**ReactJS-HOL**

**1.**

**Define SPA and its benefits**

Solution:- A **Single Page Application (SPA)** is a web application that interacts with the user by dynamically rewriting the current page, rather than loading entire new pages from the server.

**Benefits:**

Faster navigation and responsiveness.

Reduced server load.

Seamless user experience.

**Define React and identify its working**

Solution:- **React** is an open-source JavaScript library developed by Facebook for building user interfaces, especially for SPAs.

### Features of React

* Component-based architecture
* One-way data binding
* Reusable components
* Declarative UI
* Virtual DOM for performance

**Identify the differences between SPA and MPA**

Solution:-

| **Feature** | **SPA** | **MPA** |
| --- | --- | --- |
| Page Reloading | No | Yes |
| Speed | Faster | Slower |
| SEO Optimization | Harder | Easier |
| Development Effort | More complex routing | Simpler |

**Explain Pros & Cons of Single-Page Application**

Solution:- **Pros:**

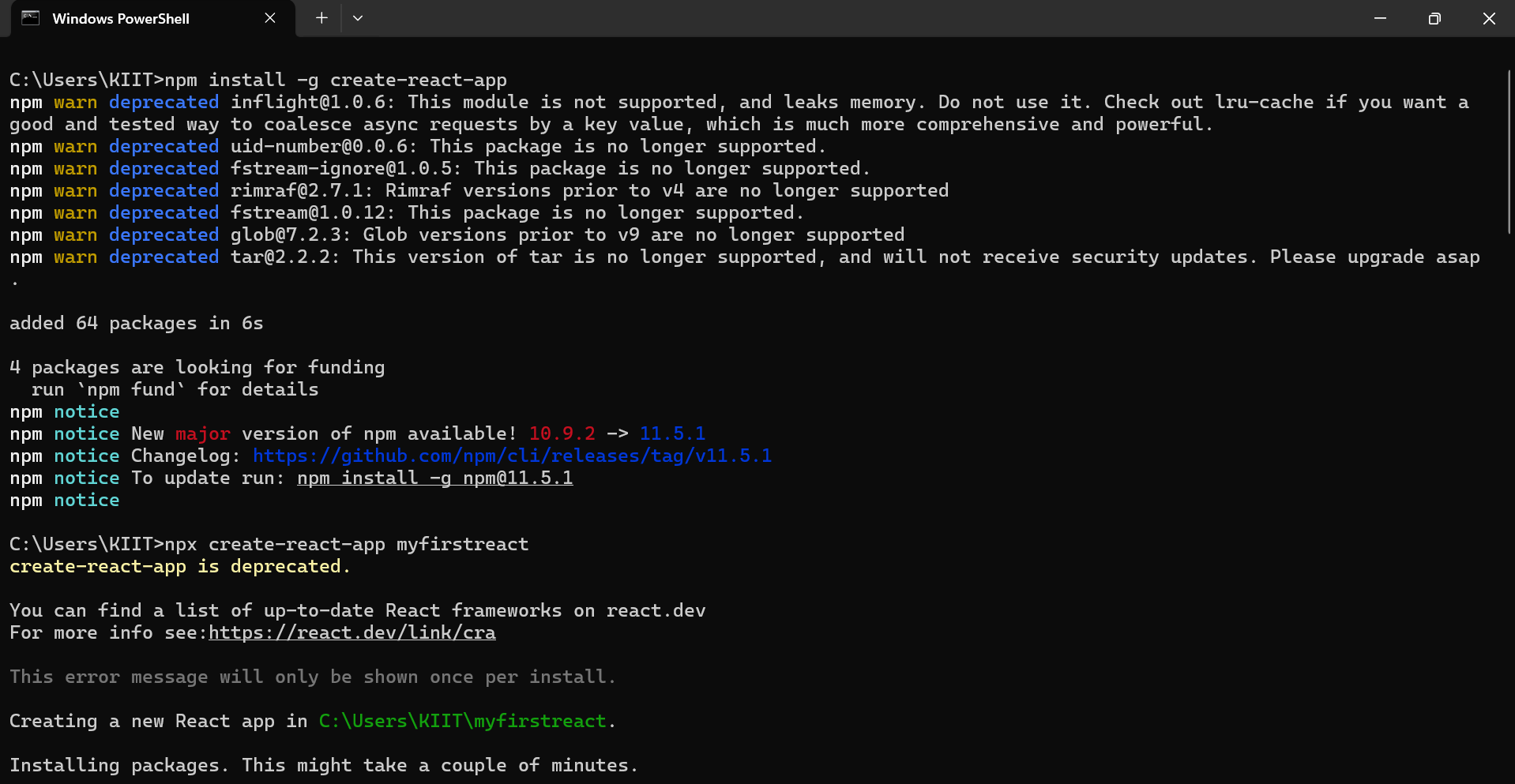
* Smooth user experience
* Faster transitions
* Better caching

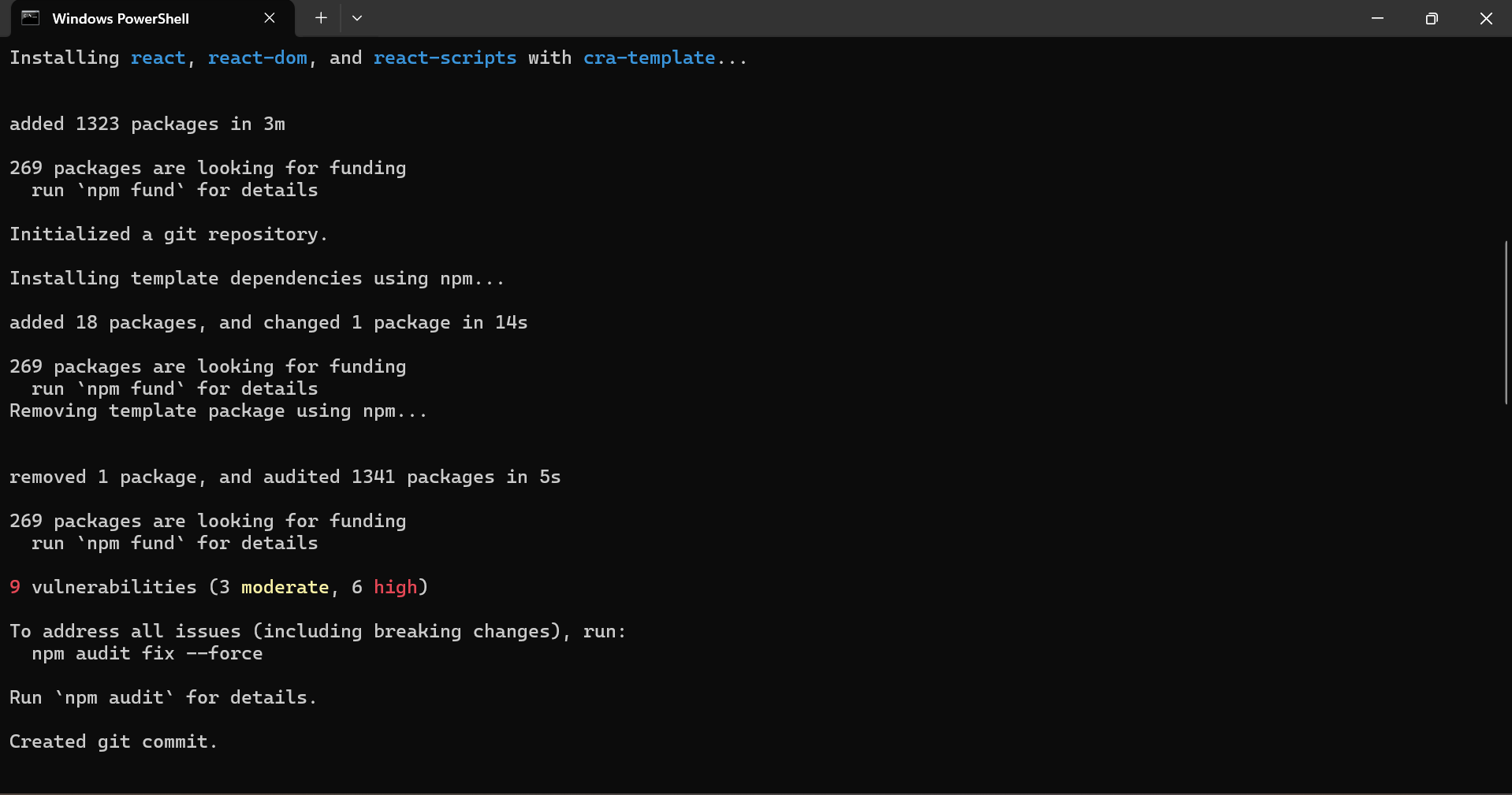
**Cons:**

* Poor SEO support
* Higher initial loading time
* Complex client-side routing

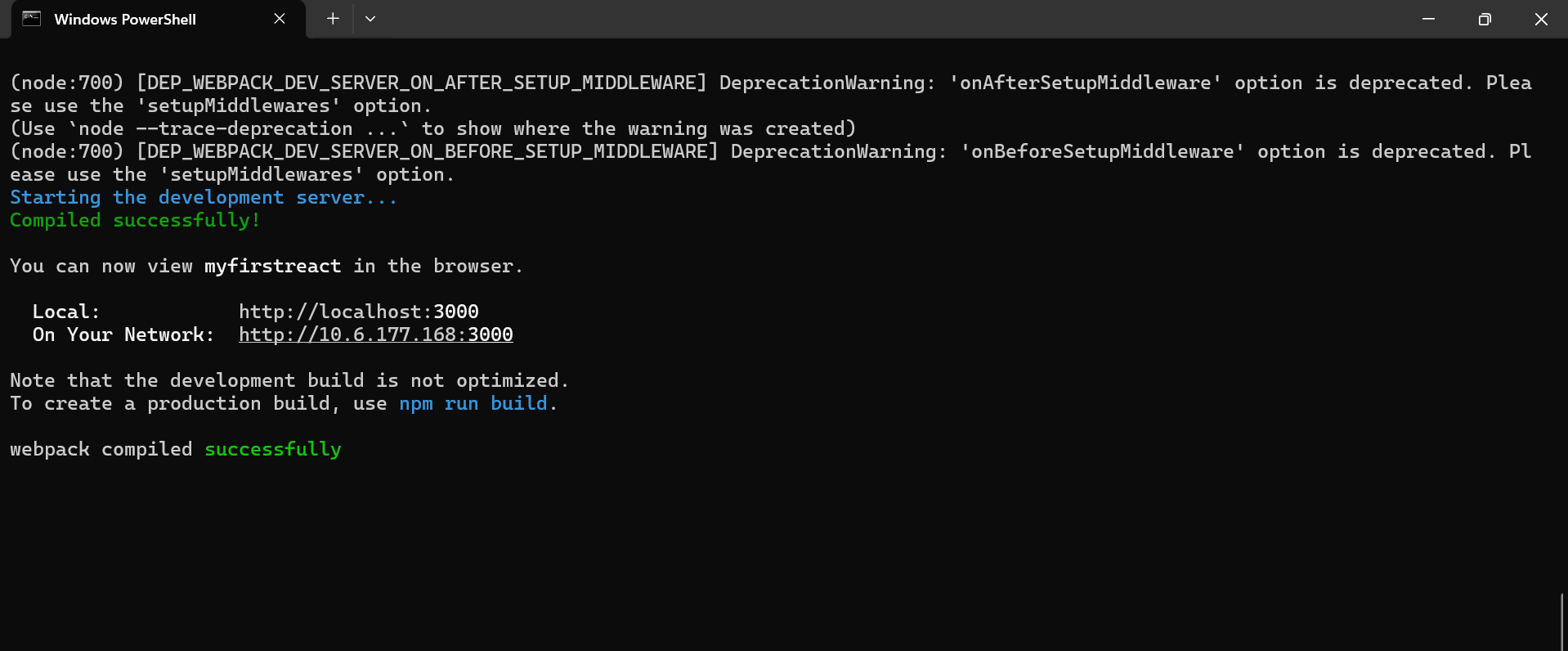
**Define virtual DOM**

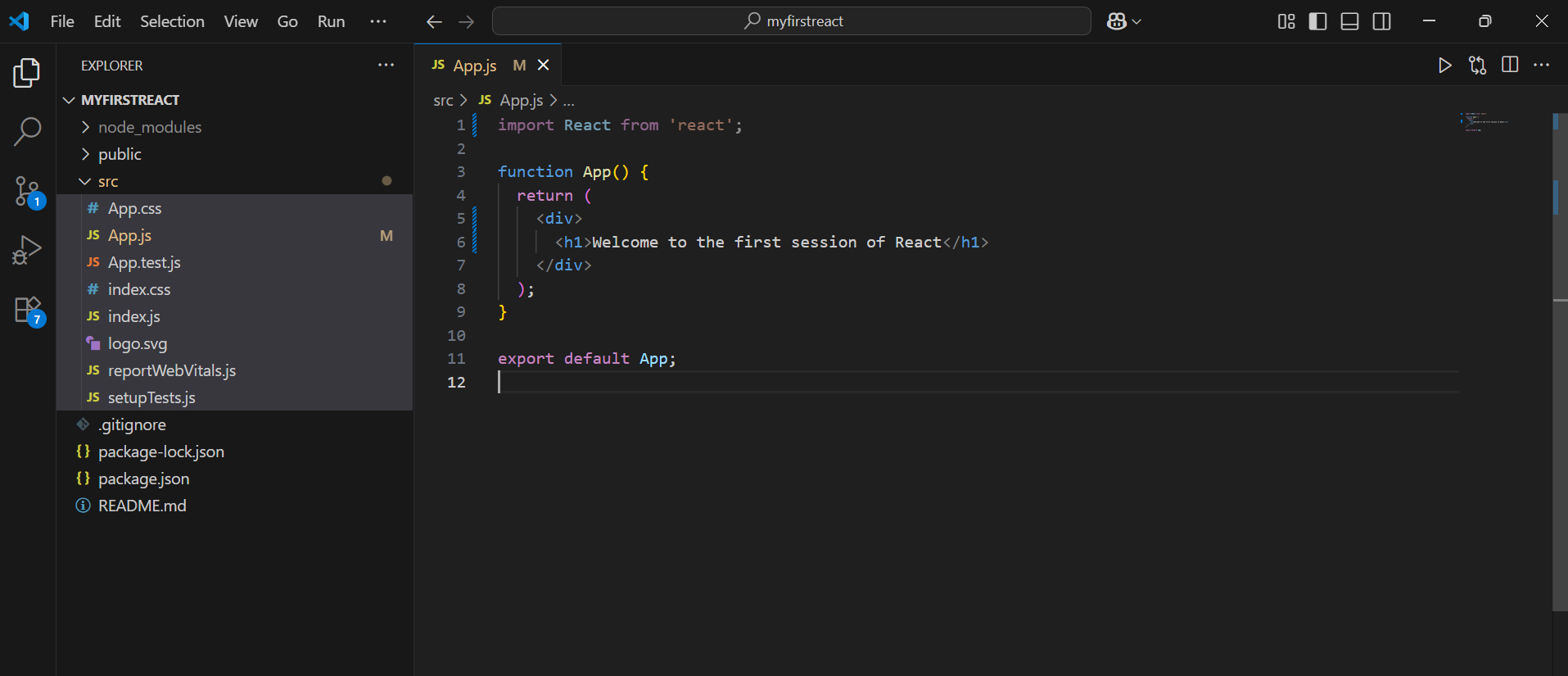
Solution:- The **Virtual DOM** is a lightweight JavaScript representation of the actual DOM. React uses it to improve performance by updating only the parts of the DOM that have changed.

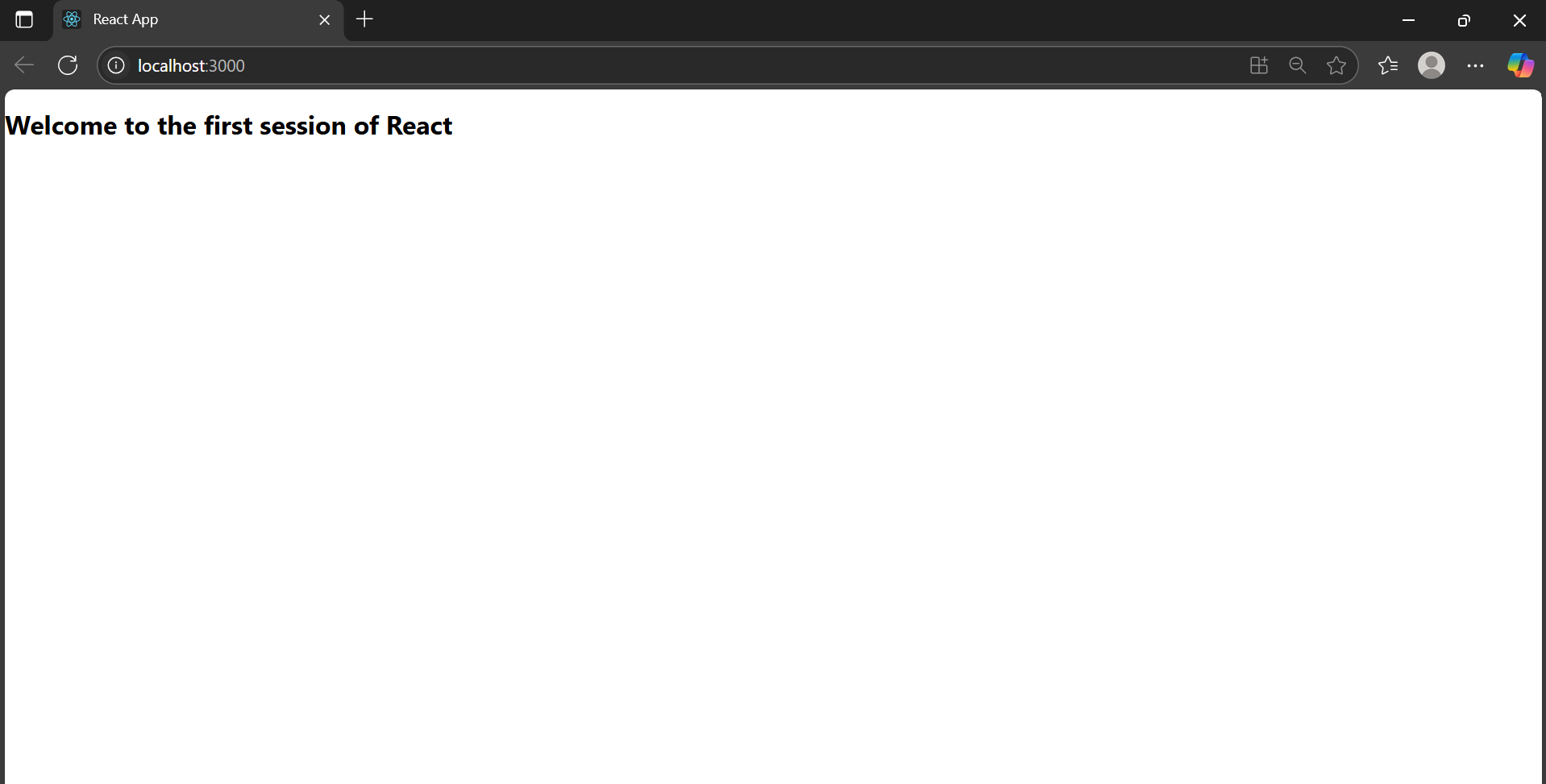












**2.**

* **Explain React components**

Solution:- **React components** are the building blocks of any React application.  
They are reusable pieces of code that return React elements (usually JSX) which describe how a section of the UI should look.

**Identify the differences between components and JavaScript functions**

Solution:-

| **Feature** | **JavaScript Function** | **React Component** |
| --- | --- | --- |
| Purpose | General-purpose logic | Used to build UI |
| Output | Any data type | Must return JSX (or React elements) |
| Naming Convention | Any valid name | Must start with **a capital letter** |
| Reusability in UI | Not reusable directly in UI | Reusable like HTML tags (<Component />) |
| Lifecycle Methods (Class) | Not available | Available in class components |

* **Identify the types of components**

Solution:- React has two main types of components:

* **Class Components**
* **Function Components**

**Explain class component**

Solution:- A **class component** is a React component created using ES6 classes.  
It can have its own **state**, **lifecycle methods**, and uses a render() method to return JSX.

**Explain function component**

Solution:- A **function component** is a simpler way to write components using JavaScript functions.  
They can use **React Hooks** (like useState, useEffect) to manage state and lifecycle behavior.

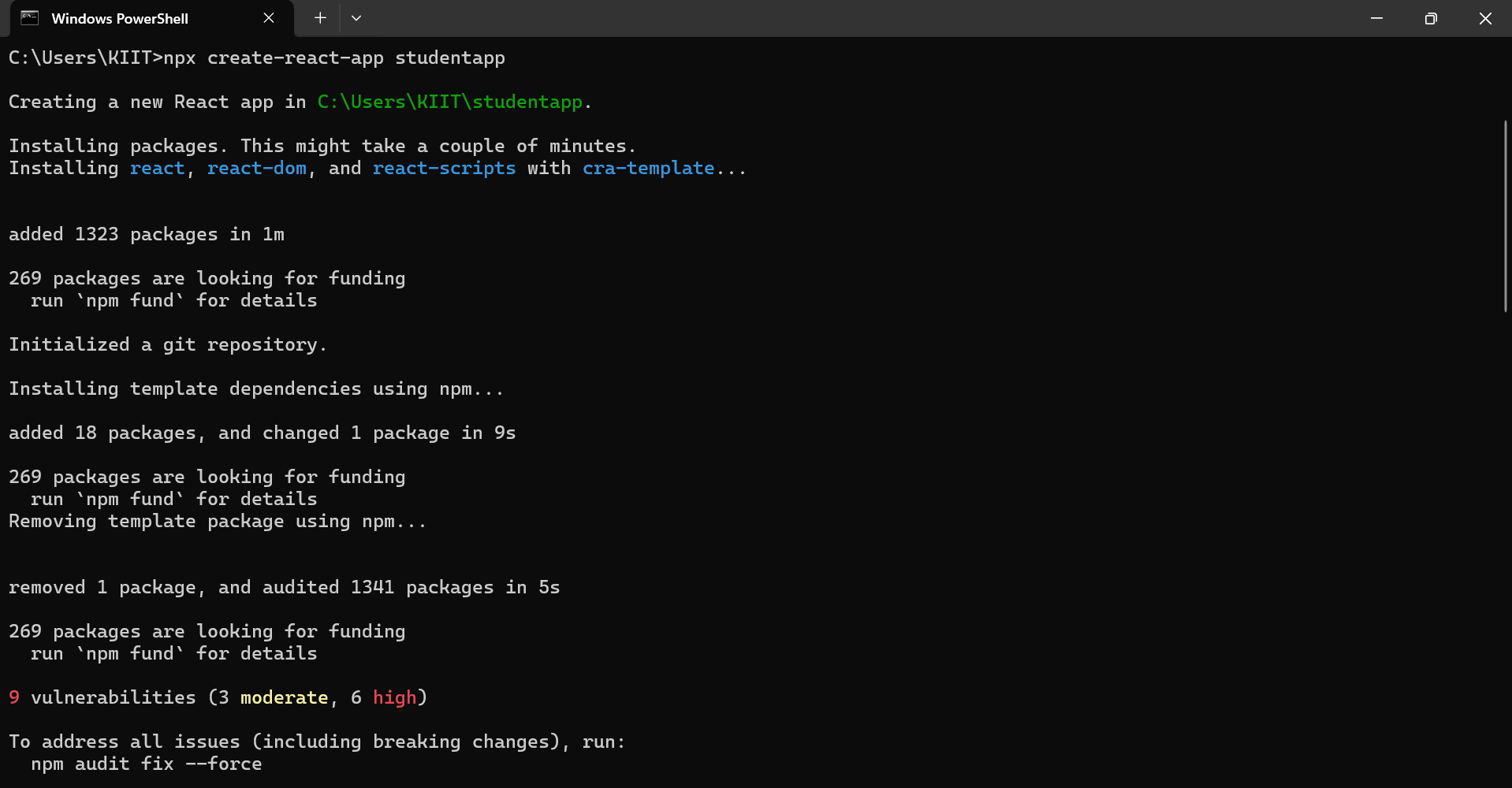
**Define component constructor**

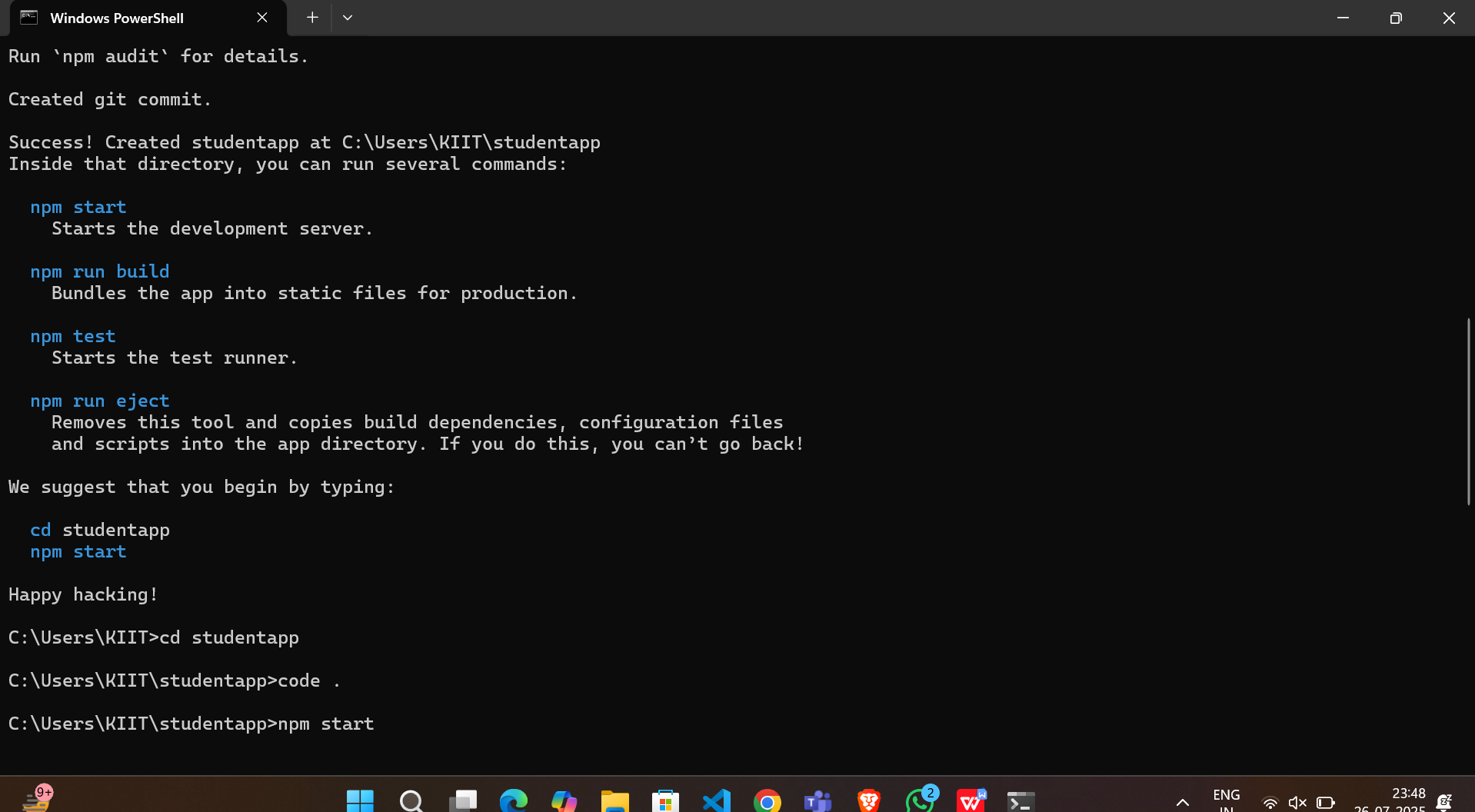
Solution:- The **constructor** method is a special function used in class components to initialize **state** and bind methods.

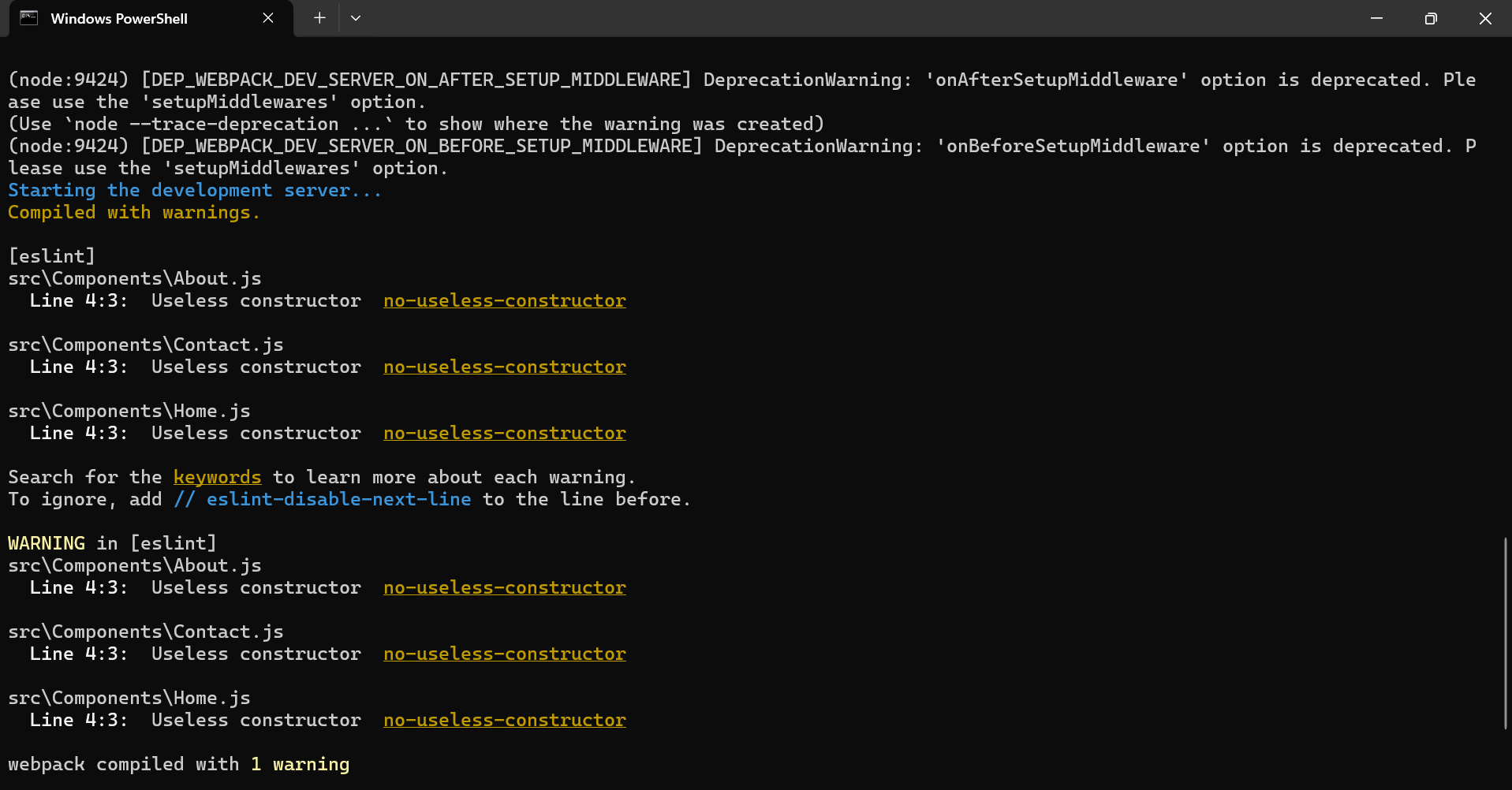
**Used only in class components**, it is called **once when the component is created**.

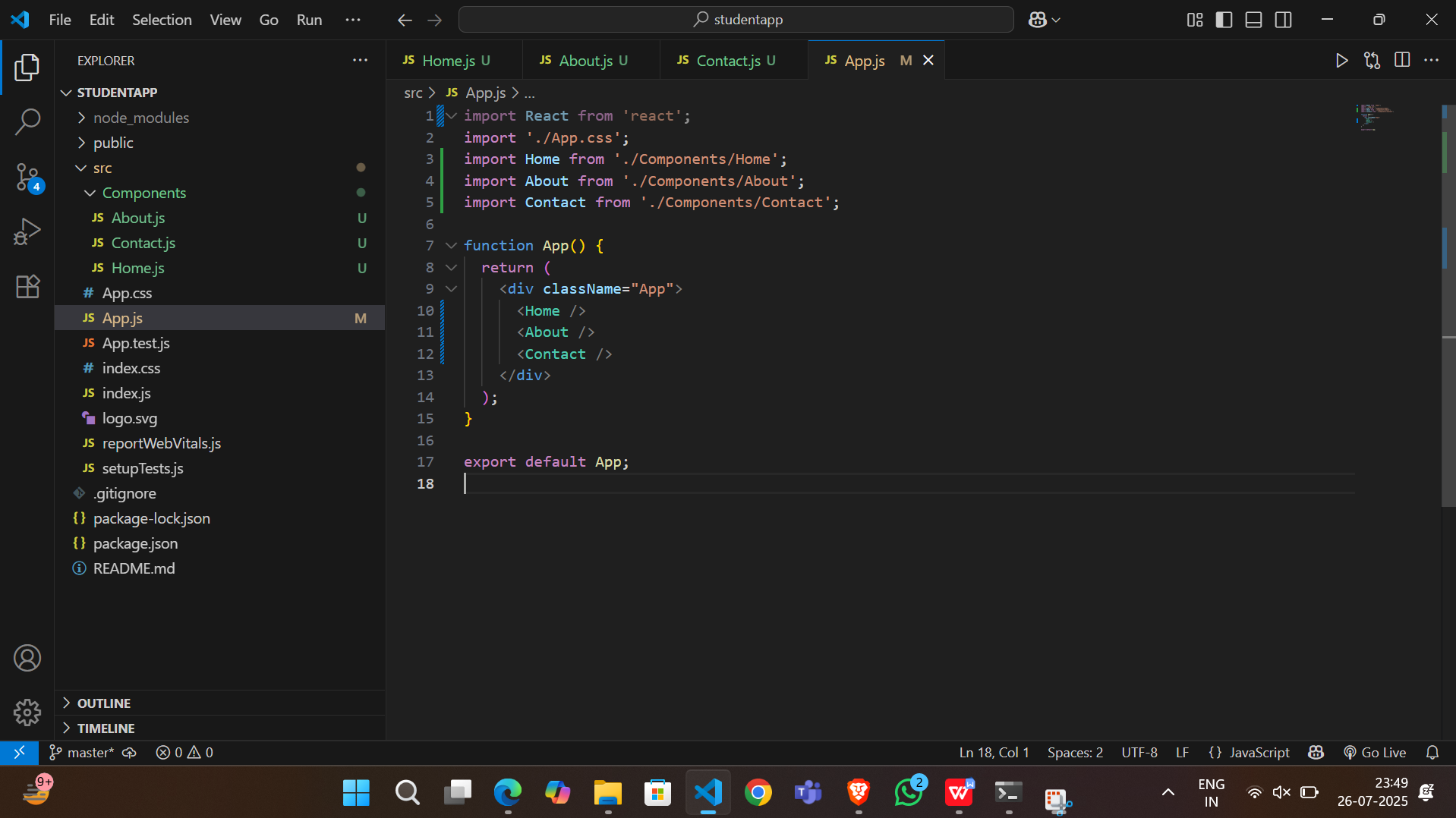
**Define render() function**

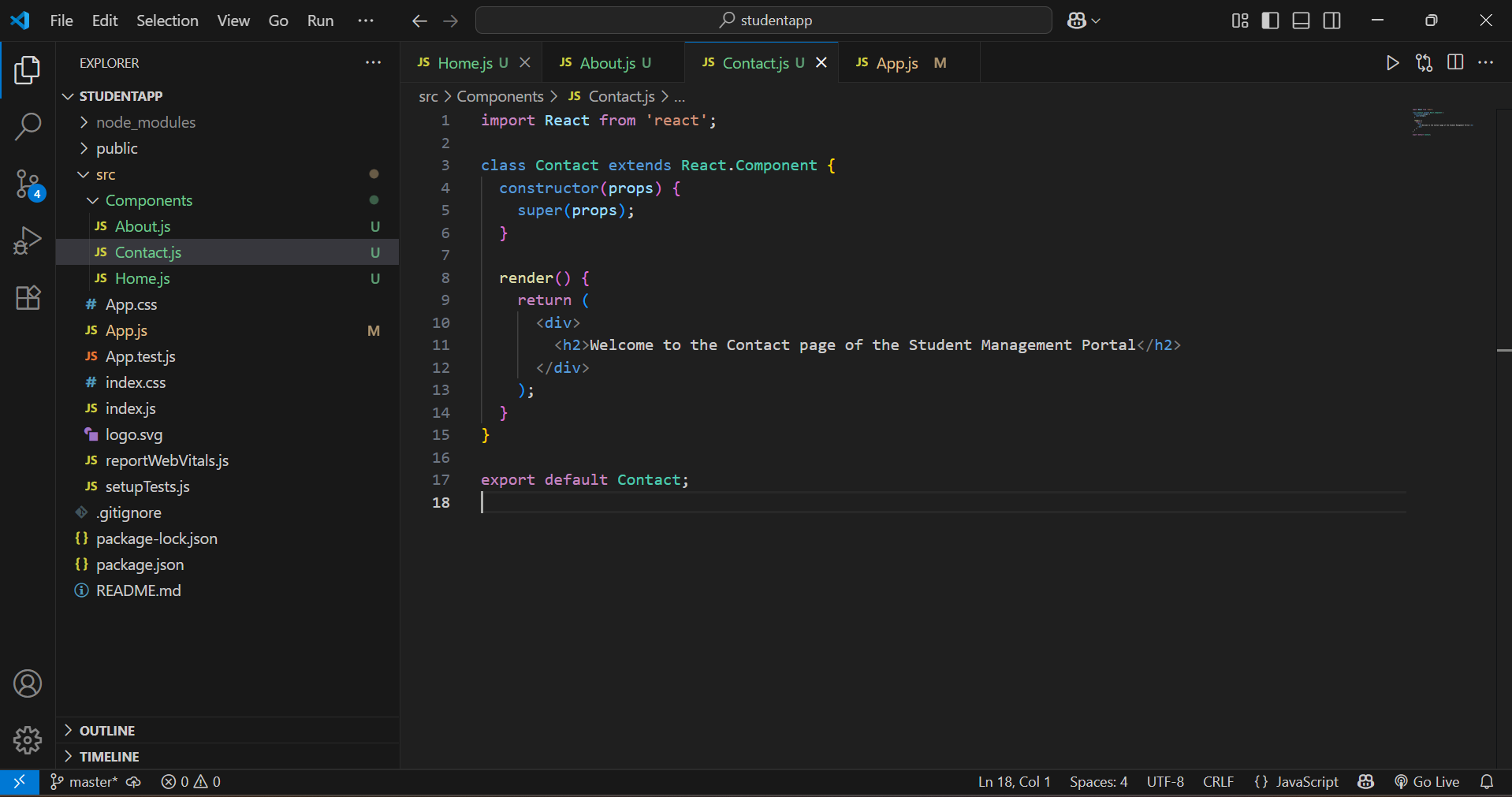
Solution:- The render() function is **required in class components**.  
It returns the JSX that should be rendered to the DOM.

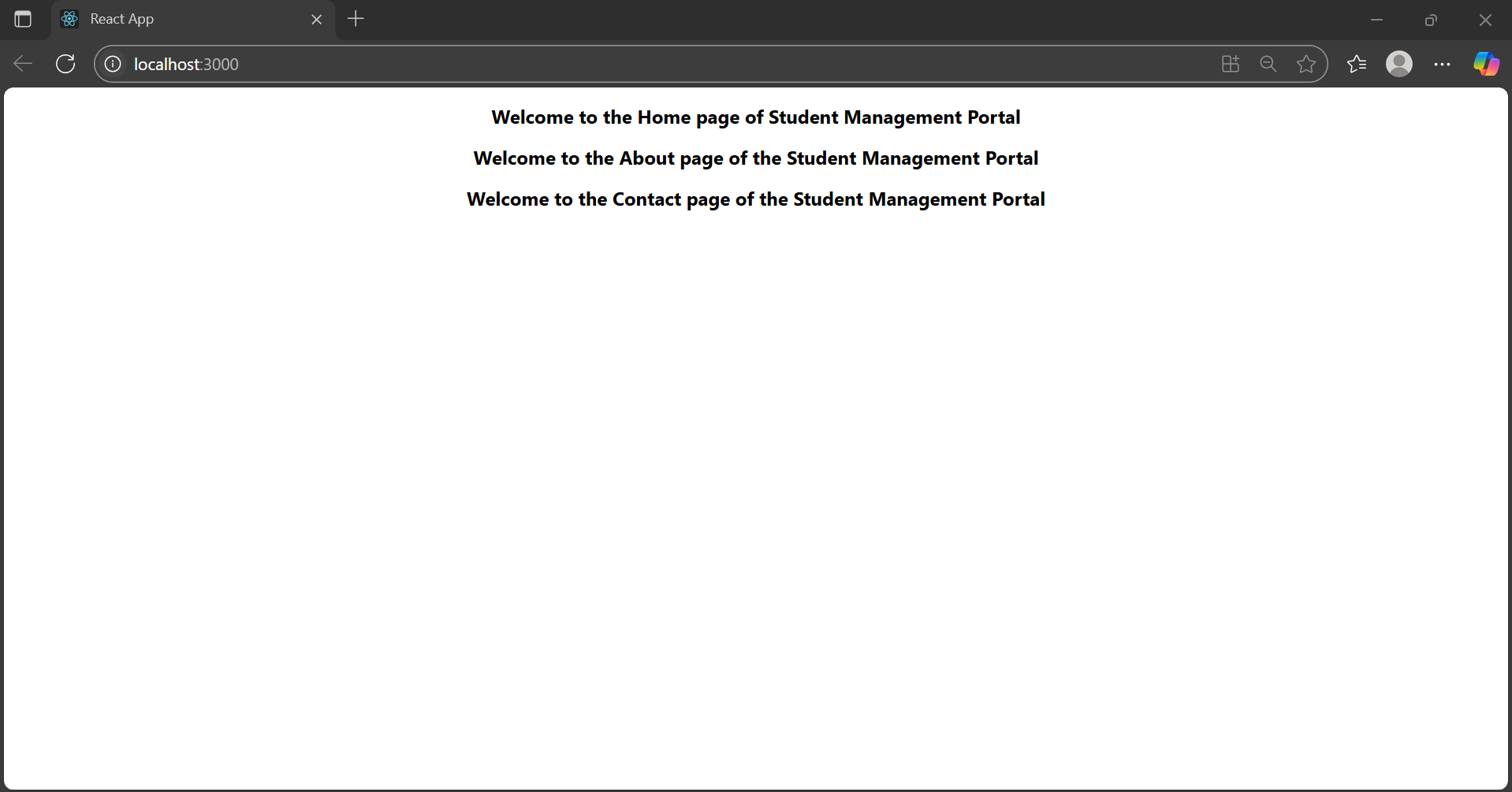




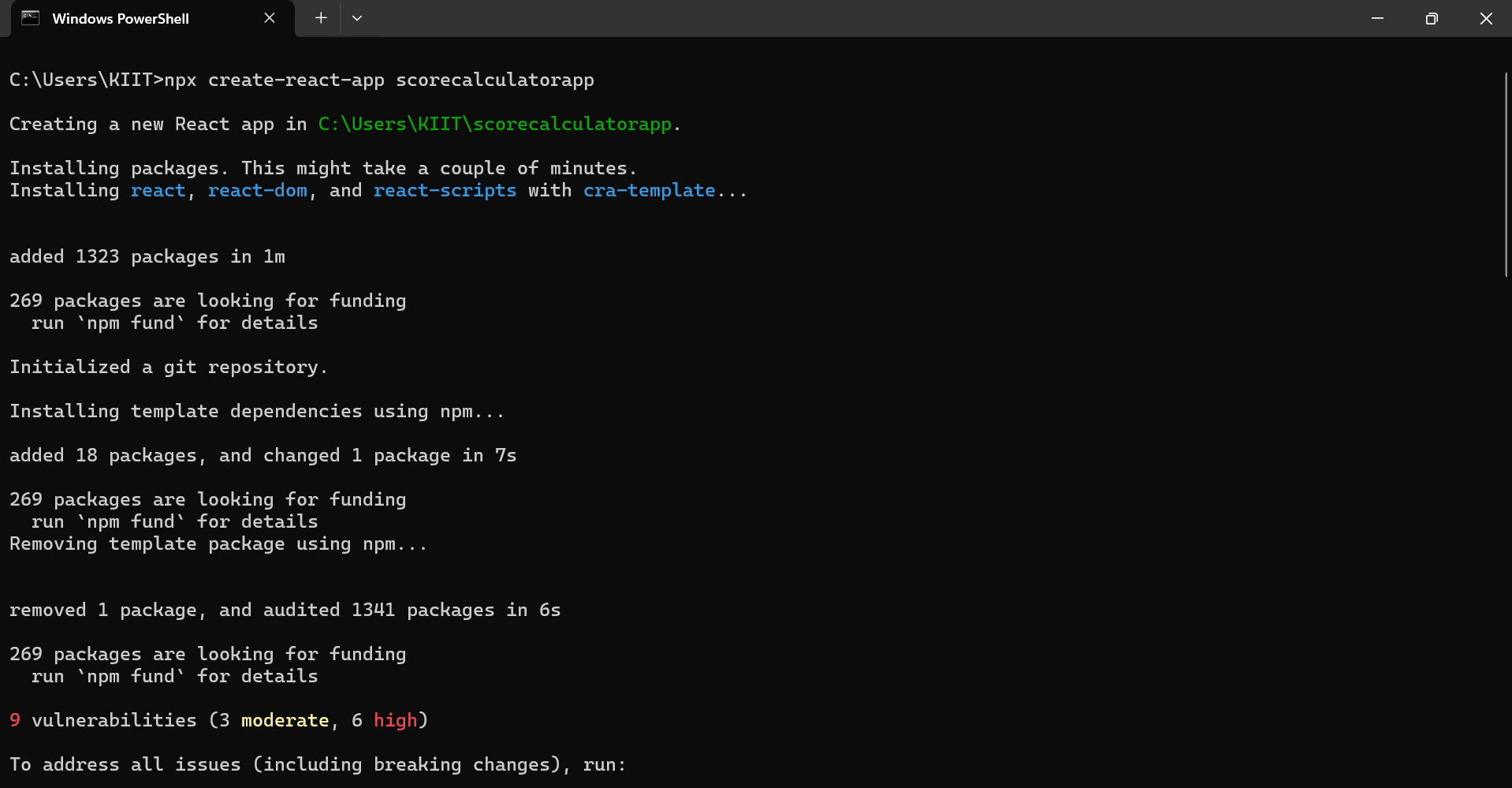


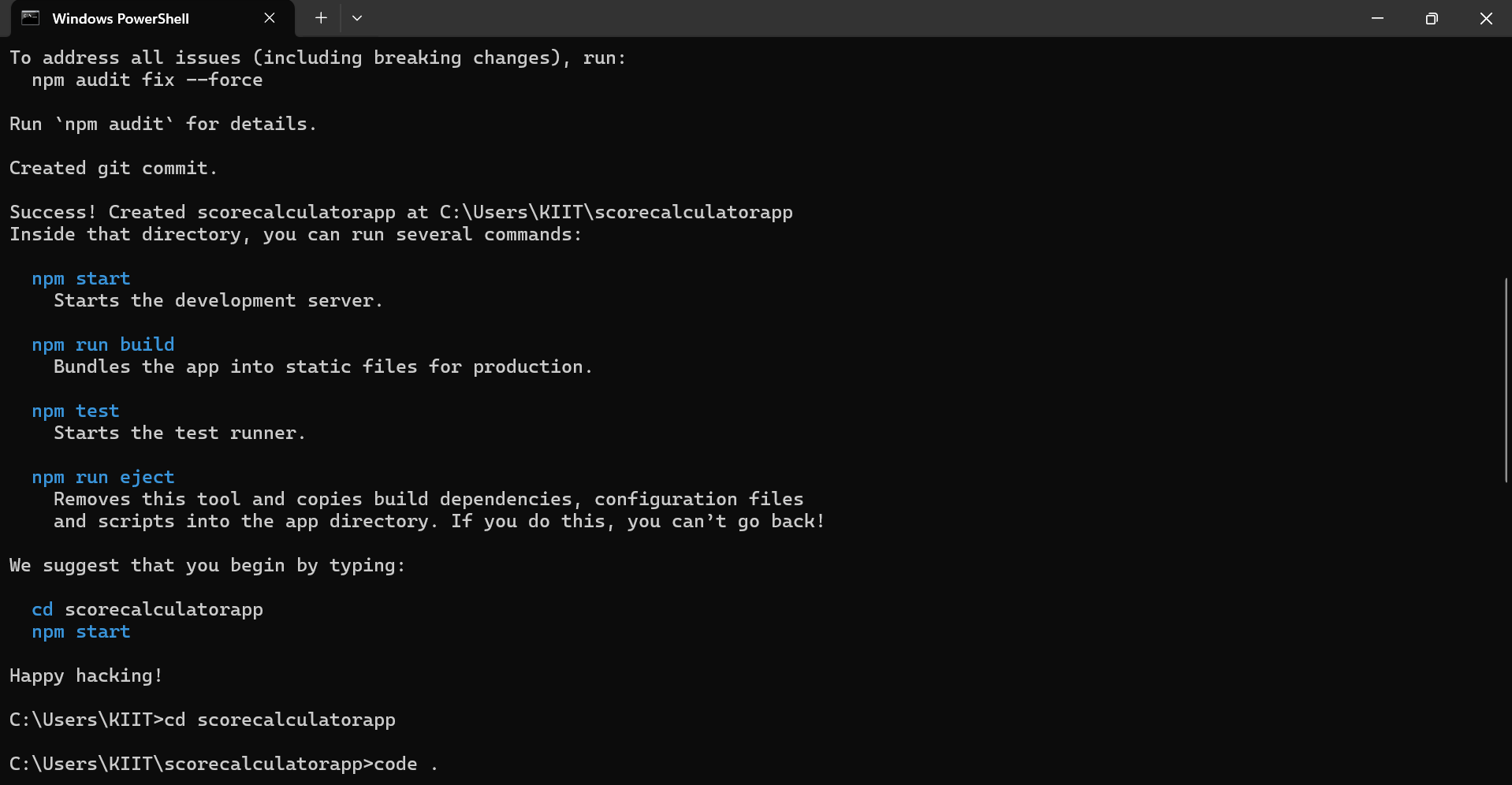


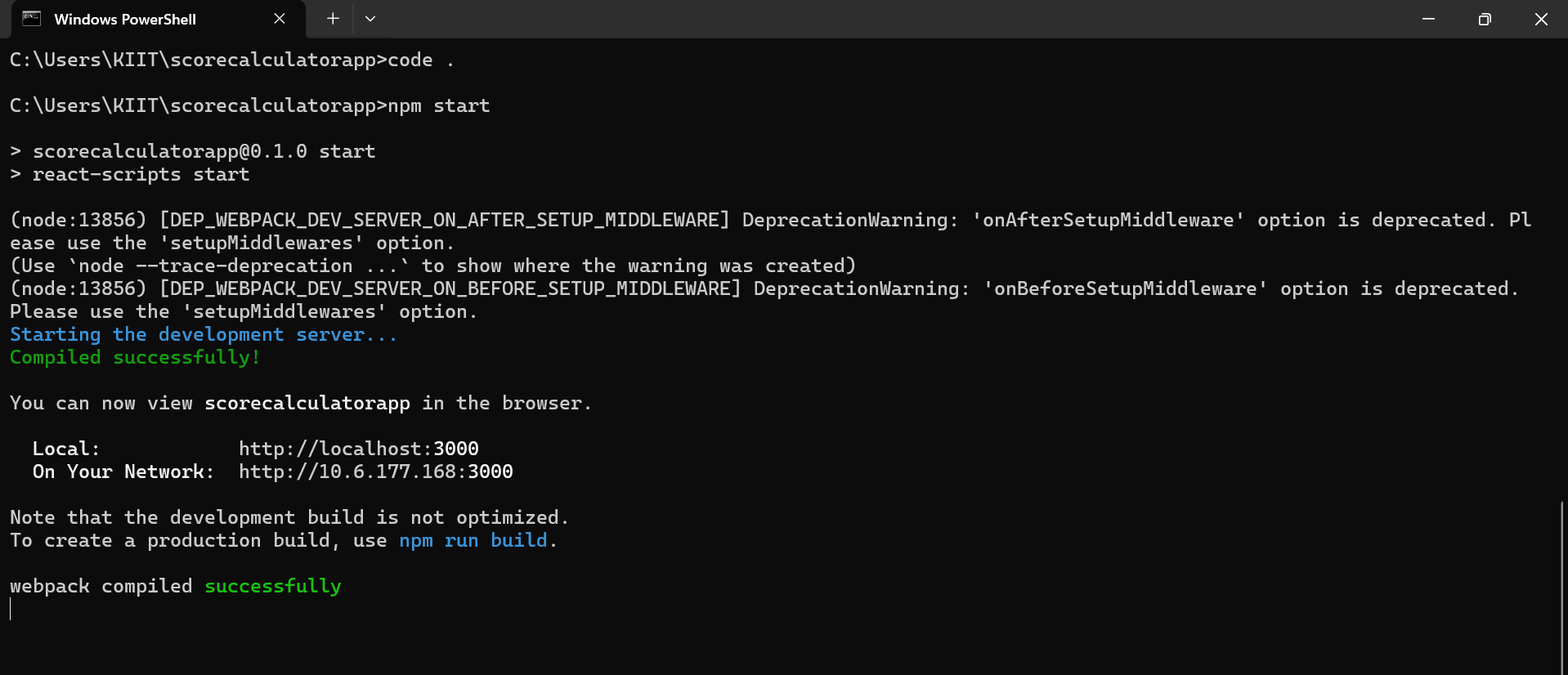


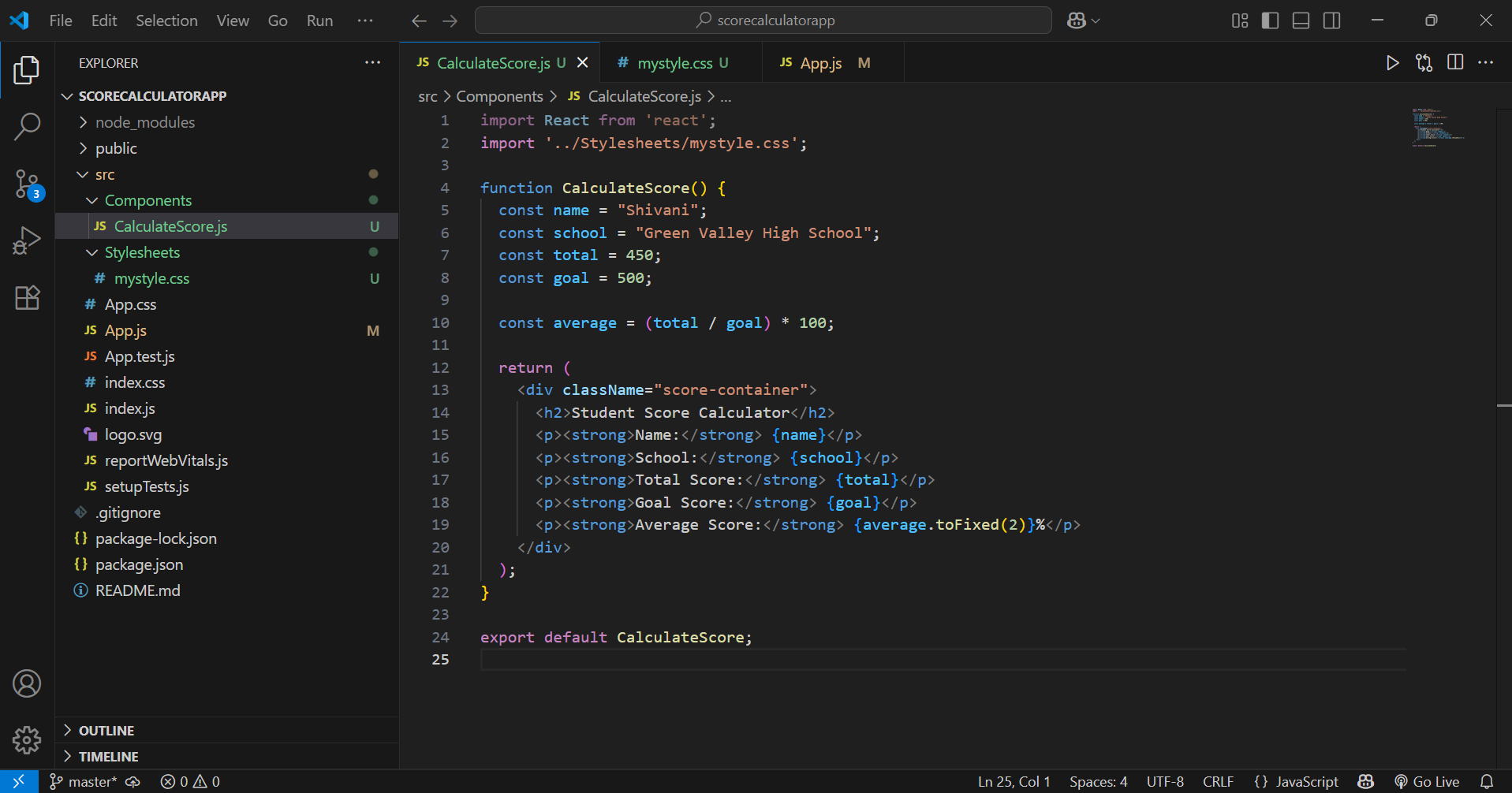


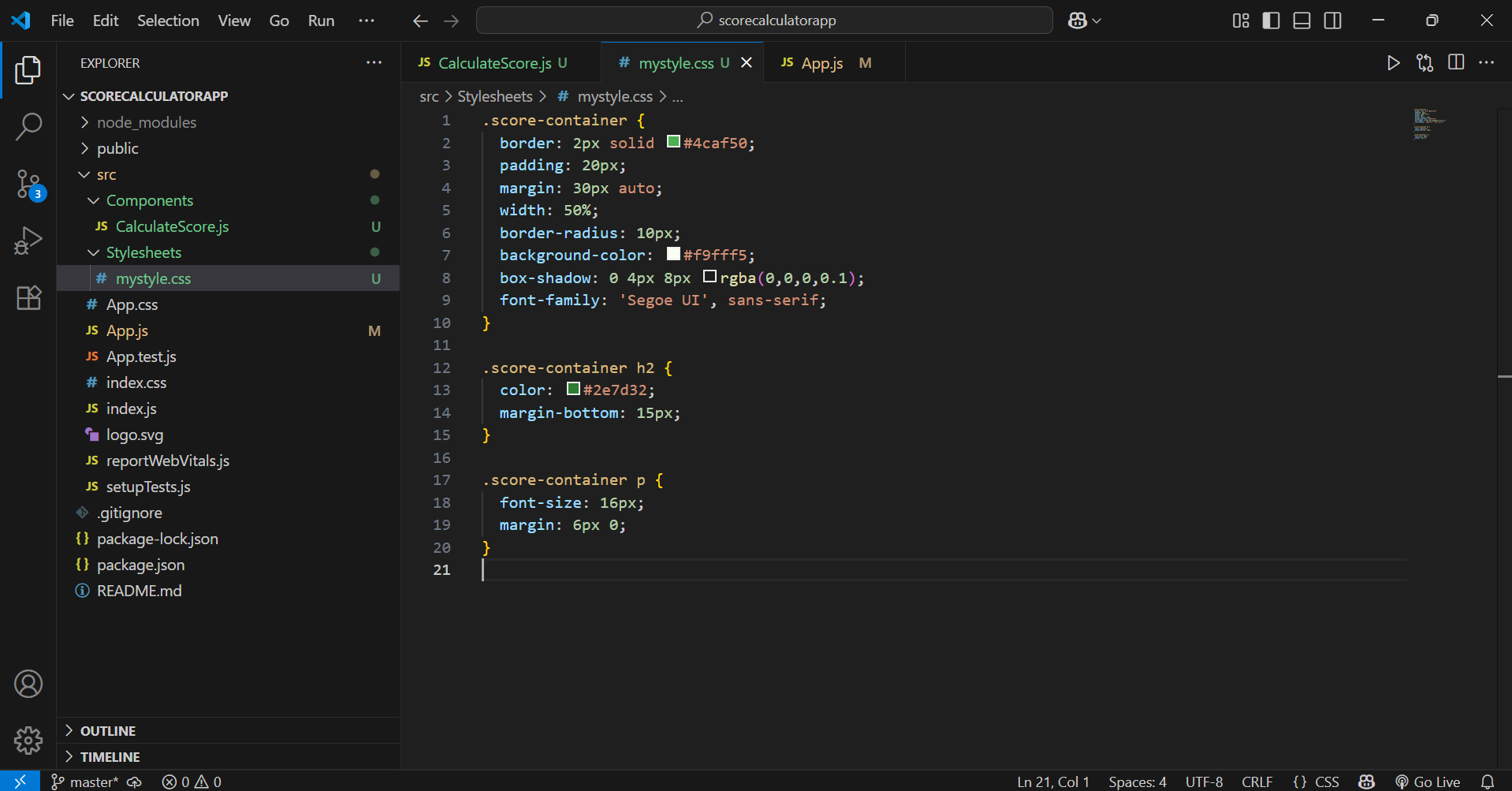
**3.**

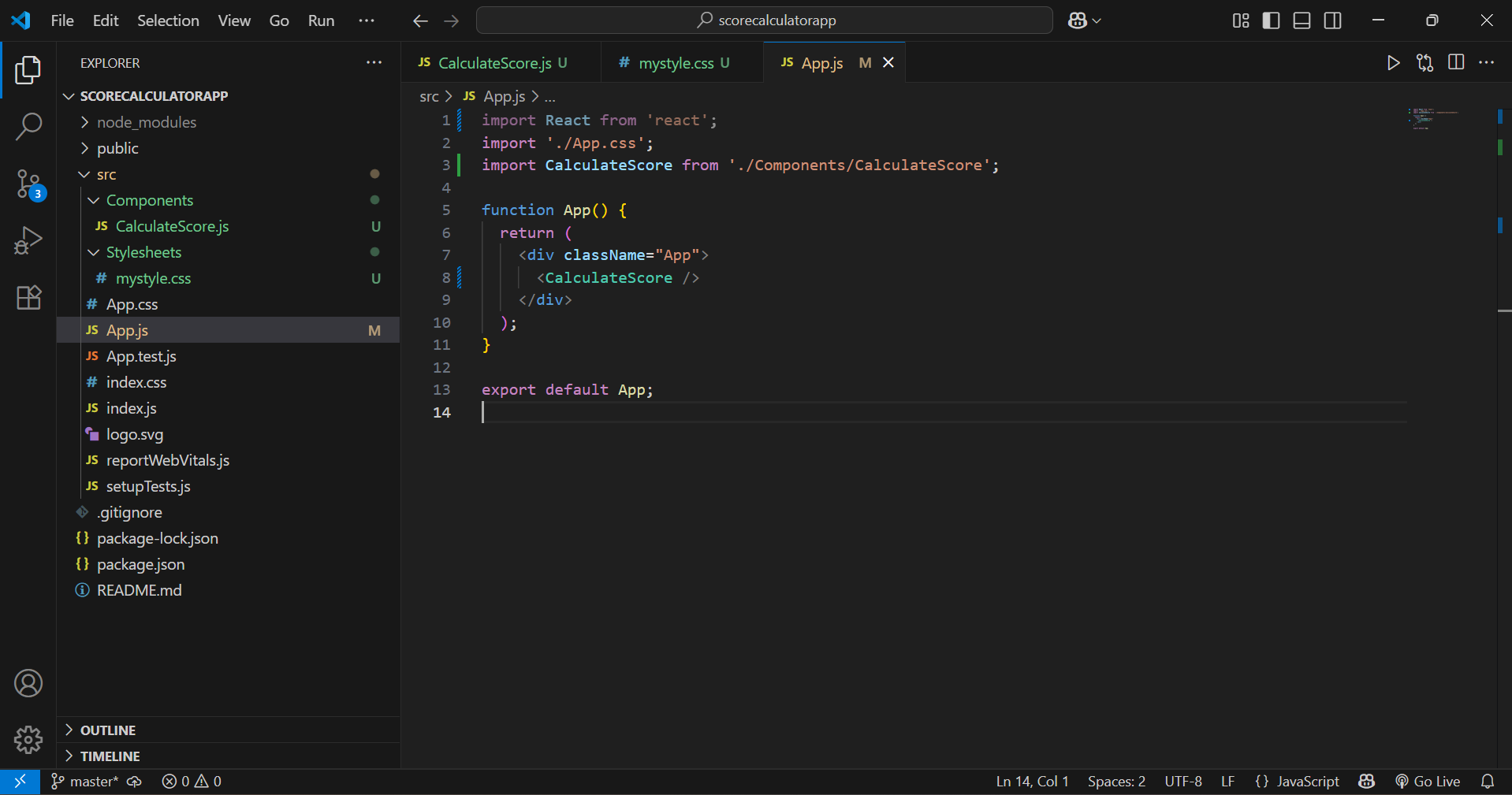


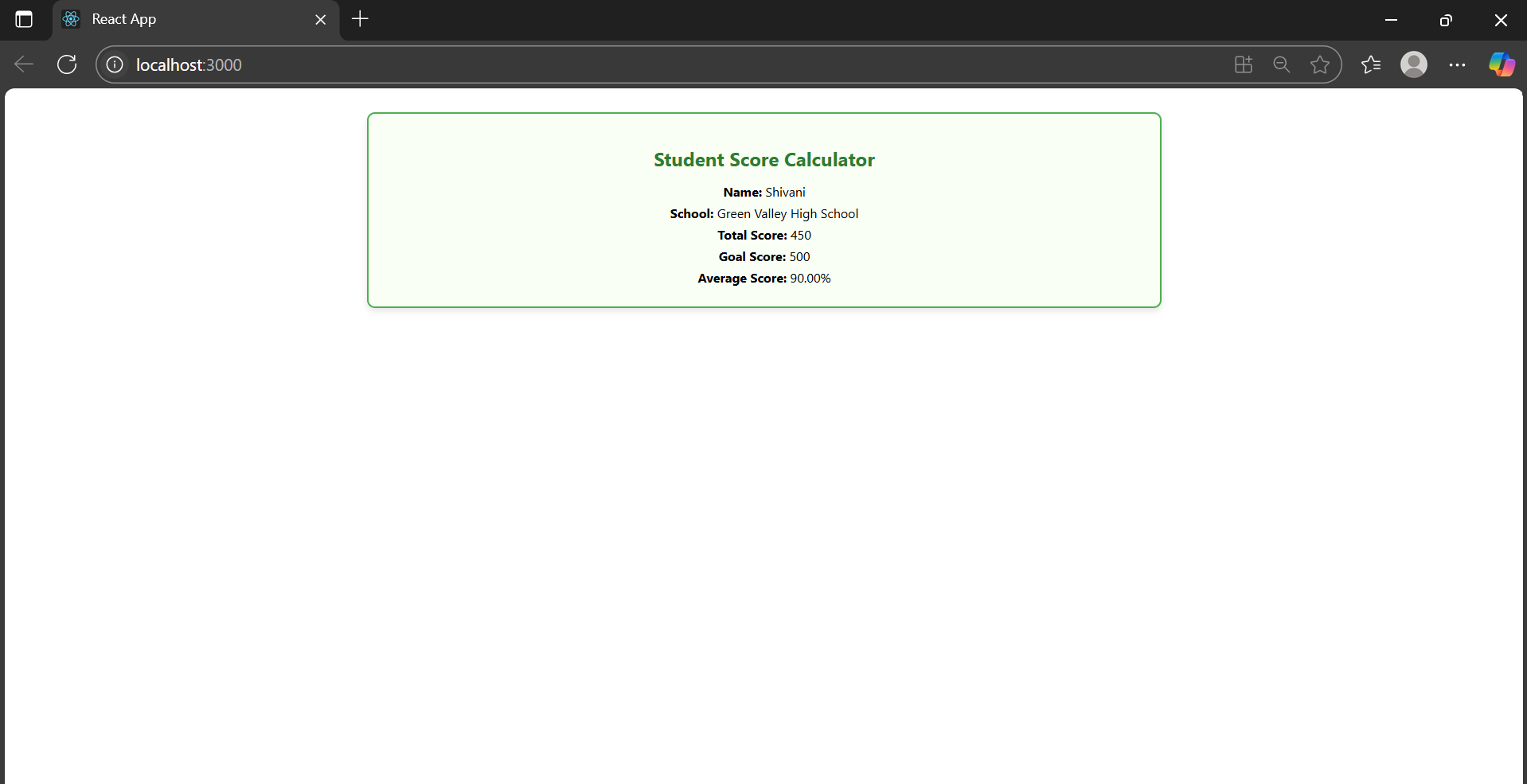












**4.**

**Explain the need and Benefits of component life cycle**

Solution:- In React, **lifecycle methods** allow you to run code at specific points in a component’s life: when it mounts, updates, or unmounts.

**Benefits:**

Fetch data after the component renders

Handle errors gracefully

Improve performance by controlling updates

Clean up side effects like timers or subscriptions

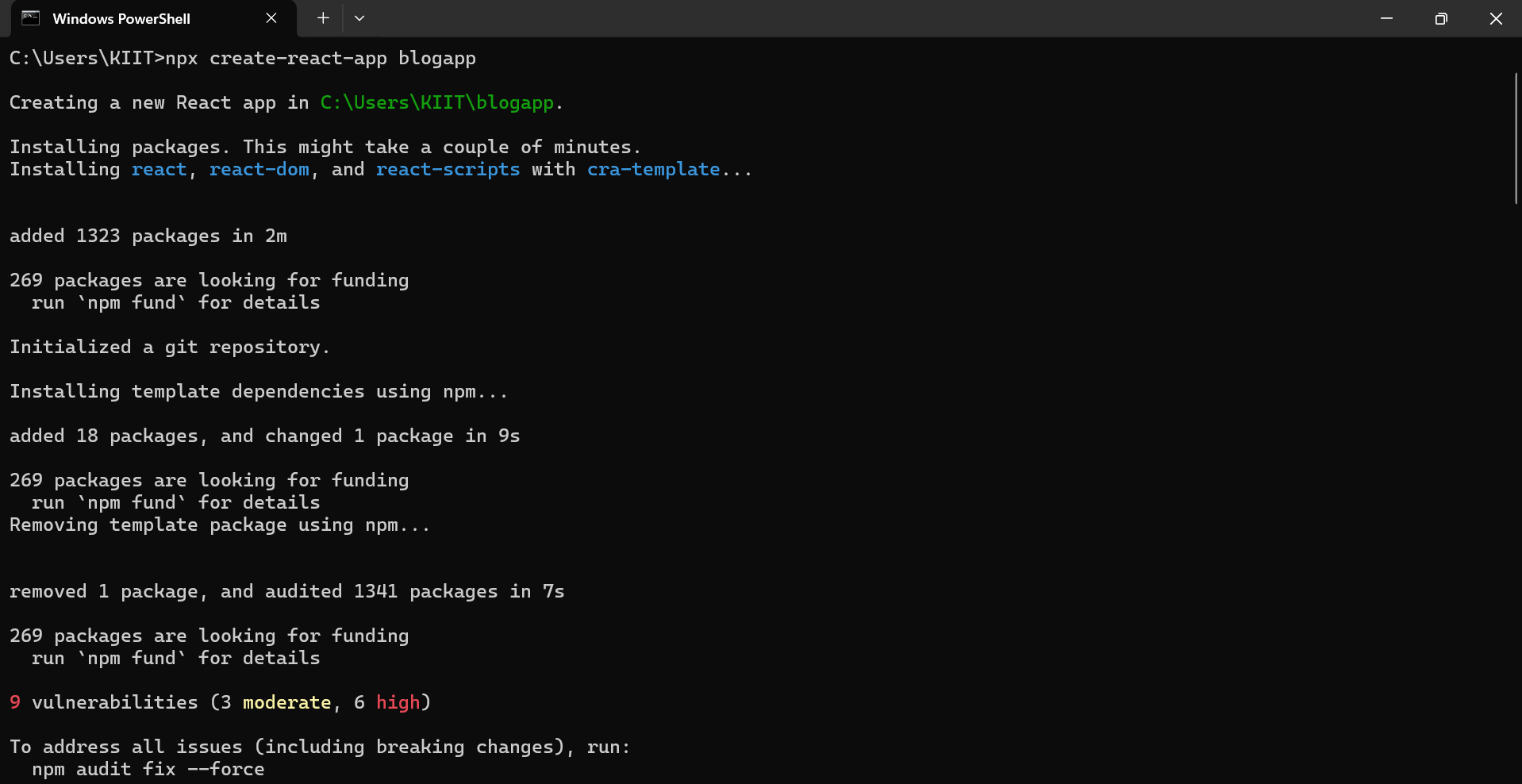
**Identify various life cycle hook methods**

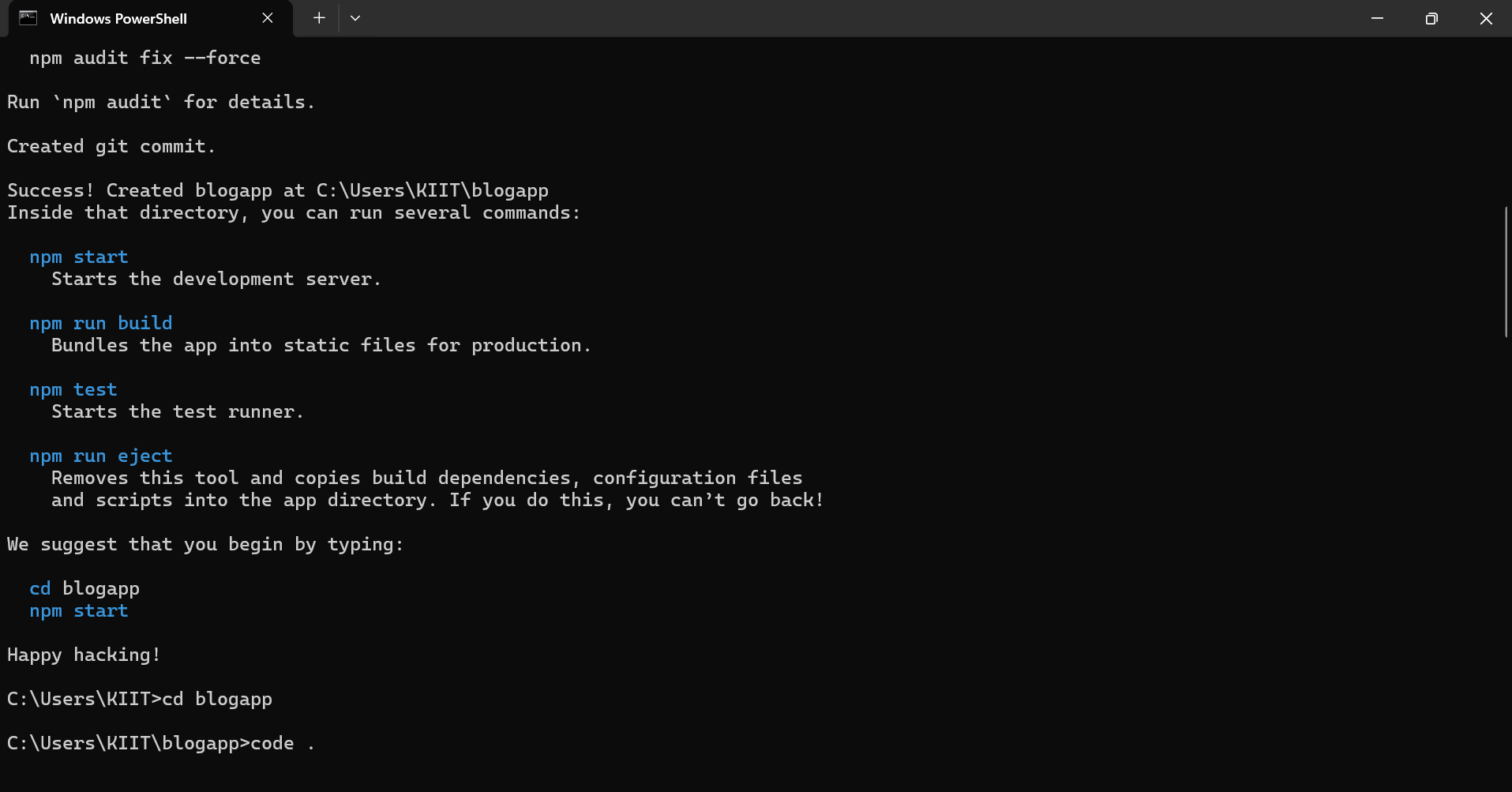
| **Lifecycle Phase** | **Method** | **Description** |
| --- | --- | --- |
| Mounting | constructor() | Initialize state |
|  | componentDidMount() | Runs once after component is added to DOM |
| Updating | componentDidUpdate() | Called after updates to props/state |
| Unmounting | componentWillUnmount() | Clean up (e.g., clear timers) |
| Error Handling | componentDidCatch() | Catch and handle rendering errors |

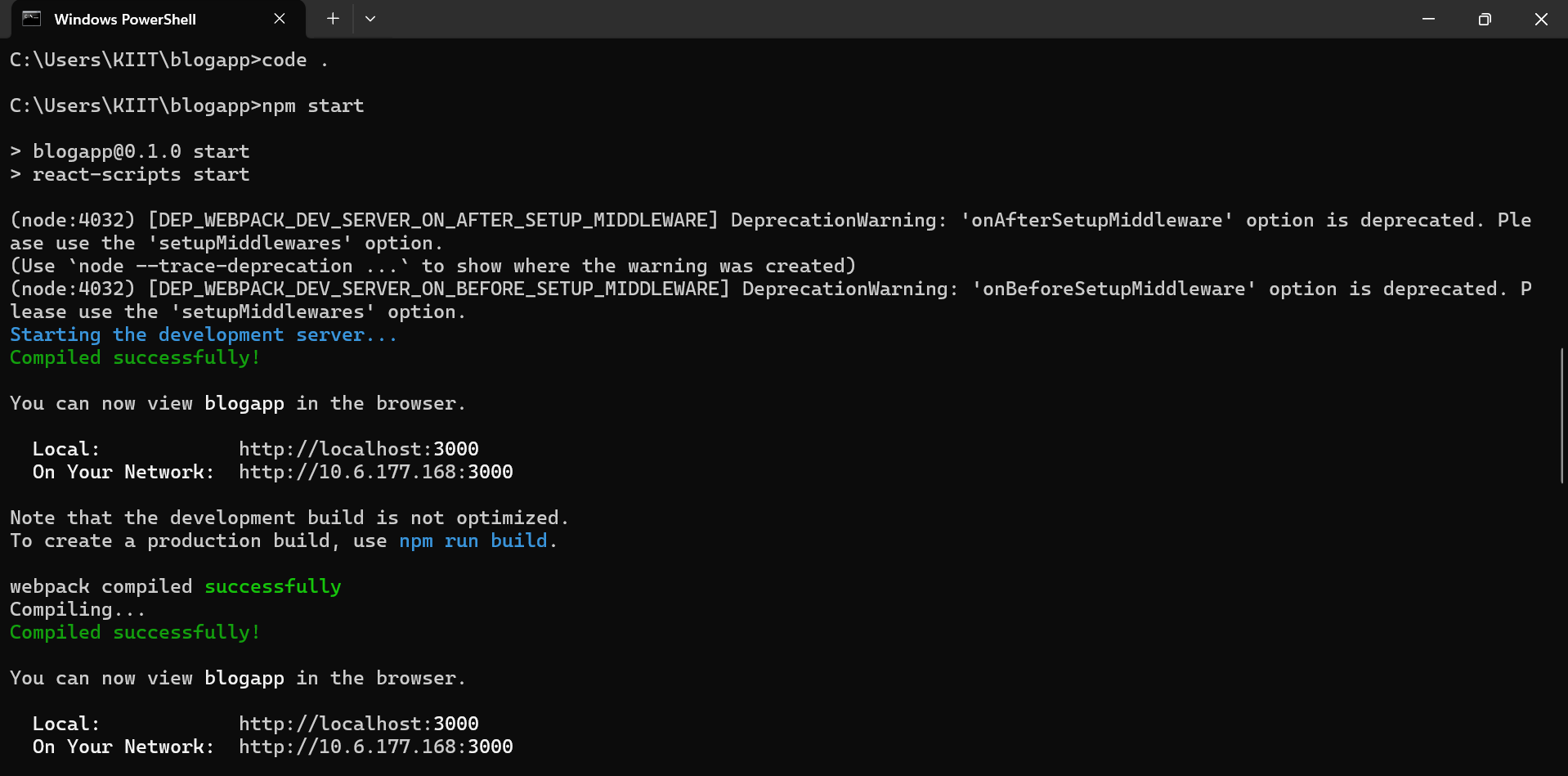
**List the sequence of steps in rendering a component**

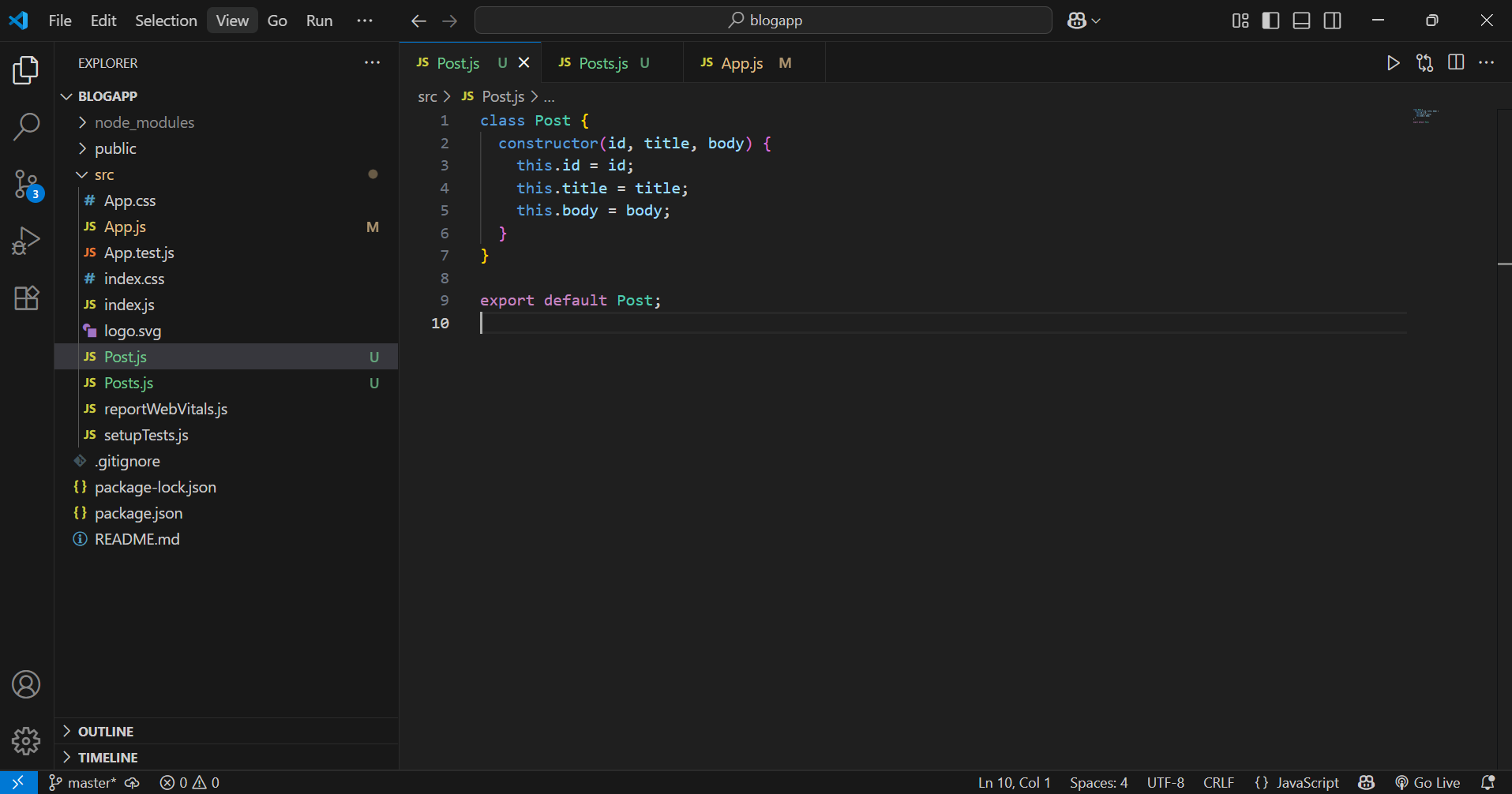
Solution:-

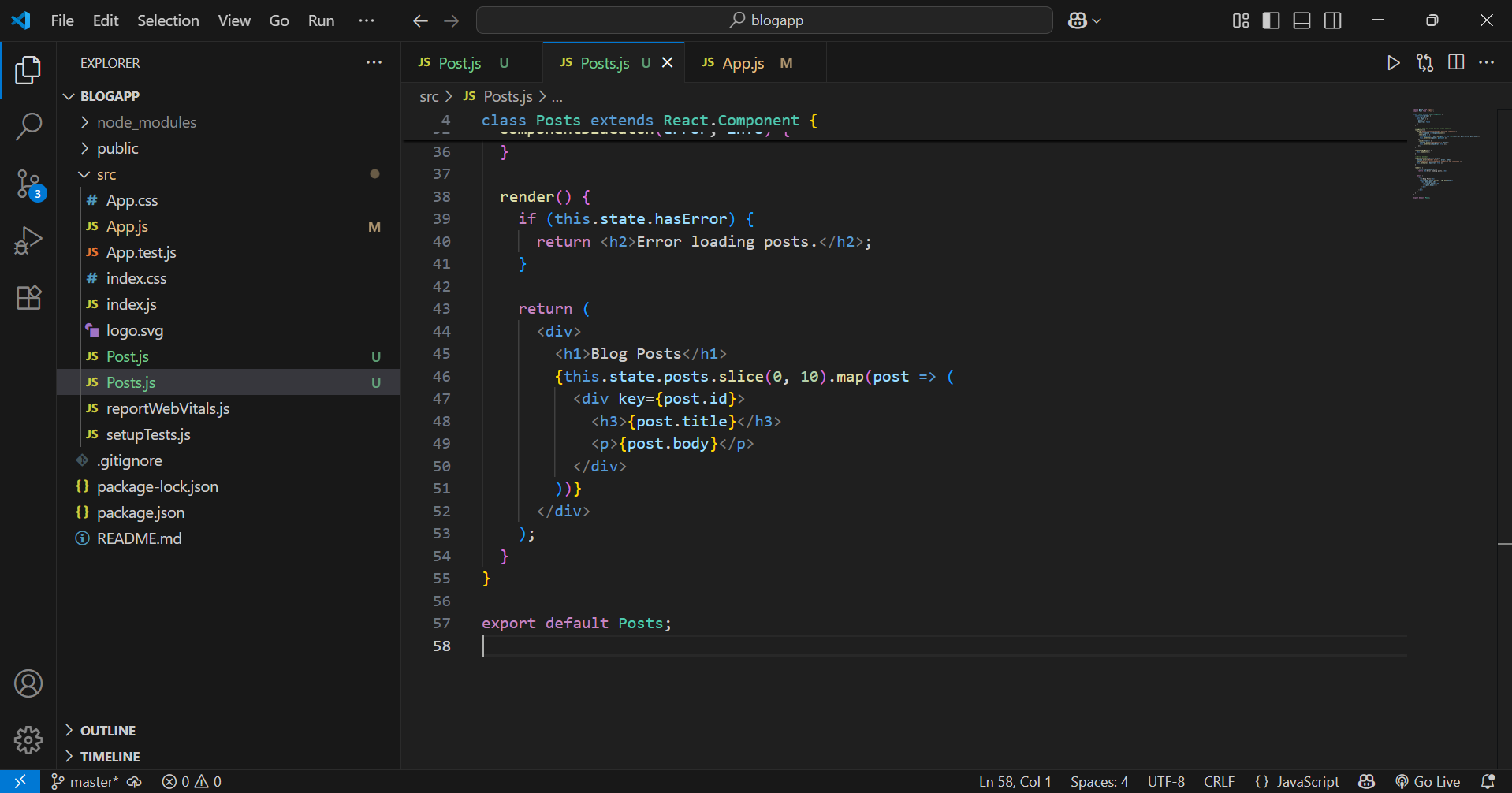
* **constructor()**
* **render()**
* **componentDidMount()**
* **(if state/props change)** → render() again
* \*\*componentDidUpdate()`
* \*\*componentWillUnmount()` on removal

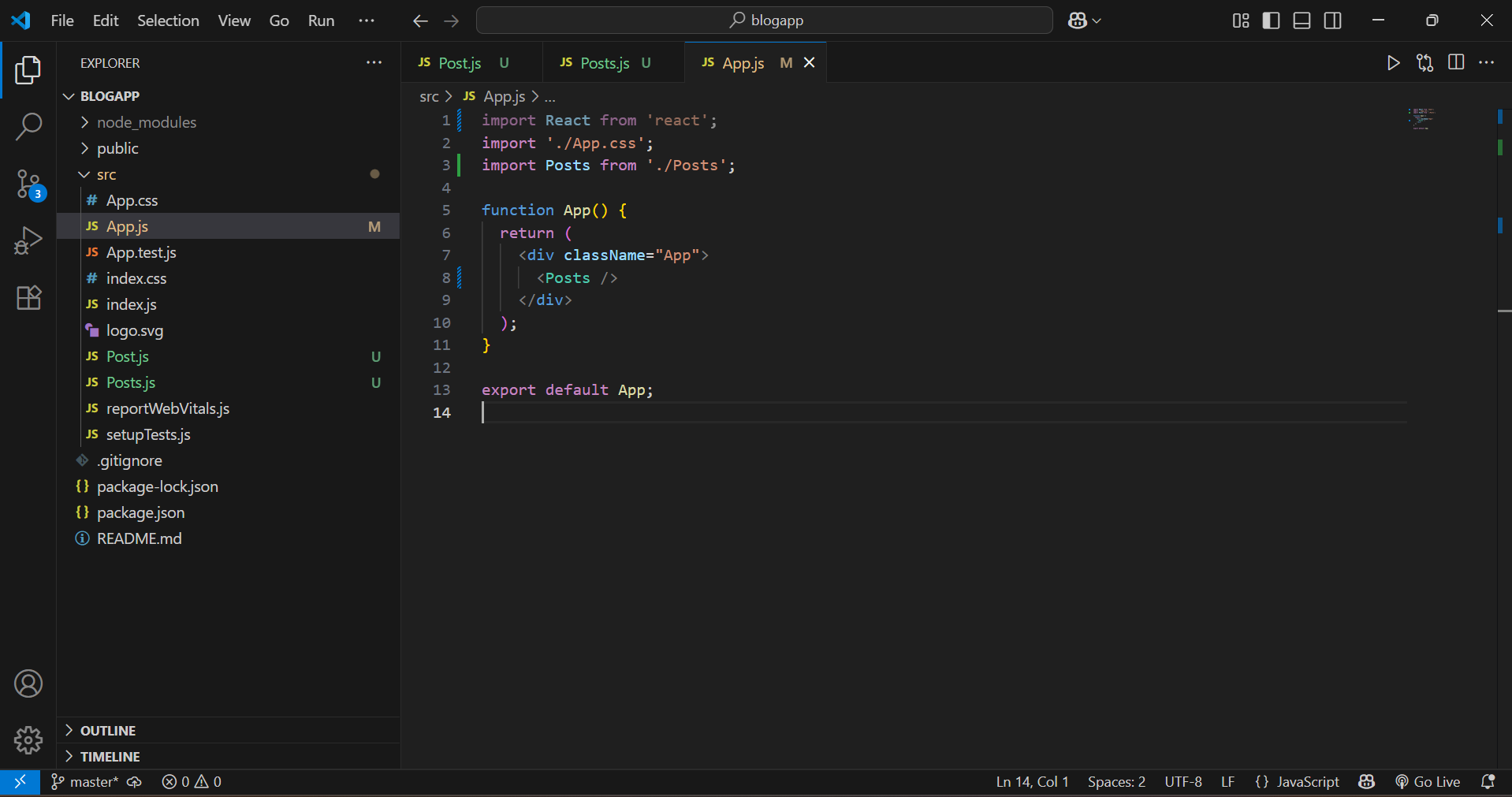


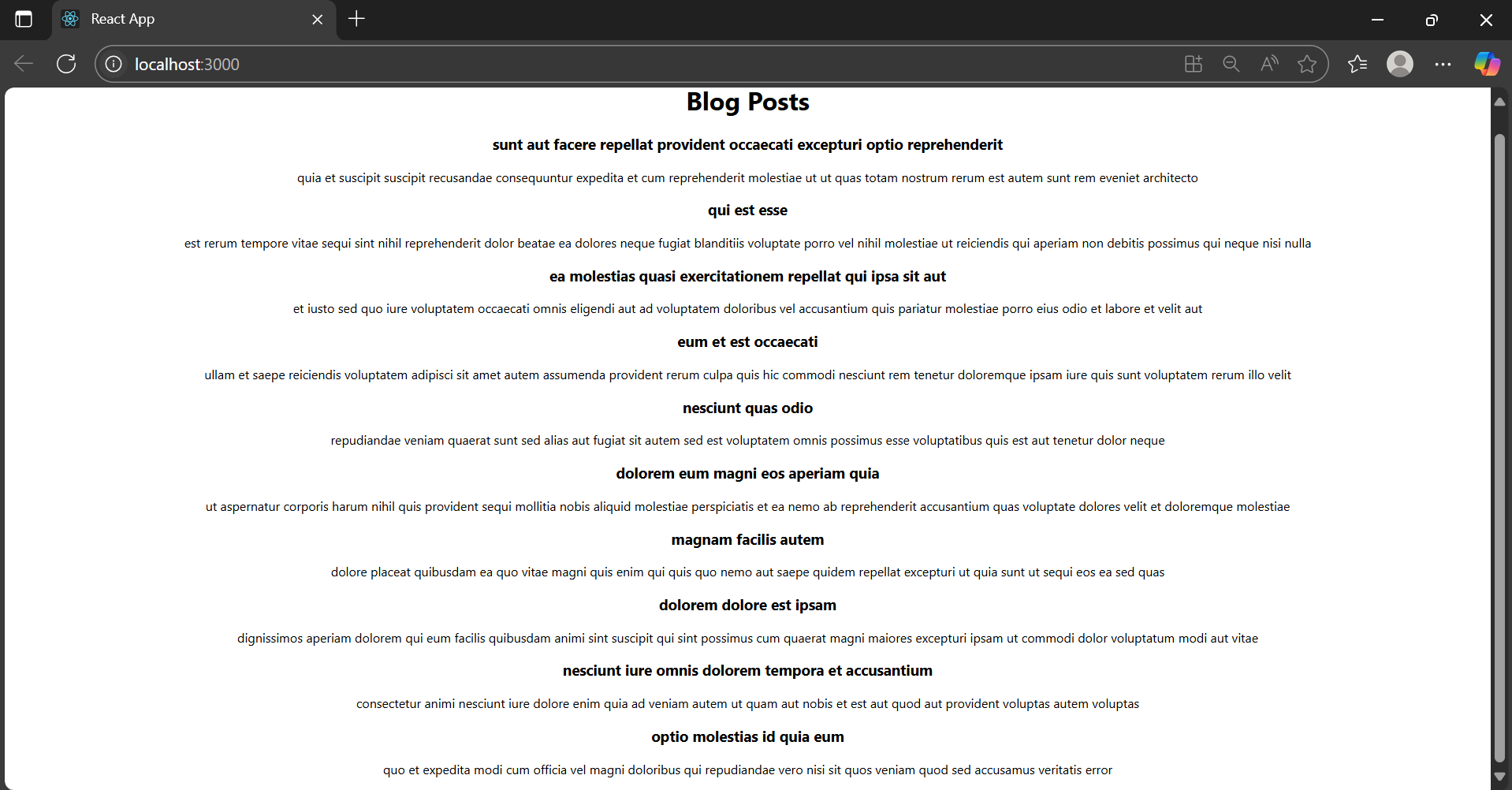












**5.**

