**2.ReactJs HOL**

**1. Explain React Components**

React components are the building blocks of a React application.  
They let you split the UI into reusable, isolated pieces.

* Each component defines part of the UI.
* Components accept props (inputs) and return JSX (UI elements).
* Think of them like functions that return HTML.

**Example:**

function Welcome(props) {

return <h1>Hello, {props.name}</h1>;

}

2. **Differences Between Components and JavaScript Functions**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **JavaScript Function** | **React Component** |
| Purpose | Perform logic or calculations | Return UI elements (JSX) |
| Return Value | Any data type (number, string, etc.) | JSX (HTML-like structure) |
| Naming | Any name | Must start with a Capital letter |
| Use in UI | Not rendered directly | Rendered by React into the DOM |
| Lifecycle Methods | Not available | Available (in Class components) |
| Props handling | Optional parameters | Props are passed in to render content |

**3. Types of Components in React**

There are two main types:

1. Function Components – simpler, use hooks
2. Class Components – older, use lifecycle methods

**4.** **Explain Class Component**

* A class component is a JavaScript class that extends React.Component.
* It has a render() method and can hold its own state and lifecycle methods.

**Example:**

import React from 'react';

class Welcome extends React.Component {

render() {

return <h1>Hello, {this.props.name}</h1>;

}

}

**Key Takeways:**

* Can use state
* Can use lifecycle methods like componentDidMount
* Uses this keyword

**5.** **Explain Function Component**

* A function component is a plain JavaScript function that returns JSX.
* It can use Hooks (like useState, useEffect) to manage state and side effects.

function Welcome(props) {

return <h1>Hello, {props.name}</h1>;

}

**With state using hooks:**

import React, { useState } from 'react';

function Counter() {

const [count, setCount] = useState(0);

return <button onClick={() => setCount(count + 1)}>Count: {count}</button>;

}

**Key Takeways:**

* Simpler syntax
* No this keyword
* Can use Hooks for state/lifecycle logic

**6.** **Define Component Constructor**

In class components, the constructor() is used to:

* Initialize state
* Bind event handlers
* Call super(props) to access this.props

**Example:**

class MyComponent extends React.Component {

constructor(props) {

super(props);

this.state = { count: 0 };

}

}

**7.** **Define render() Function**

* The render() method is required in class components.
* It returns JSX to define what the UI looks like.

**Example:**

class MyComponent extends React.Component {

render() {

return <h1>Hello, World!</h1>;

}

}

**React App : studentapp**

**Home.js**

import React from 'react';

function Home() {

  return (

    <div>

      <h2>Welcome to the Home page of Student Management Portal</h2>

    </div>

  );

}

export default Home;

**About.js**

import React from 'react';

function About() {

  return (

    <div>

      <h2>Welcome to the About page of the Student Management Portal</h2>

    </div>

  );

}

export default About;

**Contact.js**

import React from 'react';

function Contact() {

  return (

    <div>

      <h2>Welcome to the Contact page of the Student Management Portal</h2>

    </div>

  );

}

export default Contact;

**App.js**

import React from 'react';

import Home from './Components/Home';

import About from './Components/About';

import Contact from './Components/Contact';

function App() {

  return (

    <div>

      <h1>Student Management Portal</h1>

      <Home />

      <About />

      <Contact />

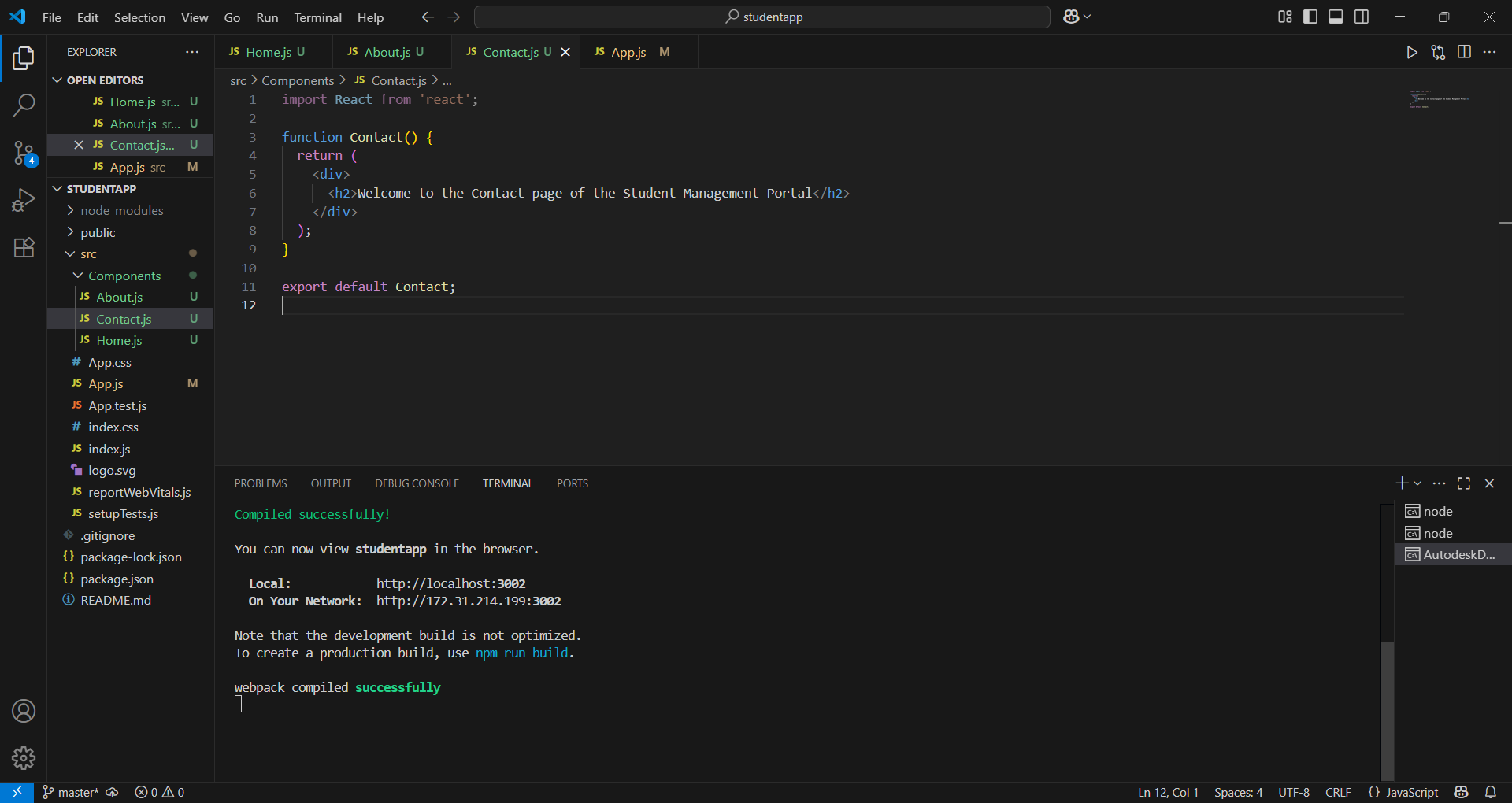
    </div>

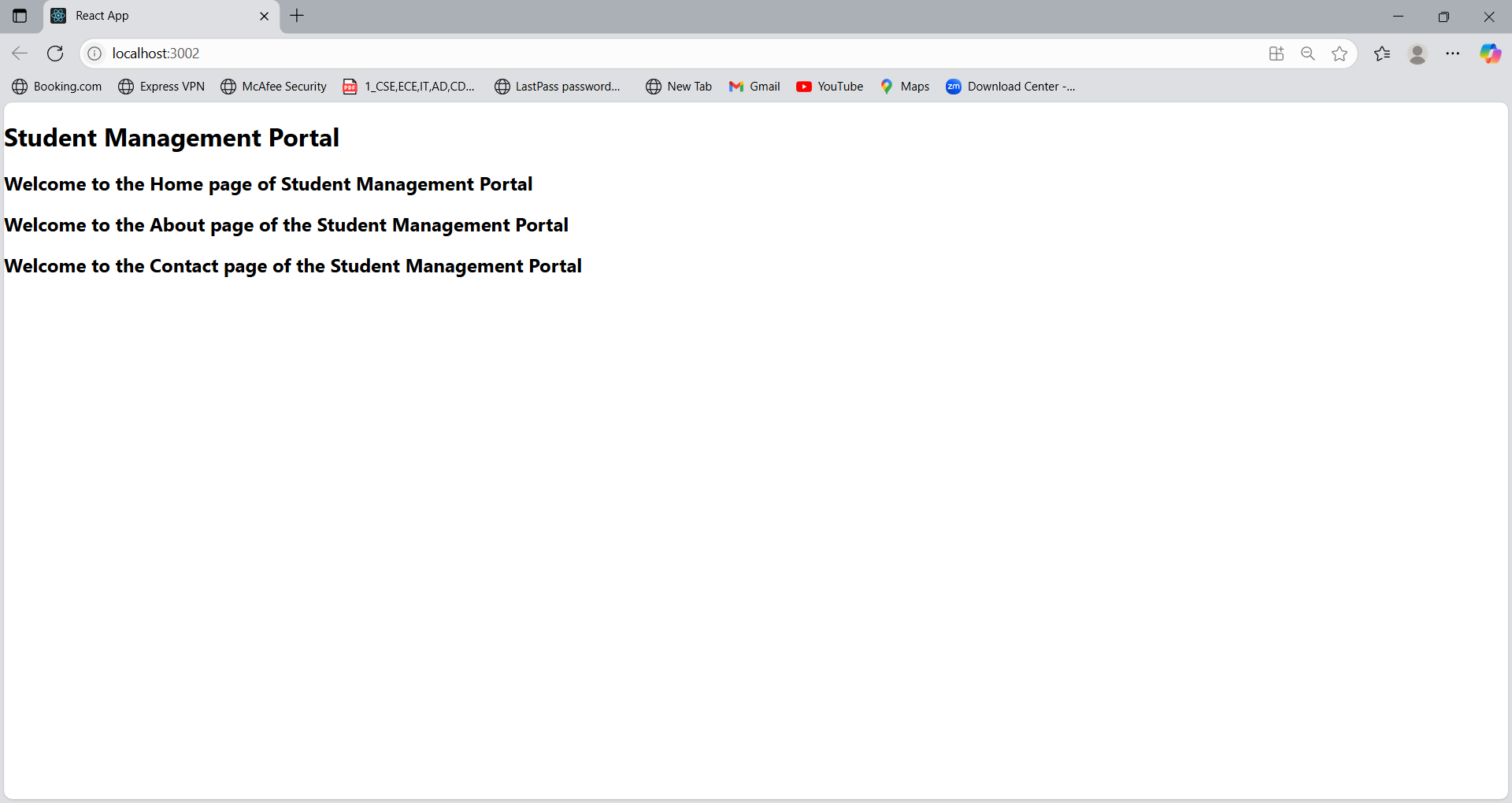
  );

}

export default App;

**Final Output:**

****

****