**Week-6 Hands-On(React)**

1.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.

**Objective:**

**Define SPA and its benefits**  
A **Single Page Application (SPA)** is a web app that loads a single HTML page and dynamically updates content as the user interacts with it.  
**Benefits:** Faster navigation, reduced server load, smoother user experience, no full page reload.

**Define React and identify its working**  
React is a JavaScript library for building user interfaces using reusable components.  
It works by maintaining a **virtual DOM**, efficiently updating only the parts of the real DOM that change.

**Identify the differences between SPA and MPA**

* SPA: Single HTML page, client‑side routing, dynamic content updates.
* MPA: Multiple HTML pages, each request loads a new page from server.

**Explain Pros & Cons of SPA**  
**Pros:** Fast transitions, better UX, less bandwidth.  
**Cons:** SEO challenges, initial load can be heavier, JavaScript dependency.

**Define virtual DOM**  
A lightweight copy of the real DOM in memory; React uses it to detect changes and update only what’s needed in the real DOM.

**Explain Features of React**  
Component‑based architecture, virtual DOM, unidirectional data flow, JSX syntax, reusable UI components.

**Code:**

import React from "react";

*function* App() {

  return (

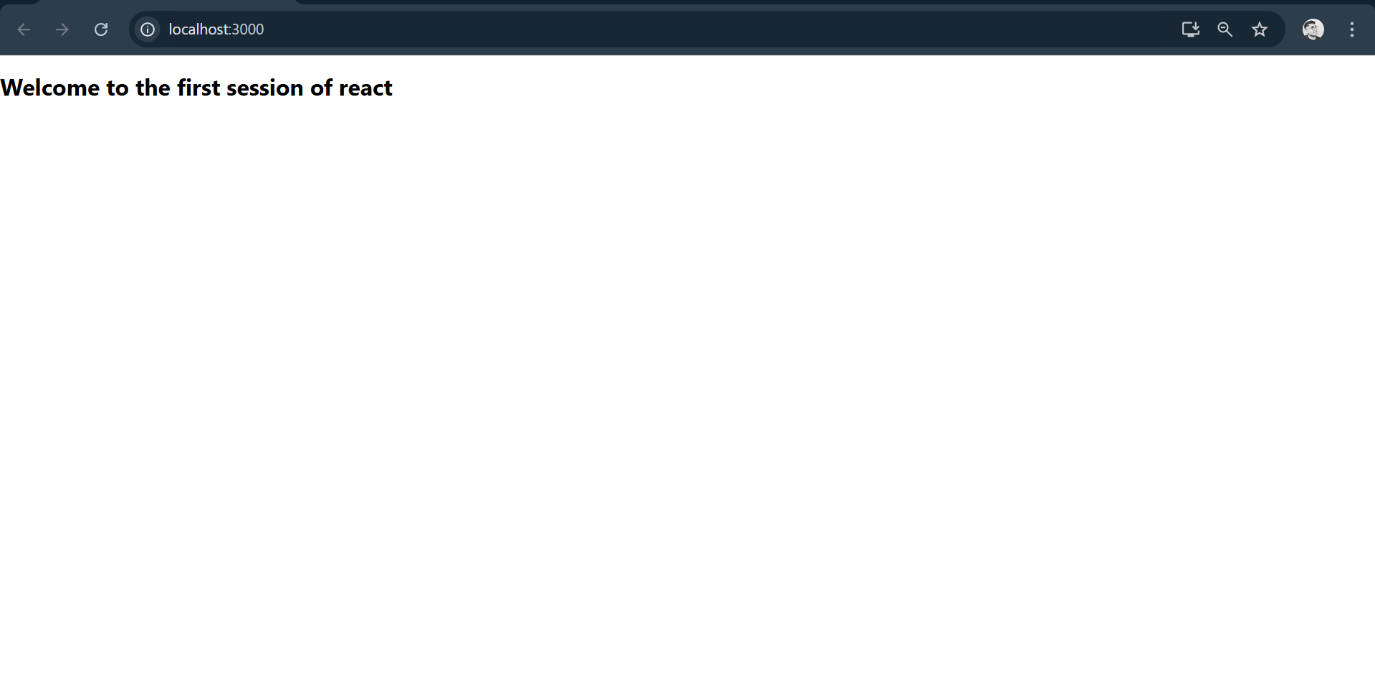
    <h1>Welcome to the first session of react</h1>

    );

}

export default App;

**Output:**



2. 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.

**Objective:**

**Explain React components**  
Independent, reusable pieces of UI. Components manage their own state and rendering.

**Identify the differences between components and JavaScript functions**  
Components return UI elements (JSX) and manage rendering & state; regular functions just perform logic.

**Identify the types of components**

* **Class components**: ES6 classes, can use lifecycle methods, manage state.
* **Function components**: Simple JS functions that return JSX; can use hooks for state and side effects.

**Explain class component**  
A React component defined as an ES6 class extending React.Component, includes lifecycle methods & constructor.

**Explain function component**  
A plain JS function that returns JSX; from React 16.8+, can use hooks for state and lifecycle.

**Define component constructor**  
A method in a class component (constructor(props)) to initialize state and bind methods.

**Define render() function**  
A required method in class components that returns JSX describing the UI.

**Home.js:**

import React from 'react';

function Home() {

return (

<div>

<h2>Welcome to the Home page of Student Management Portal</h2>

</div>

);

}

export default Home;

**About.js:**

import React from 'react';

function About() {

return (

<div>

<h2>Welcome to the About page of the Student Management Portal</h2>

</div>

);

}

export default About;

**Contact.js:**

import React from 'react';

function Contact() {

return (

<div>

<h2>Welcome to the Contact page of the Student Management Portal</h2>

</div>

);

}

export default Contact;

**App.js:**

import React from 'react';

import Home from './Components/Home';

import About from './Components/About';

import Contact from './Components/Contact';

function App() {

return (

<div className="App">

<Home />

<About />

<Contact />

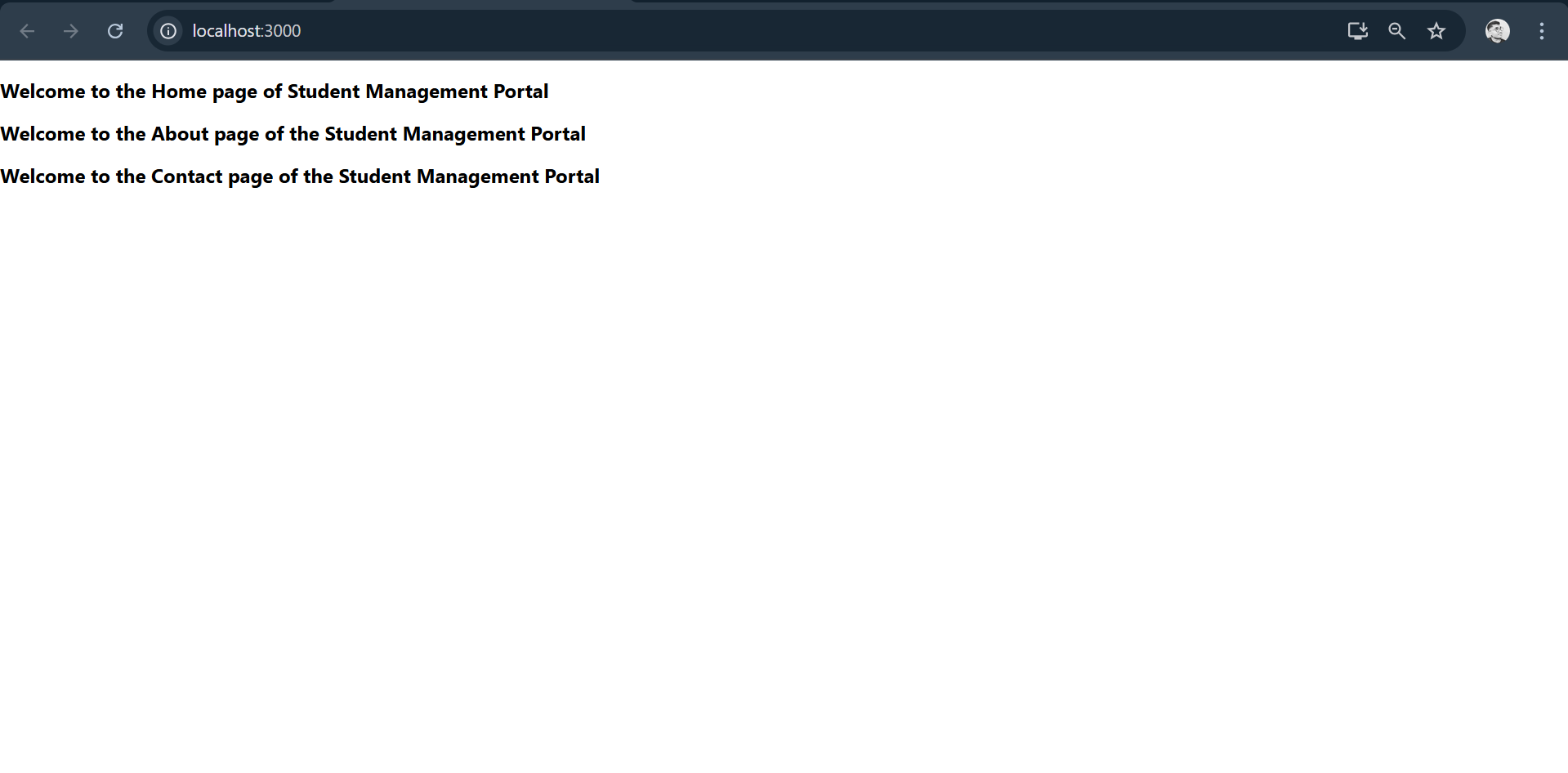
</div>

);

}

export default App;

**Output:**



3. 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.

**Objective:**

**Explain React components**

React components are independent, reusable pieces of UI that return React elements describing what should appear on the screen. Components let you split the UI into isolated, reusable sections, making your code more modular and easier to maintain.

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

* **Purpose**: Regular JavaScript functions perform computations or return values, whereas React components return JSX to describe UI.
* **Usage**: React components are intended to be used in rendering the UI tree and can manage state and lifecycle methods (in class components) or use hooks (in function components). JavaScript functions don’t have this capability.
* **Naming**: Components typically start with a capital letter (e.g., MyComponent), while regular functions can have any name.

**Identify the types of components**

There are two main types of React components:

1. **Class Components**: Use ES6 classes; can have state and lifecycle methods.
2. **Function Components**: Simple JavaScript functions; originally stateless, but now (with hooks) can manage state and side effects.

**Explain class component**

A class component is a JavaScript class that extends React.Component and must define a render() method that returns JSX. It can have:

* State (using this.state)
* Lifecycle methods (e.g., componentDidMount)
* Props accessed via this.props
* **Explain function component**
* A function component is a simpler form of component defined as a JavaScript function that returns JSX. With React hooks, function components can now also manage state and side effects.

**Define component constructor**

In a class component, the constructor is a special method used to:

* Initialize state (this.state)
* Bind methods if needed

It is called before the component is mounted.

**Define render() function**

The render() function is required in class components. It:

* Returns JSX that defines what the UI should look like.
* Is called whenever the component’s state or props change to update the UI.

**CalculateScore.js:**

import React from "react";

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

function CalculateScore() {

const name = "John Doe";

const school = "Springfield High";

const total = 450;

const goal = 500;

const average = ((total / goal) \* 100).toFixed(2);

return (

<div className="score-container">

<h2>Student Score Summary</h2>

<p><strong>Name:</strong> {name}</p>

<p><strong>School:</strong> {school}</p>

<p><strong>Total Score:</strong> {total}</p>

<p><strong>Goal:</strong> {goal}</p>

<p><strong>Average Score:</strong> {average}%</p>

</div>

);

}

export default CalculateScore;

**mystyle.css:**

.score-container {

border: 2px solid #4CAF50;

padding: 20px;

margin: 20px;

background-color: #f0f8ff;

font-family: Arial, sans-serif;

}

.score-container h2 {

color: #4CAF50;

}

**App.js:**

import React from 'react';

import CalculateScore from './Components/CalculateScore';

function App() {

return (

<div className="App">

<CalculateScore />

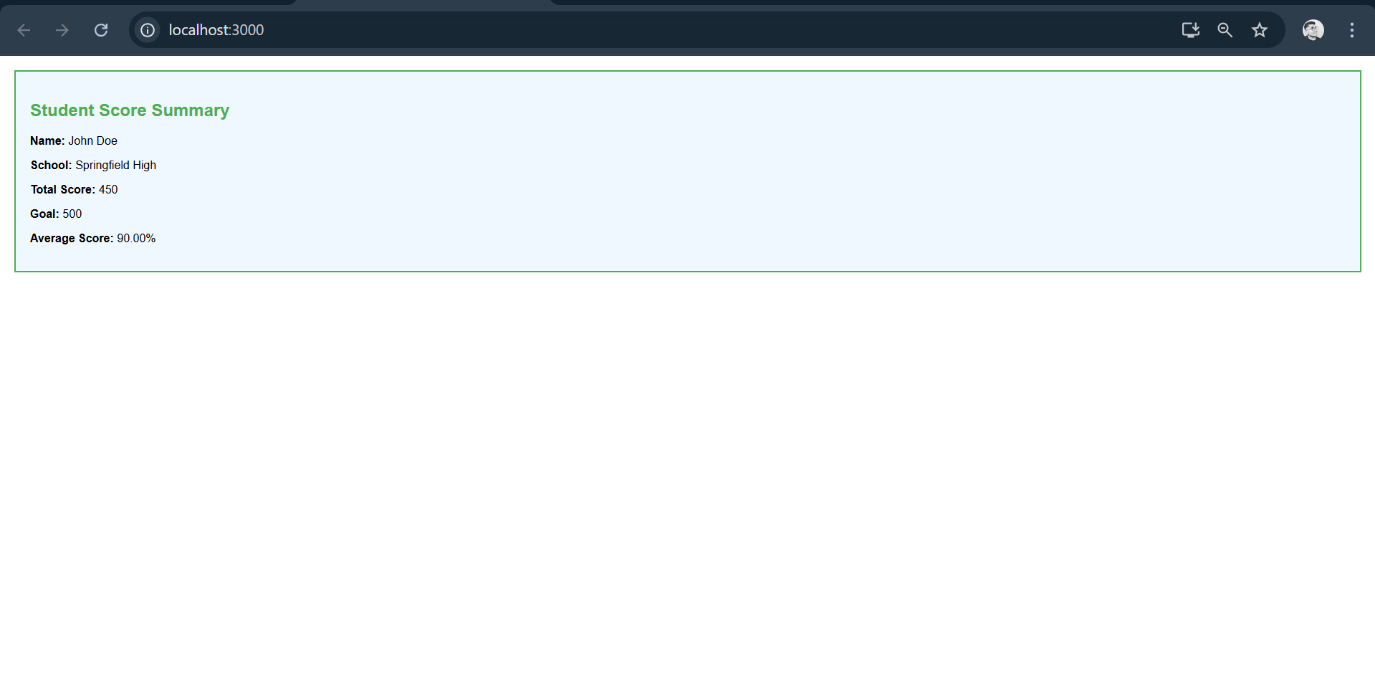
</div>

);

}

export default App;

**Output:**



4. Create a new react application using *create-react-app* tool with the name as “blogapp”

**Objective:**

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

**Need:** Understanding the component lifecycle in React is essential because it helps developers manage the different stages a component goes through (creation, updating, and removal from the DOM).

**Benefits:** It allows you to:

Efficiently initialize data (e.g., fetching data from an API when the component loads)

Update the UI in response to prop or state changes

Clean up resources (such as timers or subscriptions) before a component is removed to prevent memory leaks

Optimize rendering performance by making updates only when necessary

**Identify various life cycle hook methods**

In React (specifically class components), the lifecycle hook methods include:

* + constructor()
  + componentDidMount()
  + componentDidUpdate()
  + componentWillUnmount()
  + (There are others like shouldComponentUpdate() and getDerivedStateFromProps(), but these are the main ones.)

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

The basic sequence when a React class component renders is:

* + - **Mounting phase:**
      * Constructor is called (for initialization)
      * render() method is called (to display the component)
      * componentDidMount() is called (side-effects or data-fetching)
    - **Updating phase:**
      * When state or props change, render() is called again
      * componentDidUpdate() is called after the update renders
    - **Unmounting phase:**
      * componentWillUnmount() is called just before removing the component from the DOM

**Post.js:**

class Post {

constructor(userId, id, title, body) {

this.userId = userId;

this.id = id;

this.title = title;

this.body = body;

}

}

export default Post;

**Posts.js:**

import React, { Component } from 'react';

import Post from './Post';

class Posts extends Component {

constructor(props) {

super(props);

this.state = {

posts: [],

error: null

};

}

loadPosts = () => {

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

.then(response => {

if (!response.ok) {

throw new Error("Network response was not ok.");

}

return response.json();

})

.then(data => {

const postObjects = data.map(

item => new Post(item.userId, item.id, item.title, item.body)

);

this.setState({ posts: postObjects });

})

.catch(error => {

this.setState({ error });

throw error; // Triggers componentDidCatch

});

};

componentDidMount() {

this.loadPosts();

}

componentDidCatch(error, info) {

alert("An error occurred while loading posts.");

console.error("Error caught in component:", error, info);

}

render() {

const { posts, error } = this.state;

if (error) {

return <p style={{ color: 'red' }}>Something went wrong!</p>;

}

return (

<div>

<h2>Blog Posts</h2>

{posts.map(post => (

<div key={post.id} style={{ borderBottom: "1px solid #ccc", marginBottom: "15px" }}>

<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 />

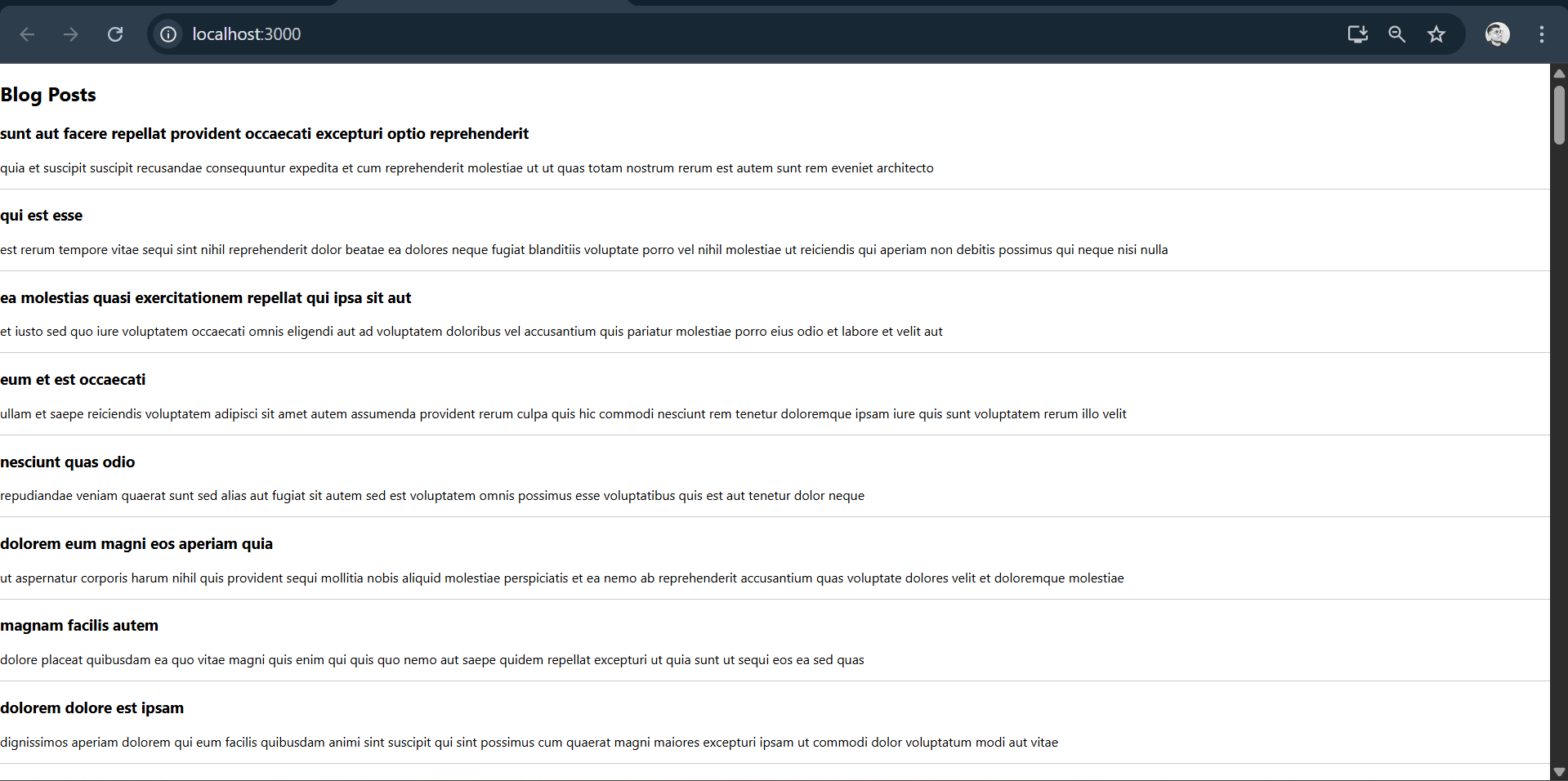
</div>

);

}

export default App;

**Output:**



5. 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

Objective:

**Understand the need for styling React components**

Styling React components is essential to make applications visually appealing, user-friendly, and consistent with brand guidelines. Styles help organize layouts, highlight content, and improve the overall user experience by ensuring that components look good and behave correctly across different devices and screens.

**Learn how to work with CSS Modules and inline styles**

* **CSS Modules:**  
  CSS Modules are files where CSS class names and animations are scoped locally by default, preventing name conflicts. In React, you import a CSS Module (e.g., styles.module.css) and apply styles using an object that maps class names to unique identifiers.
* **Inline Styles:**  
  Inline styles are applied directly to a React element using the style prop, which accepts a JavaScript object representing CSS properties (e.g., { color: "red", fontSize: "20px" }). These styles are scoped to the specific element, avoiding conflicts but lacking features like pseudo-selectors and media queries.

**Define styles using CSS Modules**

1. Create a CSS file with a .module.css extension, for example, Button.module.css.
2. Add your CSS classes inside this file:

css

*/\* Button.module.css \*/*

.primary {

background-color: blue;

color: white;

padding: 10px 20px;

border-radius: 4px;

}

1. Import the module in your component:

javascript

**import** styles **from** './Button.module.css';

**4. Apply styles to components using the className and style properties**

* **Using className with CSS Modules:**  
  Assign a class from the imported CSS Module:

javascript

<button className={styles.primary}>Click Me</button>

* **Using style for inline styles:**  
  Apply a JavaScript object directly to the style prop:

javascript

<button style={{ backgroundColor: "blue", color: "white" }}>Click Me</button>

**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;

}

**CohortDetails.js:**

import React from 'react';

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

function CohortDetails({ cohort }) {

const statusStyle = {

color: cohort.status === 'ongoing' ? 'green' : 'blue'

};

return (

<div className={styles.box}>

<h3 style={statusStyle}>{cohort.name}</h3>

<dl>

<dt>Trainer:</dt>

<dd>{cohort.trainer}</dd>

<dt>Status:</dt>

<dd>{cohort.status}</dd>

<dt>Start Date:</dt>

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

<dt>End Date:</dt>

<dd>{cohort.endDate}</dd>

</dl>

</div>

);

}

export default CohortDetails;

**App.js:**

import React from 'react';

import CohortDetails from './CohortDetails';

function App() {

const cohorts = [

{

name: 'React Bootcamp',

trainer: 'John Doe',

status: 'ongoing',

startDate: '2025-07-01',

endDate: '2025-08-15'

},

{

name: 'Node.js Essentials',

trainer: 'Jane Smith',

status: 'completed',

startDate: '2025-06-01',

endDate: '2025-07-15'

}

];

return (

<div>

{cohorts.map((cohort, index) => (

<CohortDetails key={index} cohort={cohort} />

))}

</div>

);

}

export default App;

Output:

