**WEEK6 HANDSON**

**Module 8 – Single Page Application framework - React**

**EX1: ReactJS-HOL**

* **SPA**- **Single-Page Application (SPA)** is a web application that interacts with the user by dynamically rewriting the current page rather than loading entire new pages from a server. This means that after the initial page load, only the necessary data is fetched to update the content.

Benefits-Speed and Responsiveness, Enhanced User Experience, Reduced Bandwidth Usage, Caching Capabilities, Simplified Development for Certain Architectures

* **React-** is an open-source JavaScript library, developed and maintained by Facebook, used for building user interfaces (UIs), particularly for single-page applications.It allows developers to create dynamic and interactive web applications efficiently.React is not a full framework but a library focused on the view layer of an application.

Working-React's core principle is to create a **Virtual DOM** (Document Object Model) in memory. Instead of directly manipulating the browser's real DOM, which can be slow, React first makes changes to this lightweight copy. When the state of the application changes, React creates a new virtual DOM. It then compares this new virtual DOM with the previous one in a process called **reconciliation**. By doing this, React identifies only the elements that have changed and efficiently updates only those specific parts of the real DOM.

* **Different between SPA & MPA**

|  |  |
| --- | --- |
| **SPA** | **MPA** |
| Loads a single HTML page and dynamically updates content without full page reloads. | Loads a new, separate page from the server for each user interaction or navigation. |
| Generally faster and more responsive after the initial load because only necessary data is fetched | Can have slower performance due to the need for full page reloads with each interaction.[[5](https://www.google.com/url?sa=E&q=https%3A%2F%2Fvertexaisearch.cloud.google.com%2Fgrounding-api-redirect%2FAUZIYQHCYUiKAx316_3DlI9pE-vAolpi78FTXSX3ZUuXSkcU_2gvKS8XK5q9DObPS0HXG1QKxU7xp3diINDEptCobgOGv7GXodqtVNHvhWhQQ7oN3wYyl25j-wAFnxhYAfAnZFOlzpQ6HqbmwKPdQLgq-D7KUZ3x-IeAe9gRd8Efjw%3D%3D" \t "_blank)] The initial load, however, can be faster. |
| smoother, more fluid, and app-like experience with no interruptions from page reloads. | User experience can be disjointed due to waiting for pages to load. |

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

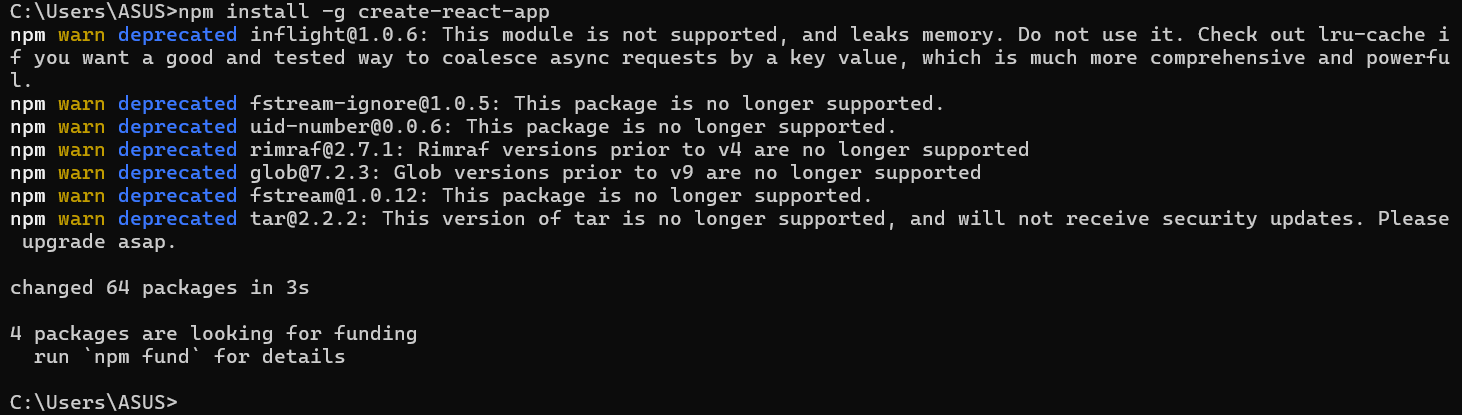
|  |  |
| --- | --- |
| Pros | Cons |
| Speed and Performance | SEO Challenges |
| Rich User Experience | Slower Initial Load Time |
| Lower Bandwidth Consumption | Security Issues |

* **Virtual DOM (VDOM)-** React uses a VDOM to optimize updates and improve performance by minimizing direct manipulation of the real DOM

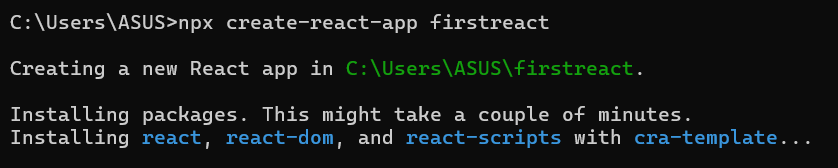
1)Create a new React Application with the name “firstreact”, Run the application to print “welcome to the first session of React” as heading of that page.

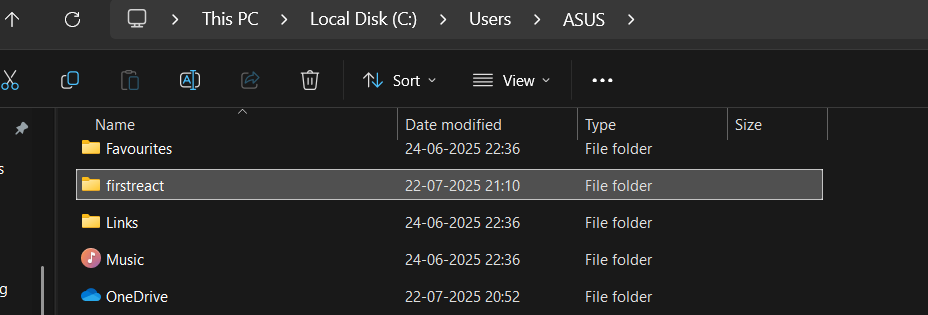
Step1: Install nodejs and npm and complete setup

Step2: Install Create-react-app by running the following command in the command prompt

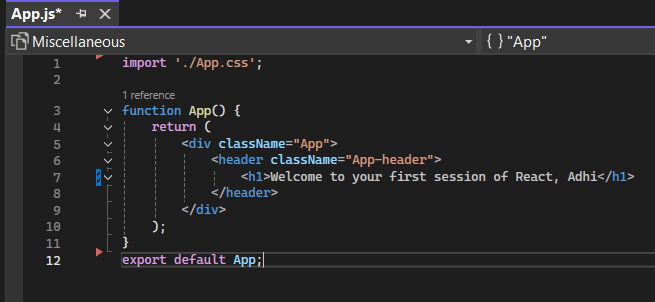


Step3: To create a React Application with the name of “firstreact”, type the following command:

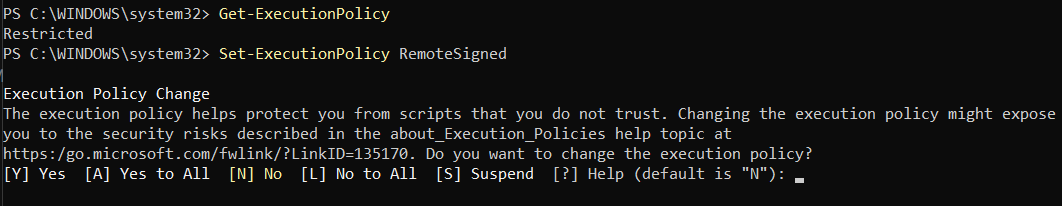


After creating we can see file at respective path

Step4: Open the project folder in visual studion community and update the “App.js” file by below

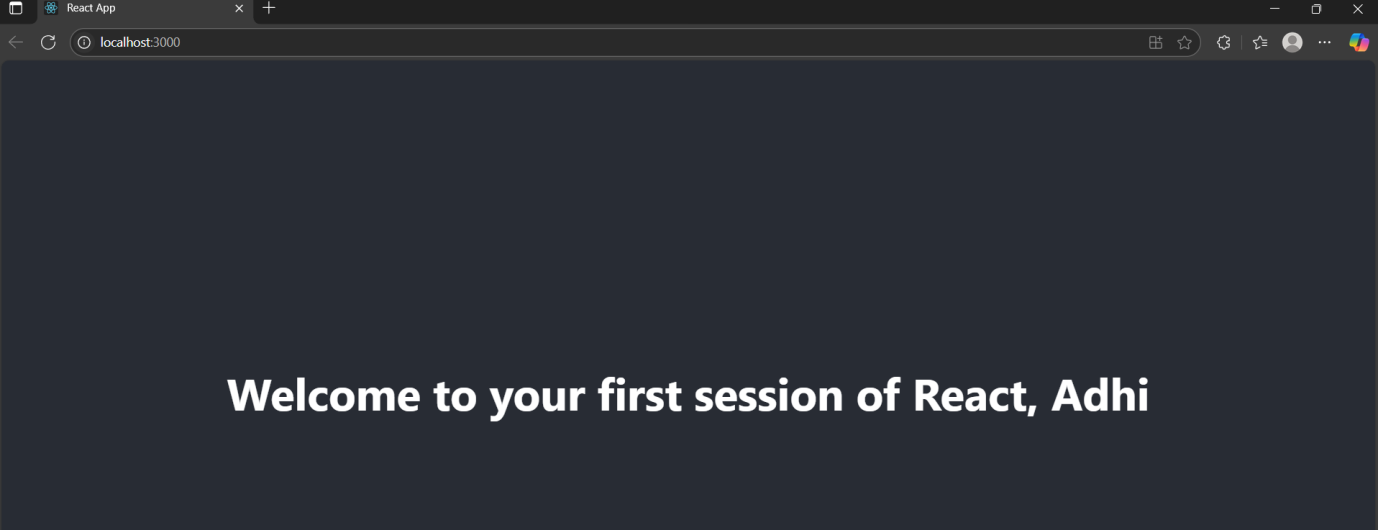


Step5: Before running the application, kindly check weather ExecutionPolicy is “RemoteSigned” else follow the below command in “PowerShell administrator”



Step6: now run the application by “npm start” in terminal of the project

Output:



**EX2: ReactJS-HOL**

**React components-** independent, reusable pieces of code that serve as the building blocks of a user interface (UI). They work like JavaScript functions but return HTML (via JSX) to be rendered on the screen. A complex UI, like a social media feed, can be broken down into smaller components like Profile, Post, Comment, and LikeButton.

|  |  |
| --- | --- |
| **Standard JavaScript Function** | **React Component** |
| To perform a specific task or calculate value. | To describe a part of the UI. |
| Can return any data type | Returns UI elements (typically JSX), null, or false. |
| Has no concept of a lifecycle. | Has lifecycle methods that run at specific times |

**Types of Components**

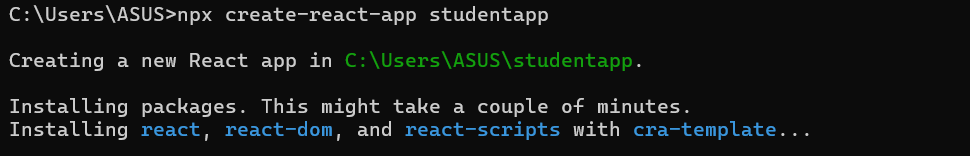
**Class Components-** A class component is an ES6 class that extends React.Component. It must include a render() method that returns JSX. Class components can manage their own state using this.state and have access to lifecycle methods like componentDidMount() and componentDidUpdate(). They are considered the more traditional way of creating stateful components.

**Functional Component-** A functional component is a plain JavaScript function that accepts an object of properties (props) as an argument and returns JSX. Initially, they were simple and stateless. However, with the introduction of **Hooks** (like useState and useEffect), functional components can now manage state and handle side effects, making them the modern and preferred way to write React components.

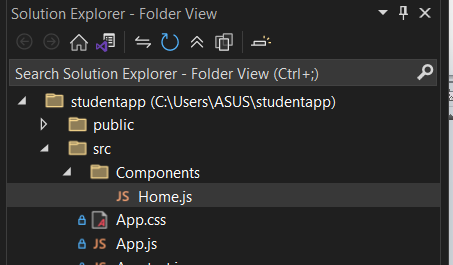
**Render() function**- Required method in class components. Its job is to describe what the UI should look like. React calls this method whenever there's a change in the component's state or props. It should be a pure function, meaning it returns the same output for the same input and does not directly interact with the browser.

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.

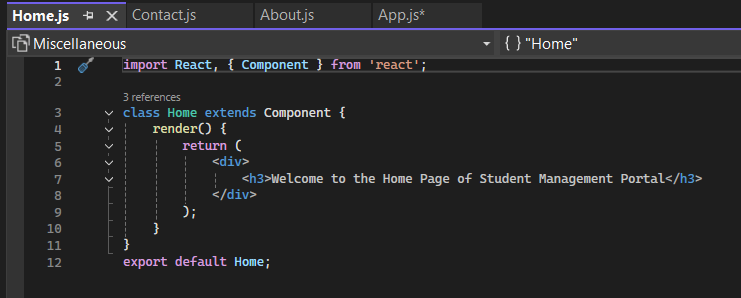
Step1: Create a React project named “StudentApp” type the following command in terminal



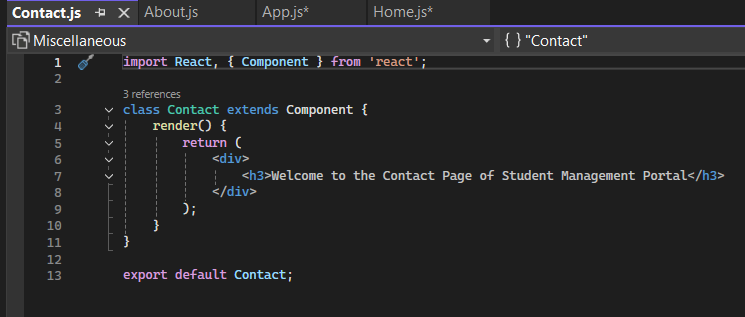
Step2: Create a new folder under Src folder with the name “Components”. Add a new file named “Home.js”

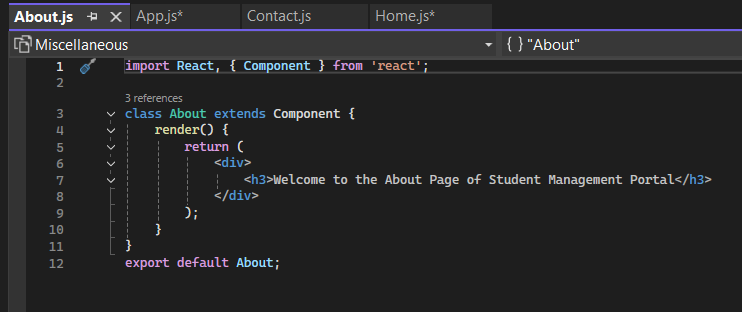


Step3: Write the below code inside “Home.js”

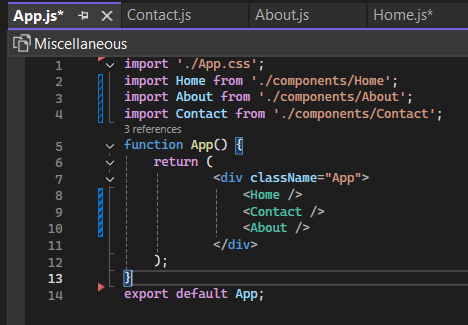
****

Step4: Repeat the same steps for Creating “About.js” and “Contact.js” by adding a new file as ”About.js”, “Contact.js” under “Components” folder and edit the code as mentioned for “Home” Component.

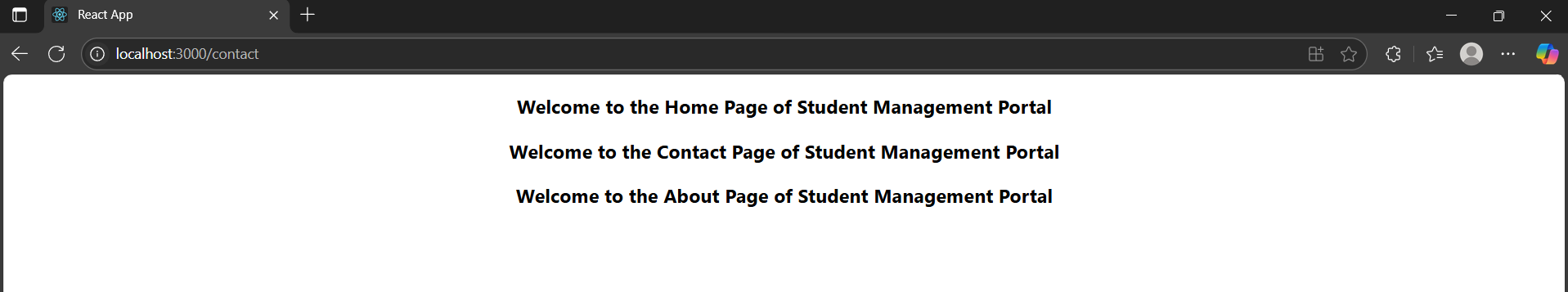




Step5: update “App.js” by below



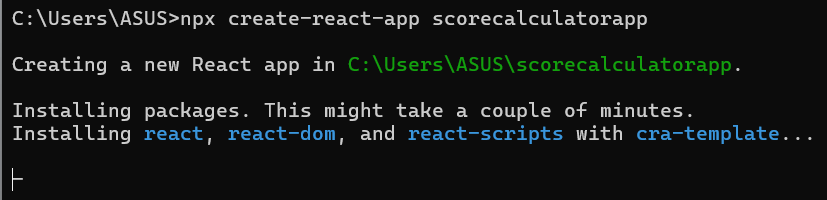
Step6: run “npm start” in terminal



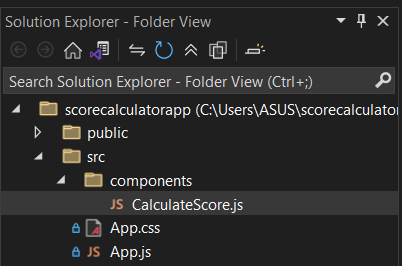
**EX3: ReactJS-HOL**

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

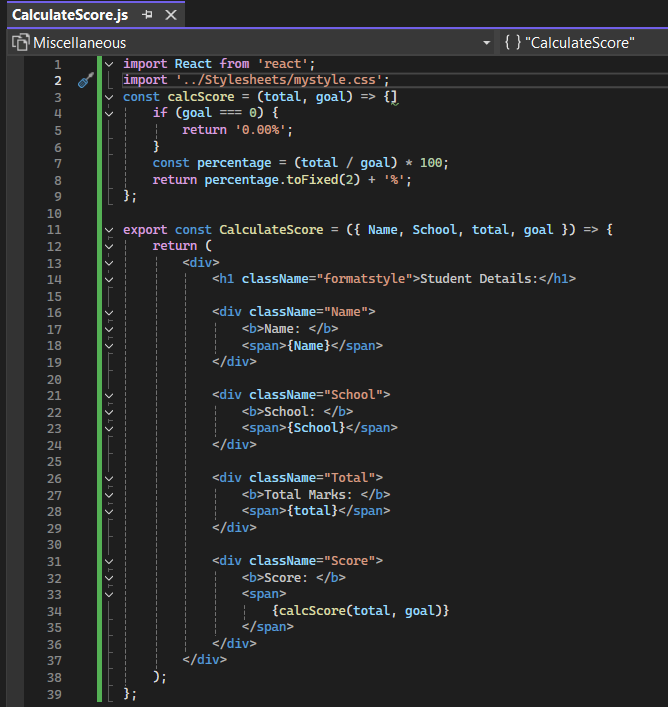
Step1: Create a React project named “scorecalculatorapp” type the following command in terminal

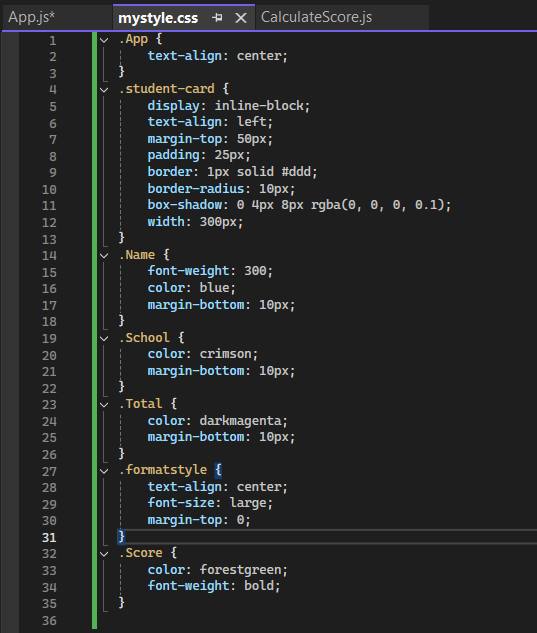


Step2: Create a new folder under Src folder with the name “Components”. Add a new file named “CalculateScore.js”

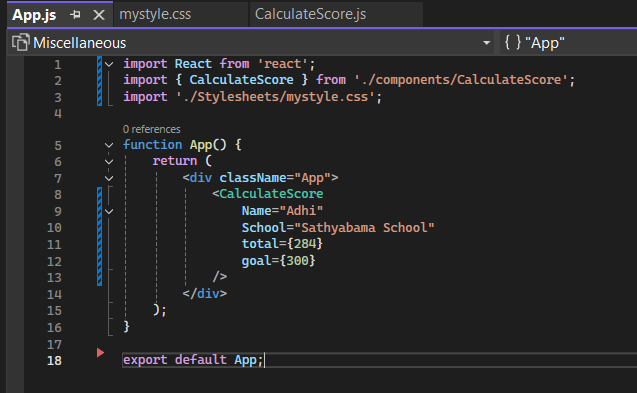


Step3: Type the following code in CalculateScore.js

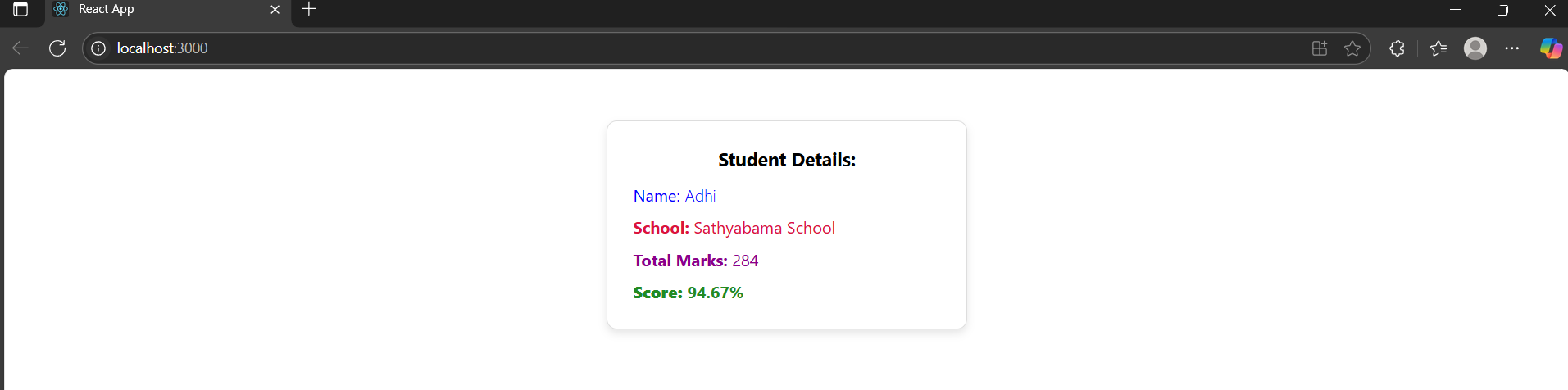


Step4: Create a Folder named Stylesheets and add a file named “mystyle.css” in order to add some styles to the components:

Step5:Update the “App.js” code by below



Step6: run application by “npm start” in terminal



**EX4: ReactJS-HOL**

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

The "component lifecycle" is simply the series of stages it goes through.

* Mounting (Birth): This is when the component first appears on the screen. It's the perfect time to grab any data it needs to display, like fetching user information or loading initial content.
* Updating (Life): As users interact with your app, the component might need to change or re-render. This is its "updating" phase. We can react to these changes, for instance, to update the display when new data arrives.
* Unmounting (The End): When the component is no longer needed and disappears from the screen, it "unmounts." This is our chance to clean up after it—like canceling timers or network requests—to prevent memory leaks and keep the app running smoothly.

**2.Identify various life cycle hook methods**

1. Mounting (When the component is born and first appears)

* constructor(): The very first step. Used for initial setup.
* render(): What the user sees. It creates the HTML for the component.
* componentDidMount(): Runs right after the component is shown. Perfect for fetching data.

2. Updating (When the component's data changes)

* render(): Runs again to show the updated component.
* componentDidUpdate(): Runs right after the component has updated.

3. Unmounting (When the component is about to be removed)

* componentWillUnmount(): The last call. Use this to clean up anything you started, like timers.

4. Error Handling (When something breaks)

* componentDidCatch(): A safety net. It catches errors in child components so the whole app doesn't crash.

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

1. When it first appears (Mounting):

Setup: constructor() runs first to get things ready.

Show: render() creates the HTML to display on the screen.

Fetch Data: componentDidMount() runs after it's on the screen, perfect for loading data.

2. When it changes (Updating):

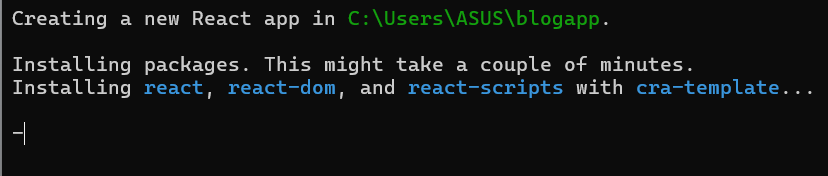
Re-show: render() runs again to show the new changes.

Aftermath: componentDidUpdate() runs after the update is complete.

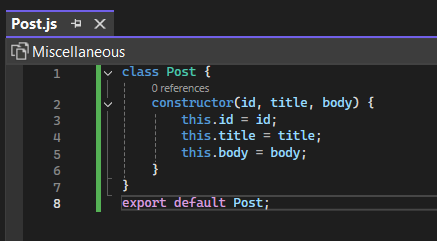
3. When it disappears (Unmounting):

Cleanup: componentWillUnmount() runs right before it's removed, giving you a chance to clean up.

Step1: Create react app “blogapp” by running below comment in cmd

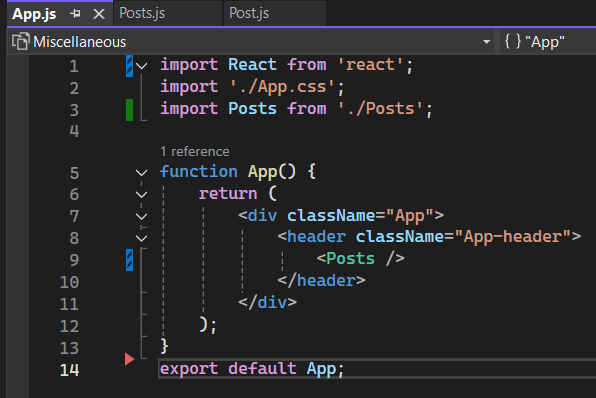


Step2: Create a new file named as “**Post.js”** in **src folder** with below code



Step3: Create a new file “Posts.js” in src folder with below code

Step4: Update “App.js” by below

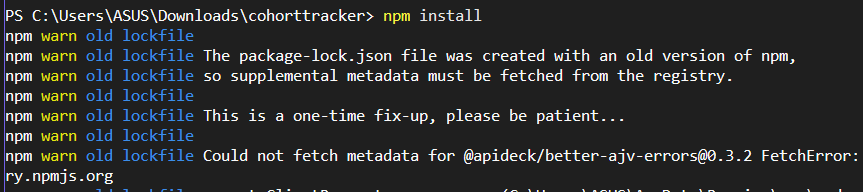


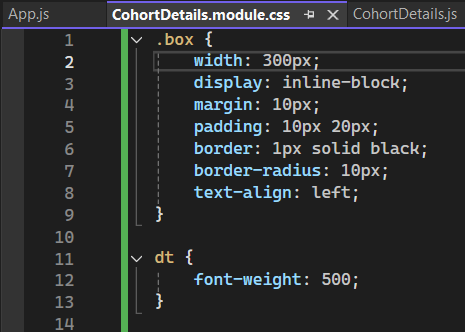
Step5: run application by “npm start” in terminal

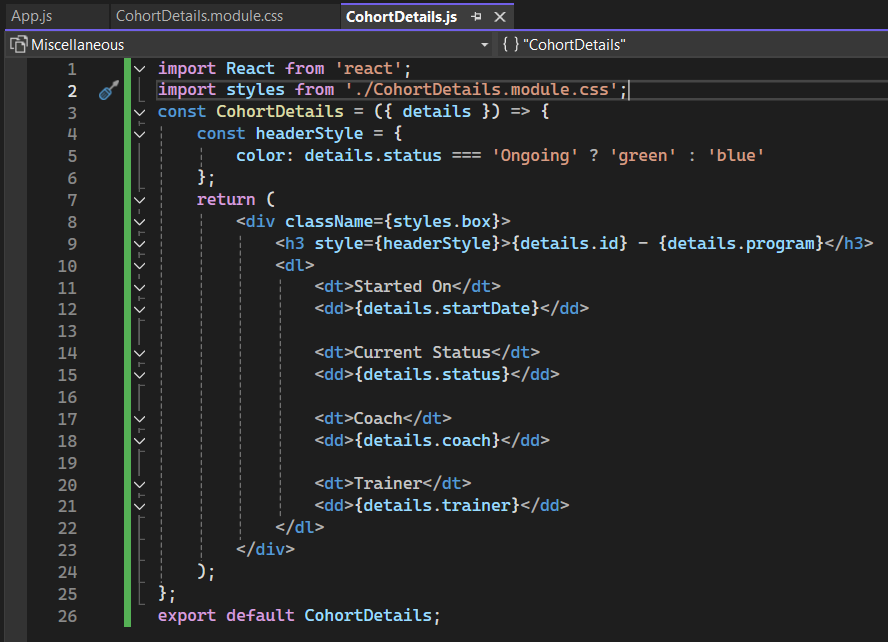


**EX5: ReactJS-HOL**

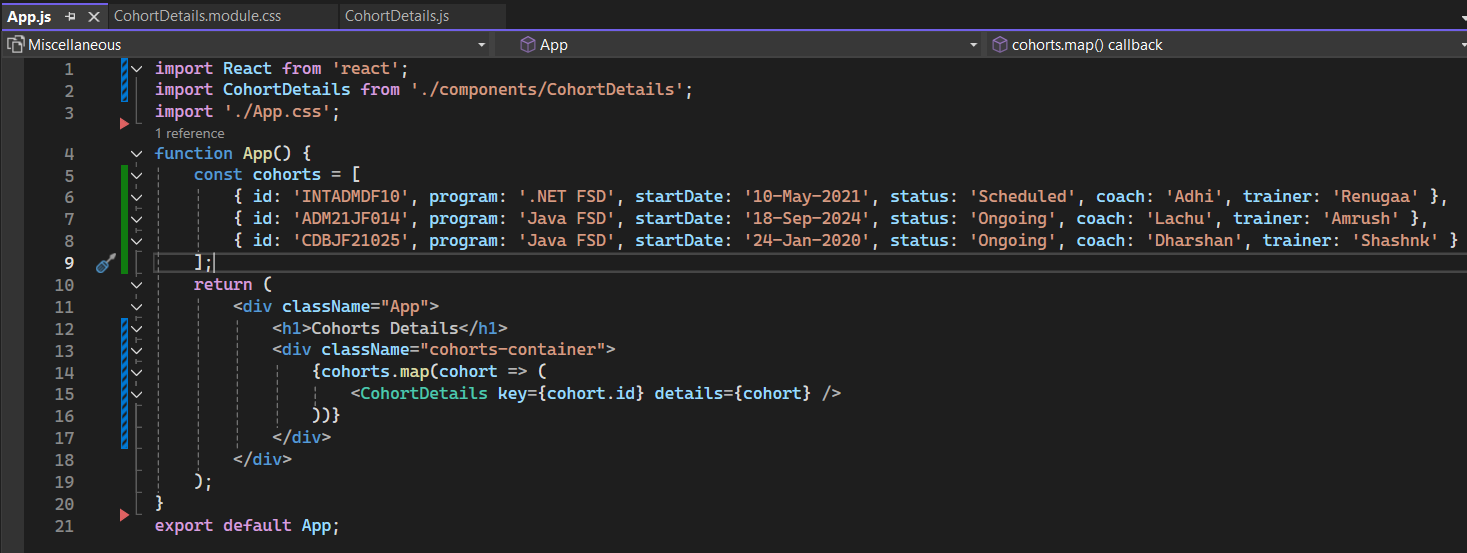
Step1: Download “[cohortstracker](https://github.com/j4ik2i5x0/Digital-Nurture-4.0-DotNetFSE-Jaikrishnan-J-42614046/tree/main/Week%206/6_cohortstracker).zip” and extract then run below in terminl



Step2: create “components” folder inside src and add “CohortDetails.module.css” & ”CohortDetails.js” with below code



Step3: update “App.js” by below code



Step4: run ”npm start” in terminal  
here we can see that green-ongoing, blue-scheduled

