**Week 6**

**1.ReactJS Hands-On**

**Creating Your First React Application**

**Objectives:**

* Understand what SPA (Single Page Application) is and its benefits.
* Understand React and how it works.

**Prerequisites:**

* Node.js installed
* NPM installed
* Visual Studio Code installed

**Code**

**App.js:**

import React from 'react';

function App() {

return (

<div>

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

</div>

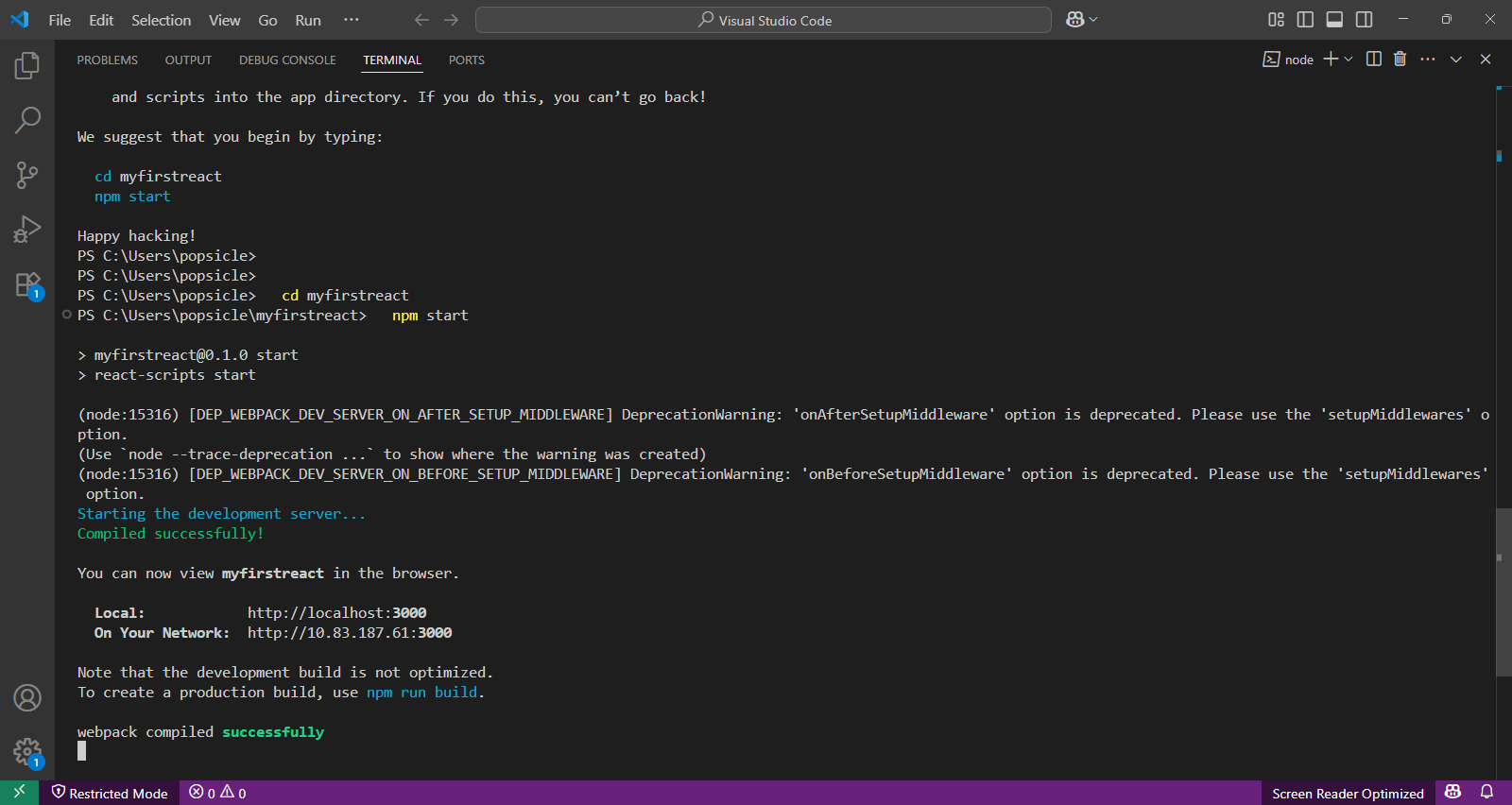
);

}

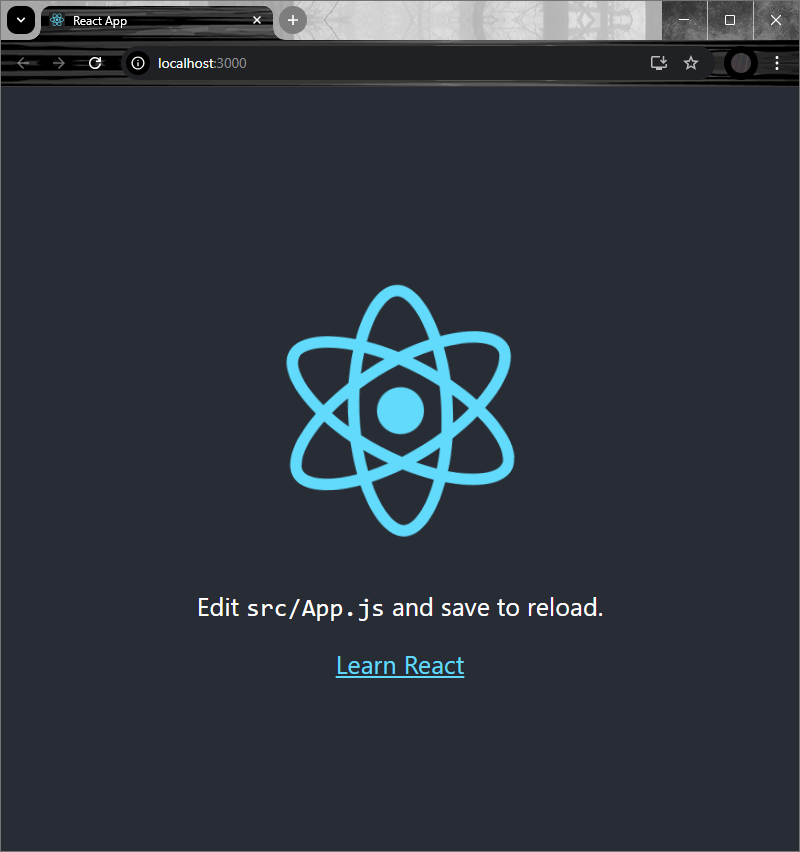
export default App;

**Output Screenshot**

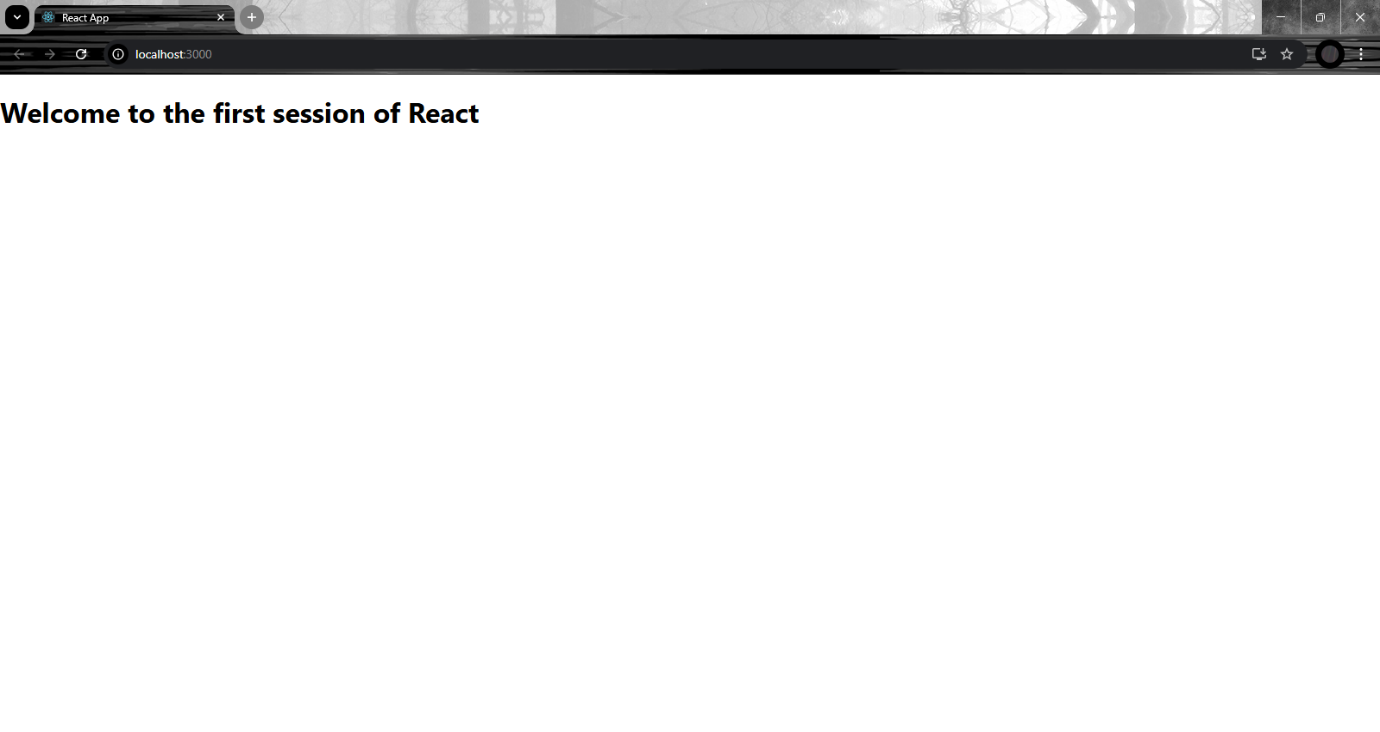
**npm start:**



React app:



Output:



**Conclusion**

You learned how to set up a React environment and create your first app using create-react-app. You successfully displayed a welcome message, gaining basic knowledge of React structure and components.

**2.ReactJS Hands-On**

*Creating a Student Management Portal with Multiple Components*

**Objectives:**

* Learn about React components
* Understand functional vs class components
* Create and render multiple components
* Pass props and organize components

**Code**

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>

<Home />

<About />

<Contact />

</div>

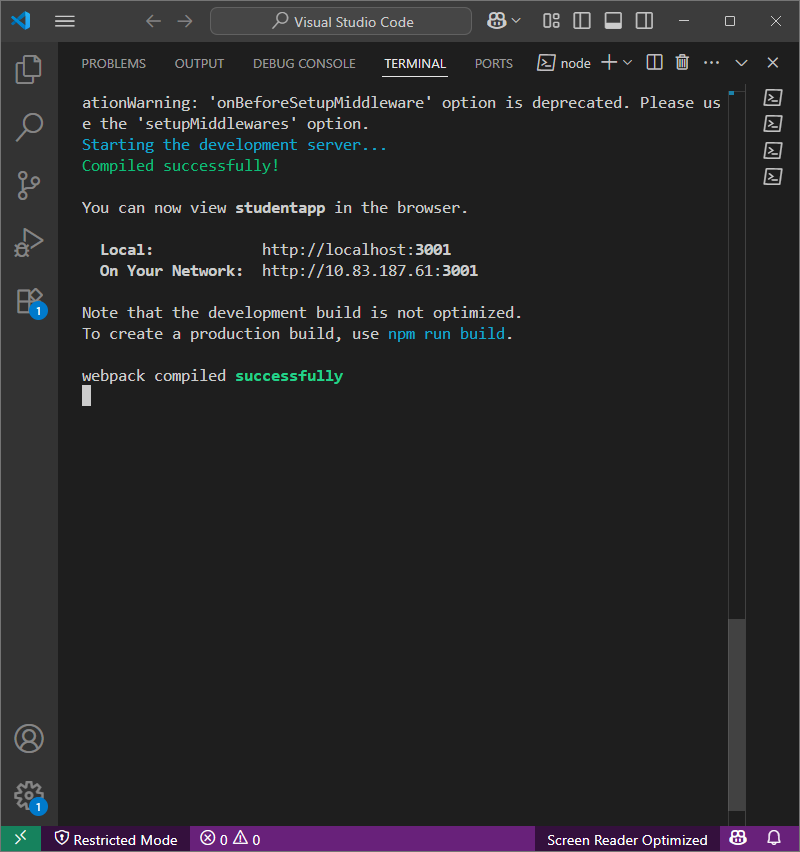
);

}

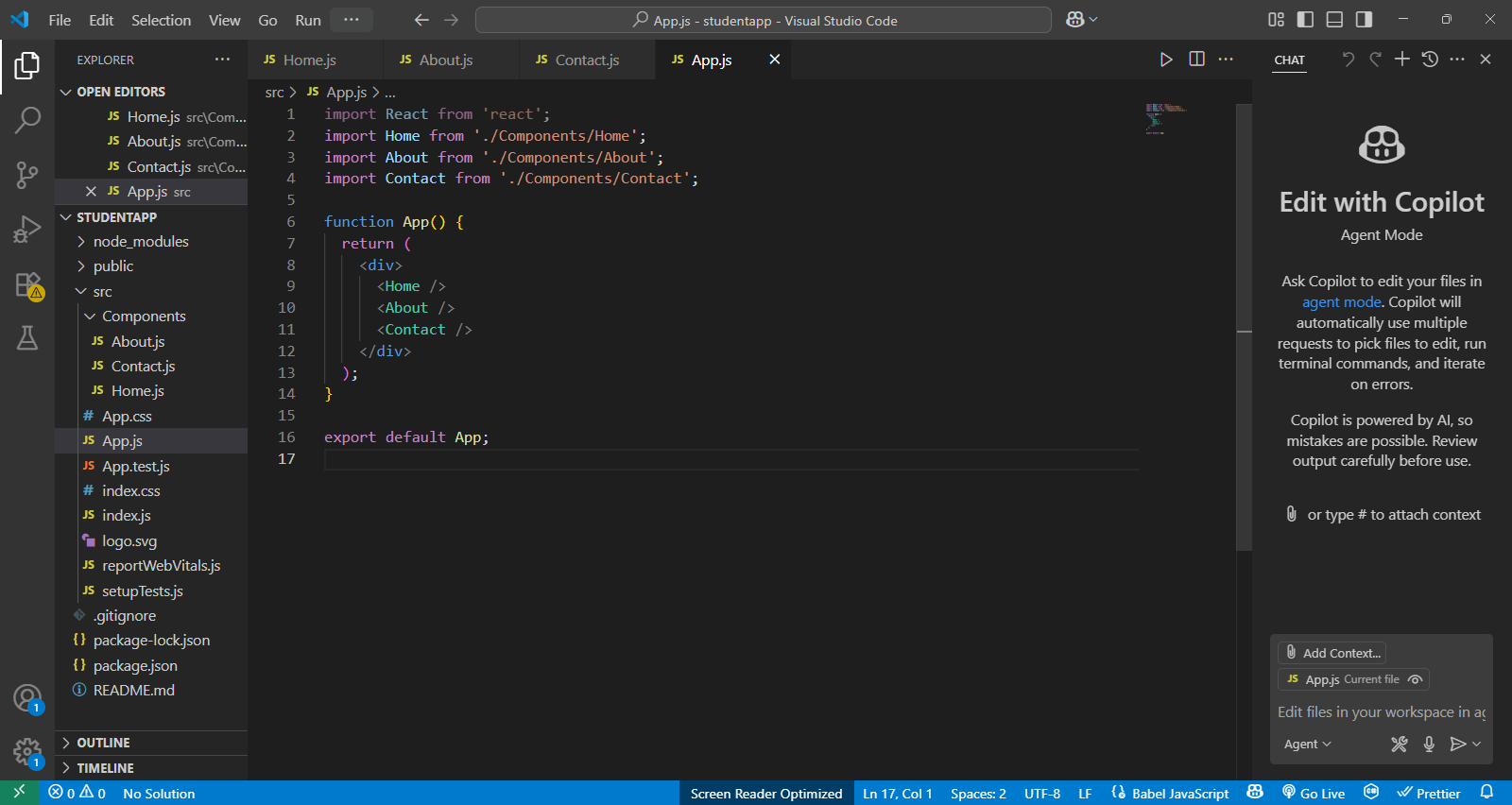
export default App;

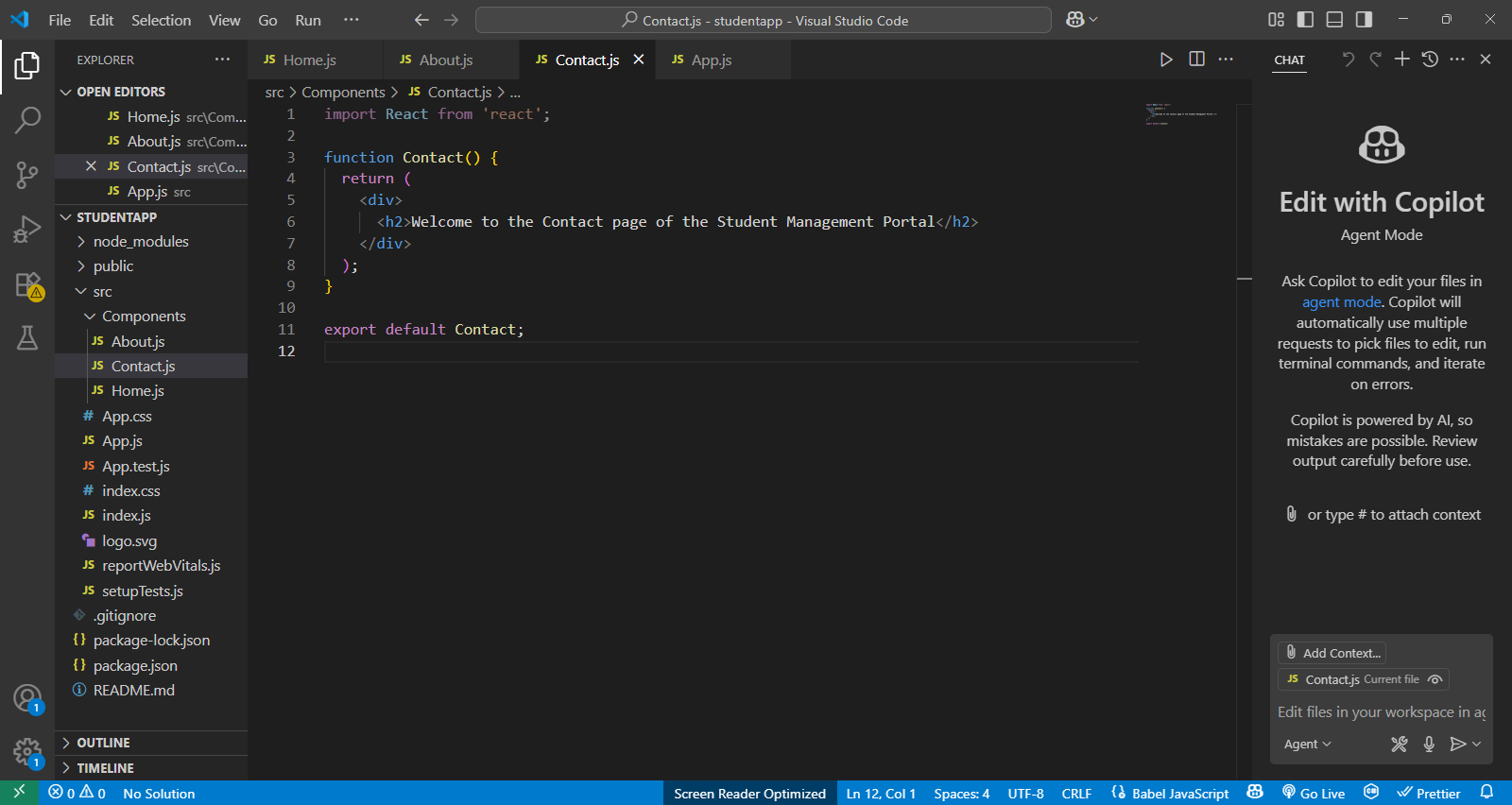
**Screenshots**

**Npm start:**

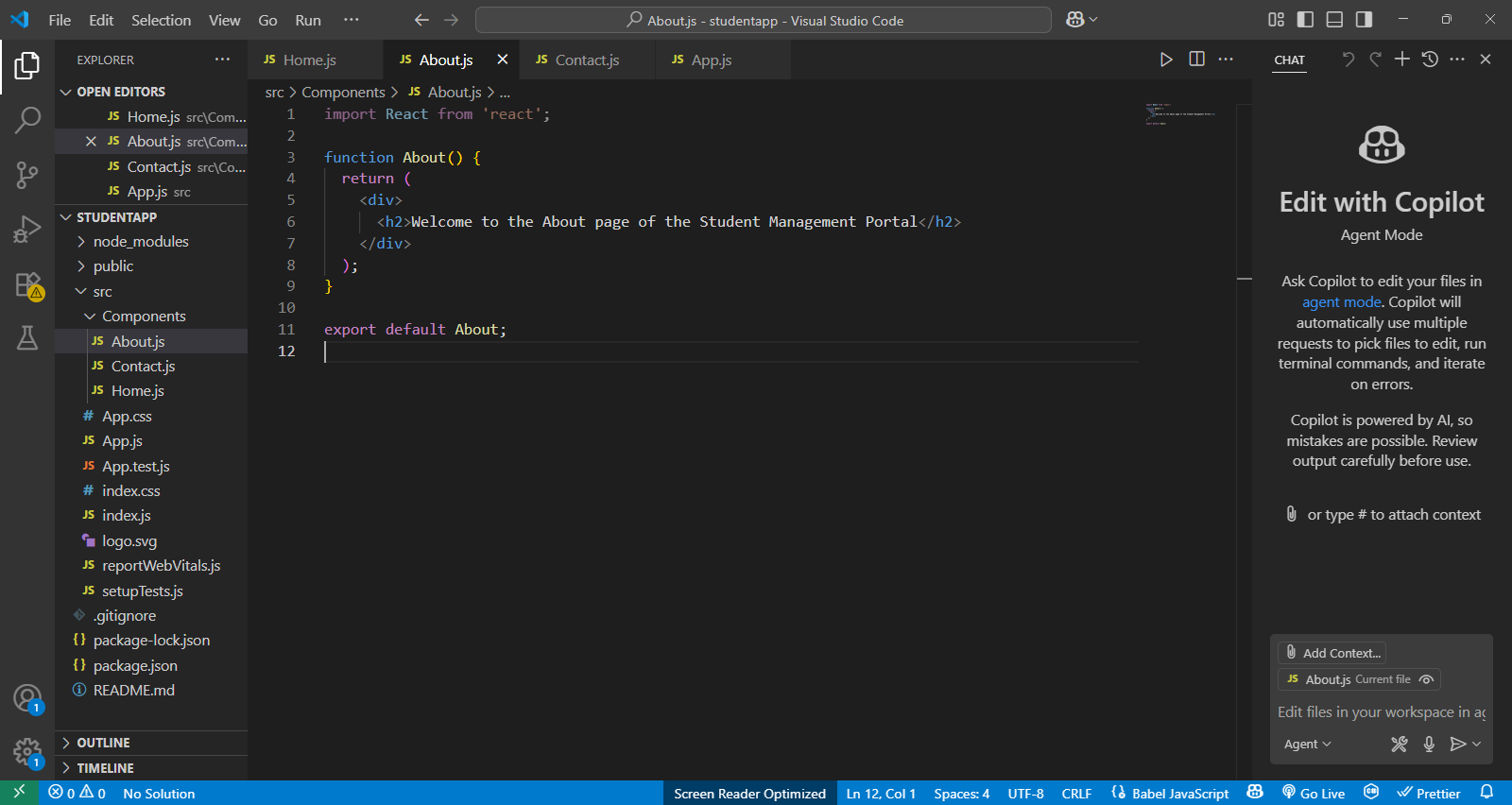


**App.js**

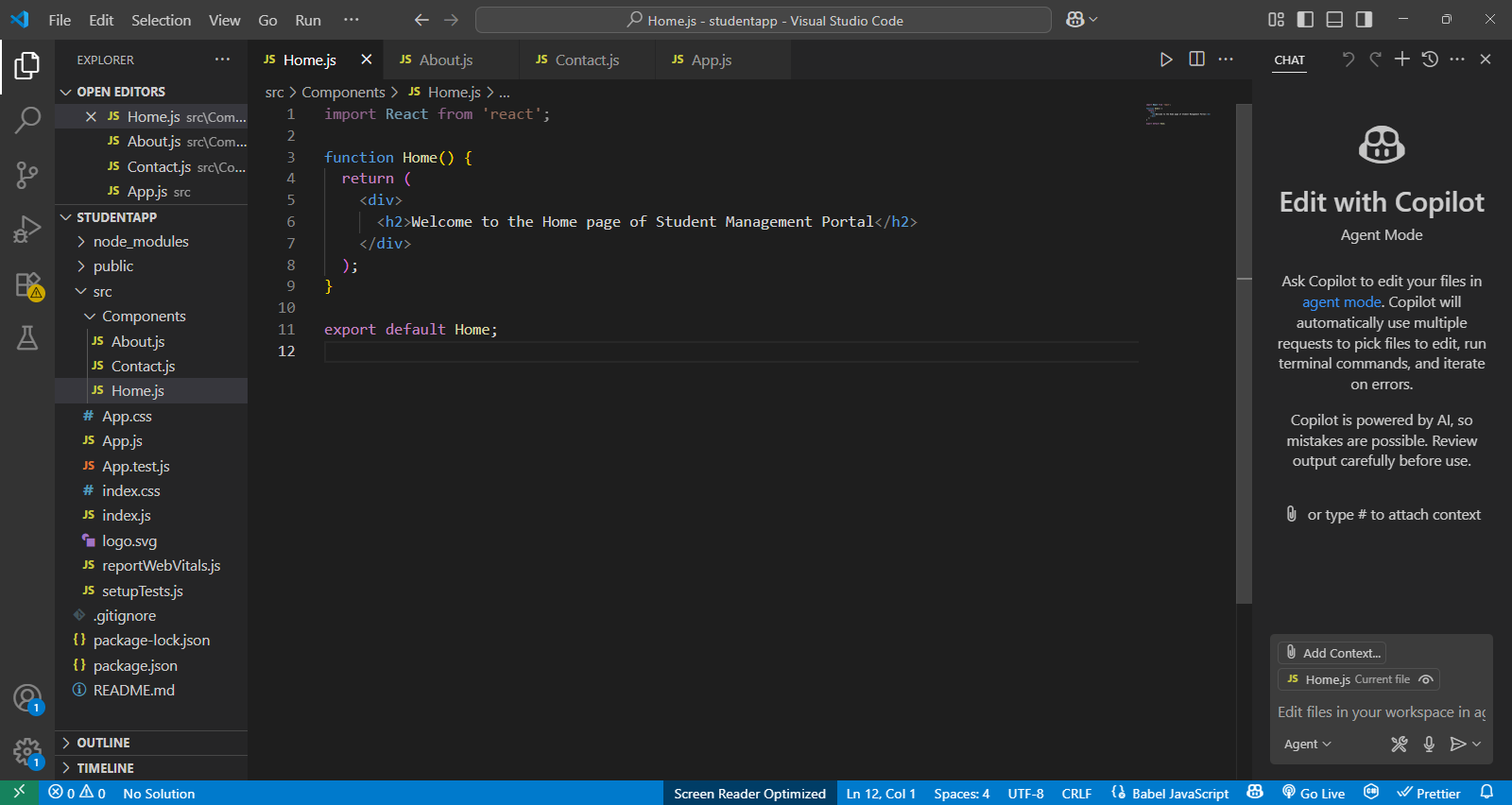


**Contact.js**

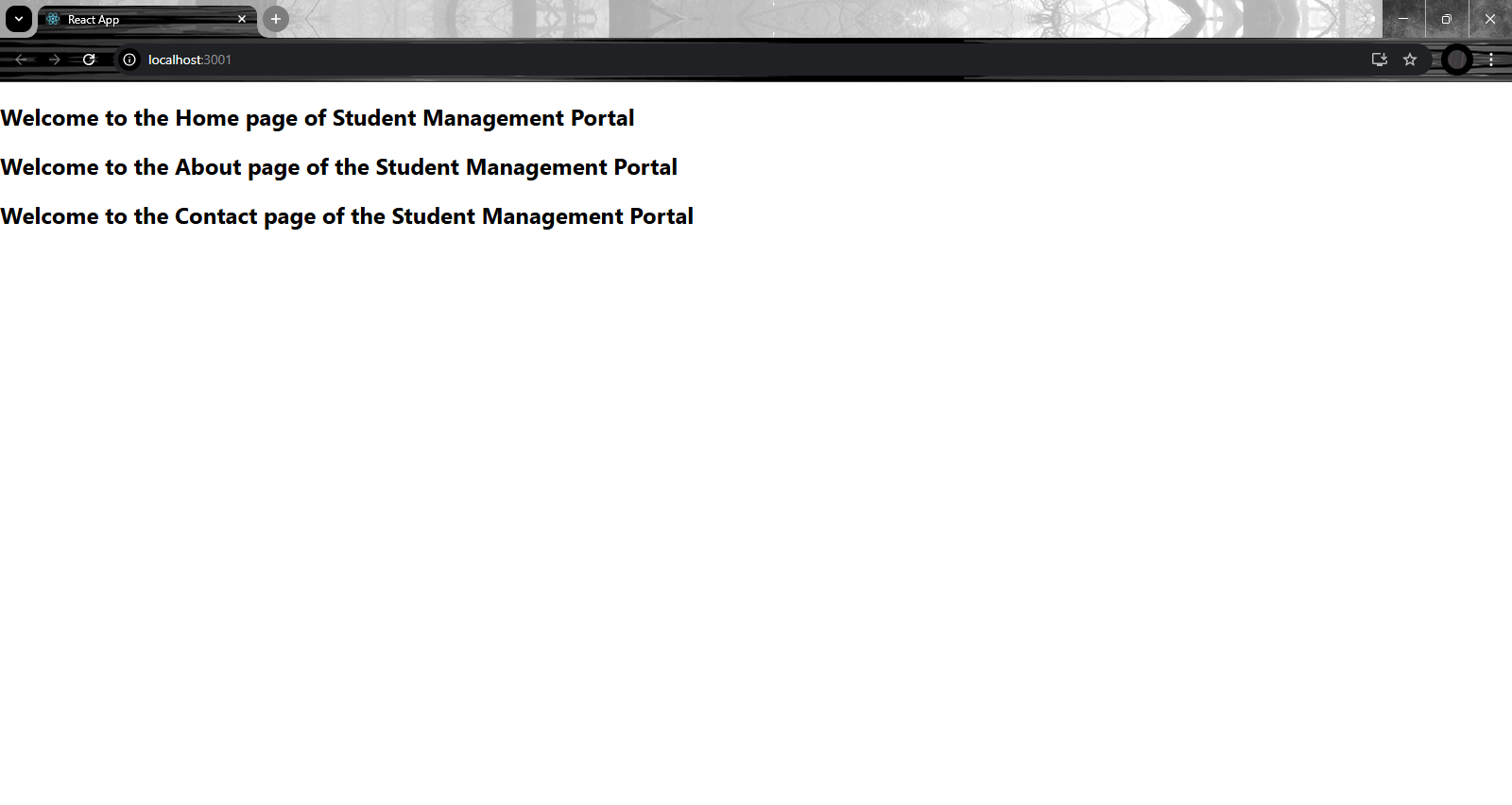
***About.js***



***Home.js***



***Output:***



**Conclusion**

You created a React app with multiple components (Home, About, and Contact). This helped you understand how to build and organize components in React for better structure and reusability.

3. ReactJS-HOL

**Creating a Functional Component to Calculate Average Student Score**

**Objective:**

* To create a **ReactJS functional component**.
* To apply styles using external CSS.
* To calculate and display the **average score** of a student using props.
* To understand the structure and usage of **function components** in React.

**Code**

***CalculateScore.js***

import React from 'react';

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

function CalculateScore({ name, school, total, subjectCount }) {

const average = total / subjectCount;

return (

<div className="score-container">

<h2>Student Name: {name}</h2>

<h3>School: {school}</h3>

<p>Total Marks: {total}</p>

<p>Number of Subjects: {subjectCount}</p>

<p><strong>Average Score: {average.toFixed(2)}</strong></p>

</div>

);

}export default CalculateScore;

***mystyle.css***

.score-container {

border: 2px solid #4CAF50;

padding: 20px;

margin: 20px auto;

background-color: #f1f1f1;

max-width: 500px;

border-radius: 10px;

font-family: Arial, sans-serif;

}

.score-container h2,

.score-container h3 {

color: #333;

margin: 10px 0;

}

App.js

import React from 'react';

import './App.css';

import CalculateScore from './Components/CalculateScore';

function App() {

return (

<div className="App">

<h1>🎓 Student Score Calculator</h1>

<CalculateScore

name="Mahalakshmi"

school="Sri Krishna College of Technology"

total={450}

subjectCount={5}

/>

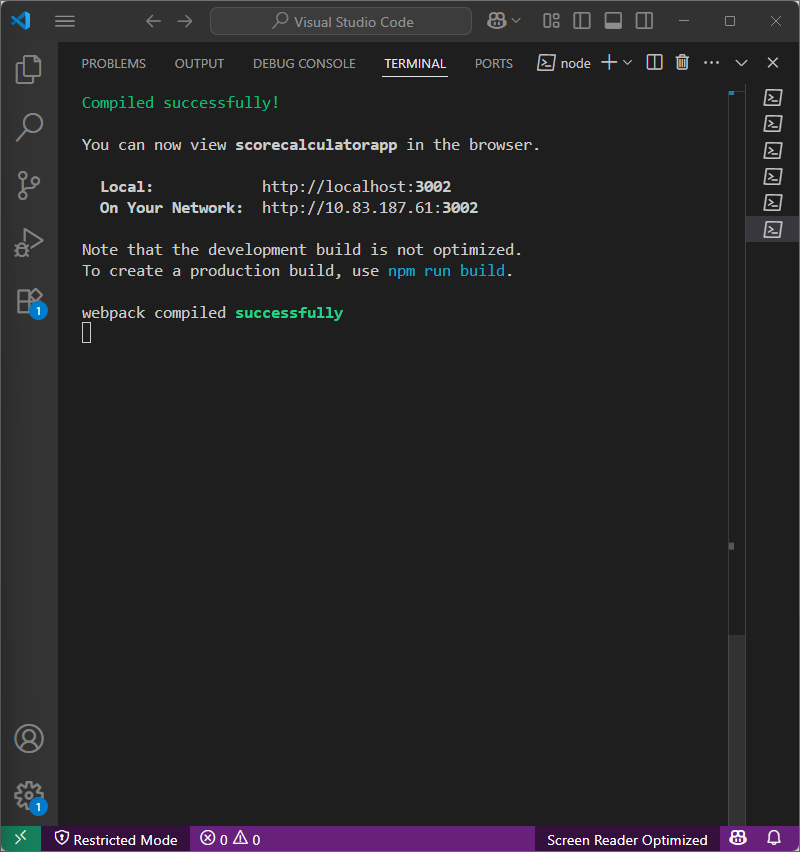
</div>

);

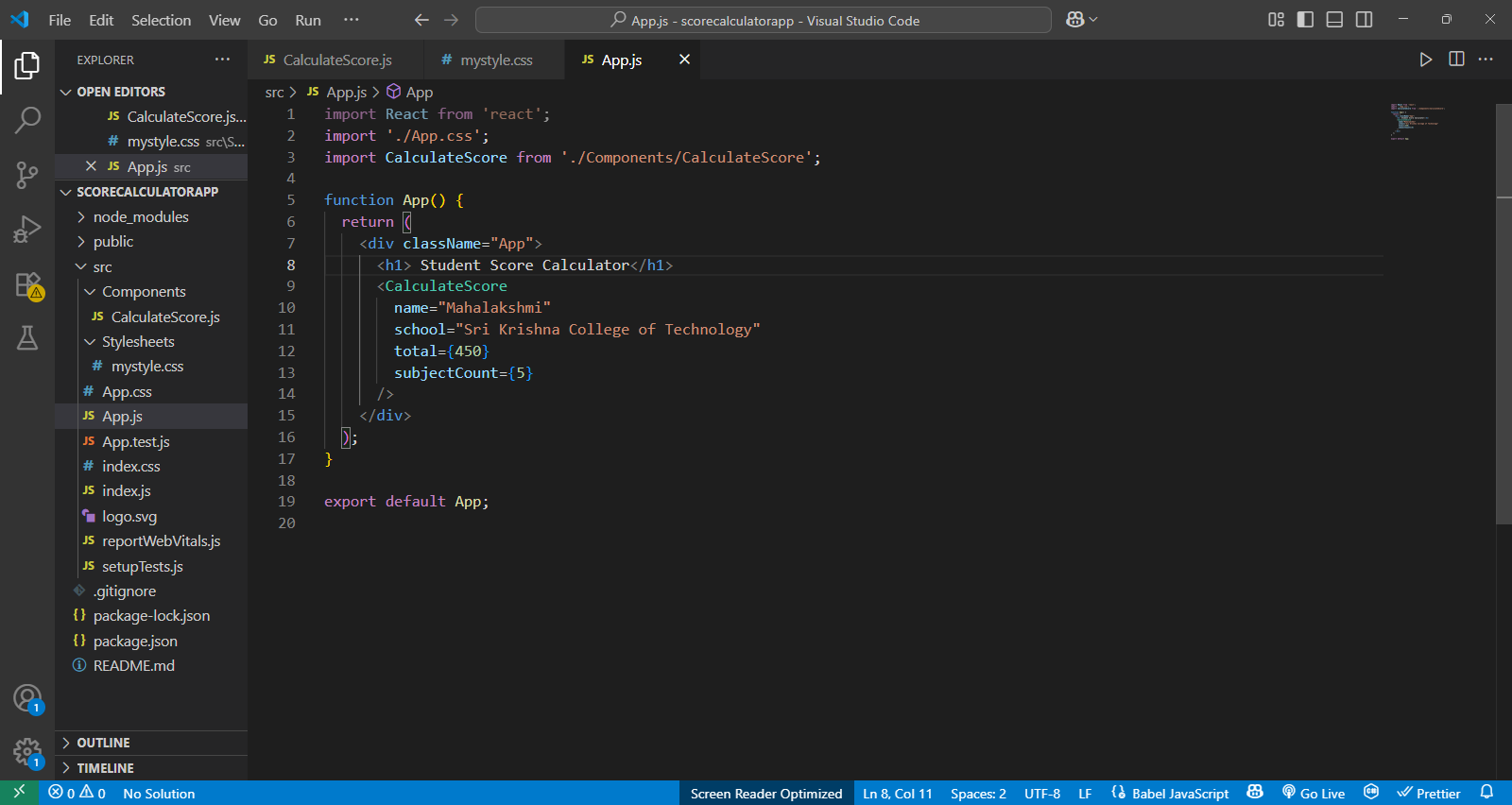
}

export default App;

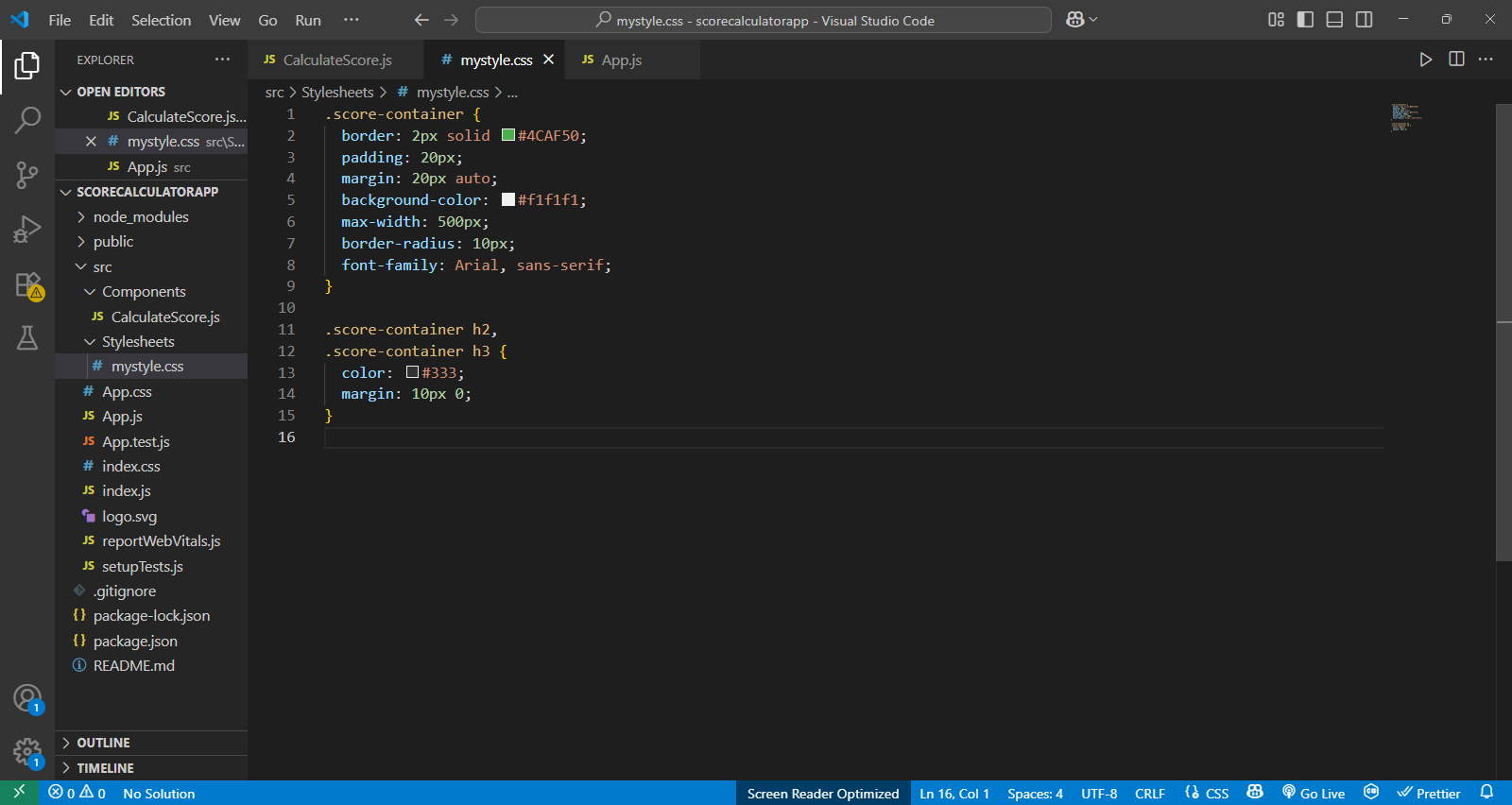
**Screenshot**



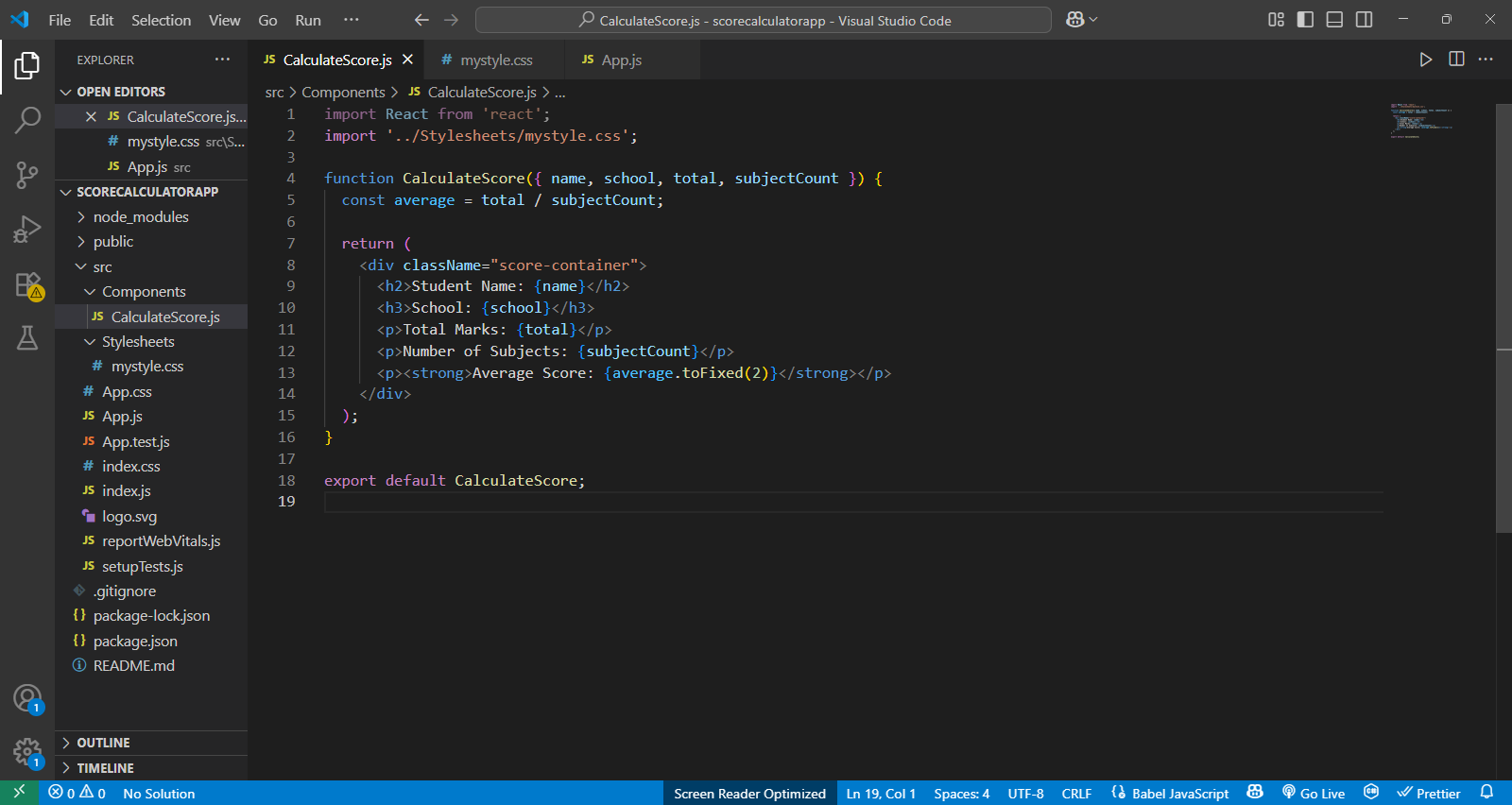
**App.js**



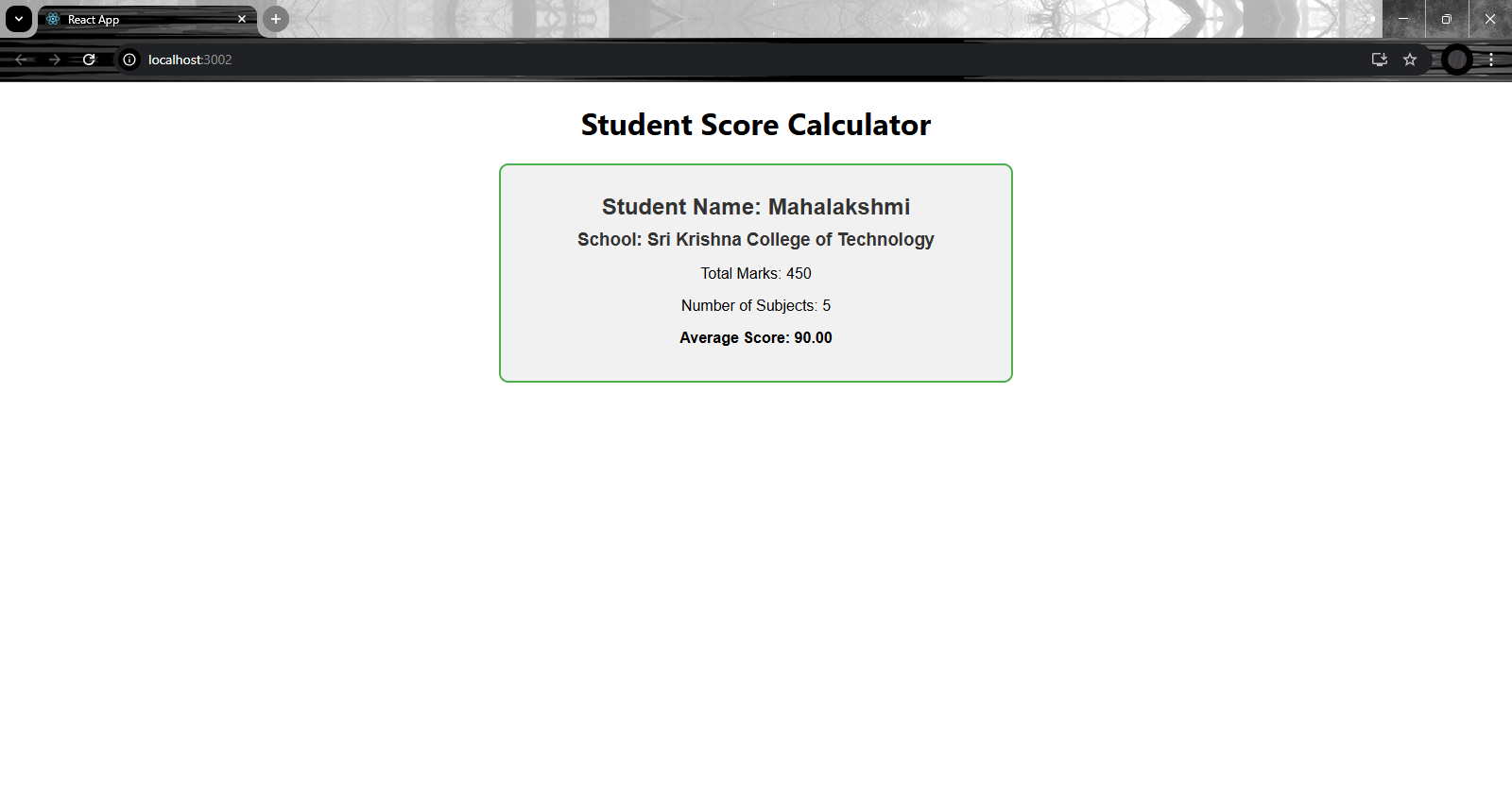
**mystyle.css**



***CalculateScore.js***



***Output :***



**Conclusion:**

We created a React functional component that accepts student details as props, calculates the average score, applies custom CSS, and displays the output neatly. This helped us understand the basics of function components, props, and styling in React.

**4. ReactJS-HOL**

**blogapp**

**Creating a Class Component with Lifecycle Methods to Display Blog Posts**

**Goal:**

Build a blog-style app using **class components** and explore React lifecycle methods

**Objective:**

* Understand and implement **React class components**
* Use **React lifecycle methods**: componentDidMount() and componentDidCatch()
* Fetch data from an external API and display it
* Handle and display potential component errors gracefully

**Code**

**Post.js**

export class Post {

constructor(userId, id, title, body) {

this.userId = userId;

this.id = id;

this.title = title;

this.body = body;

}

}

**Posts.js**

import React, { Component } from 'react';

class Posts extends Component {

constructor(props) {

super(props);

this.state = {

posts: [],

error: null

};}

loadPosts = () => {

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

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

.then((data) => {

this.setState({ posts: data });

})

.catch((error) => {

this.setState({ error: error.message });

});

}

componentDidMount() {

this.loadPosts();

}

componentDidCatch(error, info) {

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

}

render() {

const { posts, error } = this.state;

if (error) {

return <h2>Error: {error}</h2>;

}

return (

<div>

<h1>Blog Posts</h1>

{posts.slice(0, 10).map((post) => (

<div key={post.id} style={{ border: '1px solid gray', padding: '10px', margin: '10px' }}>

<h3>{post.title}</h3>

<p>{post.body}</p>

</div>

))}

</div>

);

}

}

export default Posts;

**App.js**

import React from 'react';

import './App.css';

import Posts from './Posts';

function App() {

return (

<div className="App">

<Posts />

</div>

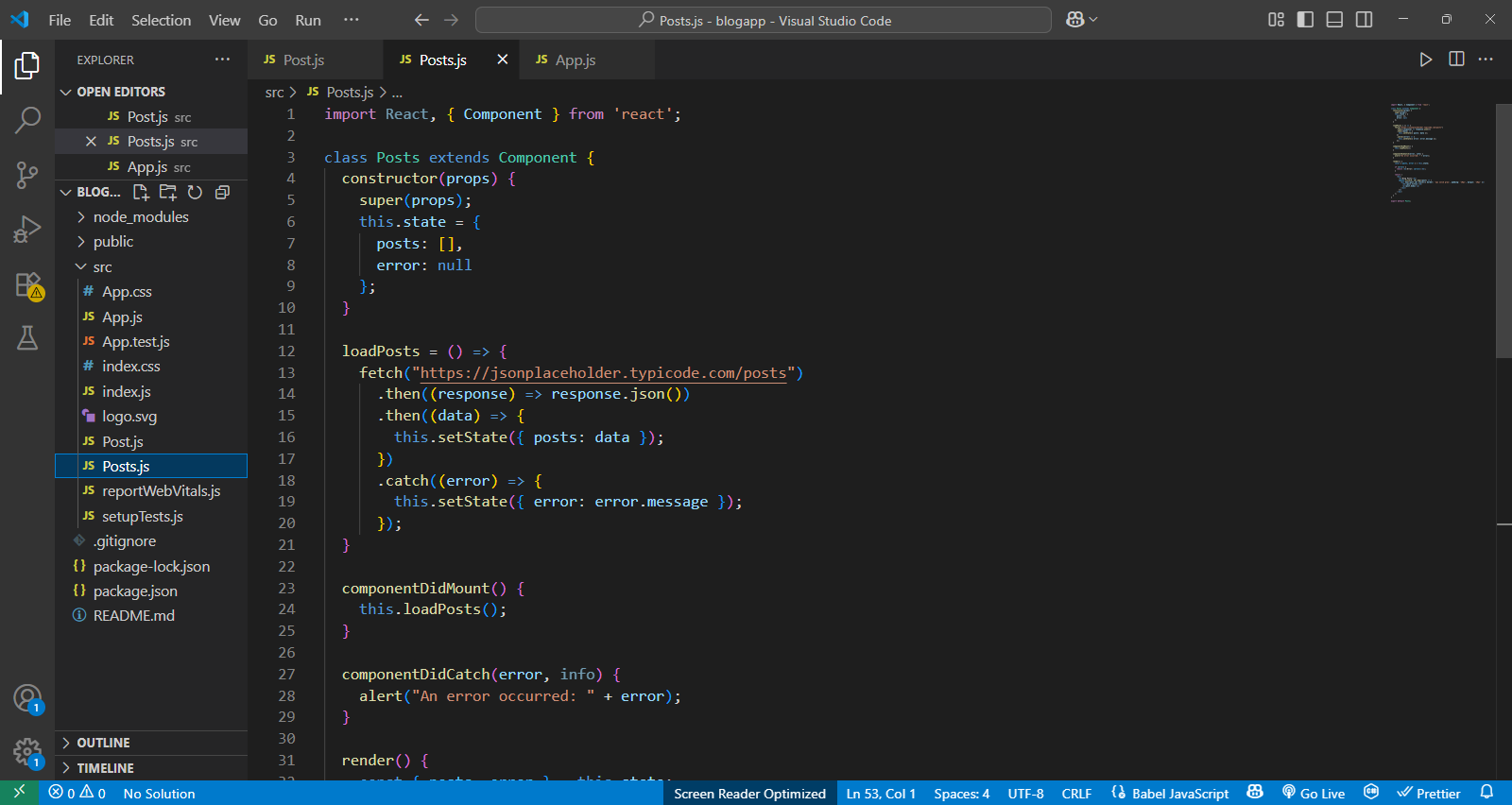
);

}

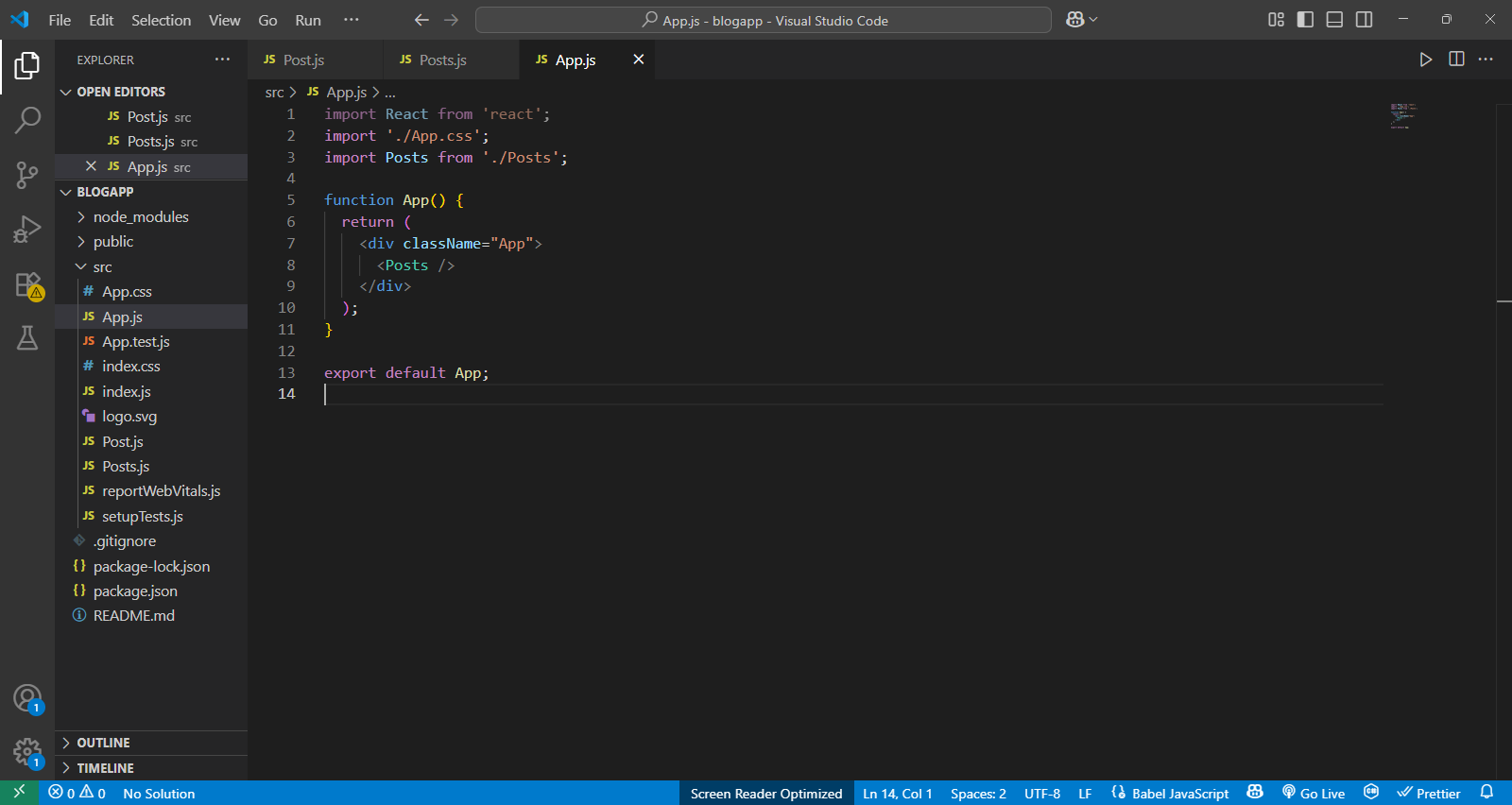
export default App;

**Screenshot**

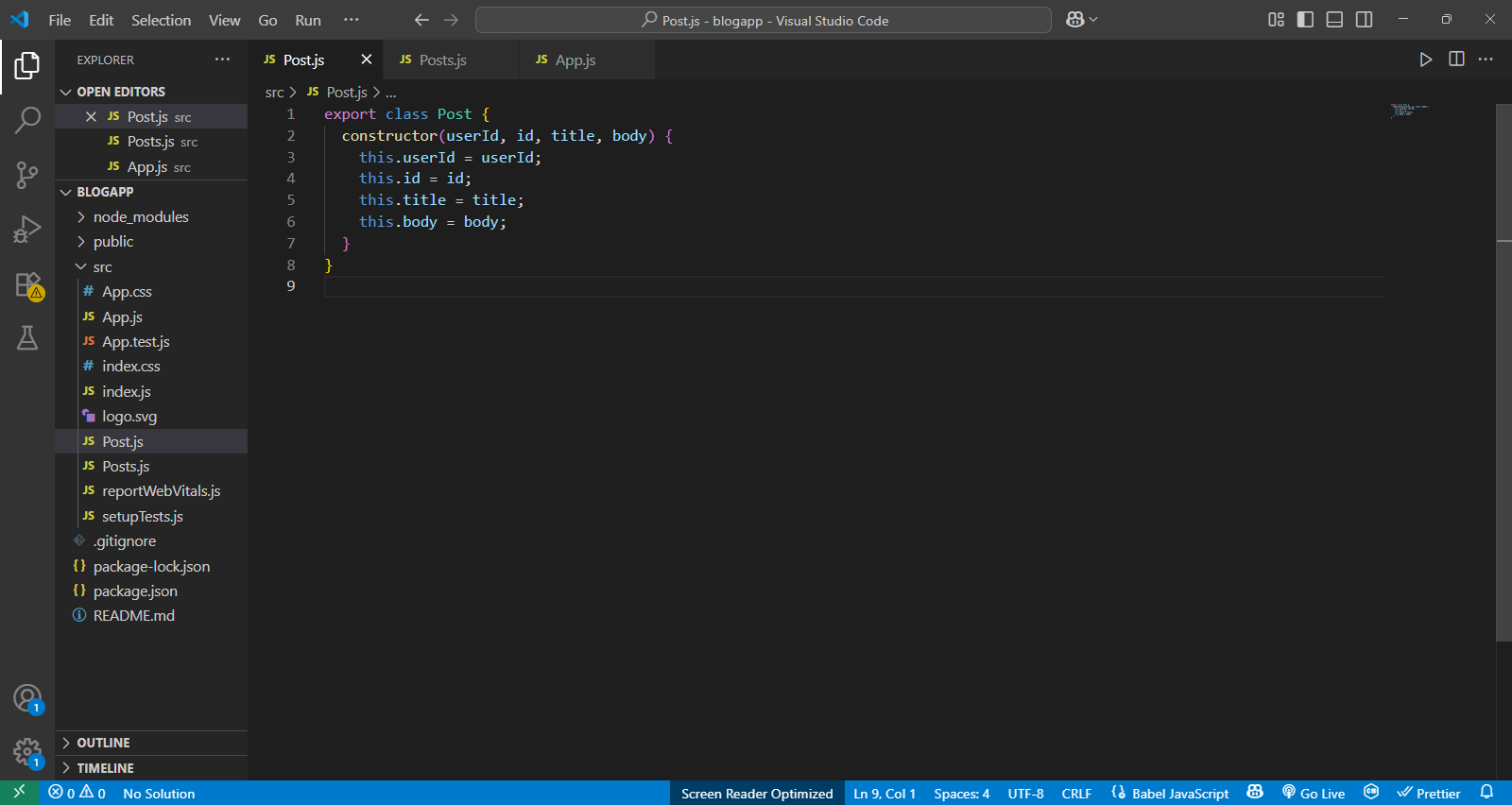
***Posts.js***



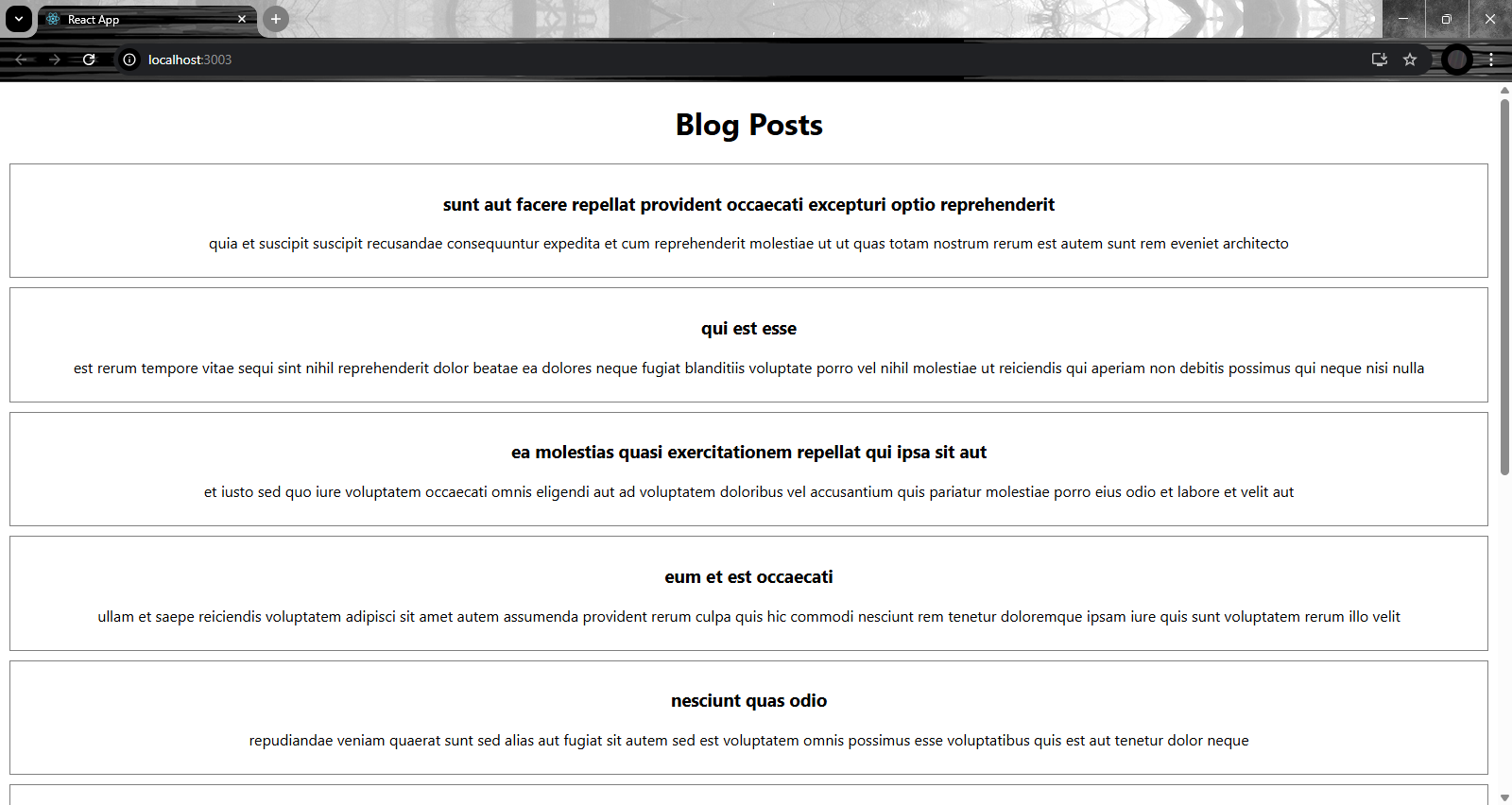
***App.js***



***Post.js***



**Output:**



**Conclusion**

We built a React **class component** that fetches API data in componentDidMount() and displays it, while componentDidCatch() handles errors. This exercise clarified how lifecycle methods enable dynamic data loading and robust error handling in React.

**5. ReactJS-HOL**

**Cohort-dashboard**

**Objective**

To learn how to style React components using CSS Modules and inline styles, and apply conditional styling based on component data.

**Code**

**CohortDetails.js**

// src/components/CohortDetails.js

import React from 'react';

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

const CohortDetails = ({ cohort }) => {

  const isOngoing = cohort.status.toLowerCase() === 'ongoing';

  const titleStyle = {

    color: isOngoing ? 'green' : 'blue',

  };

  return (

    <div className={styles.box}>

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

      <dl>

        <dt>Started On</dt>

        <dd>{cohort.startDate}</dd>

        <dt>Current Status</dt>

        <dd>{cohort.status}</dd>

        <dt>Coach</dt>

        <dd>{cohort.coach}</dd>

        <dt>Trainer</dt>

        <dd>{cohort.trainer}</dd>

      </dl>

    </div>

  );

};

export default CohortDetails;

**CohortDetails.module.css**

/\* src/components/CohortDetails.module.css \*/

.box {

  width: 300px;

  display: inline-block;

  margin: 10px;

  padding: 10px 20px;

  border: 1px solid black;

  border-radius: 10px;

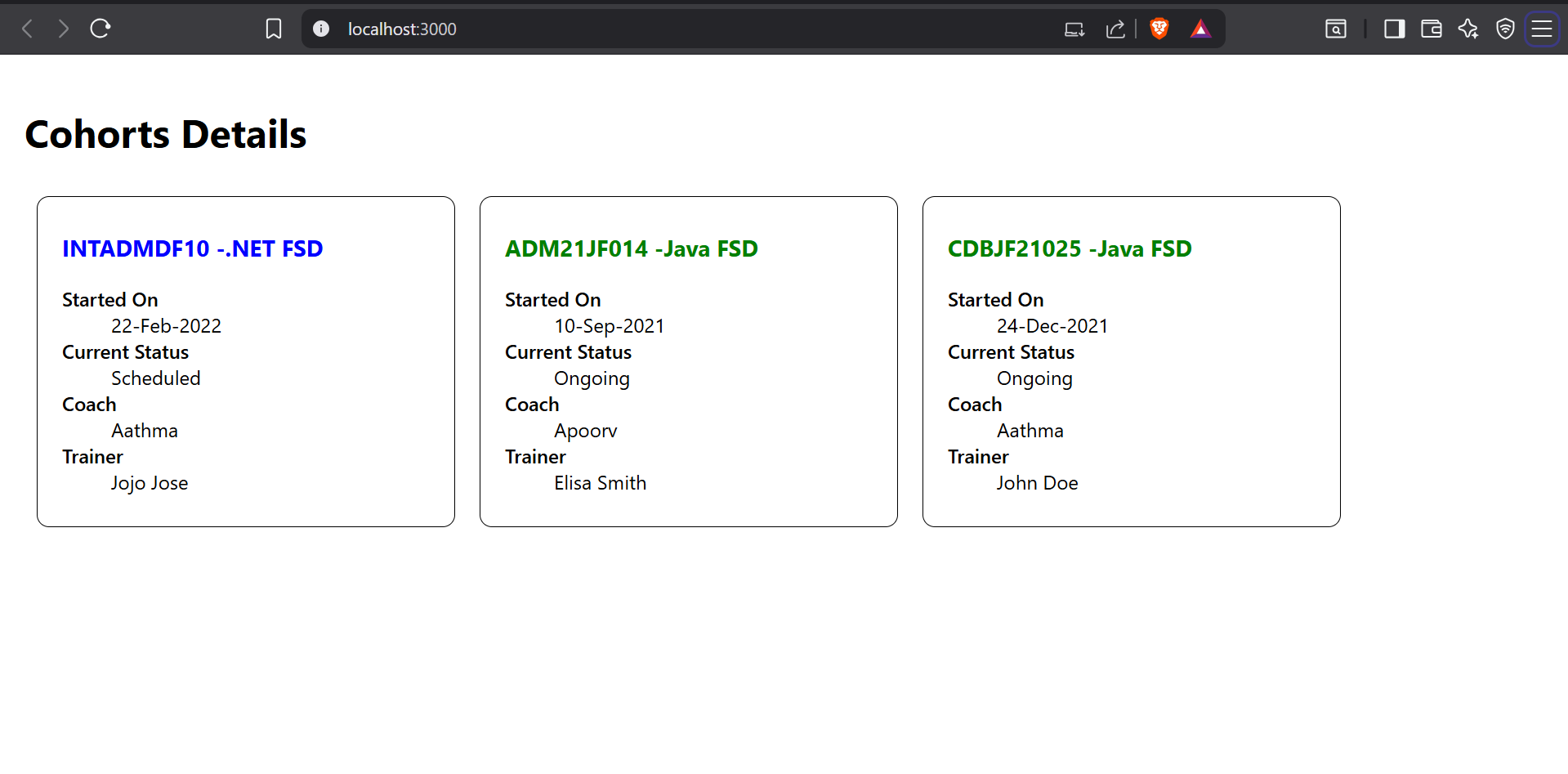
}

dt {

  font-weight: 500;

}

Output Screenshot:



**Conclusion:**

The lab helped in understanding basic React styling methods and how to apply them to improve the look and feel of a React app.