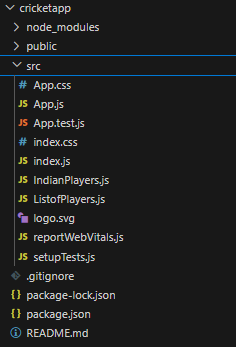
**WEEK - 7**

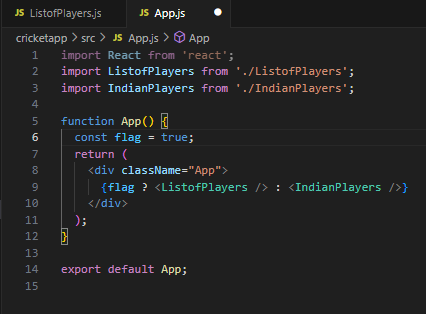
**SKILL : React Mandatory Hands-on exercises**

**Exercise: Create a React Application named “cricketapp” with the list of indian players**

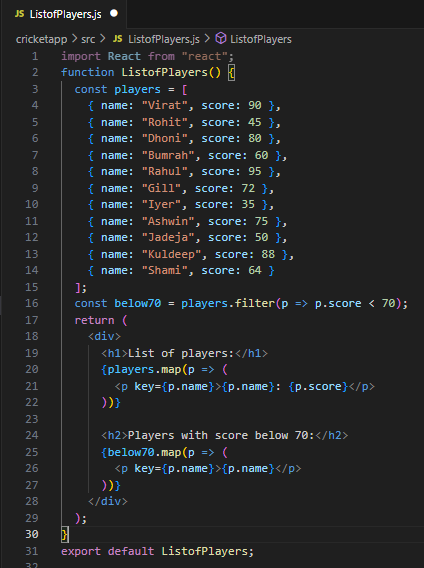
**App Structure:**



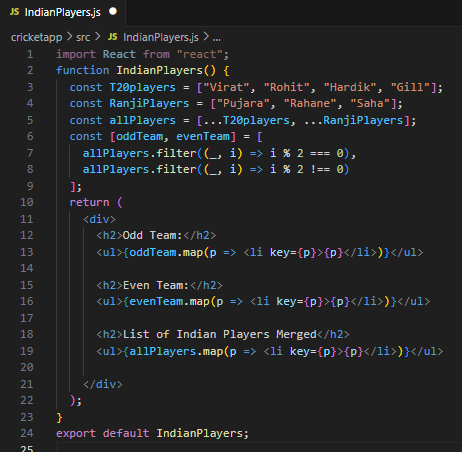
**App.js:**



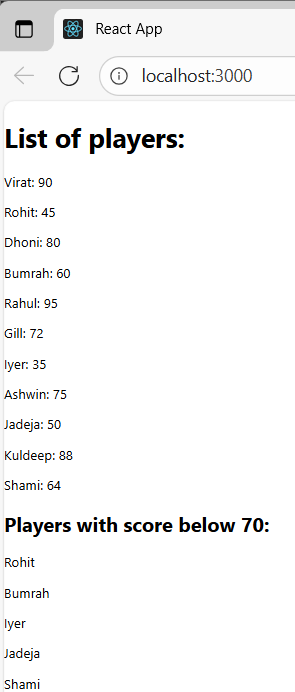
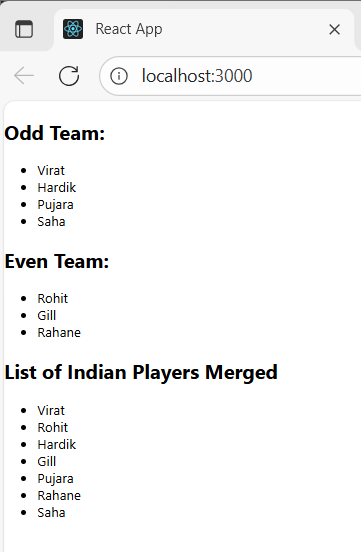
**ListofPlayers.js:**



[**IndianPlayers.js**](http://indianplayers.js/)**:**

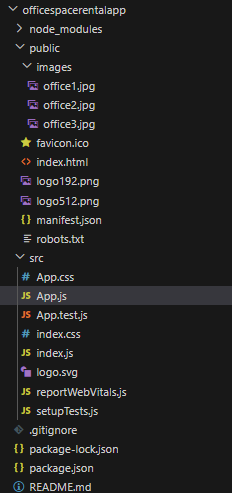


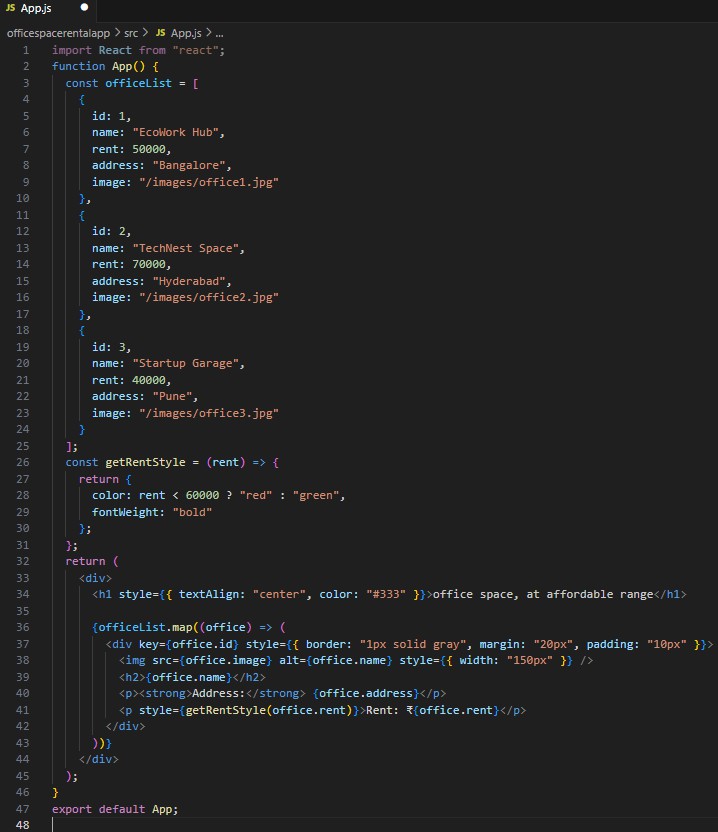
**Output:**

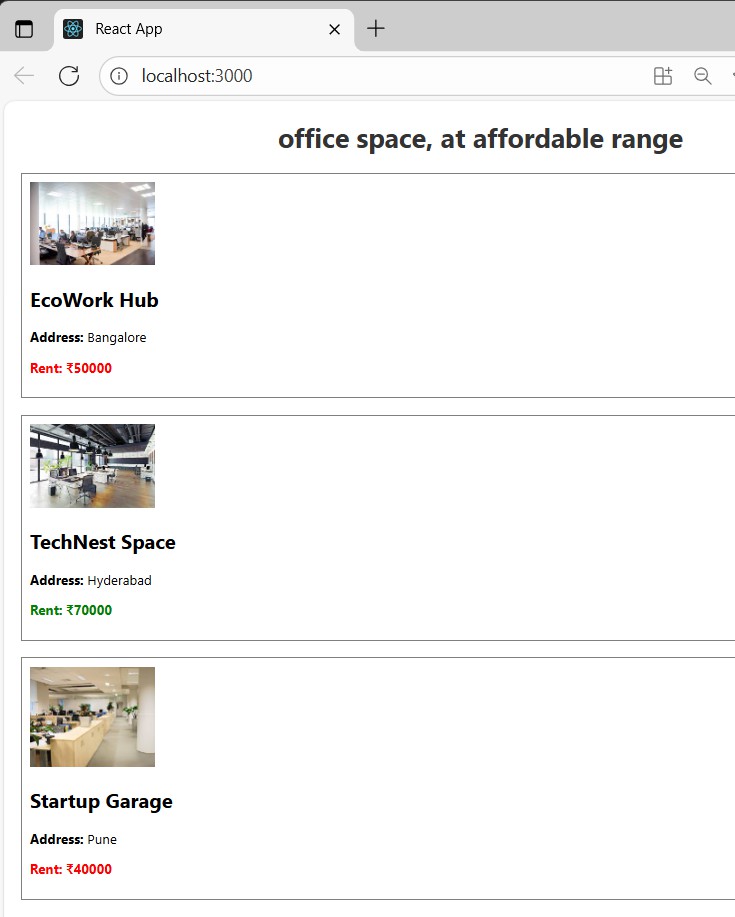


Exercise: Create a React Application named “officespacerentalapp” which uses React JSX to create elements, attributes and renders DOM to display the page.

**App Structure:**

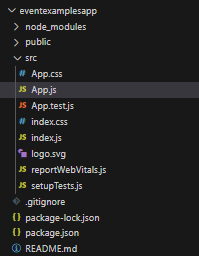


**App.js:**

**Output:**

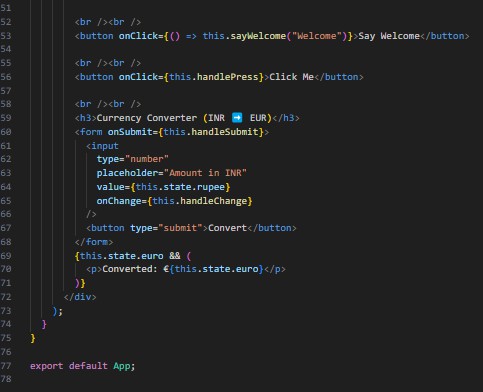
**Exercise: Create a React Application “eventexamplesapp” to handle various events of the form elements in HTML.**

**App Structure:**

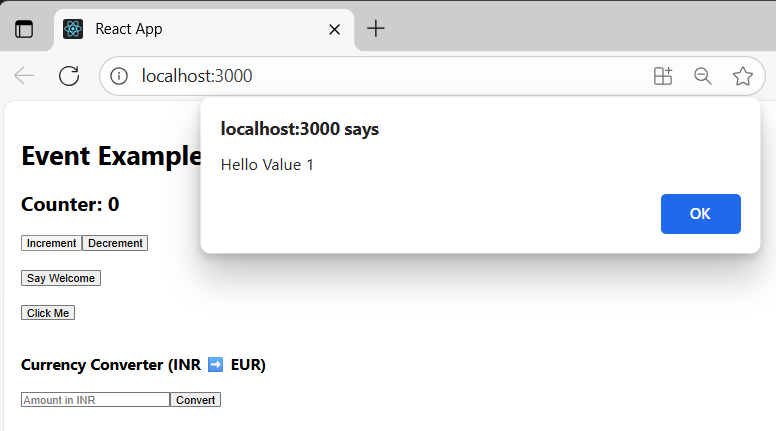


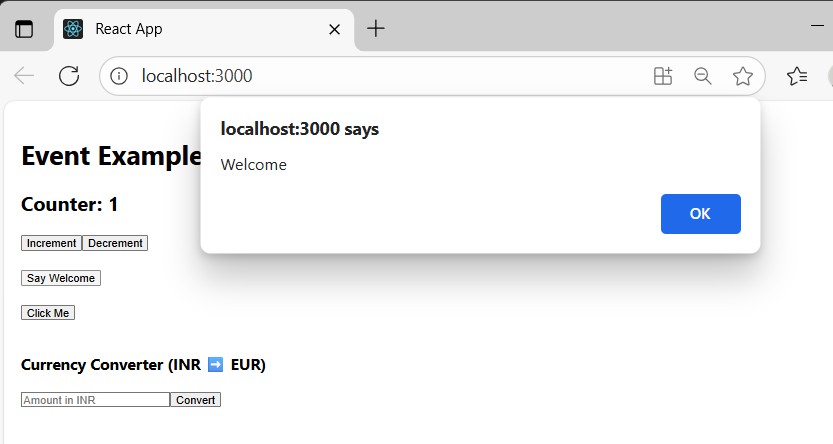
[**App.js**](http://app.js/)**:**

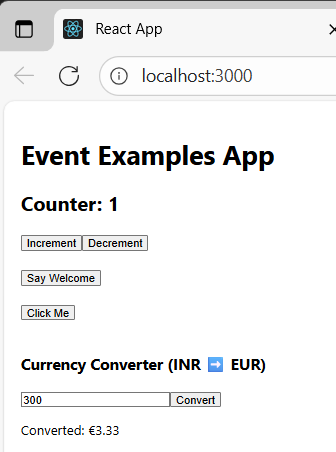
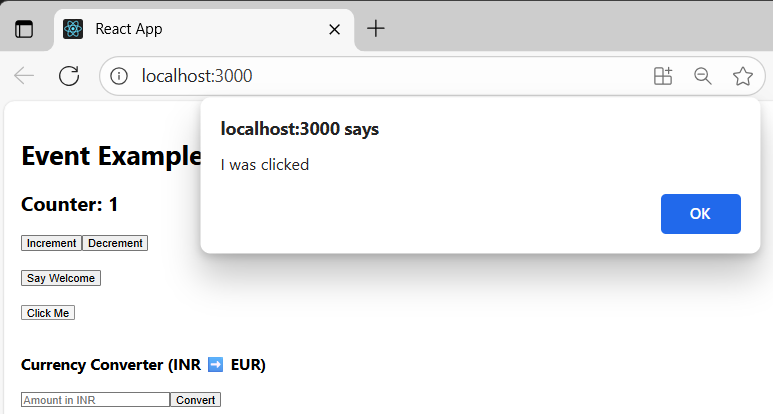




**Output:**



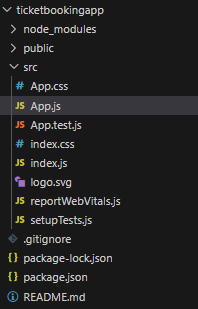




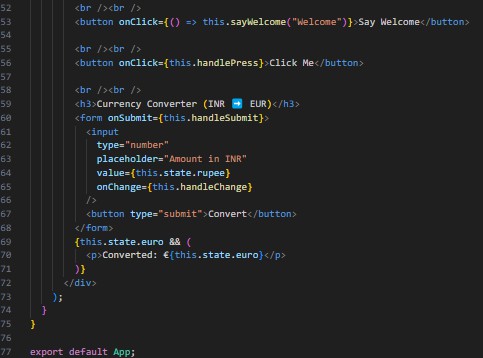
Exercise: Create a React Application named “ticketbookingapp” where the guest user can browse the page where the flight details are displayed whereas the logged in user only can book tickets.

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.

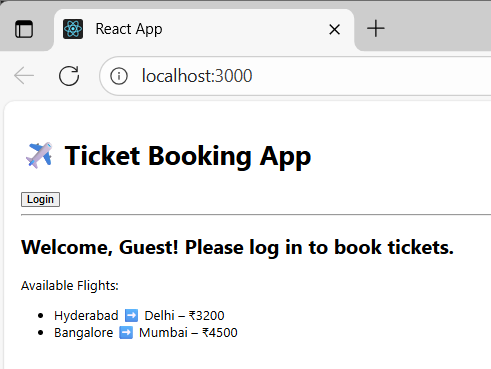
**App Structure:**

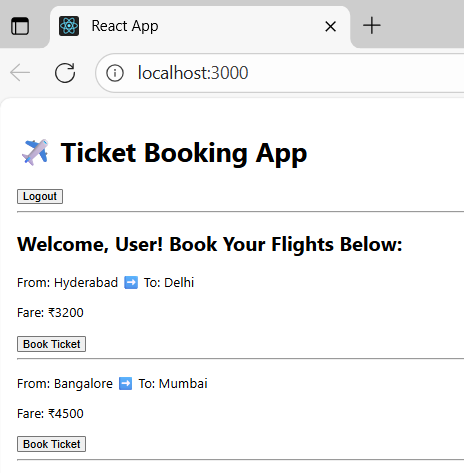


**App.js:**



**Output:**



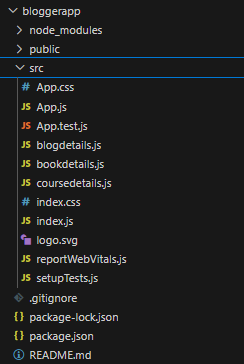


**Exercise:**

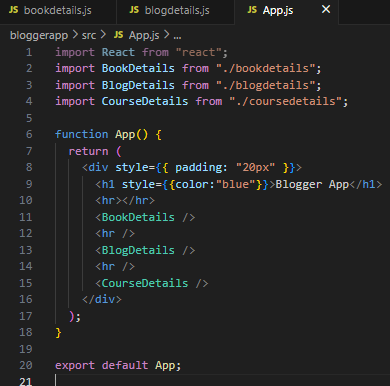
Create a React App named “bloggerapp” in with 3 components.

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

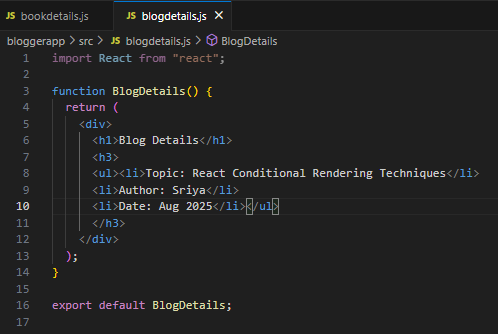
**App Structure:**



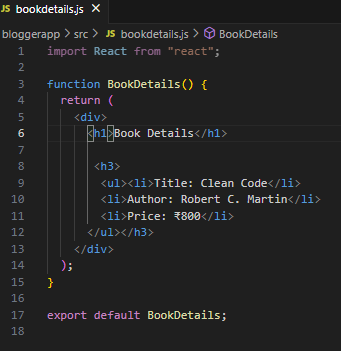
[**App.js**](http://app.js/)**:**



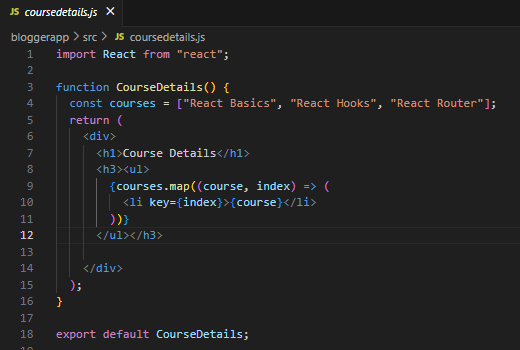
**blogdetails.js:**



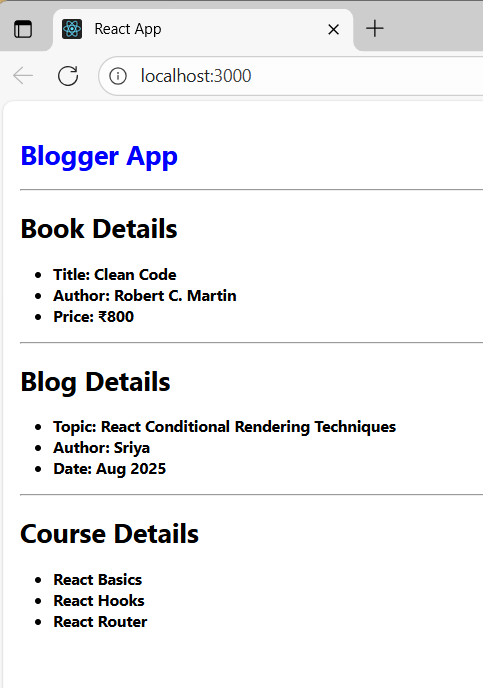
**bookdetails.js:**



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



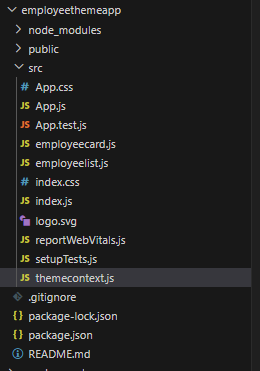
**Output:**



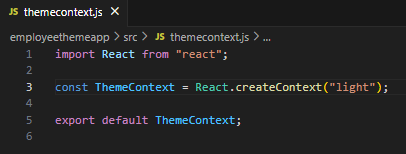
Exercise: Developers of Apps Centric Solutions have created an employee management application which supports light and dark themes for the buttons. The current solution uses the react state and props to provide the theme name to be used from App component to Employee List component and from there to 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.

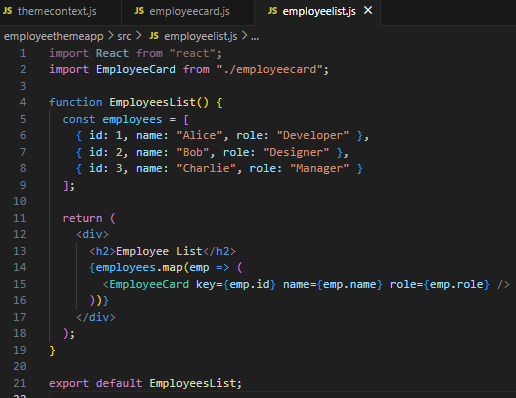
**App Structure:**



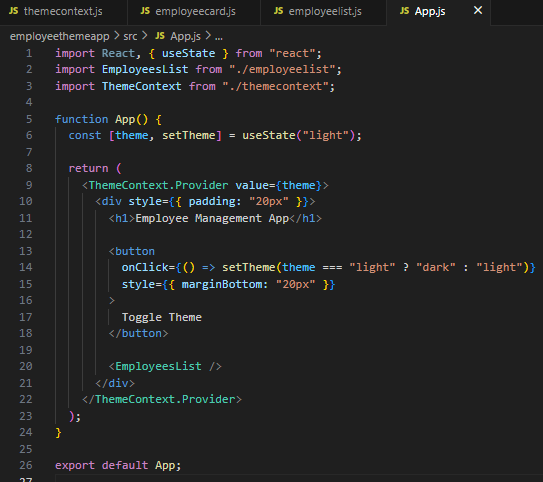
[**themecontext.js**](http://themecontext.js/)**:**



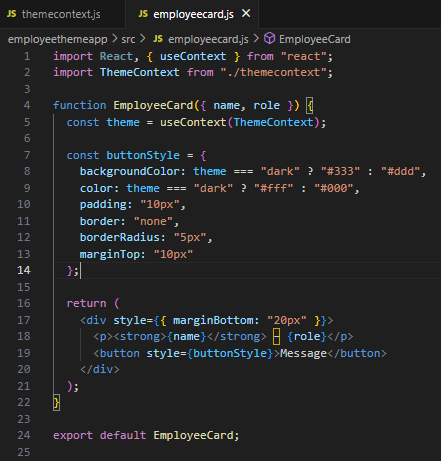
[**employeelist.js**](http://employeelist.js/)**:**



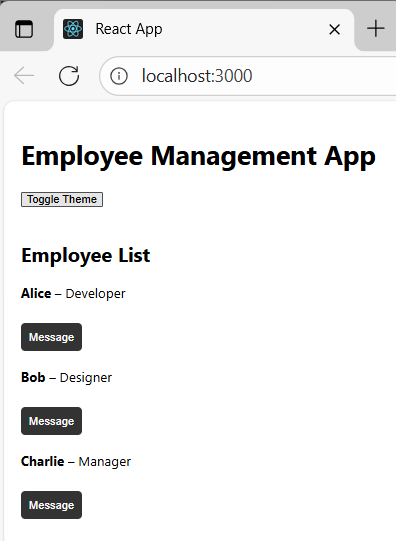
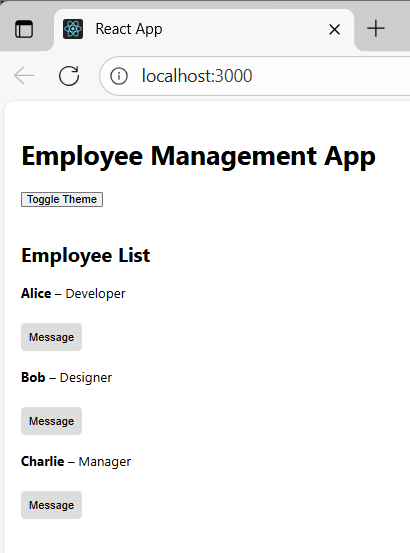
**app.js:**



**employeecard.js:**



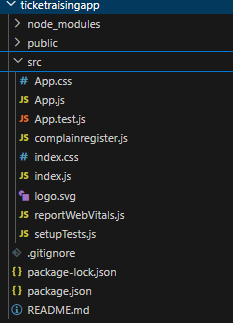
**Output:**



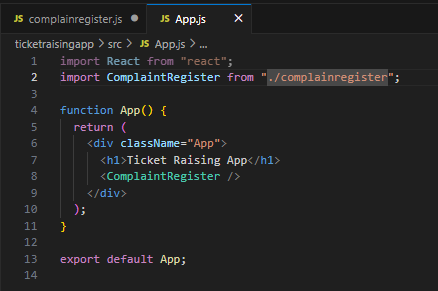
**Exercise: Create a React App named “ticketraisingapp” which will help to raise a complaint and get it resolved.**

**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.**

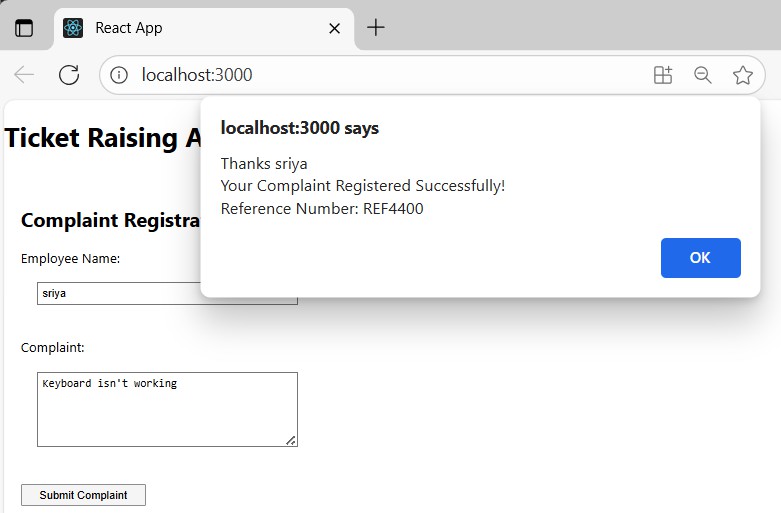
**App Structure:**



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



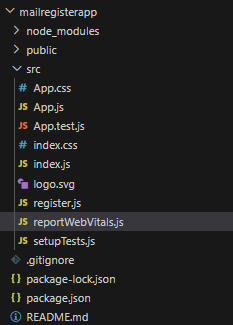
**complainregister.js:**

**Output:**

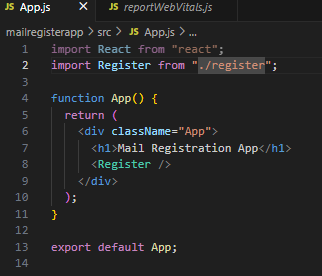
Exercise: Create a React App named “mailregisterapp” which will have a component named “register.js”. Create a form which accepts the name, email and password and validate the fields as per the following:

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

App Structure:

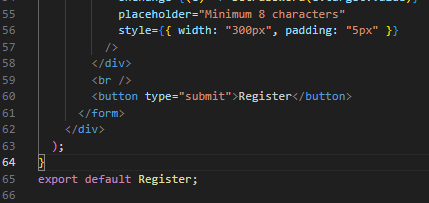


app.js:

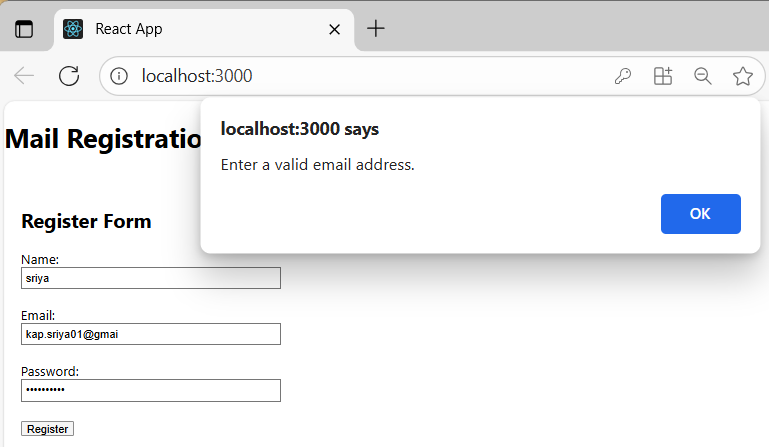
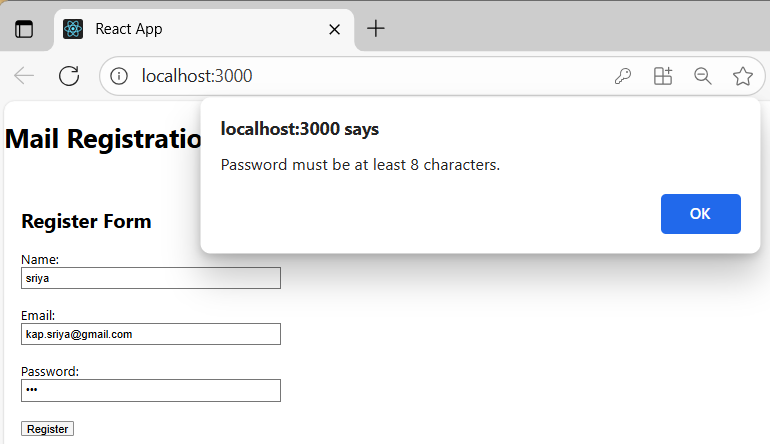
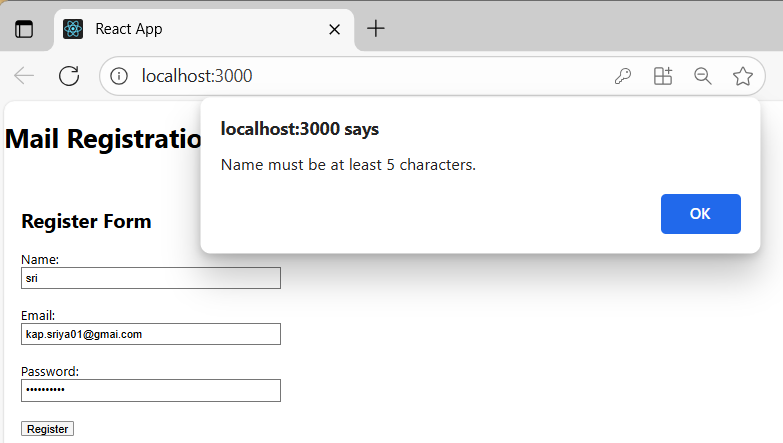
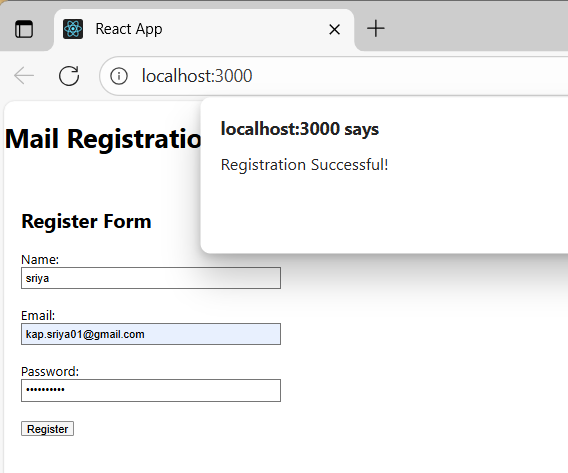


register.js:





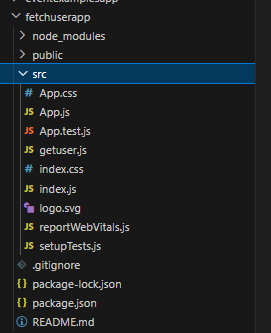
**Output:**



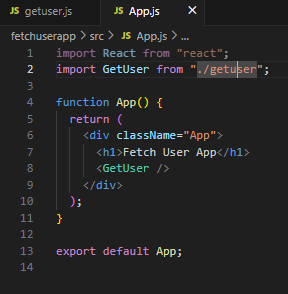
Exercise: Create a React Application “fetchuserapp” which will retrieve the user details from <https://api.randomuser.me/> and display the title, firstname and image of a user.

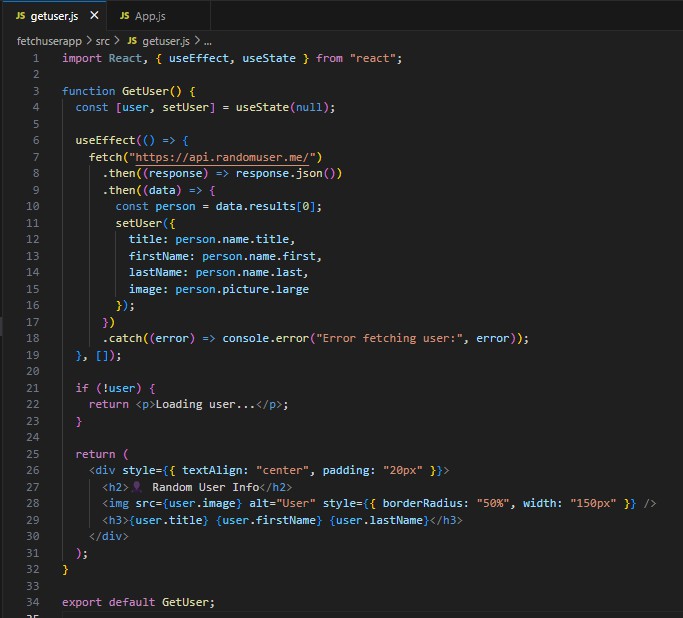
Create a component named “Getuser” and in the asynchronous method “ComponentDidMount ()” invoke the URL using fetch method and the response can be displayed in the render method of the component.

# App Structure:



**app.js:**



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

**Output:**

