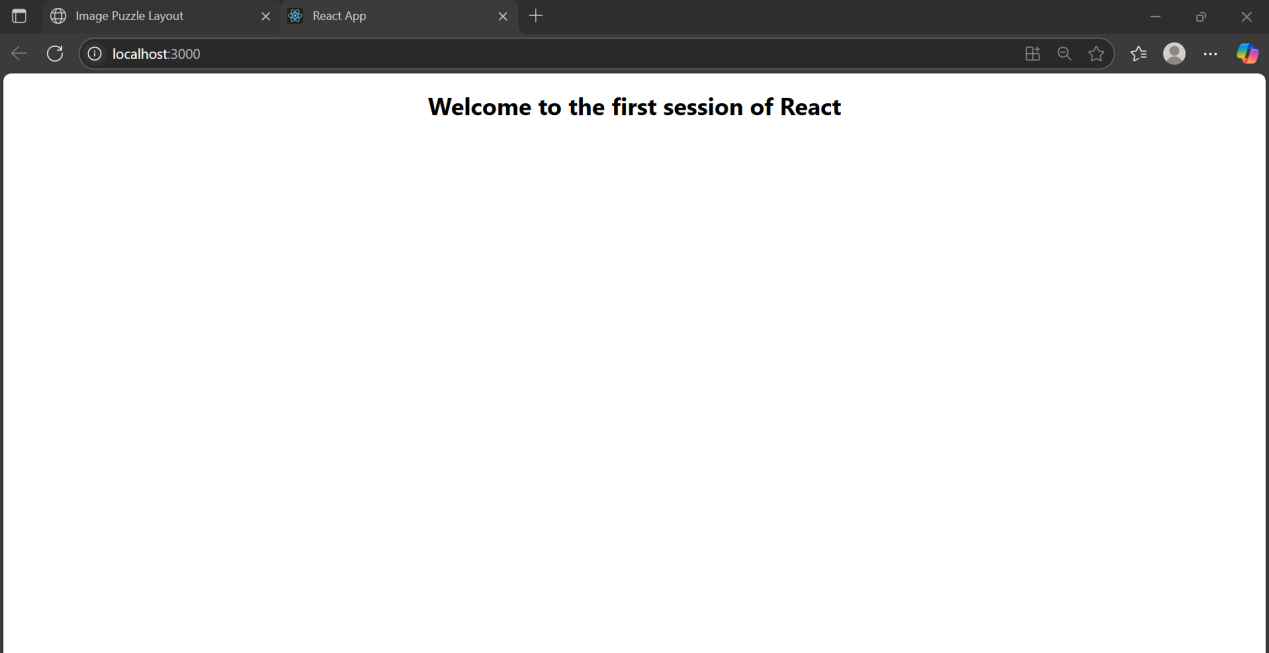
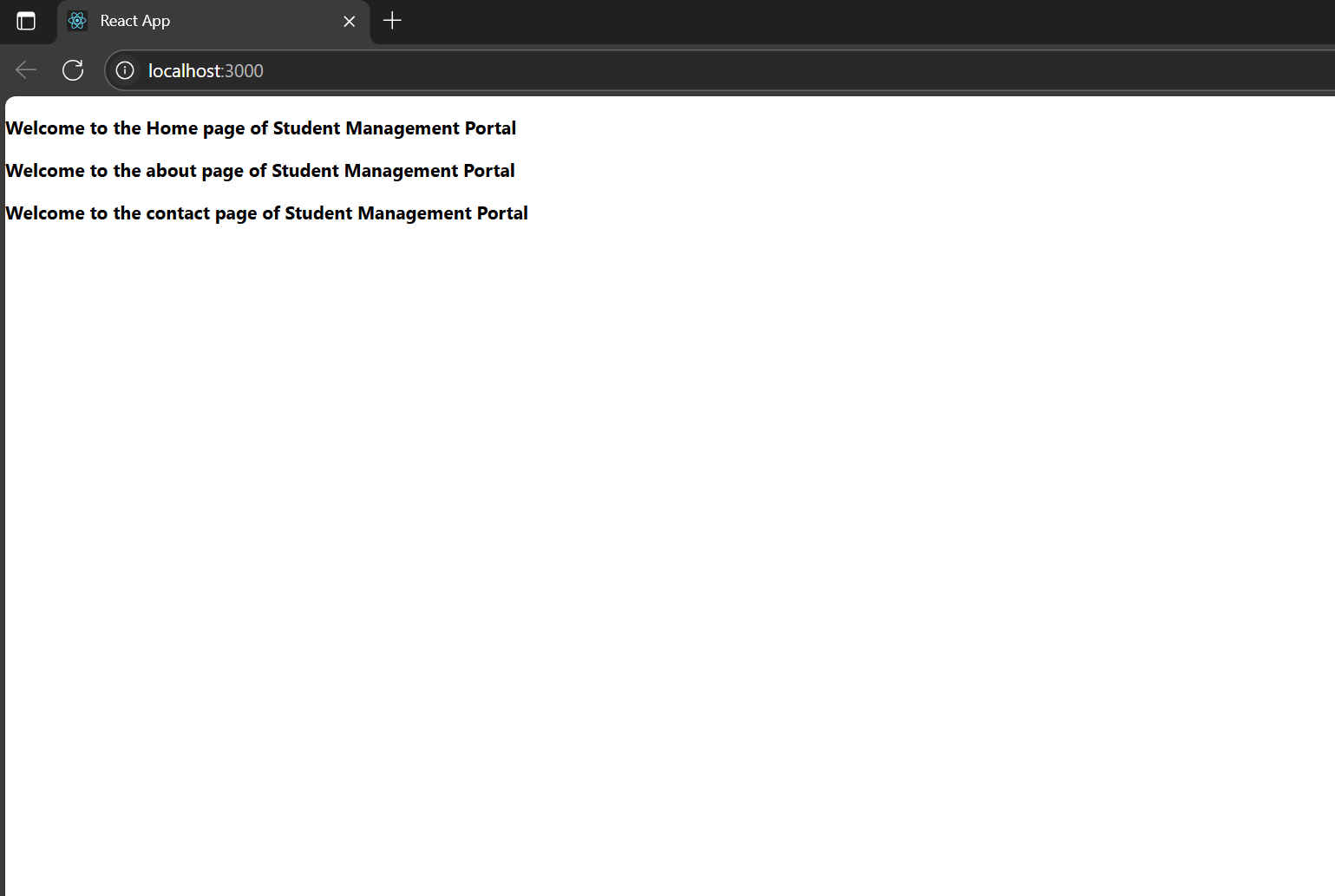
Output week - 6

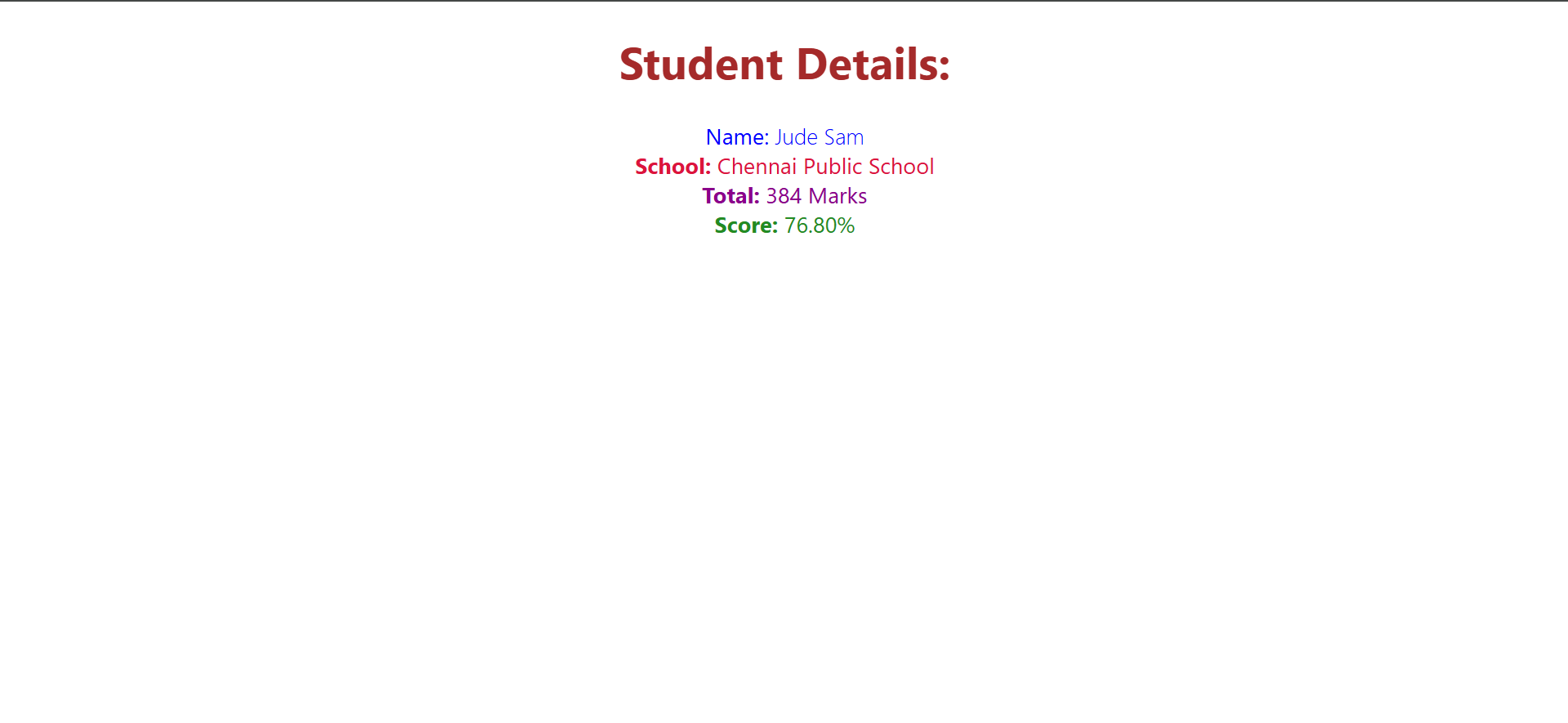
LAB 1



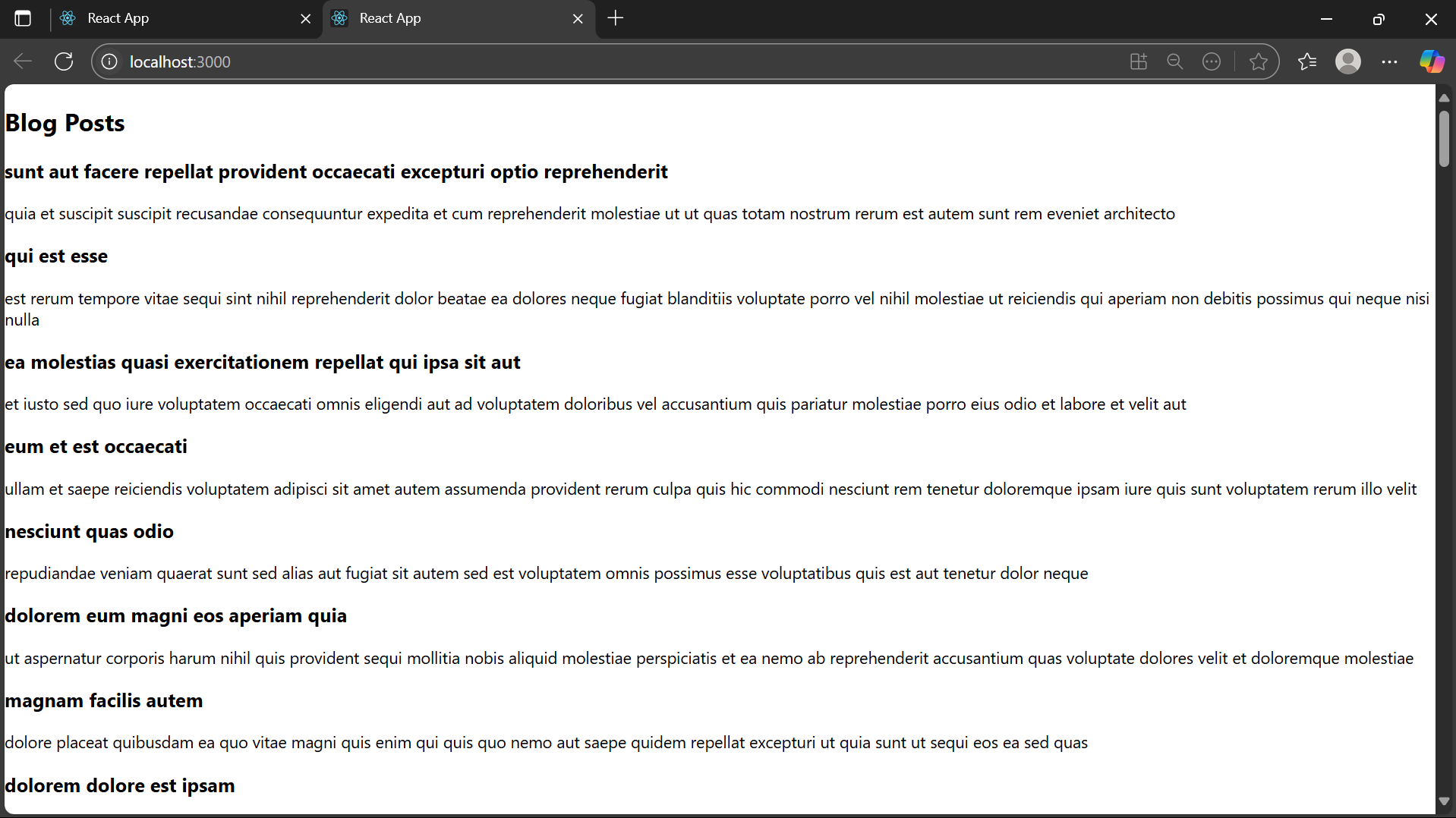
LAB 2



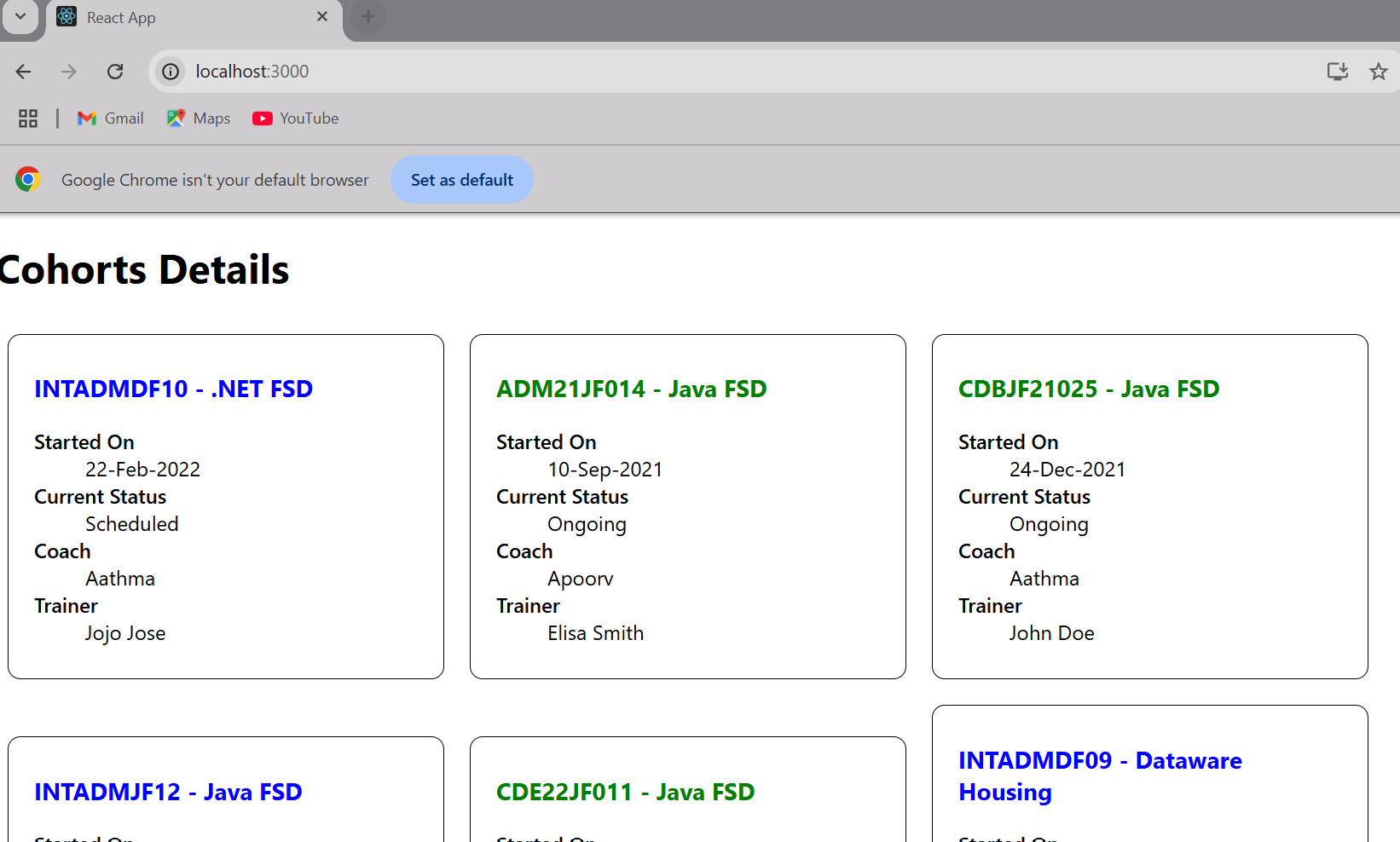
Lab 3 Scorecalculator



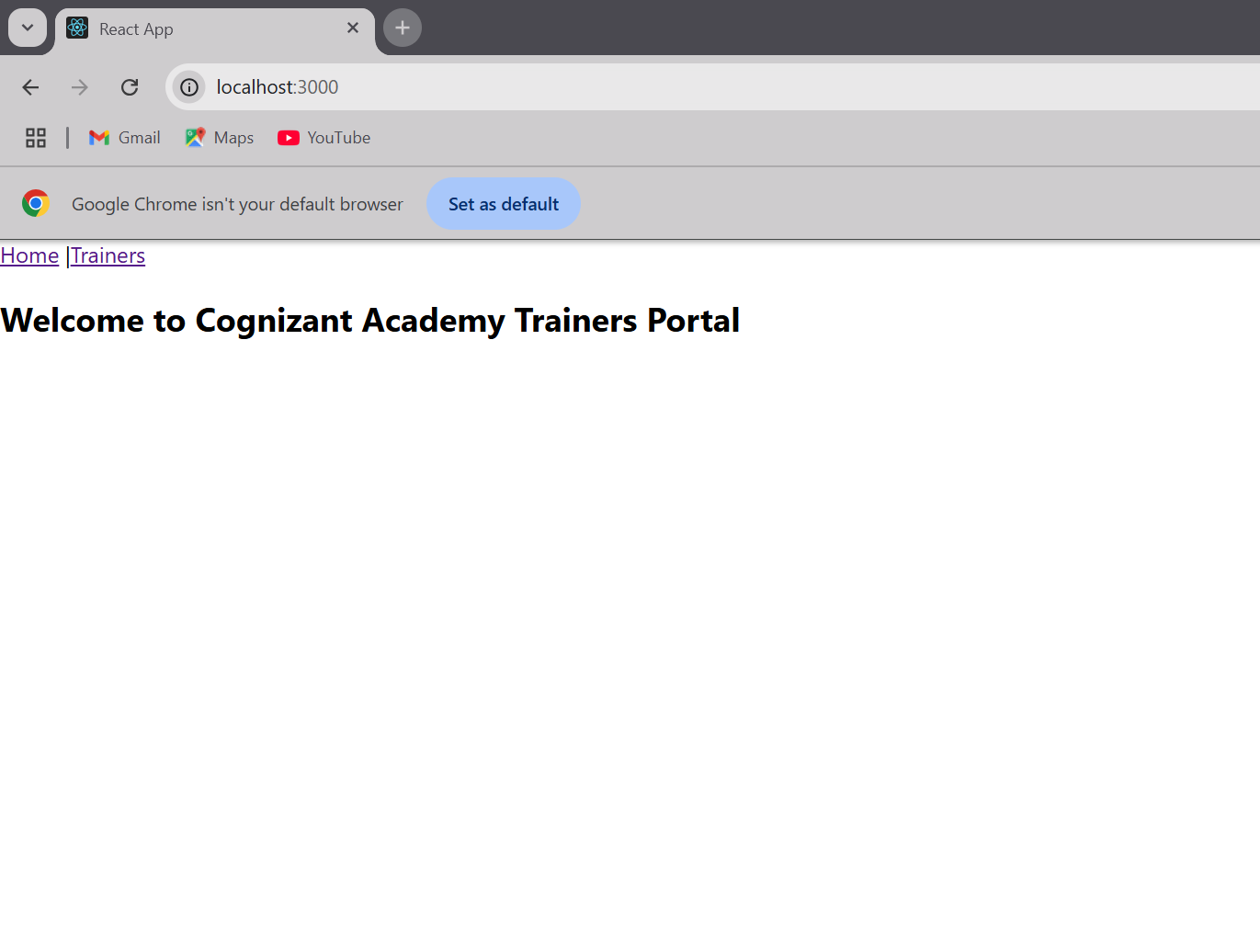
Lab 4 Blog



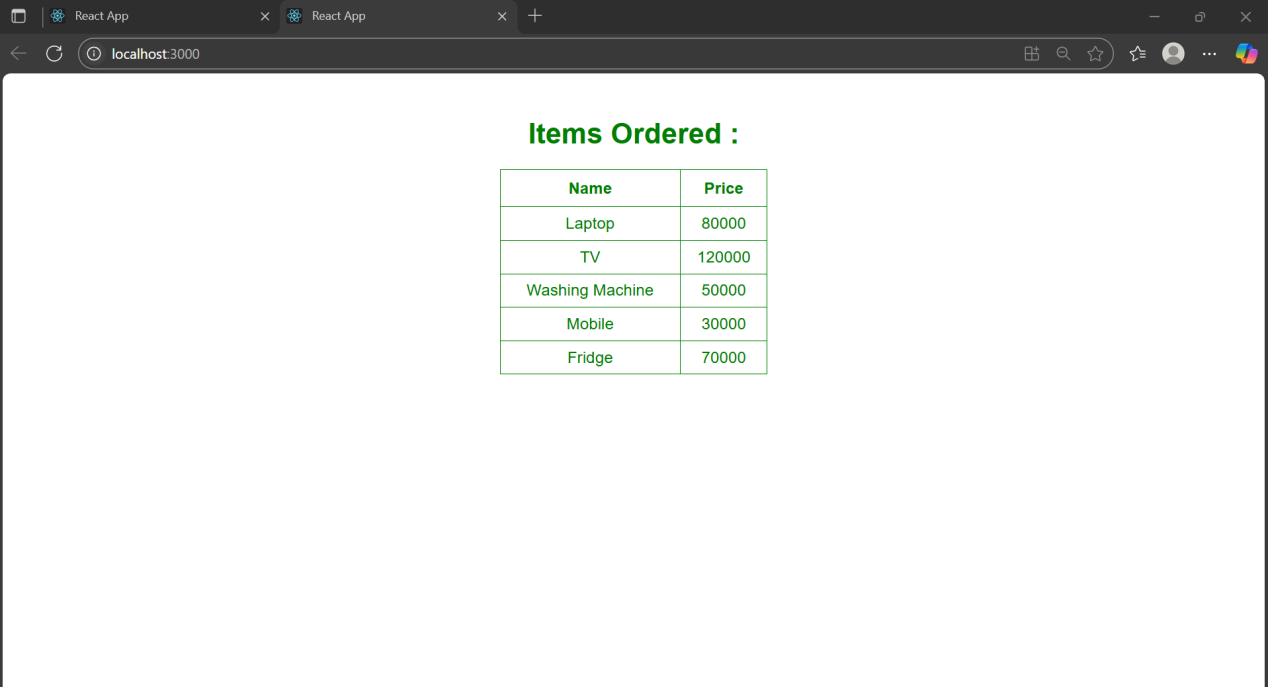
Lab 5 cohort details



Lab 6



Lab 7



Lab 8

## WEEK 6 MANDATORY HANDS ON

## 1. ReactJS-HOL

## Objectives

* **Define SPA and its benefits**

A Single Page Application (SPA) is a web application that loads a single HTML page and dynamically updates the content as the user interacts with the app, without requiring a full page reload. This approach provides a smoother, faster, and more responsive user experience, similar to desktop applications. SPAs use technologies like AJAX, JavaScript frameworks (such as React, Angular, or Vue.js), and RESTful APIs or GraphQL to fetch and render data dynamically. The key benefits of SPAs include faster page load times, reduced server load, seamless user interactions, easier debugging with modern tools, and improved code reuse between mobile and web platforms.

* **Define React and identify its working**

**React** is a popular open-source JavaScript library developed by Facebook for building user interfaces, especially for single-page applications. It allows developers to create large web applications that can update and render efficiently in response to data changes. React works by using a **component-based architecture**, where the UI is broken down into reusable, self-contained components. It uses a **virtual DOM (Document Object Model)** to track changes in the UI. When the state of a component changes, React updates the virtual DOM first, compares it with the previous version (a process called **reconciliation**), and then efficiently updates only the parts of the real DOM that have changed. This results in faster rendering and a smoother user experience.

* **Identify the differences between SPA and MPA**

Single Page Application (SPA) and Multi Page Application (MPA) are two different architectures for building web applications. Here are the key differences between them:

1. Page Loading:
   * SPA: Loads a single HTML page and updates content dynamically without refreshing the whole page.
   * MPA: Loads a new HTML page from the server every time the user navigates to a different page.
2. User Experience:
   * SPA: Provides a smoother and faster experience since only parts of the page update.
   * MPA: Slower due to full page reloads on each user action or navigation.
3. Speed:
   * SPA: Faster after the initial load, as resources are reused.
   * MPA: Slower overall due to repeated server requests for each page.
4. Development Complexity:
   * SPA: Requires more client-side JavaScript and routing management.
   * MPA: Easier to manage SEO and traditional server-side routing.
5. SEO (Search Engine Optimization):
   * SPA: More complex SEO setup; needs server-side rendering or prerendering for indexing.
   * MPA: Better SEO by default, as each page is a separate HTML file.
6. Examples:
   * SPA: Gmail, Facebook, Twitter.
   * MPA: Amazon, LinkedIn, Wikipedia.

* **Explain Pros & Cons of Single-Page Application**

Single-Page Applications (SPAs) have their own set of advantages and disadvantages depending on the use case. Here's a balanced explanation of the pros and cons:

Pros of SPA:

1. Fast and Responsive: Once loaded, only data is fetched from the server, not entire pages, resulting in a faster and smoother user experience.
2. Improved User Experience: Navigation feels seamless and app-like, as there are no page reloads.
3. Reduced Server Load: Since only data is requested after the initial load, server requests are smaller and more efficient.
4. Reusable Components: Built using frameworks like React or Angular, SPAs allow for modular and reusable UI components, improving development efficiency.
5. Rich Client-Side Features: Enables offline support and interactive UI features using JavaScript.

Cons of SPA:

1. SEO Limitations: Since content is loaded dynamically, search engines may struggle to index SPA content unless server-side rendering or prerendering is implemented.
2. Initial Load Time: The first-time load may be slower because the entire JavaScript bundle has to be downloaded before rendering.
3. Browser Compatibility: Heavily relies on JavaScript, so users with disabled JavaScript will not be able to use the app.
4. Complex State Management: Managing the client-side application state and routing can become complex in large-scale SPAs.
5. Security Concerns: SPAs are more vulnerable to cross-site scripting (XSS) attacks if not properly secured

* **Explain about React**

React is an open-source JavaScript library developed by Facebook for building dynamic and interactive user interfaces, especially for single-page applications. It follows a component-based architecture, meaning the UI is broken into small, reusable components that manage their own state and logic. One of React’s core features is the virtual DOM, which helps optimize performance by updating only the parts of the real DOM that have changed, rather than reloading the entire page. React uses JSX (JavaScript XML), a syntax extension that allows writing HTML-like code inside JavaScript, making code more readable and easier to maintain. React is widely used because of its performance, scalability, and flexibility, and it's supported by a strong ecosystem including tools like React Router, Redux, and Next.js for advanced features like routing, state management, and server-side rendering.

* **Define virtual DOM**

The **Virtual DOM (Document Object Model)** is a concept used in libraries like **React** to improve the performance and efficiency of web applications. It is a lightweight, in-memory copy of the actual DOM. When changes are made to the UI, React first updates the virtual DOM instead of directly modifying the real DOM. It then compares the new virtual DOM with the previous one using a process called **"diffing"** to identify what has changed. Finally, it updates only the changed parts in the real DOM. This approach makes updates faster and more efficient, resulting in better performance and a smoother user experience.

* **Explain Features of React**

Here are the key features of React that make it a powerful library for building user interfaces:

1. Component-Based Architecture:  
   React breaks the UI into reusable, self-contained components, making code modular, maintainable, and easier to test.
2. Virtual DOM:  
   React uses a virtual DOM to optimize performance. It minimizes direct manipulation of the real DOM by updating only the elements that have changed.
3. JSX (JavaScript XML):  
   JSX allows developers to write HTML-like code within JavaScript, making the structure of UI components more readable and easier to understand.
4. One-Way Data Binding:  
   React follows unidirectional data flow, meaning data flows from parent to child components. This makes it easier to debug and track changes in the app.
5. Declarative UI:  
   React allows developers to describe what the UI should look like for different states, and React takes care of rendering and updating it when the data changes.
6. React Hooks:  
   Hooks (like useState, useEffect) let you use state and lifecycle features in functional components, simplifying the code and logic handling.
7. React Native:  
   React has a sister library called React Native, which allows developers to build mobile apps using the same React concepts for Android and iOS.
8. Strong Community and Ecosystem:  
   React has a large community, extensive documentation, and a wide range of tools and libraries like Redux, React Router, and Next.js to extend its capabilities.

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/>

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

**2. React JS - HOL**

**Objectives**

* **Explain React components**

React components are the building blocks of a React application. They are reusable, independent pieces of UI that manage their own structure, logic, and styling. Each component can be thought of as a JavaScript function or class that returns a portion of the user interface, usually written using JSX. Components can be nested inside other components, allowing complex UIs to be built from simple, manageable pieces. They can also manage internal state and handle user interactions, making them essential for dynamic and interactive web applications.

* **Identify the differences between components and JavaScript functions**

While both React components and JavaScript functions are functions in nature, they serve different purposes. Regular JavaScript functions are used to perform specific operations and return values like numbers or strings. React components, on the other hand, return JSX elements that describe what the UI should look like. Additionally, React components can manage state and use lifecycle methods (in class components) or hooks (in function components), while ordinary JavaScript functions do not have these capabilities. React components also follow specific naming conventions (PascalCase) and integration rules within the React ecosystem.

* **Identify the types of components**

React components are mainly classified into two types: **Class Components** and **Function Components**. Class components are ES6 classes that extend React.Component and include lifecycle methods like componentDidMount and a render() method. Function components, initially used only for simple, stateless components, can now manage state and side effects using hooks like useState and useEffect. Function components are more concise and are now the preferred approach in modern React development due to their simplicity and improved performance.

* **Explain class component**

A class component in React is a traditional way of creating components using ES6 class syntax. It extends the React.Component base class and must include a render() method that returns JSX to define the UI. Class components have access to lifecycle methods such as componentDidMount, componentDidUpdate, and componentWillUnmount, which allow developers to control component behavior during different stages of its lifecycle. They also manage state using this.state and update it using this.setState(). Although still valid, class components are less commonly used in newer React projects.

* **Explain function component**

Function components are simpler and more concise React components defined as JavaScript functions. They accept props as arguments and return JSX to describe what the UI should look like. Initially limited to stateless use cases, function components gained full capabilities with the introduction of **React Hooks**, such as useState for state management and useEffect for side effects. Today, function components are widely preferred due to their simplicity, better readability, and support for modern React features.

* **Define component constructor**

The **constructor** in a React class component is a special method used to initialize the component’s state and bind event handlers. It is called before the component is mounted and is the right place to set up this.state and perform any initial setup. Inside the constructor, the super(props) method must be called first to correctly inherit from React.Component and to have access to this.props. It is not used in function components, which use hooks instead for state and logic handling.

* **Define render() function**

The render() function is a required method in every React class component. It returns the JSX that defines the structure and appearance of the component's UI. The render() method is called automatically by React during the component’s lifecycle and is re-invoked whenever the component’s state or props change. It must return a single parent element (or use a fragment) that contains all child elements. Unlike class components, function components don’t require a separate render() method; instead, they return JSX directly.

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:

****

**3. React JS - HOL**

**Objectives**

* **Explain React components**

React components are the core building blocks of any React application. They allow developers to divide the user interface into reusable and independent pieces, making it easier to manage and maintain the application. Each component represents a part of the UI and can manage its own logic, structure, and styling. Components can also accept inputs called "props" and maintain internal data called "state", making them dynamic and interactive.

* **Identify the differences between components and JavaScript functions**

While React components and JavaScript functions may look similar, they serve different purposes. A regular JavaScript function performs a task and returns a value like a number or a string. In contrast, a React component returns JSX, which describes what the UI should look like. React components can also manage state, handle lifecycle events, and interact with the DOM — features that normal JavaScript functions don’t have. Additionally, components follow specific naming conventions and are designed to be used as building blocks in a React app.

* **Identify the types of components**

There are mainly two types of React components: **Class Components** and **Function Components**. Class components are created using ES6 classes and provide full access to React’s features, including lifecycle methods and state management using this.state. Function components are simpler and defined using JavaScript functions. With the introduction of React Hooks, function components can now manage state and side effects, making them just as powerful as class components and the preferred approach in modern React development.

* **Explain class component**

A class component is a type of React component defined using a JavaScript ES6 class that extends React.Component. It must include a render() method that returns JSX to define the UI. Class components can maintain their own internal state using this.state and update it using this.setState(). They also have access to lifecycle methods such as componentDidMount and componentDidUpdate, allowing developers to run code at specific points in a component’s life. Though powerful, class components are being used less in favor of function components with hooks.

* **Explain function component**

A function component is a simpler way to write React components using JavaScript functions. It receives props as an argument and returns JSX to describe the UI. Initially, function components were stateless, but with the introduction of React Hooks like useState and useEffect, they can now handle state and side effects. Function components are easier to write, understand, and test, which is why they are widely used in modern React applications.

* **Define component constructor**

The constructor is a special method in React class components used to initialize the component’s state and bind methods. It is called before the component is mounted to the DOM. Inside the constructor, super(props) must be called to correctly set up the component and allow access to this.props. It’s typically used to set the initial state using this.state and to bind event handler methods to the component instance. Constructors are not used in function components.

* **Define render() function**

The render() function is a required method in every React class component. It returns the JSX that describes what the UI should look like. This function is automatically called by React whenever the component’s state or props change, triggering a re-render. The render() method must return a single root element, which can contain multiple child elements. In function components, the JSX is returned directly from the function, so a separate render() method is not needed.

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:

**4. React JS – HOL**

**Objectives**

* **Explain the need and Benefits of component life cycle**

The **component lifecycle** in React refers to the sequence of methods that are invoked at different stages of a component's existence — from its creation and mounting, to updating, and finally unmounting. Understanding the component lifecycle is essential because it allows developers to execute code at specific points during a component’s life. For example, you may need to fetch data from an API when the component is first displayed or clean up event listeners before it is removed from the DOM. The benefits of using lifecycle methods include better **control over rendering**, **optimized performance**, **efficient resource management**, and the ability to **hook into specific moments** like component initialization, updates, or teardown.

* Identify various life cycle hook methods

React class components provide several lifecycle hook methods that can be grouped into three main phases:

1. Mounting (when the component is created and inserted into the DOM):
   * constructor()
   * static getDerivedStateFromProps()
   * render()
   * componentDidMount()
2. Updating (when the component is re-rendered due to changes in props or state):
   * static getDerivedStateFromProps()
   * shouldComponentUpdate()
   * render()
   * getSnapshotBeforeUpdate()
   * componentDidUpdate()
3. Unmounting (when the component is removed from the DOM):
   * componentWillUnmount()  
     In function components, similar lifecycle behavior is achieved using React Hooks like useEffect().

* **List the sequence of steps in rendering a component**

The typical sequence of steps during the initial rendering (mounting) of a React class component is as follows:

1. constructor() – Initializes the state and binds methods.
2. getDerivedStateFromProps() – Called right before rendering to update state from props if needed.
3. render() – Returns JSX to display the UI.
4. componentDidMount() – Invoked after the component is mounted; ideal for side effects like API calls.

During updating (when props/state change), the order is:

1. getDerivedStateFromProps()
2. shouldComponentUpdate()
3. render()
4. getSnapshotBeforeUpdate()
5. componentDidUpdate()

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.

**5. React JS - HOL**

**Objectives**

* **Understanding the need for styling react component**

Styling React components is essential for creating visually appealing, user-friendly, and well-structured web applications. Since React breaks the UI into reusable components, it's important to style each component individually to maintain consistency and encapsulation. Proper styling ensures that each component looks and behaves as intended, improves usability, and enhances the overall user experience. React allows various styling approaches such as traditional CSS files, **CSS Modules**, **inline styles**, and **CSS-in-JS libraries** like styled-components. Choosing the right method helps manage styles efficiently, prevent global CSS conflicts, and support scalable UI development.

* **Working with CSS Module and inline styles**

**CSS Modules** are a way to locally scope CSS by default in React. When you import a CSS file as a module (e.g., import styles from './MyComponent.module.css'), each class name is scoped to the component, avoiding naming conflicts and ensuring modular design. You then apply styles using className={styles.className}.

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

**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

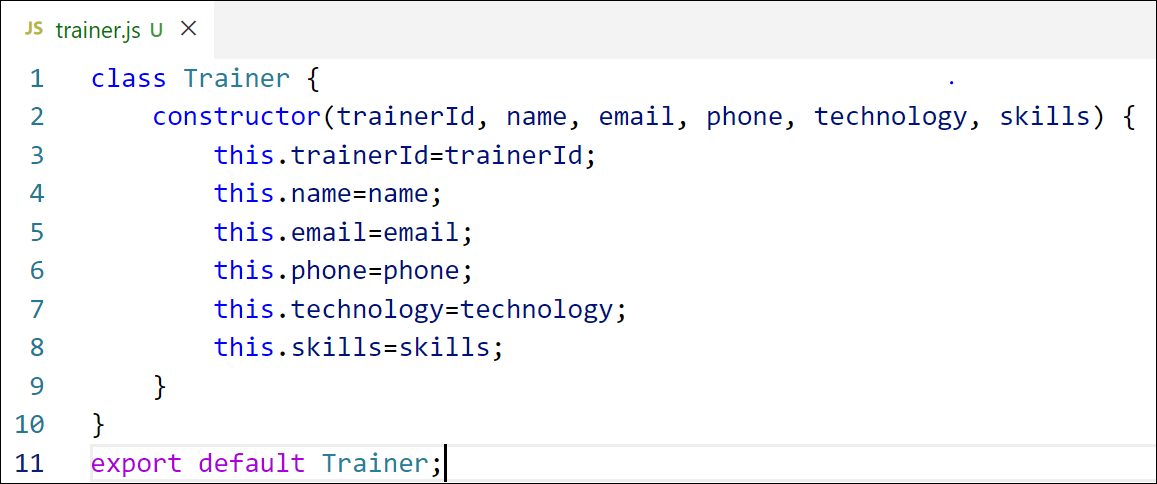


Figure 2: Trainer.js

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

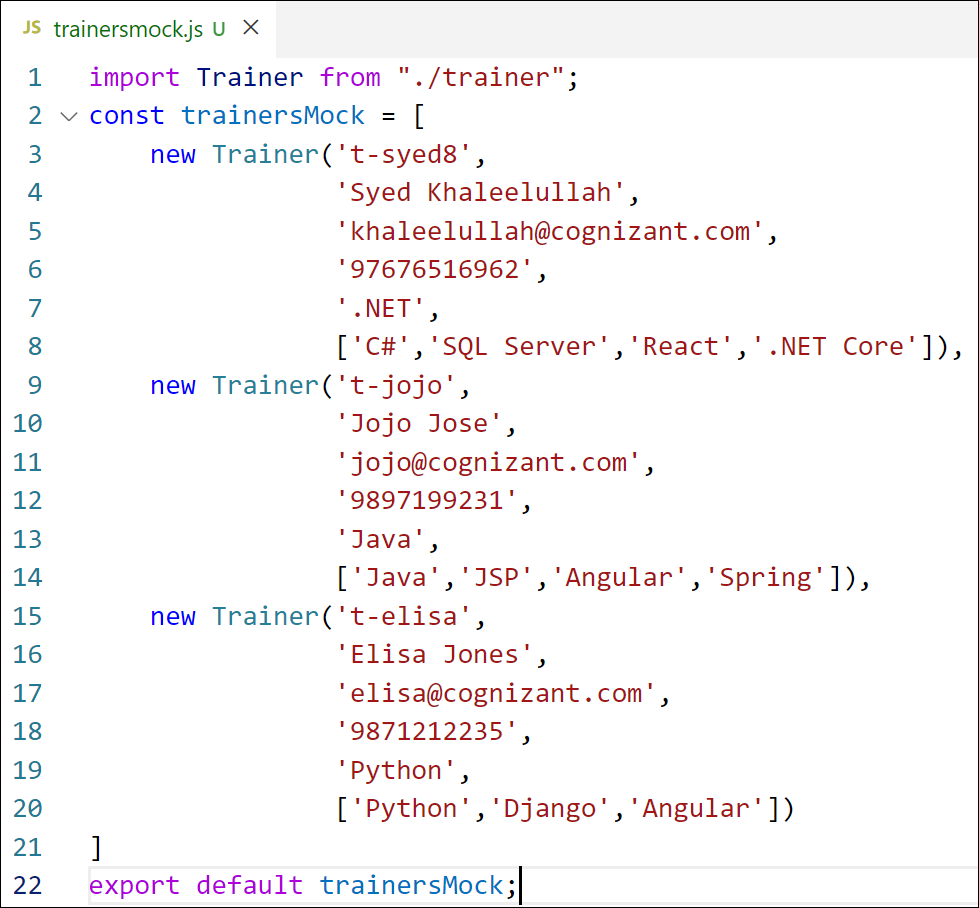


Figure 3: TrainersMock.js

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

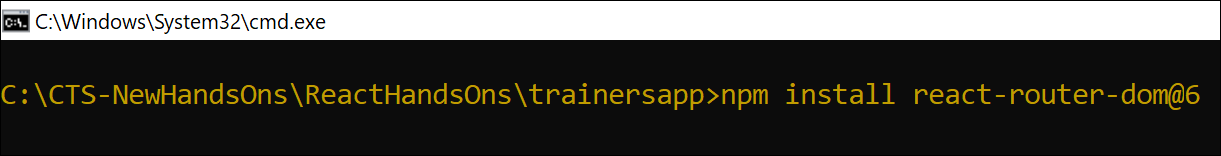


Figure 4: Install React Router

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.

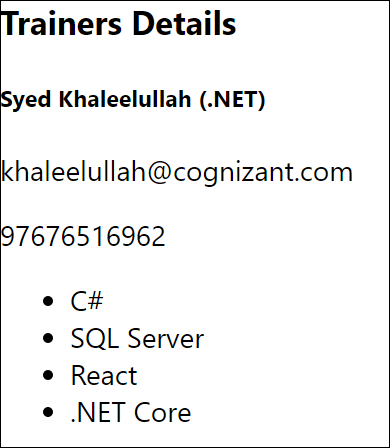


Figure 8: Trainers Detail Component

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



Figure 9: Home

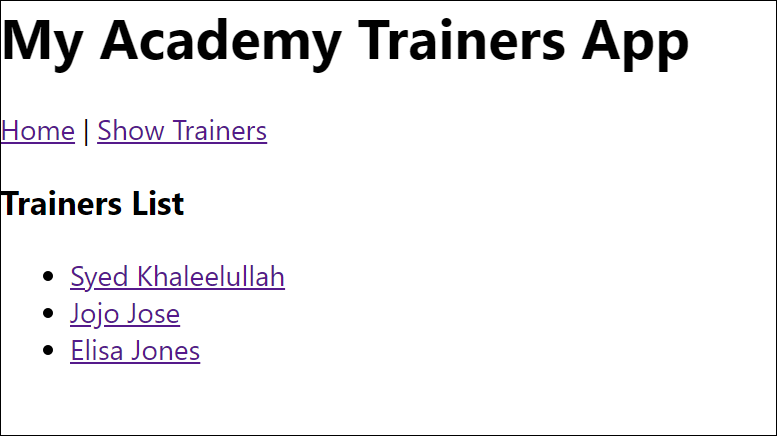


Figure 10: Trainers List

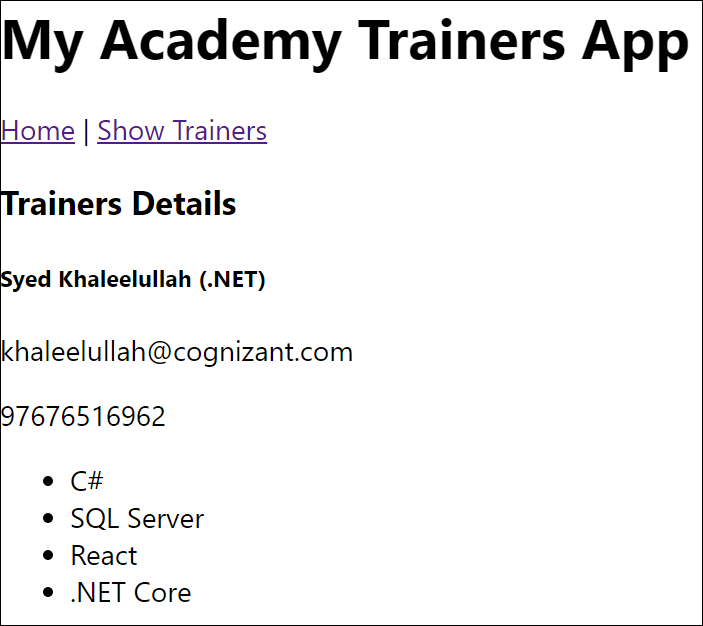


Figure 11: Trainer Details

**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

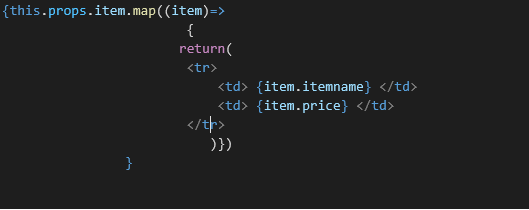
**Notes**

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

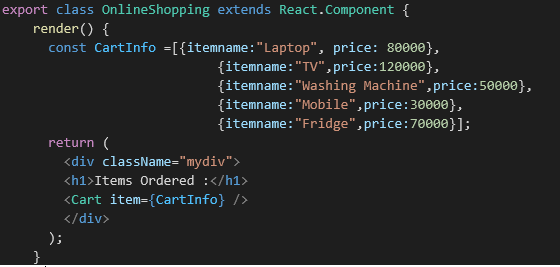
Create a React Application named “shoppingapp” with a class component named “OnlineShopping” and “Cart”.

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

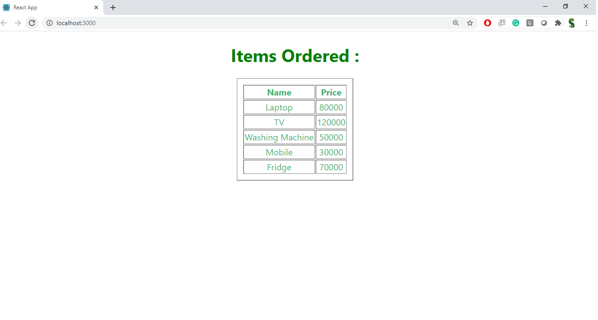
* Itemname
* Price



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:



**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.

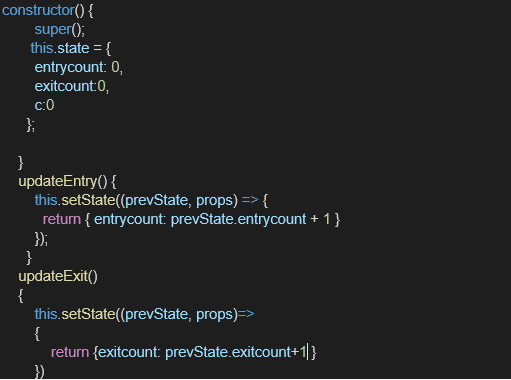
UpdateEntry() 🡪 which will display the number of people who entered the mall.

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:

