**14.ReactJS – HOL**

**1.Explain the need and Benefits of React Context API**

* Avoids Prop Drilling: It allows data to be passed from parent to deeply nested child components without passing props at every level.
* Centralized State Management: Ideal for managing global states like theme, user authentication, language preferences, etc.
* Improves Code Clarity and Reusability: Cleaner component structure, easier to maintain and reuse logic across the app.

**2. Working with createContext()**

* Create a Context: Use const MyContext = React.createContext() to define the context object.
* Provide Context Value: Wrap the component tree with <MyContext.Provider value={data}> to pass data.
* Consume Context: Access the value in child components using useContext(MyContext) or <MyContext.Consumer>.

**3. List the types of Router Components**

* <BrowserRouter>: Uses HTML5 history API, best for modern web apps with clean URLs.
* <Routes> and <Route>: <Routes> replaces the old <Switch> and is used to define multiple routes using <Route> inside it.
* <Link> and <NavLink>: Used for navigation without reloading the page; <NavLink> adds styling for active links.

**React Application – employeesapp**

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



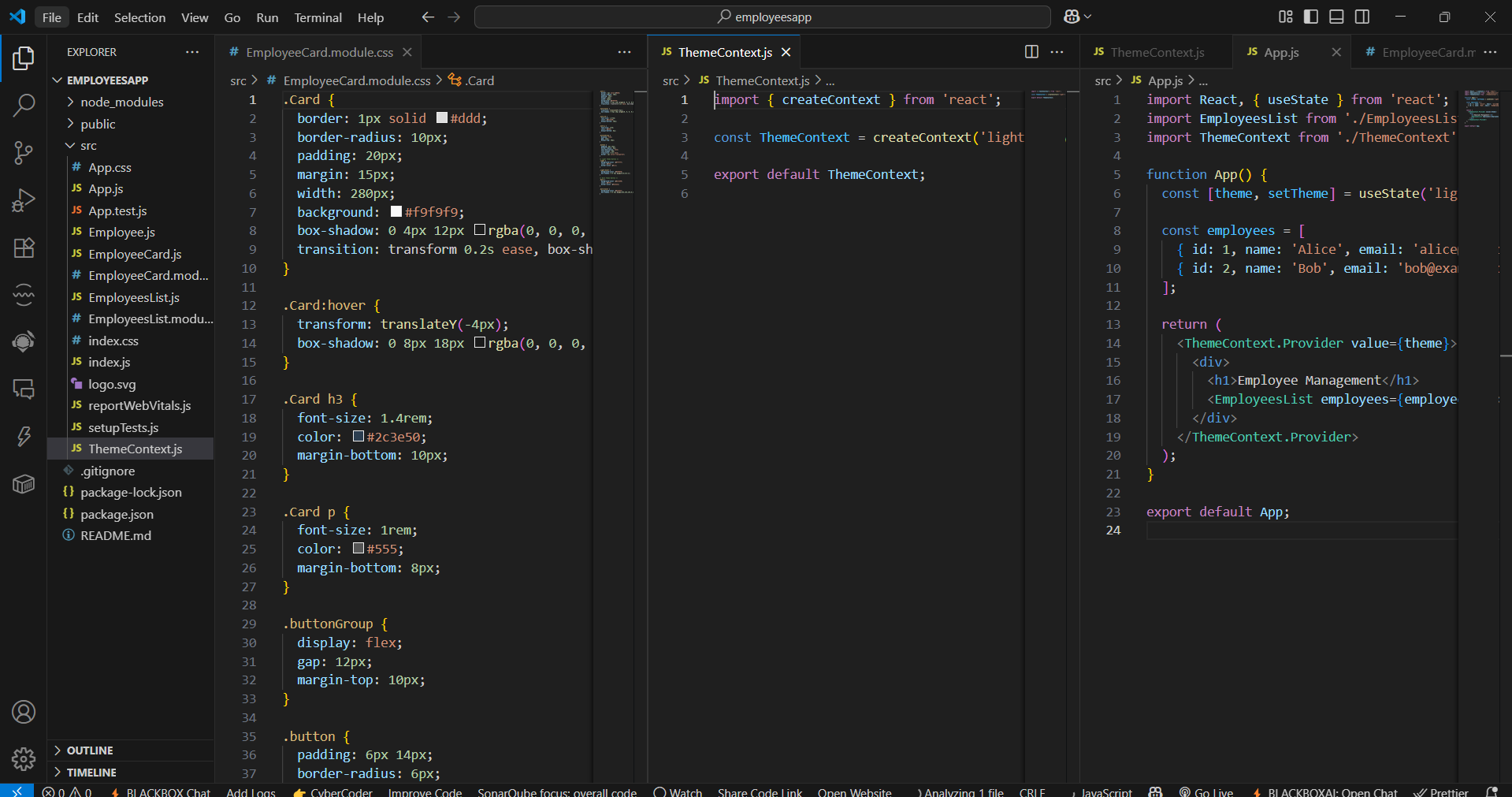
Figure 1: Restore node modules

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



Figure 2: Starting application

1. Explore the components present in **App.js**, **EmployeesList.js** and **EmployeeCard.js** files.
2. 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.
3. Open App component present in **App.js** file.
   1. Import the ThemeContext in App component.
   2. Define the theme context provider to be the entire JSX of the App component.
   3. Assign the value for the theme provider from the state of the component.
   4. Modify the call to EmployeeList component so that theme name is no longer passed as props.
4. Go to EmployeeList component present in **EmployeeList.js** file and modify it so that theme name is not passed explicitly to its child component.
5. Go to **EmployeeCard** component inside **EmployeeCard.js** file
   1. Import the ThemeContext into the component file
   2. Retrieve the value of the context with the help of **useContext()** and store it in a variable
   3. Use the variable to pass the className for the buttons.



**Output:**

