**Lesson 03 Demo 03**

**Creating a React HOC Authorization Component**

**Objective:** To create the React HOC with authorization features to showcase its seamless integration into a secure and controlled user authentication system

**Tools required:** Node.js and React.js

**Prerequisites:** None

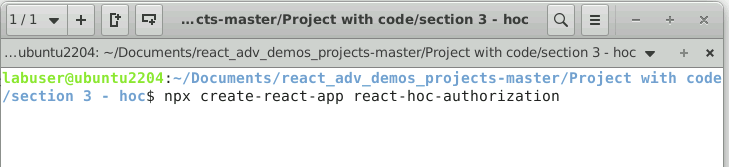
Steps to be followed:

1. Create and set up the React project
2. Create user-defined components and import them into the App.js file
3. Test the application

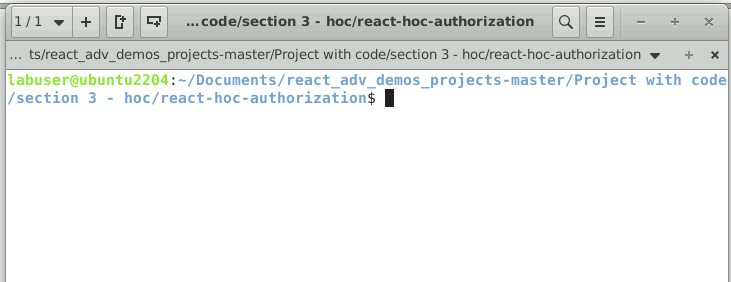
**Step 1: Create and set up the React project**

1. Open a terminal window to run the following command to create a React application:

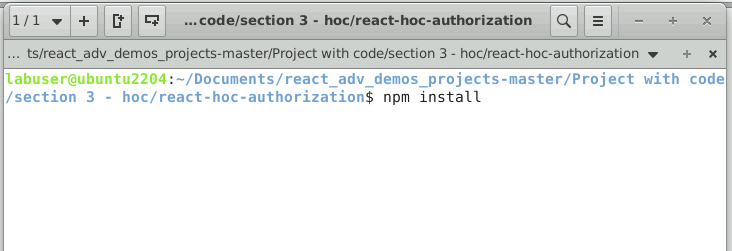
**npx create-react-app react-hoc-authorization**

****

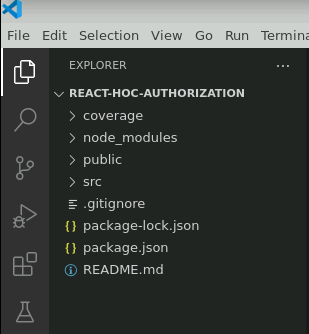
1. Open a terminal window inside a React project **react-hoc-authorization**

****

1. Run the **npm install** command to install all required dependencies mentioned in the **package.json** file on the local machine in the form of a **node\_module** folder



1. Open the **REACT-HOC-AUTHORIZATION** folder in the VS Code



**Step 2: Create user-defined components and import them into the App.js file**

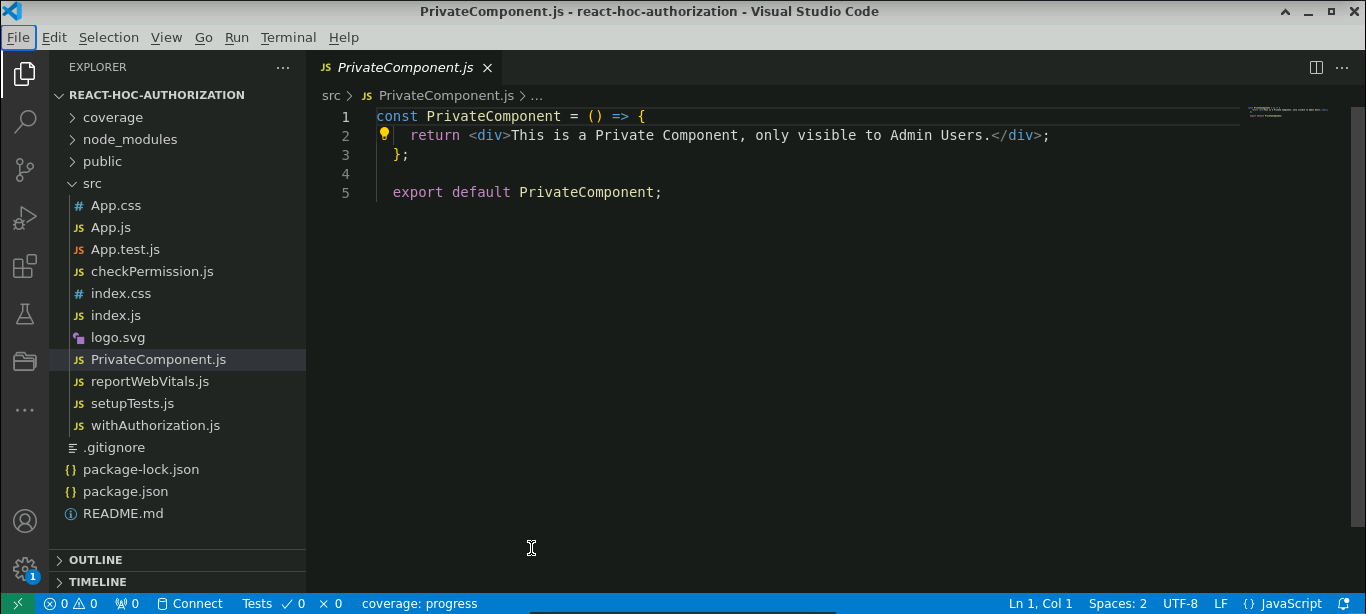
1. Create a **PrivateComponent.js** file that returns static data

**const PrivateComponent = () => {**

**return <div>This is a Private Component, only visible to Admin Users.</div>;**

**};**

**export default PrivateComponent;**

****

1. Create a **checkPermission.js** file that takes props as a parameter, checks the rule, and returns the j**sx** file

**const checkPermission = (props) => {**

**return props.userRole === "admin";**

**};**

**export default checkPermission;**

A screenshot of a computer

Description automatically generated

1. Create an HOC **withAuthorization.js** file which takes two parameters, the component and check function, and provides the new wrapper component, which is responsible for providing a new component or message

**// Higher Order Component**

**function withAuthorization(WrappedComponent, checkPermissions) {**

**return function (props) {**

**// You can also wrap it in useEffect for async permission checks.**

**return checkPermissions(props) ? (**

**<WrappedComponent {...props} />**

**) : (**

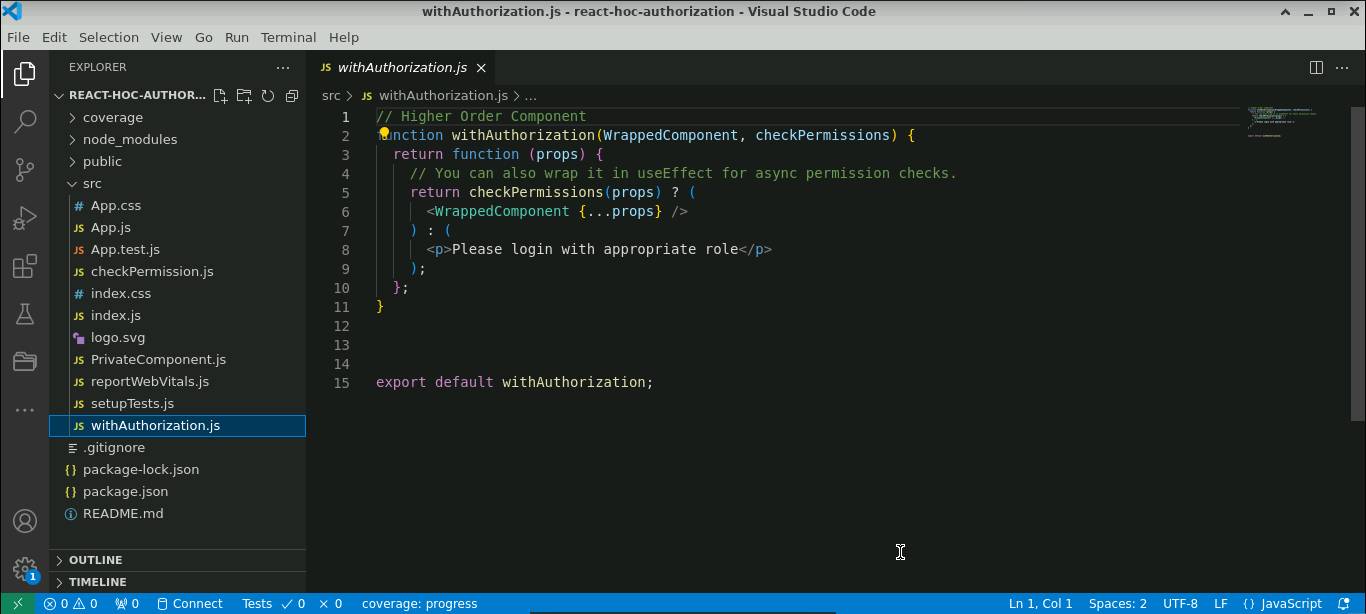
**<p>Please login with appropriate role</p>**

**);**

**};**

**}**

**export default withAuthorization;**



1. Import all the components in the **App.js** file and pass the **PrivateComponent** and **checkPermission** function as a parameter to an HOC

**import { useState } from "react";**

**import PrivateComponent from "./PrivateComponent";**

**import checkPermission from "./checkPermission";**

**import withAuthorization from "./withAuthorization";**

**const PrivateComponentComponentWithAuthorization = withAuthorization(**

**PrivateComponent,**

**checkPermission**

**);**

**export default function App() {**

**let [role,setRole]=useState("");**

**return (**

**<div className="App">**

**<input type="text" value={role} onChange={(e)=>setRole(e.target.value)}/>**

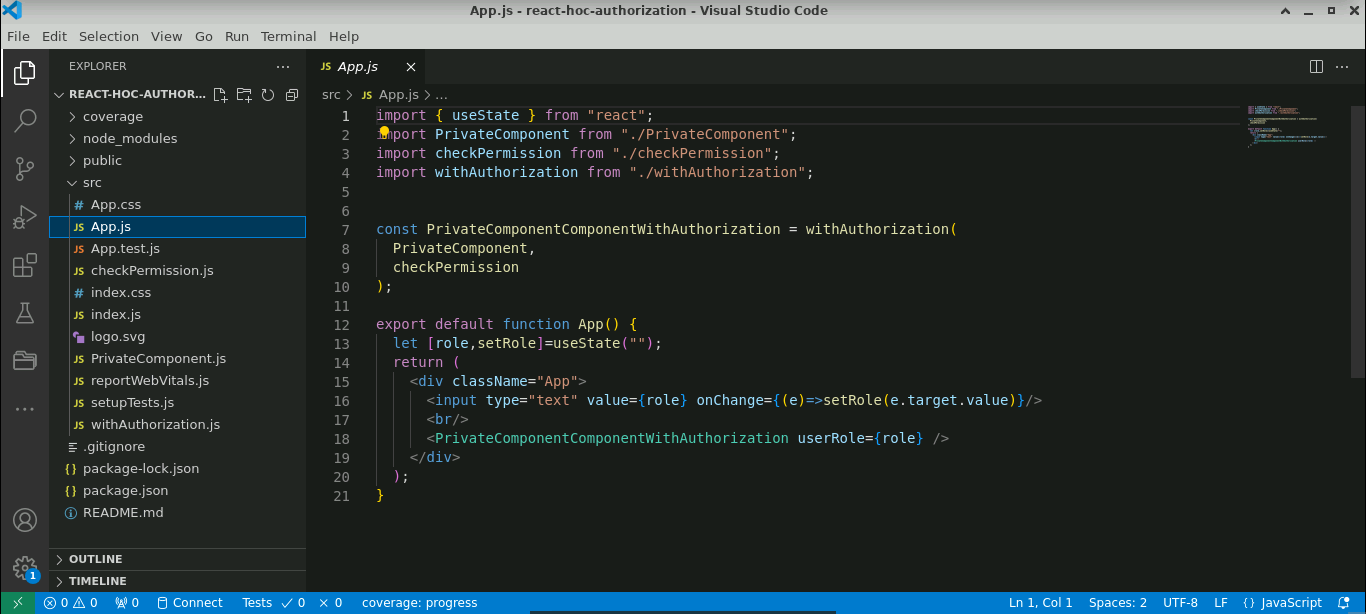
**<br/>**

**<PrivateComponentComponentWithAuthorization userRole={role} />**

**</div>**

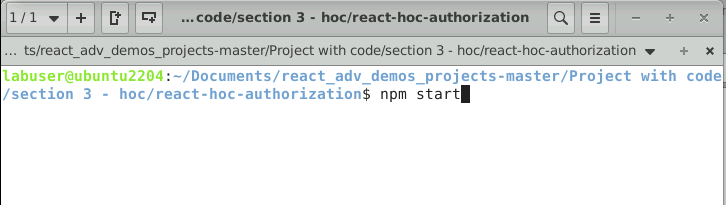
**);**

**}**

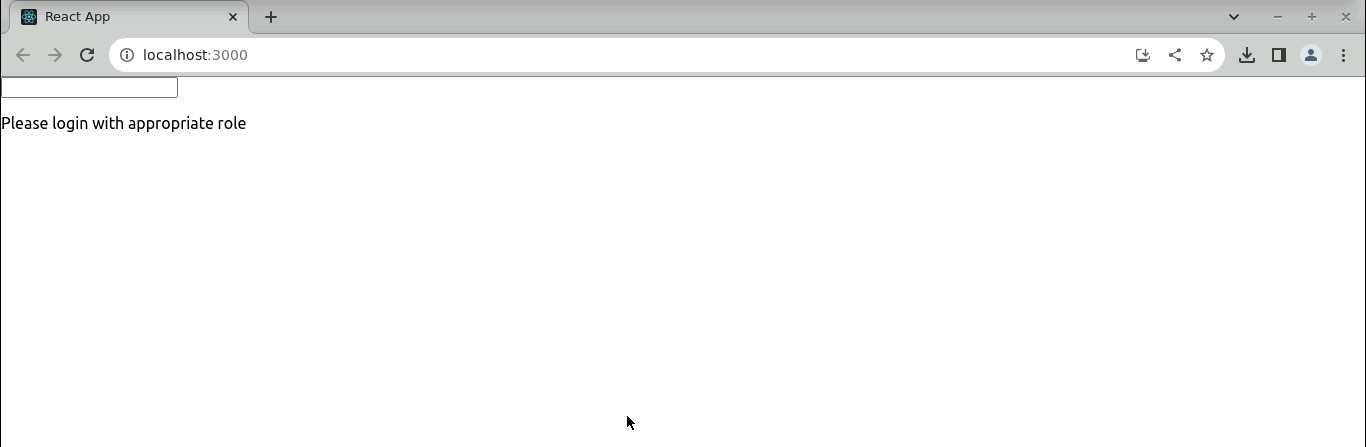
****

**Step 3: Test the application**

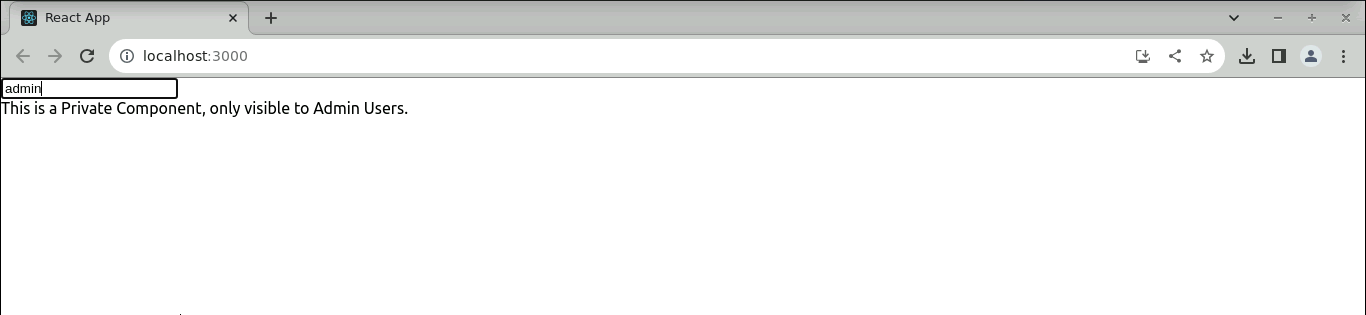
1. Test the application by running the **npm start** command in the terminal window:



The output can be seen as below:



**Note:** Private components are visible only when the role is **admin** as shown below:



With this, you have successfully created a React HOC with authorization capabilities.