## **WEEK – 7 Mandatory Hands On**

## **9. React JS - HOL**

## **Objectives**

* **List the features of ES6**

ES6, also known as ECMAScript 2015, introduced a major update to JavaScript with many powerful and modern features. It added new ways to declare variables (let, const), introduced arrow functions for concise syntax, and brought class-based object-oriented programming with the class and extends keywords. ES6 also included default function parameters, template literals for cleaner strings, destructuring for arrays and objects, the spread and rest operators for handling collections, and the introduction of Promises for better asynchronous programming.

* **Explain JavaScript let**

The let keyword in JavaScript is used to declare variables that are block-scoped. This means the variable exists only within the nearest set of curly braces ({}), such as within a loop, if-statement, or function block. Unlike var, which is function-scoped and can lead to unexpected behavior in larger programs, let prevents accidental access or modification of variables outside their intended scope.

* **Identify the differences between var and let**

While both var and let are used to declare variables in JavaScript, there are significant differences between them. var is function-scoped, which means the variable is accessible anywhere within the function it was declared in, even before the line of declaration due to hoisting. However, this can lead to unexpected results. On the other hand, let is block-scoped, meaning the variable can only be accessed within the block in which it is defined. Also, variables declared with var can be re-declared in the same scope, whereas let does not allow re-declaration within the same scope. These differences make let a safer and more predictable choice for variable declarations.

* **Explain JavaScript const**

The const keyword is used in JavaScript to declare variables that cannot be reassigned after their initial value is set. Like let, const is block-scoped, meaning it is accessible only within the block in which it is defined. const is especially useful for defining constants or fixed values that should not change during the execution of a program, such as configuration values or references to functions. However, it is important to note that if a const is used to declare an object or array, the contents of that object or array can still be modified; only the binding itself cannot be changed. This helps enforce immutability in code, leading to more reliable and easier-to-maintain programs.

* **Explain ES6 class fundamentals**

ES6 introduced a new class syntax to JavaScript, bringing a cleaner and more understandable way to create objects and implement inheritance. Prior to ES6, JavaScript used prototype-based inheritance, which could be difficult to read and manage. With ES6 classes, developers can define a blueprint for objects using a constructor method and define methods directly inside the class body.

* **Explain ES6 class inheritance**

In ES6, inheritance between classes is implemented using the extends and super keywords. When one class extends another, it inherits all properties and methods of the parent class. The super() function is used inside the child class constructor to call the constructor of the parent class, allowing access to the parent’s properties and methods. This inheritance model simplifies code reuse and makes it easier to implement hierarchical relationships, such as a Dog class inheriting from an Animal class.

* **Define ES6 arrow functions**

Arrow functions, introduced in ES6, provide a more concise way to write function expressions in JavaScript. Instead of using the traditional function keyword, arrow functions use the => syntax. One of the most important characteristics of arrow functions is that they do not have their own this context; instead, they inherit this from the surrounding lexical scope. This makes them particularly useful in situations like callbacks and event handlers, where the value of this needs to remain consistent. Arrow functions also reduce the amount of boilerplate code and improve the readability of short functions

* **Identify set(), map()**

A Set is a collection of unique values, meaning it automatically removes duplicate entries. It supports methods like .add(), .has(), and .delete() to manage its elements. Set is ideal for scenarios where duplicate values must be avoided. A Map is a collection of key-value pairs, where keys can be of any data type (not just strings or symbols as in objects). Maps maintain insertion order and provide methods like .set(), .get(), .has(), and .delete().

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

* Use map() method of ES6
* Apply arrow functions of ES6
* Implement Destructuring features of ES6

## **Prerequisites**

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

* Node.js
* NPM
* Visual Studio Code

## **Notes**

Estimated time to complete this lab: **60 minutes.**

Create a React Application named “cricketapp” with the following components:

1. ListofPlayers

* Declare an array with 11 players and store details of their names and scores using the map feature of ES6



* Filter the players with scores below 70 using arrow functions of ES6.



1. IndianPlayers
   1. Display the Odd Team Player and Even Team players using the Destructuring features of ES6



* 1. Declare two arrays T20players and RanjiTrophy players and merge the two arrays and display them using the Merge feature of ES6



Display these two components in the same home page using a simple if else in the flag variable.

**Output:**

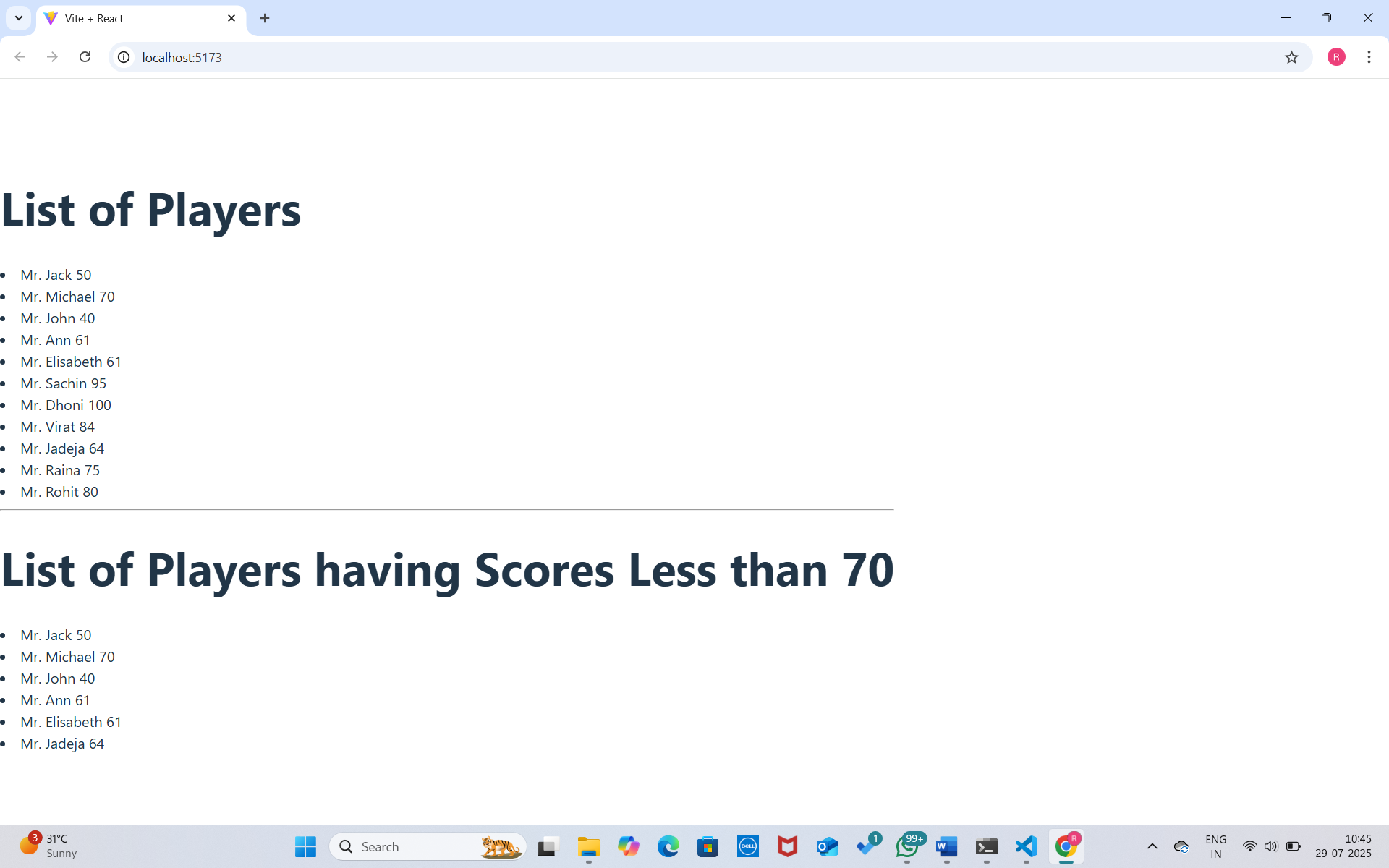
When Flag=true

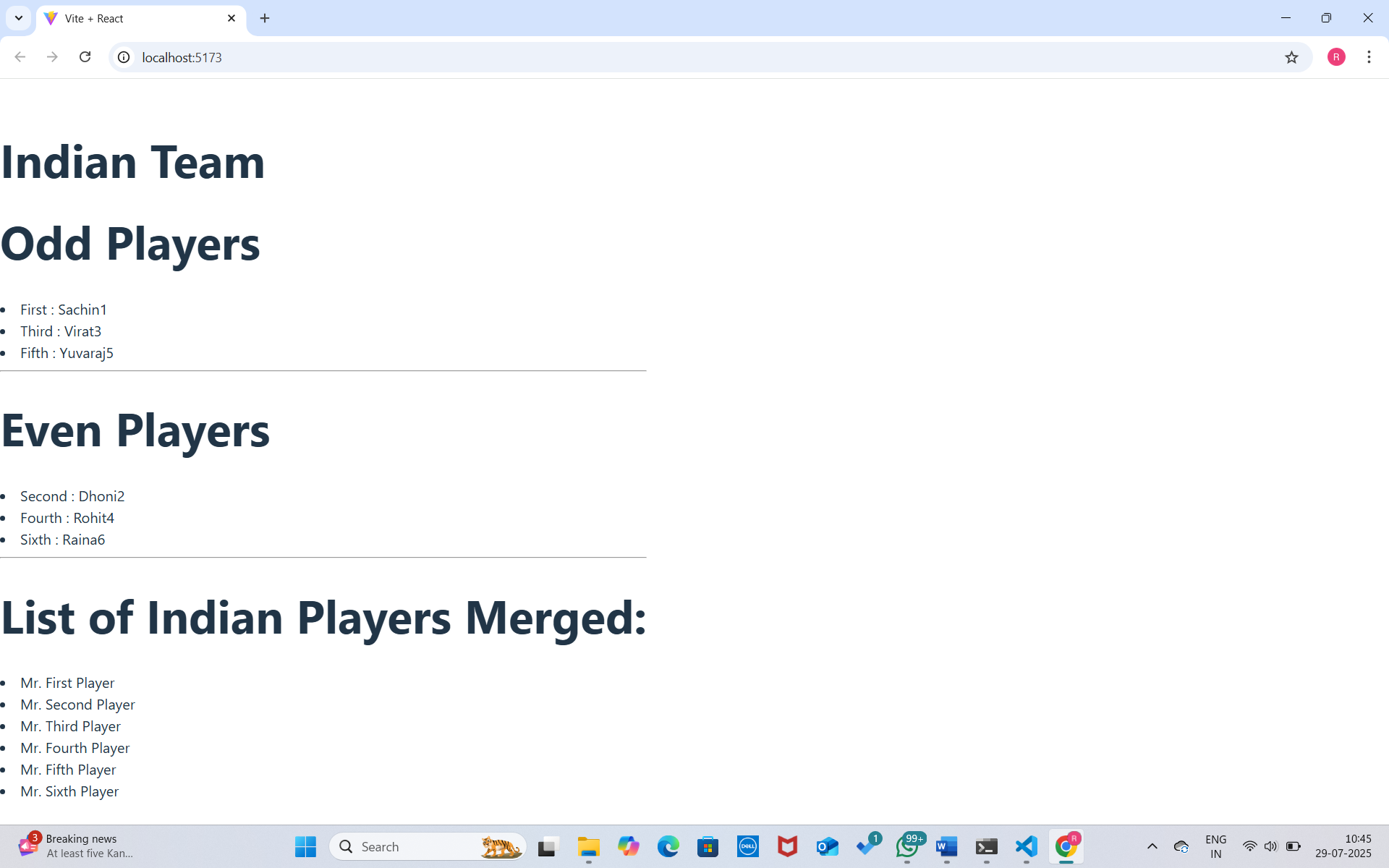


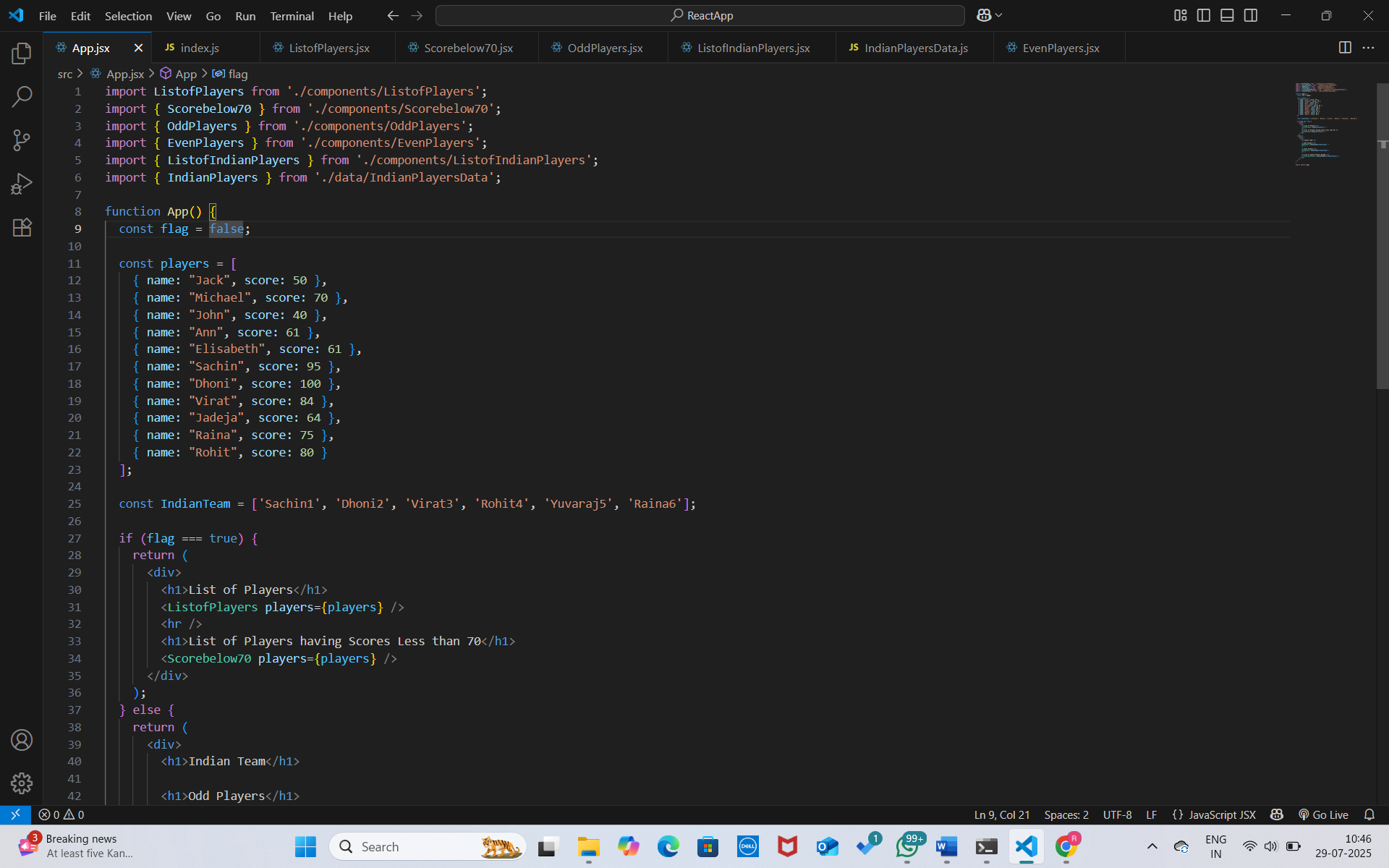
When Flag=false



**OUTPUT:**







**10. React JS - HOL**

**Objectives**

* **Define JSX**

JSX (JavaScript XML) is a syntax extension for JavaScript used in React to describe the UI structure. It allows developers to write HTML-like code directly within JavaScript, making it easier to visualize and build user interfaces. Under the hood, JSX is compiled into React.createElement() calls, which create virtual DOM nodes. JSX improves readability and simplifies the development of React components by closely resembling the final HTML output.

* **Explain about ECMA Script**

ECMAScript (often abbreviated as ES) is the standard specification that defines the scripting language JavaScript. It provides the rules, details, and guidelines that all JavaScript engines (like V8, SpiderMonkey) must follow. ECMAScript versions such as ES5, ES6 (ECMAScript 2015), and later editions introduce new features and syntax improvements to the language. React applications typically rely on modern ECMAScript features like let, const, arrow functions, destructuring, and classes to write clean and efficient code.

**Explain React.createElement()**

React.createElement() is a core function in React used to create React elements (virtual DOM nodes). Before JSX, or when JSX is transpiled, React developers manually used this function to build component trees. It takes three arguments: the type of element (like 'div' or a component), props (attributes), and children (nested elements or text). Although JSX is preferred for readability, understanding React.createElement() helps developers grasp how JSX works under the hood.

* **Explain how to create React nodes with JSX**

React nodes (or elements) can be created in JSX by writing HTML-like syntax inside a JavaScript file. For example, writing <h1>Hello World</h1> inside a React component creates a React element. These JSX elements can represent standard HTML tags, custom React components, or fragments. JSX elements must follow certain rules, like using className instead of class, and having a single parent element.

* **Define how to render JSX to DOM**

In React, JSX is rendered to the DOM using the ReactDOM.createRoot() method (React 18+) or ReactDOM.render() (in earlier versions). This method attaches the root React component (usually <App />) to a DOM element, often a <div> with the ID root in index.html. The JSX gets converted into virtual DOM, which is then efficiently compared and updated in the actual DOM through React’s reconciliation process.

* **Explain how to use JavaScript expressions in JSX**

JSX allows embedding JavaScript expressions inside curly braces {}. This includes variables, function calls, arithmetic operations, and conditional expressions (using ternary operator). These expressions are evaluated at runtime and the result is rendered. This feature enables dynamic and interactive UI development within the React component render method or return block.

* **Explain how to use inline CSS in JSX**

In JSX, inline CSS is applied using the style attribute, but instead of a string, it takes a JavaScript object. CSS properties are written in camelCase instead of kebab-case (e.g., backgroundColor instead of background-color). This allows dynamic styling based on component logic and state.

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

* Use JSX syntax in React applications
* Use inline CSS in JSX

**Prerequisites**

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

* Node.js
* NPM
* Visual Studio Code

**Notes**

Estimated time to complete this lab: 60 minutes.

Create a React Application named “officespacerentalapp” which uses React JSX to create elements, attributes and renders DOM to display the page.

Create an element to display the heading of the page.

Attribute to display the image of the office space

Create an object of office to display the details like Name, Rent and Address.

Create a list of Object and loop through the office space item to display more data.

To apply Css, Display the color of the Rent in Red if it’s below 60000 and in Green if it’s above 60000.

**Output:**

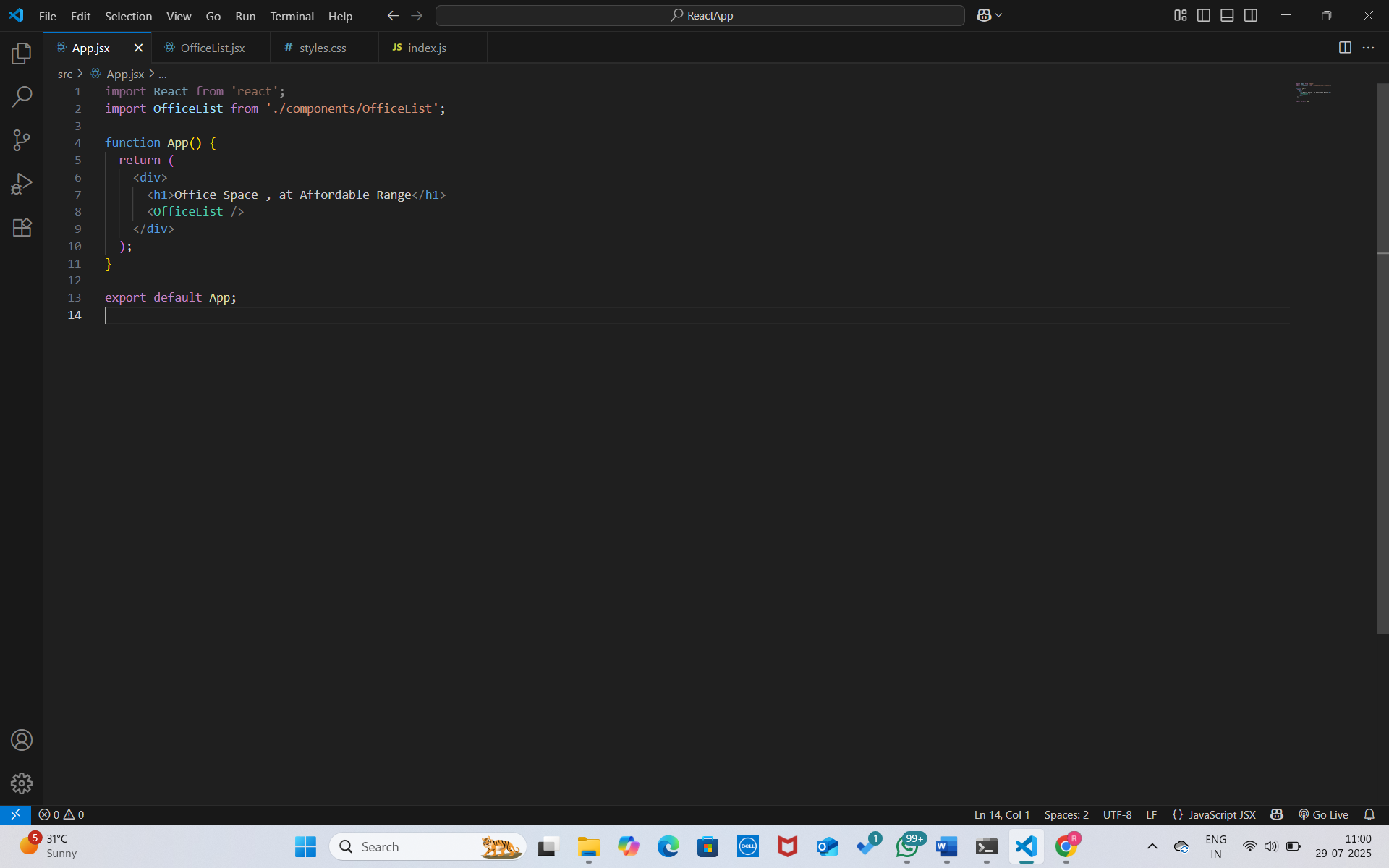
****

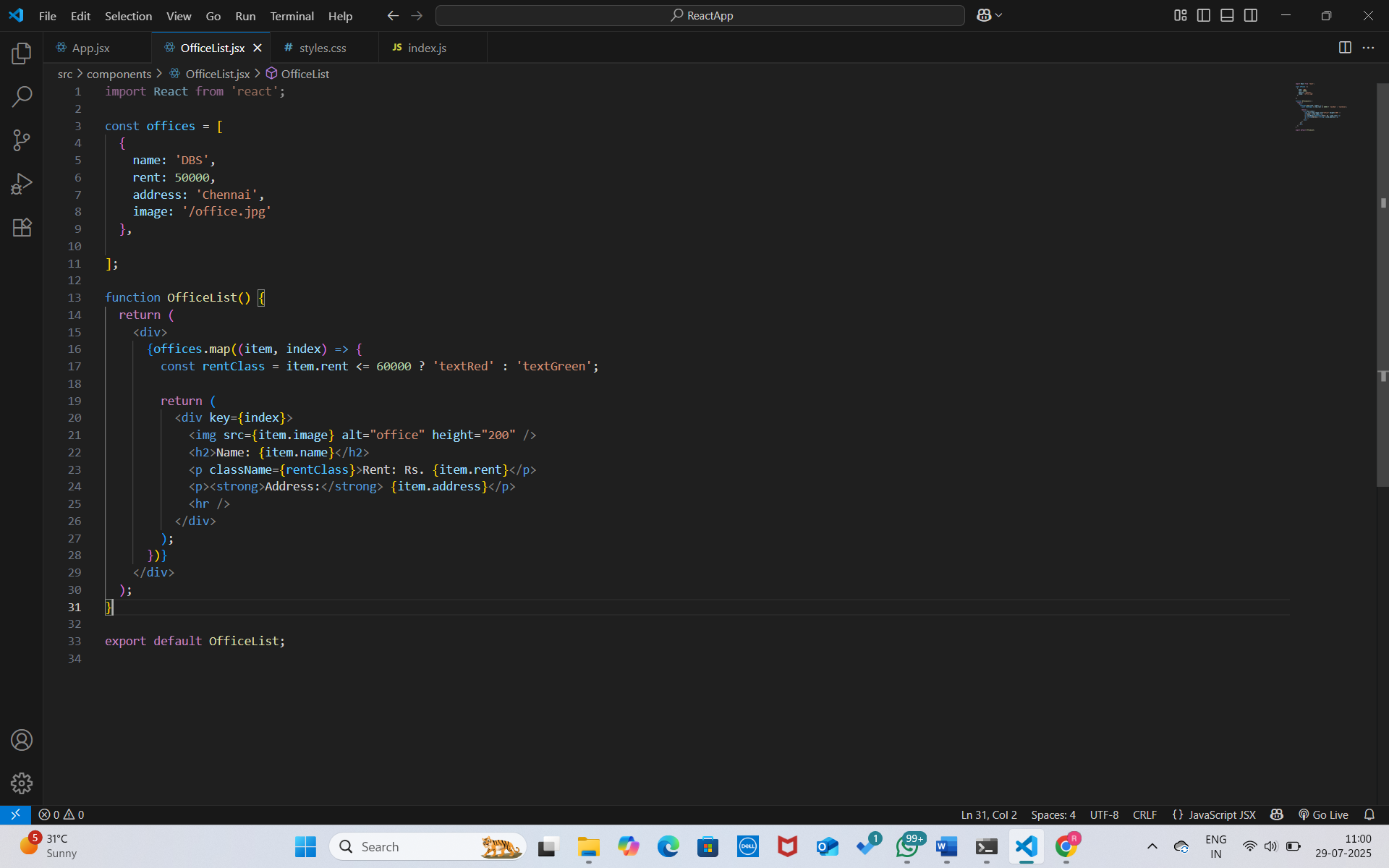
**Hint:**

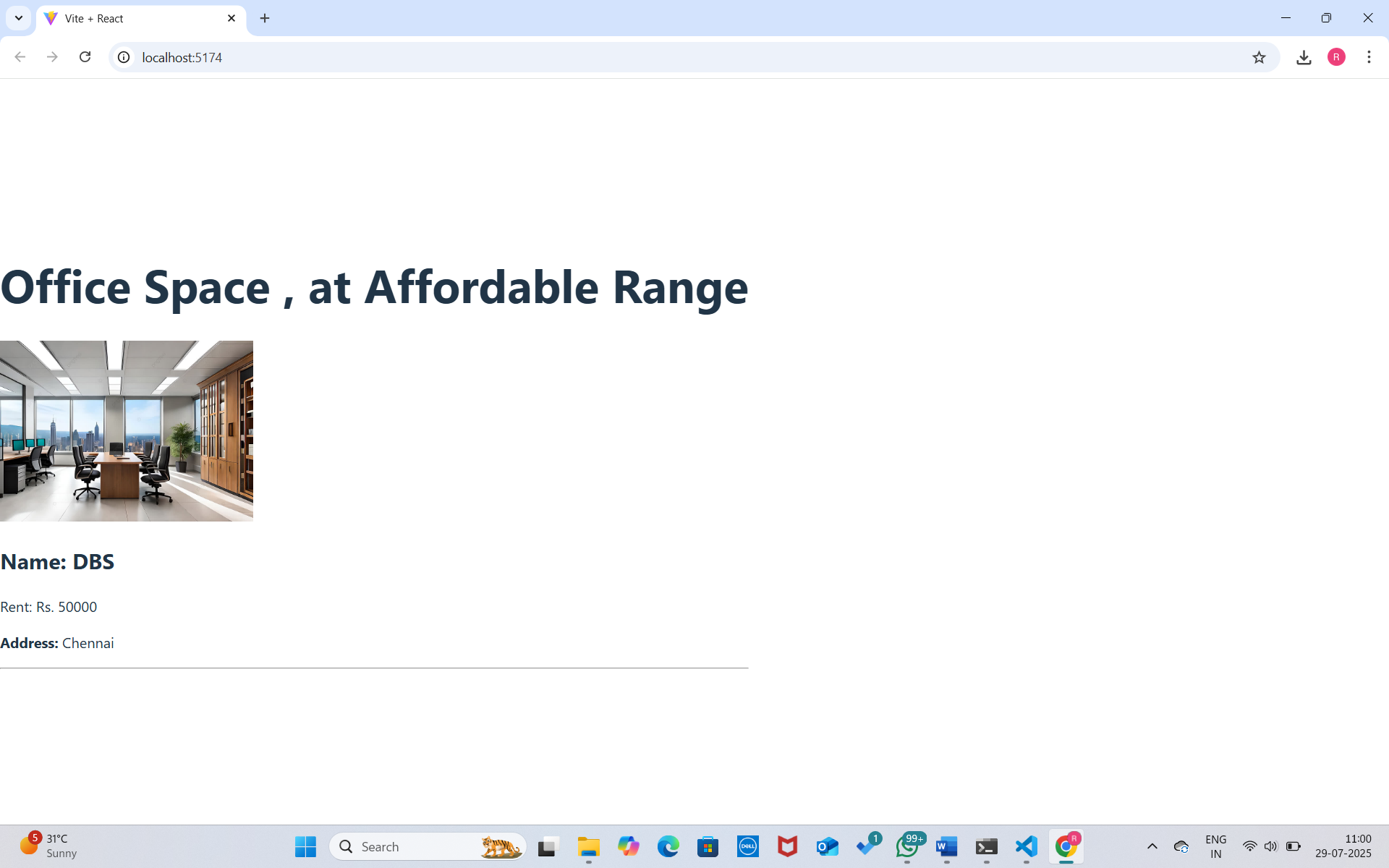
****

****

**OUTPUT:**







**11. React JS - HOL**

**Objectives**

* **Explain React events**

React events are objects that capture user interactions with the UI, such as mouse clicks, keyboard presses, form submissions, or other browser-based events. These events allow React components to detect and respond to user actions, enabling dynamic and interactive web applications.

* **Explain about event handlers**

Event handlers are functions in React that are triggered in response to specific events. These functions are typically passed as props to JSX elements using event attributes like onClick, onChange, or onSubmit. When the event occurs, the corresponding handler is executed to update the component's state, perform calculations, or carry out other logic.

* **Define Synthetic event**

A SyntheticEvent in React is a wrapper around the browser’s native event system. It normalizes events to work consistently across different browsers, providing a single interface for handling events. Synthetic events have the same interface as native events but offer better performance and compatibility in React’s virtual DOM environment.

* **Identify React event naming convention**

In React, event names are written using camelCase (e.g., onClick, onMouseEnter) instead of the lowercase convention used in HTML (e.g., onclick, onmouseenter). Event handler functions are passed as values to these camelCase attributes in JSX, ensuring consistency and integration with JavaScript's case-sensitive syntax.

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

* Implement Event handling concept in React applications
* Use this keyword
* Use synthetic event

**Prerequisites**

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

* Node.js
* NPM
* Visual Studio Code

**Notes**

Estimated time to complete this lab: 90 minutes.

Create a React Application “eventexamplesapp” to handle various events of the form elements in HTML.

1. Create “Increment” button to increase the value of the counter and “Decrement” button to decrease the value of the counter. The “Increase” button should invoke multiple methods.
   1. To increment the value
   2. Say Hello followed by a static message.



1. Create a button “Say Welcome” which invokes the function which takes “welcome” as an argument.



1. Create a button which invokes synthetic event “OnPress” which display “I was clicked”

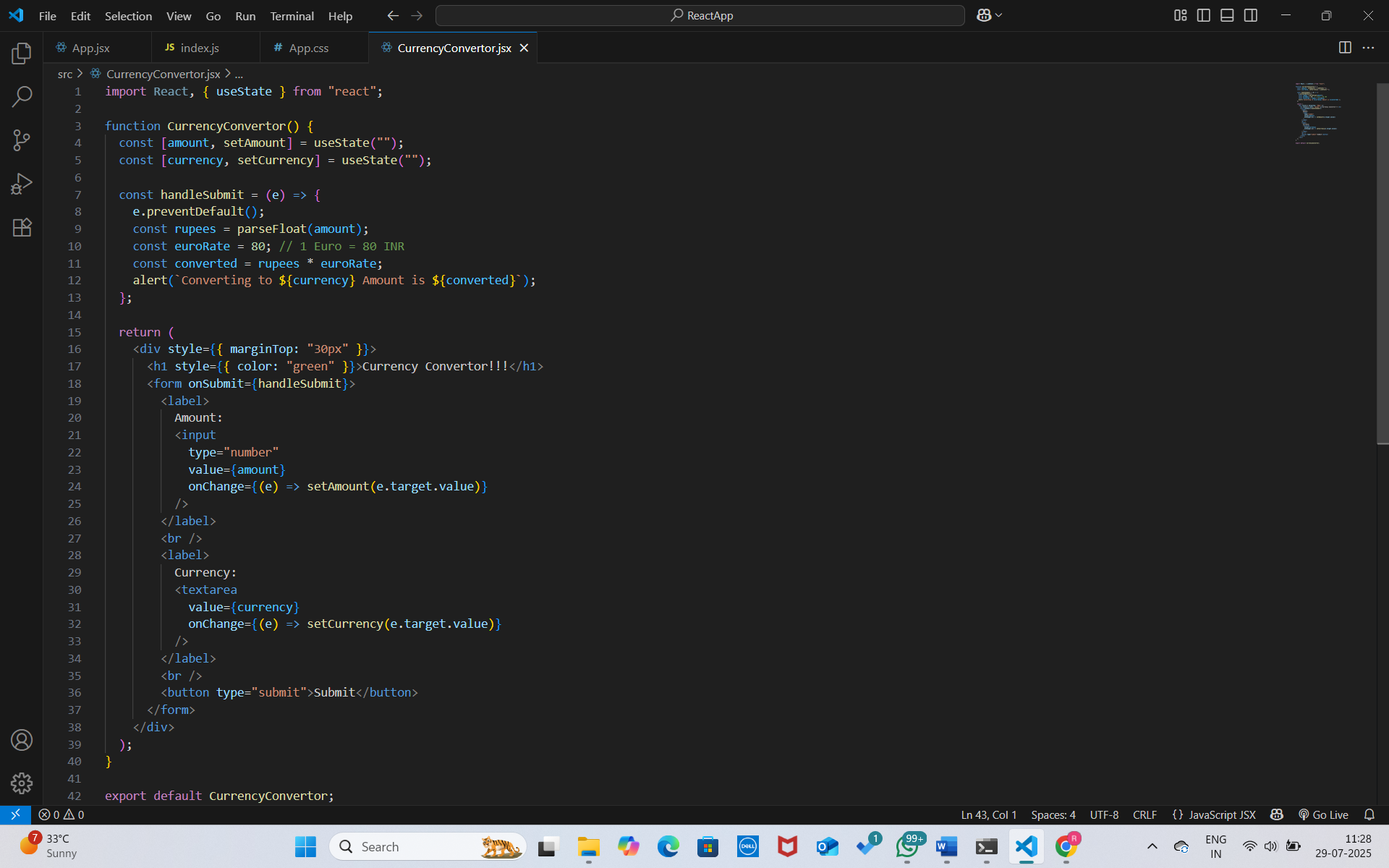


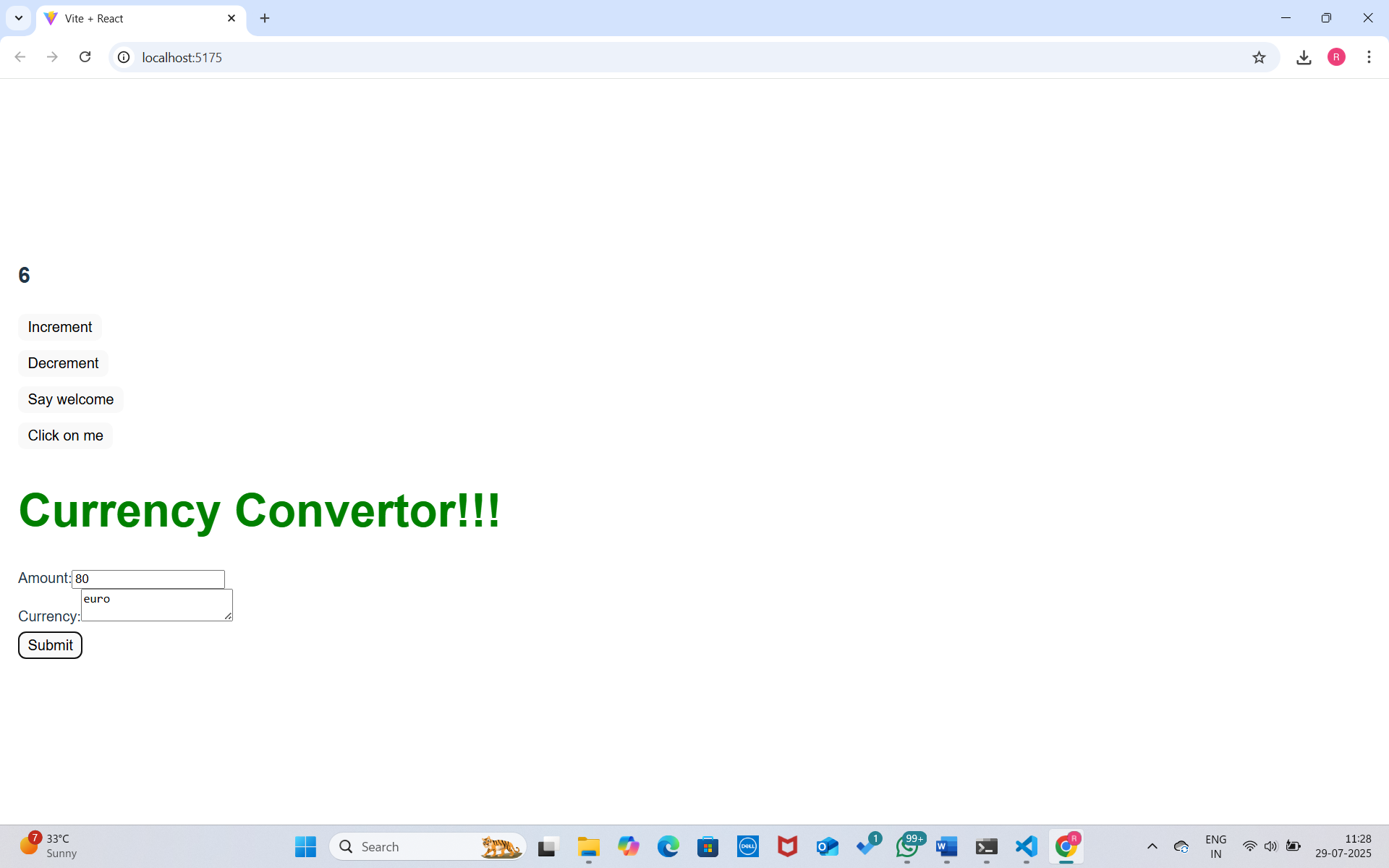
Create a “CurrencyConvertor” component which will convert the Indian Rupees to Euro when the Convert button is clicked.

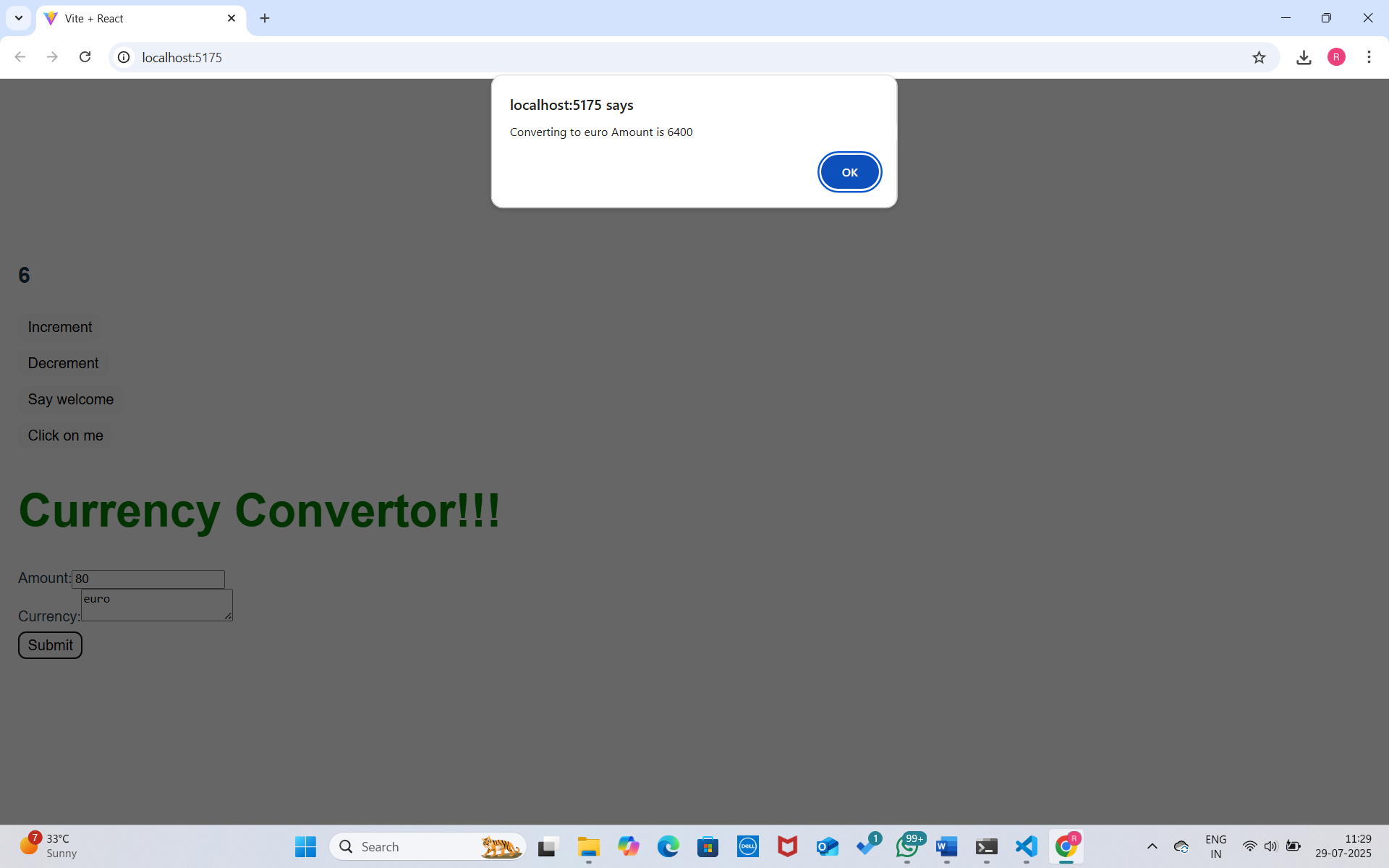
Handle the Click event of the button to invoke the handleSubmit event and handle the conversion of the euro to rupees.

****

**OUTPUT:**







**12. React JS - HOL**

**Objectives**

* **Explain about conditional rendering in React**

Conditional rendering in React refers to the process of displaying components or elements only when specific conditions are met. It works similarly to how conditions are handled in JavaScript — using if, else, ternary operators, or logical &&. In a React application, you may want to render different UI elements depending on the application's state or props. For example, you may show a login form if a user is not logged in, and a dashboard if they are.if/else,ternary and logical AND and OR.

* **Define element variables**

Element variables in React are JavaScript variables that are used to store JSX elements. These variables can then be inserted into the JSX to conditionally or dynamically render parts of the UI. This approach helps in organizing and simplifying the component's render logic, especially when multiple rendering paths or complex conditions are involved. Instead of placing conditional logic directly inside the JSX markup, you can declare a variable, assign a JSX expression to it based on some condition, and then use that variable in the return statement. This improves code readability and maintainability by separating logic from structure within a component.

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

In React, preventing a component from rendering means stopping it from being displayed in the user interface based on certain conditions. This can be achieved by returning null from the component or using conditional expressions to decide whether or not a component should be included in the JSX output. When null is returned, React knows not to render anything to the DOM for that component. This approach is useful for hiding elements that are not needed under certain circumstances, such as hiding a warning message when there is no error. Preventing unnecessary rendering not only keeps the UI clean but also helps improve performance by reducing the amount of work React has to do.

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

* Implement conditional rendering in React applications

**Prerequisites**

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

* Node.js
* NPM
* Visual Studio Code

**Notes**

Estimated time to complete this lab: 60 minutes.

Create a React Application named “ticketbookingapp” where the guest user can browse the page where the flight details are displayed whereas the logged in user only can book tickets.

The Login and Logout buttons should accordingly display different pages. Once the user is logged in the User page should be displayed. When the user clicks on Logout, the Guest page should be displayed.

****

****

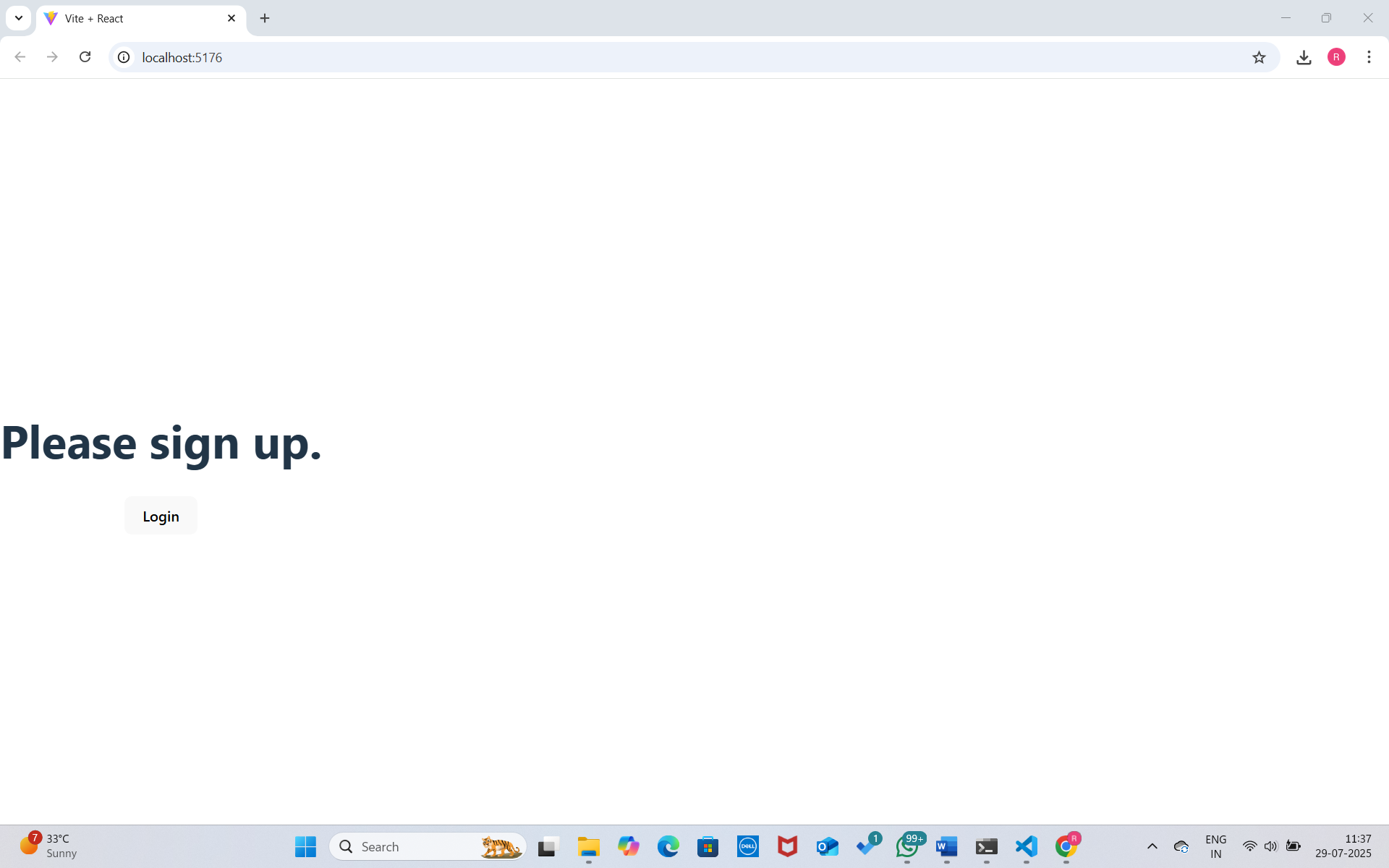
**Hint:**

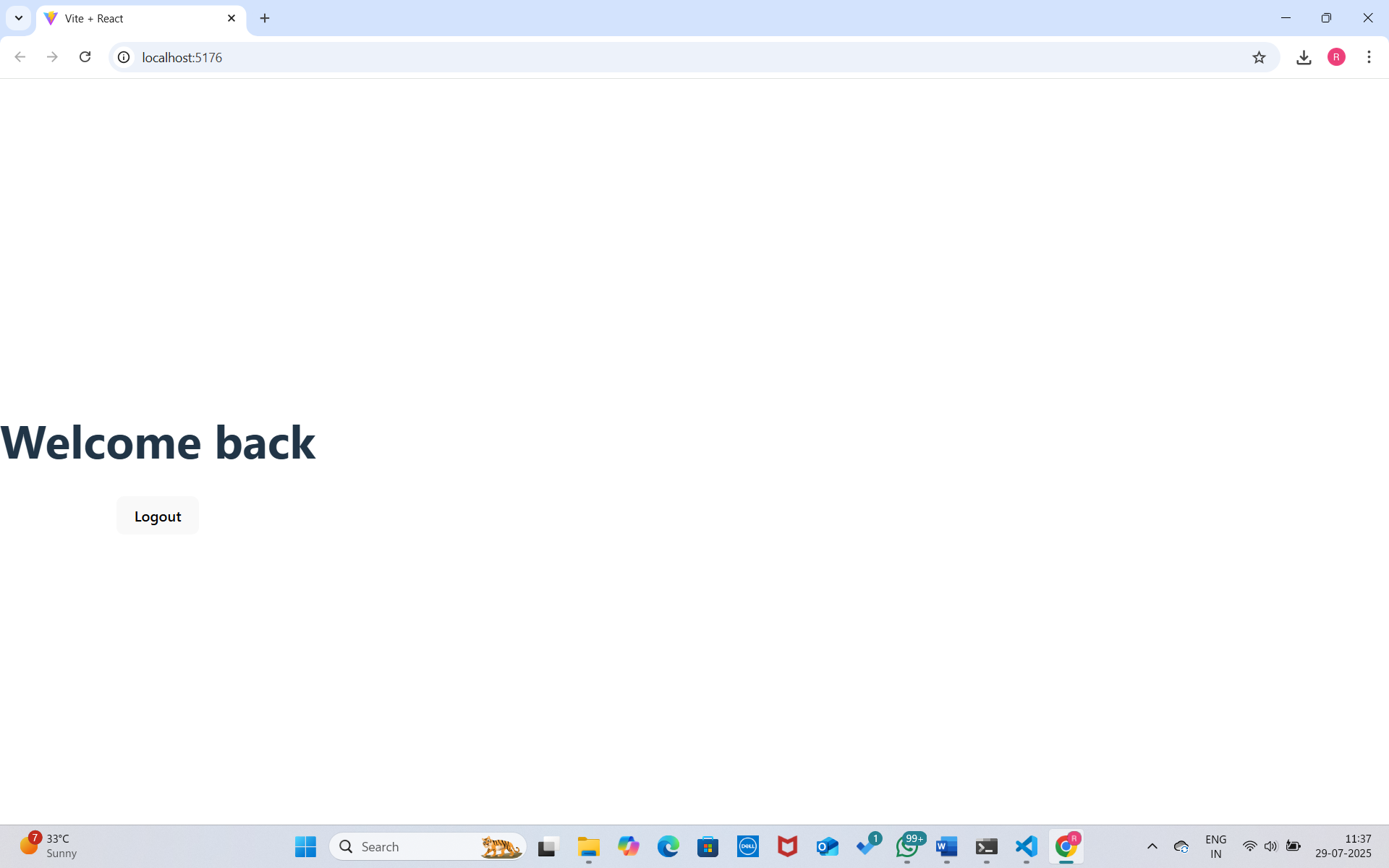
****

****

****

**OUTPUT:**





**13. React JS - HOL**

**Objectives**

* **Explain various ways of conditional rendering**

Conditional rendering in React refers to the ability to render different components or elements based on certain conditions or logic. React uses JavaScript’s conditional operators to control what content is displayed. There are multiple ways to achieve conditional rendering in React:

**Using if/else statements**: This is a straightforward way where you use an if condition to check a value and return different JSX based on the condition.

**Using ternary operators (? :)**: A concise way to render one of two elements based on a condition.

**Logical AND operator (&&)**: Used when you want to render something only if a condition is true.

**Switch case statements**: Useful when multiple conditions exist, such as displaying different content based on a selected value.

* **Explain how to render multiple components**

In React, rendering multiple components means displaying several components in a single render function. This is commonly done by wrapping components inside a single parent container like a <div>, or using a React fragment (<> ... </>) to avoid extra nodes in the DOM. You can define multiple reusable components and include them in your main component or any other component. This helps in structuring the application better and breaking down complex UIs into manageable pieces.

* **Define list component**

A list component in React is a component that is specifically designed to display a collection of data items as a list. This component usually receives an array of items as props and then maps over them to render each item using JSX. List components are vital in building user interfaces that require rendering a group of elements like to-do lists, menu items, posts, or user profiles.

* **Explain about keys in React applications**

Keys are special string attributes you must include when creating lists of elements in React. They help React identify which items have changed, been added, or removed. This is crucial for improving rendering performance, especially when dealing with dynamic lists.When you render lists using map(), React uses the key to match old and new list items to efficiently update the DOM. Keys should be unique among siblings, but stable – meaning they should not change between renders if the item's identity hasn't changed.

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

Extracting components with keys involves moving JSX used inside a list rendering function into a separate component, and still passing a unique key to each item. This improves readability, reusability, and maintainability of your code. For instance, if you're rendering a list of comments, instead of having the JSX inside the map() function directly, you create a CommentItem component and call it with props.

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

The map() function in React is borrowed from JavaScript’s Array.prototype.map() method and is commonly used to iterate over arrays and render elements for each item. In React, it's used to dynamically generate a list of elements or components based on data.

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

* Implement conditional rendering in React applications

**Prerequisites**

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

* Node.js
* NPM
* Visual Studio Code

Notes

Estimated time to complete this lab: 60 minutes.

Create a React App named “bloggerapp” in with 3 components.

1. Book Details
2. Blog Details
3. Course Details

Implement this with as many ways possible of Conditional Rendering**.**

****

**Hint:**

****

****

****

**OUTPUT:**

