1. **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

```
return(
       players.map((item)=>
      {
      return(
         <div>
            <li>Mr. {item.name}<span> {item.score} </span></li>
         </div>

      )})
      )
```

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

```
return(
     players.map((item)=>
    {
     if(item.score<=70)
      {
         players70.push(item);
      }
     }),
```

2. IndianPlayers

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

```
export function OddPlayers([first,, third,,fifth]) {
   return(
   <div>
      <li> First :  {first}  </li>
      <li> Third :  {third} </li>
      <li> Fifth : {fifth}</li>
   </div>)}
```

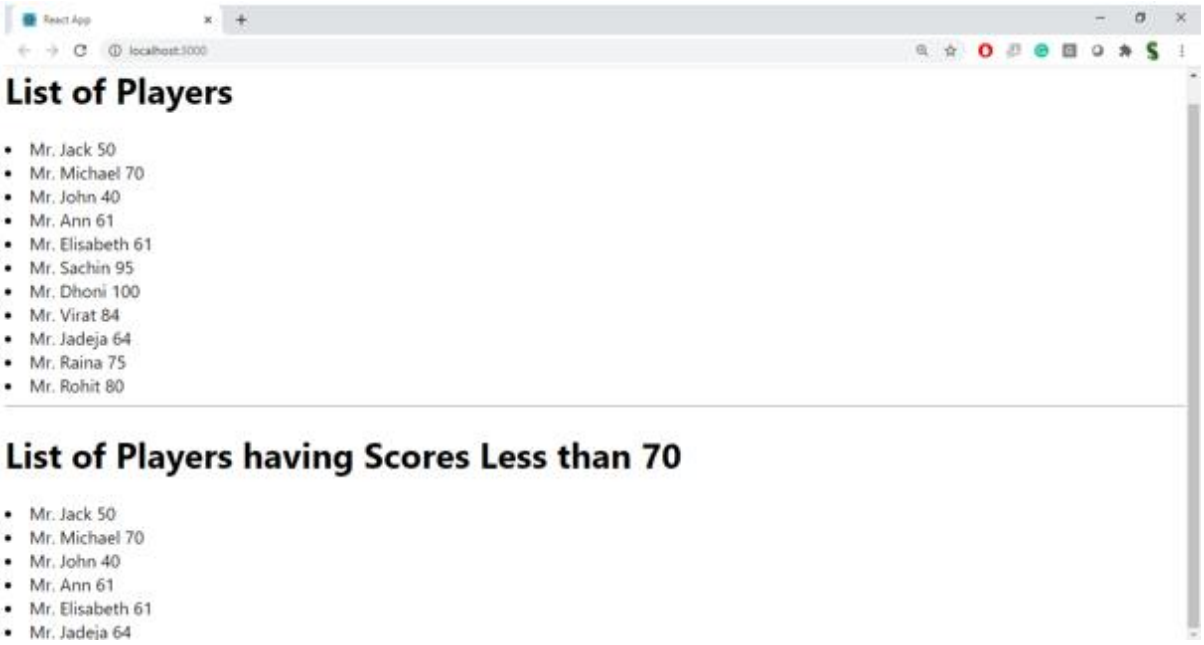
    b. Declare two arrays T20players and RanjiTrophy players and merge the two arrays and display them using the Merge feature of ES6

```
const T20Players=['First Player','Second Player','Third Player'];
    const RanjiTrophyPlayers=['Fourth Player','Fifth Player','Sixth Player'];
   export const IndianPlayers=[...T20Players, ...RanjiTrophyPlayers]
```

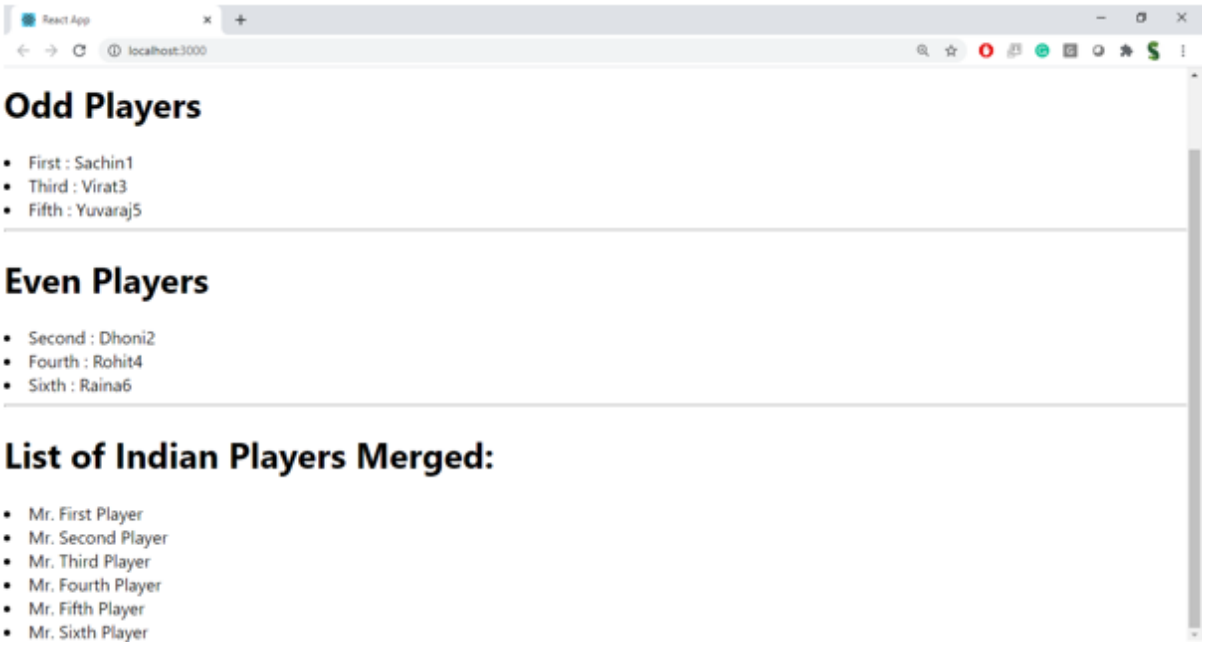
Display these two components in the same home page using a simple if else in the flag variable.

Output:

When Flag=true



When Flag=false



Answer:

1.Features of ES6 (ECMAScript 6)

- let and const for block-scoped variables
- Arrow functions
- Template literals
- Default parameters
- Destructuring (arrays and objects)
- Rest and spread operators
- Classes and inheritance
- Modules (import and export)
- Promises for async handling
- New data structures: Map and Set
- Enhanced object literals
- for...of loop

2.JavaScript let

- Declares block-scoped variables

- Can be updated
- Cannot be redeclared in the same block
- Hoisted but not initialized (temporal dead zone)

3.Differences between var and let

Feature	var	let
Scope	Function-scoped	Block-scoped
Hoisting	Yes, initialized as undefined	Yes, but not initialized
Redeclaration	Allowed	Not allowed
Temporal Dead Zone	No	Yes

4. JavaScript const

- Declares block-scoped constants
- Must be initialized during declaration
- Cannot be reassigned
- Contents of arrays/objects can be modified

5. ES6 Class Fundamentals

- Introduces the class keyword
- Uses constructor() for initialization
- Supports methods inside class body
- Provides a more readable OOP syntax

6.ES6 Class Inheritance

- Uses extends keyword for inheritance
- super() calls the parent class constructor
- Subclasses inherit methods and properties from the parent class

7. ES6 Arrow Functions

- Shorter syntax for functions
- Do not bind their own this
- Cannot be used as constructors
- Useful in callbacks and array methods

8.Set and Map

- **Set:**
    - Stores unique values
    - No duplicate entries allowed
    - Maintains insertion order
- **Map:**
    - Stores key-value pairs
    - Keys can be any type
    - Maintains insertion order

App.js:

```
import React from 'react';
import ListofPlayers from './Components/ListofPlayers';
import Scorebelow70 from './Components/Scorebelow70';
import { OddPlayers, EvenPlayers, ListofIndianPlayers } from './Components/IndianPlayers';

function App() {
  const flag = true;

  const players = [
    { name: 'Jack', score: 50 },
    { name: 'Michael', score: 70 },
    { name: 'John', score: 40 },
    { name: 'Ann', score: 61 },
    { name: 'Elisabeth', score: 61 },
```

```
  { name: 'Sachin', score: 95 },
  { name: 'Dhoni', score: 100 },
  { name: 'Virat', score: 84 },
  { name: 'Jadeja', score: 64 },
  { name: 'Raina', score: 75 },
  { name: 'Rohit', score: 80 },
];

const IndianTeam = ['Sachin1', 'Dhoni2', 'Virat3', 'Rohit4', 'Yuvaraj5', 'Raina6'];

if (flag === true) {
  return (
    <div>
      <h1>List of Players</h1>
      <ListofPlayers players={players} />
      <hr />
      <h1>List of Players having Scores Less than 70</h1>
      <Scorebelow70 players={players} />
    </div>
  );
} else {
  return (
    <div>
      <h1>Odd Players</h1>
      {OddPlayers(IndianTeam)}
      <hr />
      <h1>Even Players</h1>
      {EvenPlayers(IndianTeam)}
      <hr />
      <h1>List of Indian Players Merged:</h1>
      <ListofIndianPlayers />
```

```
    </div>
   );
  }
}

export default App;
```

ListofPlayers.js:

```
import React from 'react';

function ListofPlayers({ players }) {
  return (
    <ul style={{ paddingLeft: '20px' }}>
      {players.map((player, index) => (
        <li key={index}>
          Mr. {player.name} {player.score}
        </li>
      ))}
    </ul>
  );
}

export default ListofPlayers;
```

Scorebelow70.js:

```
import React from 'react';

const Scorebelow70 = ({ players }) => {
  const players70 = players.filter(player => player.score <= 70);

  return (
    <ul style={{ paddingLeft: '20px' }}>
      {players70.map((item, index) => (
```

```
      <li key={index}>
        Mr. {item.name} <span>{item.score}</span>
      </li>
    ))}
  </ul>
 );
};

export default Scorebelow70;
```

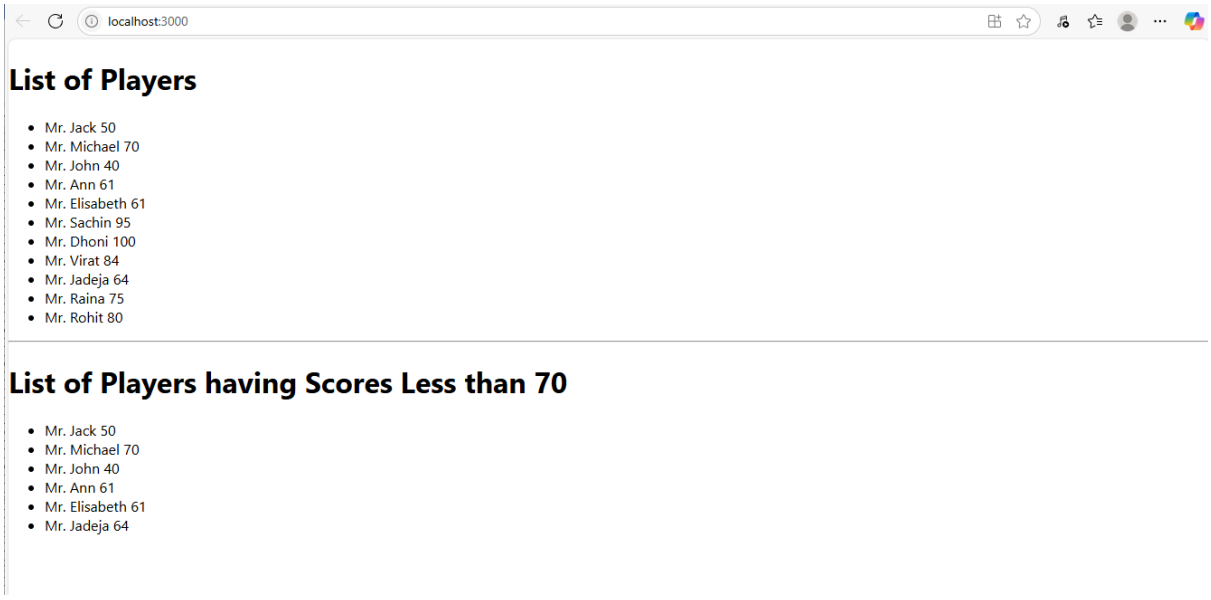
IndianPlayers.js:

```
import React from 'react';

export function OddPlayers([first, , third, , fifth]) {
 return (
  <ul style={{ paddingLeft: '20px' }}>
   <li>First: {first}</li>
   <li>Third: {third}</li>
   <li>Fifth: {fifth}</li>
  </ul>
 );
}

export function EvenPlayers([, second, , fourth, , sixth]) {
 return (
  <ul style={{ paddingLeft: '20px' }}>
   <li>Second: {second}</li>
   <li>Fourth: {fourth}</li>
   <li>Sixth: {sixth}</li>
  </ul>
 );
}
```

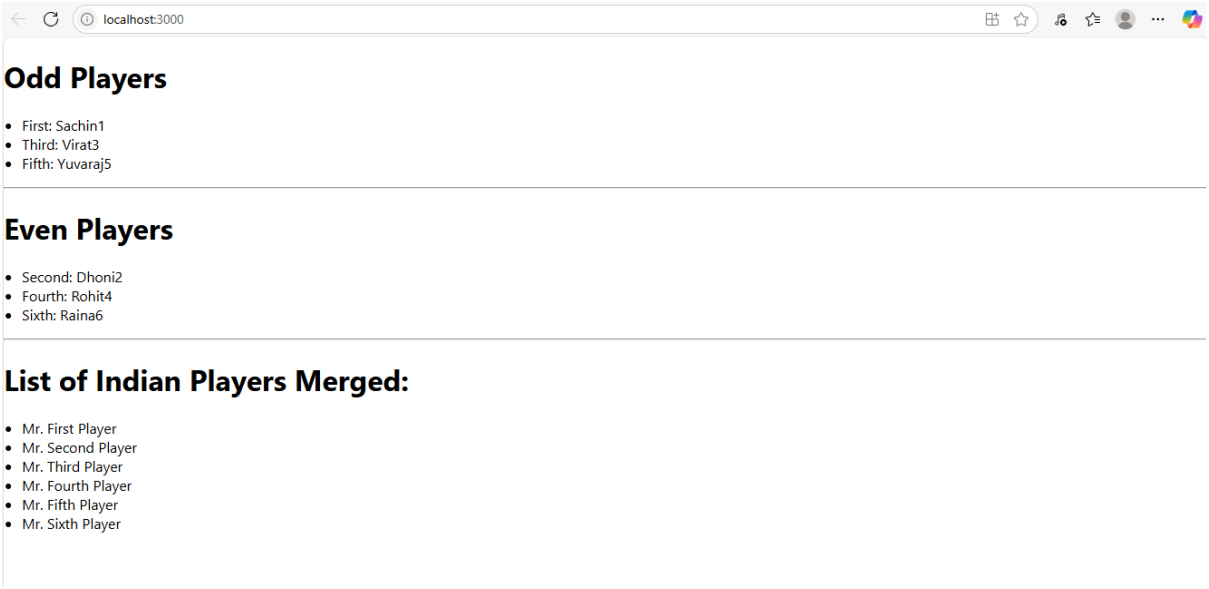
```
const T20Players = ['First Player', 'Second Player', 'Third Player'];
const RanjiTrophyPlayers = ['Fourth Player', 'Fifth Player', 'Sixth Player'];

export const IndianPlayers = [...T20Players, ...RanjiTrophyPlayers];

export function ListofIndianPlayers() {
  return (
    <ul style={{ paddingLeft: '20px' }}>
      {IndianPlayers.map((player, index) => (
        <li key={index}>Mr. {player}</li>
      ))}
    </ul>
  );
}
```

Output:

Flag==True



Flag==False:



2. 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.

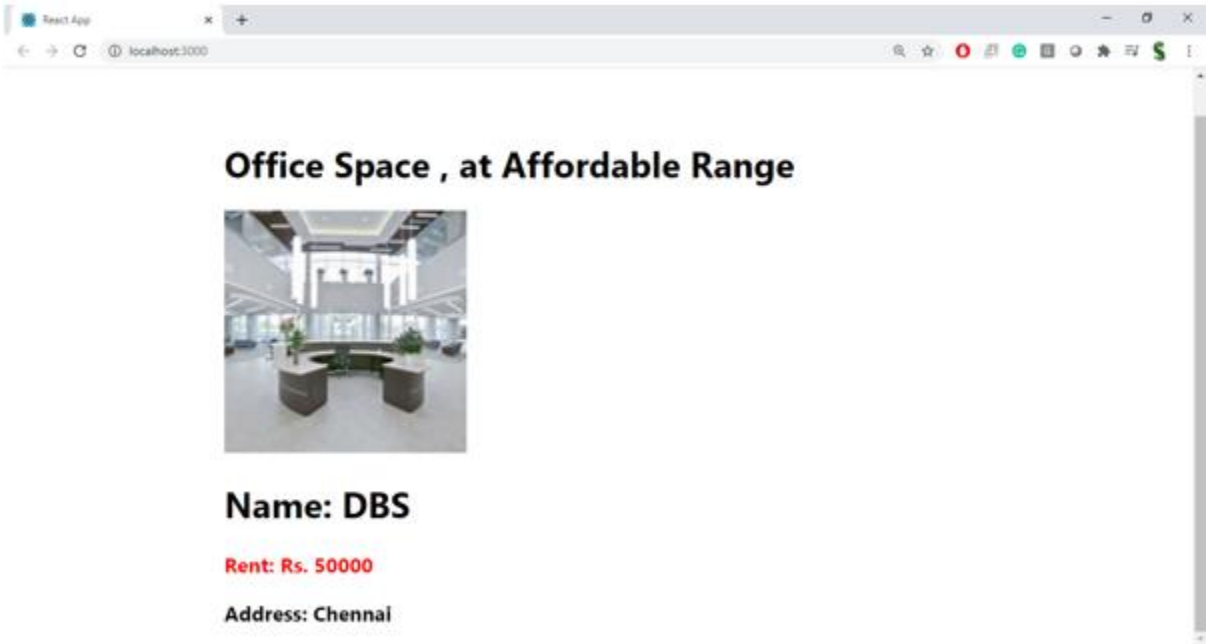
Attribute to display the image of the office space

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.

Output:



Hint:

```
{
   let colors=[];
   if(ItemName.Rent<=60000)
   {
      colors.push('textRed');
   }
   else{
      colors.push('textGreen');
   }
```

```
const element="Office Space"
const jsxatt=<img src={sr} width="25%" height="25%" alt="Office Space"/>
const ItemName={Name:"DBS", Rent: 50000, Address:'Chennai'}
<h1>{element} , at Affordable Range </h1>
{jsxatt}
<h1>Name: {ItemName.Name}</h1>
<h3> Rent: Rs. {ItemName.Rent}</h3>
<h3> Address: {ItemName.Address}</h3>
```

Answer:

- **Define JSX**

JSX is a syntax extension for JavaScript that allows writing HTML-like code inside JavaScript, used in React to describe UI components.

- **Explain about ECMA Script**

ECMAScript is the standardized version of JavaScript that defines the core features and syntax rules of the language. React commonly uses ES6 and newer versions.

- **Explain React.createElement()**

React.createElement() is a method used to create React elements without using JSX. It takes the element type, properties, and children as arguments.

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

React nodes can be created using JSX by writing HTML-like syntax inside JavaScript. Each JSX tag represents a React element or component.

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

JSX is rendered to the DOM using ReactDOM.render(), which mounts the JSX content into a specific HTML element in the web page.

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

JavaScript expressions can be used in JSX by wrapping them inside curly braces {}, allowing dynamic content to be displayed.

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

Inline CSS in JSX is written as a JavaScript object with camelCase property names and passed to the style attribute.

App.js:

```
import React from "react";

import officeImage from "./office.jpg";


function App() {
 const heading = "Office Space";


 return (
  <div
   style={{
    fontFamily: "Arial, sans-serif",
    marginTop: "30px",
    paddingLeft: "200px",
    textAlign: "left"
   }}
  >
   <h1 style={{ fontWeight: "bold", fontSize: "28px" }}>
    {heading}, at Affordable Range
   </h1>
   <img
```

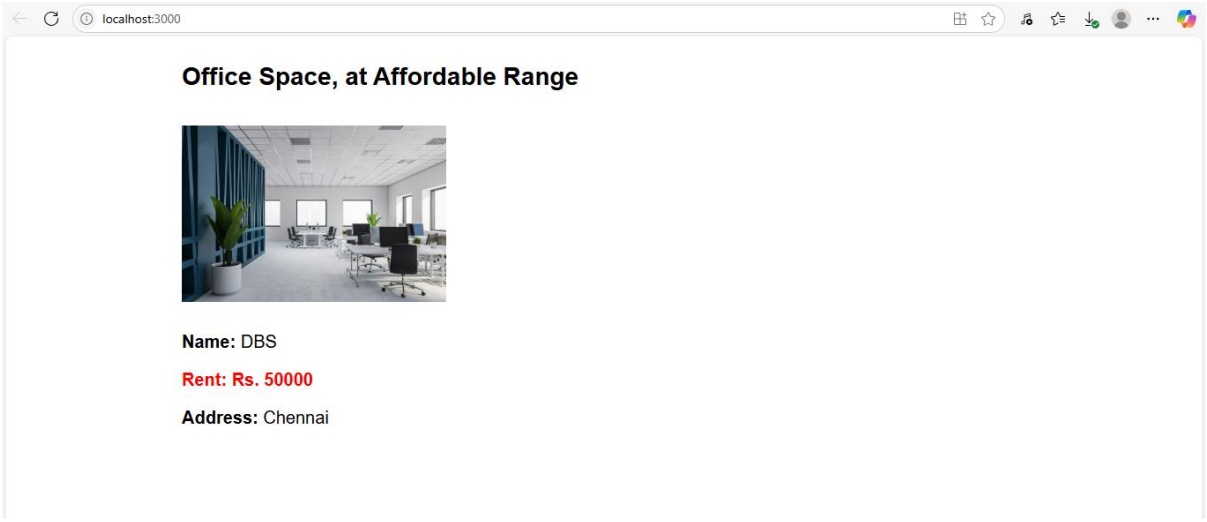
```
        src={officeImage}
        alt="Office"
        style={{ width: "300px", height: "auto", marginTop: "20px" }}
      />
      <div
        style={{
          marginTop: "30px",
          fontSize: "20px",
          maxWidth: "300px"
        }}
      >
        <p>
          <strong>Name:</strong> DBS
        </p>
        <p style={{ color: "red", fontWeight: "bold" }}>Rent: Rs. 50000</p>
        <p>
          <strong>Address:</strong> Chennai
        </p>
      </div>
    </div>
  );
}

export default App;
```

App.css:

```
.App {
  text-align: center;
  font-family: Arial, sans-serif;
  padding: 30px;
}
```

```
.card {
  margin: 20px auto;
  width: 300px;
}

.office-image {
  width: 100%;
  height: auto;
  border-radius: 8px;
  display: block;
  margin: 0 auto;
}

.office-details {
  margin-top: 15px;
  text-align: left;
  padding-left: 5px;
}
.heading {
  text-align: center;
  font-size: 28px;
  font-weight: bold;
  margin-bottom: 20px;
}
```

Output:



3. 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:

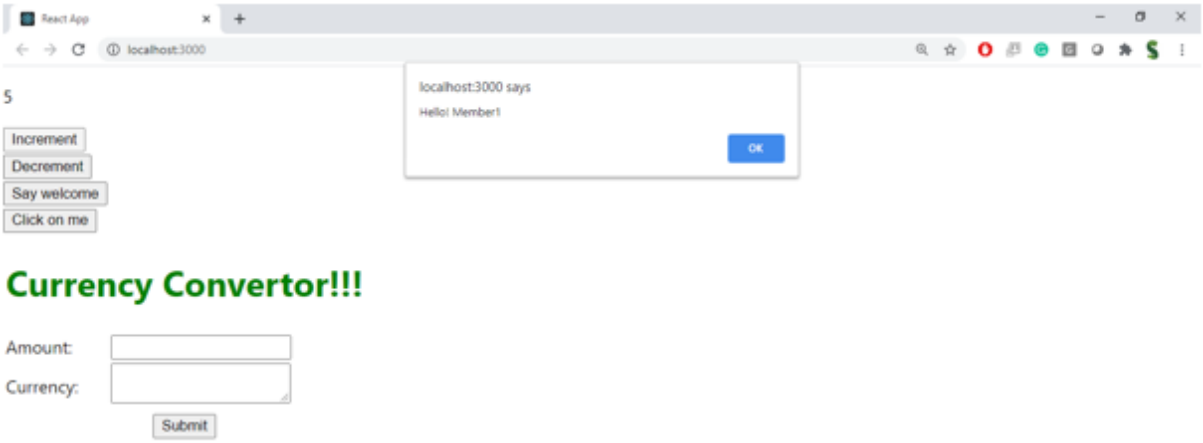
- **Node.js**
- **NPM**
- **Visual Studio Code**

Notes

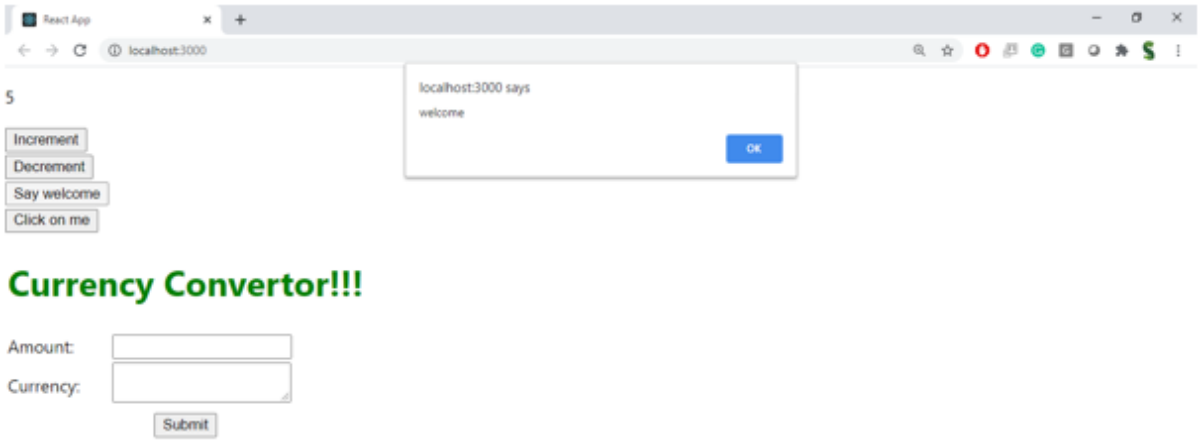
Estimated time to complete this lab: 90 minutes.

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

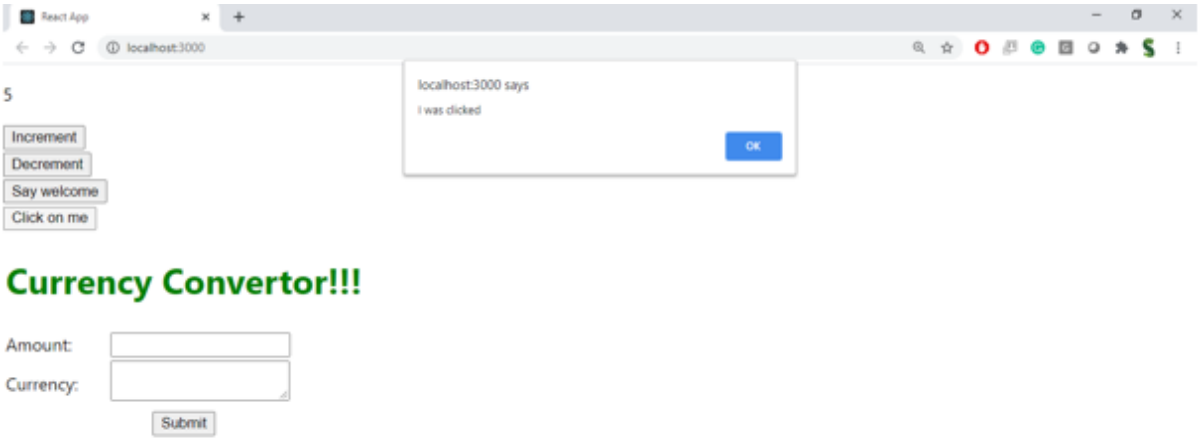
1. **Create “Increment” button to increase the value of the counter and “Decrement” button to decrease the value of the counter. The “Increase” button should invoke multiple methods.**
    a. **To increment the value**
    b. **Say Hello followed by a static message.**



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

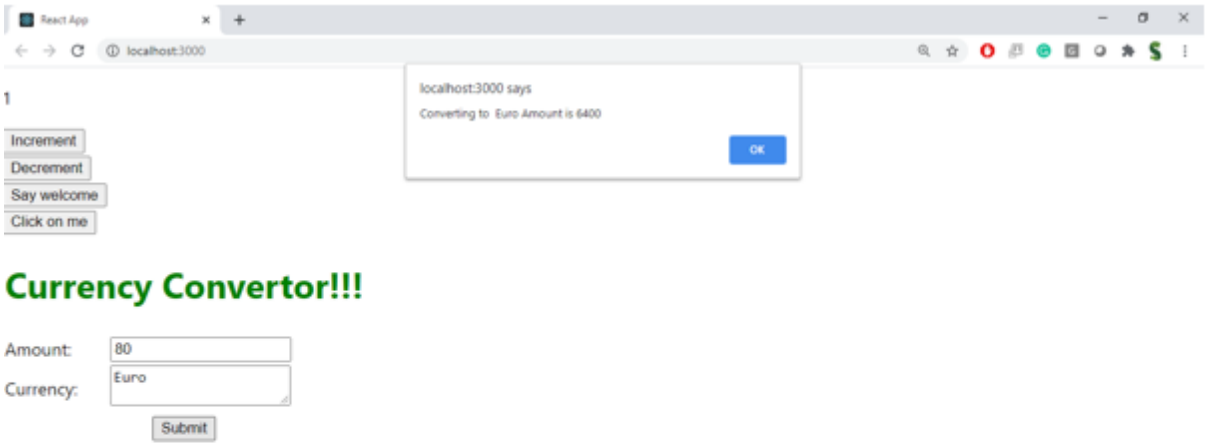


3. **Create a button which invokes synthetic event "OnPress" which display "I was clicked"**



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.



Answer:

1. React Events

React events are JavaScript-based interactions (like clicks or key presses) handled using React's own event system. They ensure consistent behavior across all browsers.

2. Event Handlers

Event handlers are functions defined in React to respond to specific events, such as clicks, form submissions, or input changes.

3. Synthetic Event

A SyntheticEvent is a cross-browser wrapper around the browser's native event. It provides a consistent interface and includes standard event methods.

4. React Event Naming Convention

React uses camelCase for event names (e.g., onClick, onChange) and assigns event handlers using functions, not strings.

CurrencyConvertor.js:

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

```
class CurrencyConvertor extends Component {
  constructor(props) {
    super(props);
    this.state = {
      amount: '',
      currency: 'Euro'
    };
  }

  handleAmountChange = (e) => {
    this.setState({ amount: e.target.value });
  };

  handleSubmit = (e) => {
    e.preventDefault();
    const { amount, currency } = this.state;
    let result = 0;

    if (currency.toLowerCase() === 'euro') {
      result = amount * 80;
    }

    alert(`Converting to ${currency} Amount is ${result}`);
  };

  render() {
    return (
      <div>
        <h1 style={{ color: "green" }}>Currency Convertor!!!</h1>
        <form onSubmit={this.handleSubmit}>
```

```
{/* Amount Field */}
<div style={{ marginBottom: '10px' }}>
  <label style={{ marginRight: '10px', display: 'inline-block', width: '80px' }}>
    Amount:
  </label>
  <input
    type="text"
    value={this.state.amount}
    onChange={this.handleAmountChange}
    style={{
      width: '200px',
      height: '20px',
      padding: '5px',
      borderRadius: '4px',
      border: '1px solid #ccc',
    }}
  />
</div>

{}
<div style={{ marginBottom: '10px' }}>
  <label style={{ marginRight: '10px', display: 'inline-block', width: '80px' }}>
    Currency:
  </label>
  <input
    type="text"
    style={{
      width: '200px',
      height: '30px',
      padding: '5px',
      borderRadius: '4px',
```

```
            border: '1px solid #ccc'
          }}
         />
        </div>

        {}
        <button type="submit">Submit</button>
       </form>
     </div>
   );
  }
}

export default CurrencyConvertor;
```

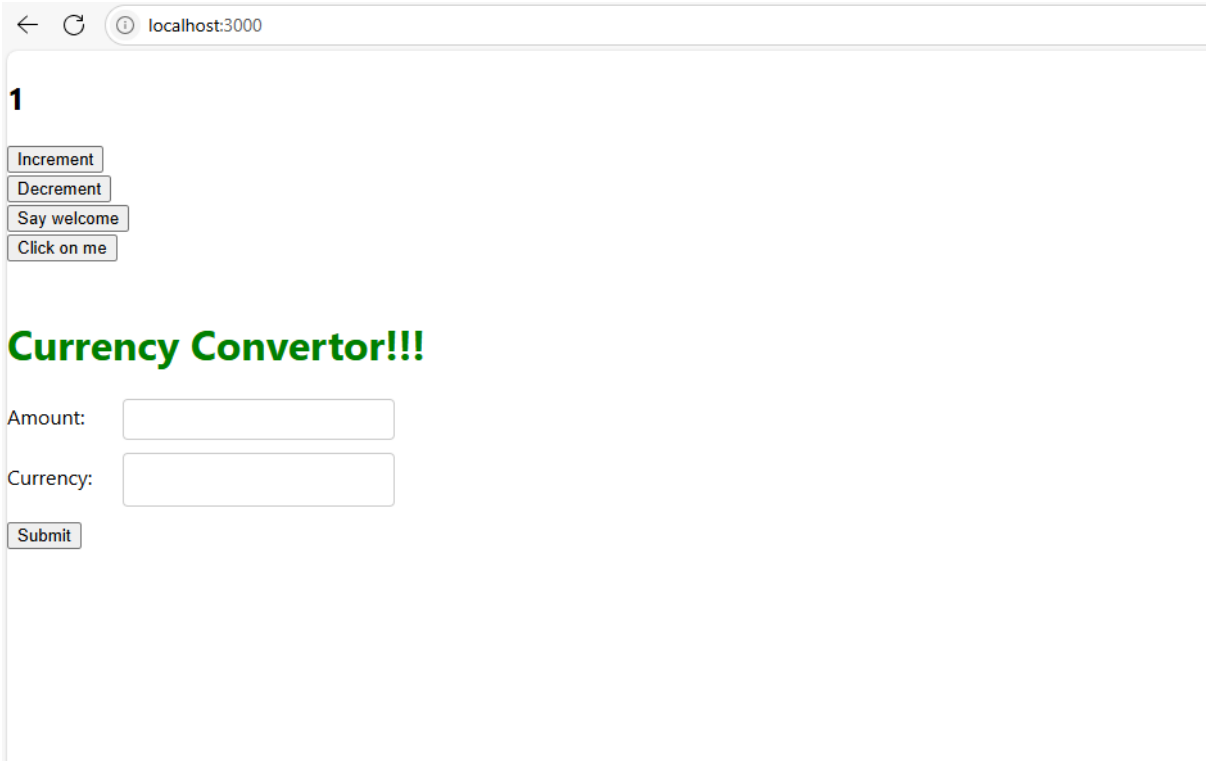
App.js:

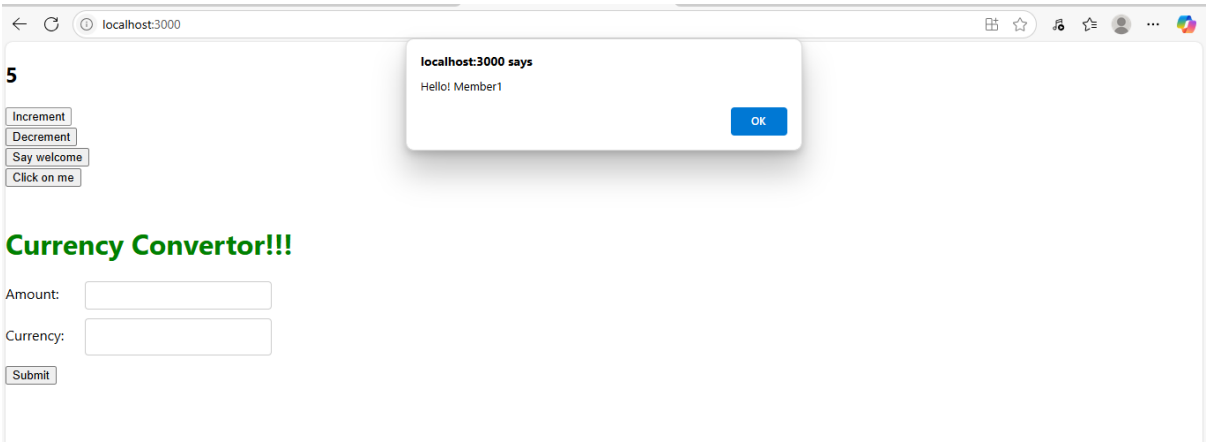
```
import React, { Component } from 'react';
import CurrencyConvertor from './Component/CurrencyConvertor';

class App extends Component {
  constructor(props) {
    super(props);
    this.state = {
      count: 1
    };
  }

  increment = () => {
    this.setState({ count: this.state.count + 1 });
    this.sayHello();
  };
```

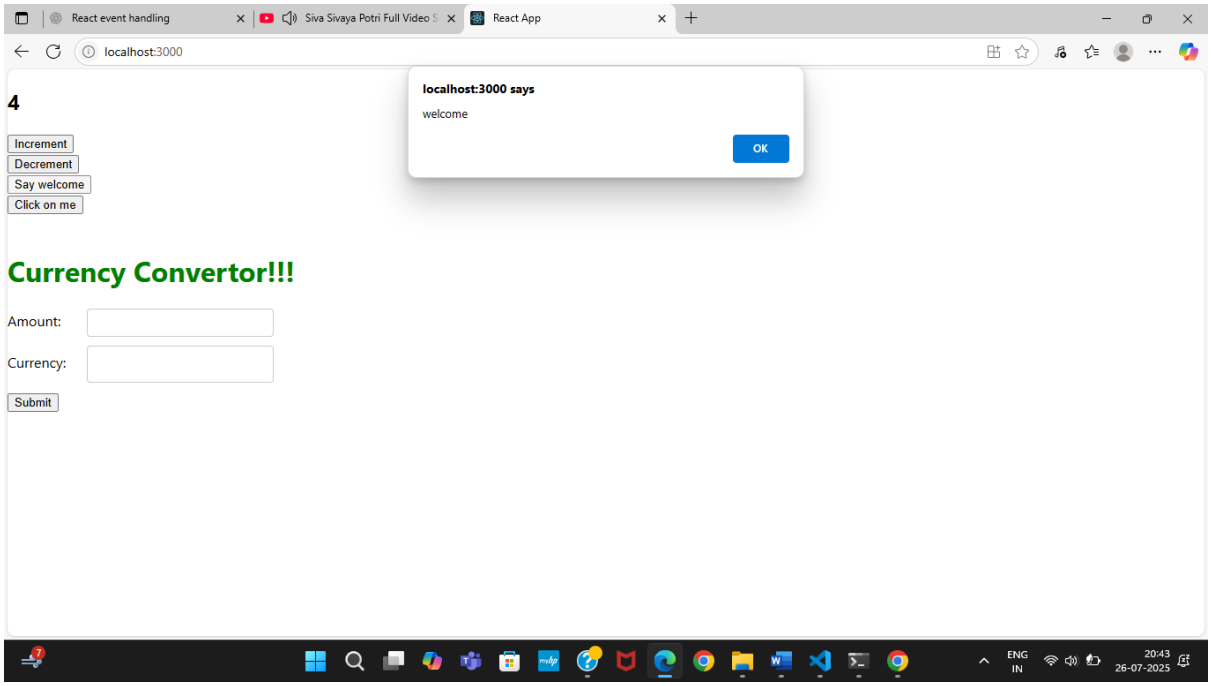
```
decrement = () => {
  this.setState({ count: this.state.count - 1 });
};

sayHello = () => {
  alert("Hello! Member1");
};

sayWelcome = (msg) => {
  alert(msg);
};

handleClick = (e) => {
  alert("I was clicked");
  console.log(e);
};

render() {
  return (
    <div>
      <h2>{this.state.count}</h2>
      <button onClick={this.increment}>Increment</button>
      <br />
      <button onClick={this.decrement}>Decrement</button>
      <br />
      <button onClick={() => this.sayWelcome("welcome")}>Say welcome</button>
      <br />
      <button onClick={this.handleClick}>Click on me</button>
      <br /><br />
      <CurrencyConvertor />
    </div>
```

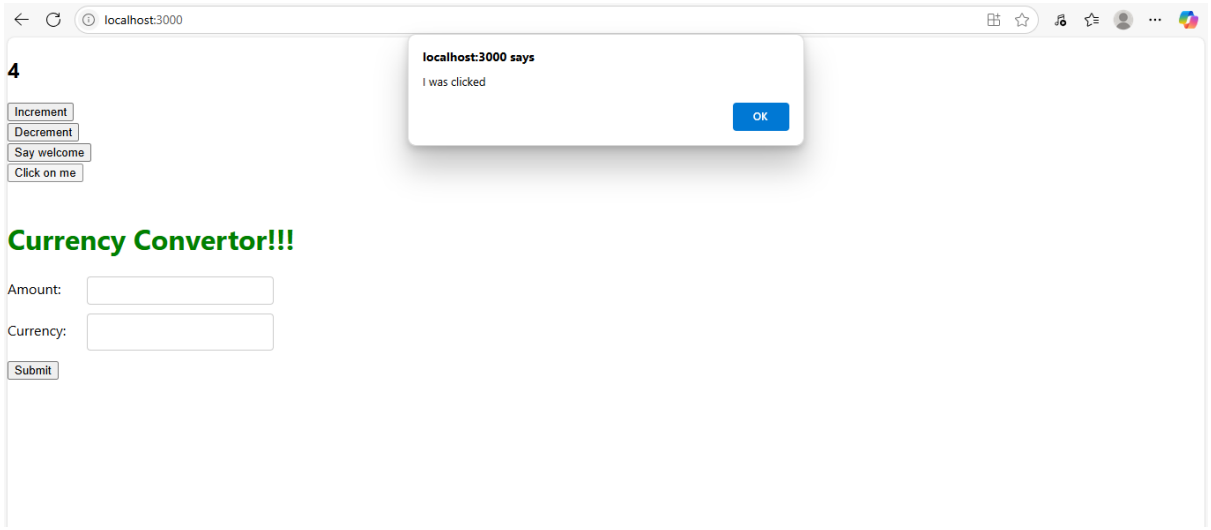
```
    );
  }
}

export default App;
```

Output:









4. 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:

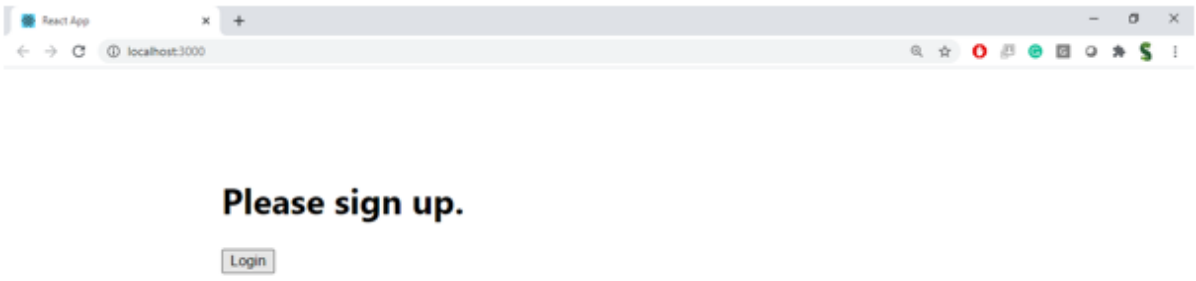
- **Node.js**
- **NPM**
- **Visual Studio Code**

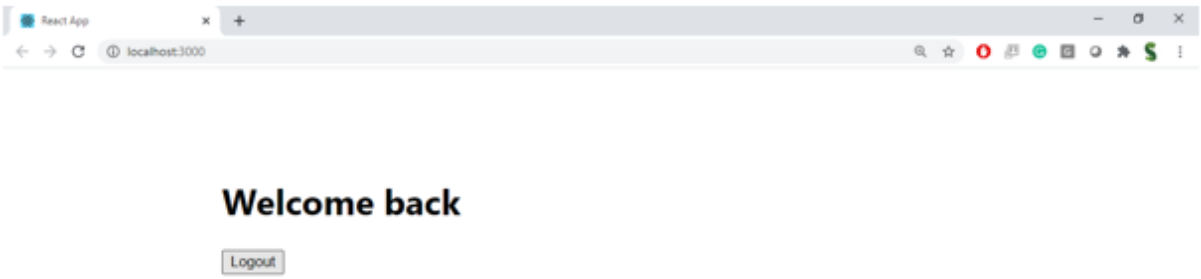
Notes

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.





Hint:

```
function LoginButton(props) {
  return (
    <button onClick={props.onClick}>
      Login
    </button>
  );
}
```

```
function LogoutButton(props) {
  return (
    <button onClick={props.onClick}>
      Logout
    </button>
  );
```

```
function Greeting(props) {
  const isLoggedIn = props.isLoggedIn;
  if (isLoggedIn) {
    return <UserGreeting />;
  }
  return <GuestGreeting />;
}
```

Answer:

1.Conditional Rendering in React

Conditional rendering is the process of displaying different content based on conditions, such as a user's login status. React evaluates conditions and decides which components or elements to display dynamically during rendering.

2.Element Variables

Element variables are JavaScript variables that store JSX elements. They allow React to assign different components or elements to a variable based on logic, making the render method cleaner and more flexible.

3.Preventing Components from Rendering

To prevent a component from rendering, React allows you to return null from the component. This stops the component from producing any HTML in the DOM, effectively hiding it without removing it from the component tree.

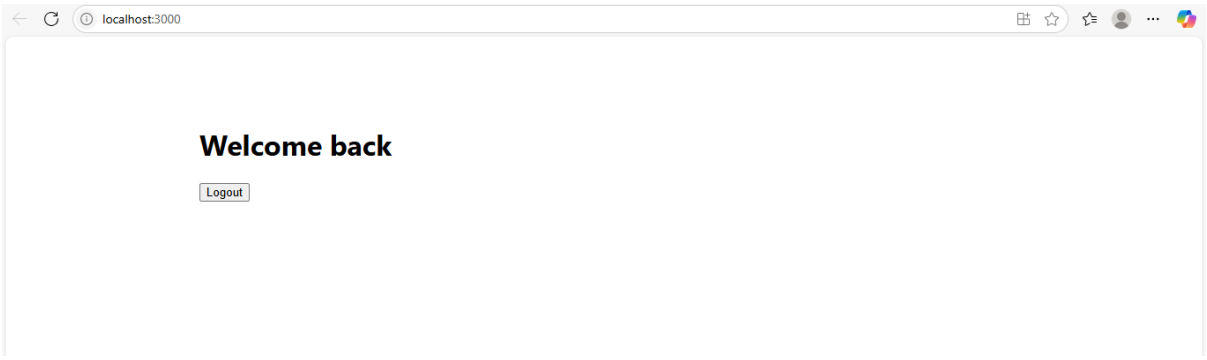
App.js:

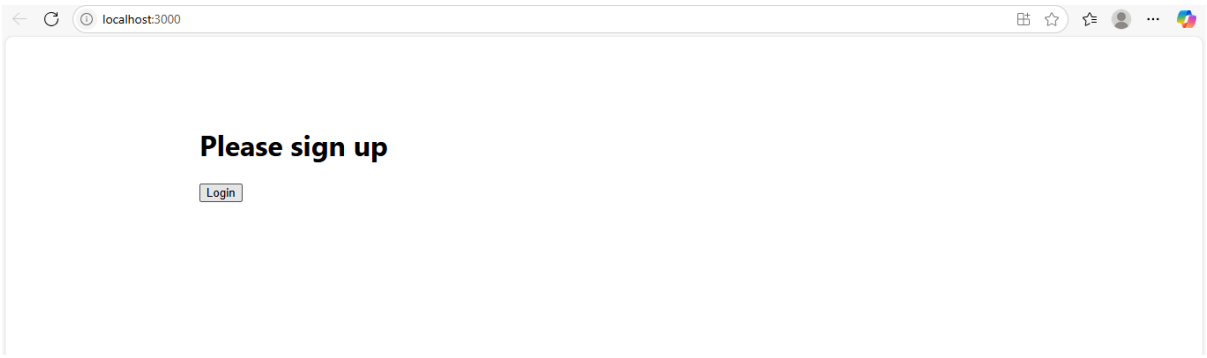
```
import React, { useState } from 'react';


function LoginButton(props) {
  return (
    <button onClick={props.onClick}>
      Login
    </button>
  );
}
```

```
function LogoutButton(props) {
  return (
    <button onClick={props.onClick}>
      Logout
    </button>
  );
}

function UserGreeting() {
  return (
    <div>
      <h1>Welcome back</h1>

      {/* Example flight details */}

    </div>
  );
}

function GuestGreeting() {
  return (
    <div>
      <h1>Please sign up</h1>

      {/* Example flight details visible to all */}

    </div>
  );
}
```

```
function Greeting(props) {
  const isLoggedIn = props.isLoggedIn;
  if (isLoggedIn) {
    return <UserGreeting />;
  }
  return <GuestGreeting />;
}

function App() {
  const [isLoggedIn, setIsLoggedIn] = useState(false);

  let button;
  if (isLoggedIn) {
    button = <LogoutButton onClick={() => setIsLoggedIn(false)} />;
  } else {
    button = <LoginButton onClick={() => setIsLoggedIn(true)} />;
  }

return (
  <div style={{ marginTop: '100px', marginLeft: '220px', textAlign: 'left' }}>
    <Greeting isLoggedIn={isLoggedIn} />
    {button}
  </div>
);

}

export default App;
```

Output:





5. 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**

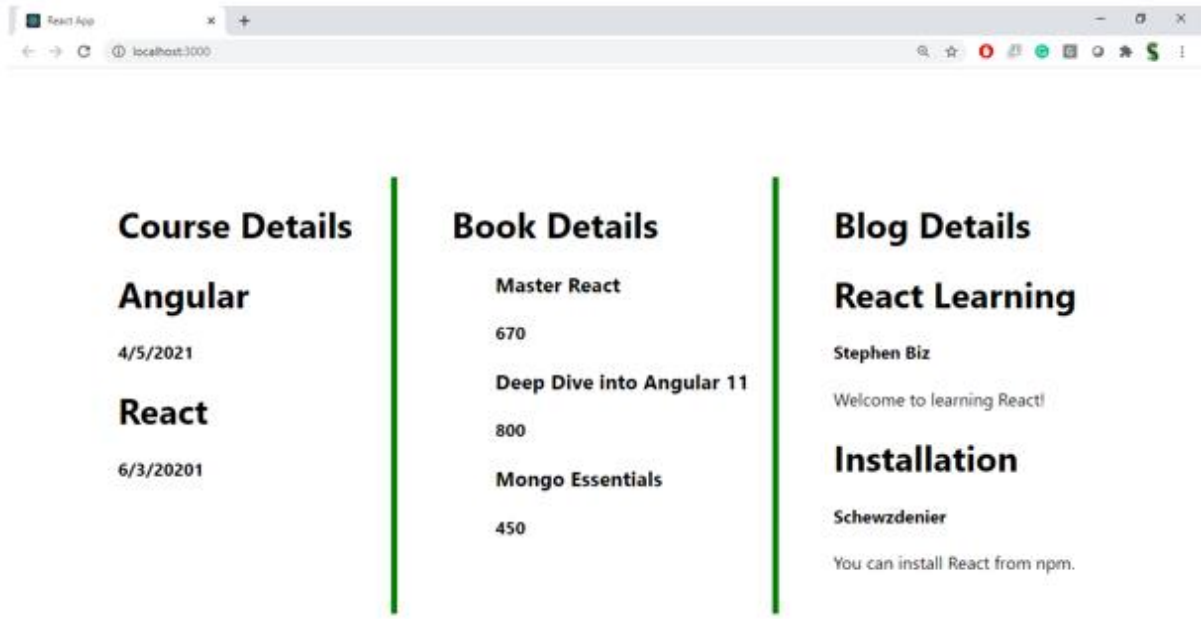
Notes

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.



Hint:

```
const bookdet = (
    <ul>
     {props.books.map((book) =>
      <div key={book.id}>
      <h3>  {book.bname}</h3>
        <h4>{book.price}</h4>
      </div>
     )}
    </ul>
  );
```

```
return (
    <div>
    <div>
    <div className="st2">
     <h1> Book Details</h1>
     {bookdet}
     </div>
     <div className="v1">
     <h1> Blog Details</h1>
     {content}
     </div>
     <div className="mystyle1">
      <h1> Course Details</h1>
     {coursedet}
    </div>
    </div>
    </div>
  );
 }
```

```
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},
];
```

Answer:

1. Various Ways of Conditional Rendering

Conditional rendering means showing different UI parts based on a condition. React allows you to render conditionally using if statements, the && operator for short conditions, the ternary (? :) operator for inline decisions, and separate functions for complex logic.

2. How to Render Multiple Components

Rendering multiple components means displaying more than one component at a time in a single UI. This is done by including all components within a common container so they appear together in the output.

3. List Component

A list component is a React component used to display a list of items such as books or products. It takes an array of data and returns multiple elements, usually by looping through that array.

4. Keys in React Applications

Keys are unique identifiers assigned to items in a list. They help React track and manage updates, insertions, and deletions efficiently. Using keys improves performance and helps prevent UI bugs.

5. Extract Components with Keys

When a list item becomes complex, it is better to move it into its own component. During this process, the key should still be applied when rendering the list, not inside the extracted component**.**

6. React Map, map() Function

The map() function is used to loop over arrays and create elements for each item. It helps in dynamically generating lists and is commonly used to render multiple components based on data.

App.js:

```
import React from 'react';

import './App.css';

import BookDetails from './Components/BookDetails';

import BlogDetails from './Components/BlogDetails';

import CourseDetails from './Components/CourseDetails';


function App() {

  return (
```

```
    <div className="container">
      <CourseDetails />
      <BookDetails />
      <BlogDetails />
    </div>
  );
}

export default App;
```

CourseDetails.js:

```
import React from 'react';

function CourseDetails() {
  const courses = [
    { name: 'Angular', date: '4/5/2021' },
    { name: 'React', date: '6/3/20201' }
  ];

  return (
    <div >
      <h2>Course Details</h2>
      {courses.map((course, index) => (
        <div key={index} >
          <h2>{course.name}</h2>
          <h5>{course.date}</h5>

        </div>
      ))}
    </div>
  );
```

```
}

export default CourseDetails;
```

BookDetails.js:

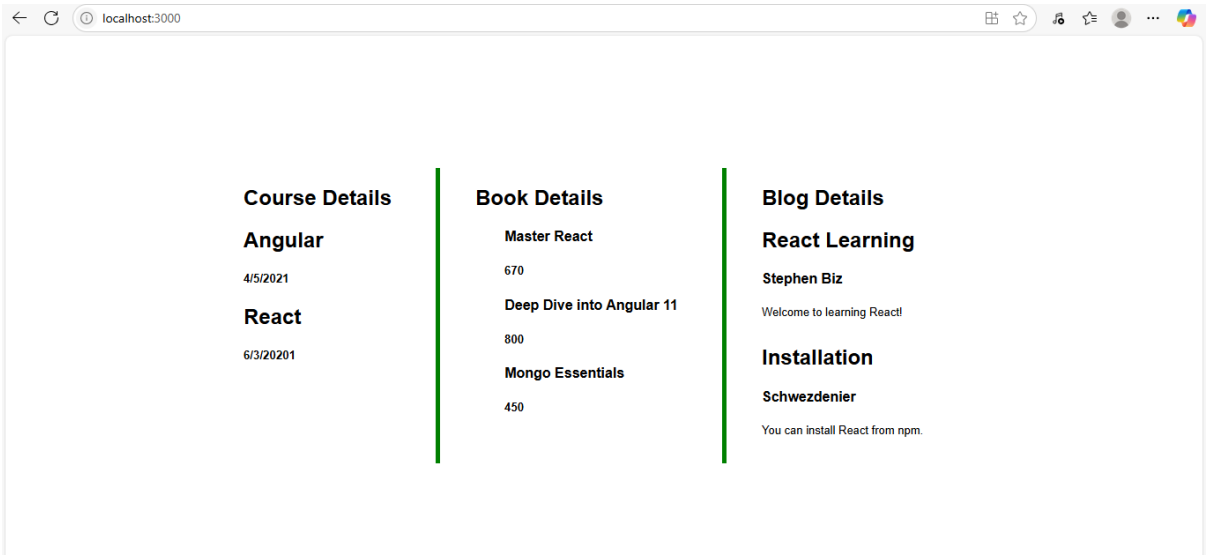
```
import React from 'react';

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 },
];

function BookDetails() {
  return (
    <div className="section">
      <h2 style={{ marginLeft: '20px' }}>Book Details</h2>
      {books.map(book => (
        <div key={book.id}>
          <h4 style={{ marginLeft: '53.2px' }}>{book.bname}</h4>
          <h5 style={{ marginLeft: '53px' }}>{book.price}</h5>

        </div>
      ))}
    </div>
  );
}

export default BookDetails
```

BlogDetails.js:

```
import React from 'react';
```

```
function BlogDetails() {
  const blogs = [
    {
      title: 'React Learning',
      author: 'Stephen Biz',
      content: 'Welcome to learning React!'
    },
    {
      title: 'Installation',
      author: 'Schwezdenier',
      content: 'You can install React from npm.'
    }
  ];

  return (
    <div className="section">
      <h2 style={{ marginLeft: '20px' }}>Blog Details</h2>
      {blogs.map((blog, index) => (
        <div key={index} className="item">
          <h2>{blog.title}</h2>
          <h4>{blog.author}</h4>
          <p style={{ fontSize: '13px' }}>{blog.content}</p>
        </div>
      ))}
    </div>
  );
}

export default BlogDetails;
```

Output:



6. 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.

Application can be downloaded from below

1. **Unzip the application and open it using VS Code**
2. **Go to terminal and execute *npm install* command to restore all the node modules**

Figure 1: Restore node modules

3. **Run the application once to see the output. Use npm start command.**

Figure 2: Starting application

4. **Explore the components present in App.js, EmployeesList.js and EmployeeCard.js files.**
5. **Create a new file with the name as ThemeContext.js. Define a new context in the file with the name as ThemeContext and assign it a default value of 'light' and export it as default form the module.**
6. **Open App component present in App.js file.**
    a. **Import the ThemeContext in App component.**
    b. **Define the theme context provider to be the entire JSX of the App component.**
    c. **Assign the value for the theme provider from the state of the component.**
    d. **Modify the call to EmployeeList component so that theme name is no longer passed as props.**
7. **Go to EmployeeList component present in EmployeeList.js file and modify it so that theme name is not passed explicitly to its child component.**

8. **Go to EmployeeCard component inside EmployeeCard.js file**
    a. **Import the ThemeContext into the component file**
    b. **Retrieve the value of the context with the help of useContext() and store it in a variable**
    c. **Use the variable to pass the className for the buttons.**

Answer:

1. Need and Benefits of React Context API

Need:
React Context API is needed when data must be accessed by multiple nested components, and passing it manually through each level using props becomes difficult and unorganized.

Benefits:

- Eliminates the problem of prop drilling
- Makes state management simpler across components
- Reduces boilerplate and improves code readability
- Enables sharing global data like themes or user information
- Works with both class and function components

2. Working with createContext()

- createContext() is a function provided by React to create a Context object.
- It allows components to provide and consume shared values.
- It is used in combination with a Provider to send the data down the tree, and a Consumer or useContext() to access the data in nested components.

3.Types of Router Components

1. BrowserRouter
Used for traditional clean URLs using the browser's history.

2. HashRouter
Uses the hash part of the URL (after #) to simulate different pages.

3. Route
Defines the path and which component should render when the URL matches.

4. Link
Provides navigation without refreshing the page.

5. NavLink
Works like Link but provides styling options for active routes.

6. Routes
A wrapper for multiple Route components (used in newer versions of React Router).

7. Navigate
Used to programmatically redirect the user.

8. Outlet
Used to render nested or child routes inside parent routes.

App.js:

```
import React, { useState, useEffect } from 'react';

import EmployeesList from './EmployeesList';

import ThemeContext from './ThemeContext';

import './App.css';


function App() {

 const [theme, setTheme] = useState('light');


 useEffect(() => {

  document.body.className = '';

  document.body.classList.add(theme);

 }, [theme]);


 const toggleTheme = () => {

  setTheme(prev => (prev === 'light' ? 'dark' : 'light'));

 };


 return (

  <ThemeContext.Provider value={theme}>

   <div>

    <h1>Employee Management App</h1>

    <button className={theme} onClick={toggleTheme}>

     Toggle Theme

    </button>

    <EmployeesList />
```

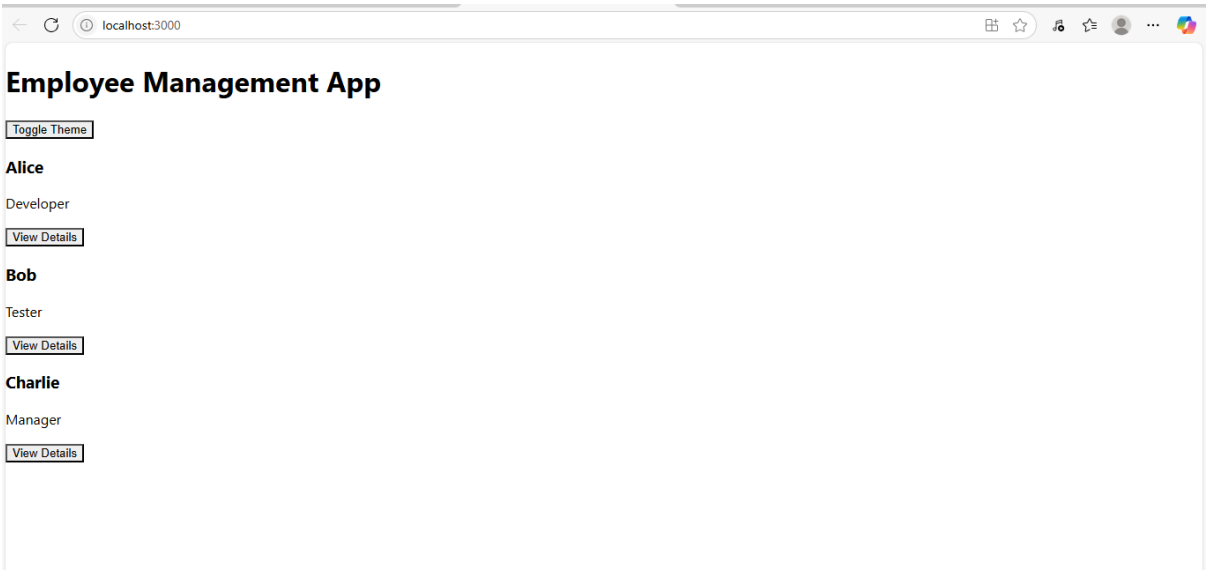
```
    </div>
   </ThemeContext.Provider>
  );
}

export default App;
```

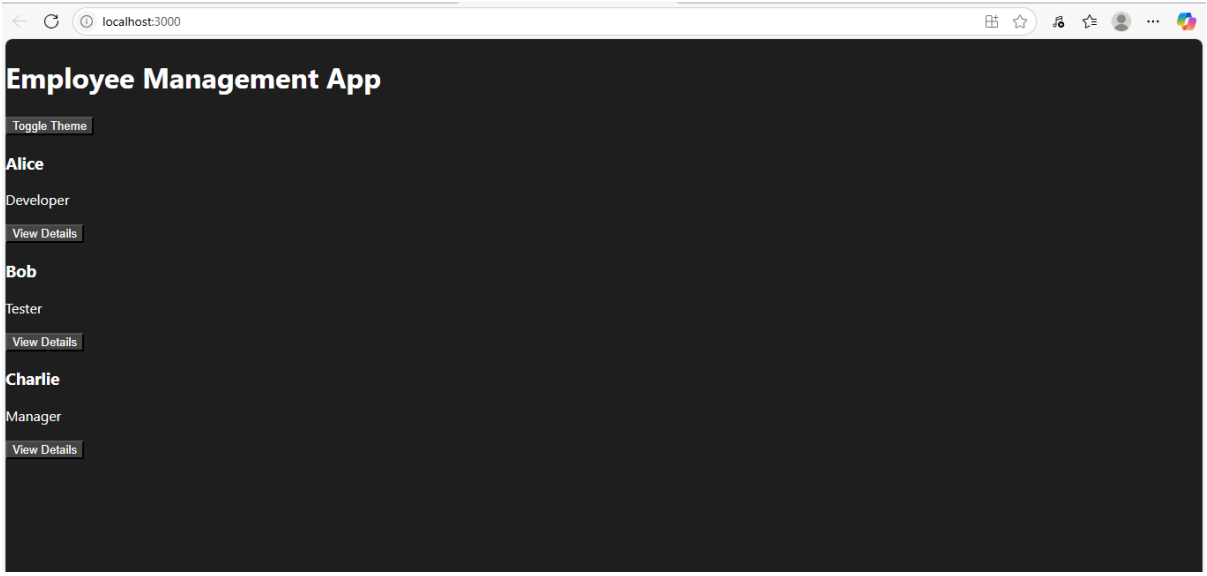
App.css:

```
import React, { useState, useEffect } from 'react';
import EmployeesList from './EmployeesList';
import ThemeContext from './ThemeContext';
import './App.css';

function App() {
  const [theme, setTheme] = useState('light');

  useEffect(() => {
    document.body.className = '';
    document.body.classList.add(theme);
  }, [theme]);

  const toggleTheme = () => {
    setTheme(prev => (prev === 'light' ? 'dark' : 'light'));
  };

  return (
    <ThemeContext.Provider value={theme}>
      <div>
        <h1>Employee Management App</h1>
        <button className={theme} onClick={toggleTheme}>
          Toggle Theme
        </button>
```

```
      <EmployeesList />
    </div>
  </ThemeContext.Provider>
 );
}
export default App;
```

Output:





7. 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**

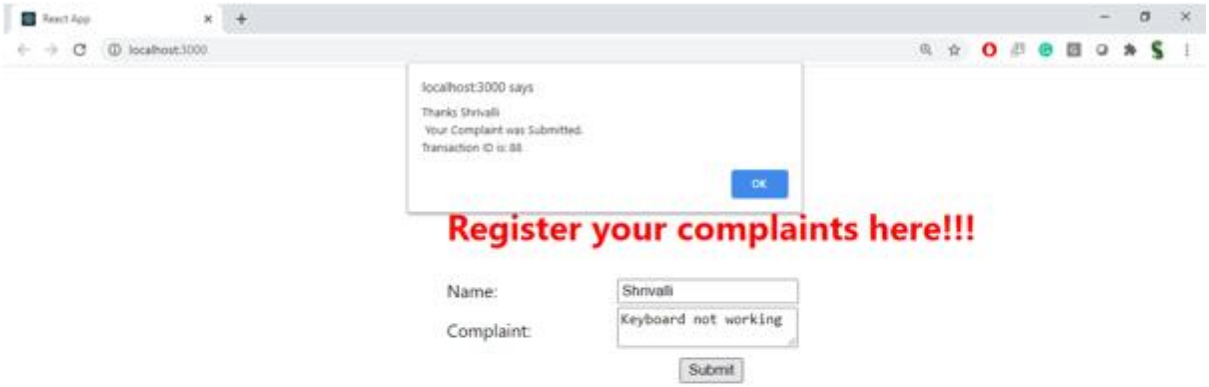
Notes

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.

Output Expected:



Hint:

```
handleChange(event) {
      this.setState({ [event.target.name] : event.target.value});
    }
```

```
handleSubmit(event) {
      var msg='Thanks '+ this.state.ename +' \n Your Complaint was Submit
ID is: ' + this.state.NumberHolder;
      alert(msg);
    event.preventDefault();
  }
```

Answer:

• Explain about React forms

React forms are used to capture user input through HTML form elements like text boxes, checkboxes, radio buttons, and dropdowns. In React, form data is usually handled by the component's state rather than the DOM, which helps in building interactive, data-driven applications.

• Define controlled components

Controlled components are input elements whose values are controlled by React through state. That means the input's value is always driven by the component's state, and any changes to the input are handled by updating the state. This ensures that React is the single source of truth for form data.

• Explain about various input controls

Various input controls include:

- **Text inputs: Used to collect single-line text input.**
- **Textarea: Used for multi-line text input.**
- **Checkbox: Allows users to select true/false options.**
- **Radio buttons: Used when only one option should be selected from multiple.**
- **Select dropdown: Lets users choose one (or more) values from a list.**

Each of these elements can be controlled by React through state and event handling.

• Explain about handling forms

Handling forms in React involves managing input values through state and updating that state using event handlers like onChange. Each time the user types or selects something, the handler updates the state. This keeps the form and the UI in sync with the underlying data.

• Explain about submitting forms

Form submission in React typically uses an event handler like onSubmit. When the user submits the form, this handler is triggered. React prevents the browser's default form submission behavior, and instead handles the data (such as validation or sending it to a server) through JavaScript, usually by calling a function.

ComplaintRegister.css:

```
.form-container {
  text-align: center;
  margin-top: 148px;
}

.title {
  color: red;
  font-size: 36px;
  font-weight: bold;
```

```
  margin-left:220px
}

form {
  display: inline-block;
  text-align: left;
  margin-top: 20px;
}

.form-group {
  margin-bottom: 15px;
  display: flex;
  align-items: center;
}

label {
  width: 100px;
  margin-right: 50px;
  margin-left: 50px;
  text-align: left;
}

input,
textarea {
  width: 200px;
  padding: 5px;
}

button {
  margin-left: 260px;
  padding: 5px 15px;
```

```
}
```

ComplaintRegister.js:

```
import React, { useState } from 'react';
import './ComplaintRegister.css';

function ComplaintRegister() {
 const [name, setName] = useState('');
 const [complaint, setComplaint] = useState('');

 const handleSubmit = (e) => {
  e.preventDefault();
  const transactionId = Math.floor(Math.random() * 100);
  alert(`Thanks ${name}\nYour Complaint was Submitted.\nTransaction ID is: ${transactionId}`);
 };

 return (
  <div className="form-container">
   <h1 className="title">Register your complaints here!!!</h1>
   <form onSubmit={handleSubmit}>
    <div className="form-group">
     <label>Name:</label>
     <input
      type="text"
      value={name}
      onChange={(e) => setName(e.target.value)}
     />
    </div>

    <div className="form-group">
```

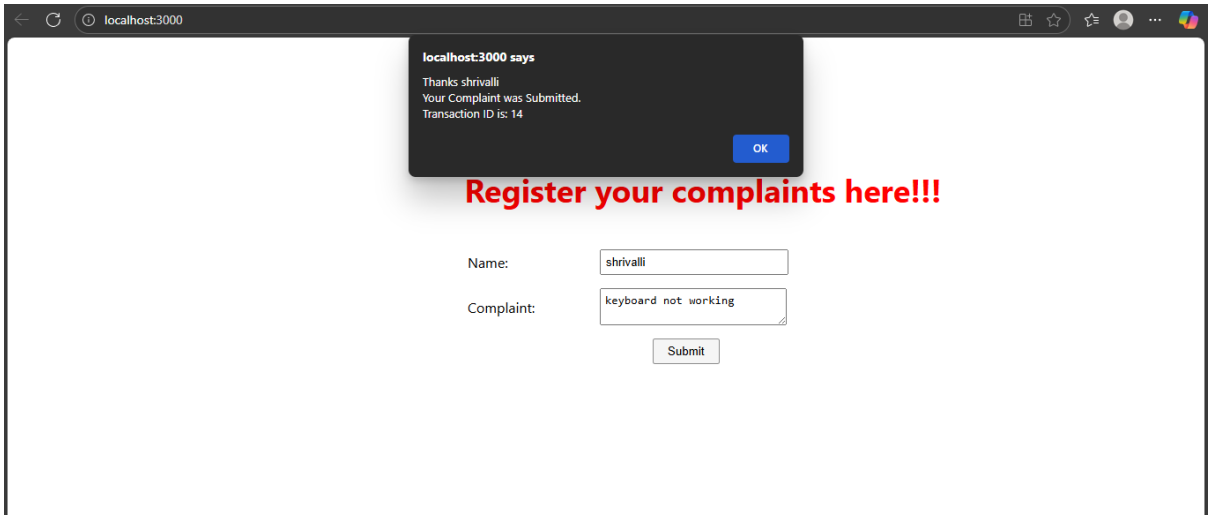
```
        <label>Complaint:</label>
        <textarea
          value={complaint}
          onChange={(e) => setComplaint(e.target.value)}
        />
       </div>

       <button type="submit">Submit</button>
     </form>
   </div>
  );
}

export default ComplaintRegister;
```

App.css:

```
.App {
  text-align: center;
}

.App-logo {
  height: 40vmin;
  pointer-events: none;
}

@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}

.App-header {
```

```
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}

.App-link {
  color: #61dafb;
}

@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}
```

Output:



8. 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

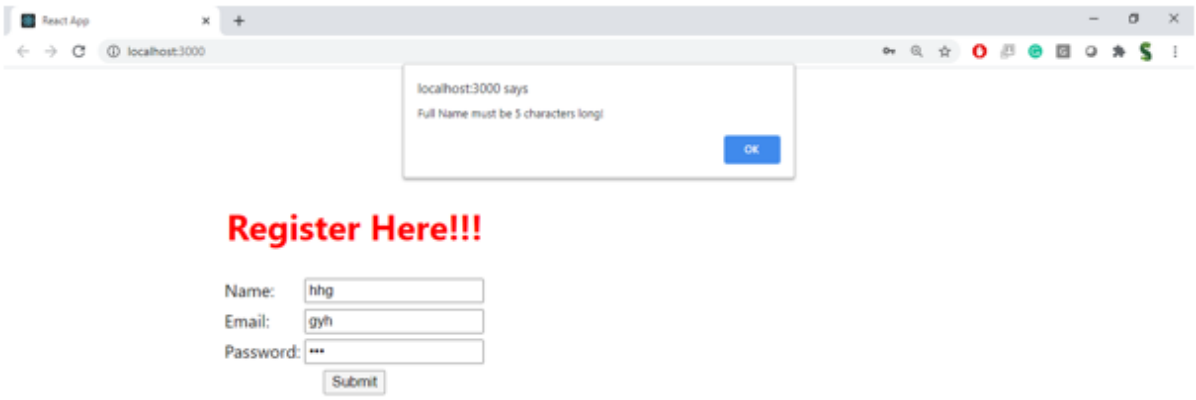
Estimated time to complete this lab: 60 minutes.

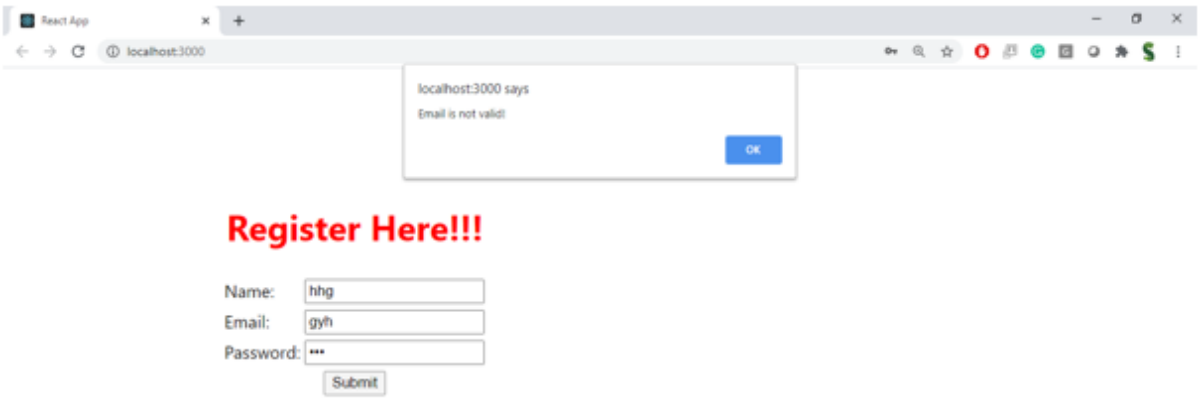
Create a React App named “mailregisterapp” which will have a component named “register.js”. Create a form which accepts the name, email and password and validate the fields as per the following:

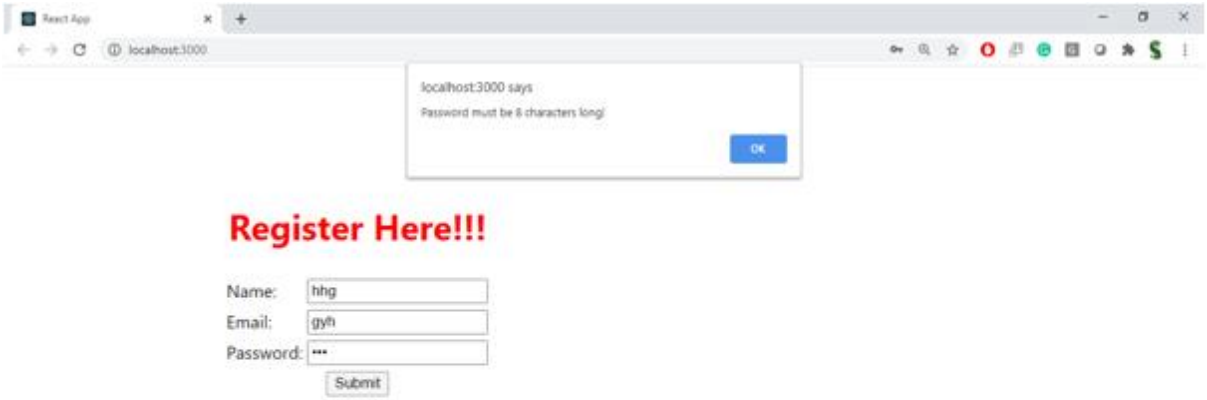
1. **Name should have atleast 5 characters**
2. **Email should have @ and .**
3. **Password should have atleast 8 characters.**

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

Output Expected:




React App
localhost:3000
localhost:3000 says
Email is not valid!
OK
Register Here!!!
Name:
hhg
Email:
gyh
Password:
Submit


React App
localhost:3000
localhost:3000 says
Password must be 8 characters long!
OK
Register Here!!!
Name:
hhg
Email:
gyh
Password:
Submit

Hint:

```
handleSubmit = (event) => {
        event.preventDefault();
        if(validateForm(this.state.errors)) {
          alert('Valid Form')
        }else{
           if(this.state.errors.fullName!='')
           {alert(this.state.errors.fullName)}
           if(this.state.errors.email !='')
          {alert(this.state.errors.email)}
          if(this.state.errors.password !='')
          {alert(this.state.errors.password)}
        }
      }
```

```
switch (name) {
          case 'fullName':
            errors.fullName =
             value.length < 5
               ? 'Full Name must be 5 characters long!'
               : '';
            break;
```

```
case 'email':
        const validEmailRegex =
        RegExp(/^(([^<>()\[\]\.,;:\s@\"]+(\.[^<>()\[\]\.,;:\s@\"]+)*)|(\".+\"))@(([^<>()[\]\.,;:\s@\"]+\.)+[^
<>()[\]\.,;:\s@\"]{2,})$/i);

          errors.email =
            validEmailRegex.test(value)
              ? ''
              : 'Email is not valid!';
          break;
```

Answer:

1. Explain React Forms Validation

React Forms Validation is the process of checking if the user's input is correct and complete before the form is submitted. It helps ensure that values meet required conditions, like the correct format or minimum length.

2. Identify the Differences Between React Form and HTML Form

- **In HTML, form handling and validation are mostly handled by the browser.**
- **In React, form handling and validation are controlled using JavaScript and React state.**
- **HTML forms often reload the page on submit, but React prevents that and handles everything within the app.**
- **React provides more flexibility and control over how data is managed and validated.**

3. Explain About Controlled Components

A controlled component in React is an input element (like a text box) whose value is controlled by React through state. This means React is responsible for updating and maintaining the value, keeping it in sync with what the user types.

4. Identify Various React Form Input Controls

React supports different input types used in forms such as:

- **Text inputs**
- **Email and password fields**
- **Textareas**
- **Checkboxes and radio buttons**
- **Dropdown menus (select options)**
- **Others like date pickers or file upload inputs**

Each of these can be managed through state in React.

5. Explain How to Handle React Forms

Handling React forms involves creating a state to store input values and updating that state whenever the user types something. This ensures the form fields always reflect the latest user input, and React can use the data for validation or submission.

6. Explain About Submitting Forms in React

When submitting a form in React, you prevent the page from reloading and instead handle the form data using a function. This allows you to validate the input, show messages, or send the data to a server — all without leaving the page.

Register.js:

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

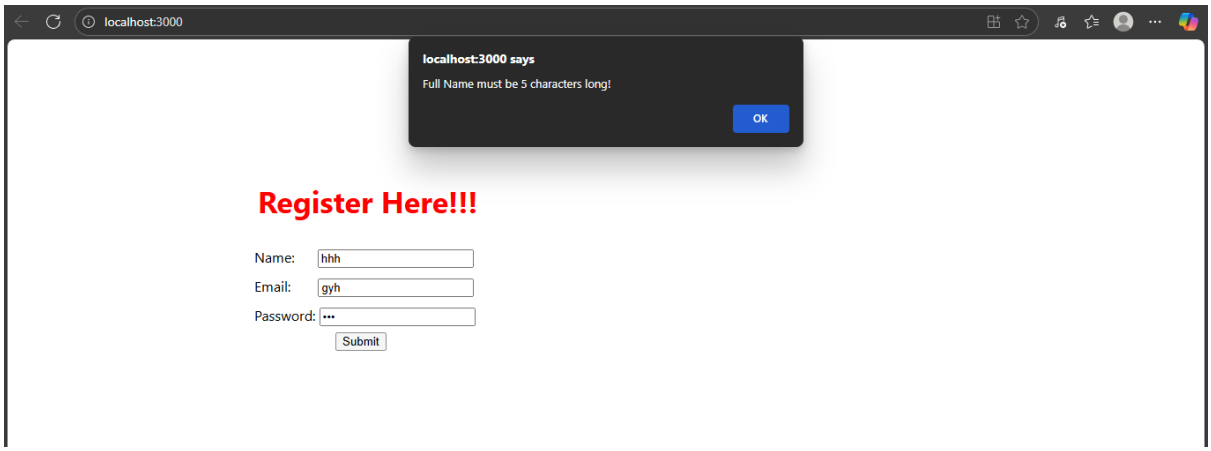
```
class Register extends Component {
  constructor(props) {
    super(props);
    this.state = {
      fullName: '',
      email: '',
      password: '',
      errors: {
        fullName: '',
        email: '',
        password: ''
      }
    };
  }

  handleChange = (event) => {
    const { name, value } = event.target;
    let errors = this.state.errors;

    switch (name) {
      case 'fullName':
        errors.fullName = value.length < 5 ? 'Full Name must be 5 characters long!' : '';
        break;
      case 'email':
        const validEmailRegex = RegExp(/^[^\s@]+@[^\s@]+\.[^\s@]+$/);
        errors.email = validEmailRegex.test(value) ? '' : 'Email is not valid!';
        break;
      case 'password':
        errors.password = value.length < 8 ? 'Password must be 8 characters long!' : '';
        break;
```

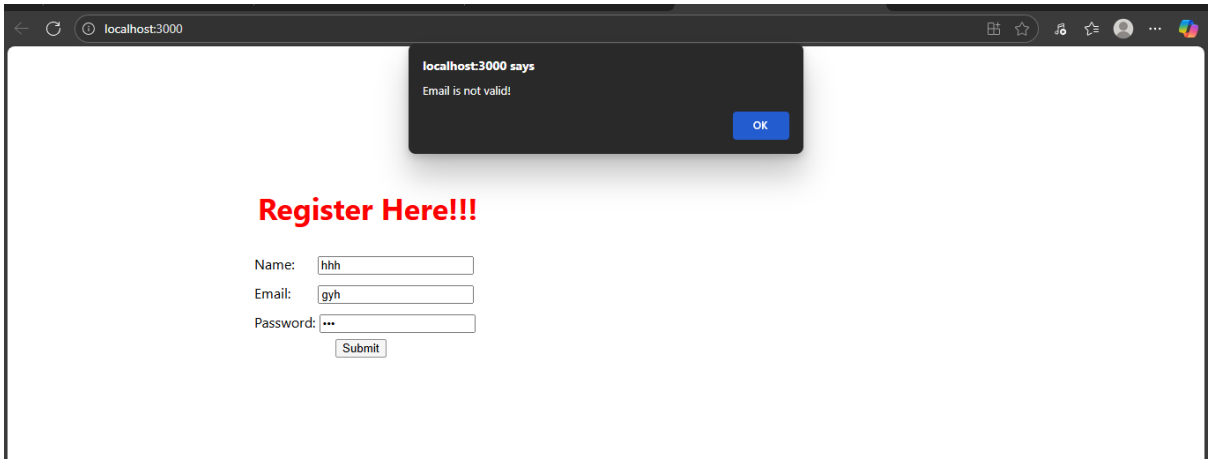
```
    default:
      break;
  }

  this.setState({ errors, [name]: value });
};

validateForm = (errors) => {
  return Object.values(errors).every(val => val === '');
};

handleSubmit = (event) => {
  event.preventDefault();
  if (this.validateForm(this.state.errors)) {
    alert('Valid Form');
  } else {
    if (this.state.errors.fullName !== '') {
      alert(this.state.errors.fullName);
    }
    if (this.state.errors.email !== '') {
      alert(this.state.errors.email);
    }
    if (this.state.errors.password !== '') {
      alert(this.state.errors.password);
    }
  }
};

render() {
  return (
    <div style={{ textAlign: 'left', marginTop: '160px', marginLeft: '270px' }}>
```

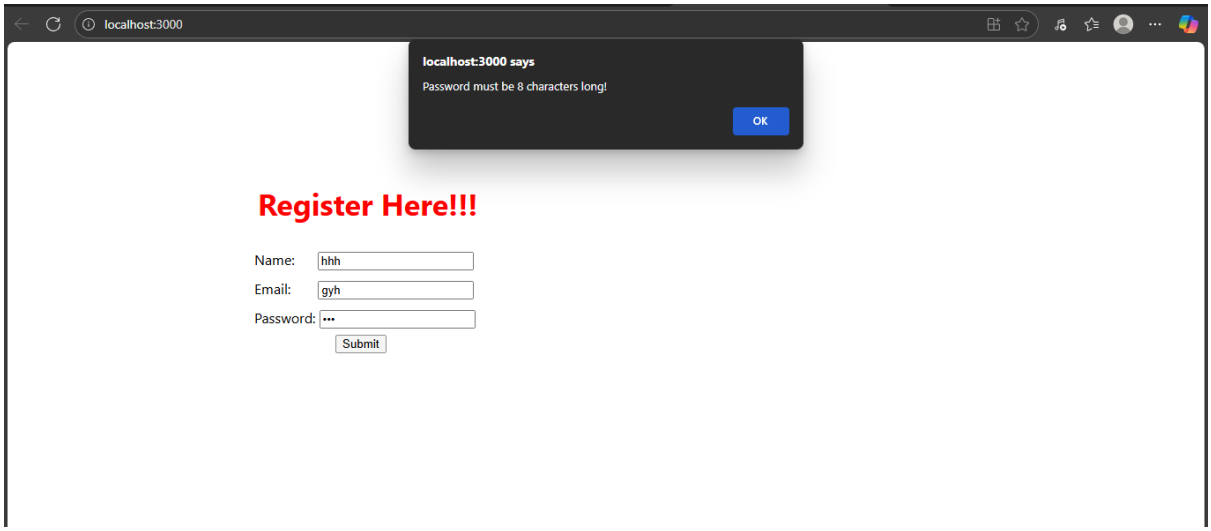
```
    <h1 style={{ color: 'red', marginBottom: '30px', marginLeft: '14px',fontSize: '34px'  }}>Register Here!!!</h1>


      <form onSubmit={this.handleSubmit} noValidate>
        <div style={{ margin: '10px' }}>
          <label style={{ marginRight: '26px' }}>Name:</label>
          <input
            type="text"
            name="fullName"
            value={this.state.fullName}
            onChange={this.handleChange}
            required
          />
        </div>
        <div style={{ margin: '10px' }}>
          <label style={{ marginRight: '31px' }}>Email:</label>
          <input
            type="email"
            name="email"
            value={this.state.email}
            onChange={this.handleChange}
            required
          />
        </div>
        <div style={{ margin: '10px' }}>
          <label style={{ marginRight: '4px' }}>Password:</label>
          <input
            type="password"
            name="password"
            value={this.state.password}
            onChange={this.handleChange}
```

```
          required
         />
        </div>
        <div style={{ marginTop: '-5px', marginLeft: '102px' }}>

         <button type="submit">Submit</button>
        </div>
       </form>
      </div>
    );
  }
}

export default Register;
```

Output:







9. 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.

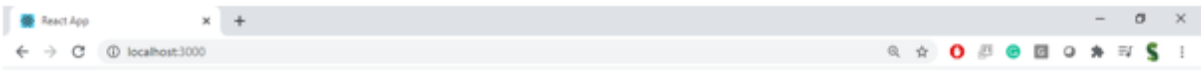
Create a React Application "fetchuserapp" which will retrieve the user details from https://api.randomuser.me/ and display the title, firstname and image of a user.

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:

```
async componentDidMount()
  {
     const url="https://api.randomuser.me/";
     const response=await fetch(url);
     const data=await response.json();
     this.setState({person:data.results[0],loading:false});
     console.log(data.results[0]);
  }
```

Expected Output:



Mr Donato Nunes

Answer:

1. **Understand the Purpose**
   Consuming a REST API means connecting your React app to an external server to get data (like user details, products, posts) or send data (like form submissions).

2. **Use a React Component**
   Create a component where you want to display or use the data from the API.
3. **Choose a Lifecycle/Event Hook**
   - **In class components, you use a lifecycle method to trigger the API call when the component mounts.**
   - **In functional components, you use a hook to do the same.**
4. **Call the API**
   Use built-in tools like fetch() or third-party libraries like axios to send a request to the API endpoint.
5. **Handle the Response**
   Once you get the response from the API, you process the data, typically converting it to JSON.
6. **Store the Data**
   Store the fetched data in React's internal state so the component can use it and re-render if needed.
7. **Render the Data**
   Finally, display the data in the user interface using JSX, depending on what you need from the API.

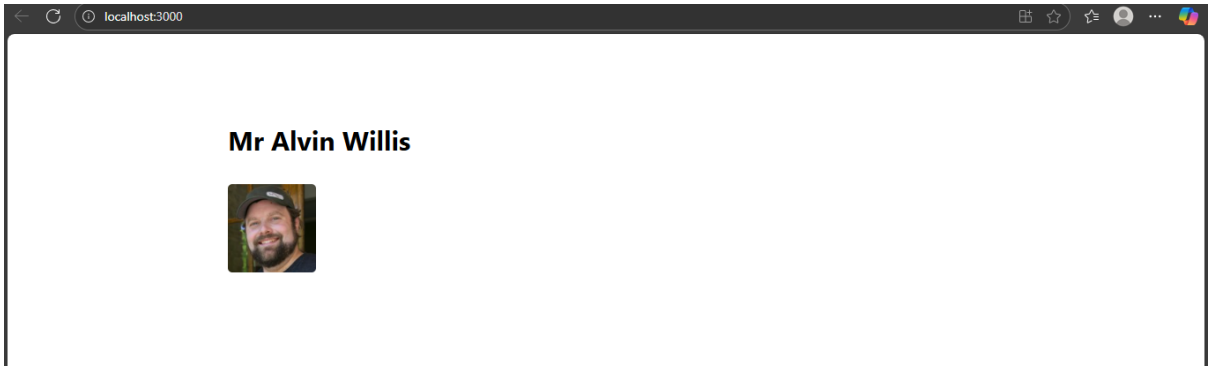
Program:

Getuser.js:

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


class Getuser extends Component {
 constructor(props) {
  super(props);
  this.state = {
   person: null,
   loading: true
  };
 }


 async componentDidMount() {
  const url = "https://api.randomuser.me/";
```

```
    const response = await fetch(url);
    const data = await response.json();
    this.setState({ person: data.results[0], loading: false });
    console.log(data.results[0]); // Debug
  }

  render() {
    const { person, loading } = this.state;

    if (loading) {
      return <h2>Loading...</h2>;
    }

    return (
  <div style={{ textAlign: 'left', marginTop: '100px', marginLeft: '250px' }}>
    <h1 style={{ fontSize: '30px', fontWeight: 'bold' }}>
      {`${person.name.title} ${person.name.first} ${person.name.last}`}
    </h1>
    <img
      src={person.picture.large}
      alt="User"
      style={{ width: '100px', height: '100px', marginTop: '10px', borderRadius: '5px' }}
    />
  </div>
);

  }
}

export default Getuser;
```

Output:


localhost:3000
Mr Alvin Willis