**SOLUTION 9:**

* **List the features of ES6:**  
   ES6 introduced new syntax like let, const, arrow functions, classes, template strings, destructuring, and modules.  
   It also brought features like default/rest/spread parameters, Promises, Map, Set, and enhanced object literals.
* **Explain JavaScript let:**  
   let declares block-scoped variables that can be updated but not redeclared in the same scope.  
   It helps avoid issues caused by the function-scoped behavior of var.
* Identify the differences between var and let:  
   var is function-scoped and can be redeclared, while let is block-scoped and cannot be redeclared within the same block.  
   Variables declared with let are not hoisted in the same way as var, helping reduce errors.
* **Explain JavaScript const:**  
   const is used to declare block-scoped, read-only variables that must be assigned a value at the time of declaration.  
   While the variable itself can't be reassigned, objects and arrays declared with const can still be modified.
* **Explain ES6 class fundamentals:**  
   ES6 classes are blueprints for creating objects with shared properties and methods, using the class keyword.  
   They make object-oriented programming in JavaScript clearer and easier than using prototypes directly.
* **Explain ES6 class inheritance:** Inheritance allows one class to extend another using the extends keyword, sharing and reusing code.  
   The child class can use super() to call the parent class's constructor or methods.
* **Define ES6 arrow functions:**  
   Arrow functions are a concise way to write functions using the => syntax.  
   They do not have their own context, which helps avoid common pitfalls in callbacks.

* **Identify set(), map():** Set stores unique values of any type and removes duplicates automatically.  
   Map stores key-value pairs, allowing keys of any type and preserving their insertion order.

**Lab:**

**I** created the cricket named react app and in the src i add the components folder for implement the components inside the components folder and i updated the [app.js](http://app.js) and then i execute the command **npm start**

**ListofPlayers.jsx**

**import React from 'react';**

**const ListofPlayers = () => {**

**const players = [**

**{ name: 'Virat', score: 85 },**

**{ name: 'Rohit', score: 90 },**

**{ name: 'Dhoni', score: 45 },**

**{ name: 'Pant', score: 35 },**

**{ name: 'Hardik', score: 65 },**

**{ name: 'Shreyas', score: 75 },**

**{ name: 'Bumrah', score: 60 },**

**{ name: 'Ashwin', score: 30 },**

**{ name: 'Jadeja', score: 80 },**

**{ name: 'Gill', score: 95 },**

**{ name: 'Shami', score: 55 }**

**];**

**const allPlayers = players.map((item, idx) => (**

**<li key={idx}>{item.name} - {item.score}</li>**

**));**

**const below70 = players.filter(item => item.score < 70);**

**return (**

**<div>**

**<h2>All Players</h2>**

**<ul>{allPlayers}</ul>**

**<h2>Players with Score &lt; 70</h2>**

**<ul>**

**{below70.map((item, idx) => (**

**<li key={idx}>{item.name} - {item.score}</li>**

**))}**

**</ul>**

**</div>**

**);**

**};**

**export default ListofPlayers;**

**IndianPlayers.jsx**

**import React from 'react';**

**const IndianPlayers = () => {**

**const T20Players = ['Virat', 'Rohit', 'Pant', 'Bumrah', 'Jadeja'];**

**const RanjiPlayers = ['Rahane', 'Pujara', 'Iyer'];**

**const allPlayers = [...T20Players, ...RanjiPlayers];**

**const oddTeam = [];**

**const evenTeam = [];**

**allPlayers.forEach((item, index) => {**

**if (index % 2 === 0) {**

**oddTeam.push(item);**

**} else {**

**evenTeam.push(item);**

**}**

**});**

**return (**

**<div>**

**<h2>Merged Players:</h2>**

**<p>{allPlayers.join(', ')}</p>**

**<h3>Odd Team Players:</h3>**

**<ul>{oddTeam.map((p, i) => <li key={i}>{p}</li>)}</ul>**

**<h3>Even Team Players:</h3>**

**<ul>{evenTeam.map((p, i) => <li key={i}>{p}</li>)}</ul>**

**</div>**

**);**

**};**

**export default IndianPlayers;**

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

**import React from 'react';**

**import ListofPlayers from './Components/ListofPlayers';**

**import IndianPlayers from './Components/IndianPlayers';**

**function App() {**

**const flag = true;**

**return (**

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

**<h1>Welcome to CricketApp</h1>**

**{flag ? <ListofPlayers /> : <IndianPlayers />}**

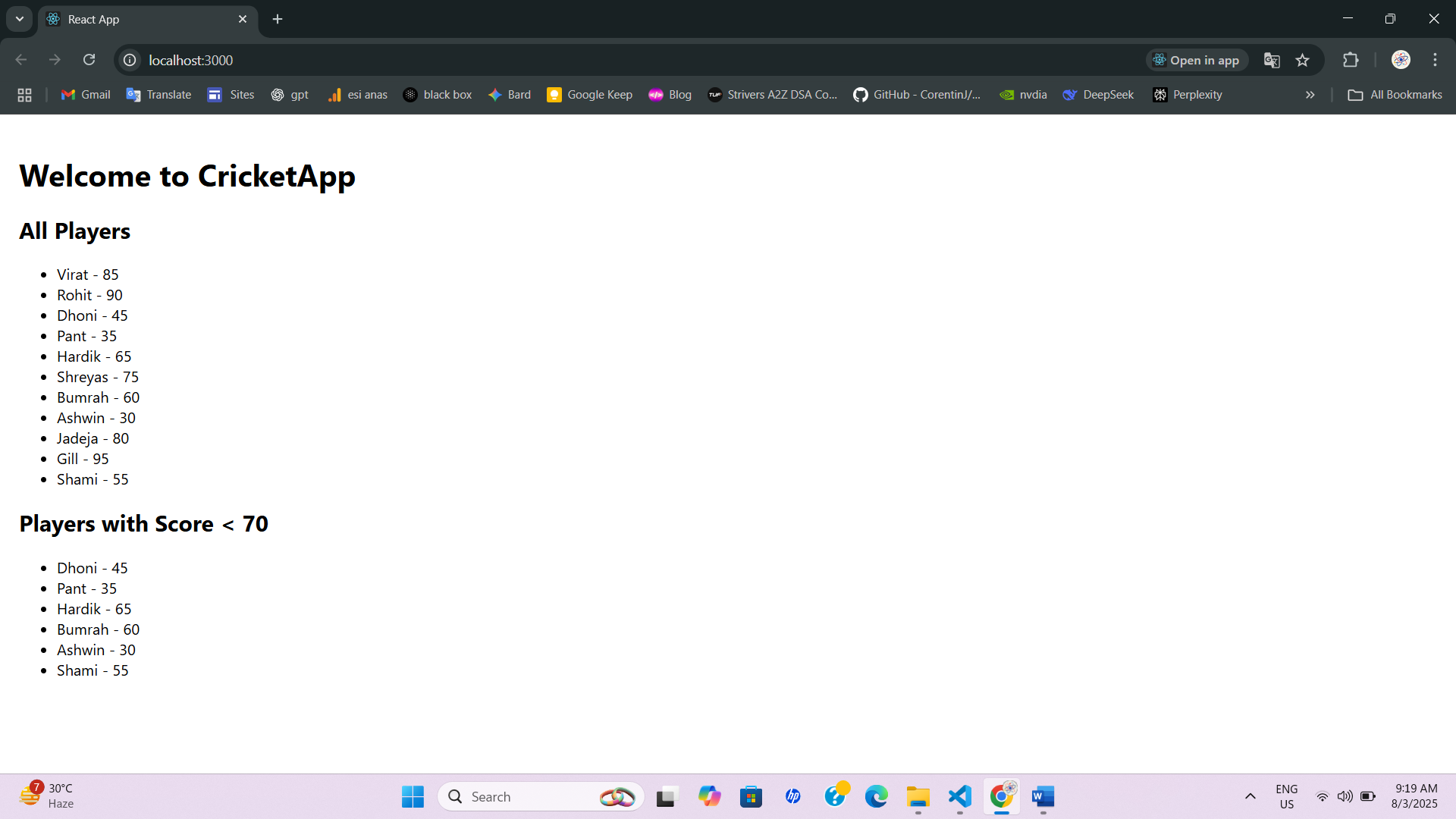
**</div>**

**);**

**}**

**export default App;**

**RESULT:**

****

**Solution 10:**

* Define JSX

JSX (JavaScript XML) lets you write HTML-like syntax inside JavaScript in React.

Eg: **const heading = <h1>Hello React</h1>;**

* Explain about ECMA Script

ECMAScript is the standard specification of JavaScript (React uses ES6 features like let, const, arrow functions).

Eg: **const add = (a,b) => a+b;**

* Explain React.createElement()

React.createElement() creates React elements without using JSX; JSX is compiled into this function.

Eg: **const el = React.createElement('h1', null, 'Hello');**

* Explain how to create React nodes with JSX

React nodes are UI elements created using JSX tags which represent components or DOM elements.

Eg: **const node = <p>This is a React node</p>;**

* Define how to render JSX to DO

JSX elements are rendered to the DOM using ReactDOM.render() in React 17 or createRoot().render() in React 18+.

Eg: **ReactDOM.render(<App />, document.getElementById('root'));**

* Explain how to use JavaScript expressions in JSX

JSX allows embedding JS expressions inside {} to dynamically display values.

Eg: **const name = "John"; <h1>Hello {name}</h1>;**

* Explain how to use inline CSS in JSX

Inline CSS in JSX is written as a JS object using style={{}} syntax with camelCase properties.

Eg: **<h1 style={{ color: "blue", fontSize: "20px" }}>Hello</h1>;**

**Lab:**

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

**import React from "react";**

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

**import officeImage from "./office.jpg";**

**function App() {**

**const element = "Office Space";**

**const jsxatt = <img src={officeImage} width="10%" height="15%" alt="Office Space" />;**

**const ItemName = { Name: "DBS", Rent: 50000, Address: "Chennai" };**

**const colors = ItemName.Rent <= 60000 ? "textRed" : "textGreen";**

**return (**

**<div className="App">**

**<h1>{element}, at Affordable Range</h1>**

**{jsxatt}**

**<h1>Name: {ItemName.Name}</h1>**

**<h3 className={colors}>Rent: Rs. {ItemName.Rent}</h3>**

**<h3>Address: {ItemName.Address}</h3>**

**</div>**

**);**

**}**

**export default App;**

**App.css**

**.App {**

**text-align: center;**

**}**

**.textRed {**

**color: red;**

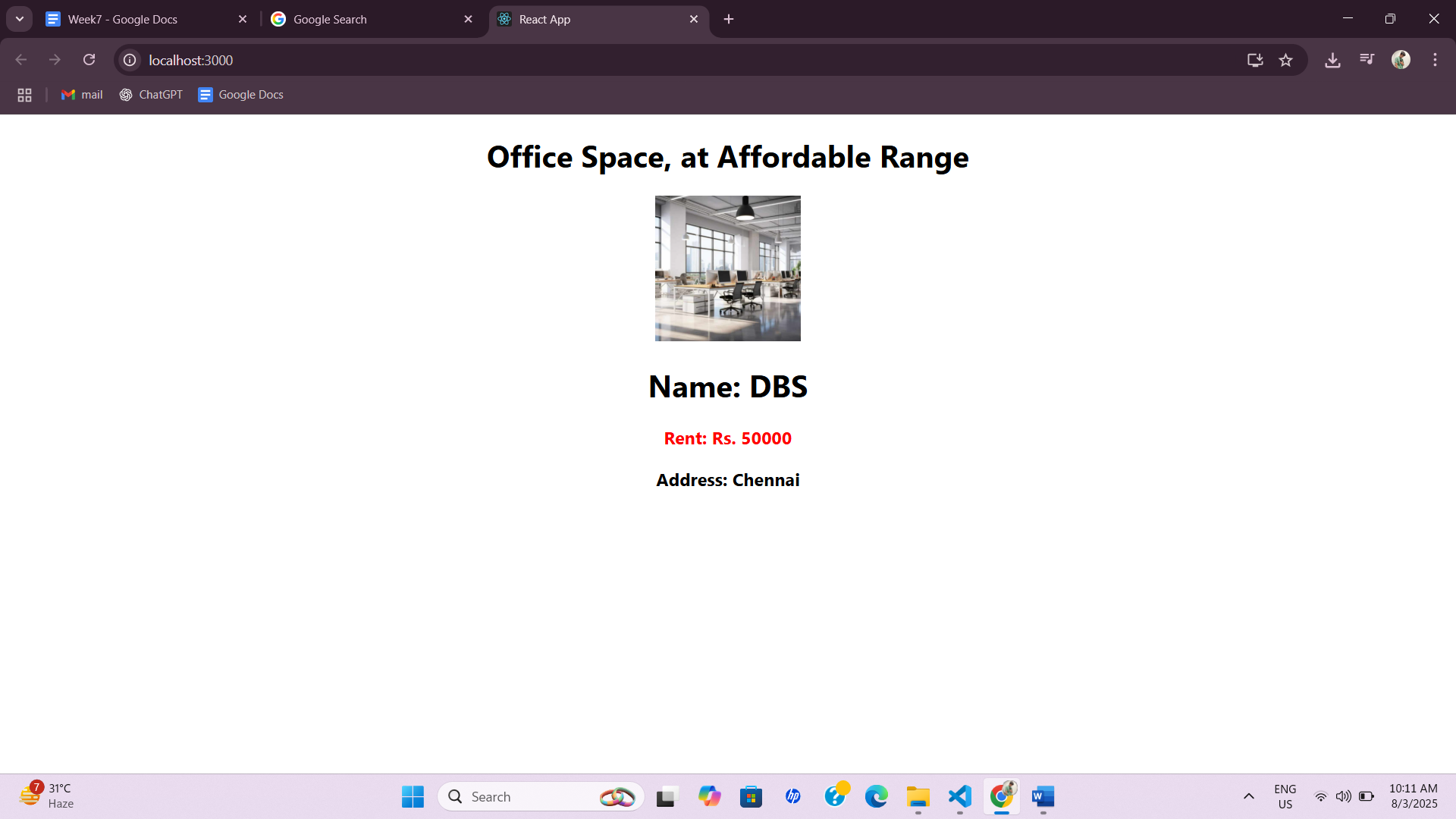
**}**

**.textGreen {**

**color: green;**

**}**

**Result:**

****

**Solution 11:**

* **Explain React events**

React events are functions triggered by user interactions like clicks, input changes, or form submissions. They are similar to DOM events but work through React's Synthetic Event system.

* **Explain about event handlers**

Event handlers are functions in React that execute when a specific event occurs. They are passed as props to elements using attributes like onClick or onChange.

* **Define Synthetic event**

A Synthetic Event is React’s wrapper around native browser events. It provides consistent behavior across all browsers and uses a pooled event system for performance.

* **Identify React event naming convention**

React uses camelCase for event names instead of lowercase HTML events. Example: onClick, onSubmit instead of onclick, onsubmit.

**Lab:**

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

**import React, { useState } from "react";**

**import CurrencyConvertor from "./CurrencyConvertor";**

**function App() {**

**const [count, setCount] = useState(0);**

**const increment = () => {**

**setCount(count + 1);**

**};**

**const decrement = () => {**

**setCount(count - 1);**

**};**

**const sayHello = () => {**

**alert("Hello! Static Message");**

**};**

**const handleIncrease = () => {**

**increment();**

**sayHello();**

**};**

**const sayWelcome = (msg) => {**

**alert(msg);**

**};**

**const handlePress = (e) => {**

**e.preventDefault();**

**alert("I was clicked");**

**};**

**return (**

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

**<h2>{count}</h2>**

**<button onClick={handleIncrease}>Increment</button>**

**<button onClick={decrement}>Decrement</button>**

**<br /><br />**

**<button onClick={() => sayWelcome("Welcome")}>Say welcome</button>**

**<br /><br />**

**<button onClick={handlePress}>Click on me</button>**

**<CurrencyConvertor />**

**</div>**

**);**

**}**

**export default App;**

[**CurrencyConverter.js**](http://currencyconverter.js)

**import React, { useState } from "react";**

**function CurrencyConvertor() {**

**const [rupees, setRupees] = useState("");**

**const handleSubmit = (e) => {**

**e.preventDefault();**

**const conversionRate = 80;**

**const euroAmount = rupees \* conversionRate;**

**alert(`Converting to Euro Amount is ${euroAmount}`);**

**};**

**return (**

**<div style={{ marginTop: "20px" }}>**

**<h1 style={{ color: "green" }}>Currency Convertor!!!</h1>**

**<form onSubmit={handleSubmit}>**

**<div>**

**<label>Amount: </label>**

**<input**

**type="number"**

**value={rupees}**

**onChange={(e) => setRupees(e.target.value)}**

**/>**

**</div>**

**<div>**

**<label>Currency: </label>**

**<select>**

**<option>Euro</option>**

**</select>**

**</div>**

**<button type="submit">Submit</button>**

**</form>**

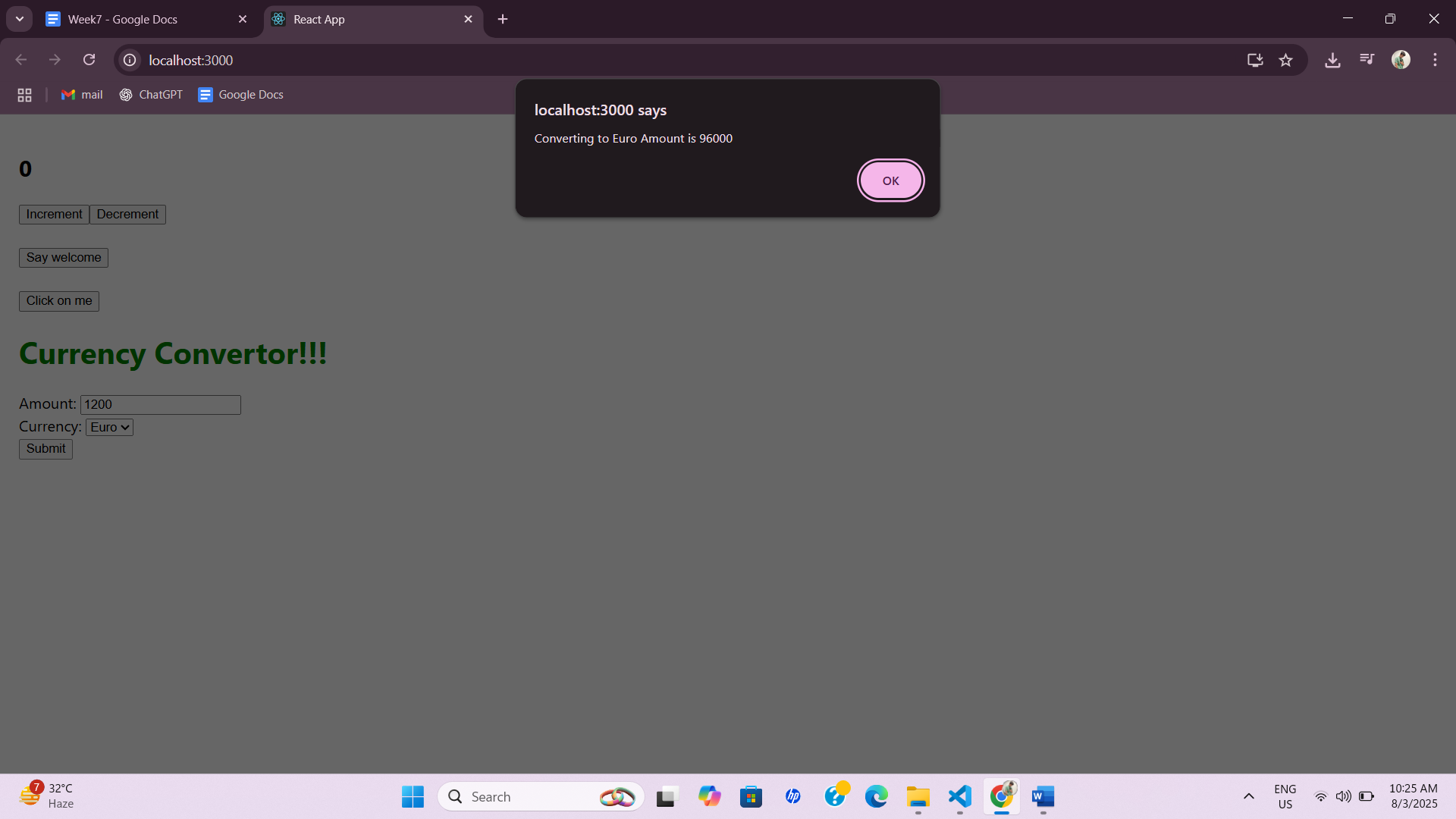
**</div>**

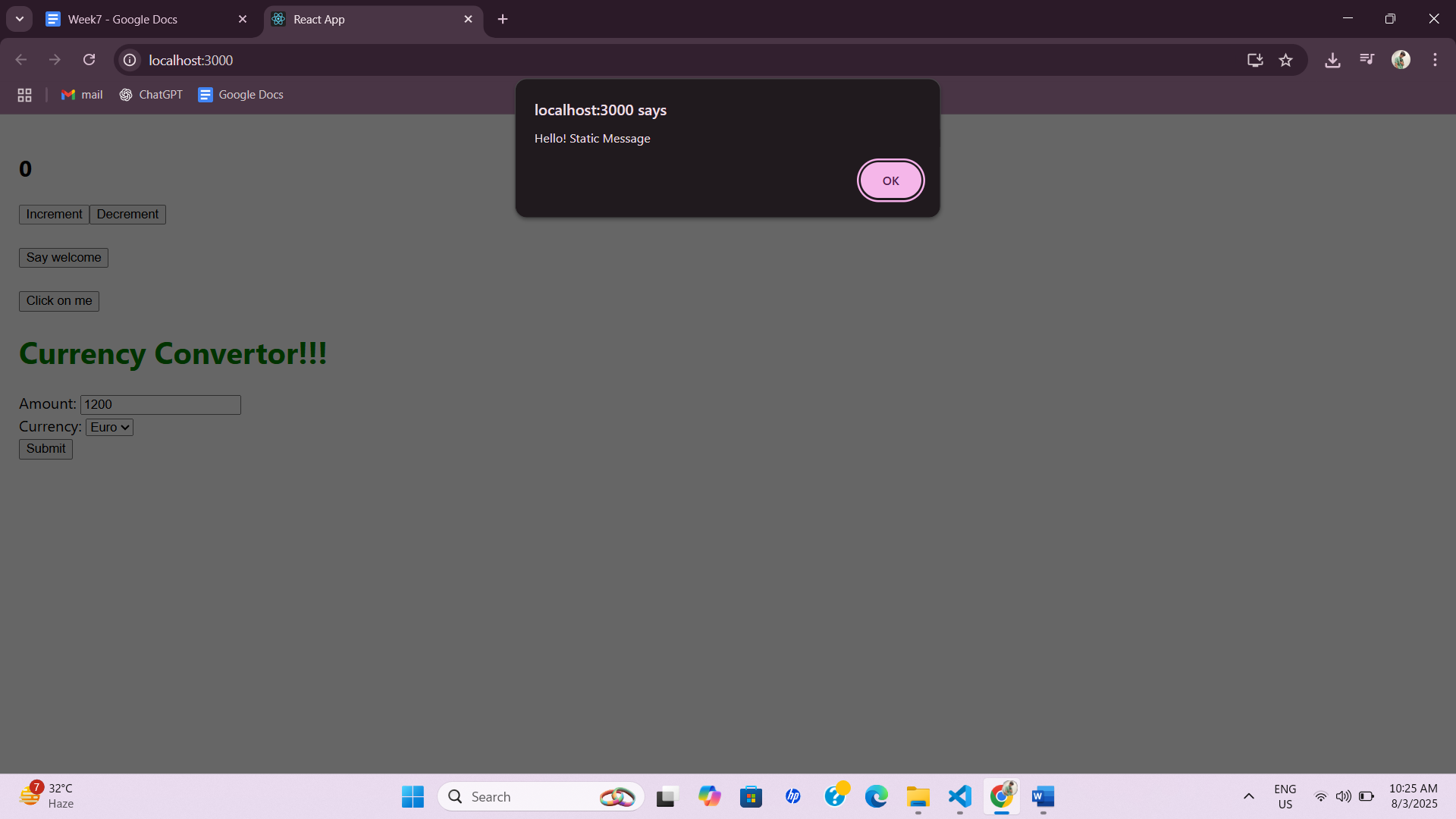
**);**

**}**

**export default CurrencyConvertor;**

**Result:**

****

****

**Solution 12:**

* **Explain about conditional rendering in React**

It allows you to show or hide components based on conditions like user login status. You can use if, ternary (? :), or logical && to render different UI.

* **Define element variables**

You can store React elements in variables and render them dynamically. This makes it easy to switch between components based on conditions.

* **Explain how to prevent components from rendering**

Returning null from a component stops it from rendering anything on the UI. React ignores components that return null.

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

**import React, { useState } from "react";**

**function UserGreeting() {**

**return <h2>Welcome User!</h2>;**

**}**

**function GuestGreeting() {**

**return <h2>Please log in.</h2>;**

**}**

**function Greeting(props) {**

**const isLoggedIn = props.isLoggedIn;**

**if (isLoggedIn) {**

**return <UserGreeting />;**

**}**

**return <GuestGreeting />;**

**}**

**function LoginButton(props) {**

**return (**

**<button onClick={props.onClick}>**

**Login**

**</button>**

**);**

**}**

**function LogoutButton(props) {**

**return (**

**<button onClick={props.onClick}>**

**Logout**

**</button>**

**);**

**}**

**function App() {**

**const [isLoggedIn, setIsLoggedIn] = useState(false);**

**const handleLogin = () => setIsLoggedIn(true);**

**const handleLogout = () => setIsLoggedIn(false);**

**let button;**

**if (isLoggedIn) {**

**button = <LogoutButton onClick={handleLogout} />;**

**} else {**

**button = <LoginButton onClick={handleLogin} />;**

**}**

**return (**

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

**<Greeting isLoggedIn={isLoggedIn} />**

**{button}**

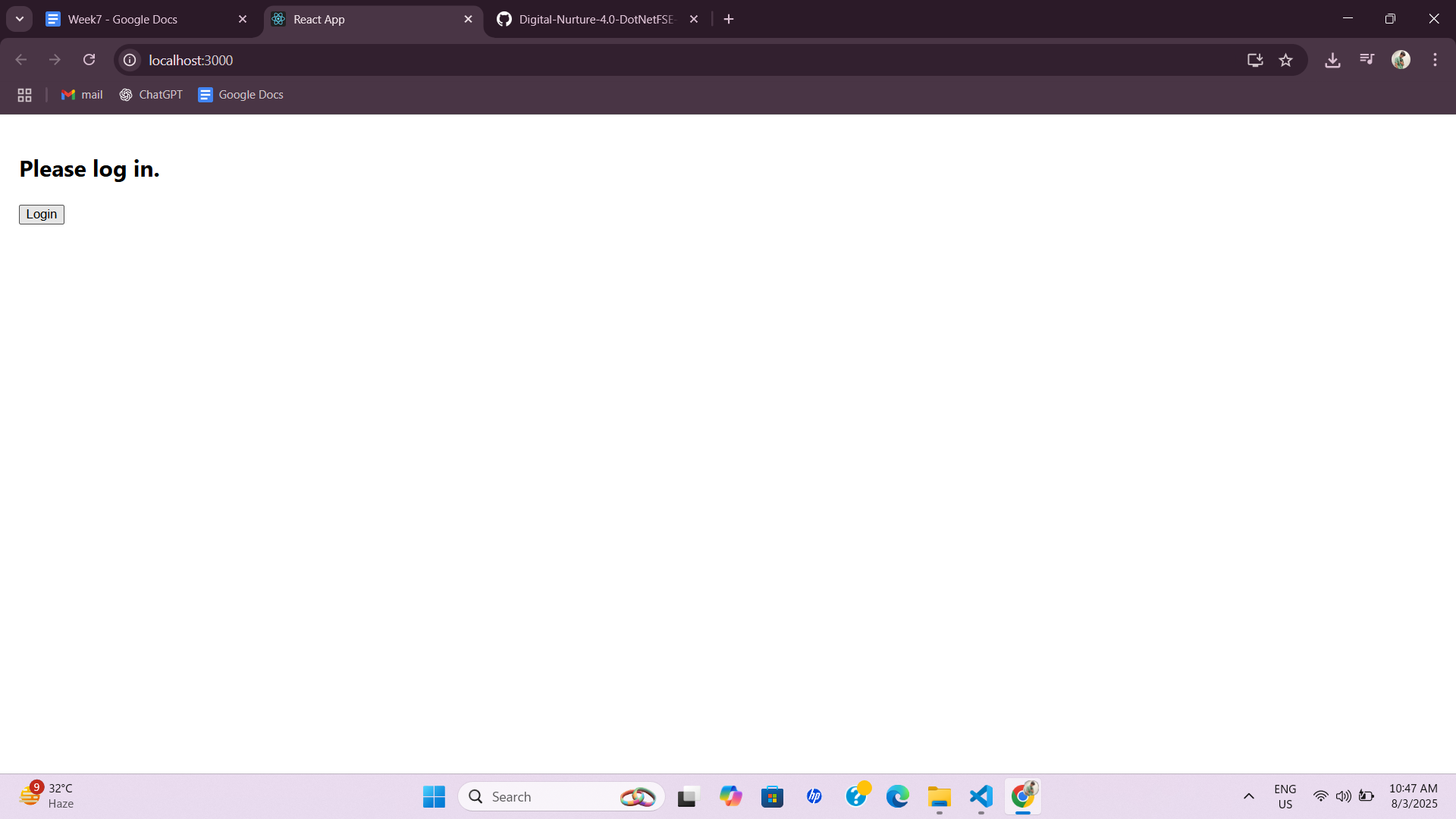
**</div>**

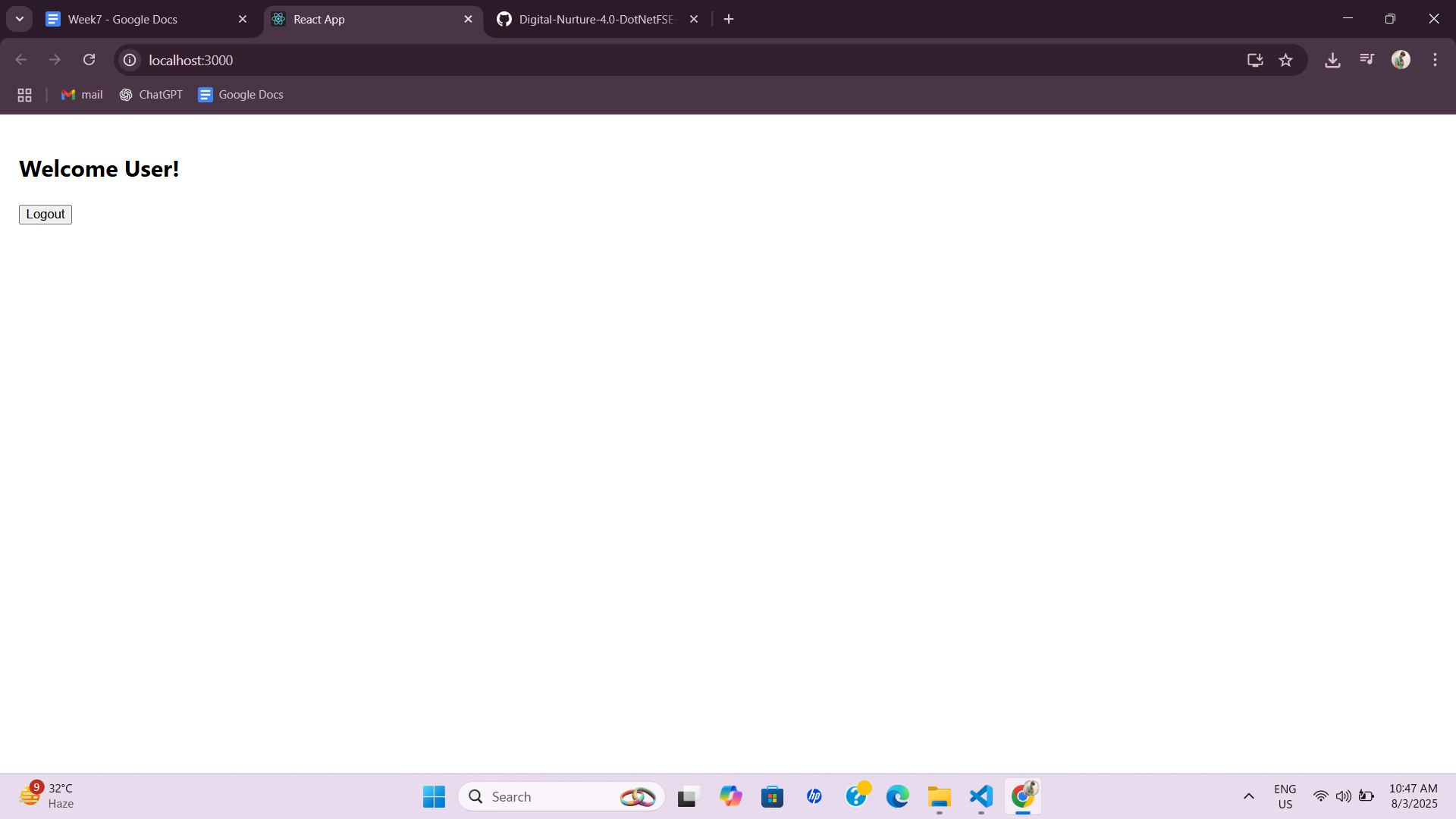
**);**

**}**

**export default App;**

**Result:**

****

****

**Solution 13:**

* **Explain various ways of conditional rendering**

Various ways of conditional rendering: You can use if/else, ternary operators (condition ? A : B), or logical && to render components based on conditions. React also allows returning null to hide elements conditionally.

* **Explain how to render multiple components**

Render multiple components: Wrap multiple components inside a parent element like <div> or use React Fragments (<> </>). Arrays can also be used to render multiple components dynamically.

* **Define list component**

List component: A list component in React displays a collection of data items using .map() to render each element as a child component. It helps in managing and displaying dynamic lists efficiently.

* **Explain about keys in React applications**

Keys in React applications: Keys are unique identifiers used to help React track elements when lists change. They improve rendering performance and prevent unnecessary re-renders.

* **Explain how to extract components with keys**

Extract components with keys: When mapping lists, you can extract list items into separate components and pass unique keys as props. This keeps the component reusable and maintains correct rendering order.

* **Explain React Map, map() function**

React Map, map() function: The .map() function is used to iterate over arrays and return a list of React elements. It’s essential for dynamically rendering multiple components from data.

**Lab:**

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

**import React from 'react';**

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

**import BookDetails from './BookDetails';**

**import BlogDetails from './BlogDetails';**

**import CourseDetails from './CourseDetails';**

**import { books, courses, blogs } from './data';**

**function App() {**

**return (**

**<div style={{ display: 'flex', justifyContent: 'space-around' }}>**

**<CourseDetails courses={courses} showCourses={true} />**

**<BookDetails books={books} showBooks={true} />**

**<BlogDetails blogs={blogs} showBlogs={true} />**

**</div>**

**);**

**}**

**export default App;**

[**BookDetails.js**](http://bookdetails.js)

**import React from 'react';**

**function BookDetails({ books, showBooks }) {**

**return (**

**<div className="st2">**

**<h1>Book Details</h1>**

**{showBooks && books.length > 0 ? (**

**books.map((book) => (**

**<div key={book.id}>**

**<h3>{book.bname}</h3>**

**<h4>{book.price}</h4>**

**</div>**

**))**

**) : (**

**<p>No Books Available</p>**

**)}**

**</div>**

**);**

**}**

**export default BookDetails;**

[**BlogDetails.js**](http://blogdetails.js)

**import React from 'react';**

**function BlogDetails({ blogs, showBlogs }) {**

**return (**

**<div className="v1">**

**<h1>Blog Details</h1>**

**{showBlogs ? (**

**blogs.map((blog) => (**

**<div key={blog.id}>**

**<h3>{blog.title}</h3>**

**<h5>{blog.author}</h5>**

**<p>{blog.content}</p>**

**</div>**

**))**

**) : (**

**<p>No Blogs to Display</p>**

**)}**

**</div>**

**);**

**}**

**export default BlogDetails;**

[**CourseDetails.js**](http://coursedetails.js)

**import React from 'react';**

**function CourseDetails({ courses, showCourses }) {**

**if (!showCourses || courses.length === 0) {**

**return (**

**<div className="mystyle1">**

**<h1>Course Details</h1>**

**<p>No Courses Available</p>**

**</div>**

**);**

**}**

**return (**

**<div className="mystyle1">**

**<h1>Course Details</h1>**

**{courses.map((course) => (**

**<div key={course.id}>**

**<h3>{course.cname}</h3>**

**<h4>{course.date}</h4>**

**</div>**

**))}**

**</div>**

**);**

**}**

**export default CourseDetails;**

[**data.js**](http://data.js)

**export const books = [**

**{ id: 101, bname: 'Master React', price: 670 },**

**{ id: 102, bname: 'Deep Dive into Angular 11', price: 800 },**

**{ id: 103, bname: 'Mongo Essentials', price: 450 },**

**];**

**export const courses = [**

**{ id: 1, cname: 'Angular', date: '4/5/2021' },**

**{ id: 2, cname: 'React', date: '6/3/2021' },**

**];**

**export const blogs = [**

**{**

**id: 1,**

**title: 'React Learning',**

**author: 'Stephen Biz',**

**content: 'Welcome to learning React!',**

**},**

**{**

**id: 2,**

**title: 'Installation',**

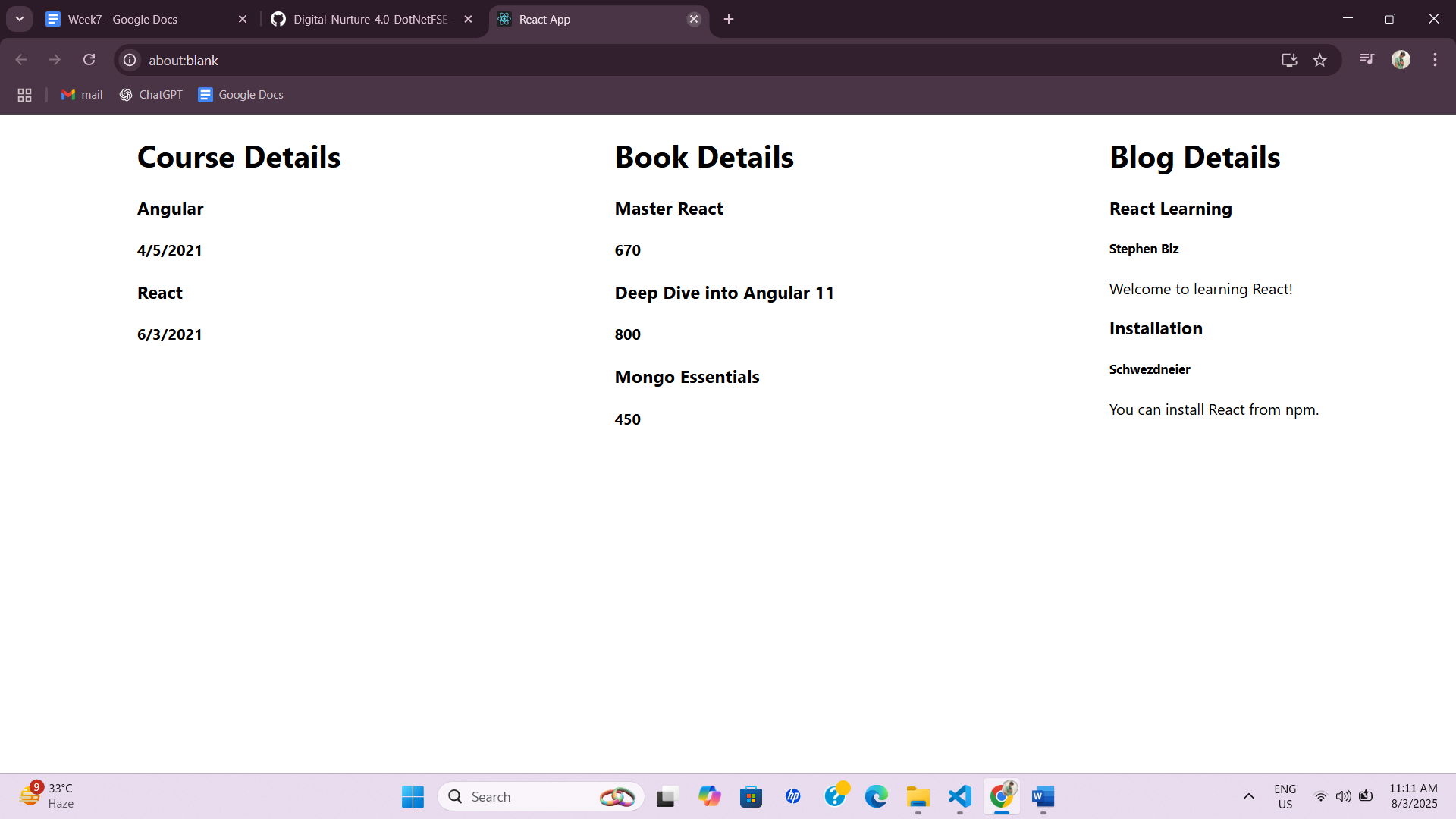
**author: 'Schwezdneier',**

**content: 'You can install React from npm.',**

**},**

**];**

**RESULT:**

****

**-:Thank you:-**