**REACT JS**

**HANDS-ON 1**

**1. Define SPA and Its Benefits**

A **Single-Page Application (SPA)** is a web app or site that dynamically updates the page content without requiring a full page reload. It loads a single HTML page and updates content through JavaScript as the user interacts.

**Benefits of SPA:**

* Faster interactions and better performance after initial load
* Smoother user experience that feels like a native app
* Reduced server load since it avoids full page refreshes
* Enables client-side routing for seamless navigation

**2. Define React and Identify Its Working**

**React** is a JavaScript library developed by Facebook for building user interfaces, especially for SPAs.

**How React works:**

* Uses components to organize and reuse UI sections
* Manages state and props to control how components behave
* Utilizes a virtual DOM to efficiently track and apply changes without directly modifying the real DOM

**3. Differences Between SPA and MPA**

| **Feature** | **SPA (Single-Page Application)** | **MPA (Multi-Page Application)** |
| --- | --- | --- |
| Page Reload | No | Yes |
| Performance | Faster after initial load | Slower due to full reloads |
| SEO Optimization | Requires special configuration | Naturally SEO-friendly |
| Server Communication | Minimal | Frequent |
| Complexity | More complex (client-side logic) | Simpler server-side logic |
|  |

**4. Pros and Cons of Single-Page Application**

**Pros:**

* Fast performance and responsiveness
* Smooth and consistent user experience
* Efficient data handling using APIs

**Cons:**

* Challenging for search engines without extra setup
* Requires robust client-side architecture
* Can expose security vulnerabilities through APIs

**5. Explain About React**

React is focused on building the view layer of web applications. It allows developers to create modular, reusable components that manage their own state and update efficiently. This results in faster development and easier maintenance.

**6. Define Virtual DOM**

The **Virtual DOM** is a lightweight, in-memory representation of the real DOM. React uses it to detect changes in the UI efficiently. When a component's state or props change, React updates the virtual DOM first, calculates the minimal changes needed, and then updates the real DOM accordingly.

**7. Features of React**

* Component-based architecture for modular development
* Reusable components across different parts of the app
* Efficient updates using the virtual DOM
* Hooks for managing state and lifecycle in functional components
* React Router for handling navigation in SPAs
* Developer tools for inspecting and debugging React applications

**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.**

**Prerequisites:**

Make sure the following are installed:

1. **Node.js & npm**  
   Download from: https://nodejs.org/en/download/
2. **Visual Studio Code**

**Steps:**

**1. Install create-react-app globally:**

npm install -g create-react-app

**2. Create your React app:**

npx create-react-app myfirstreact

**3. Navigate into your app folder:**

cd myfirstreact

**4. Open the project in Visual Studio Code:**

code .

**5. Edit App.js file:**

* Go to src/App.js
* Delete all code inside it
* Replace it with:

function App() {

return (

<div>

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

</div>

);

}

export default App;

**6. Run the application:**

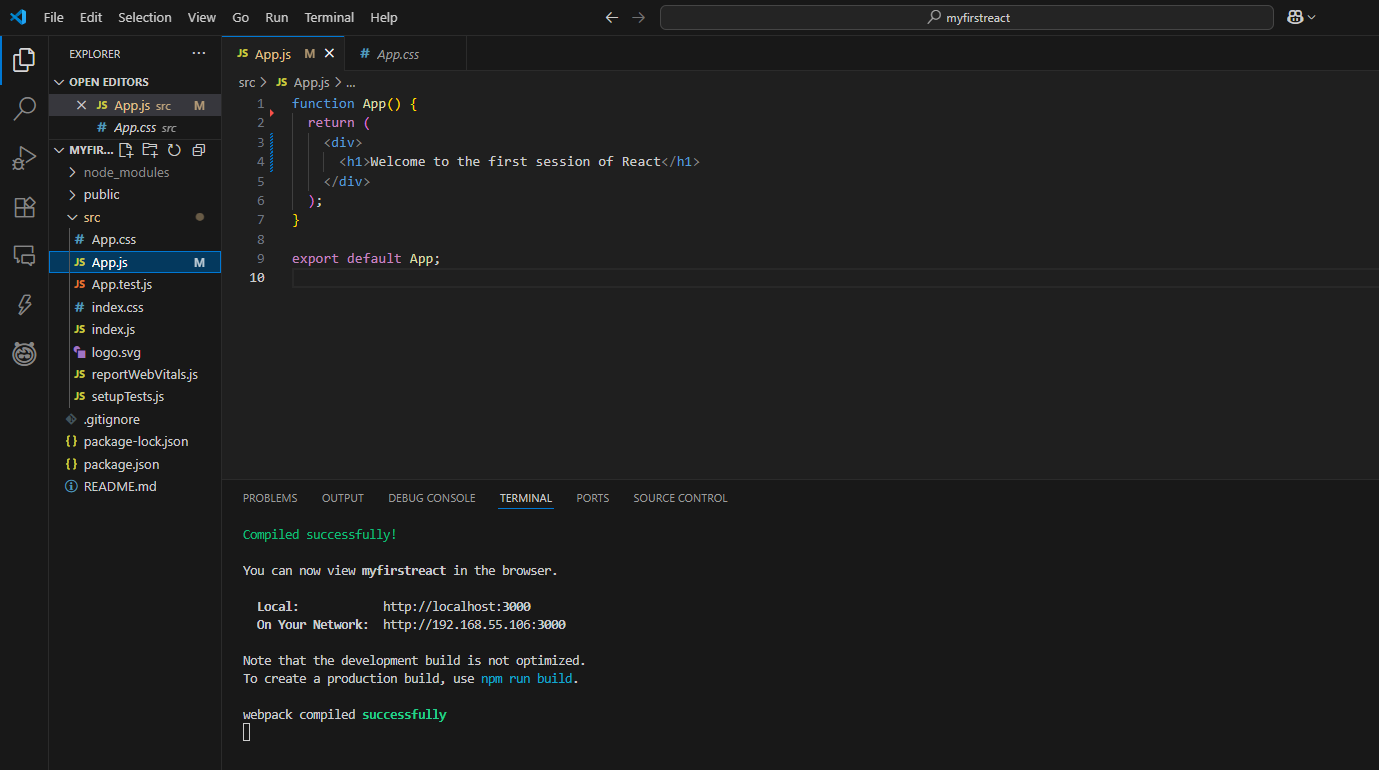
npm start

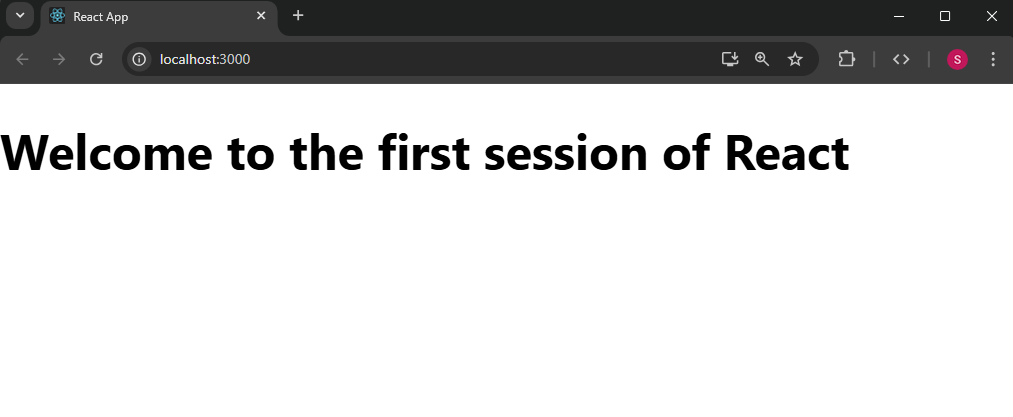
**7. Open your browser and go to:**

http://localhost:3000

You should now see:

**Welcome to the first session of React**





**HANDS-ON 2**

**1. Explain React components**

React components are the building blocks of a React application’s UI. Each component is a JavaScript function or class that returns a React element (JSX), describing how a portion of the UI should appear. Components make the UI modular and reusable.

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

| **Criteria** | **React Components** | **JavaScript Functions** |
| --- | --- | --- |
| Purpose | Render UI | Perform logic or computations |
| Return | JSX / React Elements | Any data type |
| Lifecycle | Has lifecycle (class-based) | No lifecycle |
| Integration | Can manage state and props | Stateless by default |

**3. Identify the types of components**

There are two main types of components in React:

* **Class Components**: Traditional components with lifecycle methods.
* **Function Components**: Simpler and modern, can use Hooks for state and side effects.

**4. Explain class component**

A **class component** is defined using ES6 class syntax and extends React.Component. It includes:

* A constructor() to initialize state.
* A render() method to return JSX.
* Lifecycle methods like componentDidMount() or componentDidUpdate().

**Example:**

class Home extends React.Component {

render() {

return <h1>Welcome to the Home page</h1>;

}

}

**5. Explain function component**

A **function component** is a simpler way to define a React component using a plain JavaScript function that returns JSX. From React 16.8 onwards, function components can manage state using Hooks (e.g., useState, useEffect).

**Example:**

function About() {

return <h1>Welcome to the About page</h1>;

}

**6. Define component constructor**

The constructor(props) method is used in class components to:

* Initialize state (this.state = {})
* Bind event handlers (this.method = this.method.bind(this))

It is called once when the component is instantiated.

**7. Define render() function**

The render() function is mandatory in class components. It returns the JSX that React uses to build the DOM tree. Whenever the state or props of a component change, render() is called again to reflect the changes in the UI.

**1. Create the React project**

Open the terminal and run:

npx create-react-app studentapp

**2. Navigate to the project folder**

cd StudentApp

**3. Open the project in VS Code**

code .

**Project Structure**

**. Create a folder named Components**

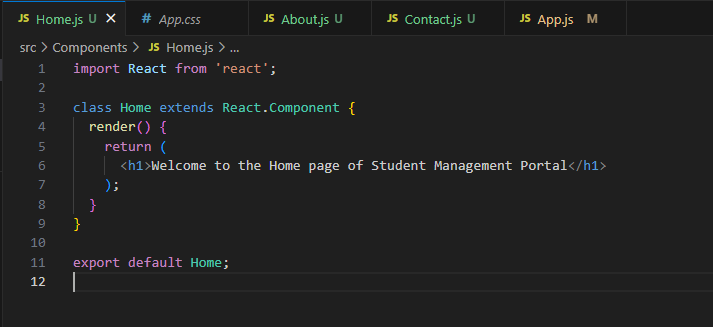
In VS Code:

* Right-click src → New Folder → Name it Components

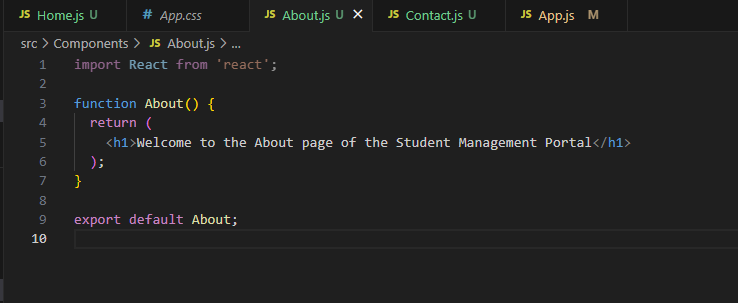
**5. Create the following component files inside src/Components/:**

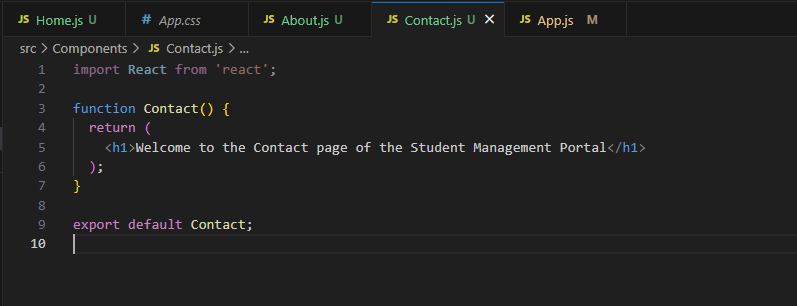
* Home.js
* About.js
* Contact.js

**Home.js**

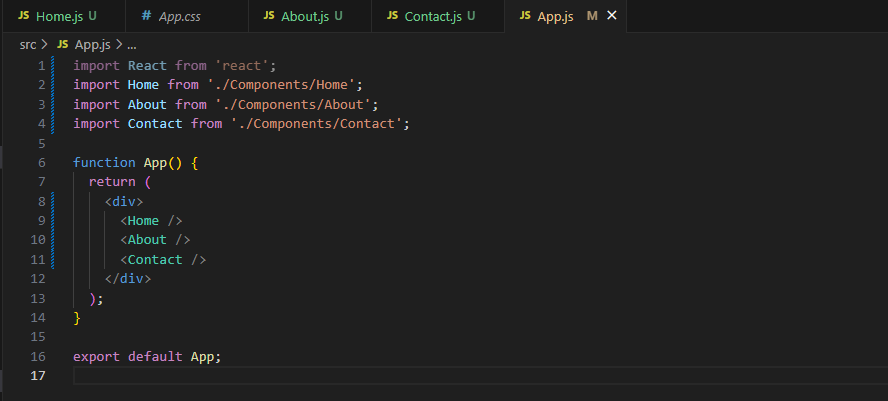


**About.js**



**Contact.js**

**Edit App.js to use the components**



**Run the App**

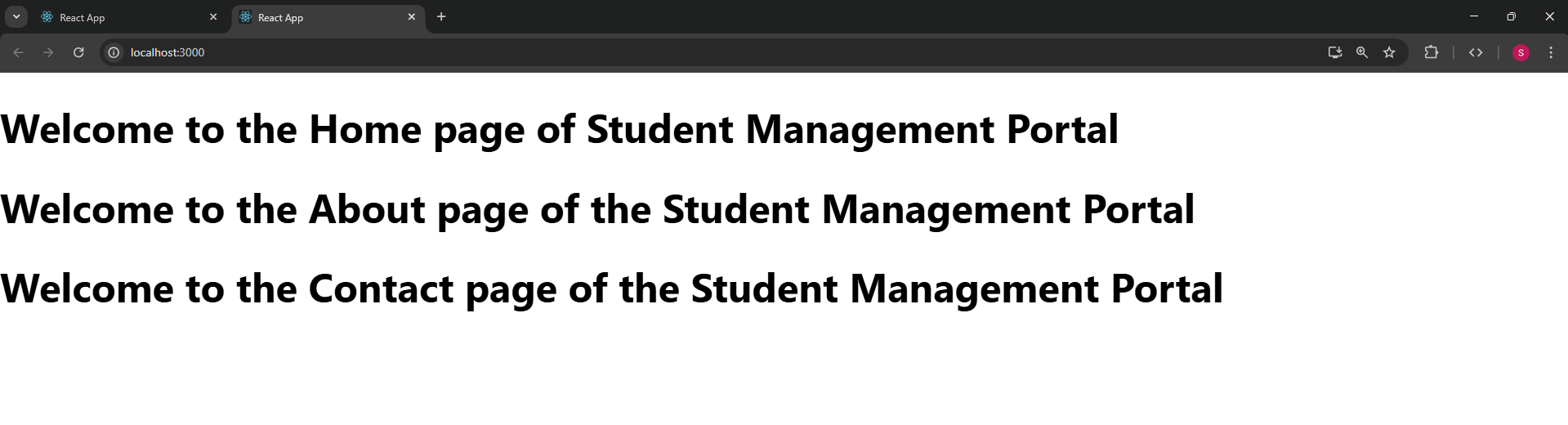
In terminal:

npm start

http://localhost:3000

You should see all three messages:

* “Welcome to the Home page of Student Management Portal”
* “Welcome to the About page of the Student Management Portal”
* “Welcome to the Contact page of the Student Management Portal”



**HANDS-ON 3**

**1. Explain React components**

React components are reusable, self-contained building blocks that define how a part of the UI should appear and behave. Components can be either class-based or function-based, and they help in structuring the application into modular units.

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

|  |  |  |
| --- | --- | --- |
| **Feature** | **React Components** | **JavaScript Functions** |
| Purpose | Renders UI and manages UI logic | Performs generic tasks or calculations |
| Return Type | JSX (React elements) | Any JavaScript value |
| Lifecycle | May have lifecycle hooks (especially class components) | No concept of lifecycle |
| State | Can hold and update component state | Stateless |
| Integration | Part of React's rendering system | Not directly used for UI |

**3. Identify the types of components**

React has two main component types:

* Class Components: Use ES6 classes, support lifecycle methods, and use this.state.
* Function Components: Declared as functions; support state and side effects using React Hooks like useState and useEffect.

**4. Explain class component**

A class component is a React component defined using the class keyword. It extends React.Component and must include a render() method that returns JSX. Class components are more complex but support lifecycle methods and state management.

**5. Explain function component**

A function component is a simpler form of React component. It is just a JavaScript function that returns JSX. With the introduction of React Hooks, function components can now manage state and perform side effects.

**6. Define component constructor**

In class components, the constructor is a special method used for:

* Initializing the component's state (this.state)
* Binding event handlers

**7. Define render() function**

The render() method is required in every class component. It returns the JSX that tells React what to display on the screen. Whenever the component's state or props change, React calls render() again to update the UI.

**STEPS:**

**STEP 1: Create the React App**

Open your terminal and type:

npx create-react-app scorecalculatorapp

cd scorecalculatorapp

**STEP 2: Open the Project in VS Code**

Type in terminal :

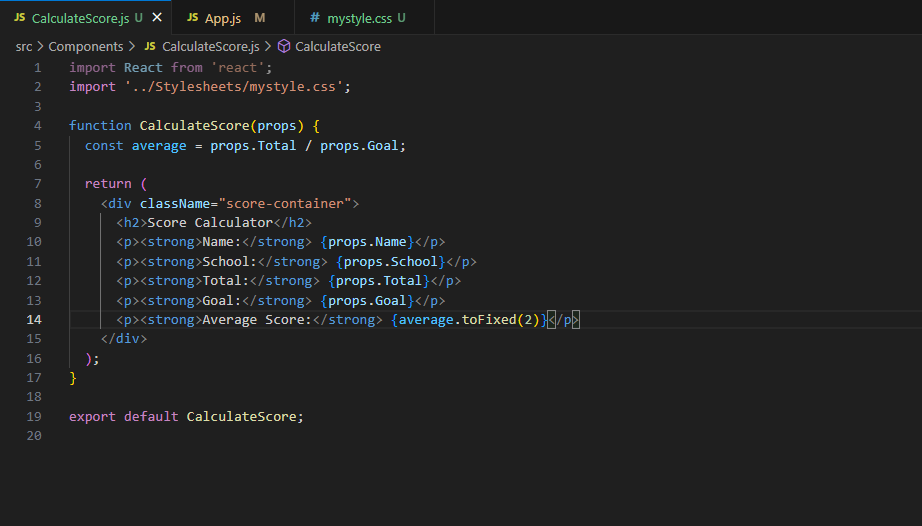
code .

**STEP 3: Create a Folder for Components**

* Go to src/
* Right-click → **New Folder** → Name it: Components
* Inside the Components folder, add a file named: CalculateScore.js

**STEP 4: Add Component Code**

CalculateScore.js:

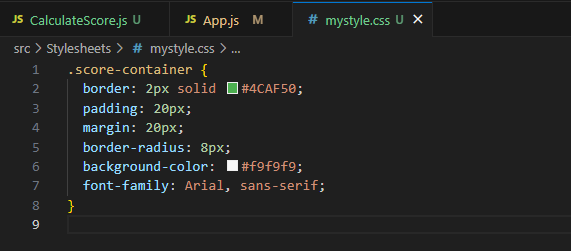


**STEP 5: Create Stylesheet**

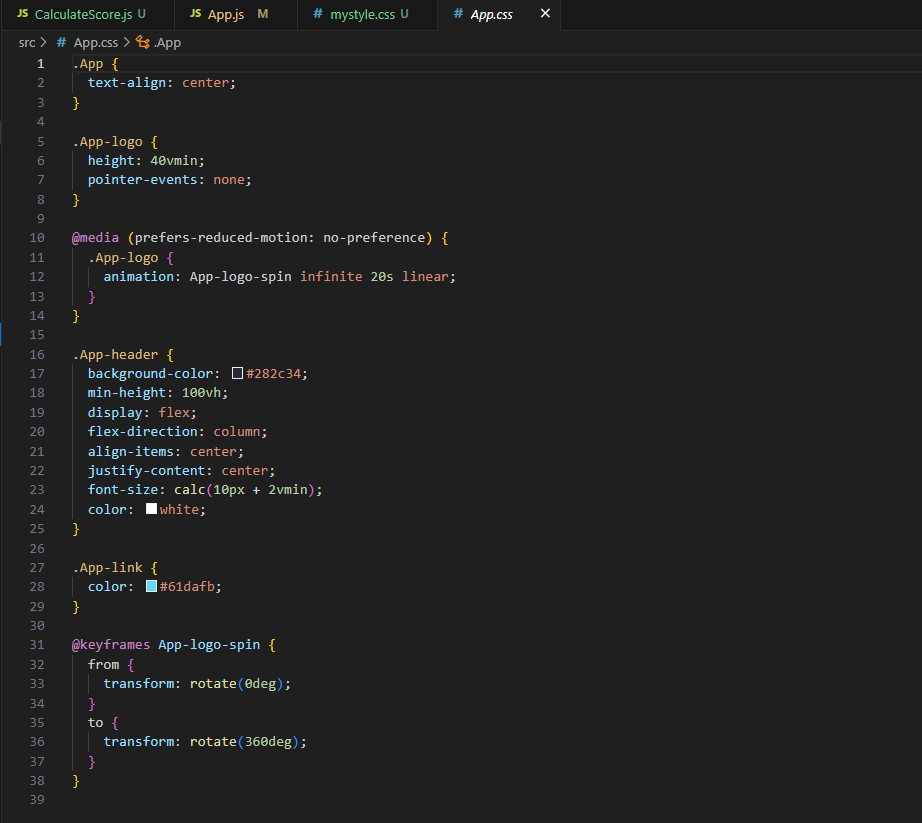
In src/:

* Create a new folder named: Stylesheets
* Inside it, add a file: mystyle.css

Mystyle.css



**STEP 6: Modify App.js**



**STEP 7: Run the App**

In your terminal type:

npm start

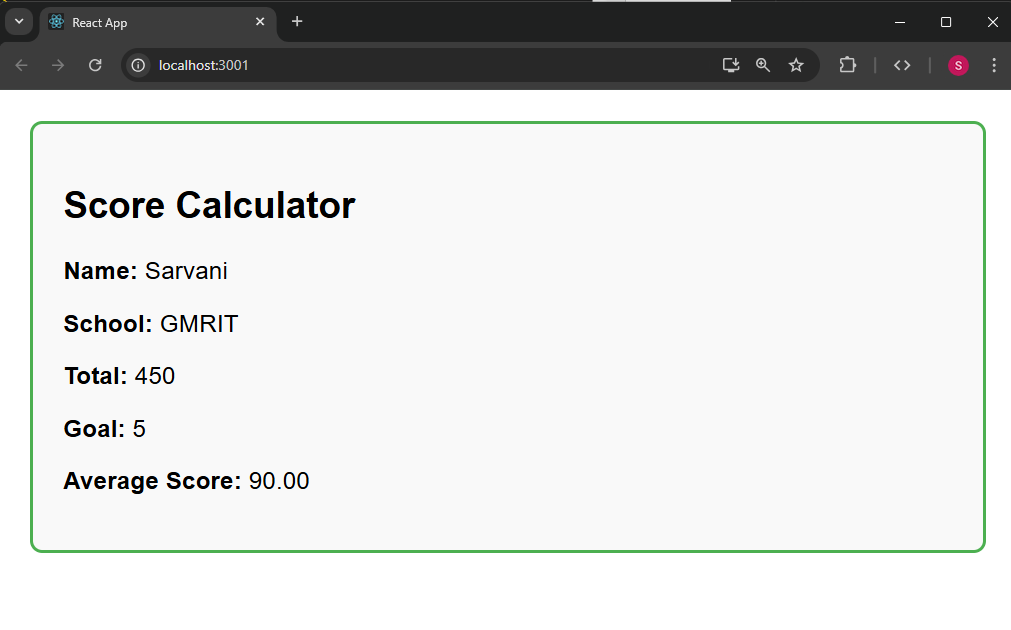
This will:

* Launch your browser automatically
* Open the app at http://localhost:3001

You should see:

**Score Calculator**  
Name: Sarvani  
School: GMRIT  
Total: 450  
Goal: 5  
Average Score: 90.00

,



**HANDS-ON 4**

**1. Explain the need and benefits of component lifecycle**

In React, the component lifecycle refers to the series of phases that a component goes through during its existence—from creation and rendering to updating and unmounting.

Why it's needed:

* To perform side effects (e.g., fetch data, add event listeners)
* To manage external resources
* To control what happens before or after rendering

Benefits:

* Greater control over component behavior
* Efficient resource management
* Improved app performance
* Easier debugging and maintenance

**2. Identify various lifecycle hook methods**

React class components have lifecycle methods grouped into 3 phases:

a. Mounting (when the component is added to the DOM)

* constructor()
* static getDerivedStateFromProps()
* render()
* componentDidMount()

b. Updating (when props or state change)

* static getDerivedStateFromProps()
* shouldComponentUpdate()
* render()
* getSnapshotBeforeUpdate()
* componentDidUpdate()

c. Unmounting (when the component is removed from the DOM)

* componentWillUnmount()

d. Error Handling

* componentDidCatch()
* getDerivedStateFromError()

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

**When a component is rendered:**

1. constructor() — initializes state
2. render() — returns JSX
3. componentDidMount() — called after component is mounted
4. Updates trigger render() again
5. If an error occurs, componentDidCatch() handles it

**Step 1: Create the React App**

Open terminal and run:

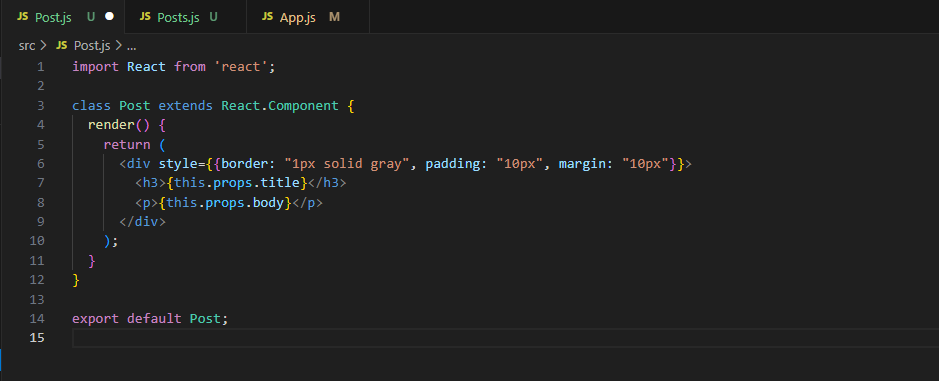
npx create-react-app blogapp

cd blogapp

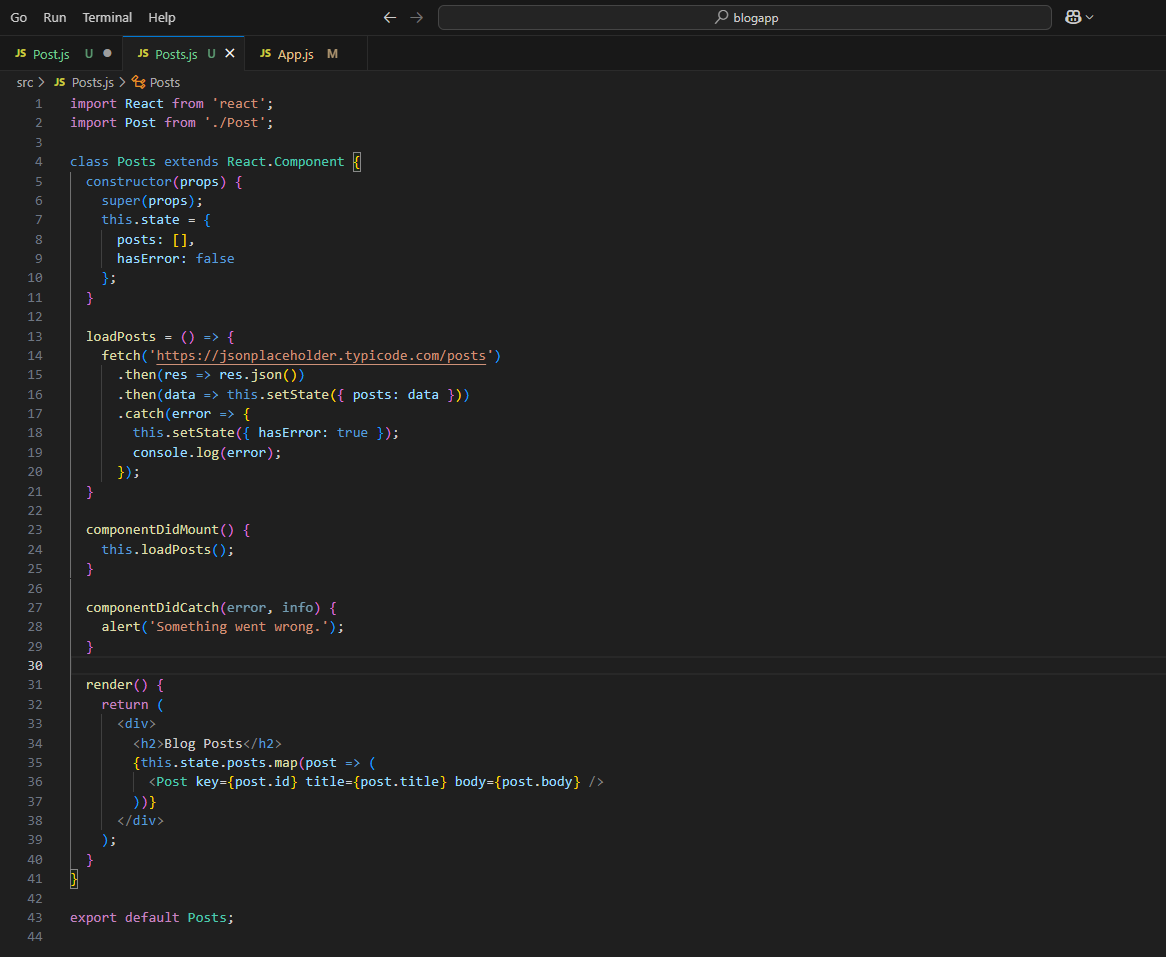
Open it in Visual Studio Code:

code .

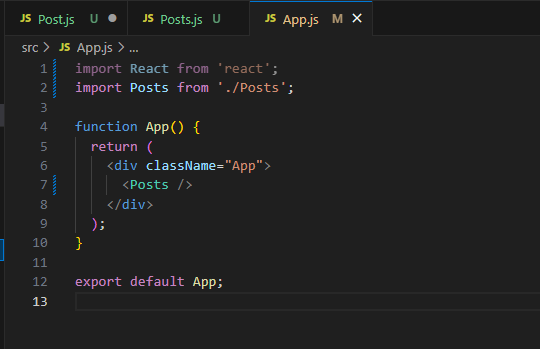
**Step 2: Create Post.js inside src folder**

****

**Step 3: Create Posts.js inside src folder**



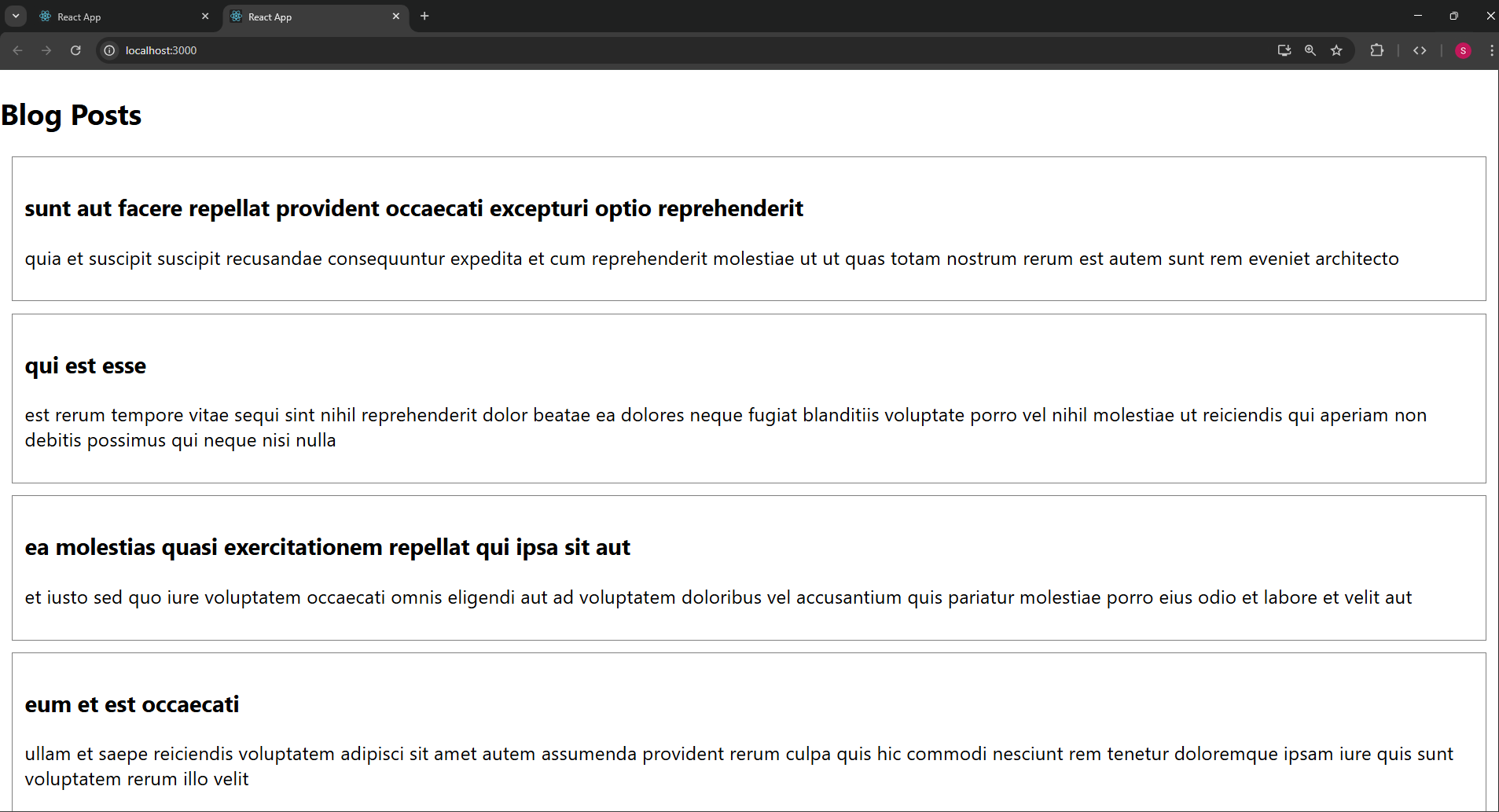
**Step 4: Edit App.js**



**Step 5: Run the App**

in terminal, run:

npm start

**HANDS-ON 5**

**CohortDetails.js**

import styles from './CohortDetails.module.css';

function CohortDetails(props) {

const isOngoing = props.cohort.currentStatus === 'Ongoing'; // note: case-sensitive match

return (

<div className={styles.box}>

<h3 style={{ color: isOngoing ? 'green' : 'blue' }}>

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

</h3>

<dl>

<dt>Started On</dt>

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

<dt>Current Status</dt>

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

<dt>Coach</dt>

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

<dt>Trainer</dt>

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

</dl>

</div>

);

}

export default CohortDetails;

**CohortDetails.module.css**

.box {

width: 300px;

display: inline-block;

margin: 10px;

padding: 10px 20px;

border: 1px solid black;

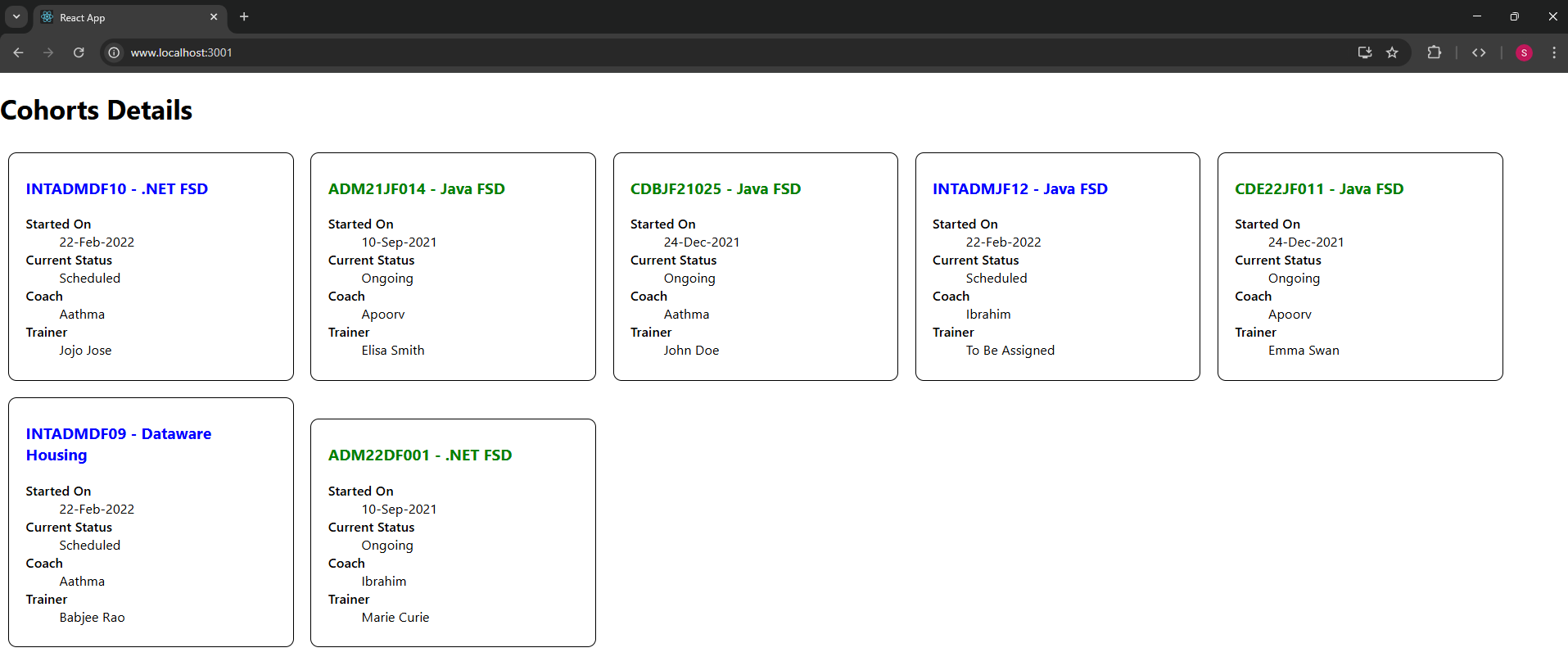
border-radius: 10px;

}

dt {

font-weight: 500;

}

****