**Solution 1:**

* **Define SPA and its benefits**

A SPA is a web application that loads a single HTML page and updates the content dynamically without reloading the entire page.

**Benefits:** Faster navigation and smoother user experience, Reduced server requests, Works well with API-driven apps.

## **React and How It Works**

React is a JavaScript library for building user interfaces. It lets you create reusable components and updates only the parts of the page that change using something called the **Virtual DOM**. This makes React apps fast and efficient.

## **SPA vs MPA**

**SPA:** Loads one HTML page and changes content dynamically.  
**MPA:** Loads a new HTML page from the server for each request.

## **Pros and Cons of SPA**

**Pros:** Smooth navigation  
 Better user experience  
 Less server load

**Cons:** Can have SEO challenges  
 Larger initial load time  
 Requires more JavaScript on the client side

## **Virtual DOM**

React doesn’t update the real DOM directly. Instead, it uses a **Virtual DOM**, which is a lightweight copy. React compares the Virtual DOM to the real DOM and updates only what has changed, making it very efficient.

## **Features of React**

Component-based architecture  
Uses Virtual DOM for speed  
One-way data binding  
Reusable UI components  
JSX (HTML inside JavaScript)

**Hands-on Lab: Creating React App**

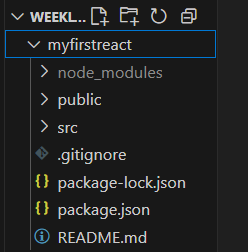
### **Step 1: Create a new React project**

I createD a project named **myfirstreact**, run:

**npx create-react-app myfirstreact**

### **Step 2: Navigate to your project**

cd myfirstreact



[**App.js**](http://app.js)

**import './App.css';**

**function App() {**

**return (**

**<div className="h1"><h1>Hello World</h1></div>**

**);**

**}**

**export default App;**

**App.css**

**.h1{**

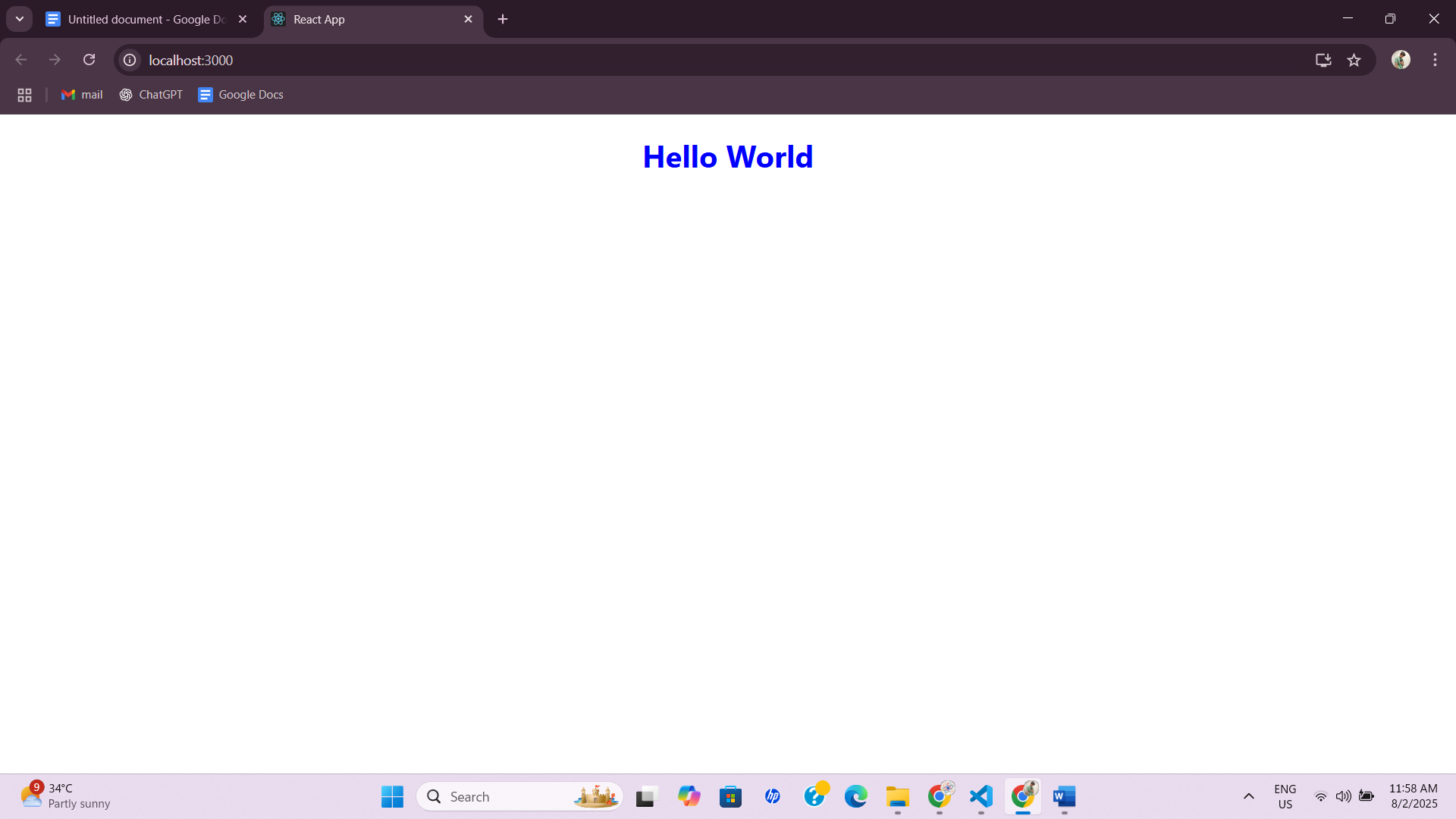
**font-size: 1rem;**

**color: blue;**

**text-align: center;**

**}**

**Result:**

****

**Solution 2:**

* **Explain What React Components are**

Each React component is a self-contained block of code which can be used again to show a fragment of the user interface. In a React application, components enable the user interface to be divided into parts that can be controlled and displayed separately.

Accept inputs (called **props**)

Maintain internal data (**state**)

Return UI elements using JSX

* **Differences Between Components and JavaScript Functions (Paragraph Format)**

Both React components and JavaScript functions have similar syntax and structure, but they serve different purposes. A JavaScript function is used to perform logic or computations and can return any type of value, such as a number, string, or object. In contrast, a React component is designed to render UI elements and usually returns JSX, which specifies how the user interface should look.

React components include features unique to React, such as state, props, and lifecycle methods. These features are important for creating dynamic and interactive user interfaces. JavaScript functions do not include these features. Another notable difference is in naming conventions: JavaScript functions can have any valid name, but React components must start with a capital letter so that React can tell them apart from standard HTML tags. Finally, React components are used in JSX with the `<ComponentName />` syntax, while regular JavaScript functions are not used this way in JSX.

* **Types of Components in React**

React supports two main types of components:

**a. Class Components**

These use ES6 classes. They can hold state and use lifecycle methods.

**b. Function Components**

These use JavaScript functions. With React Hooks like useState and useEffect, they can also manage state and handle lifecycle logic.

### **Explain Class Component**

A Class Component is an ES6 class that extends React.Component. It must have a render() method that returns JSX.

* **Explain Function Component**

A Function Component is a simple JavaScript function that returns JSX. With React Hooks, it can also handle state and side effects.

### **Define Component Constructor**

The class components, the constructor is a special method used to initialize state and bind event handlers.

### **Define render() Function**

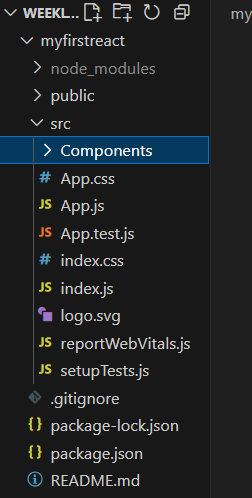
The render() method is required in every class component. It returns the JSX (UI structure) that should be displayed on the screen.

**Handsons:**

### **Create Components Folder**

Inside src/ folder:

* Create a new folder: Components



Components/[Home.js](http://home.js)

import React, { Component } from 'react';

class Home extends Component {

render() {

return (

<div>

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

</div>

);

}

}

export default Home;

Components/[About.js](http://about.js)

import React, { Component } from 'react';

class About extends Component {

render() {

return (

<div>

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

</div>

);

}

}

export default About;

Components/[Contact.js](http://contact.js)

import React, { Component } from 'react';

class Contact extends Component {

render() {

return (

<div>

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

</div>

);

}

}

export default Contact;

src/[App.js](http://app.js)

import './App.css';

import Home from './Components/Home';

import About from './Components/About';

import Contact from './Components/Contact';

function App() {

return (

<div className="App">

<h1>Student Management Portal</h1>

<Home />

<About />

<Contact />

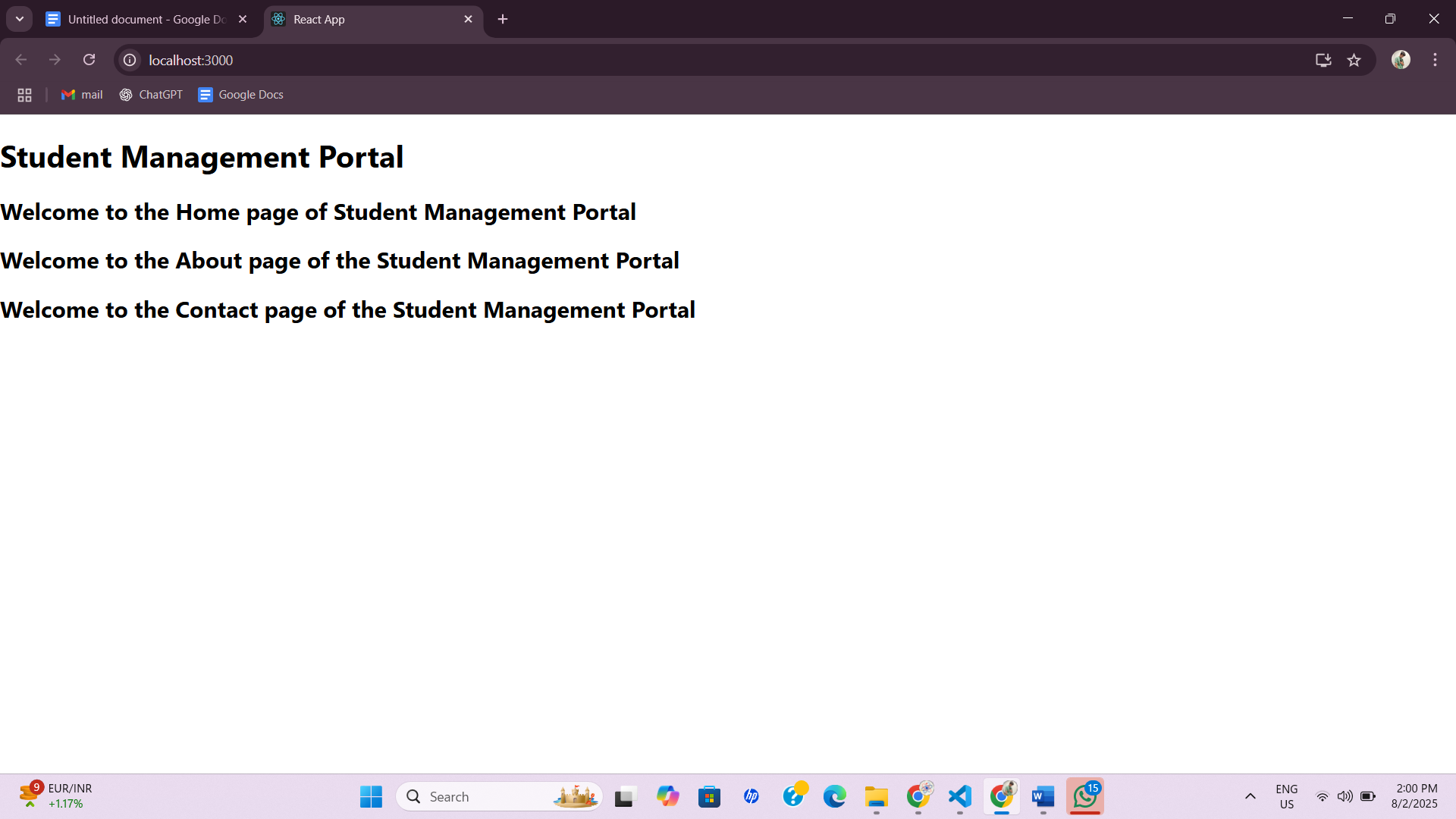
</div>

);

}

export default App;

**Result:**

****

**Solution3:**

I created the React application and i named it as **“scorecalculatorapp”** and i add the 2 folders inside the src folder and i named it as components and stylesheet and the components folder contains the [calclatescore.js](http://calcscore.js) and [scoreutil.js](http://scoreutil.js) and stylesheet having the [mystyles.cs](http://mystyles.cs) and i modified the [App.js](http://app.js) and after completing the code i run the application through the command **npm start.**

[calclatescore.js](http://calcscore.js)

**import React from 'react';**

**import { calcScore } from './ScoreUtils'; // Adjust the path if needed**

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

**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>**

**);**

[scoreutil.js](http://scoreutil.js)

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

**const percentToDecimal = (decimal) => {**

**return (decimal \* 100).toFixed(2) + '%';**

**};**

**const calcScore = (total, goal) => {**

**return percentToDecimal(total / goal);**

**};**

**export { calcScore };**

[mystyles.cs](http://mystyles.cs)

**.Name {**

**font-weight: 300;**

**color: blue;**

**}**

**.School {**

**color: crimson;**

**}**

**.Total {**

**color: darkmagenta;**

**}**

**.formatstyle {**

**text-align: center;**

**font-size: large;**

**}**

**.Score {**

**color: forestgreen;**

**}**

[App.js](http://app.js)

import React from 'react';

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

function App() {

return (

<div className="App">

<CalculateScore Name="Mouli" School="ABC School" total={85} goal={100} />

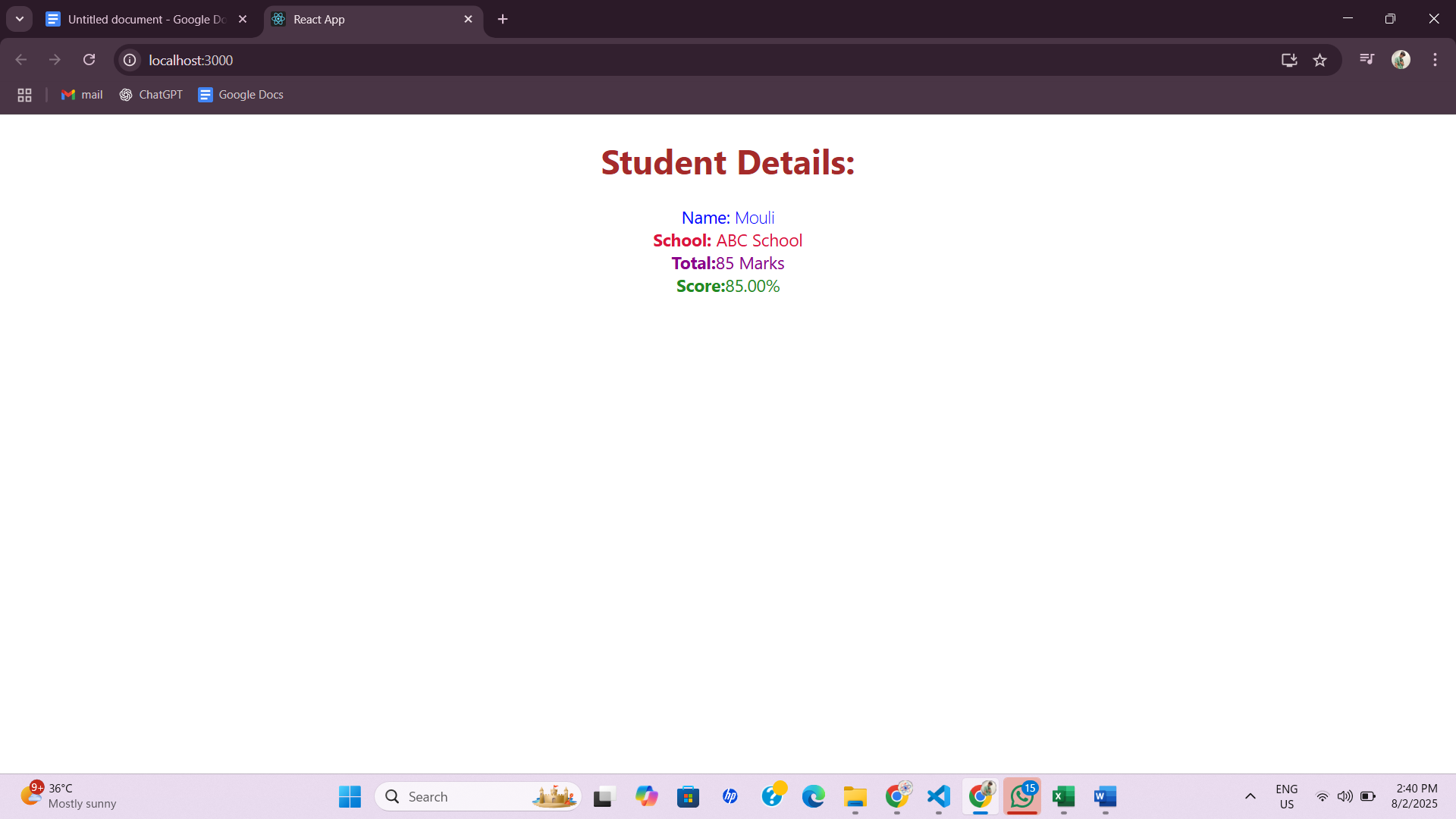
</div>

);

}

export default App;

**Result:**



**Solution 4:**

* Need & Benefits: Component lifecycle allows React to efficiently control setup, update, and cleanup in your app, ensuring components behave predictably and resources are managed well.
* Lifecycle Hook Methods: constructor, render, componentDidMount, componentDidUpdate, componentWillUnmount.
* Rendering Sequence: React creates the component, renders it to the DOM, runs post-render logic, and later updates or cleans up as needed.

In this hands-on lab, you will learn how to:

• Implement componentDidMount() hook

• Implementing componentDidCatch() life cycle hook.

Prerequisites

The following is required to complete this hands-on lab:

• Node.js

• NPM

• Visual Studio Code

// src/Post.js

class Post {

constructor(id, title, body) {

this.id = id;

this.title = title;

this.body = body;

}

}

export default Post;

**// src/Posts.js**

**import React, { Component } from 'react';**

**import Post from './Post';**

**import PostCard from './PostCard';**

**class Posts extends Component {**

**constructor(props) {**

**super(props);**

**this.state = {**

**posts: [],**

**hasError: false,**

**errorMessage: ''**

**};**

**}**

**componentDidMount() {**

**this.loadPosts();**

**}**

**async loadPosts() {**

**try {**

**const res = await fetch("https://jsonplaceholder.typicode.com/posts");**

**const data = await res.json();**

**const posts = data.map(post => new Post(post.id, post.title, post.body));**

**this.setState({ posts });**

**} catch (error) {**

**console.error("Failed to load posts", error);**

**}**

**}**

**componentDidCatch(error, info) {**

**console.error("Error in rendering:", error);**

**this.setState({ hasError: true, errorMessage: error.toString() });**

**}**

**render() {**

**if (this.state.hasError) {**

**return <div>Error occurred: {this.state.errorMessage}</div>;**

**}**

**return (**

**<div>**

**<h1>Posts</h1>**

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

**<PostCard key={post.id} title={post.title} body={post.body} />**

**))}**

**</div>**

**);**

**}**

**}**

**export default Posts;**

// src/PostCard.js

import React from 'react';

function PostCard({ title, body }) {

return (

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

<h2>{title}</h2>

<p>{body}</p>

</div>

);

}

export default PostCard;

// src/App.js

import React from 'react';

import Posts from './Posts';

function App() {

return (

<div>

<Posts />

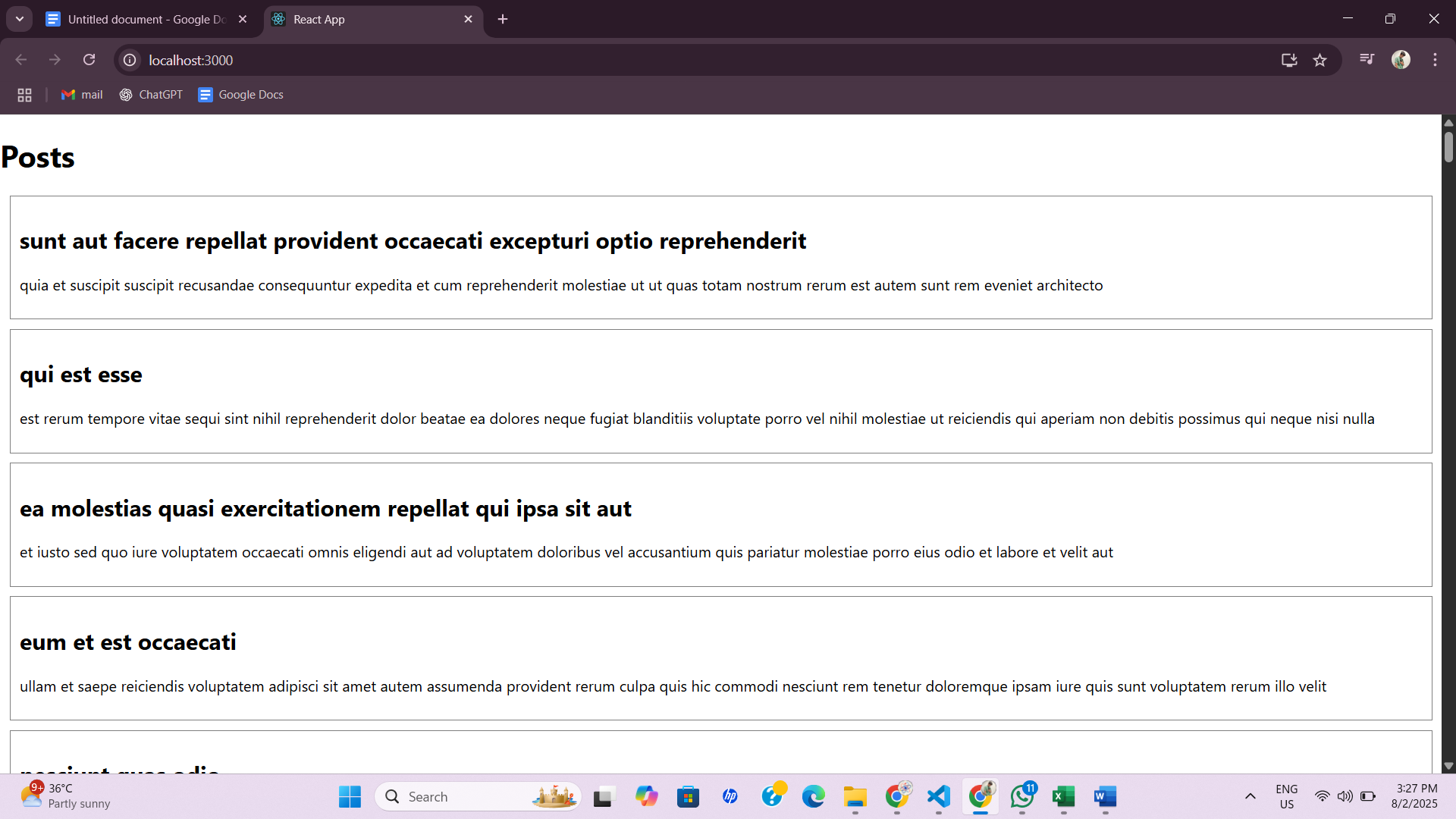
</div>

);

}

export default App;

**RESULT:**



**Solution 5:**

In this hands-on lab, you will learn how to:

• Style a react component

• Define styles using the CSS Module

• Apply styles to components using className and style properties

**Prerequisites**

The following is required to complete this hands-on lab:

• Node.js

• NPM

• Visual Studio Code

[**CohortDetails.js**](http://cohortdetails.js)

import React from 'react';

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

function CohortDetails({ cohort }) {

const headingStyle = {

color: cohort.status.toLowerCase() === 'ongoing' ? 'green' : 'blue'

};

return (

<div className={styles.box}>

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

<dl>

<dt>ID:</dt>

<dd>{cohort.id}</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;

[CohortDetails.module.cs](http://cohortdetails.module.cs)

**.box {**

**width: 300px;**

**display: inline-block;**

**margin: 10px;**

**padding: 10px 20px;**

**border: 1px solid black;**

**border-radius: 10px;**

**}**

**dt {**

**font-weight: 500;**

**}**

[**App.js**](http://app.js)

**import React from 'react';**

**import CohortDetails from './components/CohortDetails';**

**const cohorts = [**

**{**

**id: 'C001',**

**name: 'React Bootcamp',**

**status: 'Ongoing',**

**startDate: '2025-07-01',**

**endDate: '2025-08-01'**

**},**

**{**

**id: 'C002',**

**name: 'Java Fundamentals',**

**status: 'Completed',**

**startDate: '2025-06-01',**

**endDate: '2025-07-01'**

**}**

**];**

**function App() {**

**return (**

**<>**

**<h1 style={{ textAlign: 'center' }}>Cohort Dashboard</h1>**

**<div style={{ padding: '20px' }}>**

**{cohorts.map((cohort) => (**

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

**))}**

**</div>**

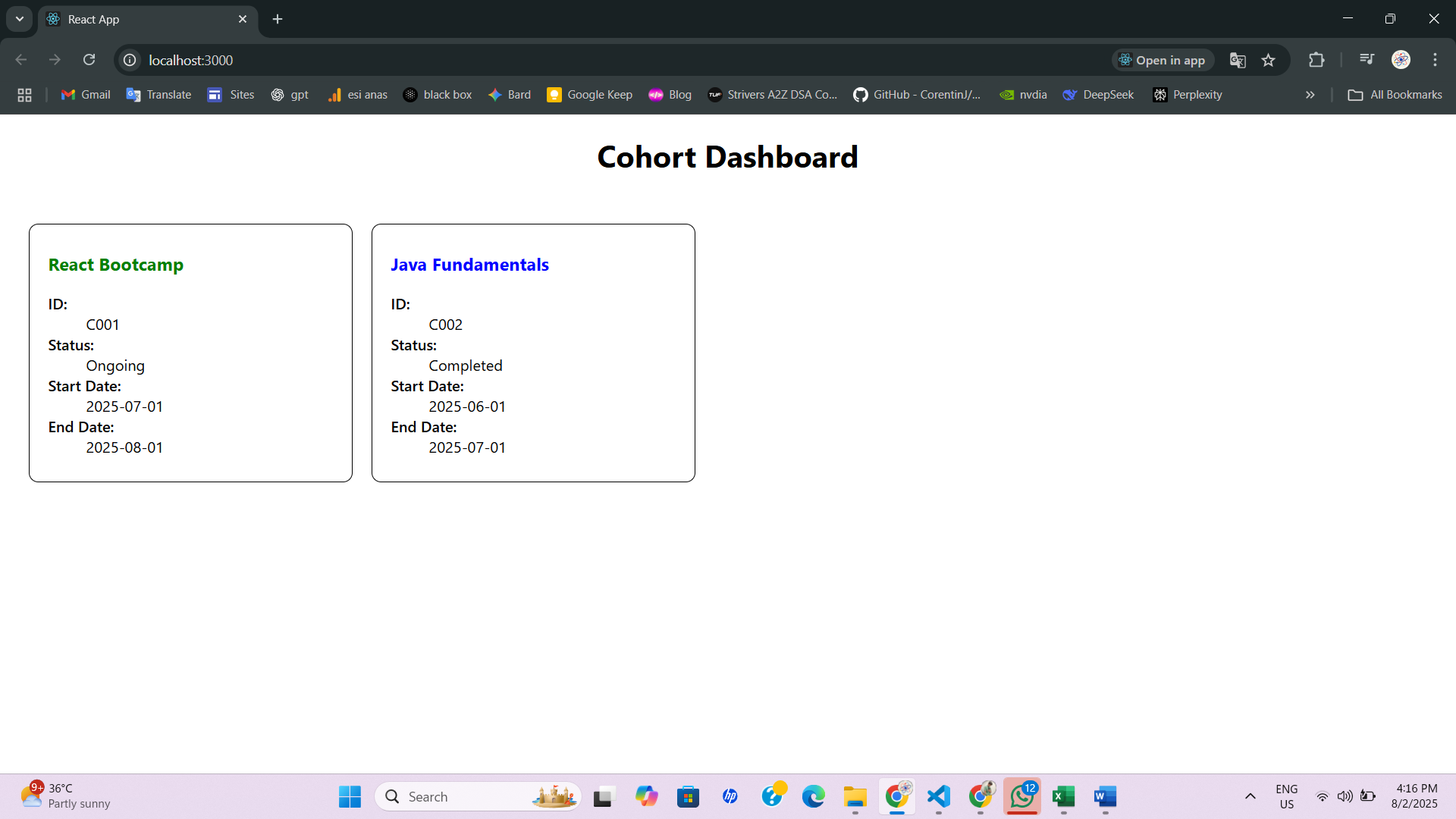
**</>**

**);**

**}**

**export default App;**

**RESULT:**

****