**Cognizant Digital Nurture 4.0**

**Name: Siri Chandana Chittipolu**

**Email:** [**sirichittipolu11@gmail.com**](mailto:sirichittipolu11@gmail.com)

**Superset ID: 6386277**

**Mandatory Hands-On Exercises**

1. **ReactJS-HOL:**

**Define SPA and its benefits:**

* A SPA is a web application that loads a single HTML page and updates content dynamically without refreshing the page.
* Benefits: Fast navigation, better user experience, less server load.

**Define React and identify its working:**

* React is a JavaScript library developed by Facebook for building user interfaces, especially SPAs. It uses components and the virtual DOM to efficiently render changes.

**Working:**

* + Uses virtual DOM for efficient UI updates.
  + Follows component-based structure.
  + Supports one-way data flow (parent to child).
  + Automatically updates UI when state or props change.

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

|  |  |
| --- | --- |
| SPA (Single Page App) | MPA (Multi Page App) |
| Loads one HTML page | Loads new pages from server |
| Fast and smooth navigation | Slower due to full reloads |
| Uses client-side routing | Uses server-side routing |

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

Pros:

* Faster performance.
* Smooth user experience.
* Reusable components.

Cons:

* Initial load may be heavy.
* SEO can be tricky.
* Requires JavaScript.

**Explain about React:**

* React is an open-source JavaScript library used for building user interfaces, especially for single-page applications (SPAs). It was developed by Facebook in 2013 and has become one of the most popular tools for frontend development.

**Define virtual DOM:**

* Virtual DOM is a lightweight copy of the actual DOM. React uses it to detect changes and update only the parts of the real DOM that changed, improving speed and efficiency.

**Explain Features of React:**

* Component-based architecture
* Virtual DOM
* JSX syntax
* Unidirectional data flow
* Reusable components

**Steps Followed to complete the task:**

1. Installed Node.js and npm from <https://nodejs.org/en/download>.
2. Installed Create React App globally using:

npm install -g create-react-app

1. Created a new React project:

npx create-react-app myfirstreact

1. Opened the project in Visual Studio Code:

cd myfirstreact

code .

1. Edited the App.js file:

import React from 'react';

import './App.css';

function App() {

return (

<div className="app-container">

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

</div>

);

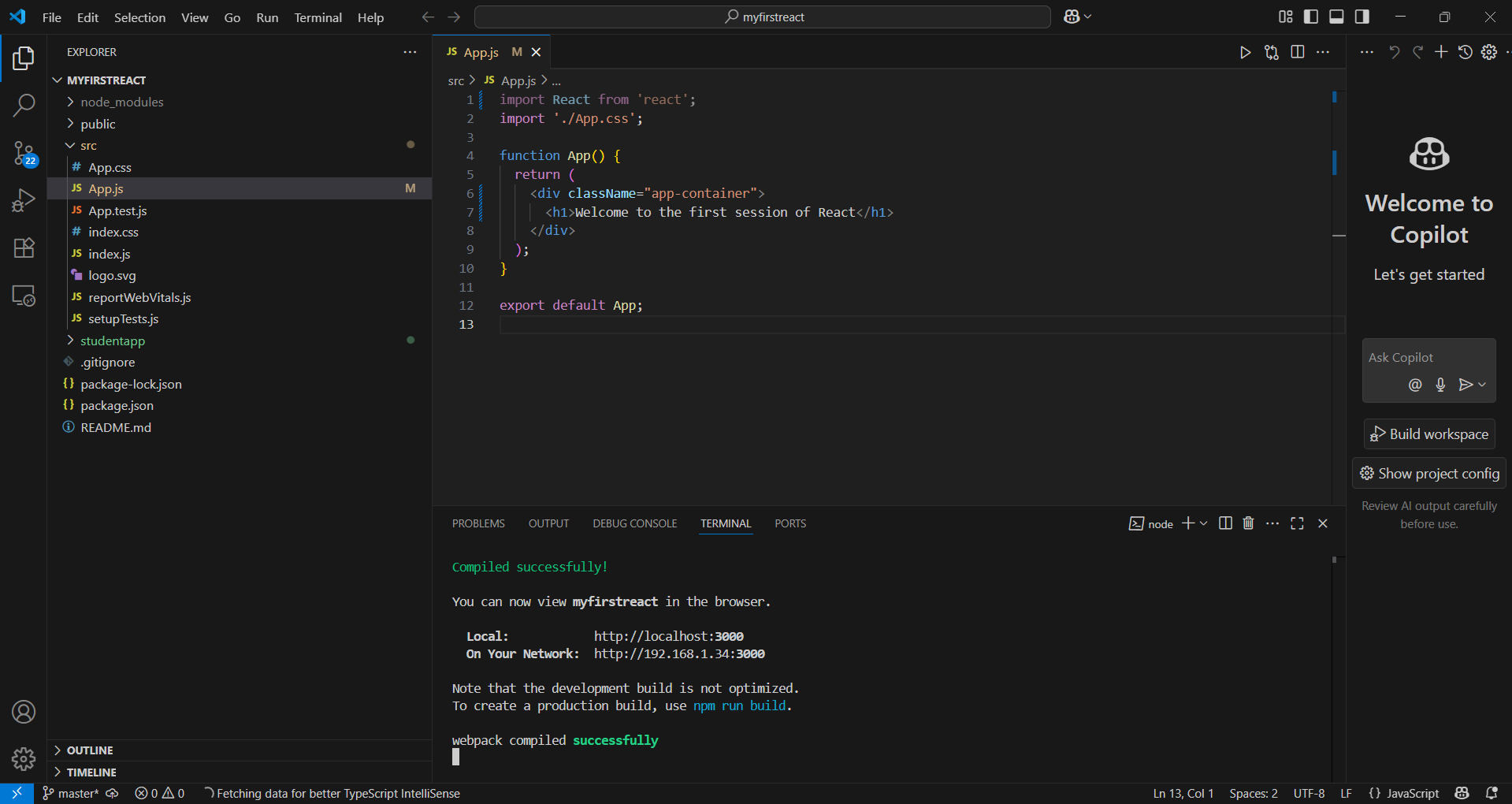
}

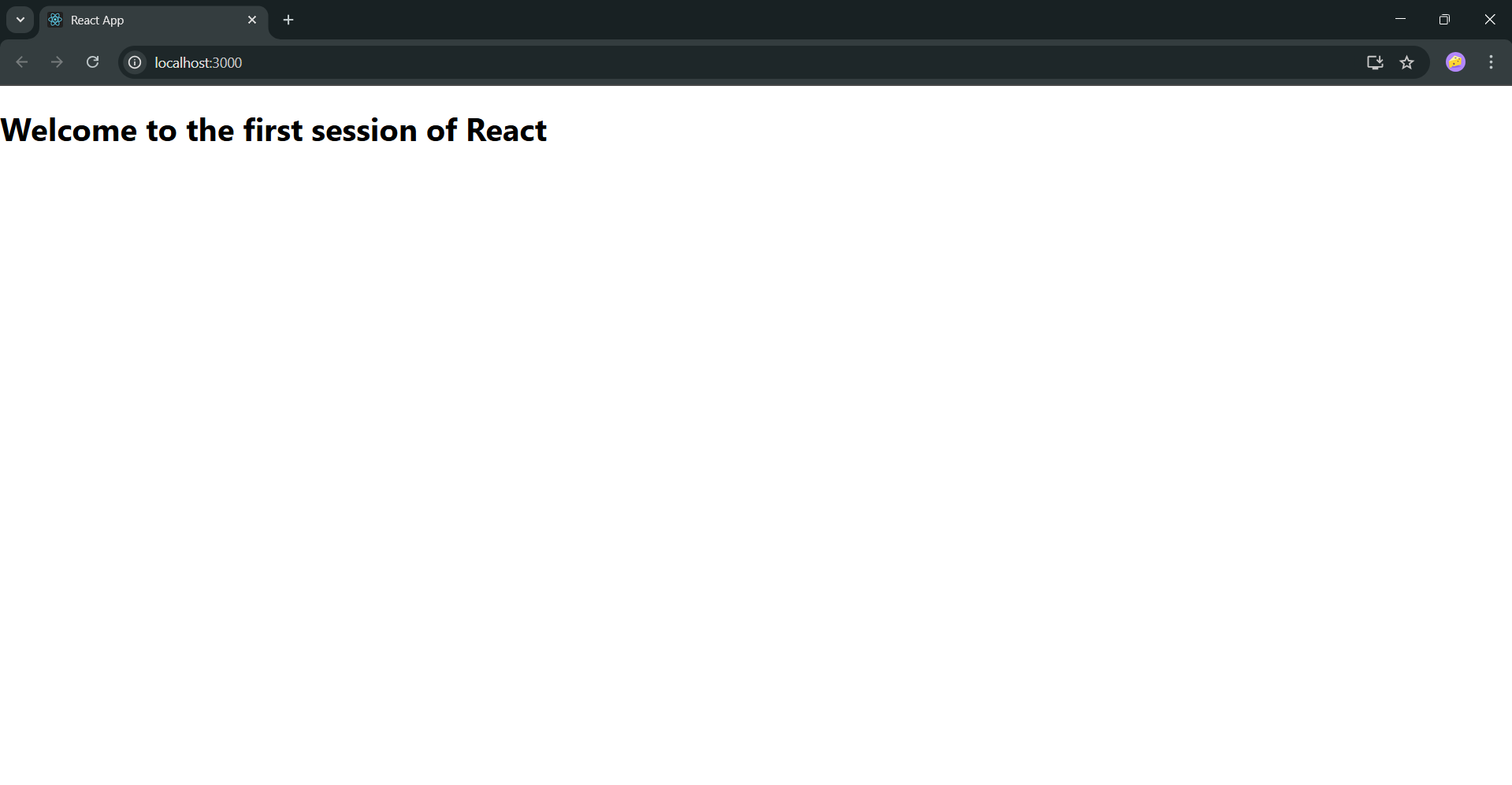
export default App;

1. Started the app using:

npm start

1. **Opened** http://localhost:3000 in the browser to view the output.



****

1. **ReactJS-HOL:**

**Explain React Components:**

React components are the building blocks of any React application. Each component represents a part of the UI and can be reused across the app. They can manage their own state and accept inputs called props.

**Identify Differences Between Components and JavaScript Functions:**

|  |  |  |
| --- | --- | --- |
| Feature | React Components | JavaScript Functions |
| Purpose | Builds UI | Performs logic or calculation |
| Return | JSX (UI elements) | Primitive values or objects |
| Lifecycle methods | Yes (in class components) | No |
| Can manage state | Yes | No |

**Identify Types of Components:**

**React has two main types of components:**

1. Class Components
   * Use ES6 classes
   * Have access to lifecycle methods
   * Use render() to return JSX
2. Function Components
   * Simple JavaScript functions
   * Can use Hooks (like useState) for state
   * Commonly used in modern React

**Explain Function Component:**

A function component is a simple JavaScript function that returns JSX. It does not use a class and is mainly used for UI rendering. You can use React Hooks (like useState, useEffect) inside function components to handle state and side effects.

function Welcome() {

return <h1>Hello from Function Component</h1>;

}

**Define Component Constructor:**

The constructor is a special method used in class components to initialize state or bind methods. It is called before the component is mounted.

constructor(props) {

super(props);

this.state = { name: "Student" };

}

**Define render() Function:**

The render() function is required in class components. It returns the JSX (UI code) that should be displayed on the screen.

render() {

return <h3>Welcome to React</h3>;

}

**Steps Followed to complete the task:**

* Created a new React App using npx create-react-app StudentApp
* Opened project in VS Code
* Created a Components folder inside src/
* Created 3 components using class syntax:

**// Home.js**

import React, { Component } from 'react';

class Home extends Component {

  render() {

    return (

      <div>

        <h3>Welcome to the Home Page of Student Management Portal</h3>

      </div>

    );

  }

}

export default Home;

**//About.js**

import React, { Component } from 'react';

class About extends Component {

  render() {

    return (

      <div>

        <h3>Welcome to the About Page of Student Management Portal</h3>

      </div>

    );

  }

}

export default About;

**//Contact.js**

import React, { Component } from 'react';

class Contact extends Component {

  render() {

    return (

      <div>

        <h3>Welcome to the Contact Page of Student Management Portal</h3>

      </div>

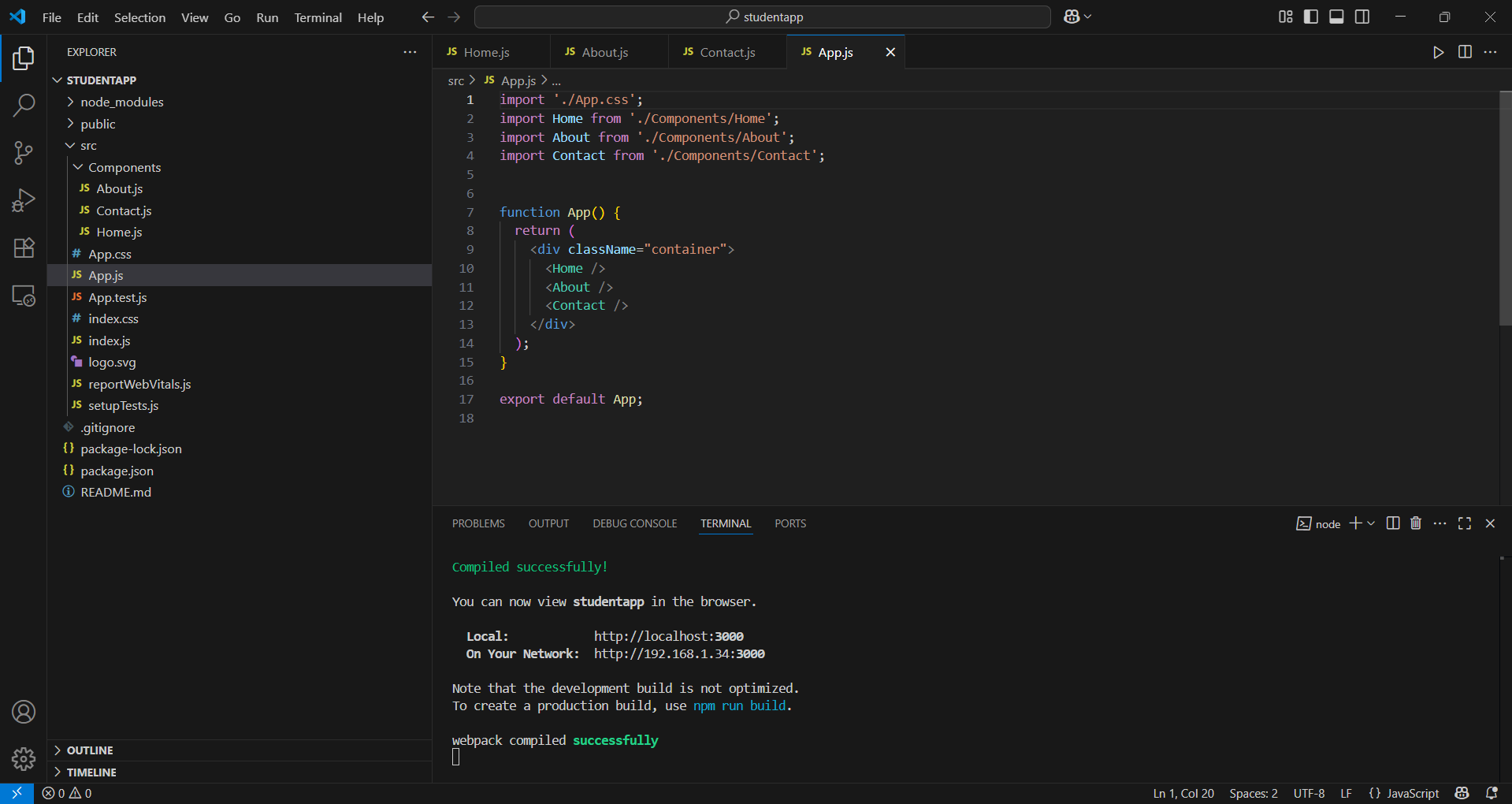
    );

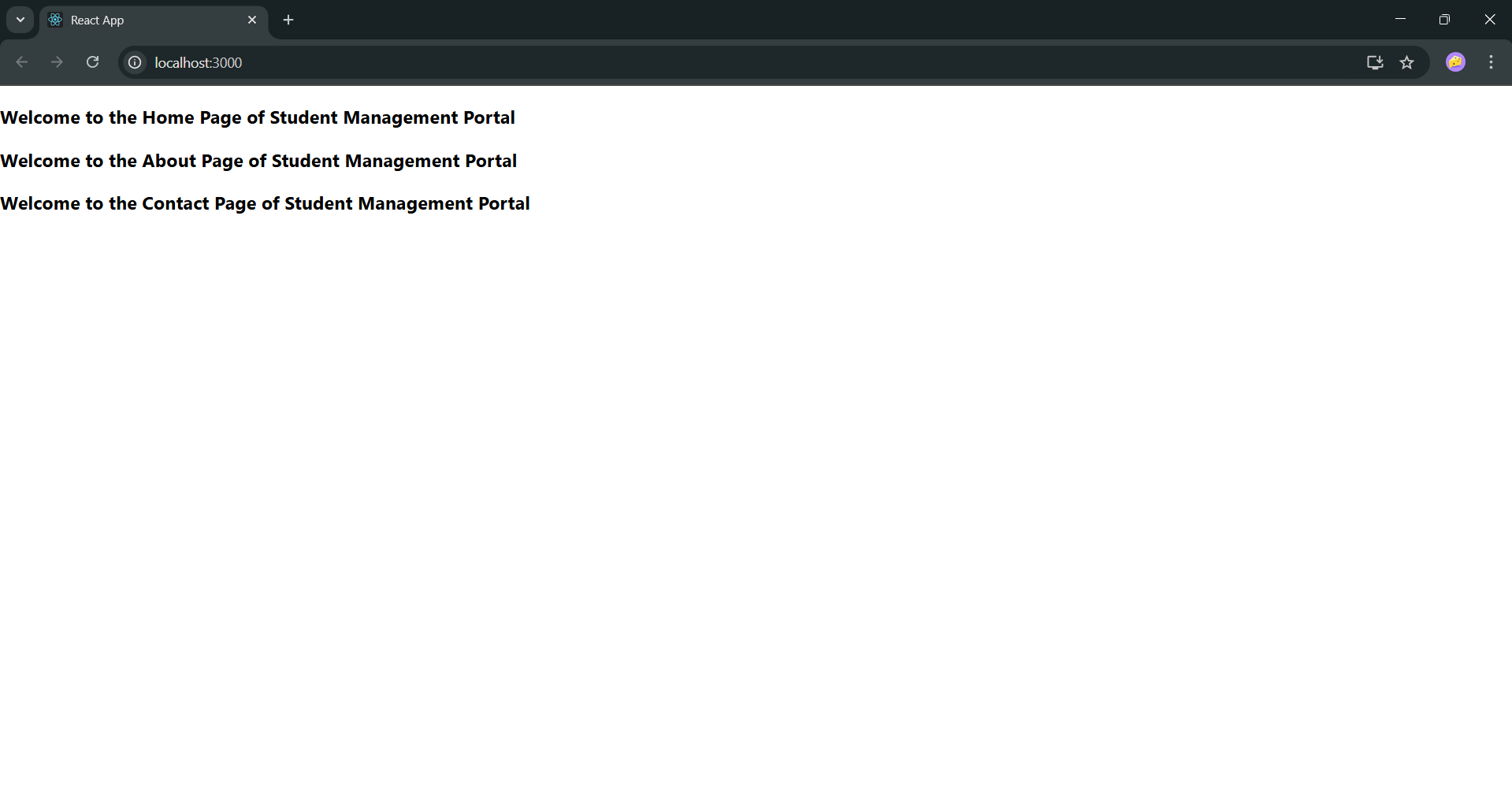
  }

}

export default Contact;

* Each component uses render() method to return a message
* Updated App.js to display all three components
* Started the project using npm start
* Viewed the output in the browser at localhost:3000



****

**3. ReactJS-HOL:**

**Explain React Components:**

React components are independent, reusable pieces of UI in a React application. Each component returns JSX and can be used multiple times across the app. They help organize the UI into smaller, manageable parts.

**Identify the Differences Between Components and JavaScript Functions**

|  |  |  |
| --- | --- | --- |
| Feature | React Component | JavaScript Function |
| Purpose | Builds UI elements | Executes logic or returns values |
| Returns | JSX (UI layout) | Data (numbers, strings, etc.) |
| State Management | Yes (via hooks or class state) | No |
| Lifecycle Methods | Available (in class components) | Not available |

**Identify the Types of Components:**

React has two main types of components:

1. Class Components – Use ES6 classes, support lifecycle methods, and have a render() method.
2. Function Components – Use regular functions. Modern React uses hooks (useState, useEffect) in these components for state and logic.

**Explain Class Component:**

A class component is a component written using a JavaScript class that extends React.Component. It must include a render() method and can hold internal state and lifecycle methods.

Example:

class Welcome extends React.Component {

render() {

return <h1>Hello from Class Component</h1>; } }

**Explain Function Component:**

A function component is a simple JavaScript function that returns JSX. It can accept props and use hooks for state and side effects. It's the preferred way to write components in modern React.

**Example:**

function Welcome() {

return <h1>Hello from Function Component</h1>;

}

**Define Component Constructor:**

The constructor is used in class components to initialize state and bind methods. It is called before the component is mounted.

Syntax:

constructor(props) {

super(props);

this.state = { name: 'React' };

}

**Define render() Function**

The render() function is required in class components. It returns the JSX that will be displayed in the browser.

Example:

render() {

return <h3>Welcome to React</h3>;

}

**Steps:**

1. Creating a New React App

Open the terminal in Visual Studio Code and run:

* npx create-react-app scorecalculatorapp
* cd scorecalculatorapp

2.Create Component Folder

* Navigate to the src directory.
* Create a folder named Components.
* Inside Components, create a file named CalculateScore.js.

**// CalculateScore.js:**

import '../Stylesheets/mystyle.css';

const percentToDecimal = (decimal) => {

  return (decimal.toFixed(2) + '%');

}

const calcScore = (total, goal) => {

  return percentToDecimal(total / goal);

}

export const CalculateScore = ({ Name, School, total, goal }) => (

  <div className="formatstyle">

    <h1><font color="Brown">Student Details:</font></h1>

    <div className="Name">

      <b><span> Name: </span></b>

      <span>{Name}</span>

    </div>

    <div className="School">

      <b><span> School: </span></b>

      <span>{School}</span>

    </div>

    <div className="Total">

      <b><span>Total:</span></b>

      <span>{total}</span>

      <span> Marks</span>

    </div>

    <div className="Score">

      <b>Score:</b>

      <span>

        {calcScore(total, goal)}

      </span>

    </div>

  </div>

)

3.Add Styling

* In the src folder, create a folder named Stylesheets.
* Inside it, create mystyle.css.
* Add the following CSS:

.Name {

font-weight: 300;

color: blue;

}

.School {

color: crimson;

}

.Total {

color: darkmagenta;

}

.formatstyle {

text-align: center;

font-size: large;

}

.Score {

color: forestgreen;

}

4.Update App.js

Replace the contents of src/App.js with:

import { CalculateScore } from './Components/CalculateScore';

function App() {

return (

<div>

<CalculateScore

Name={"Steeve"}

School={"DNV Public School"}

total={284}

goal={3}

/>

</div>

);

}

export default App;

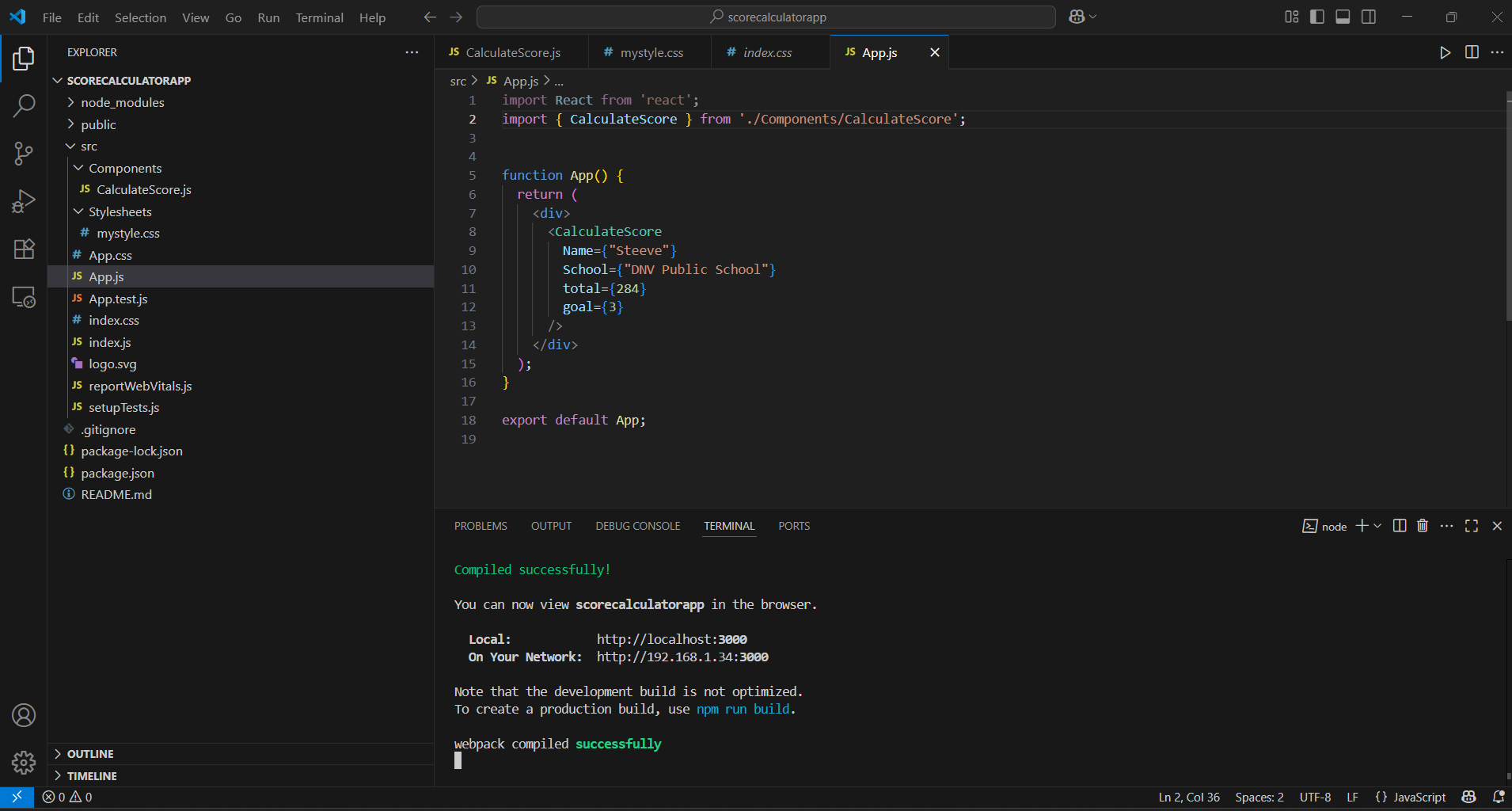
5. Start the React App

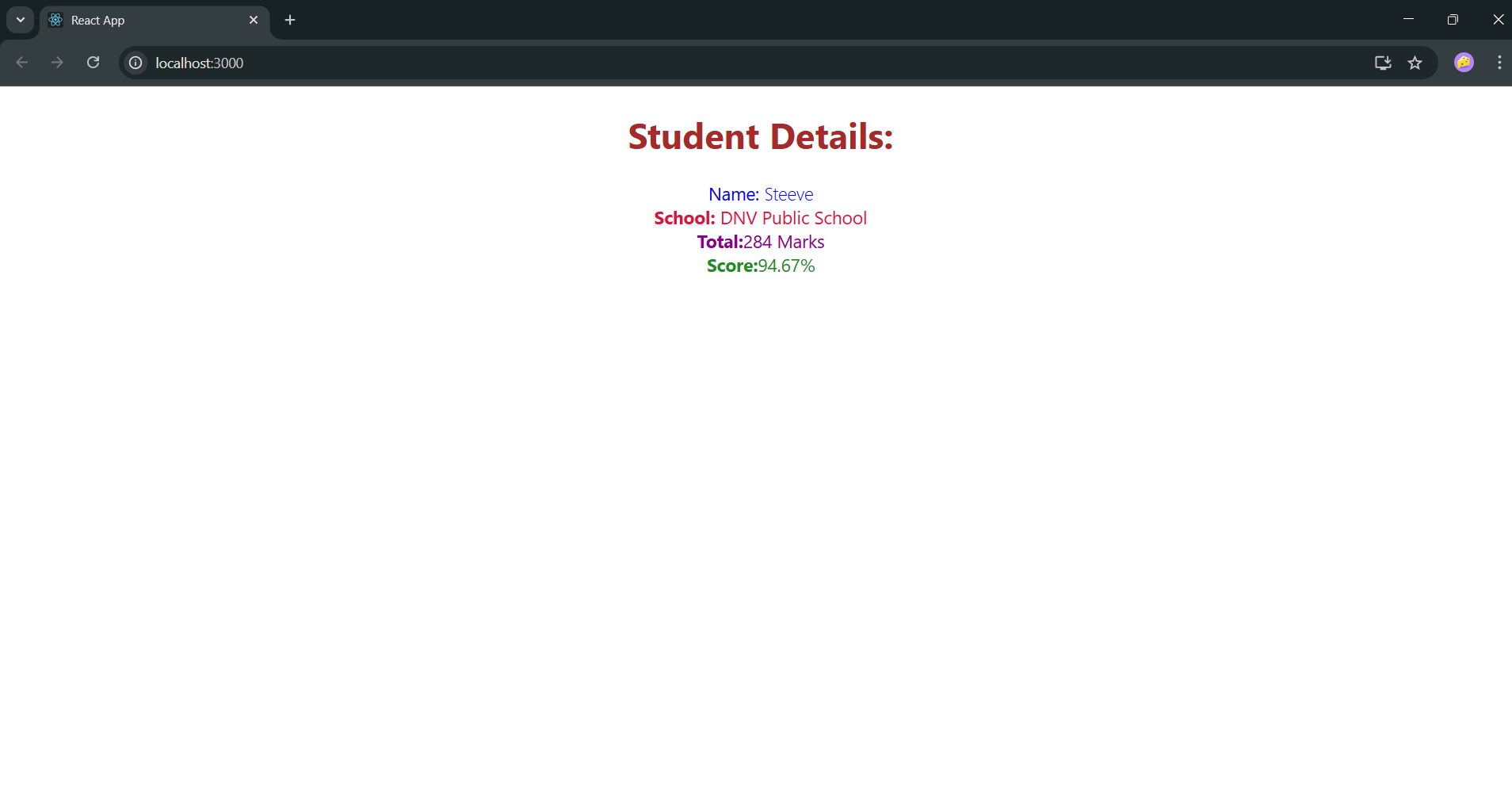
In the terminal, run:

* npm start

Open a browser and go to:

* http://localhost:3000





**4. ReactJS-HOL:**

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

* Controls component behavior during creation, update, and removal.
* Helps manage side effects (like API calls).
* Improves performance and resource cleanup.

**Identify various life cycle hook methods:**

* componentDidMount() – Runs after the component is rendered.
* componentDidUpdate() – Runs after updates.
* componentWillUnmount() – Runs before the component is removed.

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

* Mounting – Component is created (constructor, render, componentDidMount).
* Updating – Component updates due to state/props change.
* Unmounting – Component is removed from the DOM.

**Steps:**

1.Creating a React App:

npx create-react-app blogapp

cd blogapp

code .

2.Creating Post.js in src/ :

class Post {

constructor(id, title, body) {

this.id = id;

this.title = title;

this.body = body;

}

}

export default Post;

3.Creating and Implementing Posts.js Component:

* Import React and the Post class, then define a Posts class component with an initial empty posts array in state.
* Add a loadPosts() method to fetch post data from https://jsonplaceholder.typicode.com/posts, map it to Post objects, and update state.
* Use the componentDidMount() lifecycle hook to call loadPosts() when the component mounts.
* Implement render() to display post titles in <h2> and bodies in <p>, and use componentDidCatch() to show alert messages on errors.

**//Posts.js**

import React from "react";

import Post from "./Post";

class Posts extends React.Component {

  constructor(props) {

    super(props);

    this.state = {

      posts: [],

    };

  }

  loadPosts() {

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

      .then((response) => response.json())

      .then((data) => {

        const postObjects = data.map(

          (p) => new Post(p.id, p.title, p.body)

        );

        this.setState({ posts: postObjects });

      })

      .catch((error) => {

        console.error("Error fetching posts:", error);

      });

  }

  componentDidMount() {

    this.loadPosts();

  }

  componentDidCatch(error, info) {

    alert("An error occurred: " + error);

    console.log(info);

  }

  render() {

    return (

      <div>

        <h1>Blog Posts</h1>

        {this.state.posts.map((post) => (

          <div key={post.id}>

            <h2>{post.title}</h2>

            <p>{post.body}</p>

          </div>

        ))}

      </div>

    );

  }

}

export default Posts;

4. Update App.js:

import Posts from "./Posts";

function App() {

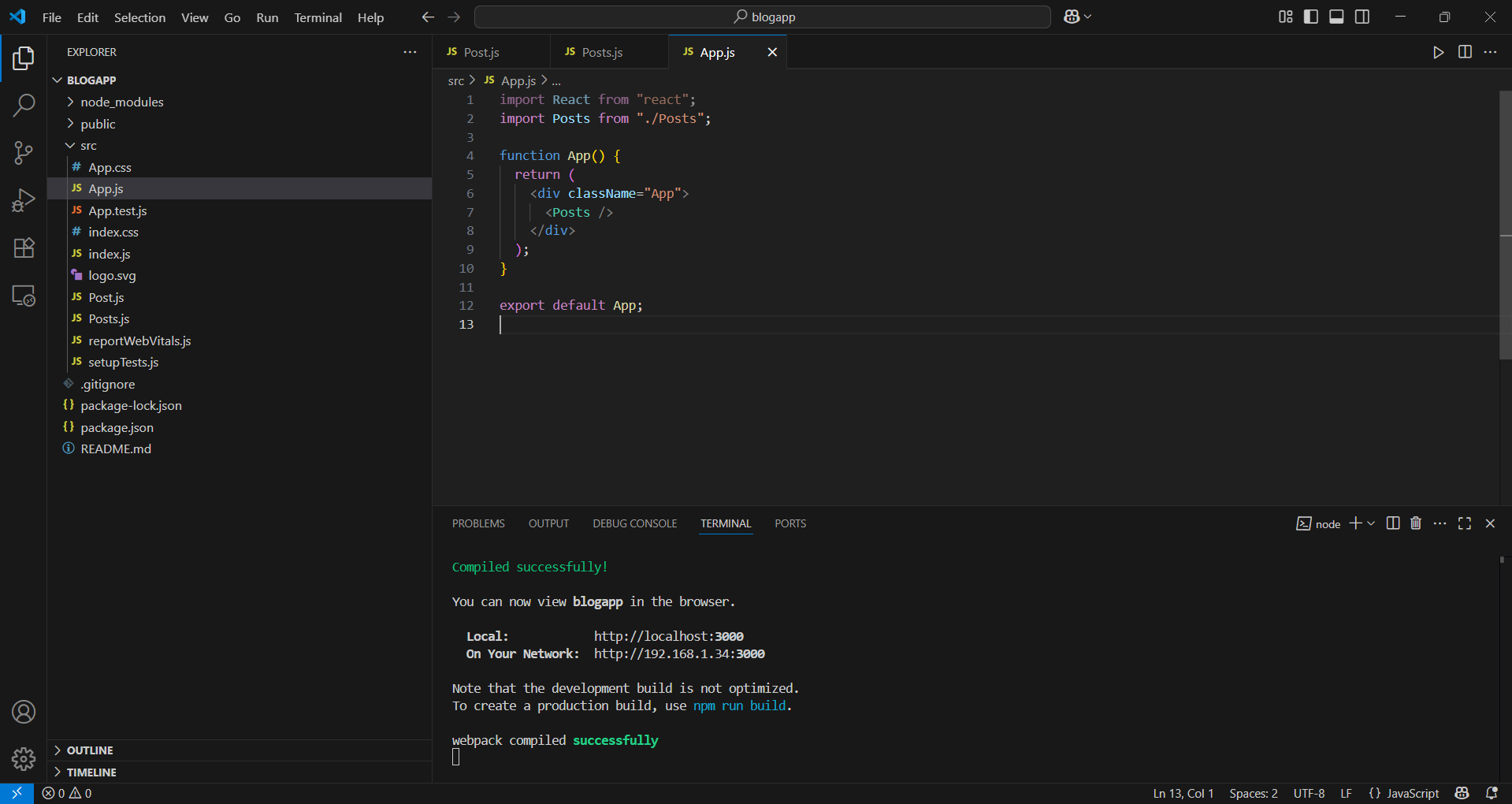
return <Posts />;

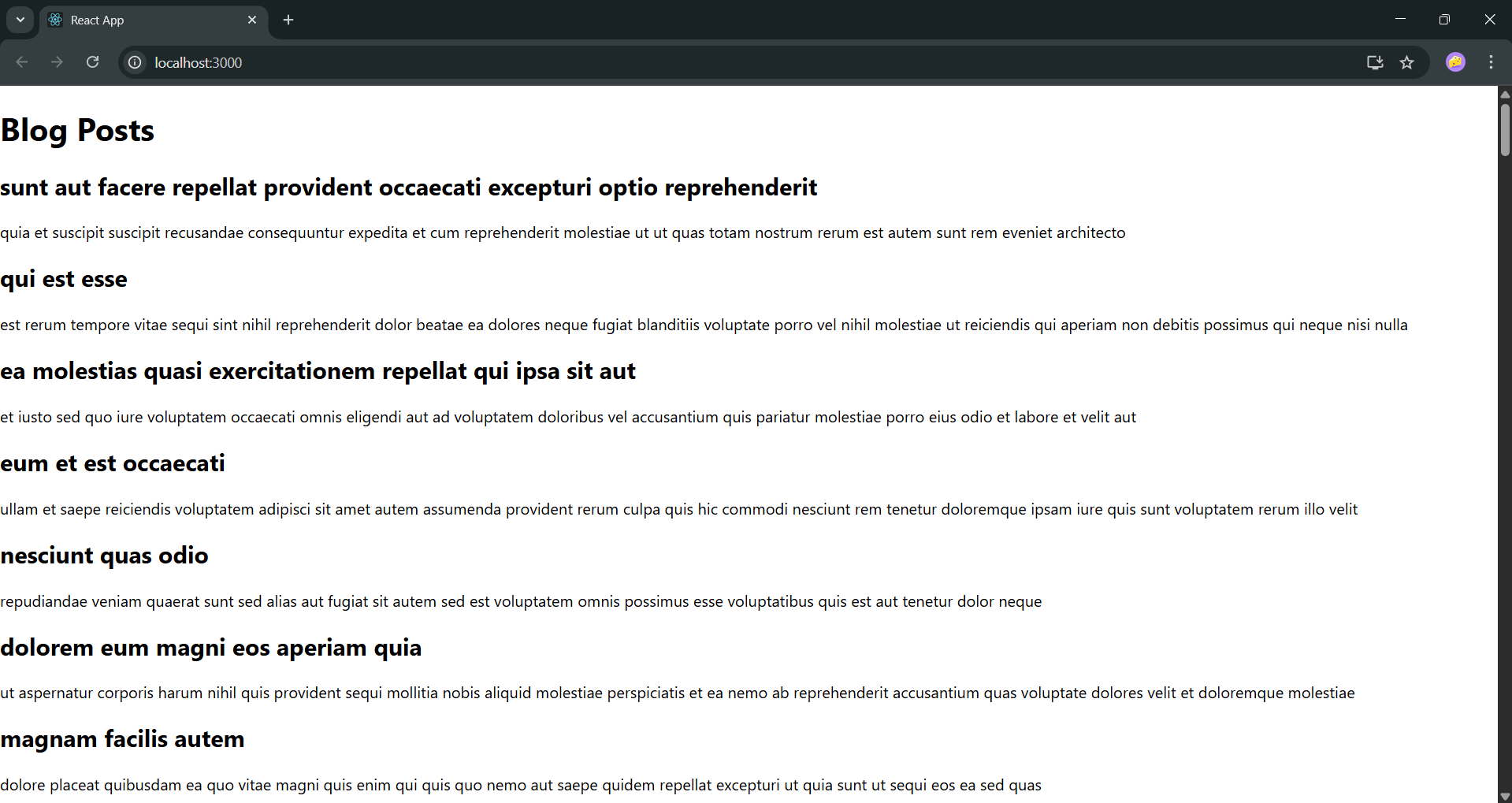
}

export default App;

5.Running the App:

* npm start

****

****

**5.ReactJS-HOL:**

**Understanding the Need for Styling React Components:**  
Styling in React ensures components are visually appealing and user-friendly, enhancing UI consistency and usability across the app.

**Working with CSS Modules and Inline Styles**

* CSS Modules: Scoped styling using .module.css files, preventing class name conflicts.

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

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

* Inline Styles: JavaScript-style objects for dynamic, component-scoped styles.

const style = { color: 'blue', padding: '10px' };

<div style={style}>Hello</div>

**Steps:**

1.Unzip the recent React application folder.

2.Open the folder in VS Code and run npm install to install dependencies.

3.Create a CSS Module file named CohortDetails.module.css with the following styles:

.box {

width: 300px;

display: inline-block;

margin: 10px;

padding: 10px 20px;

border: 1px solid black;

border-radius: 10px;

}

dt {

font-weight: 500;

}

4.Create or open CohortDetails.js component and:

* Import the CSS module:  
  import styles from './CohortDetails.module.css';
* Use the styles and add inline style for <h3> based on cohort status:

<div className={styles.box}>

<h3 style={{ color: cohort.status === 'ongoing' ? 'green' : 'blue' }}>

{cohort.name}

</h3>

<dl>

<dt>Start Date:</dt>

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

<dt>Status:</dt>

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

</dl>

</div>

5.In App.js, define a cohort data array and render the component for each cohort:

const cohorts = [

{ id: 1, name: 'React Basics', startDate: '2025-07-01', status: 'ongoing' },

{ id: 2, name: 'Advanced Node', startDate: '2025-05-15', status: 'completed' },

{ id: 3, name: 'Fullstack Development', startDate: '2025-08-10', status: 'ongoing' }

];

function App() {

return (

<div className="app-container">

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

{cohorts.map(cohort => (

<CohortDetails key={cohort.id} cohort={cohort} />

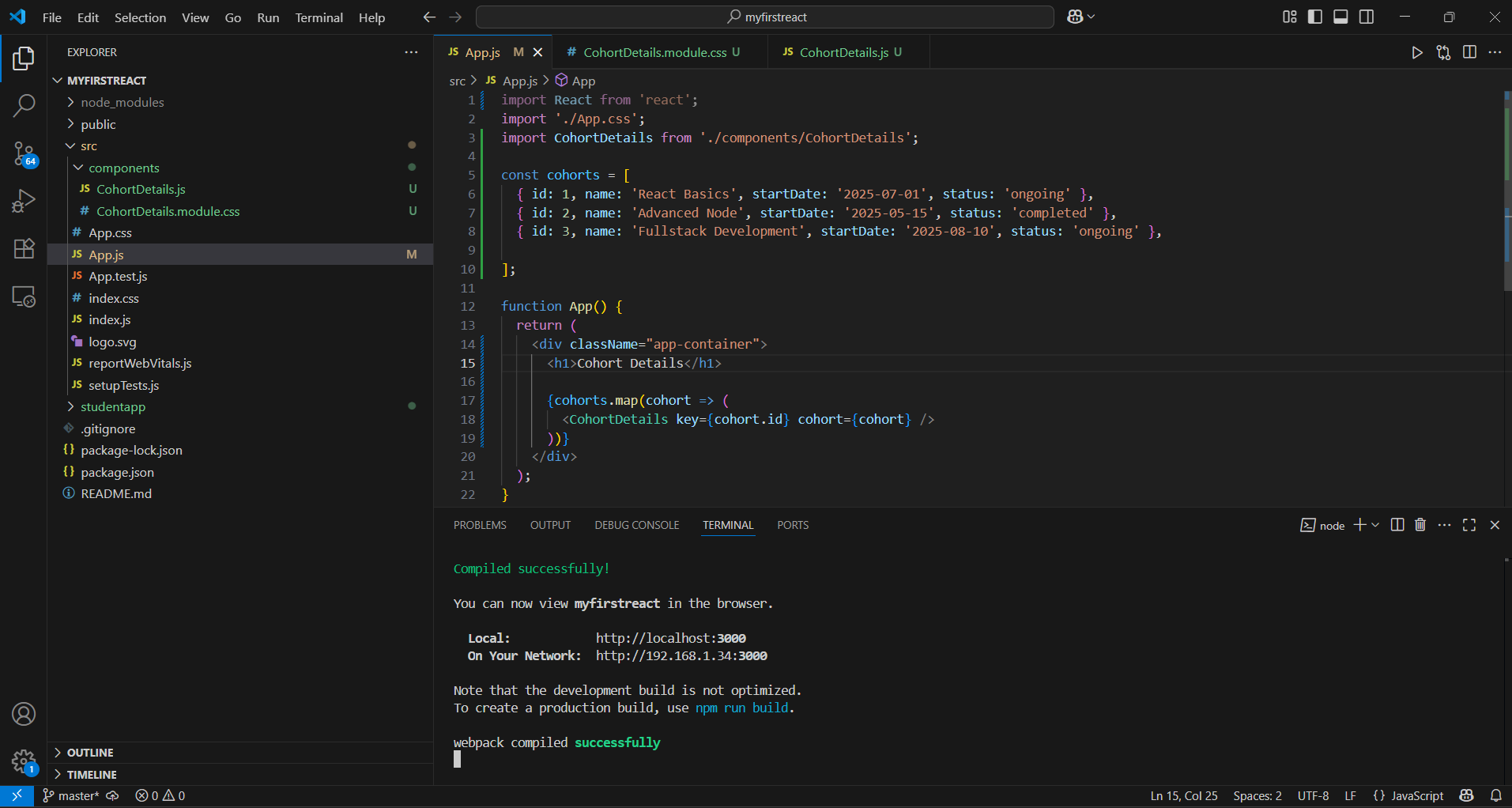
))}

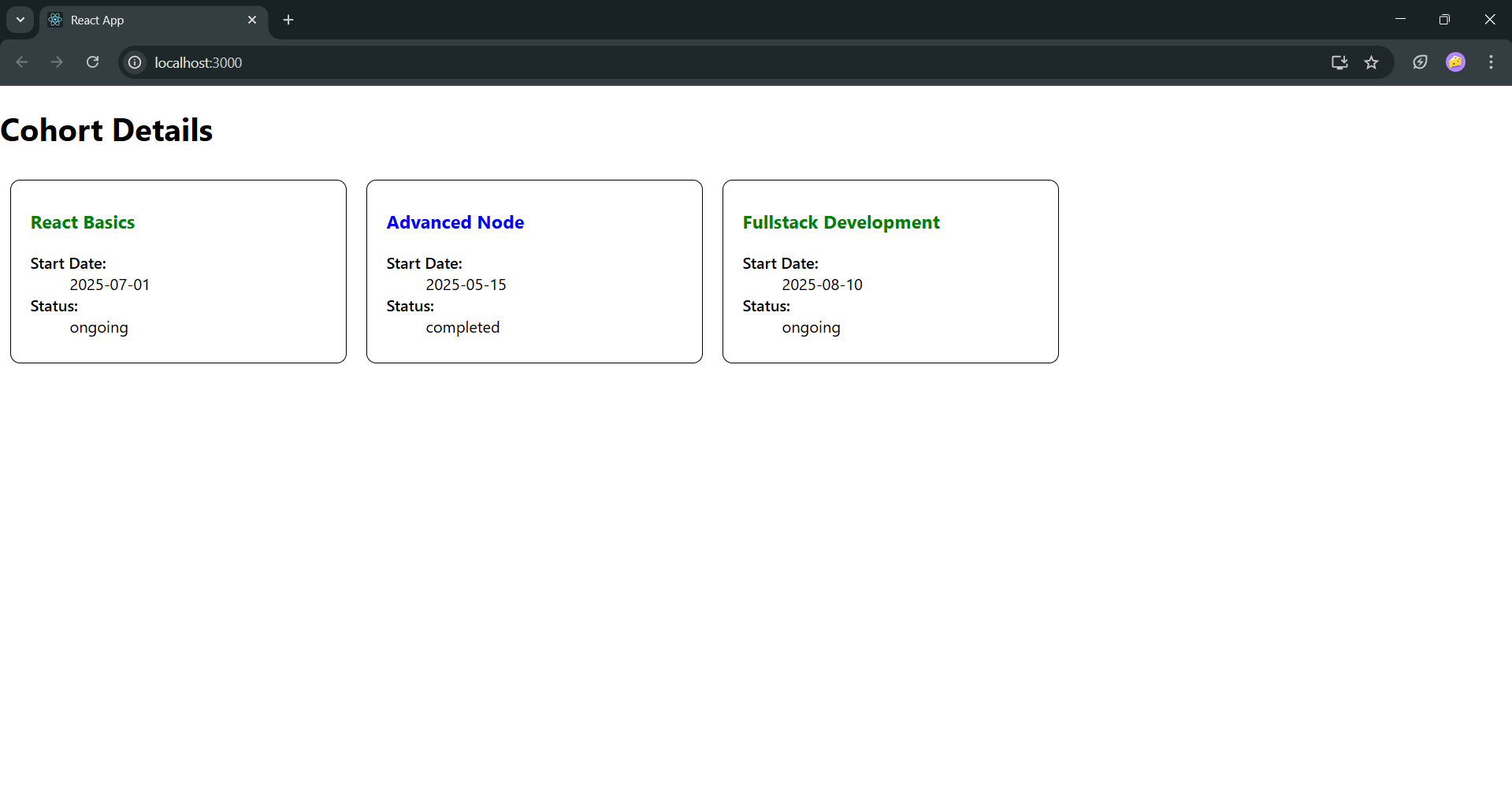
</div>

);

}

6.Run the app using npm start and verify the styled components.

****

****