**React Handson - 2**

**1. React Components: An Introduction**

In React, the **component** is the foundational building block of the user interface. A component is essentially a **self-contained, reusable piece of code** that defines how a part of the UI should appear and behave. Each component manages its own content, structure, and behavior, which allows for efficient and scalable development.

Components in React help developers break the UI into small, isolated sections that can be developed, tested, and maintained independently. These components can be combined to create complex UIs while maintaining code modularity and readability.

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

While React function components and JavaScript functions may appear similar syntactically, they serve different purposes and have key differences in behaviour.

| **Feature** | **React Components** | **JavaScript Functions** |
| --- | --- | --- |
| Purpose | Used to create UI elements | Used to perform calculations or return values |
| Return Value | Returns JSX (React UI markup) | Returns plain data values |
| Lifecycle Methods | Supported in class components | Not applicable |
| Can maintain state | Yes (with hooks or this. state) | No |
| JSX support | Yes, inside the return statement | No, unless wrapped in a React component |

In summary, a JavaScript function performs a task, while a React component defines how a piece of the UI should be rendered.

**3. Types of React Components**

React offers two main types of components:

**a. Class Components**

These are defined using ES6 class syntax and must extend React.Component. They include a render() method and can have their own **state** and **lifecycle methods** such as componentDidMount() and componentDidUpdate().

Example:

javascript

CopyEdit

import React, { Component } from 'react';

class ExampleComponent extends Component {

render() {

return <h1>This is a class component</h1>;

}

}

**b. Function Components**

These are simpler, defined using standard JavaScript functions. In modern React, they can also use **hooks** like useState and useEffect to manage state and side effects.

Example:

javascript

CopyEdit

import React from 'react';

function ExampleComponent() {

return <h1>This is a function component</h1>;

}

Function components are preferred in modern React due to their simplicity and support for hooks.

**4. Class Component**

A **class component** in React is a component that is created using the ES6 class syntax. It is ideal when you need to manage component-level state or lifecycle methods.

Basic structure of a class component:

javascript

CopyEdit

class ComponentName extends React.Component {

render() {

return (

<div>

// JSX goes here

</div>);

}

}

Class components can contain:

* constructor() to initialize state
* Lifecycle methods (e.g., componentDidMount, componentDidUpdate)
* this.props and this.state for data access and management

**5. Function Component**

A **function component** is defined using a JavaScript function that returns JSX. These components are used for displaying content and managing logic in a concise and readable manner.

In the past, function components were considered "stateless," but since the introduction of **React Hooks**, they are now fully capable of handling local state and side effects.

Example:

javascript

CopyEdit

function Welcome() {

return <h1>Hello from Function Component</h1>;

}

With hooks:

javascript

CopyEdit

import React, { useState } from 'react';

function Counter() {

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

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

}

**6. Component Constructor**

The **constructor** is a special method used only in **class components**. It is primarily used to:

* Initialize the component’s state
* Bind event handler functions

The constructor is called automatically when a component is created and mounted.

Example:

javascript

CopyEdit

constructor(props) {

super(props);

this.state = { message: "Hello" };

}

It’s important to call super(props) before accessing this in the constructor, or else it will result in an error.

**7. render() Function**

The render() function is a **mandatory method** in class components. It tells React what to display on the screen. It must return a valid JSX structure.

React calls the render() method each time the component’s state or props change, allowing the UI to be updated accordingly.

Example:

javascript

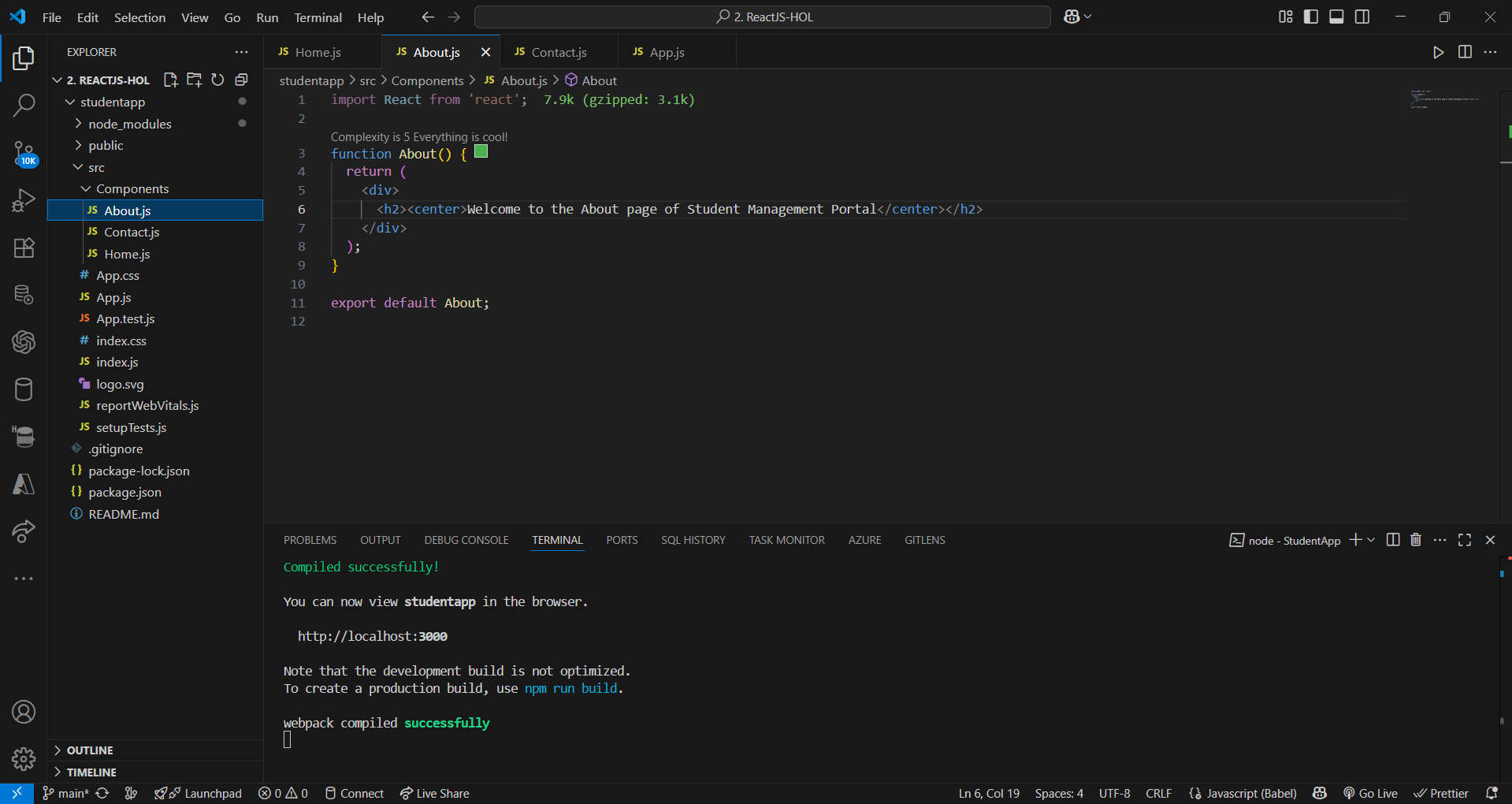
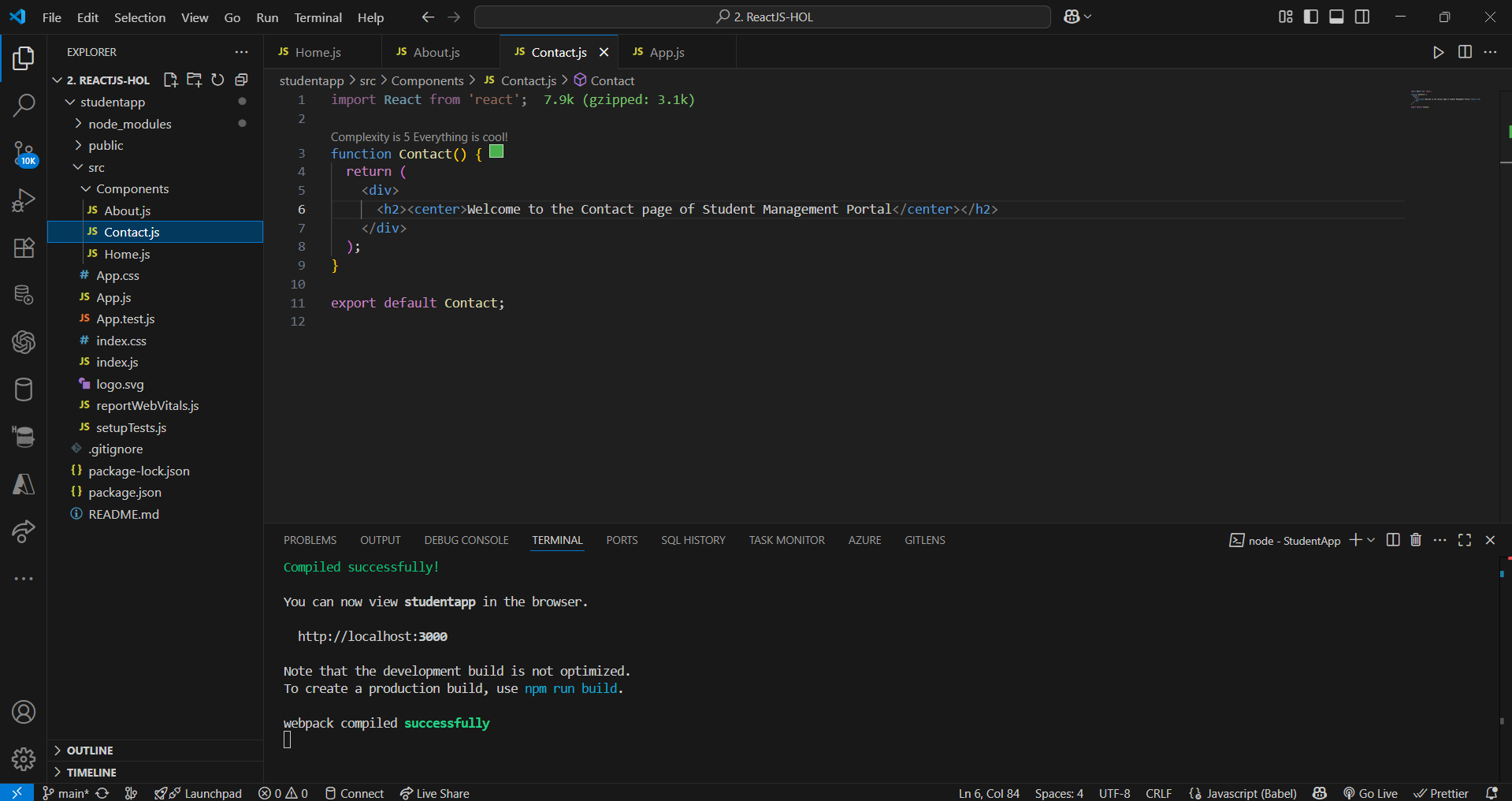
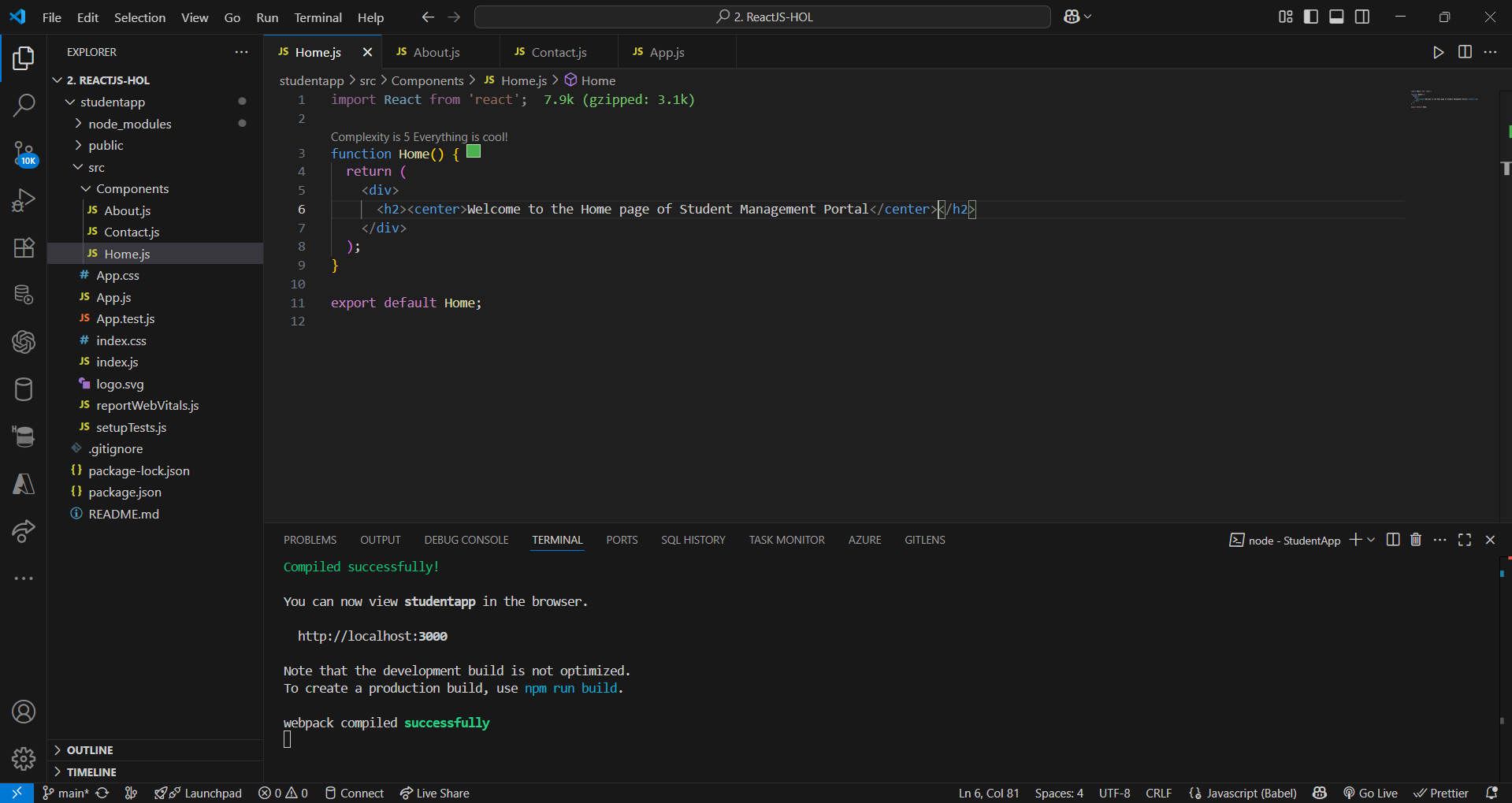
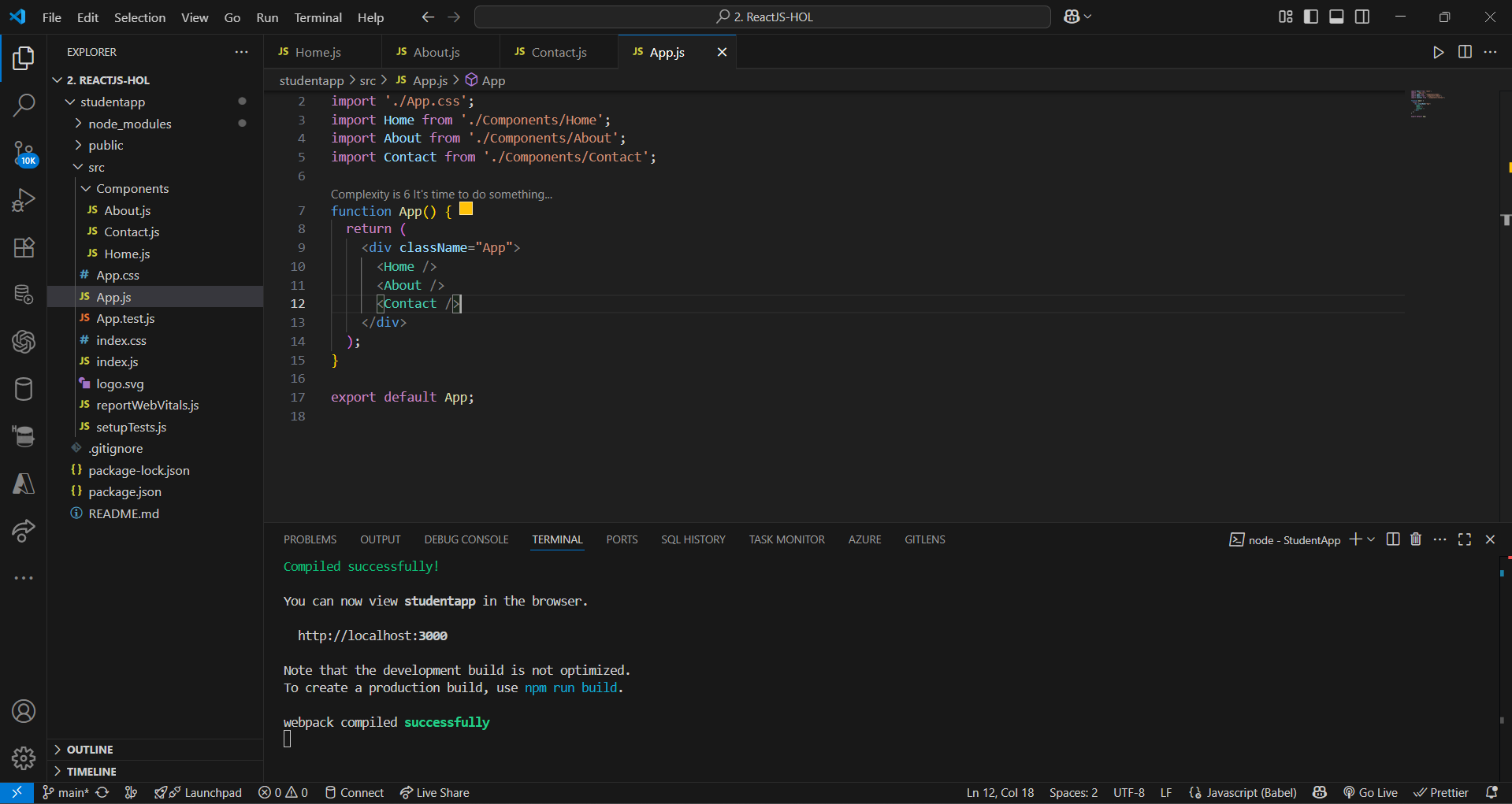
CopyEdit

render() {

return <h2>This is rendered on the screen</h2>;

}

**8.Codes**

****

**9.Output**

**A screenshot of a computer

AI-generated content may be incorrect.**