**REACTJS-HOL**

**MANDATORY HANDS-ON QUESTIONS**

**Question 9**

Objectives

List the features of ES6

Explain JavaScript let

Identify the differences between var and let

Explain JavaScript const

Explain ES6 class fundamentals

Explain ES6 class inheritance

Define ES6 arrow functions

Identify set(), map()

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

Answer and Output:

Definitions:

1. List the features of ES6.

* let, const for declaring variables
* Arrow functions
* Classes and inheritance
* Template literals (for strings with variables)
* Default parameters
* Destructuring assignment
* Spread and Rest operators
* Promises
* Modules (import/export)
* map(), set(), weakmap(), etc.

1. Explain JavaScript let

The let keyword is used to declare variables that can be updated later. Unlike var, variables declared with let are block scoped which means they only exist inside the block ({}) where they are defined. This prevents bugs caused by accidentally using variables outside of their intended scope.

1. Identify the differences between var and let

* Scope: var is function-scoped, let is block-scoped.
* Hoisting: Both are hoisted, but let is not initialized, so using it before declaration gives an error.
* Re-declaration: var can be re-declared in the same scope, let cannot.

1. Explain JavaScript const

const is used to declare constant variables – values that cannot be reassigned. Just like let, const is also block-scoped. Note: If the const is an object or array, you can still change the properties or elements inside, just not reassign the whole object/array.

1. Explain ES6 class fundamentals

ES6 introduced the class keyword to create objects and deal with inheritance in a simpler way than traditional JavaScript prototype-based inheritance. A class can have a constructor and methods, and you can create multiple objects from a class.

1. Explain ES6 class inheritance

You can create a class that inherits properties and methods from another class using the extends keyword. The super() function is used to call the parent class constructor.

1. Define ES6 arrow functions

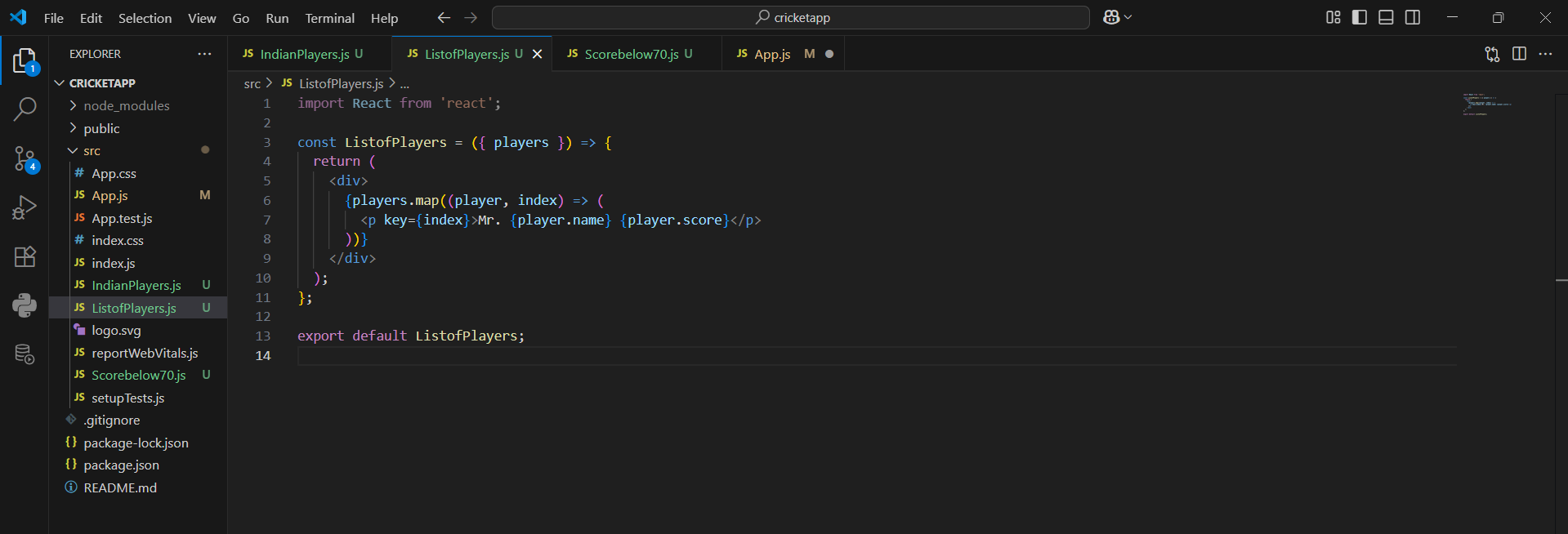
Arrow functions are a shorter way to write functions. They do not have their own this keyword, which makes them useful in many cases (like inside map(), filter(), etc.)

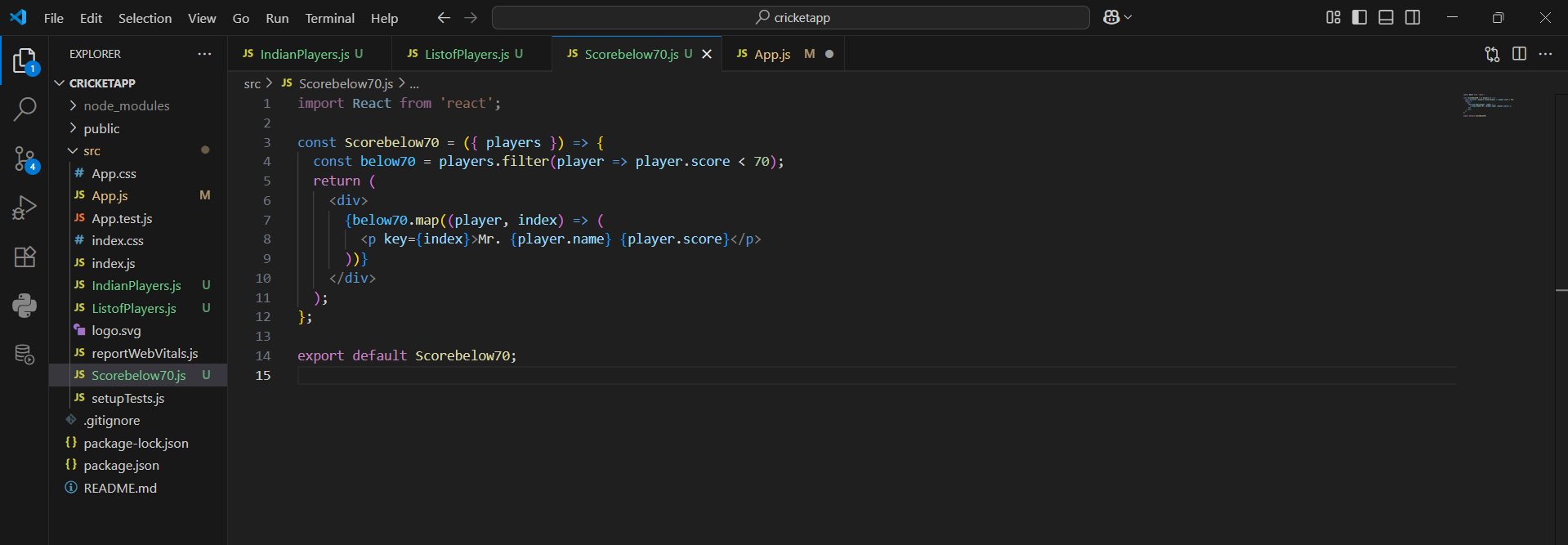
1. Identify set(), map()

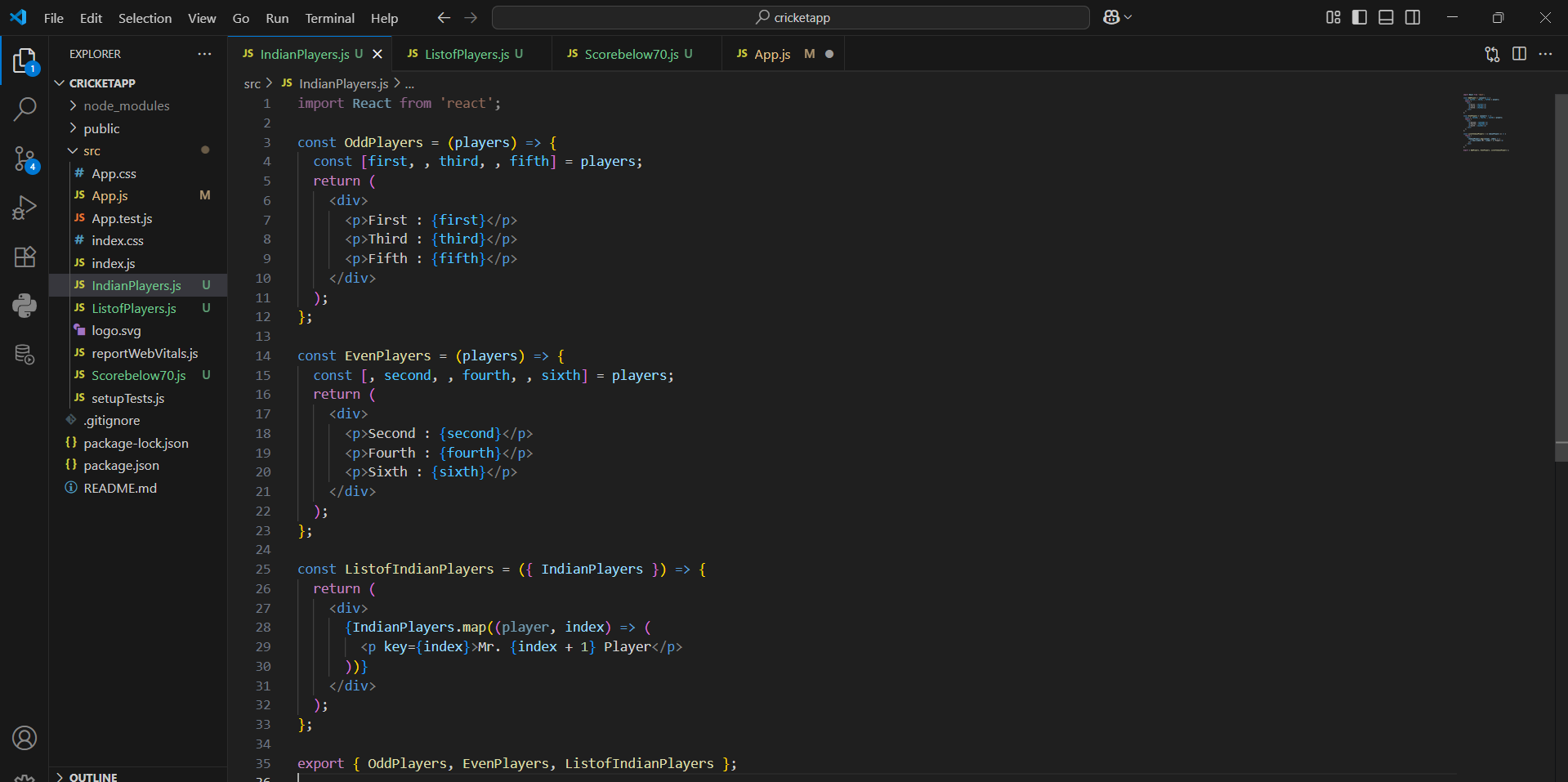
Set: Stores unique values. You can't have duplicates.

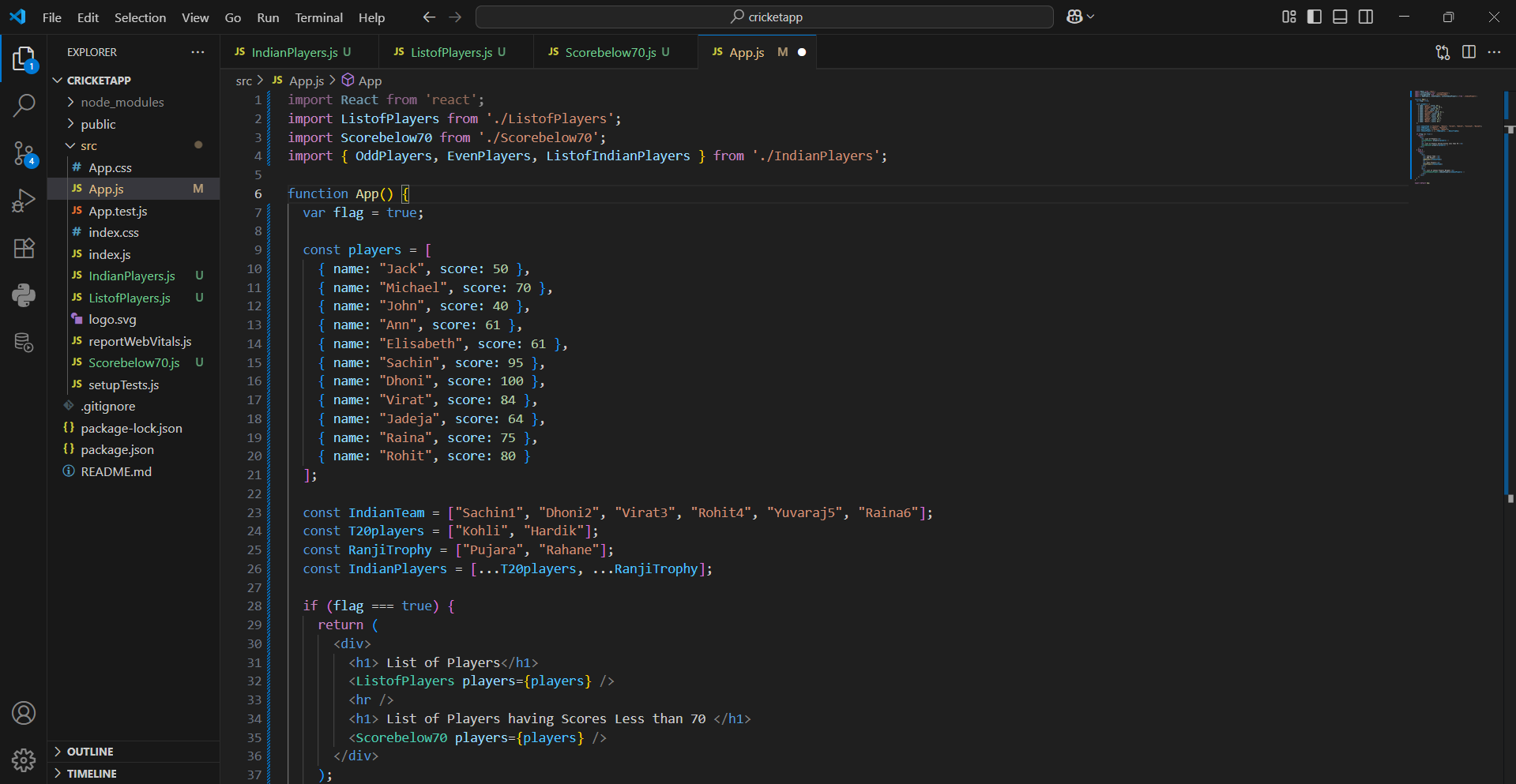
Map: A key-value collection, like an object, but keys can be of any type.

CODE:



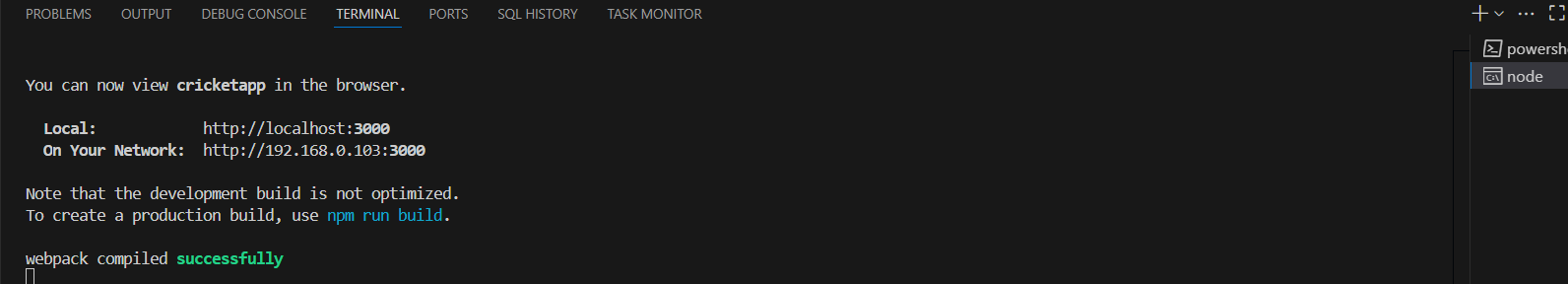




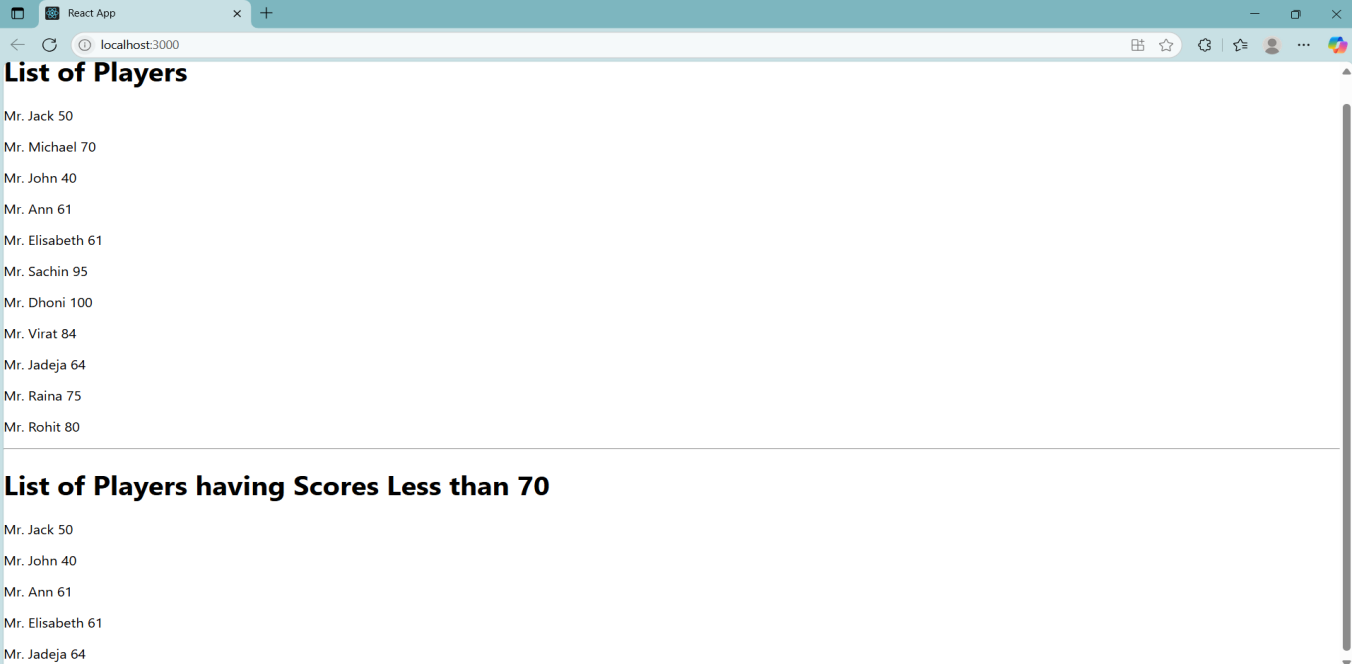




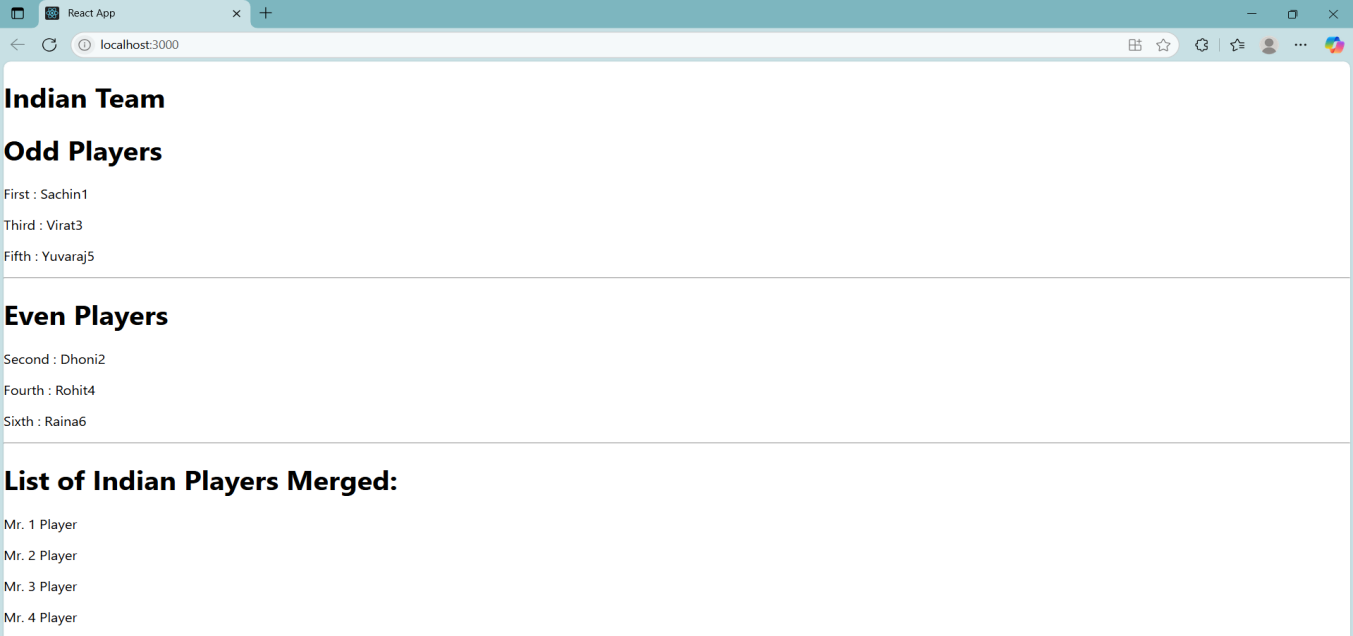
OUTPUT:



When flag=true



When flag=false



**Question 10**

Objectives

Define JSX

Explain about ECMA Script

Explain React.createElement()

Explain how to create React nodes with JSX

Define how to render JSX to DOM

Explain how to use JavaScript expressions in JSX

Explain how to use inline CSS in JSX

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.

Answer and Output:

Definitions:

1. Define JSX

JSX (JavaScript XML) is a syntax used in React to write HTML-like code inside JavaScript. It looks like HTML, but it’s actually syntactic sugar for React.createElement() calls.

1. Explain about ECMA Script

ECMAScript (often shortened to ES) is the standard on which JavaScript is based. ES6 is the 6th edition and includes features like let, const, arrow functions, destructuring, classes, etc.

1. Explain React.createElement()

This is how React builds the virtual DOM behind the scenes. JSX gets compiled into React.createElement() calls. You can also use it directly without JSX.

|  |
| --- |
| React.createElement('h1', null, 'Hello World') |

1. Explain how to create React nodes with JSX

|  |
| --- |
| const title = <h1>Hello World</h1>; |

1. Define how to render JSX to DOM

|  |
| --- |
| ReactDOM.render(<App />, document.getElementById('root')); |

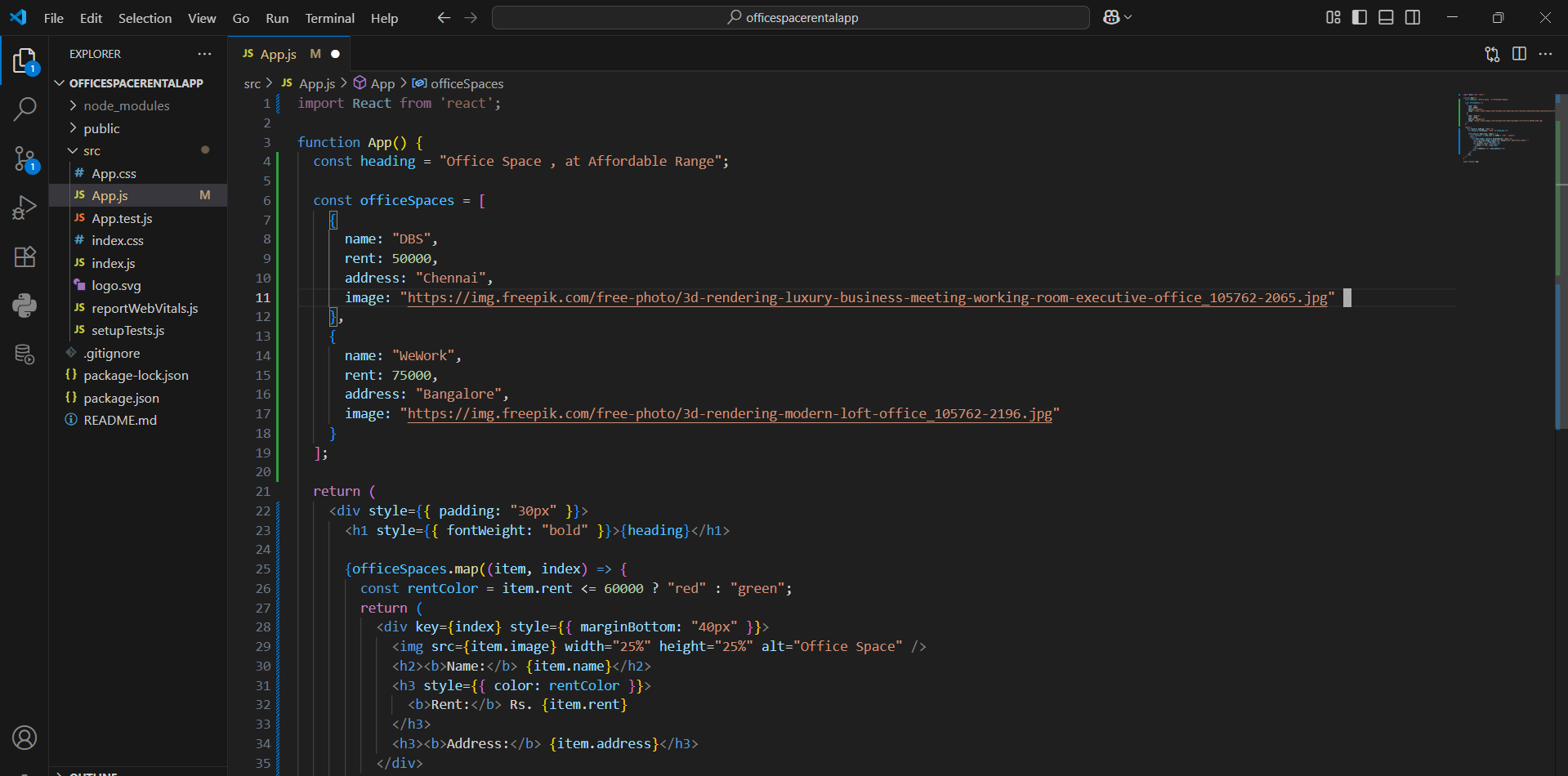
1. Explain how to use JavaScript expressions in JSX

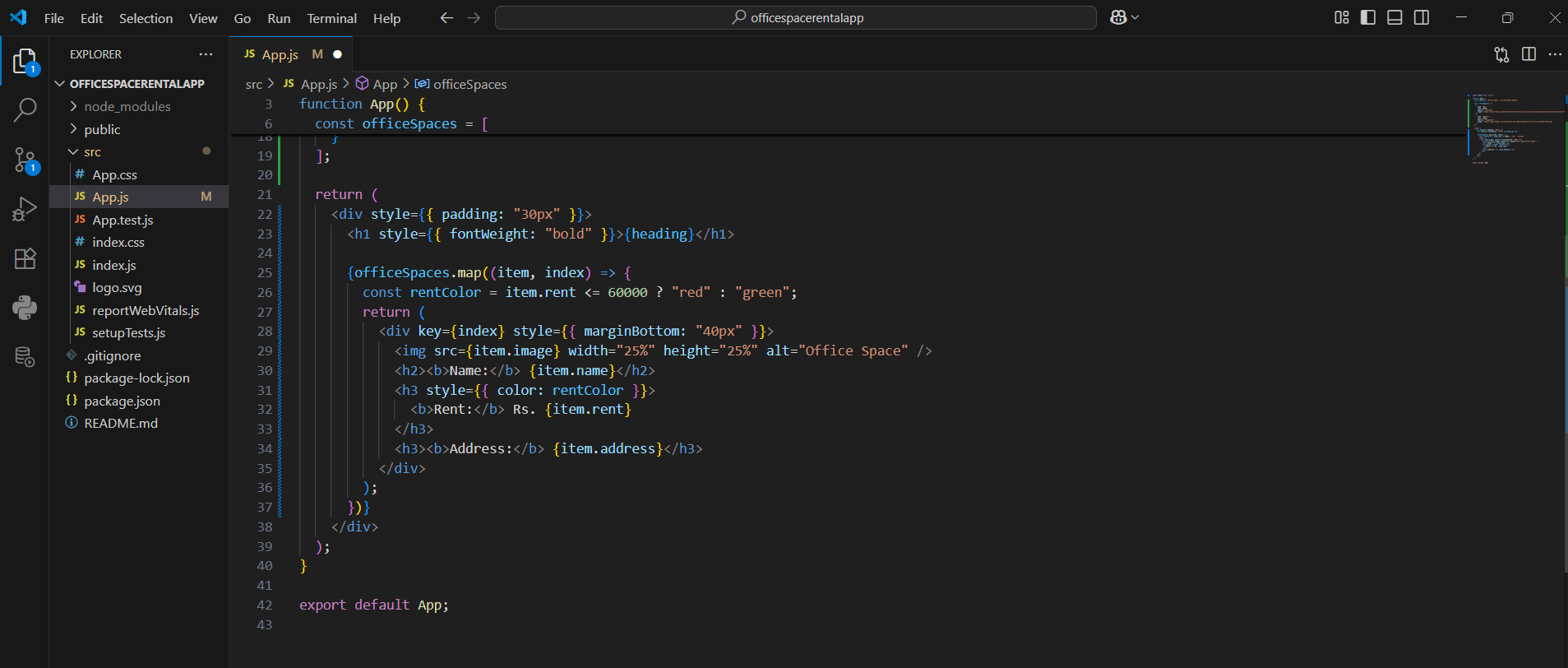
|  |
| --- |
| const name = "DBS";  <h1>Name: {name}</h1> |

1. Explain how to use inline CSS in JSX

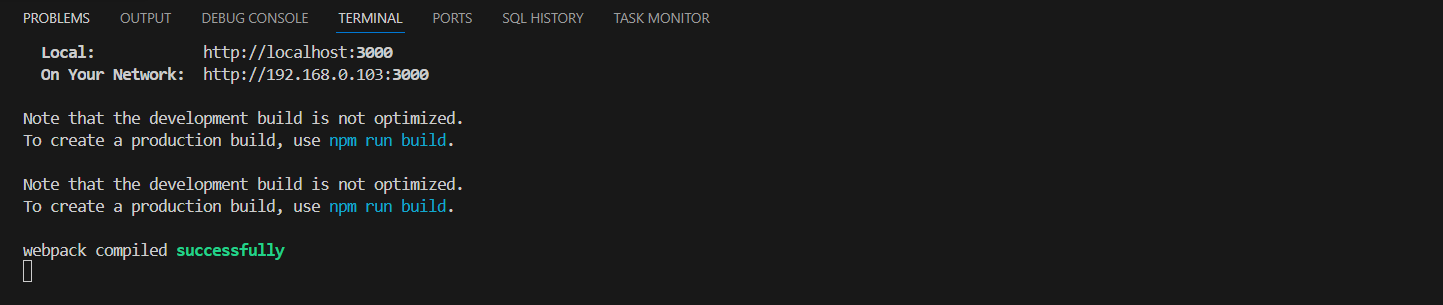
|  |
| --- |
| <h3 style={{ color: "red" }}>Rent: 50000</h3> |

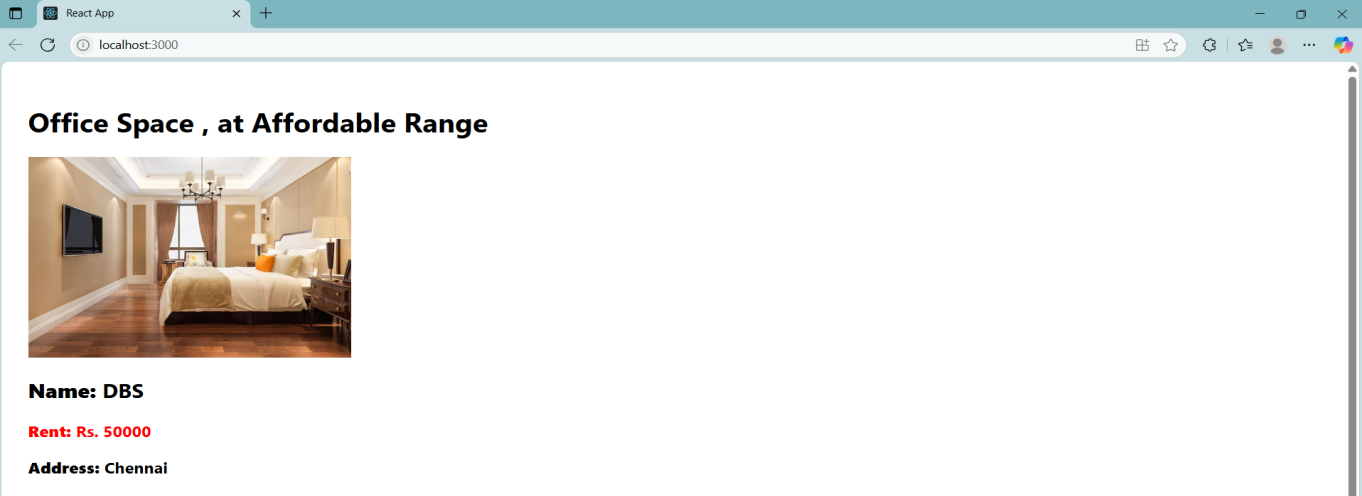
CODE:

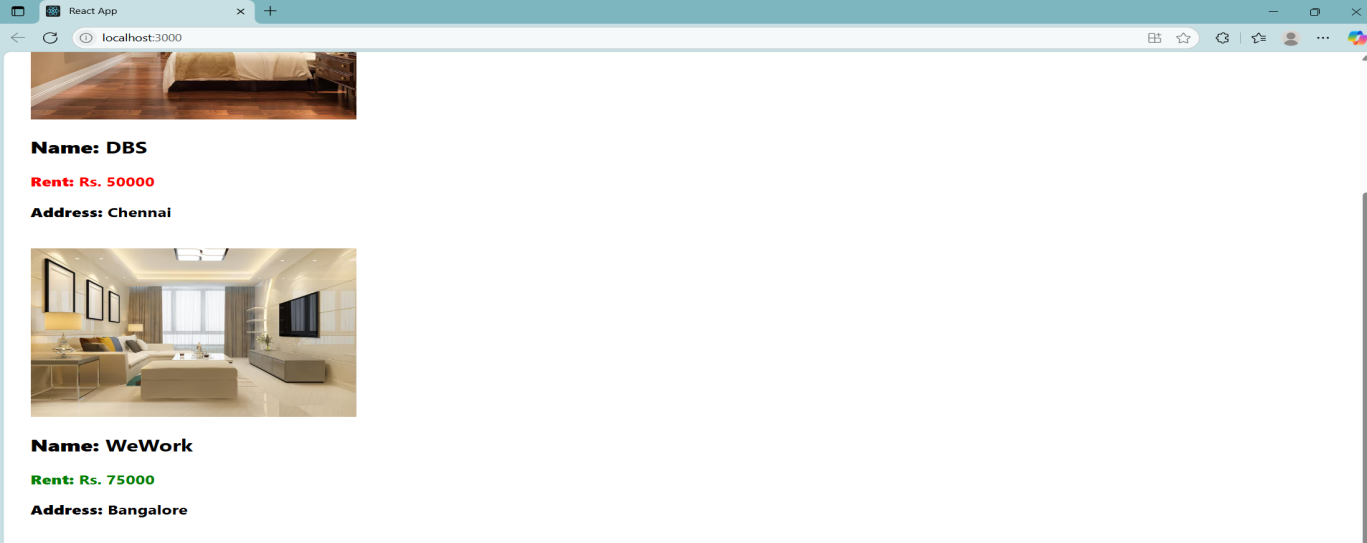




OUTPUT:







**Question 11**

Objectives

Explain React events

Explain about event handlers

Define Synthetic event

Identify React event naming convention

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.

a.To increment the value

b.Say Hello followed by a static message.

Answer and Output:

Definitions:

1. Explain React events

React handles user interactions (like clicks, key presses, form submissions) through its own event system that wraps native browser events for better performance and consistency.

1. Explain about event handlers

An event handler is a function that gets called when an event happens. Example:

|  |
| --- |
| <button onClick={handleClick}>Click me</button> |

1. Define Synthetic event

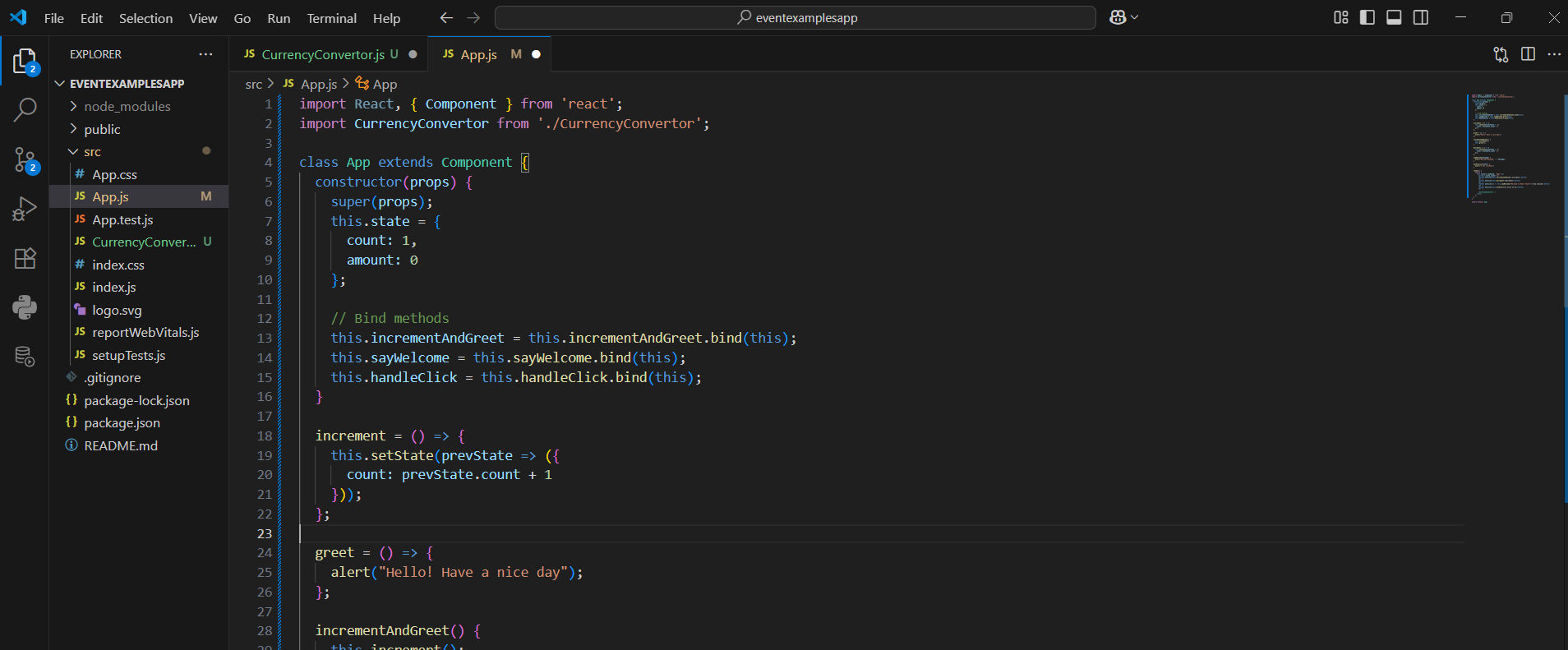
React wraps native events in a SyntheticEvent wrapper to ensure it works the same in all browsers. It behaves like a regular event but has consistent behavior across platforms.

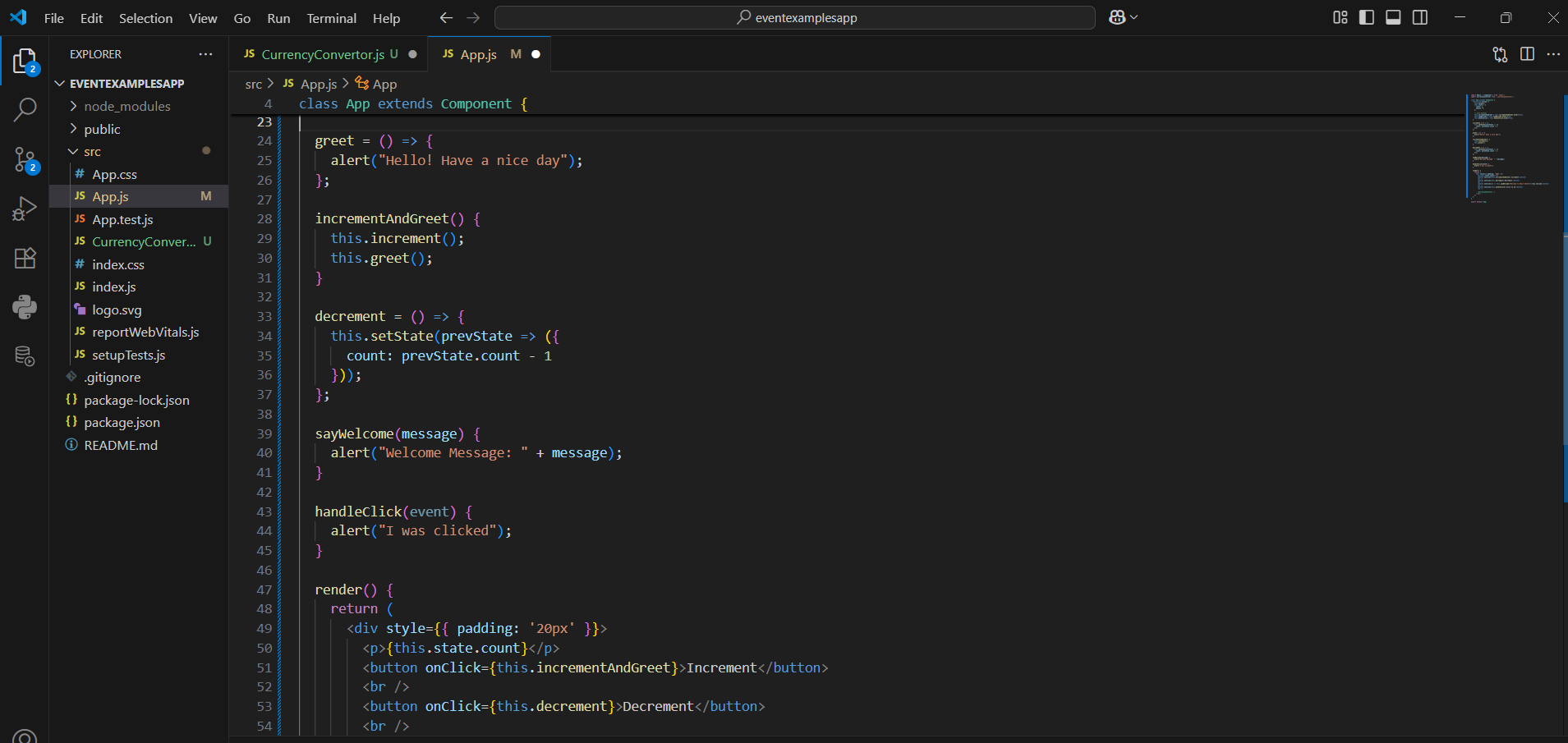
1. Identify React event naming convention

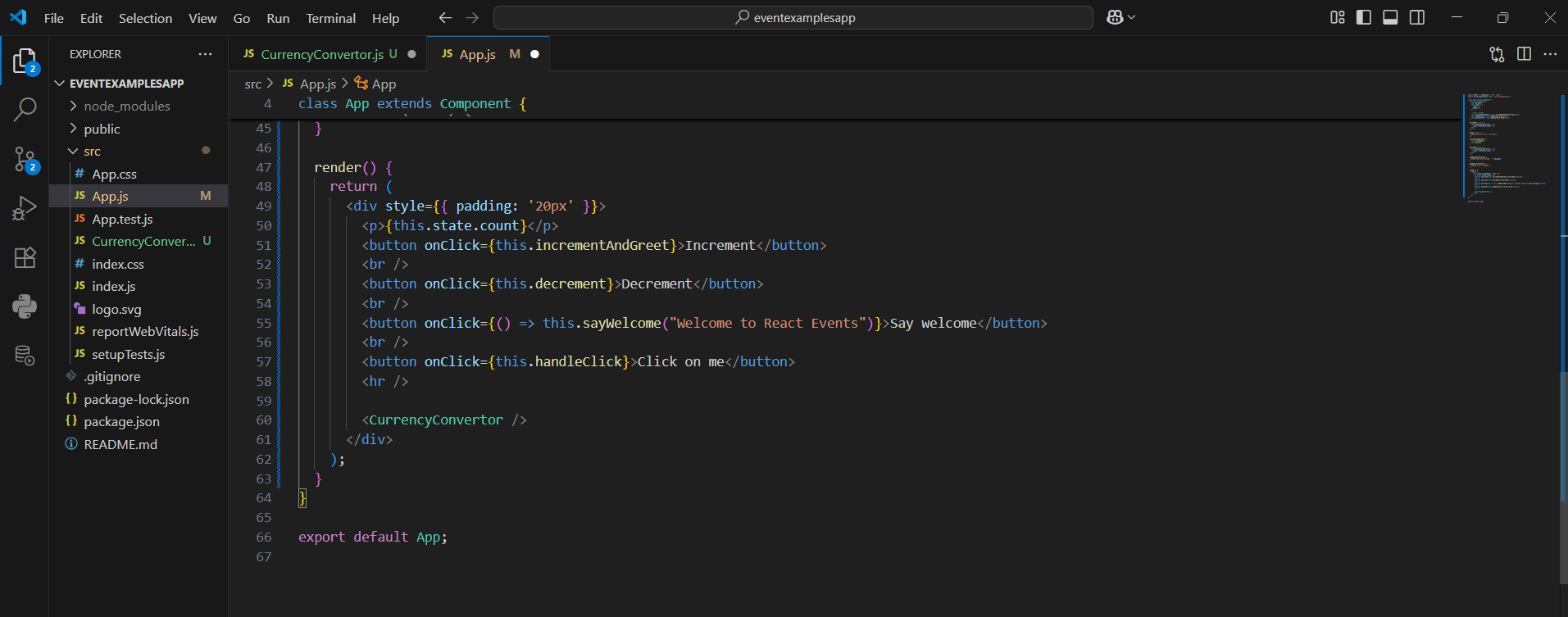
React uses camelCase for event names:

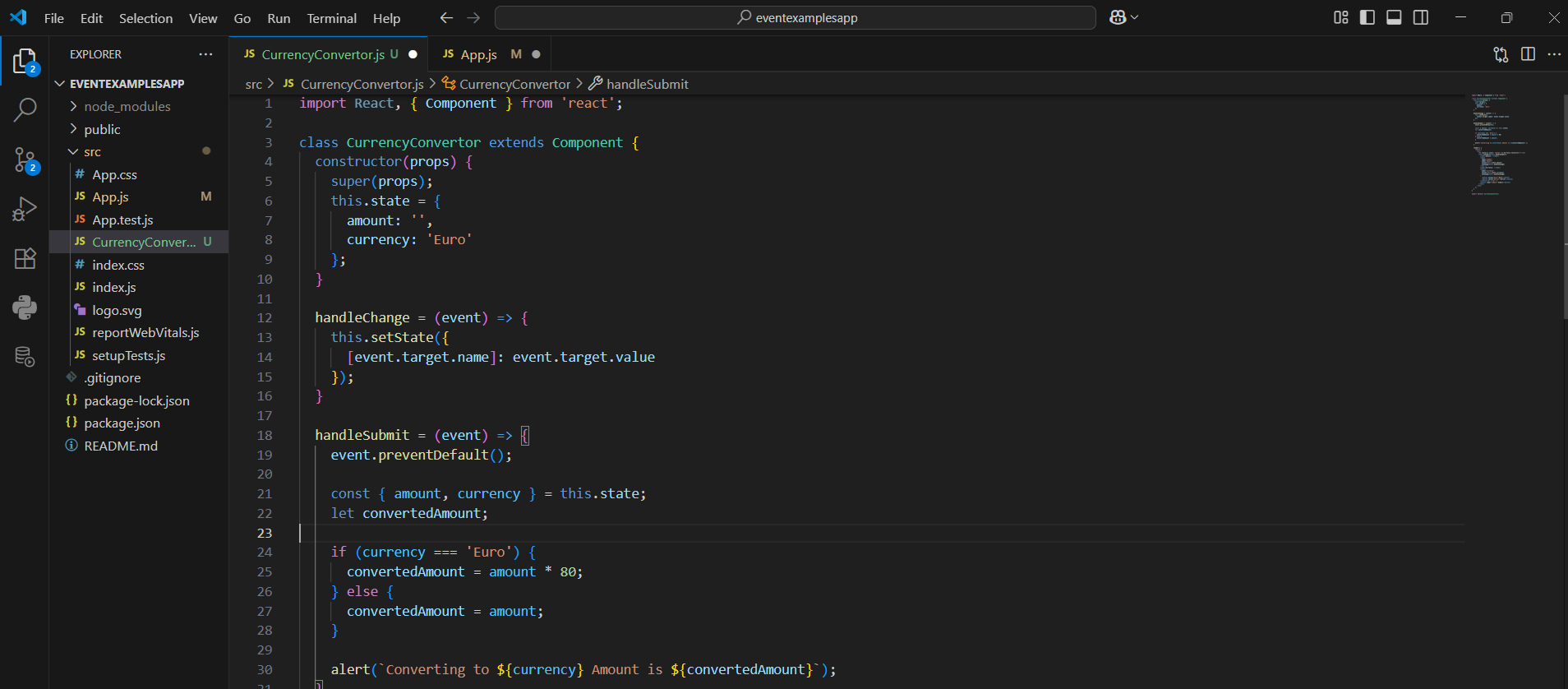
onClick ( correct) onclick(wrong)

CODE:



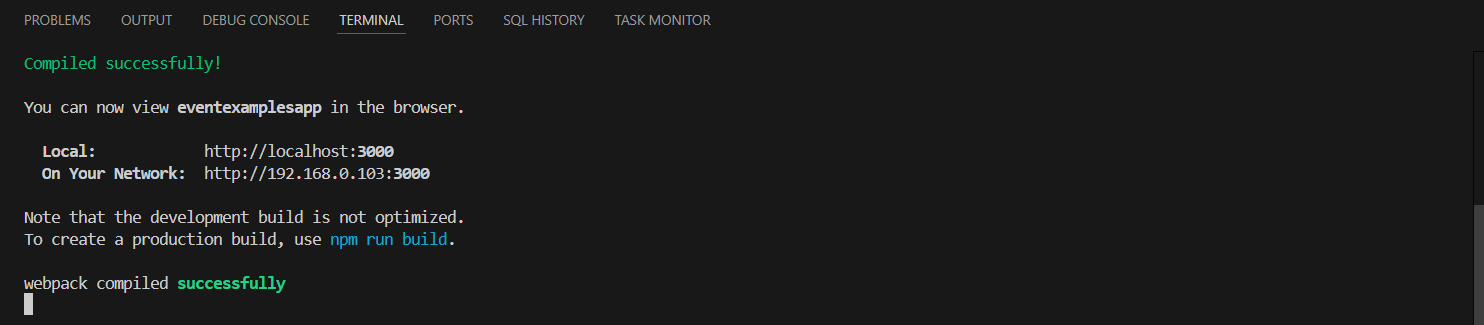


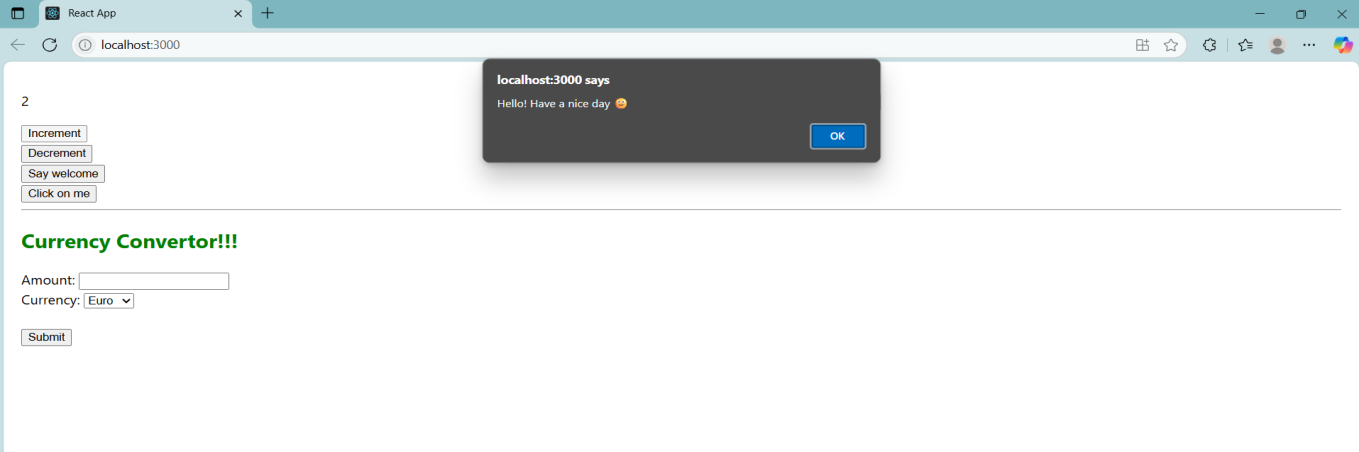


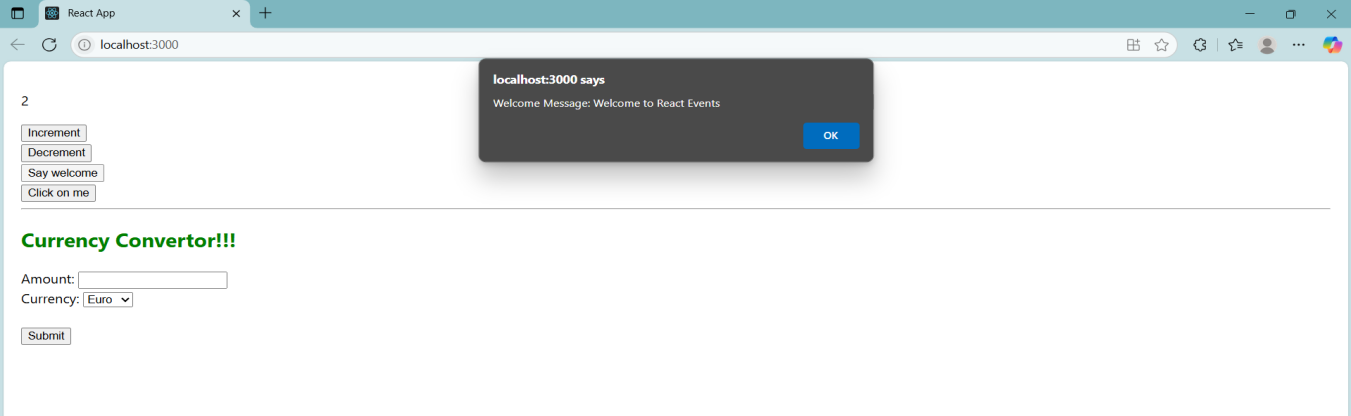


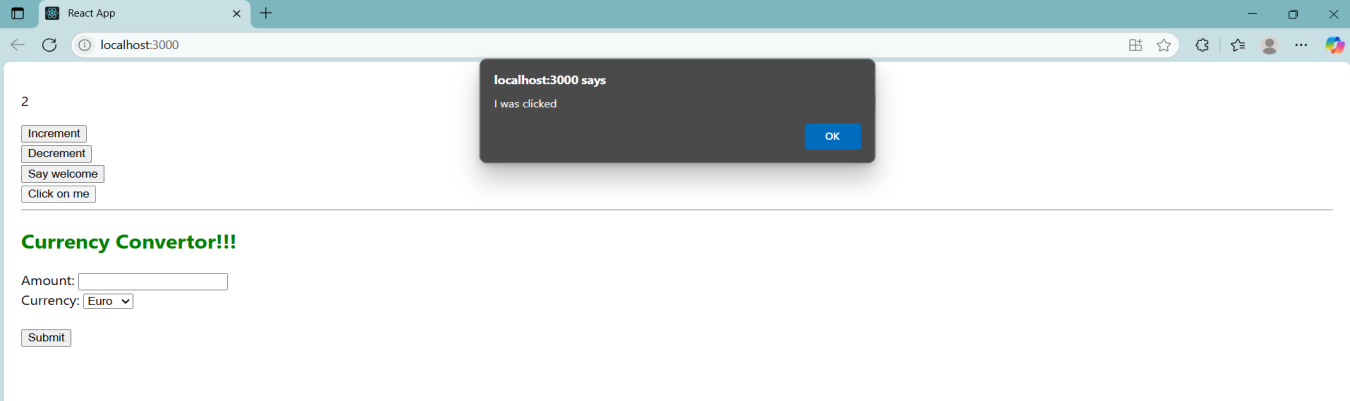


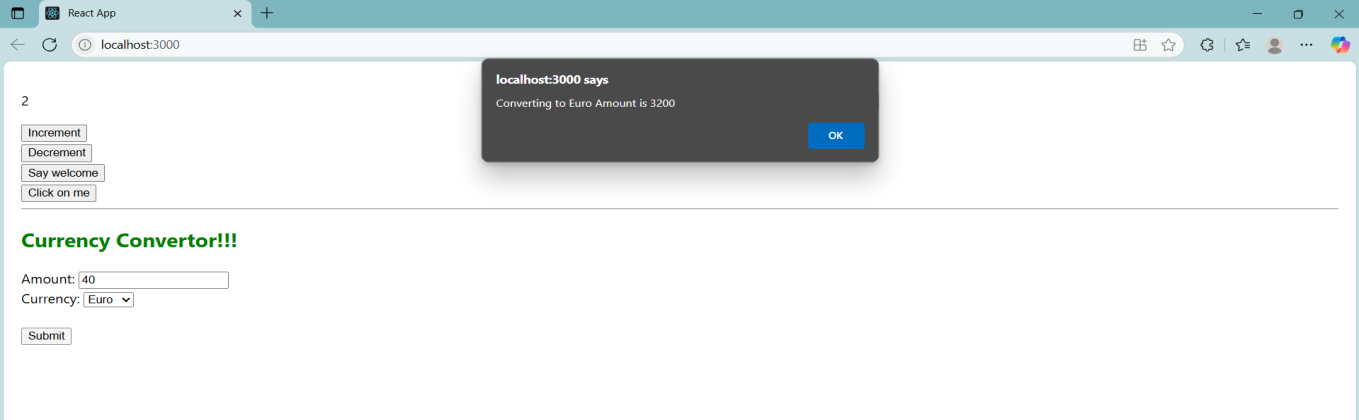
OUTPUT:











**Question 12**

Objectives

Explain about conditional rendering in React

Define element variables

Explain how to prevent components from rendering

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.

Answer and Output:

Definitions:

1. Explain about conditional rendering in React

React lets you show different UI elements based on conditions (like if, else). For example:

|  |
| --- |
| {isLoggedIn ? <UserPage /> : <GuestPage />} |

1. Define element variables

We can store JSX in variables before returning it. Example:

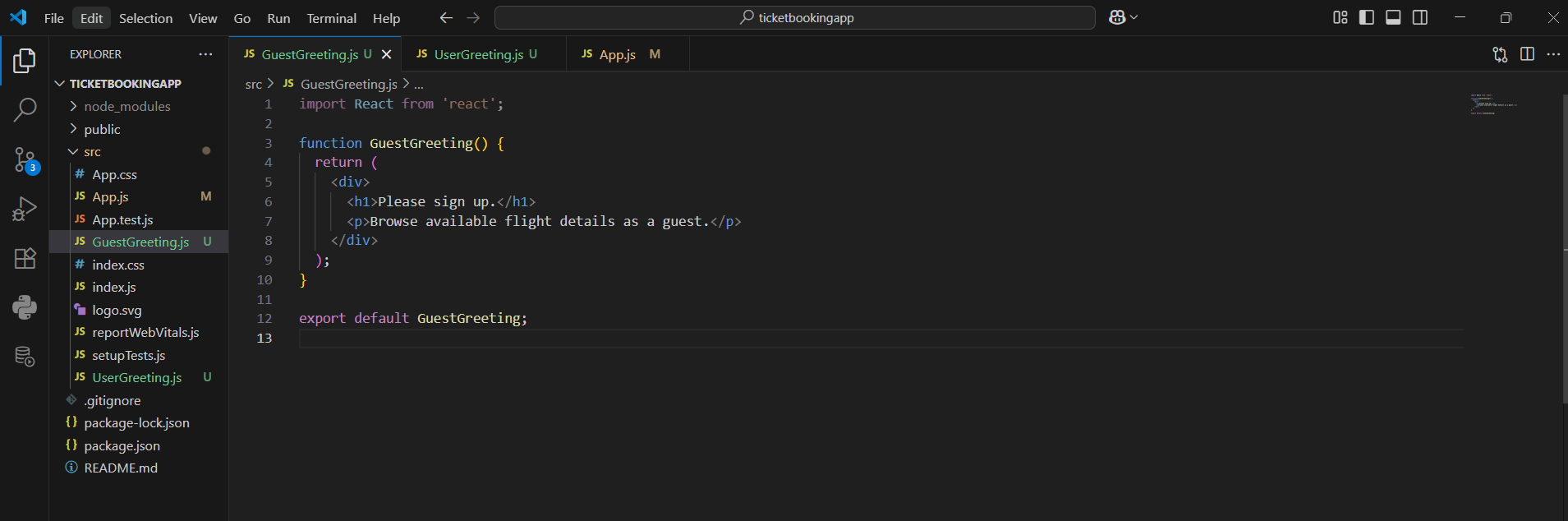
|  |
| --- |
| let button;  if (isLoggedIn) {  button = <LogoutButton />;  } else {  button = <LoginButton />;  } |

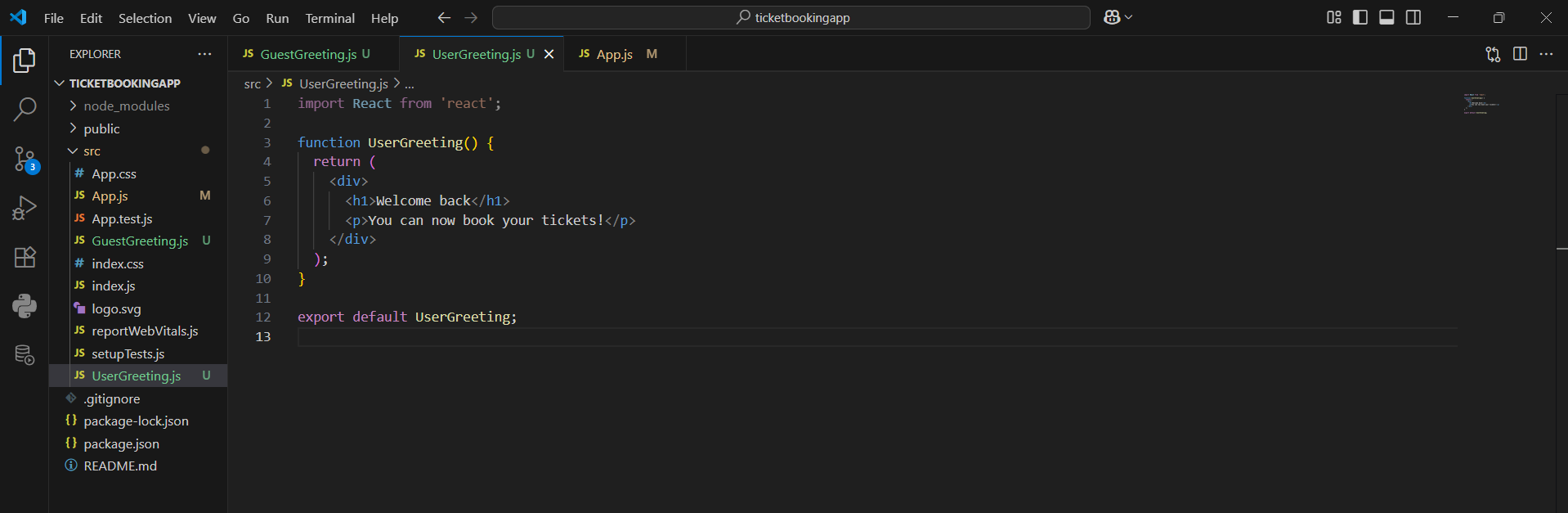
1. Explain how to prevent components from rendering

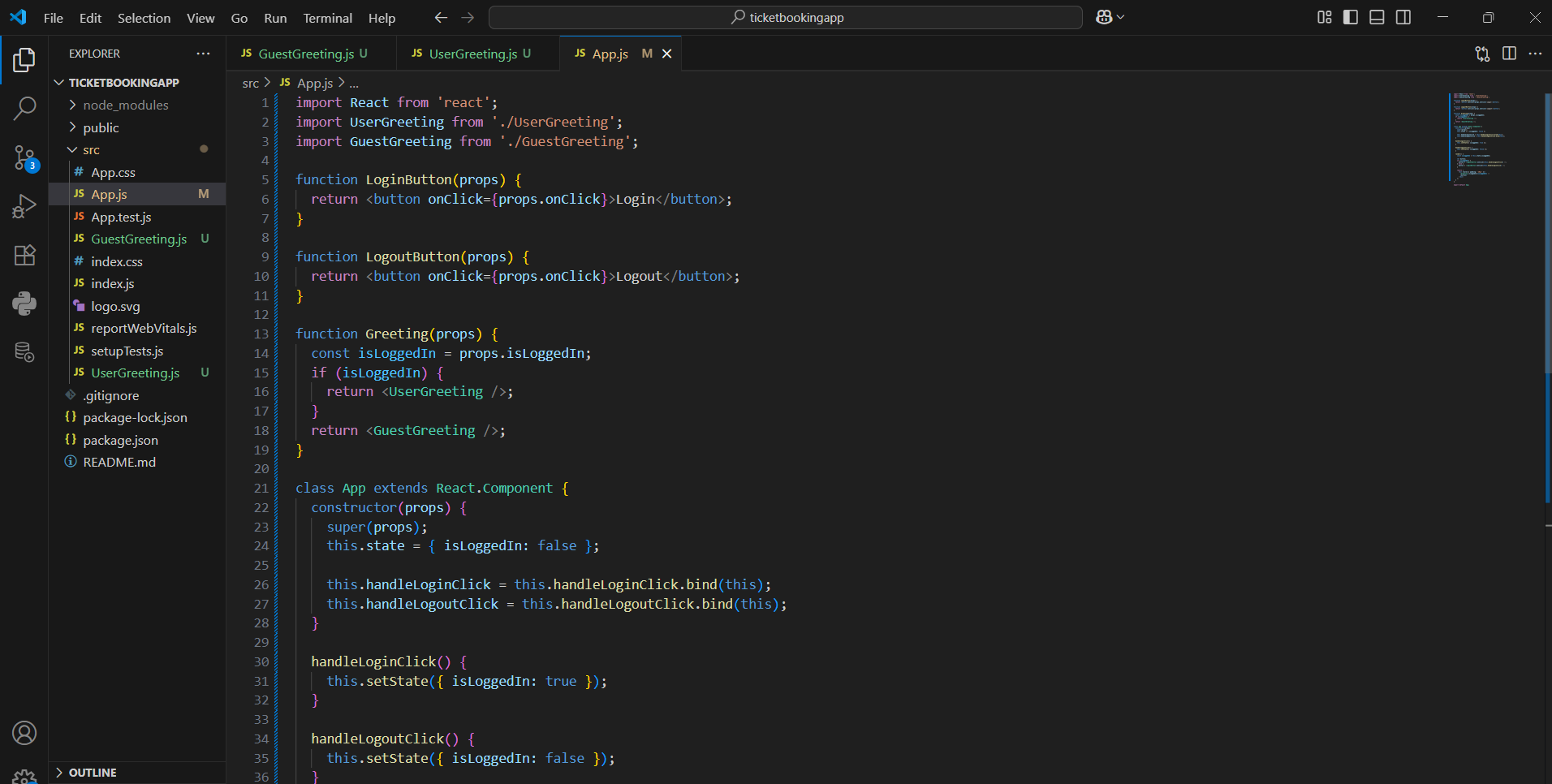
You can return null from a component if you don’t want it to render anything:

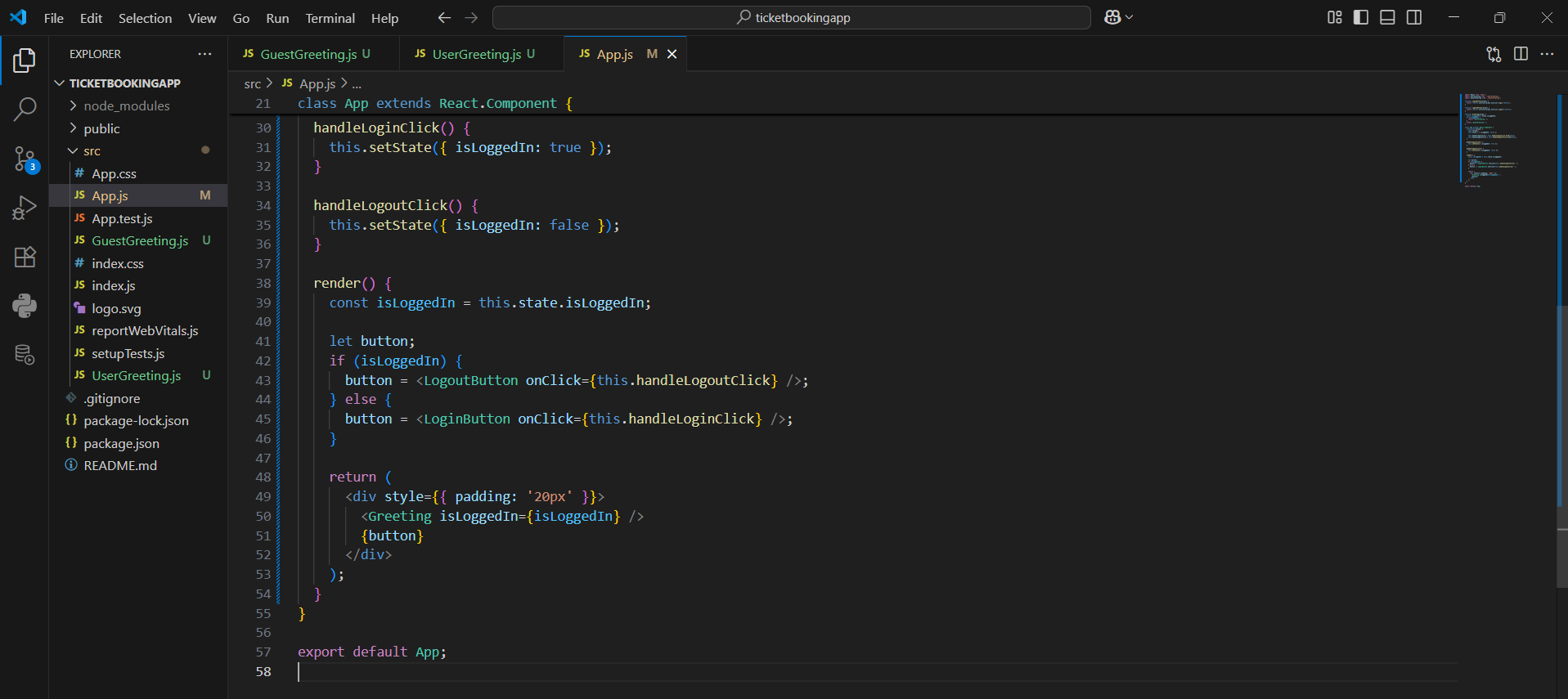
|  |
| --- |
| if (!shouldRender) {  return null;  } |

CODE:

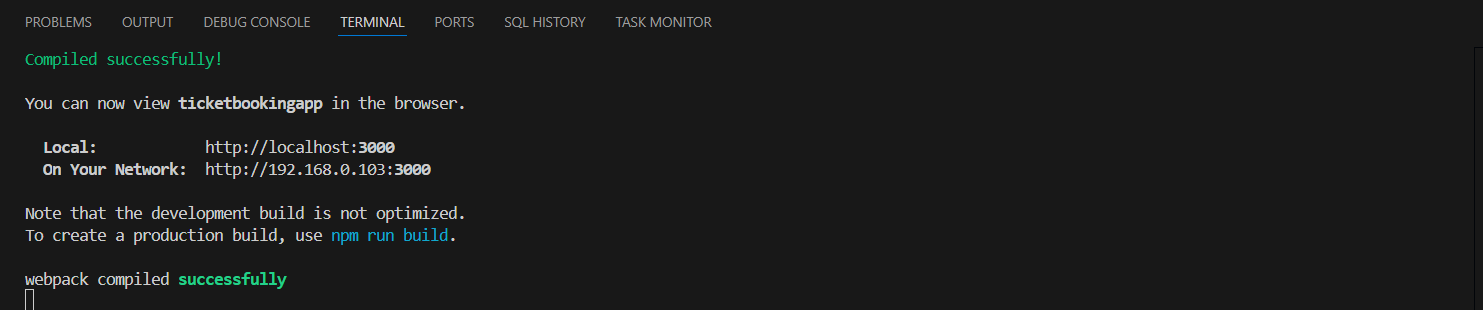


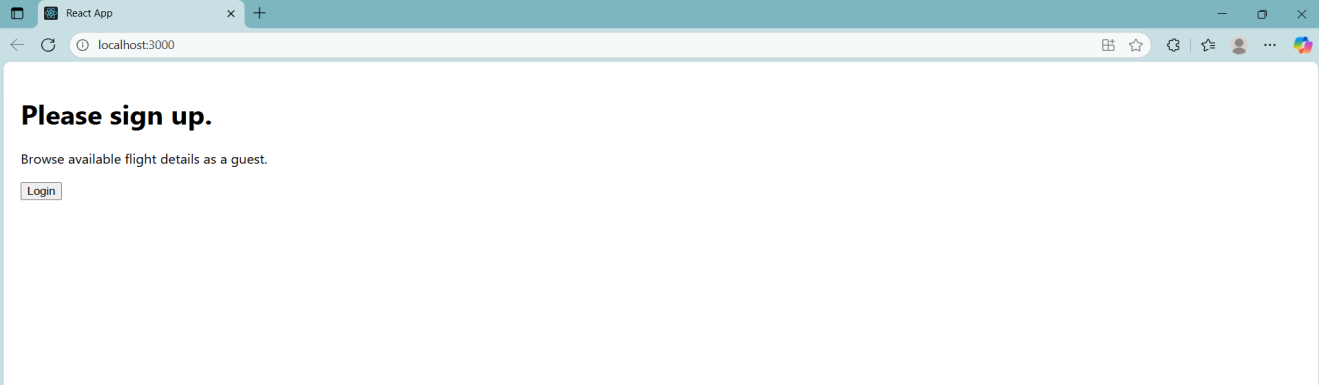


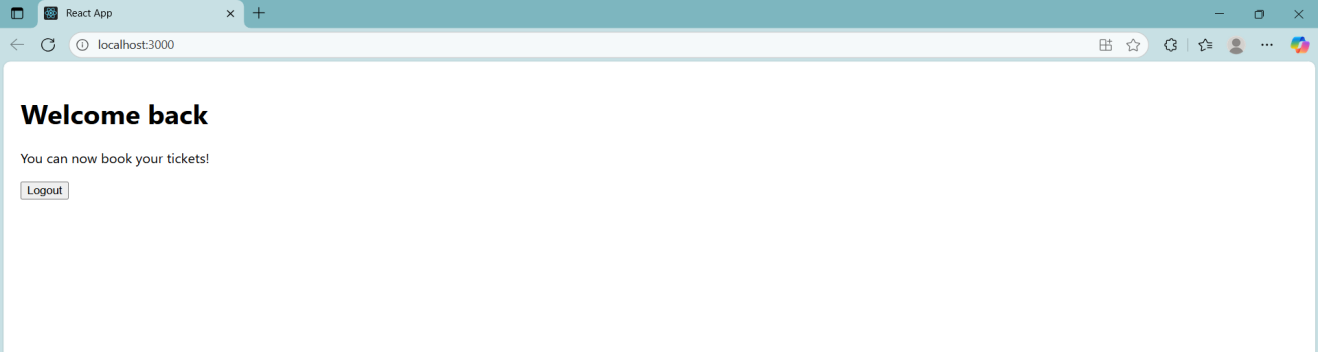




OUTPUT:







**Question 13**

Objectives

Explain various ways of conditional rendering

Explain how to render multiple components

Define list component

Explain about keys in React applications

Explain how to extract components with keys

Explain React Map, map() function

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.

Definitions:

1. Explain various ways of conditional rendering

React allows you to render different components or elements based on certain conditions like booleans, user input, etc. You can use if, ternary operators (? :), or logical && to conditionally show elements.

1. Explain how to render multiple components

You can return multiple components from one component using fragments (<> </>) or by placing them inside a <div>. This helps organize the UI into smaller, reusable parts.

1. Define list component

In React, when you need to display repeated UI (like a list of books), you use the map() function. It loops through an array and returns a JSX element for each item.

1. Explain about keys in React applications

When rendering lists, React needs a unique "key" prop to identify each element. This helps React optimize and track elements when updating the UI.

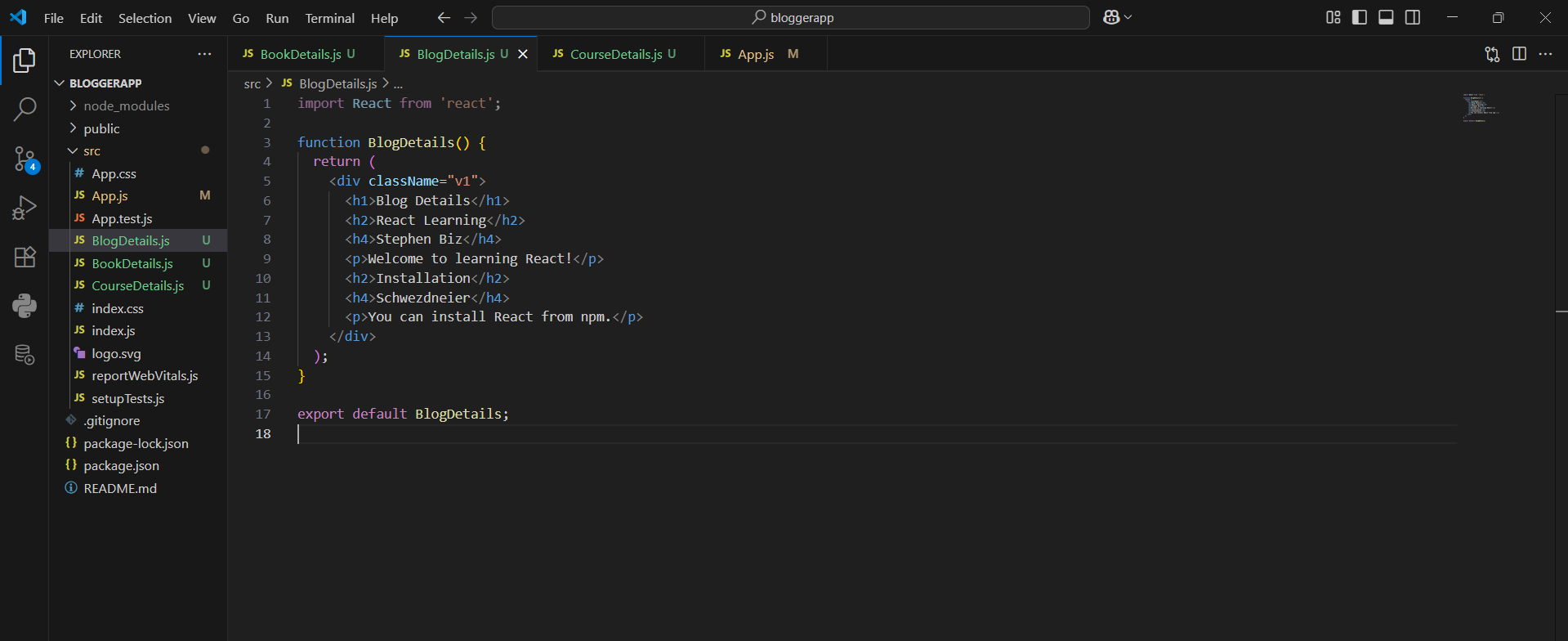
1. Explain how to extract components with keys

You can create separate components (like <BookItem />) and pass key to them when rendering in a list. This makes your code cleaner and each item can manage its own rendering logic.

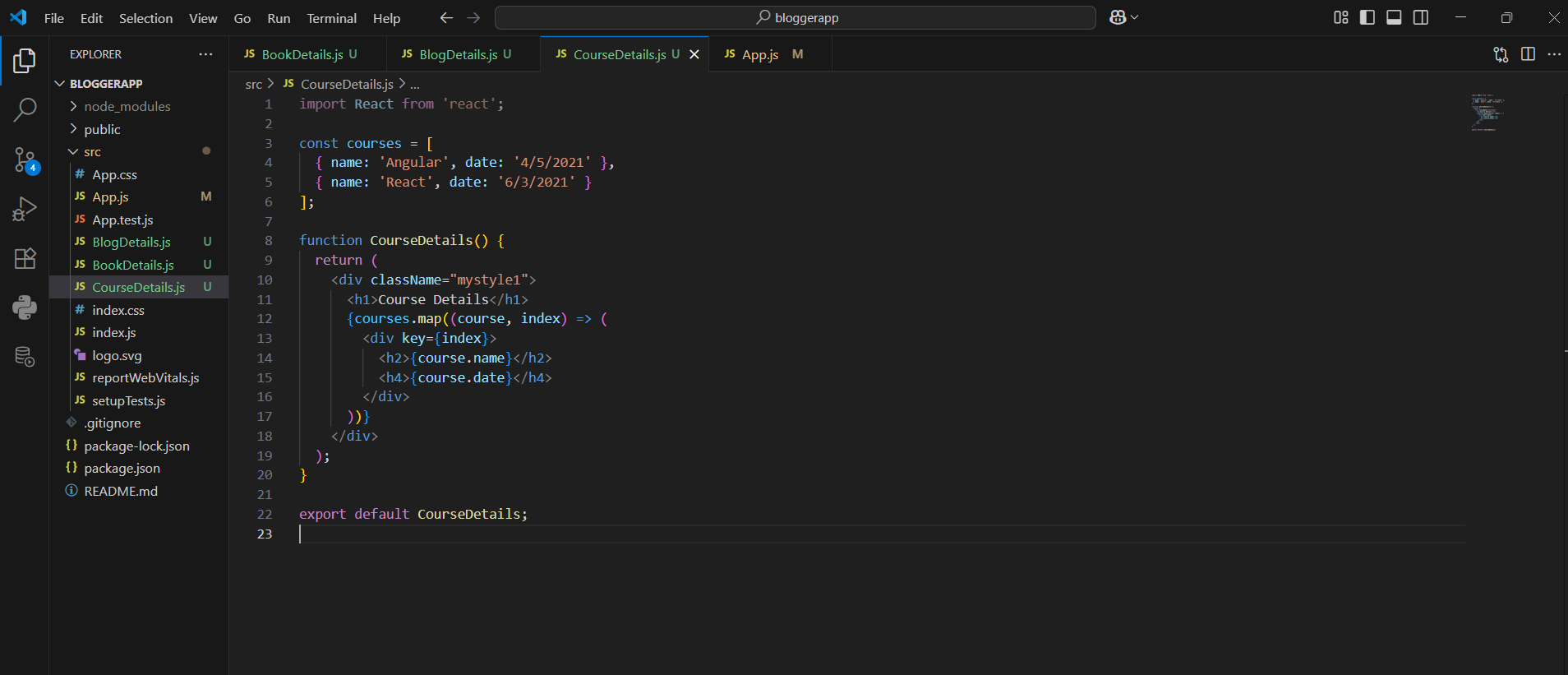
1. Explain React Map, map() function

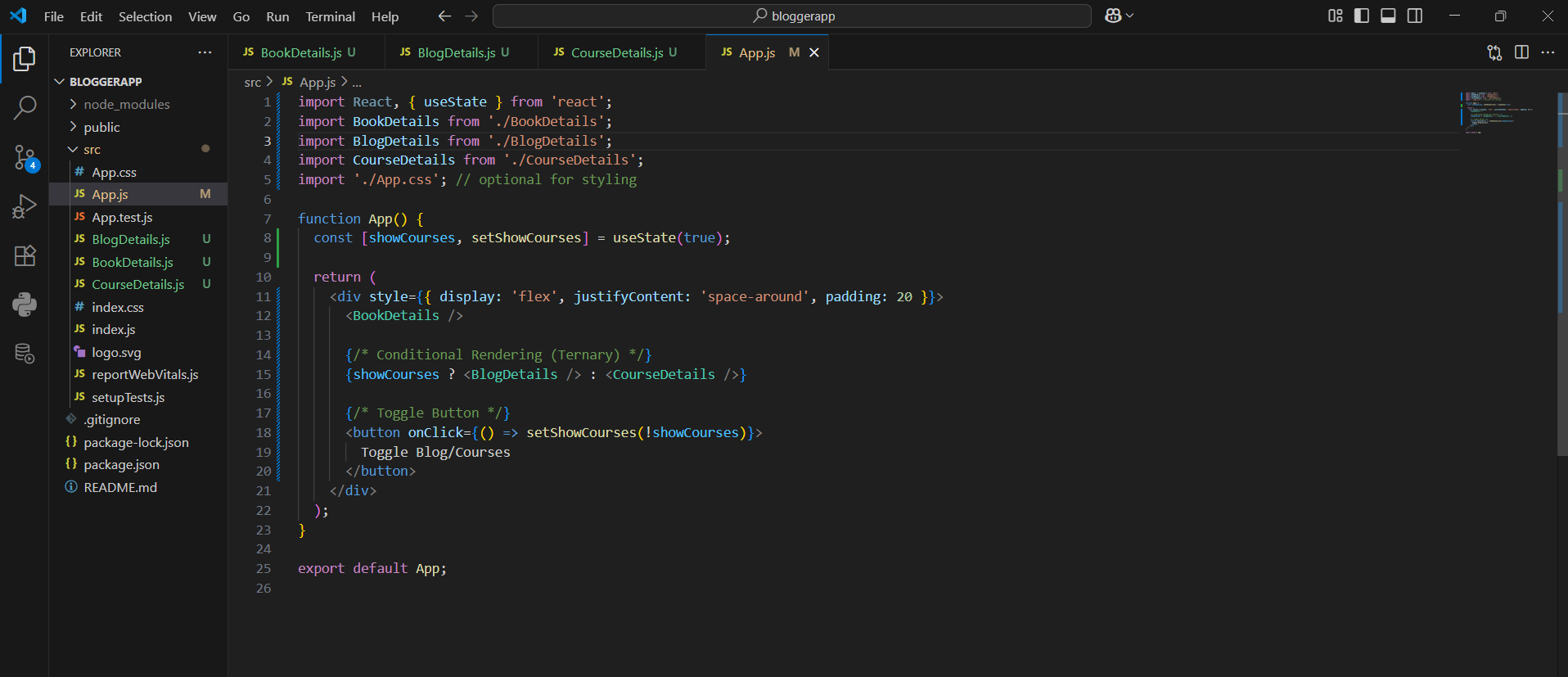
The map() function in React is used to loop over arrays and return JSX for each item. It's the most common way to dynamically render lists of content.

CODE:

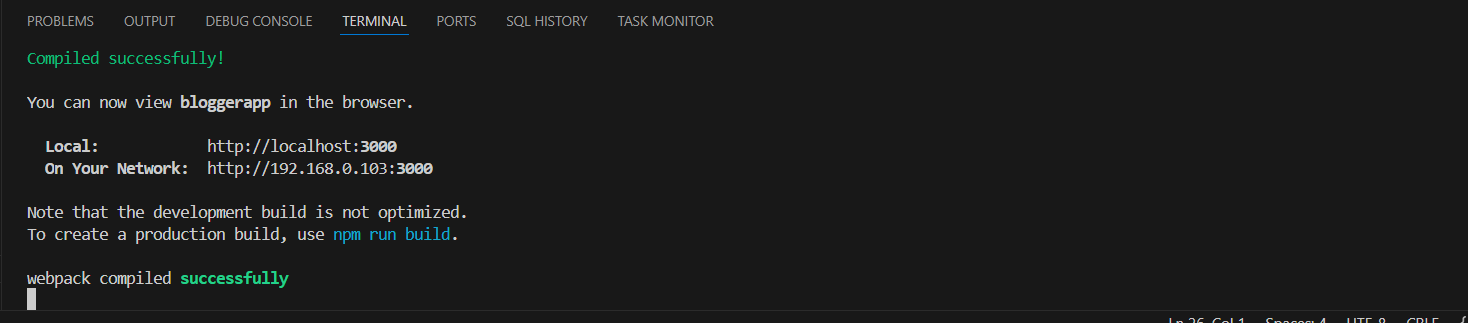


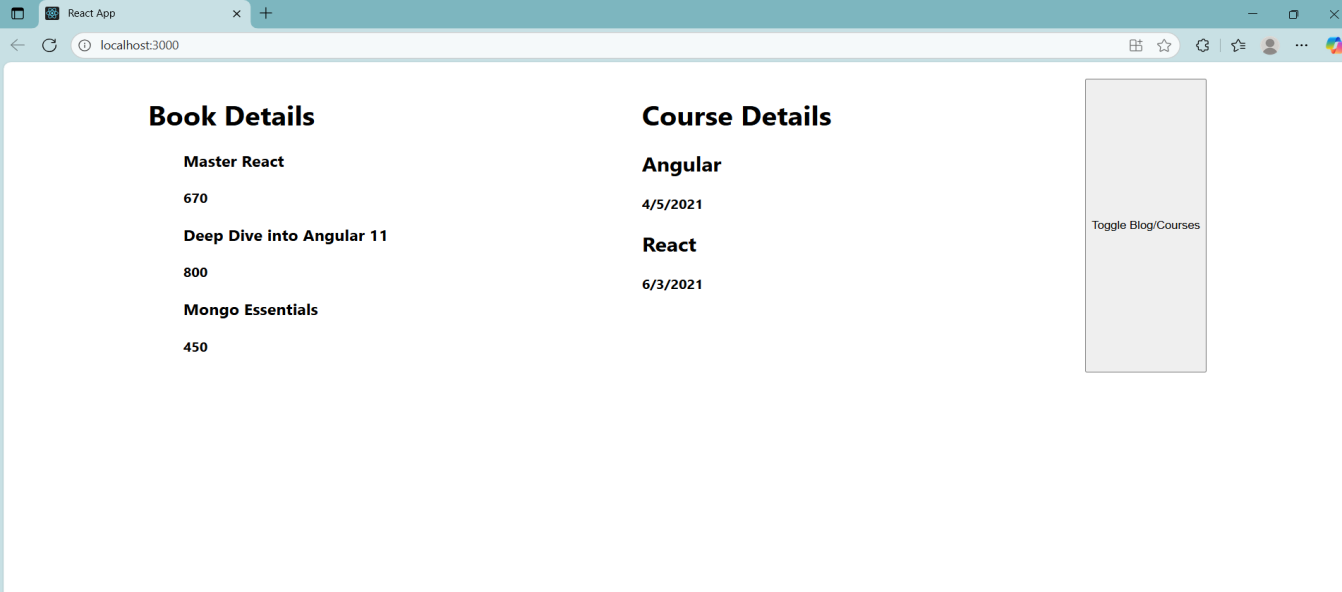


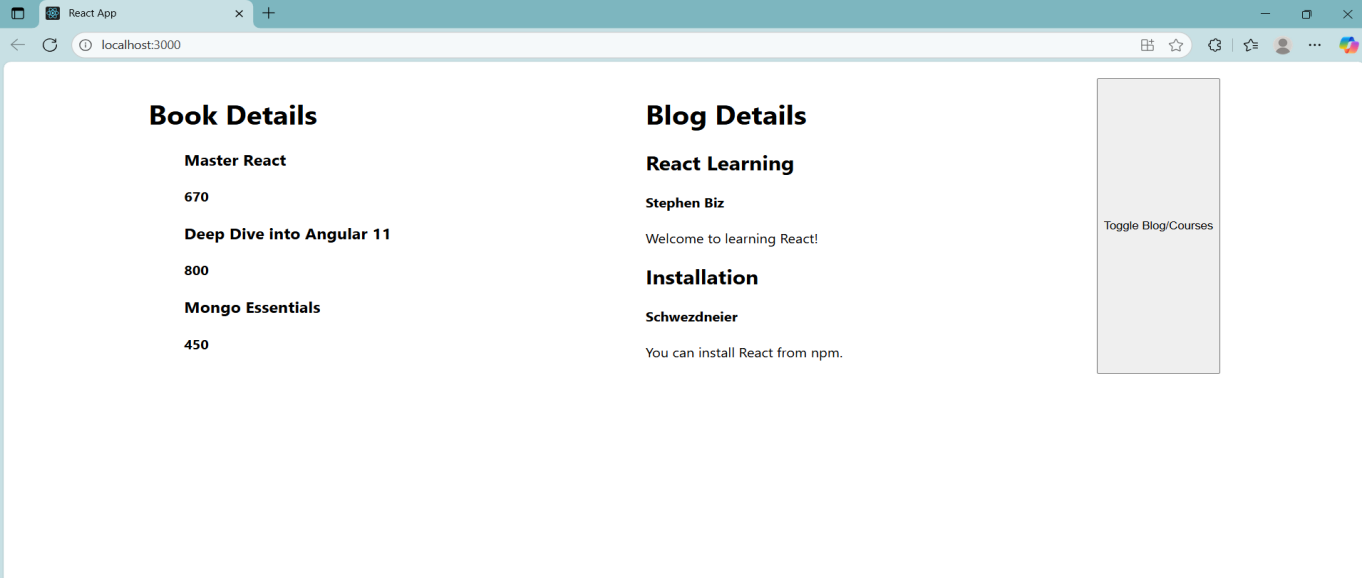




OUTPUT:







**ADDITIONAL HANDS-ON QUESTIONS**

**Question 14**

Objectives

Explain the need and Benefits of React Context API

Working with createContext()

List the types of Router Components

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

Create a context to be used by child components

Create a provider and consumer of the context

Prerequisites

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

Node.js

NPM

Visual Studio Code

Notes

Estimated time to complete this lab: 30 minutes.

Developers of Apps Centric Solutions have created an employee management application which supports light and dark themes for the buttons. The current solution uses the react state and props to provide the theme name to be used from App component to Employee List component and from there to Employee Card component. Quality assurance team analyzed the solutions and found the technique being used to be a substandard one. React architect suggested to use the react context API to share the theme name with nested child components instead of passing them down using props from the parent component.

You are assigned the task of converting the application form props only to React Context API.

Answer and Output:

Definitions:

1. Explain the need and Benefits of React Context API

When building large React apps, you often have to pass data like a user's info or app theme through many layers of components — from parent to child, to grandchild, and so on. This is called "prop drilling," and it becomes messy and hard to manage. The React Context API solves this by letting you share data across your app without passing props manually at every level.

Benefits:

1. Avoids prop drilling
2. Makes state management easier for global data (like user login status, theme, language, etc.)
3. Keeps your code cleaner and more maintainable
4. Good alternative to external state management tools like Redux for simpler use cases
5. Working with createContext()

The createContext() function is how you start using the Context API in React. It creates a Context object that you can use to provide and consume data anywhere in your component tree.

|  |
| --- |
| import React, { createContext, useState } from 'react';  const MyContext = createContext();  function MyProvider({ children }) {  const [value, setValue] = useState("Hello Context!");    return (  <MyContext.Provider value={{ value, setValue }}>  {children}  </MyContext.Provider>  );  }  import { useContext } from 'react';  function MyChildComponent() {  const { value } = useContext(MyContext);  return <p>{value}</p>;  } |

1. List the types of Router Components

a. BrowserRouter

Think of BrowserRouter as the brain of routing in your React app. It uses the browser's normal URL path (like example.com/home) to decide what to show. It keeps the UI in sync with the URL using the HTML5 History API. You wrap your whole app in it, and then use Routes inside.

b. Routes

The Routes component is like a container that checks all your route paths and figures out which component should show up. It replaces the older Switch component and must be used inside BrowserRouter.

c. Route

Route is where you define the actual path and what to show when the user goes to that path. It has a path (the URL) and an element (the component to show).

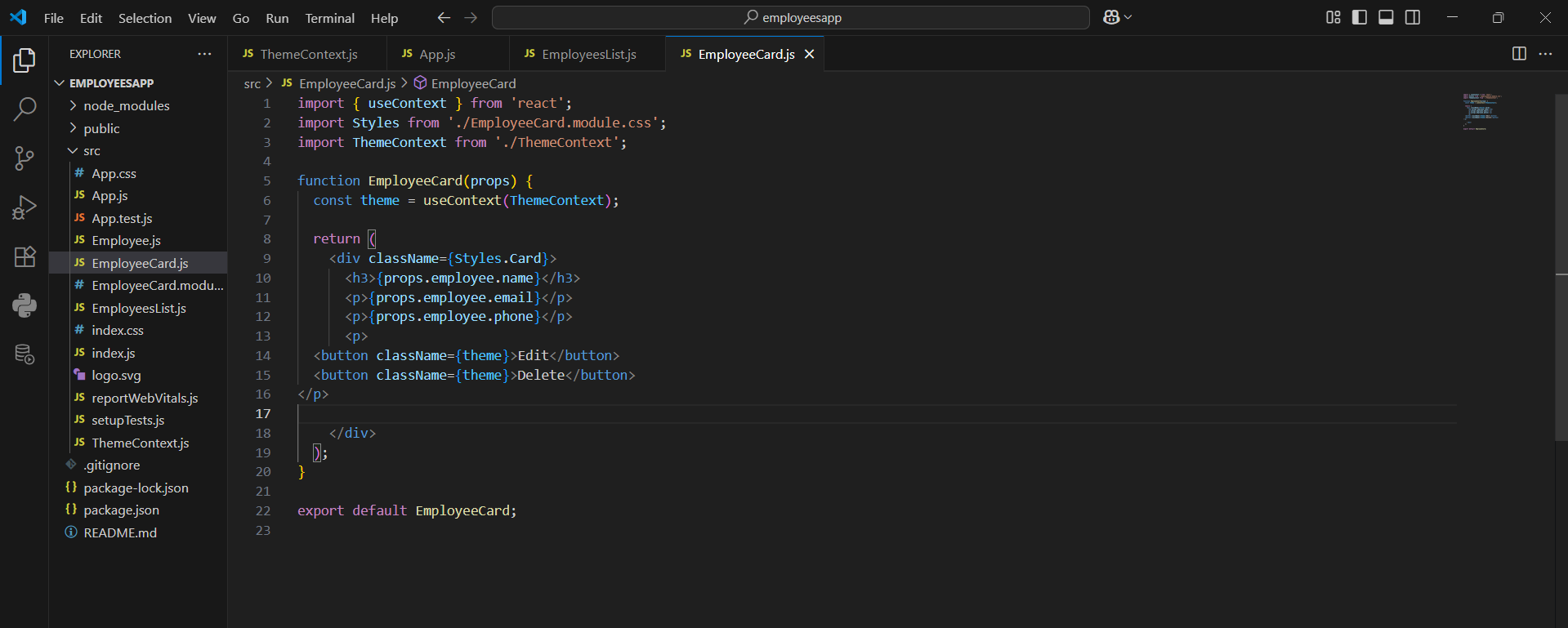
d. Link

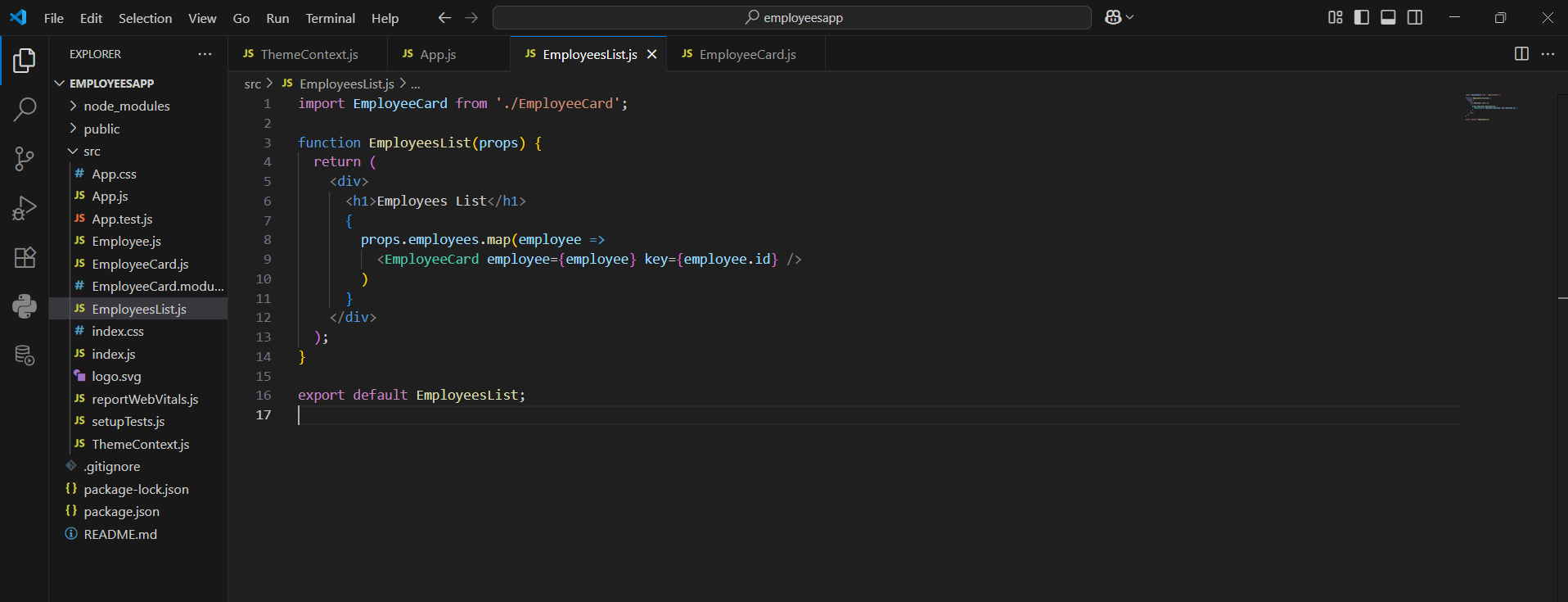
Instead of using <a> tags that reload the page, React Router gives you the Link component. Clicking it changes the URL without reloading, making your app feel fast and smooth.

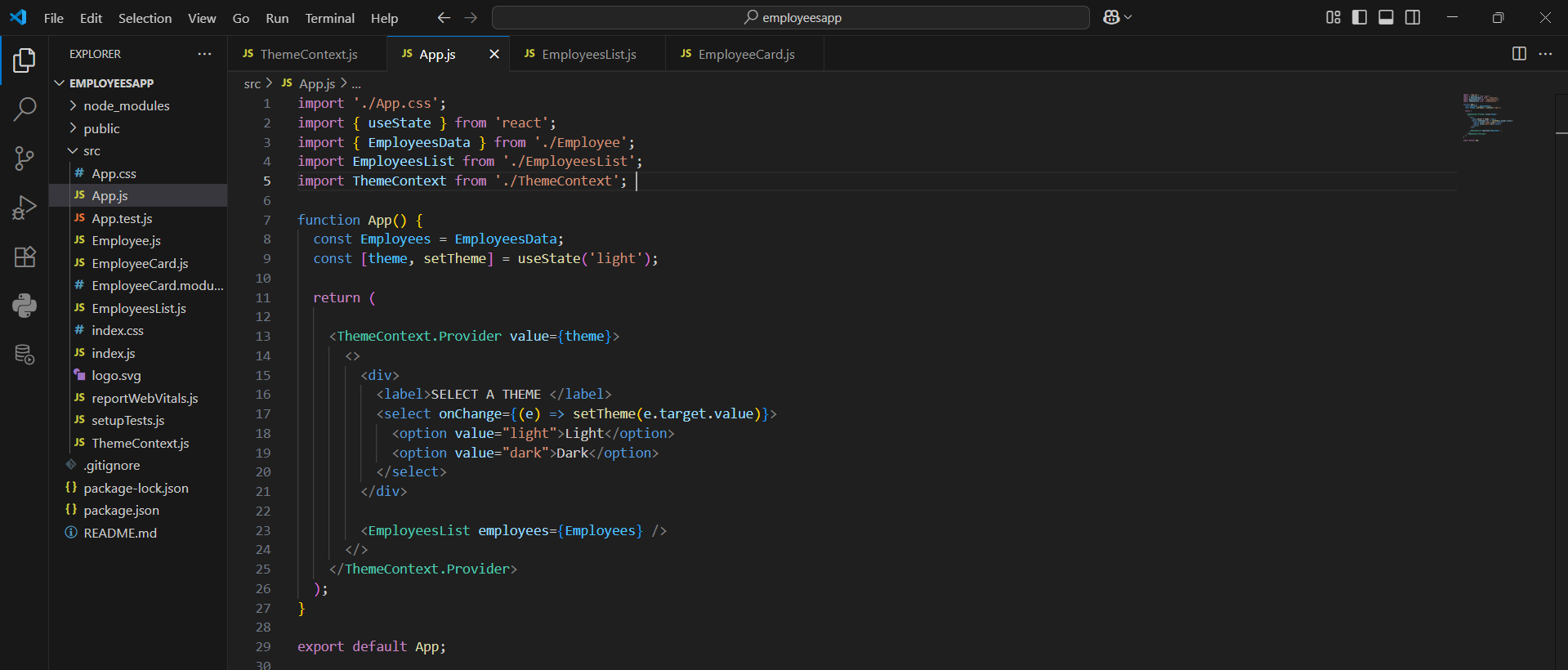
e. Navigate

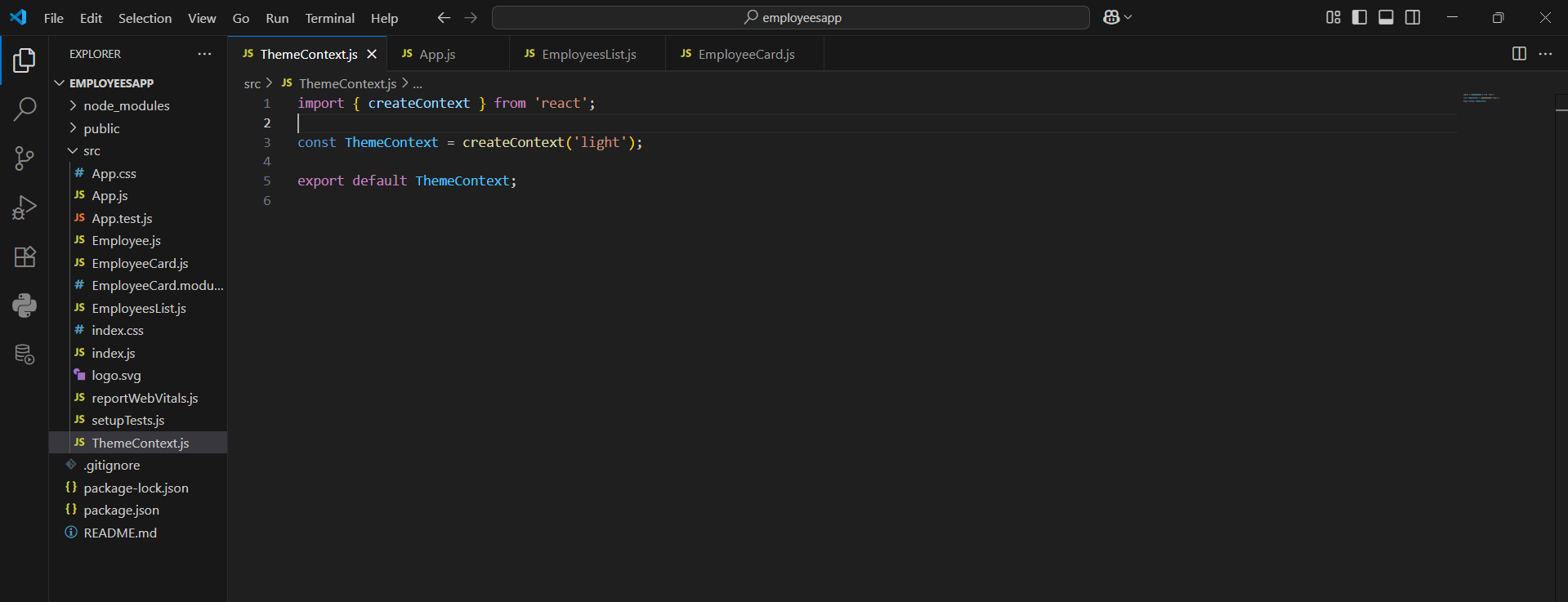
Navigate is like a programmatic redirect. You use it to send users to another route based on some condition, like after logging in or submitting a form.

CODE:

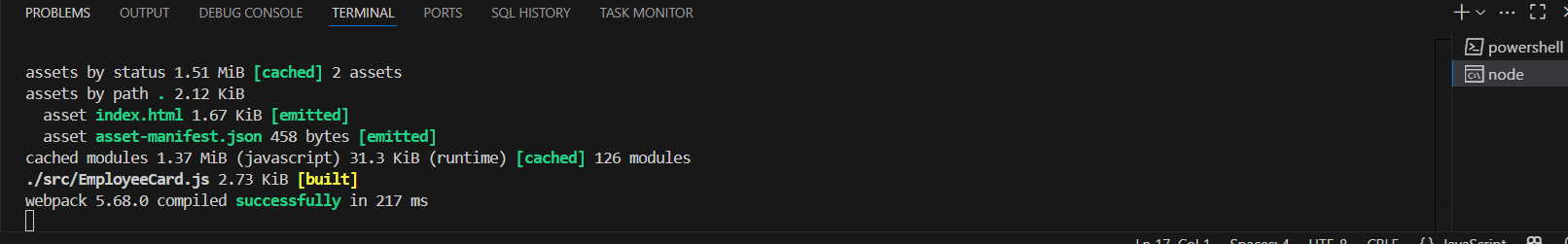


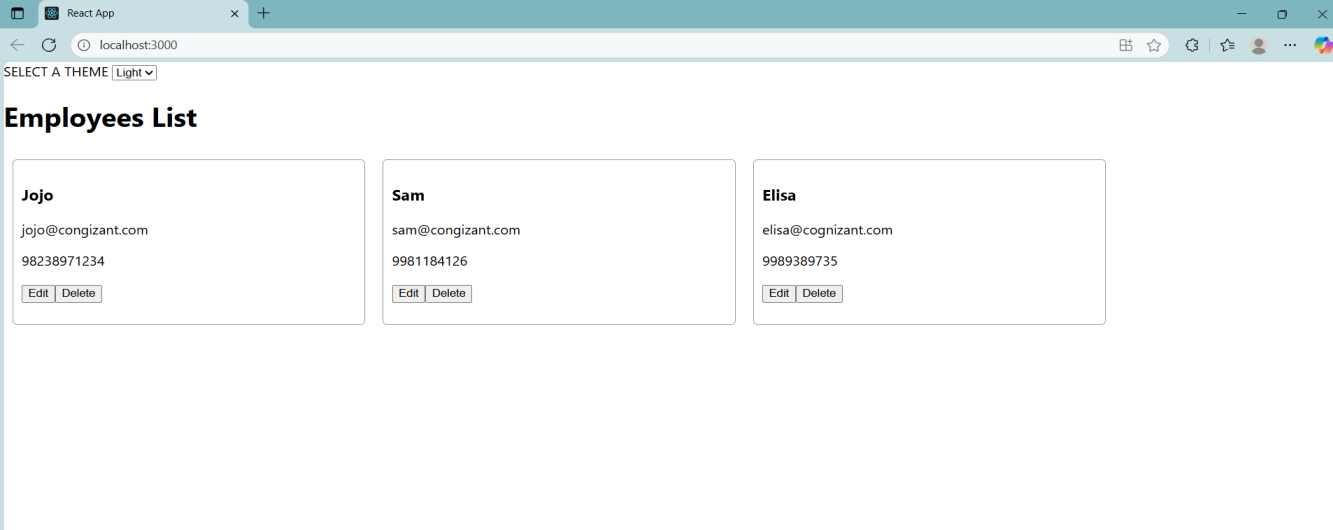






OUTPUT:





**Question 15**

Objectives

Explain about React forms

Define controlled components

Explain about various input controls

Explain about handling forms

Explain about submitting forms

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

Implement React forms

Use various input controls like textbox, button and textarea

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 “ticketraisingapp” which will help to raise a complaint and get it resolved.

Create a component named “ComplaintRegister” with a form containing a textbox to enter the employee name and a textarea to enter the complaint. Use “handleSubmit” event of the button to submit the complaint and generate a Reference number for further follow ups in the alert box.

Answer and Output:

Definitions:

1. Explain about React forms

In normal HTML, a form works using tags like <input>, <textarea>, and <button>. But in React, we use something called Controlled Components to manage form input using state.

1. Define controlled components

In React, a controlled component is an input element (like <input>, <textarea>, or <select>) whose value is controlled by React state. This means that the input's value is not stored in the DOM , but is instead stored in the component's state using useState or this.state. As the user types or interacts with the input, React updates the state using an event handler like onChange, and then re-renders the input with the new value. This gives you full control over the form data and allows validation, conditional rendering, and custom behaviors to be easily added.

1. Explain about various input controls

React supports all standard HTML form elements, but uses controlled components to manage them. Common input controls include:

* Textbox (<input type="text" />) – used for short text input like names.
* Textarea (<textarea></textarea>) – used for multi-line input like descriptions or complaints.
* Button (<button></button>) – used to trigger events like submitting a form.
* Radio buttons and checkboxes – used for selecting options.
* Select dropdown (<select><option></option></select>) – used for choosing from a list.

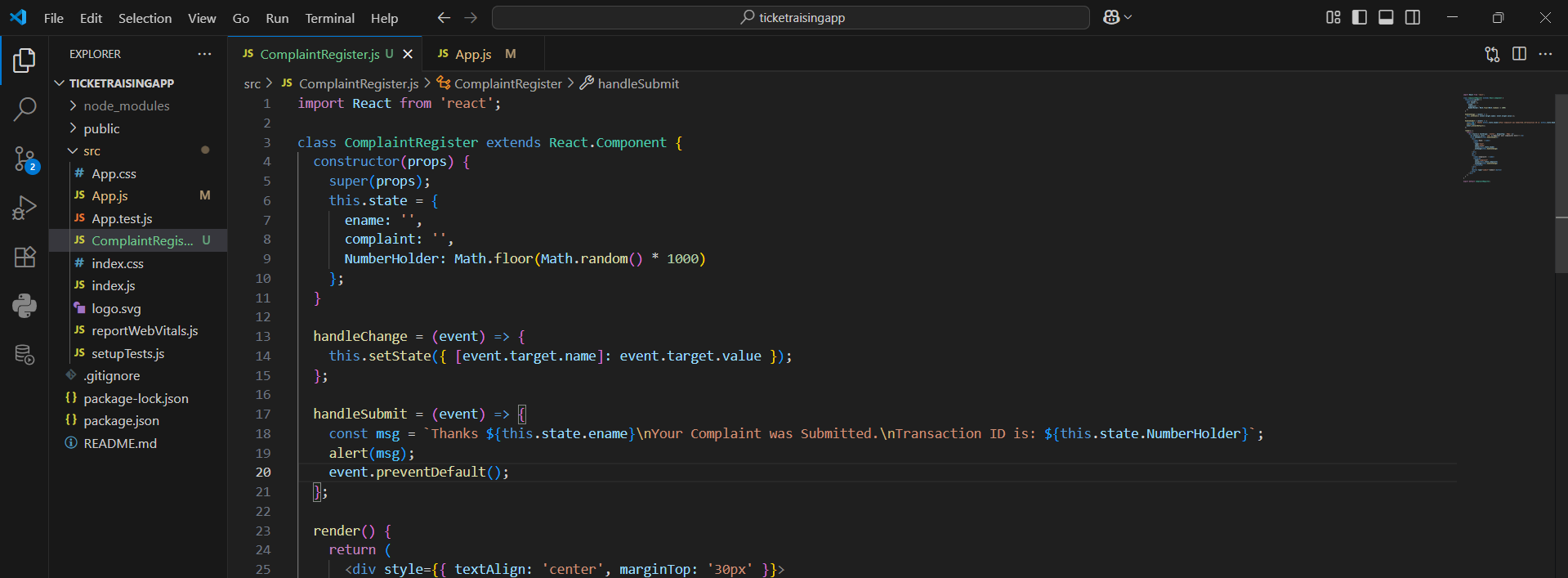
1. Explain about handling forms

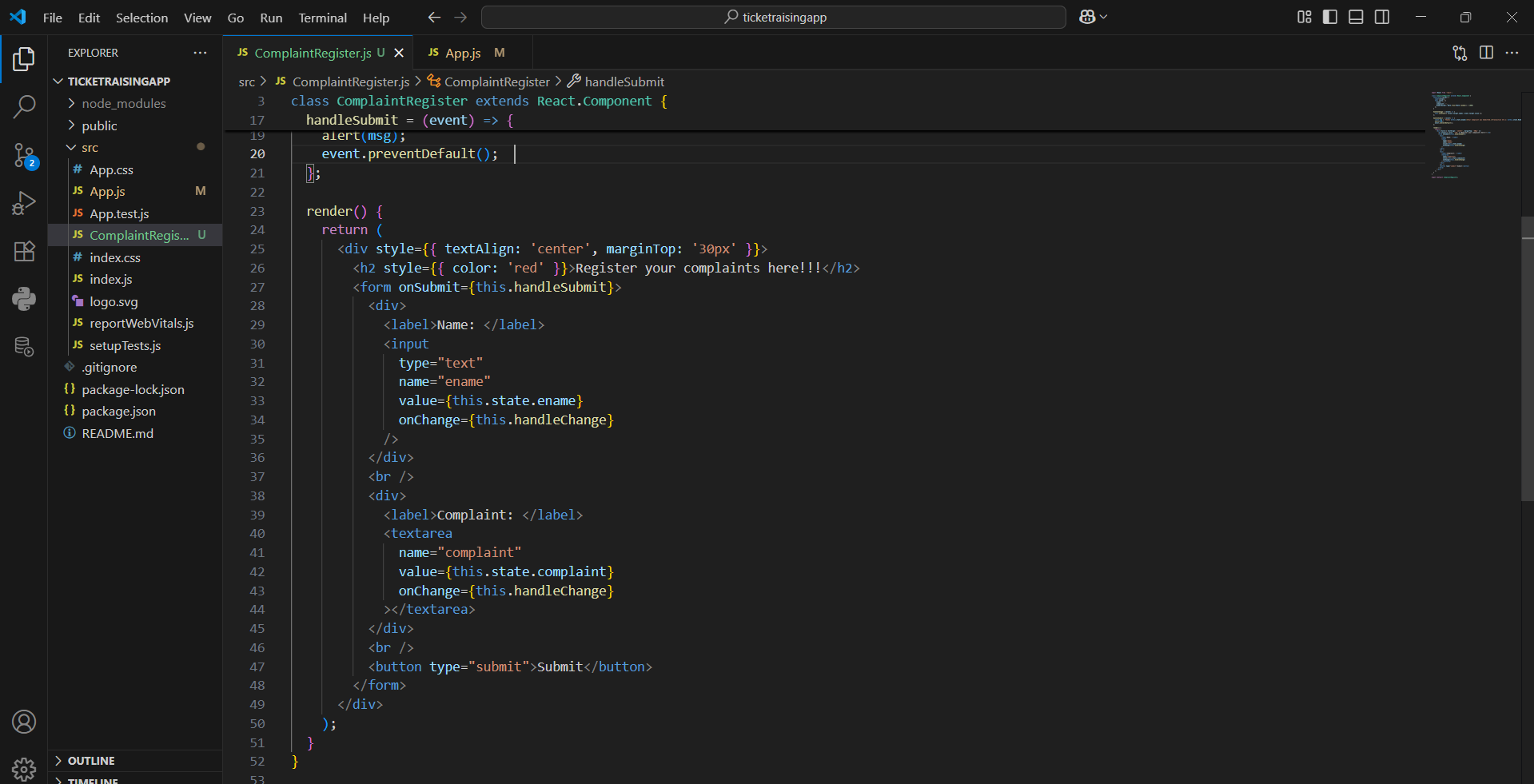
Handling forms in React involves capturing user input and storing it in the component's state. Each form element should have a name attribute and an onChange handler. The onChange function reads the event.target.name and event.target.value, and then updates the corresponding state value using setState (in class components) or useState (in functional components). This approach ensures that the form is controlled entirely by React, and it allows you to perform actions like validation, formatting, or live updates based on user input.

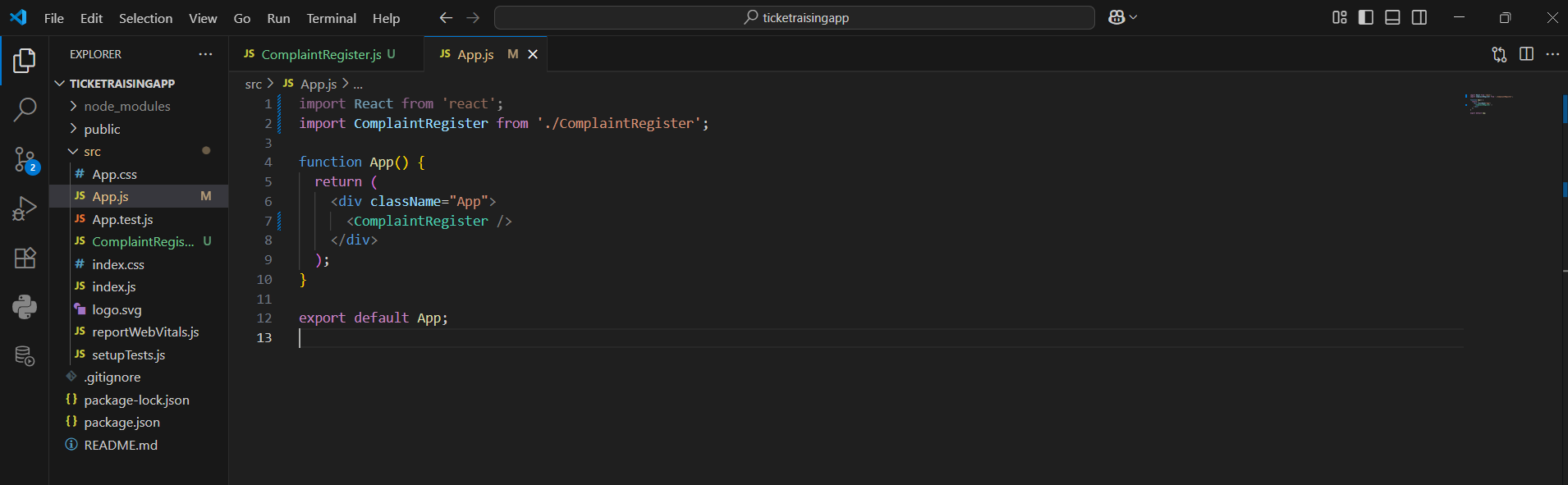
5.Explain about submitting forms

To handle form submission in React, you add an onSubmit handler to the <form> element. When the user clicks the submit button, the function gets triggered. Inside the handler, you usually call event.preventDefault() to prevent the page from reloading (which is default HTML behavior). Then, you can access the data stored in the state and process it as needed—like showing an alert, sending it to a server, or saving it locally. This approach gives full control over what happens during the form submission and helps you handle the data in a clean, React-friendly way.

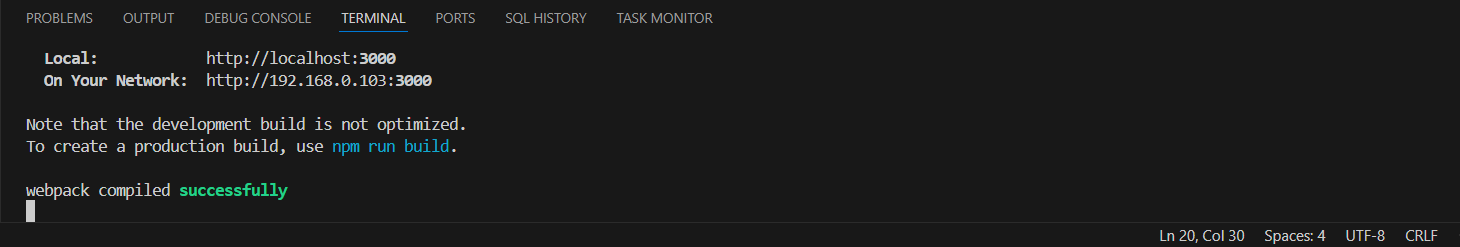
CODE:

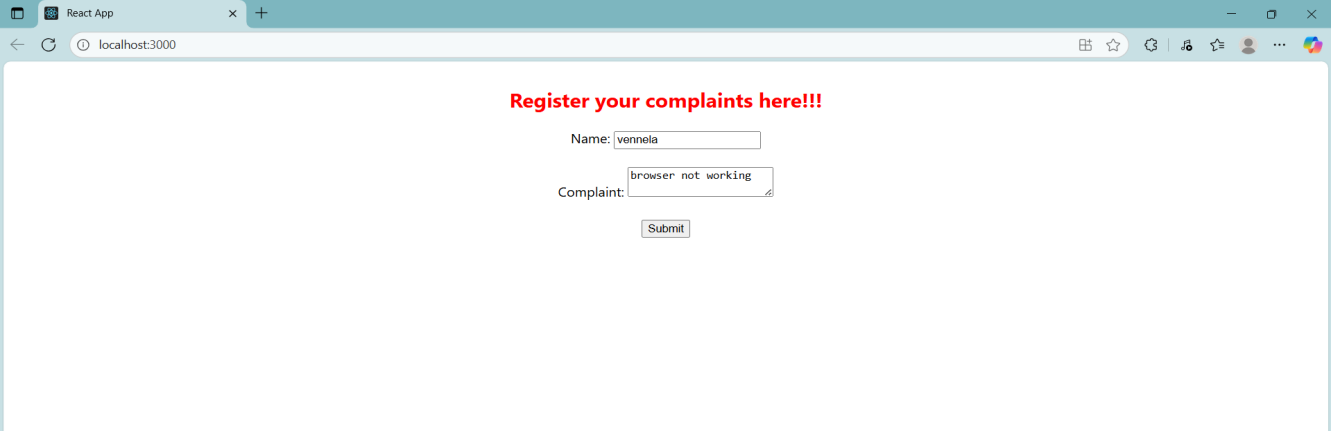


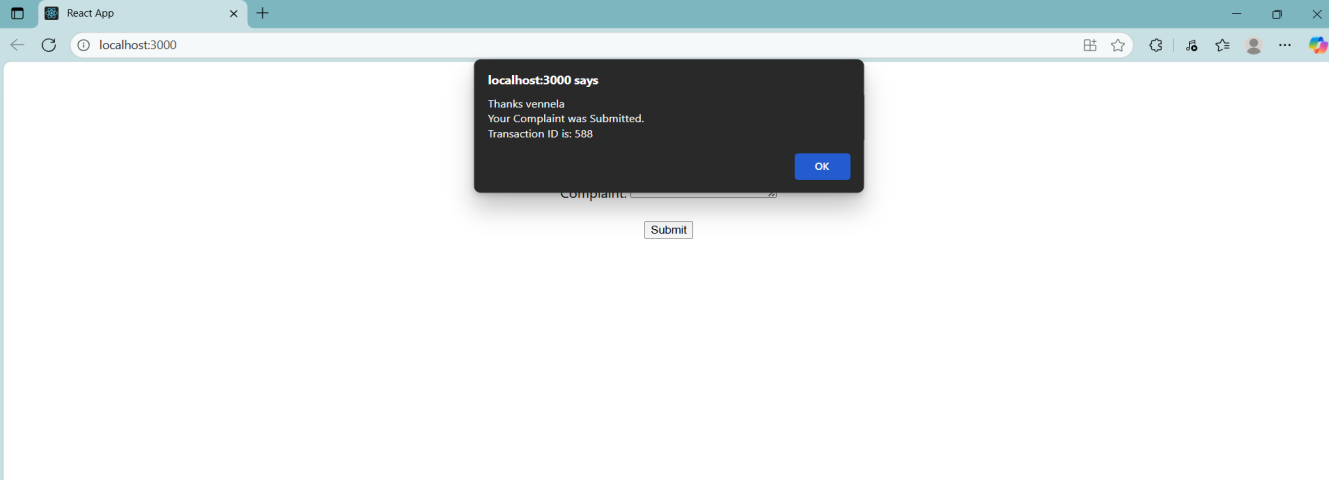




OUTPUT:







**Question 16**

Objectives

Explain React Forms validation

Identify the differences between React Form and HTML Form

Explain about controlled components

Identify various React Form input controls

Explain how to handle React Forms

Explain about submitting forms in React

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

Implement React forms validation

Use various input controls like textbox, button and textarea

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 “mailregisterapp” which will have a component named “register.js”. Create a form which accepts the name, email and password and validate the fields as per the following:

1.Name should have atleast 5 characters

2.Email should have @ and .

3.Password should have atleast 8 characters.

Ensure that validations are implemented through eventhandle and eventsubmit of a form.

Definitions:

1. Explain React Forms validation

React form validation is the process of checking user input to ensure it meets certain rules, like required fields or valid email formats. You can do this manually using JavaScript or with libraries like Formik or React Hook Form, which make validation easier and more organized.

1. Identify the differences between React Form and HTML Form

HTML forms work on their own and submit data by refreshing the page, while React forms are controlled using JavaScript. In React, you manage the form data using state, giving you full control over how data is handled, validated, and submitted.

1. Explain about controlled components

Controlled components in React are input elements like text boxes or checkboxes whose values are controlled by React state. This means the displayed value of the input always comes from the state, and changes are handled through onChange functions that update that state.

1. Identify various React Form input controls

React supports various form input controls like text inputs, checkboxes, radio buttons, dropdowns (<select>), and textareas. All of these can be made controlled components by linking them to state and handling their changes using event handlers.

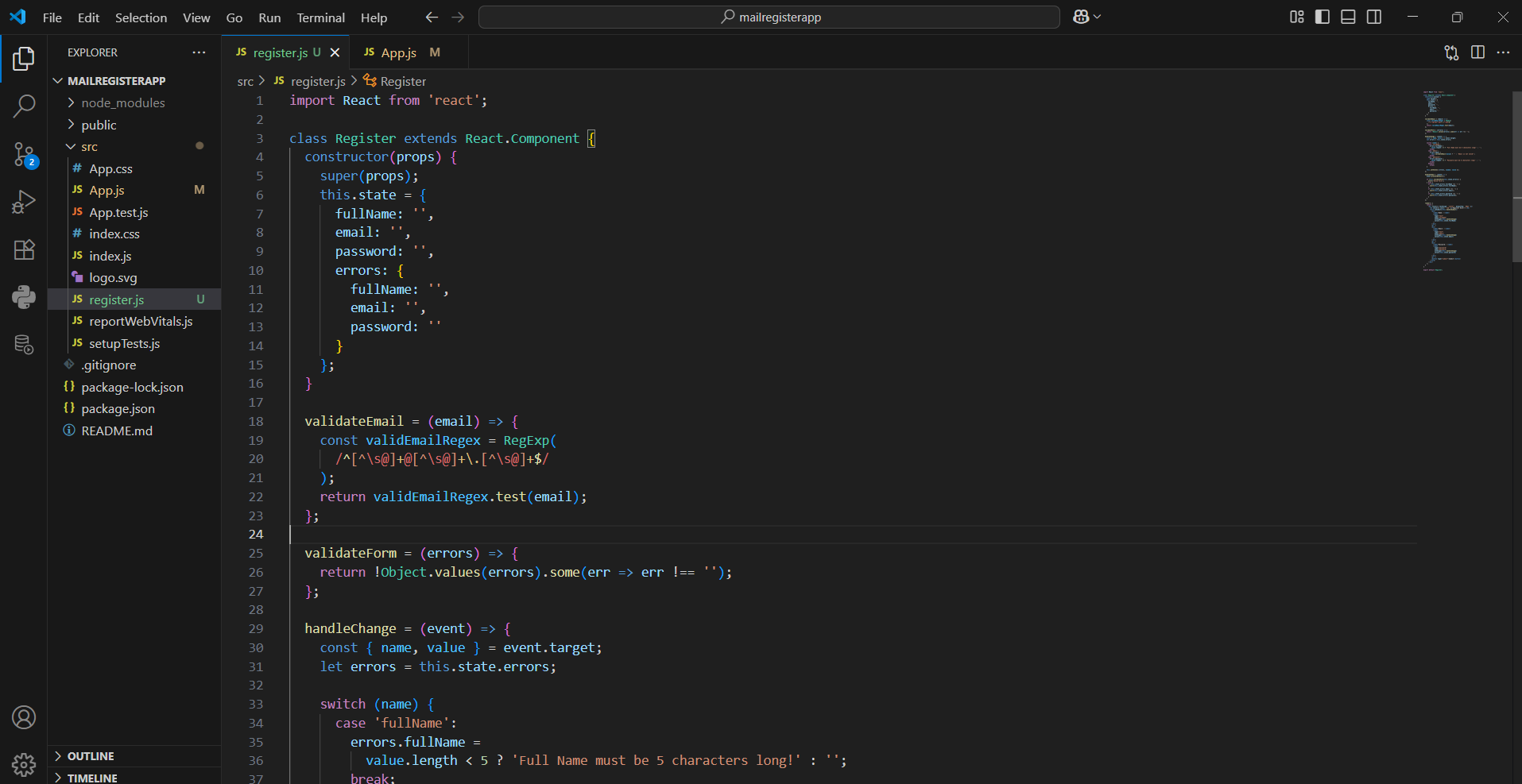
1. Explain how to handle React Forms

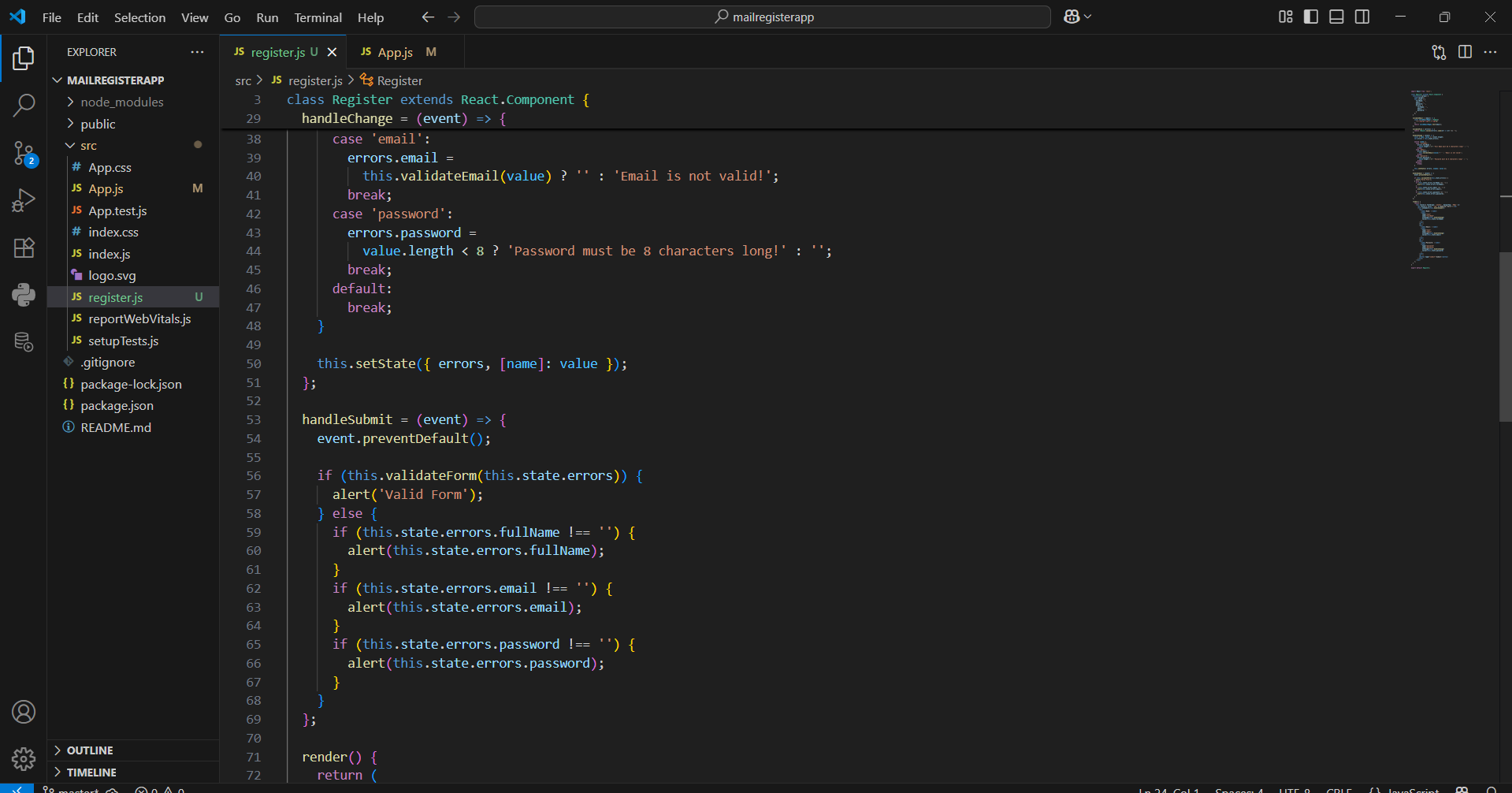
To handle forms in React, you create state variables for each input, use onChange to update the state as the user types, and handle the form submission with a function that gathers all the state values. This way, you can use the input data however you want without refreshing the page.

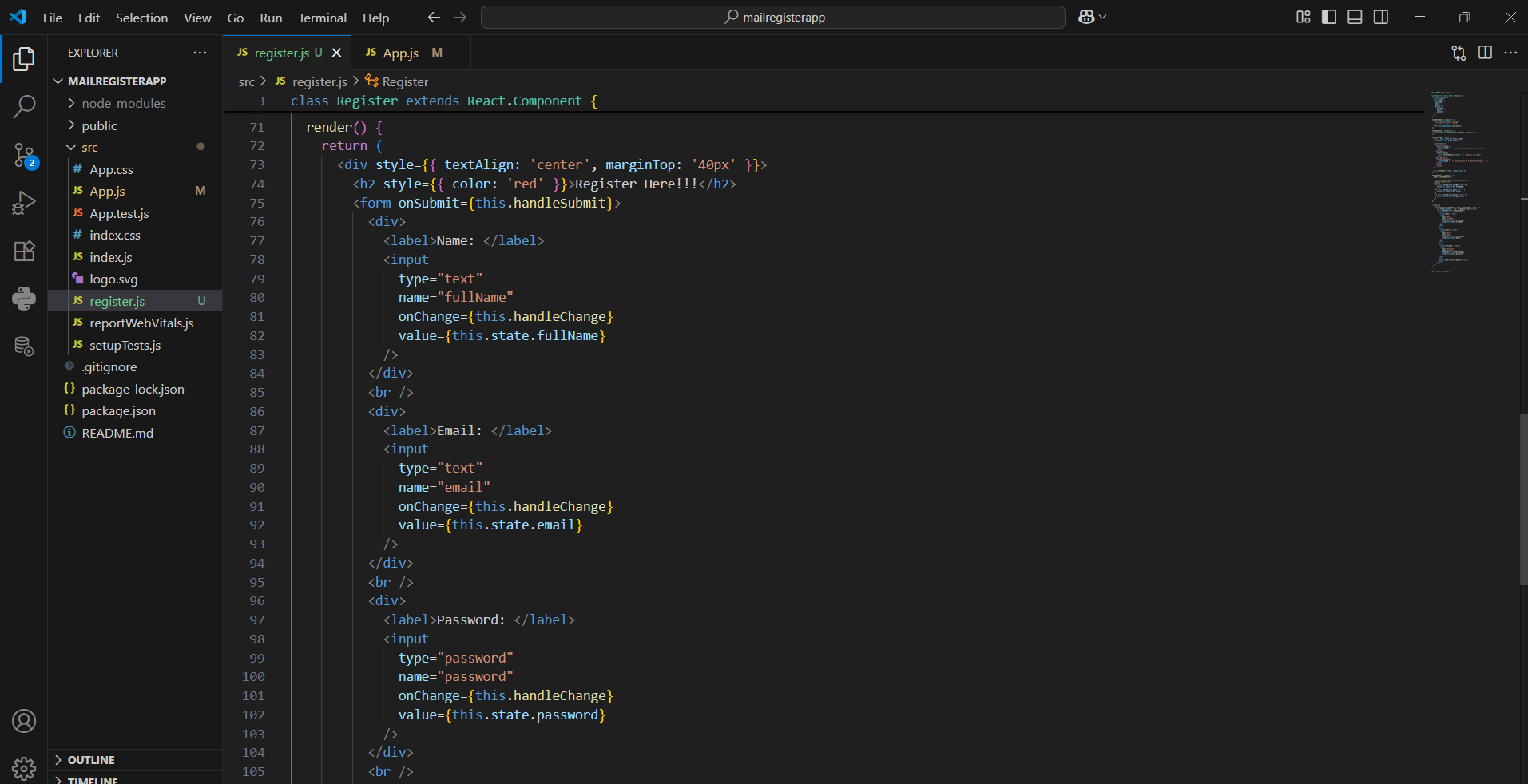
1. Explain about submitting forms in React

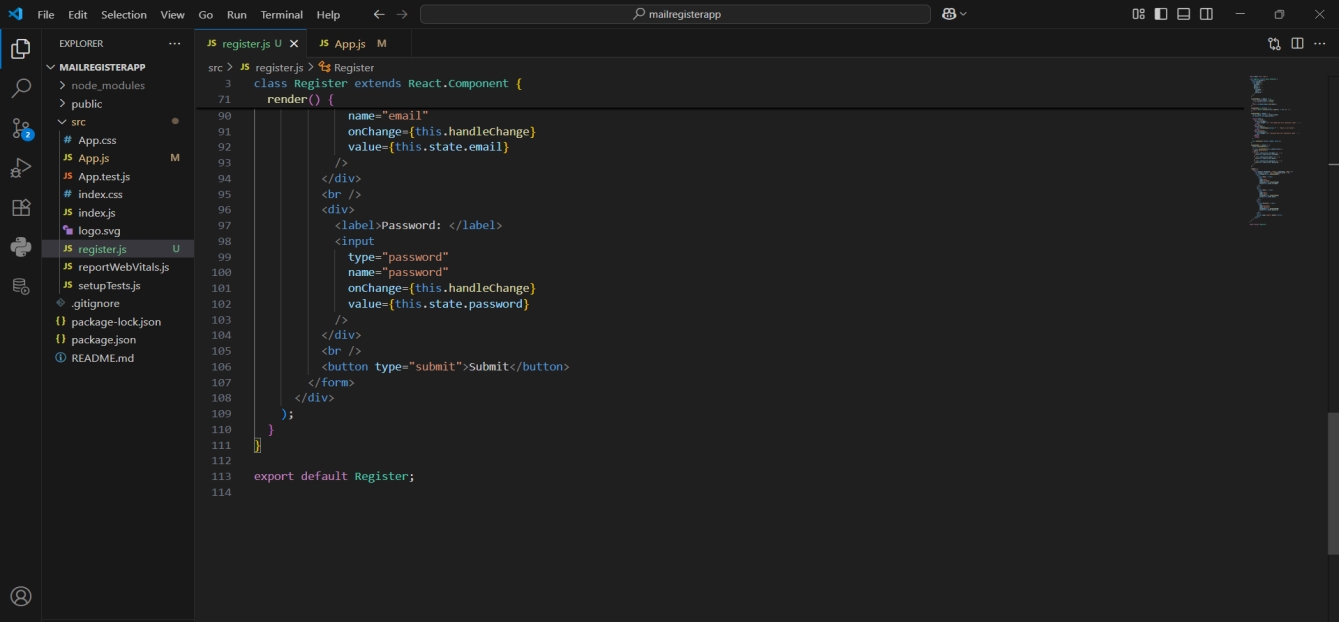
In React, when you submit a form, you usually prevent the default page reload using event.preventDefault(). Then you collect the data from state and do something with it like send it to a server. This gives you complete control over what happens when a form is submitted.

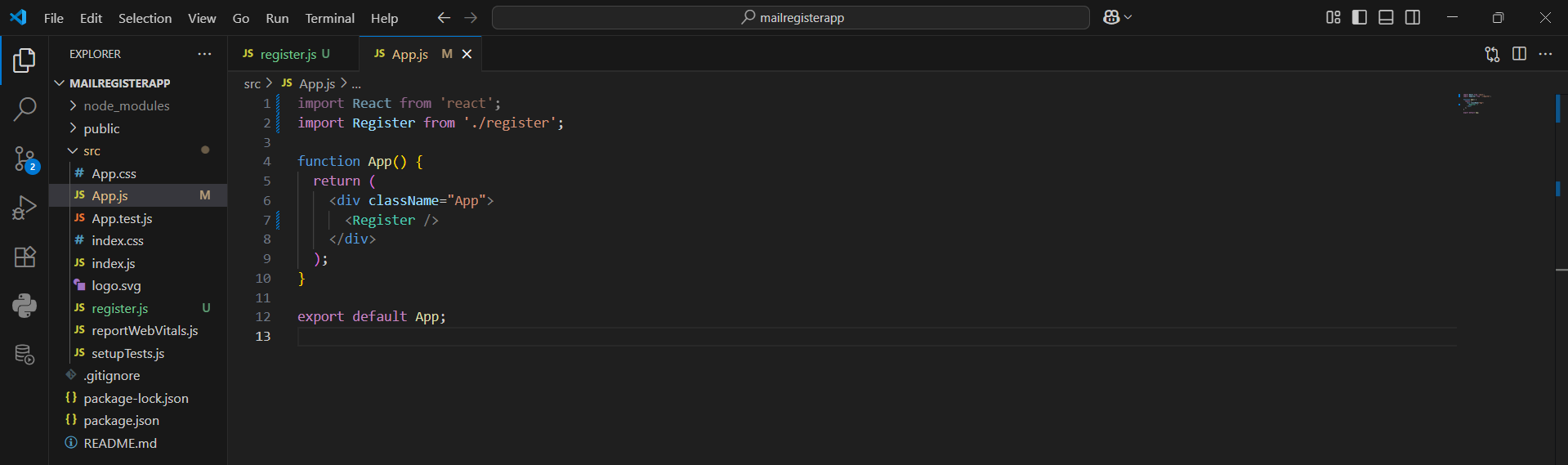
CODE:



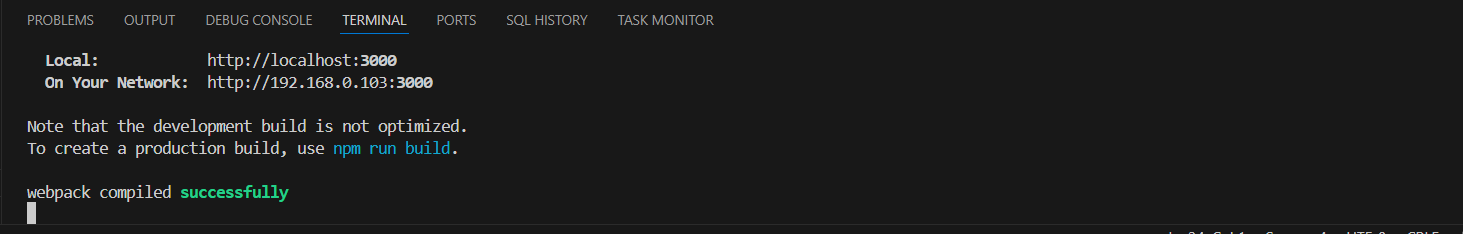


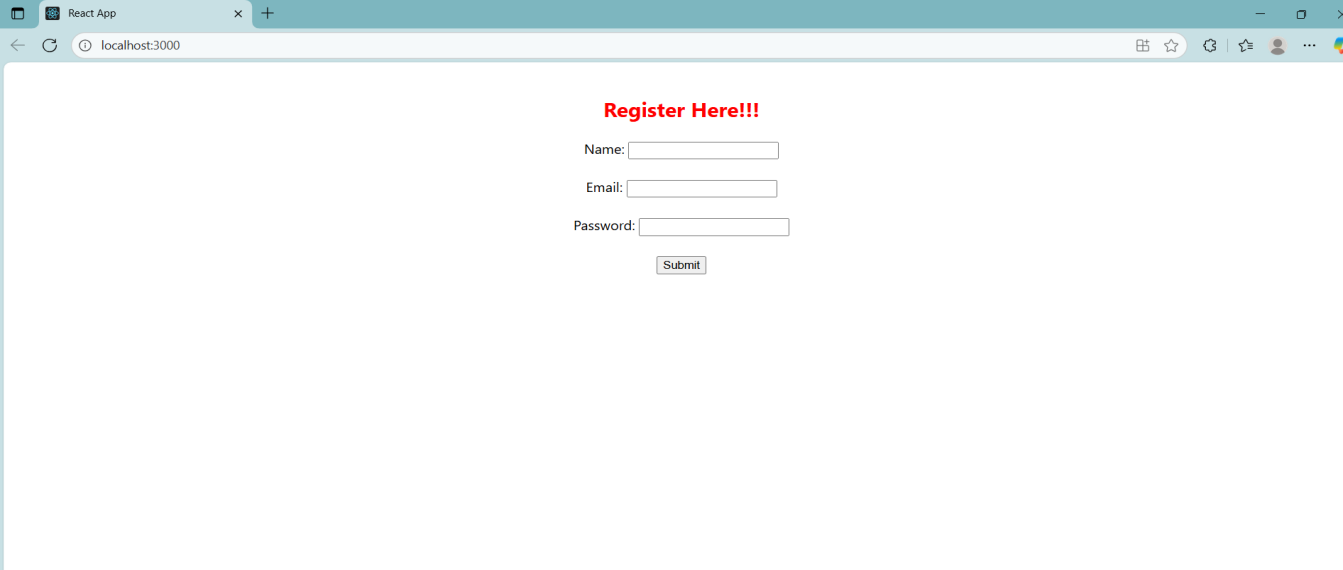


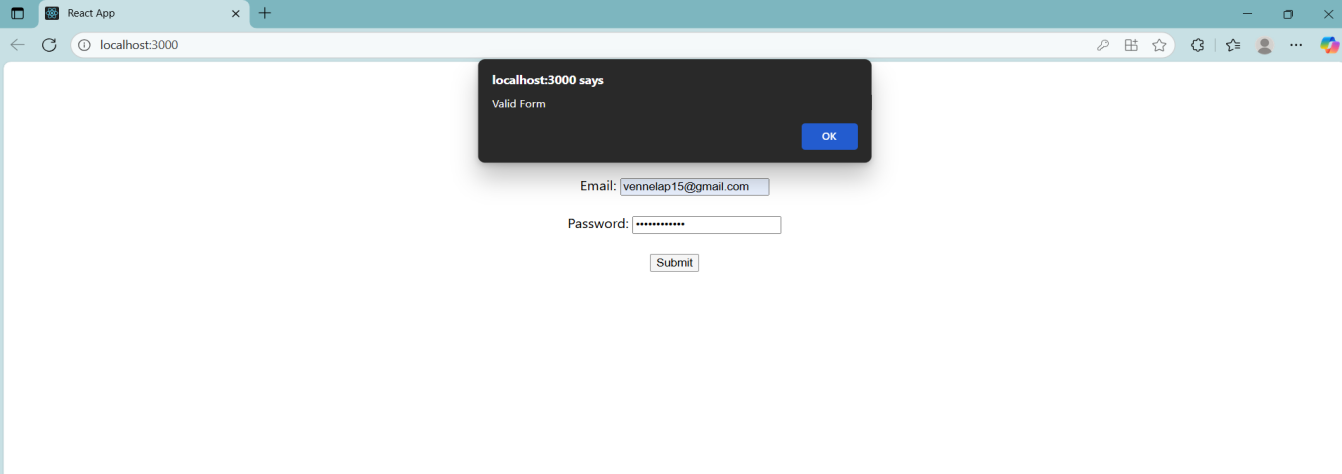


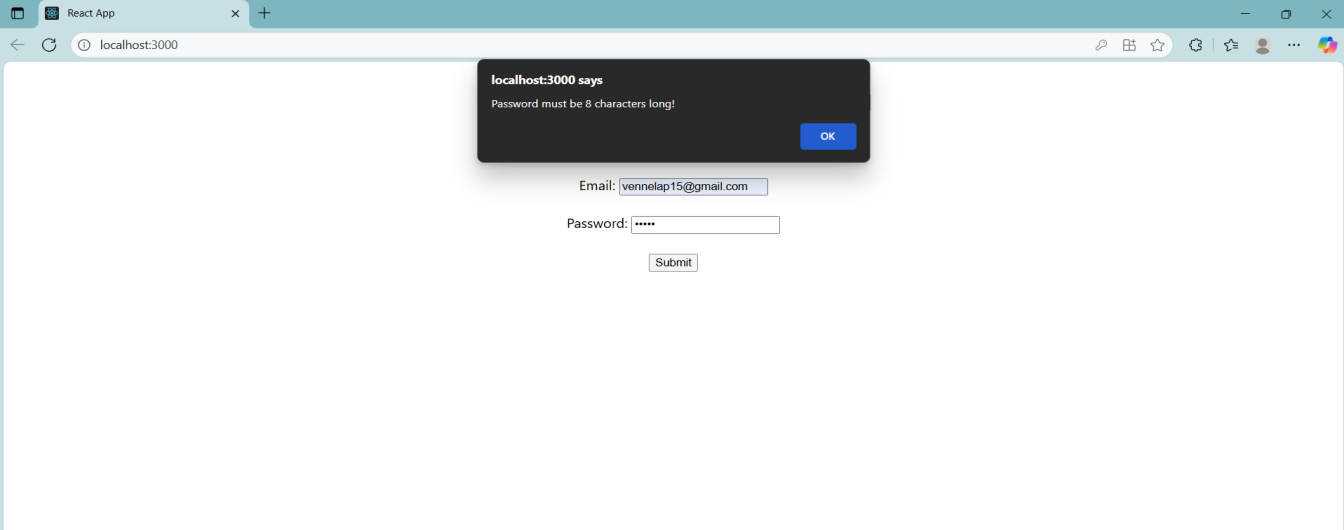


OUTPUT:









**Question 17**

Objectives

Explain how to consume REST APIs from React applications

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

Construct a React application that invokes the REST API and fetch data from the API

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 “fetchuserapp” which will retrieve the user details from https://api.randomuser.me/ and display the title, firstname and image of a user.

Create a component named “Getuser” and in the asynchronous method “ComponentDidMount ()” invoke the URL using fetch method and the response can be displayed in the render method of the component.

Code Snippet in Getuser Component:

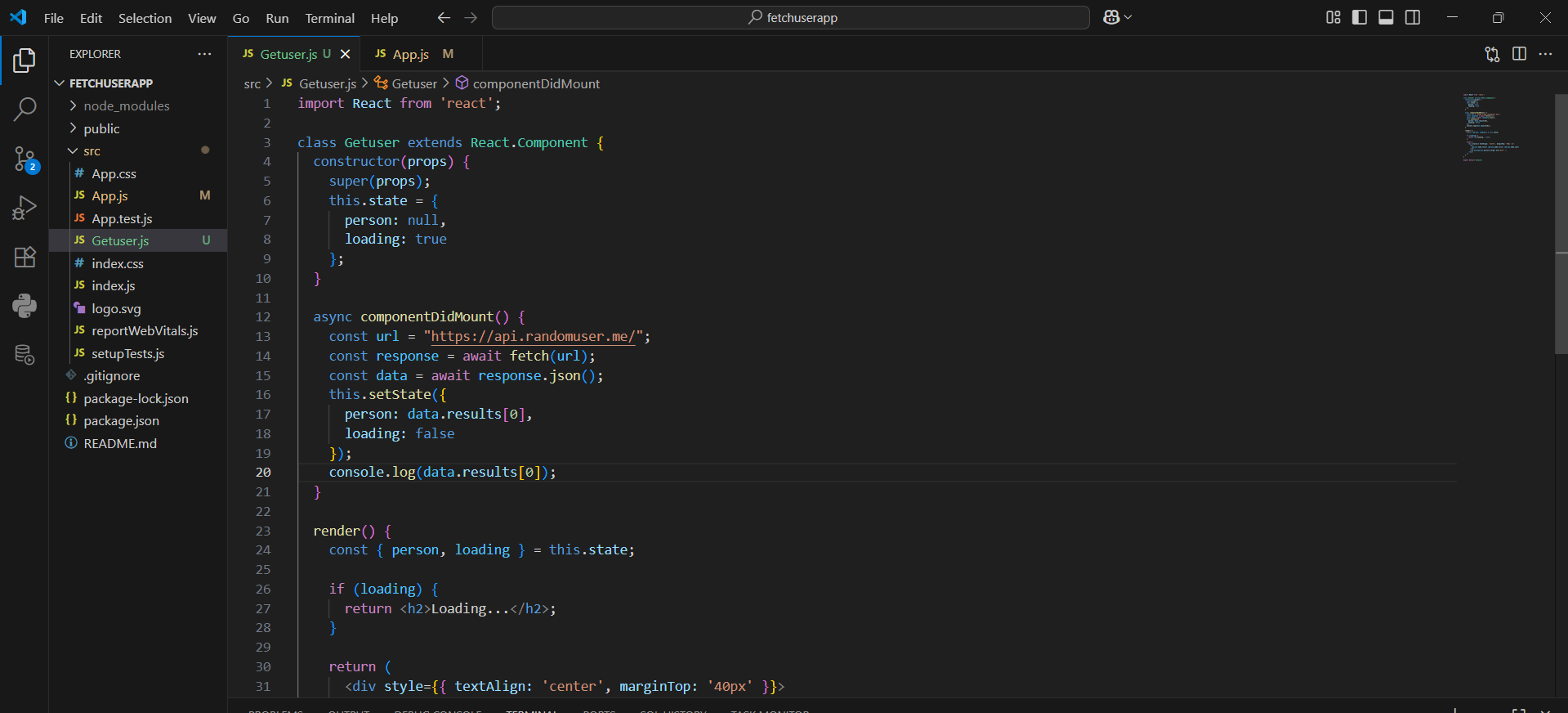
Answer and Output:

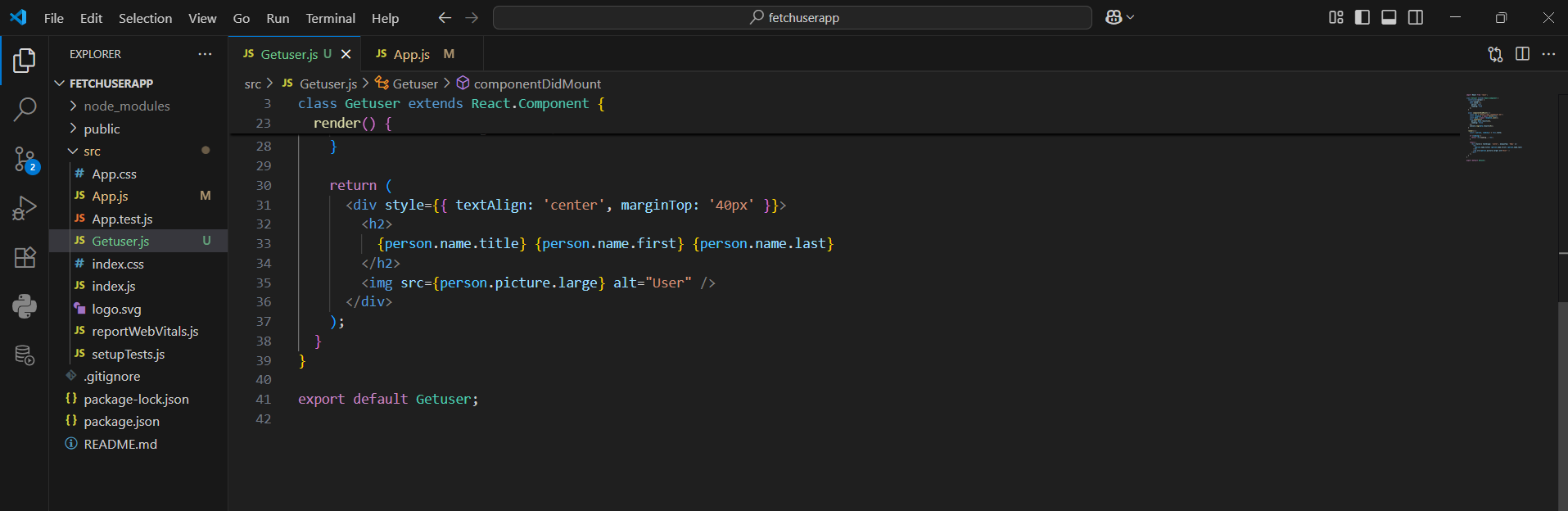
Definitions:

1. Explain how to consume REST APIs from React applications

To consume REST APIs in a React application, you typically use JavaScript functions like `fetch` or libraries such as `axios` to make HTTP requests to external services. This process usually happens inside a React component, often within the `useEffect` hook to trigger the request when the component mounts. You send a GET, POST, PUT, or DELETE request to the API endpoint, handle the response (usually in JSON format), and update the component’s state using `useState` to reflect the data in the UI. For example, you might fetch a list of users from an API and display them in a list once the data is successfully loaded. It's also important to handle loading states and errors to ensure a smooth user experience. For more complex apps, you might use tools like React Query or Redux Toolkit Query to manage API data more efficiently.

CODE:







OUTPUT:

