**Module : ReactJS Hands-On Lab Guide**

**Hands On 1 Setup React Development Environment**

**Overview:**

ReactJS is a JavaScript library for building user interfaces. It is maintained by Facebook and a community of developers. React encourages the creation of reusable UI components and offers high performance through the use of a virtual DOM.

**Learning Objectives:**

* Define SPA and its benefits
* Define React and explain how it works
* Identify the differences between SPA and MPA
* List the pros and cons of Single-Page Applications
* Describe React's core features
* Explain the concept of the virtual DOM

**Theory:**

**Single Page Applications (SPA):**

**SPAs** load a single HTML page and dynamically update the content as the user interacts with the app.

**Benefits:**

* Faster navigation
* Better user experience
* Reduced server load

**Multi Page Applications (MPA):**

**MPAs** load a new page from the server every time a user navigates.

Comparison with SPA:

* Higher server load
* Slower navigation

**Introduction to React:**

**React** is a component-based library used to build user interfaces. It emphasizes reusable components and one-way data binding.

**Virtual DOM):**

A lightweight representation of the actual **DOM**. React uses it to update the UI efficiently.

**Benefits:**

* Improved performance
* Efficient UI updates

**Features of React:**

* Declarative UI
* Component-Based Architecture
* Learn Once, Write Anywhere

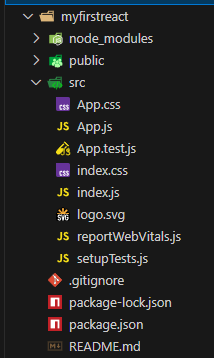
**Hands-On Lab:**

* created a new React app, Install Nodejs and Npm from the following link:
  + - * <https://nodejs.org/en/download/>
* Installed Create-react-app by running the following command in the command prompt:

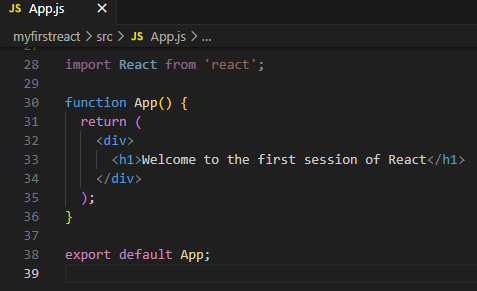


* To create a React Application with the name of “myfirstreact”, type the following command:

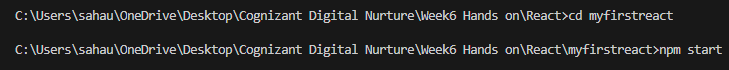




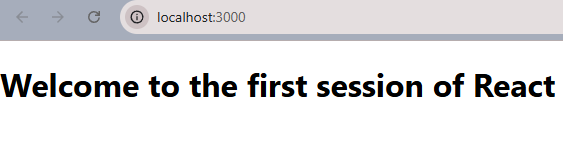
* Once the App is created, navigate into the folder of myfirstreact by typing the following command:
* Open the folder of myfirstreact in Visual Studio Code
* Open the App.js file in Src Folder of myfirstreact
* Remove the current content of “App.js”
* Replace it with the following:



* Run the following command to execute the React application:



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



**Hands On 2: React App for Student Management Portal**

**Overview:**

React components are the building blocks of any React application. They let you split the UI into independent, reusable pieces that can be processed separately.

**Learning Objectives:**

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

**Theory:**

**React Components:**

**SPAs** load a single HTML page and dynamically update the content as the user interacts with the app.

**Benefits:**

* Faster navigation
* Better user experience
* Reduced server load

**Components vs. JavaScript Functions:**

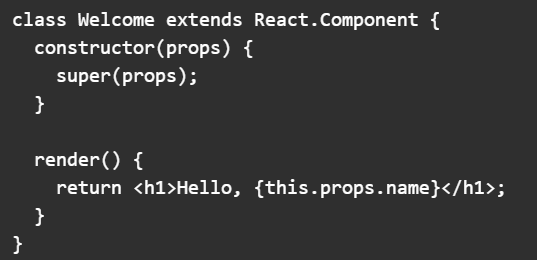
While both are functions, React components return JSX and are used in the UI context, whereas regular JavaScript functions perform logic or return values.

**Types of Components:**

* **Class Components:** ES6 classes that extend React.Component and must include a render() method.
* **Function Components:** Plain JavaScript functions that return JSX.

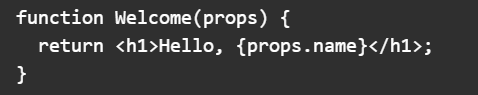
**Class Component:**

Includes constructor and lifecycle methods.

****

**Function Component:**

Simpler and recommended for most use cases.



**Constructor:**

Used in class components to initialize state and bind methods.

**render() Function:**

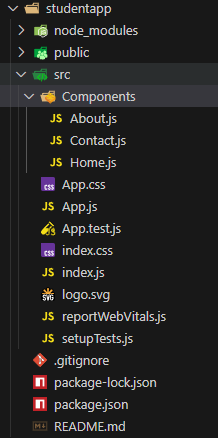
Returns the JSX to be rendered on the screen

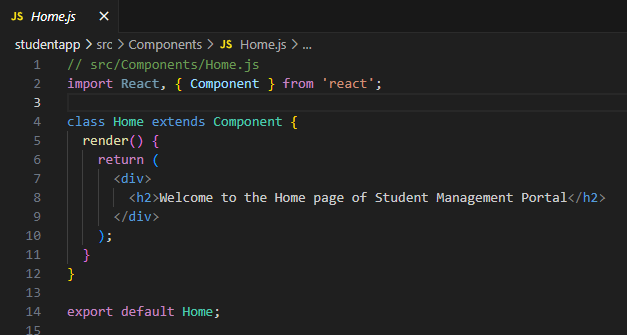
**Hands-On Lab:**

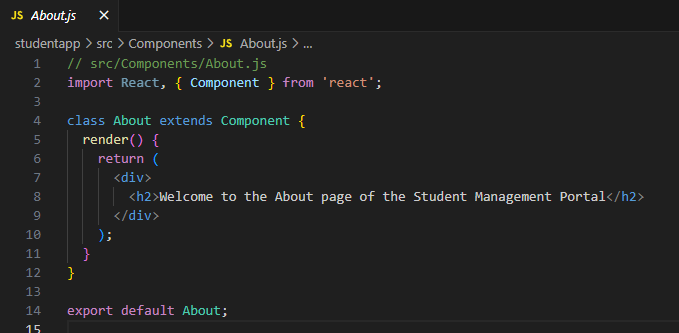
* Create a React project named “StudentApp” type the following command in terminal of Visual studio:

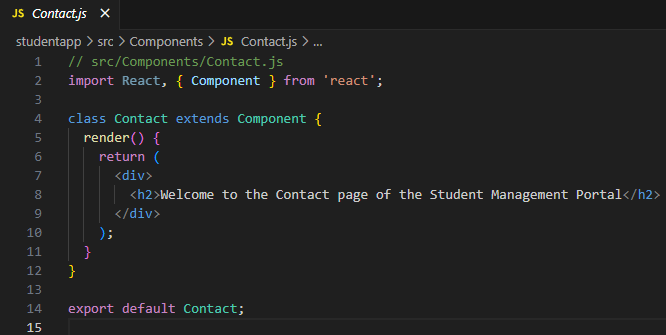


* Create a new folder under Src folder with the name “Components”. Add a new file named “Home.js” “About .js “ and “ Contact.js”

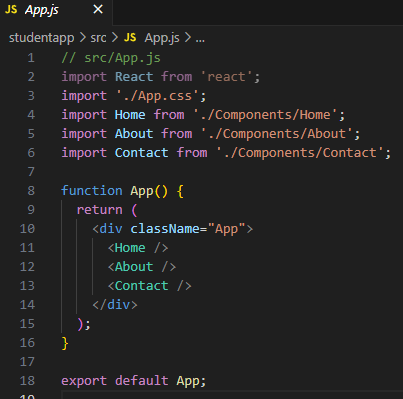




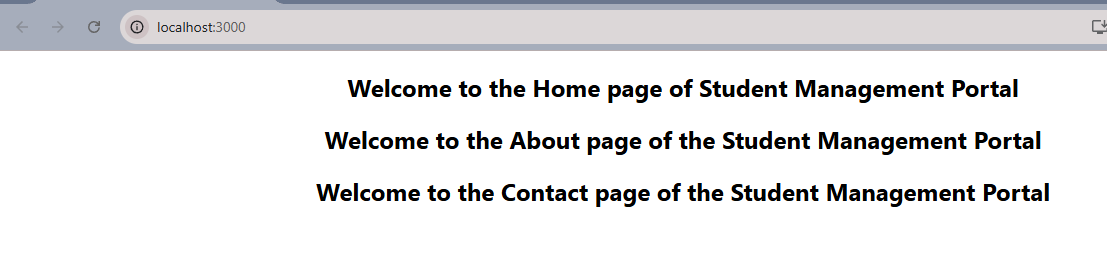




* Edit the App.js to invoke the Home, About and Contact component as follows:



* In command Prompt, navigate into StudentApp and execute the code by typing the following command:
* 
* Open browser and type “localhost:3000” in the address bar:



**Hands On 3: React App for scorecalculatorapp**

**Overview:**

Function components are the recommended way to write components in modern React. This lab introduces the creation of a styled functional component that calculates and displays average scores.

**Learning Objectives:**

* Create a function component
* Apply external CSS styles to components
* Render styled content within a component

**Theory:**

**Function Components:**

Function components are JavaScript functions that return JSX. They can receive inputs via props and do not require a render method or constructor.

**Styling in React:**

You can style React components using:

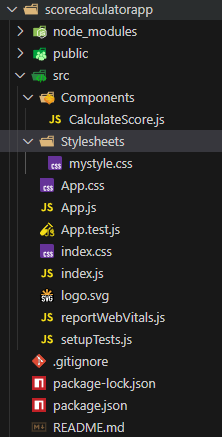
* Inline styles
* CSS modules
* External CSS files (used in this lab)

**Hands-On Lab:**

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



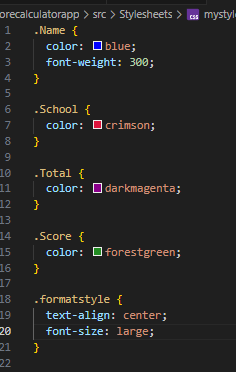
* Create a new folder under Src folder with the name “Components”. Add a new file named “CalculateScore.js”



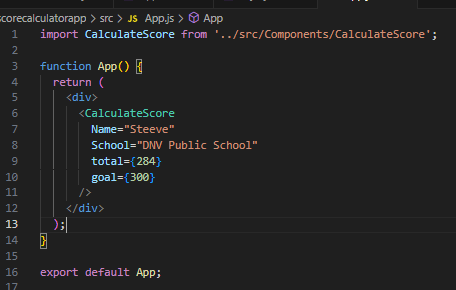
* The following code in CalculateScore.js



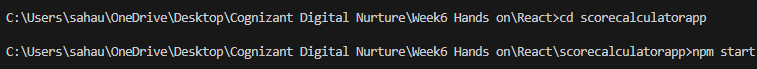
* Create a Folder named Stylesheets and add a file named “mystyle.css” in order to add some styles to the components:



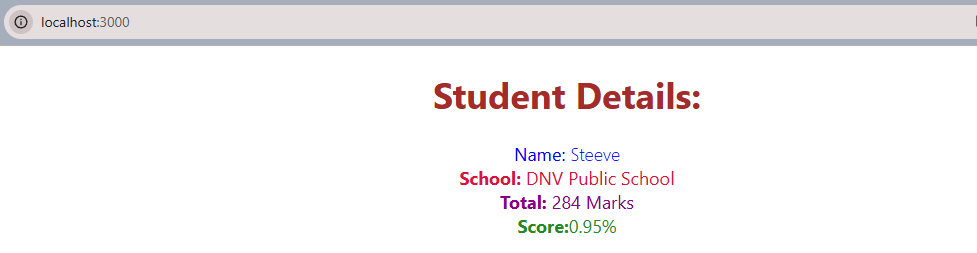
* Edit the App.js to invoke the CalculateScore functional component as follows:



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



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



**Hands On 4:React App tool called blogapp**

**Overview:**

Lifecycle methods are special class methods in React that are called at various points during a component's existence, such as mounting, updating, and unmounting. This lab helps you understand their use by building a simple blog app that loads posts from an API.

**Learning Objectives:**

* Explain the need and benefits of component lifecycle
* Identify various lifecycle hook methods
* Understand the sequence of steps in rendering a component
* Implement **componentDidMount()** for data fetching
* Implement **componentDidCatch()** for error handling

**Theory:**

**Component Lifecycle Benefits:**

* + - Enables actions at key points (fetching data, cleaning up, logging)
    - Helps maintain application performance and stability

**Lifecycle Hook Methods:**

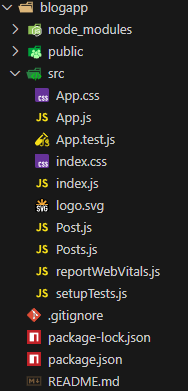
* constructor()
* componentDidMount()
* componentDidUpdate()
* componentWillUnmount()
* componentDidCatch()

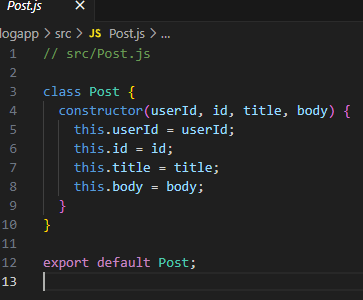
**Lifecycle Flow (Simplified):**

* Constructor
* Render
* componentDidMount
* Updates (triggered by props/state changes)
* componentWillUnmount

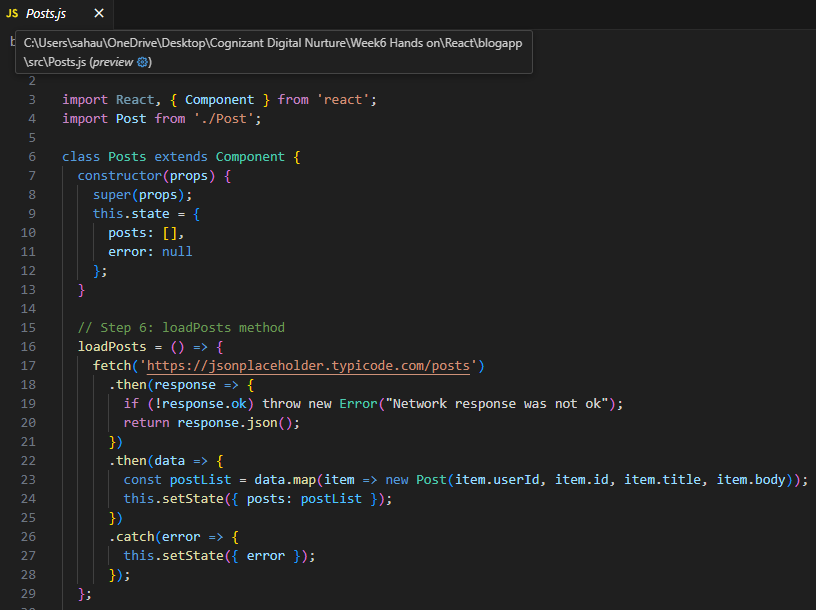
**Hands-On Lab:**

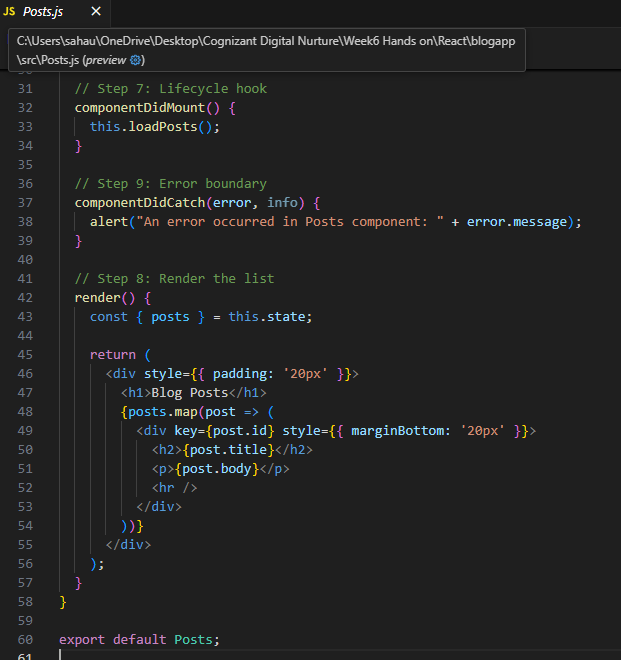
* Create a new react application using *create-react-app* tool with the name as “blogapp”
* Create a new file named as **Post.js** in **src folder** with following properties
* Create a new class based component named as **Posts** inside **Posts.js** file



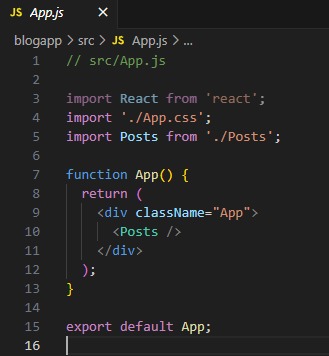


* Initialize the component with a list of Post in state of the component using the constructor
* Create a new method in component with the name as **loadPosts()** which will be responsible for using Fetch API and assign it to the component state created earlier. To get the posts use the url (<https://jsonplaceholder.typicode.com/posts>)
* Implement the **componentDidMount()** hook to make calls to **loadPosts()** which will fetch the posts
* Implement the **render()** which will display the title and post of posts in html page using heading and paragraphs respectively.
* Define a **componentDidCatch()** method which will be responsible for displaying any error happing in the component as alert messages.

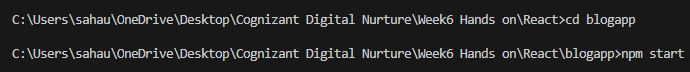


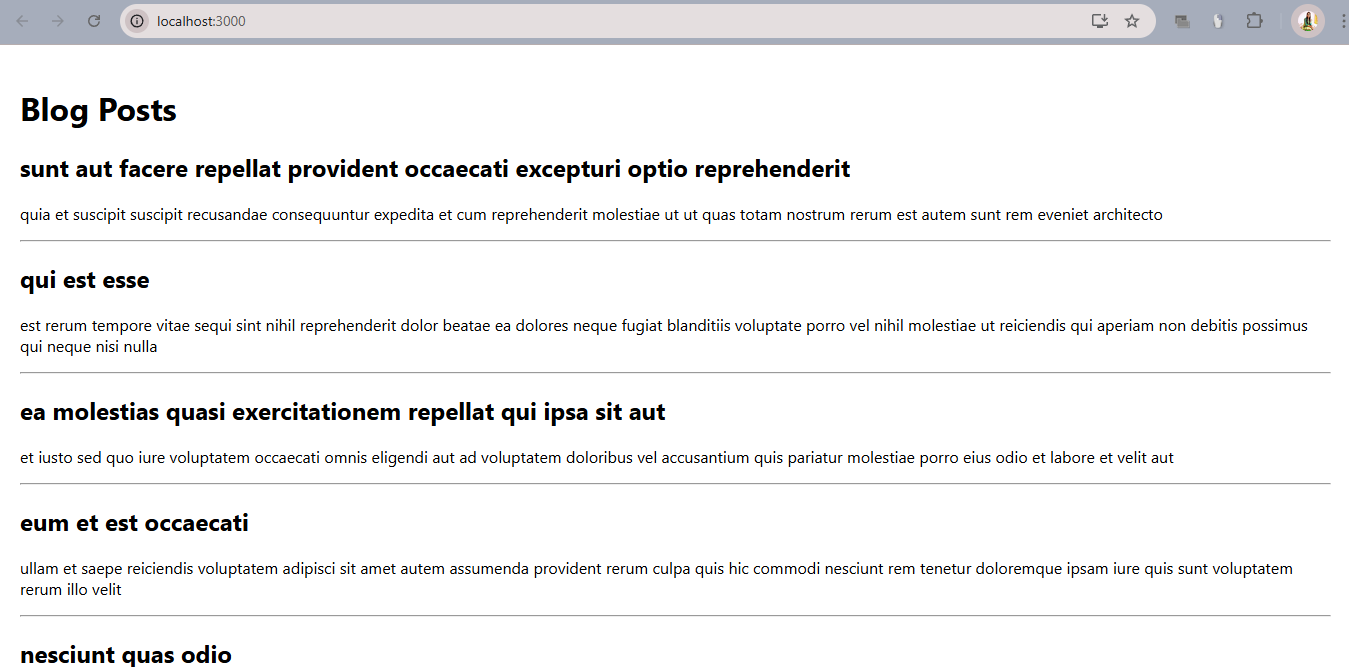


* Add the Posts component to App component.



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





**Hands On 5 :** **Cohort Dashboard Styling**

**Overview:**

Styling React components can be achieved using traditional CSS, inline styles, or scoped CSS Modules. This lab demonstrates each through a real-world use case: a dashboard showing cohort details.

**Learning Objectives:**

* Understand the need for styling in React components
* Work with CSS Modules for scoped styling
* Use inline styles based on component data
* Apply styles using className and style attributes

**Theory:**

**Why Style React Components?**

To improve readability, maintainability, and user experience. Consistent styling makes components modular and reusable.

**CSS Modules vs. Inline Styles:**

* **CSS Modules**: Local scoped styles preventing name conflicts.
* **Inline Styles**: Useful for dynamic styling based on logic/props.

**Hands-On Lab:**

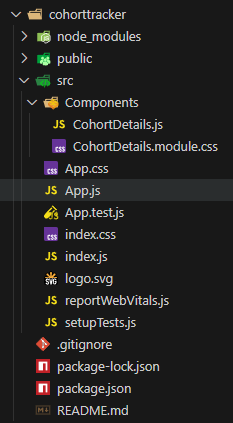
Download and build the attached react application.



* Unzip the react application in a folder
* Open command prompt and switch to the react application folder

Note : This could not be unzipped because the application was not provided so I created a manual one

* Create a new CSS Module in a file called “CohortDetails.module.css”



* Define a css class with the name as “box” with following properties

*Width = 300px;*

*Display = inline block;*

*Overall 10px margin*

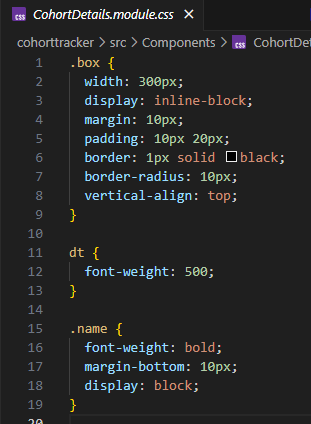
*Top and bottom padding as 10px*

*Left and right padding as 20px*

*1 px border in black color*

*A border radius of 10px*

* Define a css style for html <dt> element using tag selector. Set the font weight to 500.



* Open the cohort details component and import the CSS Module
* Apply the box class to the container div
* Define the style for <h3> element to use “green” color font when cohort status is “ongoing” and “blue” color in all other scenarios.



