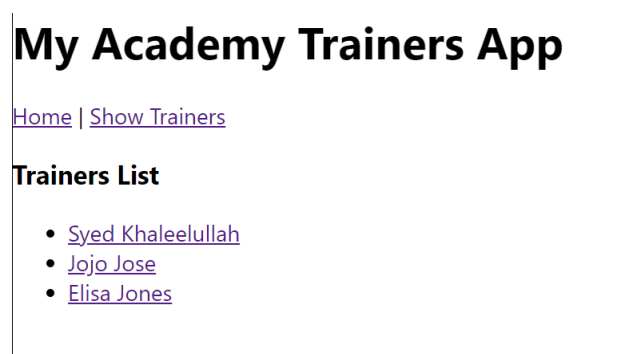


Figure 17: Trainers List

My Academy Trainers App

[Home](#) | [Show Trainers](#)

Trainers Details

Syed Khaleelullah (.NET)

khaleelullah@cognizant.com

97676516962

- C#
- SQL Server
- React
- .NET Core

Figure 18: Trainer Detail

Answer:

1. Need and Benefits of React Router

Need:

- React is a Single Page Application (SPA) framework, which doesn't support traditional page-based navigation by default.
- React Router allows navigation between components based on URL without reloading the page.

Benefits:

- Enables navigation without full page reload.
- Keeps the UI in sync with the URL.
- Allows deep linking and bookmarking.
- Supports dynamic routing and parameterized URLs.
- Enhances performance through route-based code splitting.

2. Components in React Router (v6)

- BrowserRouter: Wraps the application and enables routing using the browser's history API.
- Routes: Container for all Route elements.
- Route: Defines a path and associates it with a component.
- Link: Used for navigation; replaces anchor tags (<a>) for internal routing.
- Navigate: Used for redirection.
- useParams: Hook to access route parameters.

- `useNavigate`: Hook for navigating programmatically.

3. Types of Router Components

- `BrowserRouter`: Default for web apps; uses clean URLs.
- `HashRouter`: Uses URLs with hashes (e.g., `/#/home`); good for static sites.
- `MemoryRouter`: Keeps routing in memory; used in tests or non-browser environments.
- `StaticRouter`: Used for server-side rendering.

4. Parameter Passing via URL

Define route with parameter:

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

Access parameter:

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

const { id } = useParams();
```

Link to route with parameter:

```
<Link to="/user/123">User 123</Link>
```

Program:

Home.js:

```
import React from 'react';

const Home = () => {
  return (
    <div>
      <h2>Welcome to My Academy trainers page</h2>
    </div>
  );
};

export default Home;
```

trainer.js:

```
class Trainer {  
  constructor(trainerId, name, email, phone, technology, skills) {  
    this.trainerId = trainerId;  
    this.name = name;  
    this.email = email;  
    this.phone = phone;  
    this.technology = technology;  
    this.skills = skills;  
  }  
}  
  
export default Trainer;
```

TrainerDetails.js:

```
import React from 'react';  
import { useParams } from 'react-router-dom';  
import trainersMock from './TrainersMock';  
  
const TrainerDetails = () => {  
  const { id } = useParams();  
  const trainer = trainersMock.find(t => t.trainerId === id);  
  
  if (!trainer) {  
    return <h2>Trainer not found</h2>;  
  }  
  
  return (  
    <div>  
      <h2>Trainer Details</h2>  
      <h4>{trainer.name} {trainer.technology}</h4>  
      <p>{trainer.email}</p>  
    )  
  );  
}
```

```

    <p>{trainer.phone}</p>

    <ul>

      {trainer.skills.map((skill, index) => (
        <li key={index}>{skill}</li>

      ))}

    </ul>

  </div>

);
};

```

```
export default TrainerDetails;
```

TrainerList.js:

```

import React from 'react';
import { Link } from 'react-router-dom';

const TrainerList = ({ 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 TrainerList;

```


TrainersMock.js:

```
import Trainer from "./trainer";
```

```
const trainersMock = [
```

```
  new Trainer('t-syed8', 'Syed Khaleelullah', 'khaleelullah@cognizant.com', '97676516962', '.NET',  
  ['C#', 'SQL Server', 'React', '.NET Core']),
```

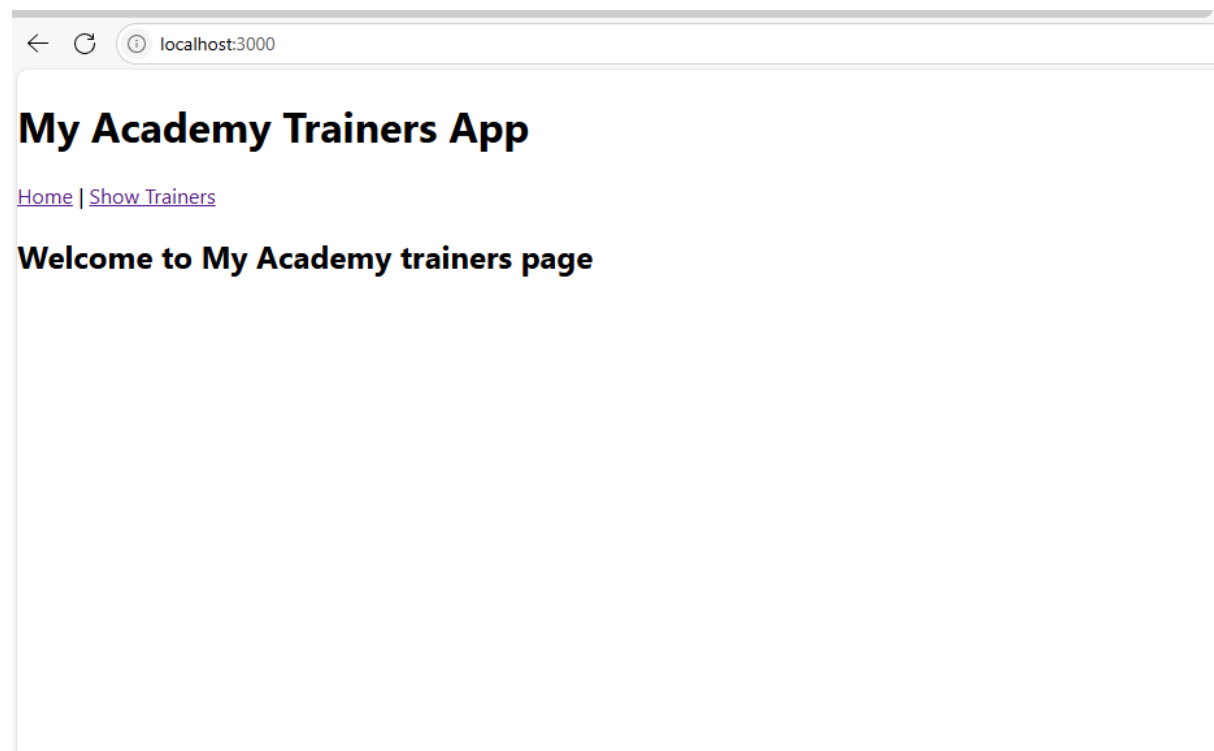
```
  new Trainer('t-jojo', 'Jojo Jose', 'jojo@cognizant.com', '9897199231', 'Java', ['Java', 'JSP', 'Angular',  
  'Spring']),
```

```
  new Trainer('t-elisa', 'Elisa Jones', 'elisa@cognizant.com', '9871212235', 'Python', ['Python',  
  'Django', 'Angular'])
```

```
];
```

```
export default trainersMock;
```

Output:



My Academy Trainers App

[Home](#) | [Show Trainers](#)

Trainers List

- [Syed Khaleelullah](#)
- [Jojo Jose](#)
- [Elisa Jones](#)

My Academy Trainers App

[Home](#) | [Show Trainers](#)

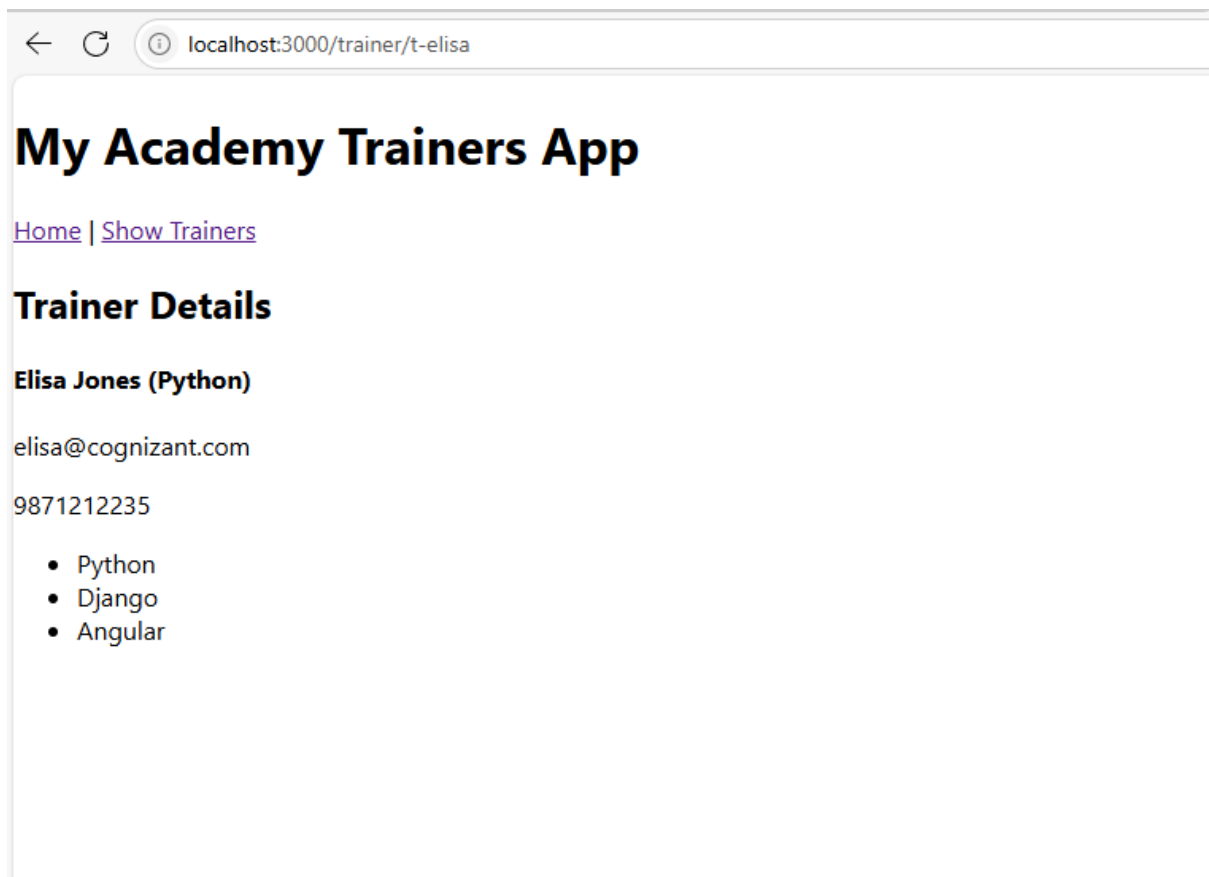
Trainer Details

Syed Khaleelullah (.NET)

khaleelullah@cognizant.com

97676516962

- C#
- SQL Server
- React
- .NET Core



7. Objectives

- Define Props
- Explain Default Props
- Identify the differences between State and Props
- Explain `ReactDOM.render()`

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

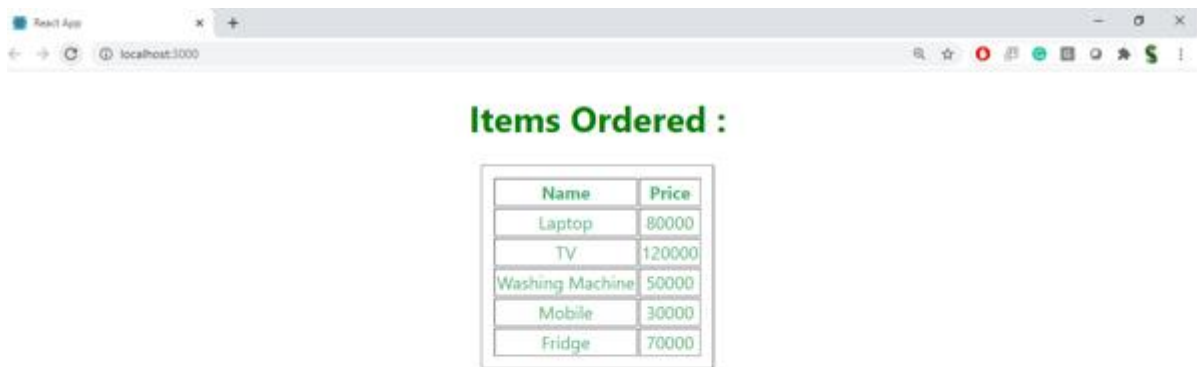
```
{this.props.item.map((item)=>
  {
    return(
      <tr>
        <td> {item.itemname} </td>
        <td> {item.price} </td>
      </tr>
    )}}
}
```

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

```
export class OnlineShopping extends React.Component {
  render() {
    const CartInfo =[{itemname:"Laptop", price: 80000},
                      {itemname:"TV",price:120000},
                      {itemname:"Washing Machine",price:50000},
                      {itemname:"Mobile",price:30000},
                      {itemname:"Fridge",price:70000}];

    return (
      <div className="mydiv">
        <h1>Items Ordered :</h1>
        <Cart item={CartInfo} />
      </div>
    );
  }
}
```

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



The screenshot shows a web browser window with the title 'React App' and the address bar displaying 'localhost:3000'. The main content of the page is a green heading 'Items Ordered :' followed by a table. The table has two columns: 'Name' and 'Price'. The rows of the table are: Laptop (80000), TV (120000), Washing Machine (50000), Mobile (30000), and Fridge (70000).

Name	Price
Laptop	80000
TV	120000
Washing Machine	50000
Mobile	30000
Fridge	70000

Answer:

1. Define Props

Props are read-only inputs passed from a parent component to a child component. They are used to pass data and configuration between components.

2. Explain Default Props

Default props are fallback values assigned to props when no value is explicitly provided by the parent component. They ensure that a component behaves predictably even if some props are missing.

3. Differences Between State and Props

Aspect	Props	State
Mutability	Immutable (read-only)	Mutable (can be changed)
Ownership	Passed from parent to child	Maintained within the component
Usage	Used for configuration	Used for internal data management
Access	Accessed externally	Accessed and modified internally

4. Explain ReactDOM.render()

ReactDOM.render() is a method used to render React components or elements into the DOM. It connects the virtual DOM structure with the actual HTML page by mounting the component to a specific DOM element.

App.js:

```
import React from 'react';
```

```
import './App.css';
```

```
class Cart extends React.Component {
```

```
  render() {
```

```
    return (
```

```
      <div className="outer-box">
```

```
        <table className="cart-table">
```

```
          <thead>
```

```
            <tr>
```

```
              <th>Name</th>
```

```
              <th>Price</th>
```

```
            </tr>
```

```
          </thead>
```

```
          <tbody>
```

```
            {this.props.item.map((item, index) => (
```

```
              <tr key={index}>
```

```
                <td>{item.itemname}</td>
```

```
                <td>{item.price}</td>
```

```
              </tr>
```

```
            )}}
```

```
          </tbody>
```

```
        </table>
```

```
      </div>
```

```
    );
```

```
  }
```

```
}
```

```
class OnlineShopping extends React.Component {  
  render() {  
    const CartInfo = [  
      { itemname: "Laptop", price: 80000 },  
      { itemname: "TV", price: 120000 },  
      { itemname: "Washing Machine", price: 50000 },  
      { itemname: "Mobile", price: 30000 },  
      { itemname: "Fridge", price: 70000 }  
    ];  
  
    return (  
      <div className="mydiv">  
        <h1 className="title">Items Ordered :</h1>  
        <Cart item={CartInfo} />  
      </div>  
    );  
  }  
}
```

```
export default OnlineShopping;
```

App.css:

```
.mydiv {  
  text-align: center;  
  margin-top: 30px;  
}
```

```
.title {  
  color: green;  
  font-weight: bold;  
}
```

```
.outer-box {  
  display: inline-block;  
  padding: 20px;  
  border: 2px solid rgb(160, 159, 159);  
  border-radius: 0px;  
}
```

```
.cart-table {  
  border-collapse: separate;  
  border-spacing: 2px 2px;  
  margin: auto;  
  text-align: center;  
}
```

```
.cart-table th,  
.cart-table td {  
  border: 1px solid rgb(160, 159, 159);  
  padding: 0.5px 2px;  
  border-radius: 0px;  
  font-weight: bold;  
}
```

```
.cart-table th {  
  box-shadow: 1px 1px 4px rgba(14, 14, 14, 0.4);  
  color: rgb(97, 177, 97);  
  background-color: white;  
}
```

```
.cart-table td {  
  box-shadow: 1px 1px 4px rgba(12, 12, 12, 0.4);  
  background-color: white;
```

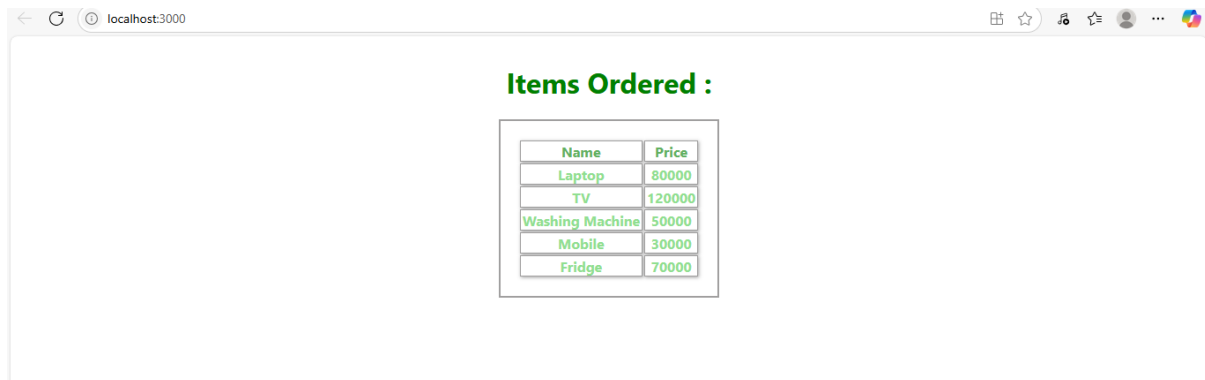


```
    color: rgb(144, 223, 144);  
  }  
}
```

Index.js:

```
import React from 'react';  
import ReactDOM from 'react-dom/client';  
import './index.css';  
import OnlineShopping from './App';  
  
const root = ReactDOM.createRoot(document.getElementById('root'));  
root.render(<OnlineShopping />);
```

Output:



8. Objectives

- Explain React State

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

```

constructor() {
  super();
  this.state = {
    entrycount: 0,
    exitcount: 0,
    c: 0
  };
}

updateEntry() {
  this.setState((prevState, props) => {
    return { entrycount: prevState.entrycount + 1 }
  });
}

updateExit()
{
  this.setState((prevState, props) =>
  {
    return { exitcount: prevState.exitcount + 1 }
  })
}

```

The output should be as follows:



Answer:

React state:

React State is used to store information inside a component that can change over time. When the state changes, React automatically updates the screen. It's useful for handling things like clicks, form inputs, or counters.

Program:

App.js:

```
import React from 'react';

class CountPeople extends React.Component {

  constructor() {

    super();

    this.state = {

      entrycount: 0,

      exitcount: 0

    };

  }


  updateEntry = () => {

    this.setState((prevState) => ({

      entrycount: prevState.entrycount + 1

    }));

  };


  updateExit = () => {

    this.setState((prevState) => ({

      exitcount: prevState.exitcount + 1

    }));

  };


  render() {

    const buttonStyle = {

      backgroundColor: 'lightgreen',

      border: '2px solid black',

      borderRadius: '5px',
```

```
padding: '10px 15px',  
fontSize: '16px',  
fontWeight: 'bold',  
cursor: 'pointer'  
};
```

```
const wrapperStyle = {  
  display: 'flex',  
  justifyContent: 'center',  
  alignItems: 'center',  
  gap: '300px',  
  marginTop: '192px',  
  marginLeft: '60px'  
};
```

```
const pairStyle = {  
  display: 'flex',  
  alignItems: 'center',  
  gap: '10px',  
  fontSize: '20px',  
  textAlign: 'center'  
};
```

```
return (  
  <div style={wrapperStyle}>  
    <div style={pairStyle}>  
      <button onClick={this.updateEntry} style={buttonStyle}>Login</button>  
      <span>{this.state.entrycount} People Entered!!!!</span>  
    </div>  
  
    <div style={pairStyle}>
```

```
        <button onClick={this.updateExit} style={buttonStyle}>Exit</button>

        <span>{this.state.exitcount} People Left!!!</span>

    </div>

</div>

);
}
}
```

```
export default CountPeople;
```

App.css:

```
.App {
  text-align: center;
}
```

```
.App-logo {
  height: 40vmin;
  pointer-events: none;
}
```

```
@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}
```

```
.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
```

```
justify-content: center;
font-size: calc(10px + 2vmin);
color: white;
}
```

```
.App-link {
  color: #61dafb;
}
```

```
@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}
```

Index.js:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import CountPeople from './App'; // make sure you renamed the default export
```

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <CountPeople />
  </React.StrictMode>
);
```

Output:

