**15 . REACTJS – HOL**

**Ticket Raising App Using React Forms**

**Objectives**

**1. React Forms**

React forms are components that allow users to input and submit data through fields like textboxes, checkboxes, dropdowns, etc. Unlike traditional HTML forms, React forms use state to manage the values of input elements, making them dynamic and interactive.

**2. Controlled Components**

A controlled component is a form element (like <input>, <textarea>, etc.) that is controlled by React state. In these components:

* The value of the input is set by the state.
* Changes to the input update the state via onChange.

Example:

const [name, setName] = useState('');

<input type="text" value={name} onChange={(e) => setName(e.target.value)} />

**3. Input Controls in React**

Input controls are the form elements used to capture user input:

* Textbox (<input type="text" />)
* Textarea (<textarea></textarea>)
* Button (<button>Submit</button>)
* Checkbox, Radio buttons, Select dropdown, etc.

React uses useState to bind these controls to component state.

**4. Handling Forms**

Handling forms means capturing and managing user input:

* Use useState to hold input values.
* Use onChange handlers to update state.
* Use onSubmit to define what happens when the form is submitted.

**5. Submitting Forms**

Submitting a form involves:

* Listening to the onSubmit event of the <form> element.
* Preventing default form submission using e.preventDefault().
* Accessing the state values for processing (e.g., sending data to a server or showing an alert).

Example:

const handleSubmit = (e) => {

e.preventDefault();

alert(`Submitted: ${name}`);

};

**Prerequisites**

* Node.js
* NPM
* Visual Studio Code

**Tools & Technologies Used**

* React.js
* JavaScript (ES6+)
* JSX
* HTML & CSS (optional for styling)

**Steps to Implement the Application**

**Step 1: Create React App**

npx create-react-app ticketraisingapp

cd ticketraisingapp

code .

**Step 2: Create Component**

Inside the src folder, create a file called ComplaintRegister.js.

**ComplaintRegister.js**

import React, { useState } from 'react';

function ComplaintRegister() {

const [name, setName] = useState('');

const [complaint, setComplaint] = useState('');

const handleSubmit = (e) => {

e.preventDefault();

const referenceNumber = Math.floor(1000 + Math.random() \* 9000);

alert(`Complaint submitted!\nName: ${name}\nComplaint: ${complaint}\nReference No: ${referenceNumber}`);

setName('');

setComplaint('');

};

return (

<div>

<h2>Complaint Register</h2>

<form onSubmit={handleSubmit}>

<label>Employee Name:</label><br />

<input type="text" value={name} onChange={(e) => setName(e.target.value)} required /><br /><br />

<label>Complaint:</label><br />

<textarea value={complaint} onChange={(e) => setComplaint(e.target.value)} required /><br /><br />

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

</form>

</div>

);

}

export default ComplaintRegister;

**Step 3: Update App.js**

Replace the contents of App.js:

**App.js**import React from 'react';

import ComplaintRegister from './ComplaintRegister';

function App() {

return (

<div className="App">

<ComplaintRegister />

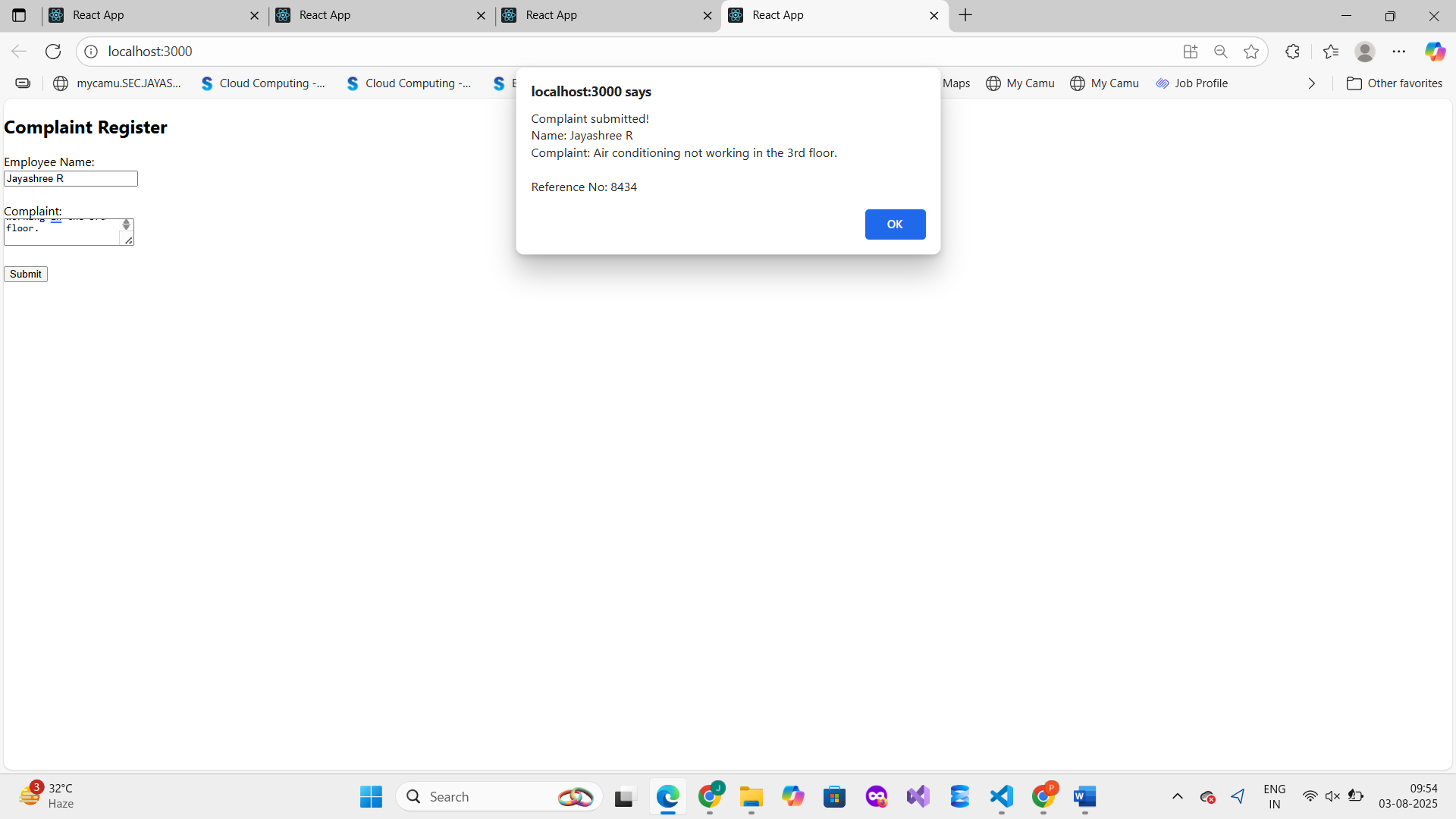
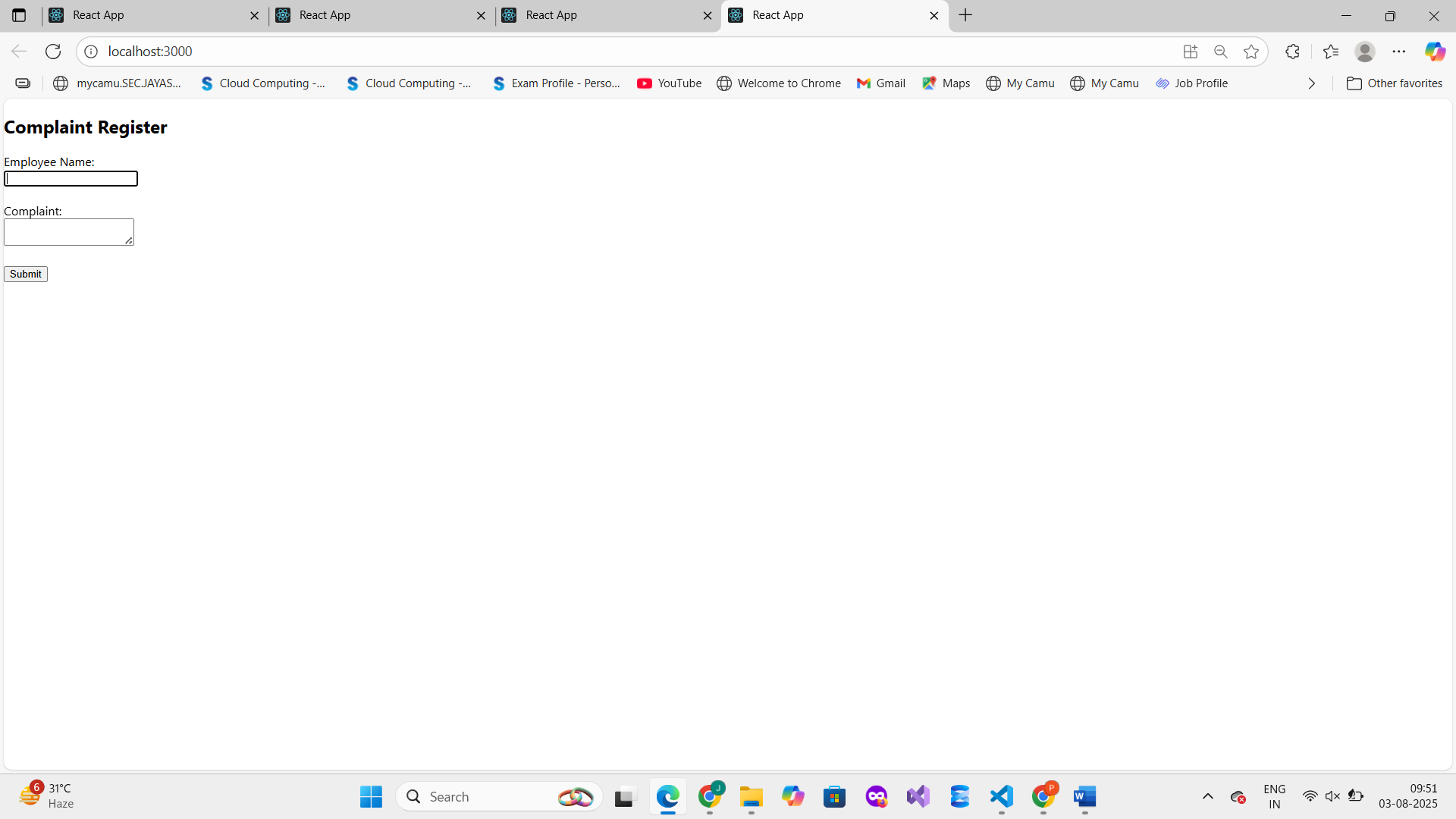
</div>

);

}

export default App;

**7. Output**



**8. Key Concepts Covered**

* React controlled components using useState
* Handling form inputs: input, textarea, button
* Form submission using handleSubmit
* Generating a unique reference number using Math.random()
* Resetting form fields after submission

**9. Conclusion**

In this lab, we successfully built a React application with a complaint registration form. We understood the importance of controlled components, how to capture and submit user input using event handlers, and how to provide feedback through alerts with generated reference numbers.