**4. ReactJS – HOL**

**Objectives**

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

It helps react know **when to run code** during different stages (like when a component mounts, updates, or unmounts).

Benefits:  
 i) Load data when the component appears (e.g., API calls)  
 ii) Clean up when the component goes away (e.g., stop timers)  
 iii) Catch errors during rendering

* **Identify various life cycle hook methods**

| **Stage** | **Method** | **Purpose** |
| --- | --- | --- |
| Mounting | constructor() | Initialize state and bind methods |
|  | componentDidMount() | Run code after component appears |
| Error | componentDidCatch() | Catch and handle rendering errors |
| Updating | componentDidUpdate() | Run code after props/state changes |
| Unmounting | componentWillUnmount() | Cleanup before component is removed |

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

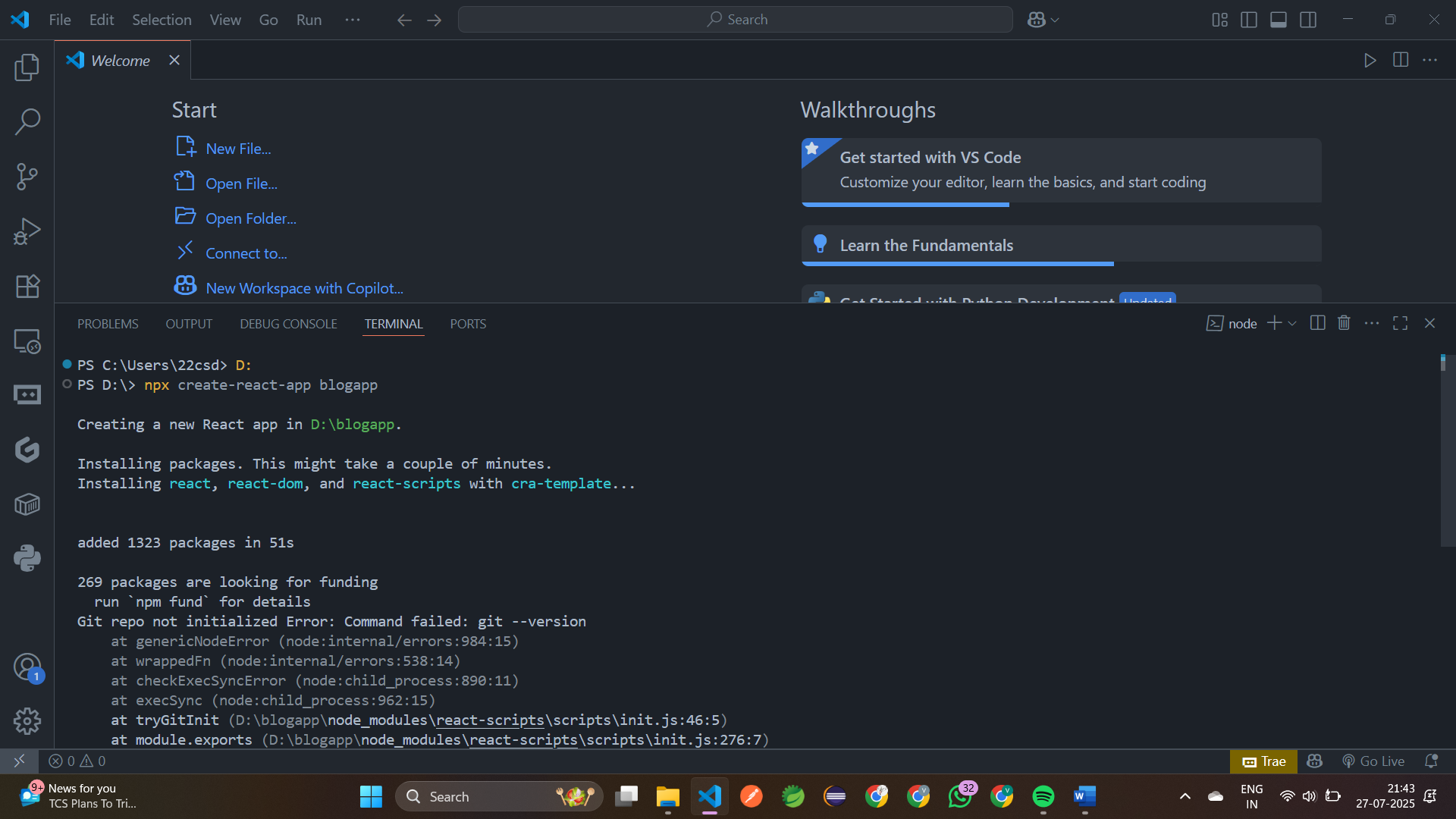
1. **constructor()** – setup initial state
2. **render()** – returns the UI
3. **componentDidMount()** – runs after first render
4. On state/props change i) **render()** – updates UI ii) **componentDidUpdate()** – runs after update
5. When removed i) **componentWillUnmount()** – clean up
6. On error i) **componentDidCatch()** – handle the error

**EXERCISE – 2:** Creation of a BlogApp

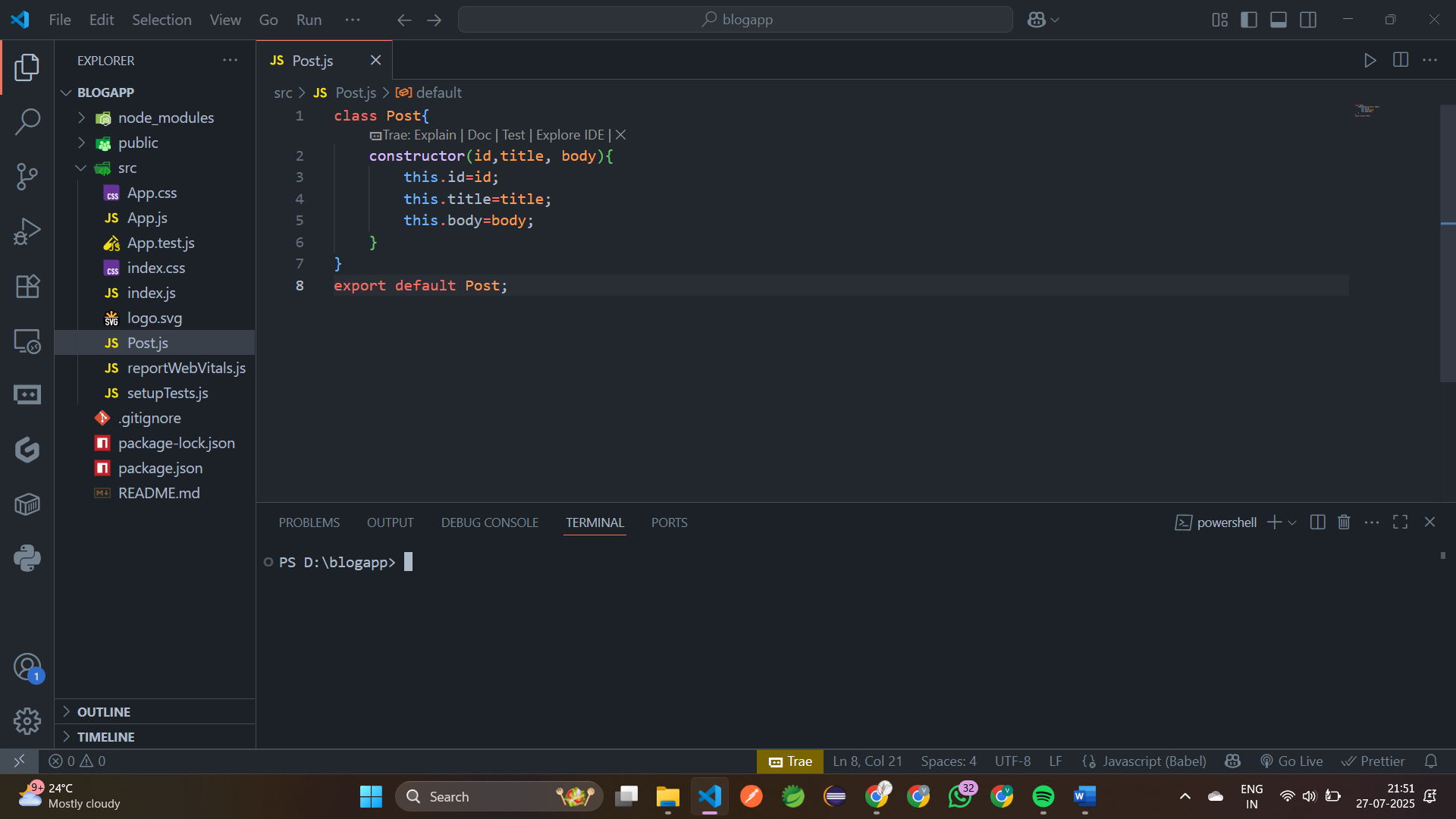
Step 1: Open **VS Code**

Step 2: Create a New React App using **npx create-react-app blogapp**

step 3: Navigate into the project folder using **cd blogapp**



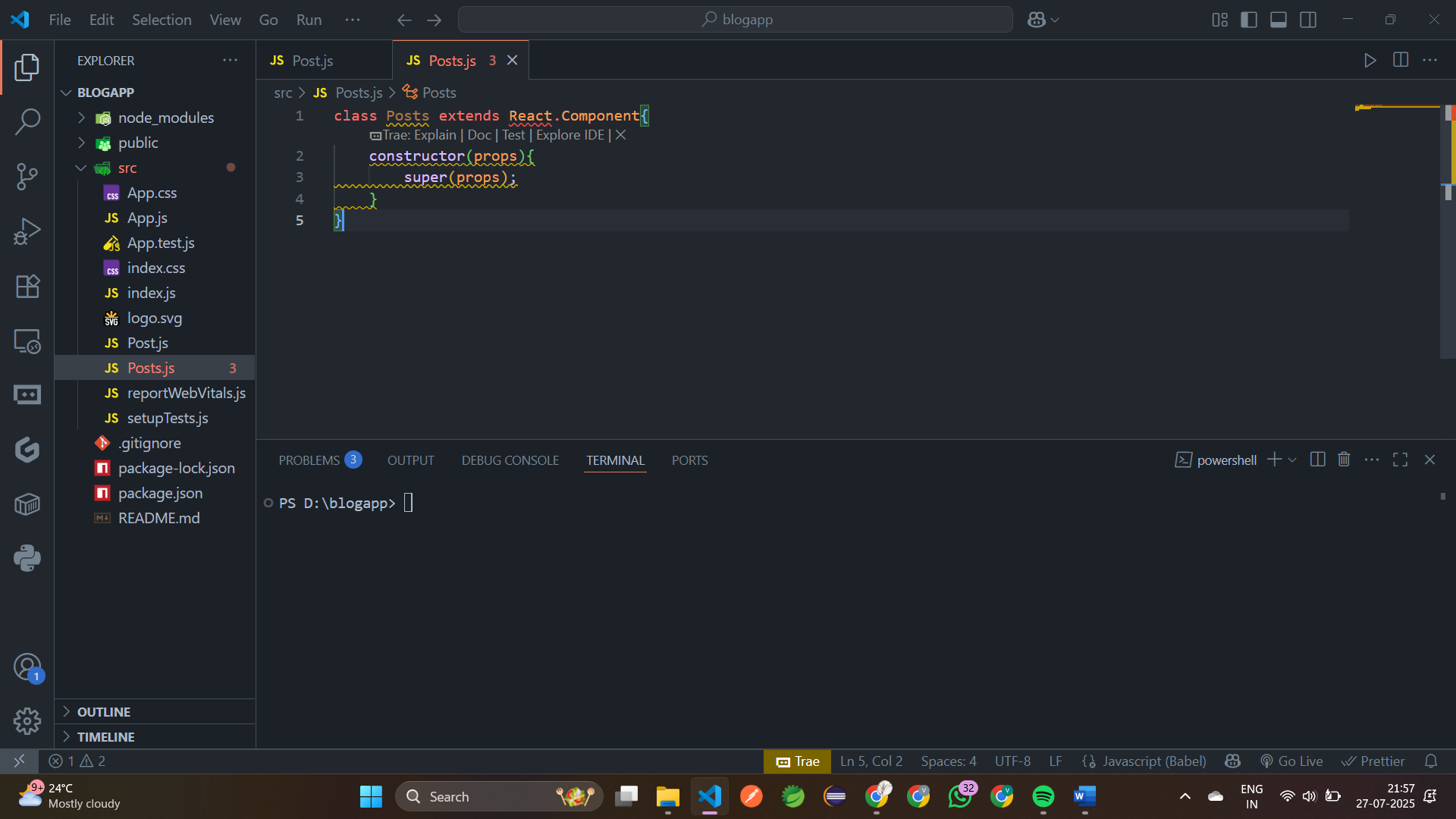
step 4: Create a new file named as **Post.js** in **src folder**



step 5: Inside **src/Post.js**

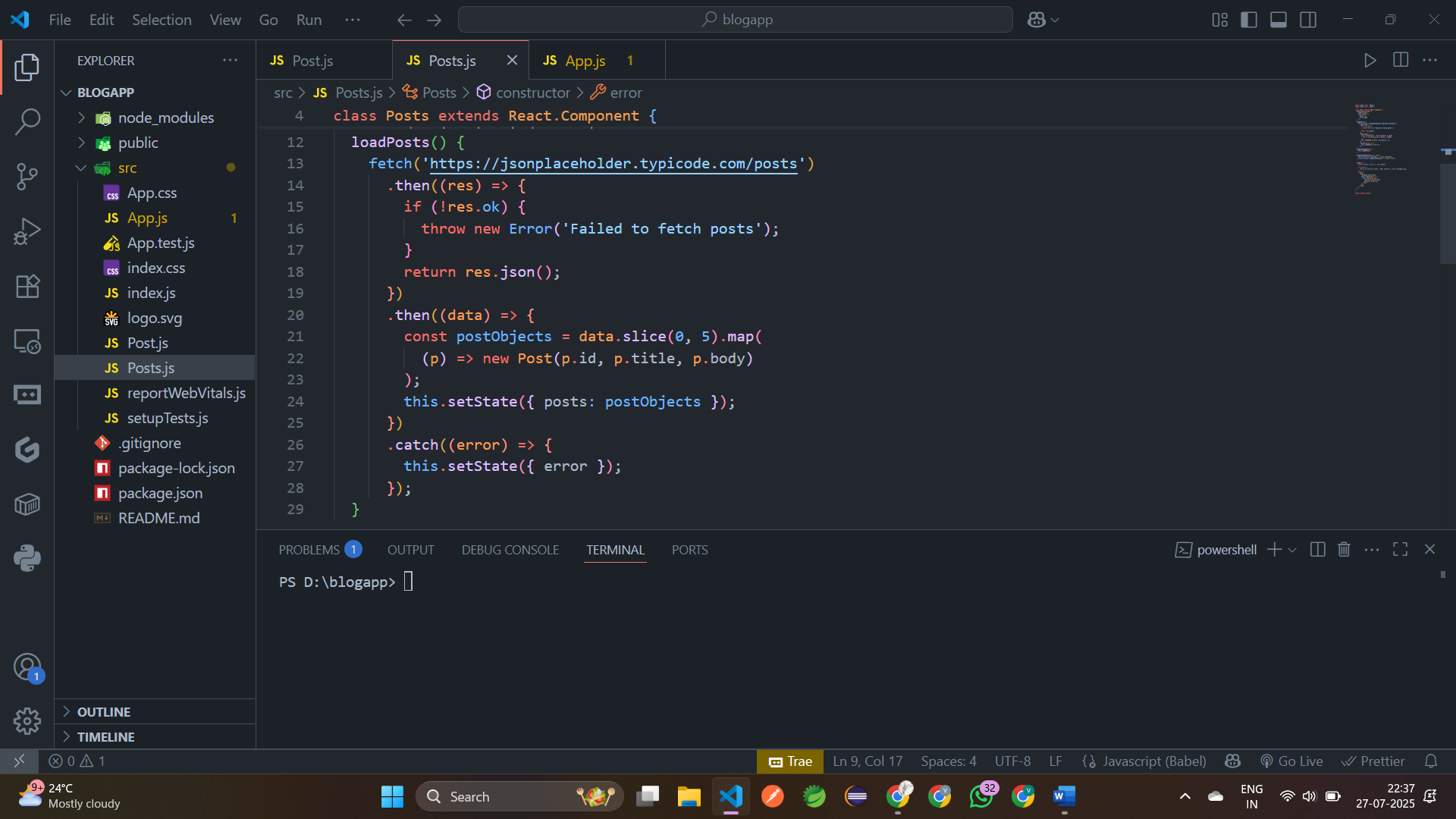
class Post{  
     constructor(id,title, body){  
         this.id=id;  
         this.title=title;  
         this.body=body;  
     }  
 }export default Post;

step 6: Create a class based component named **Posts** named **Post.js** in src folder

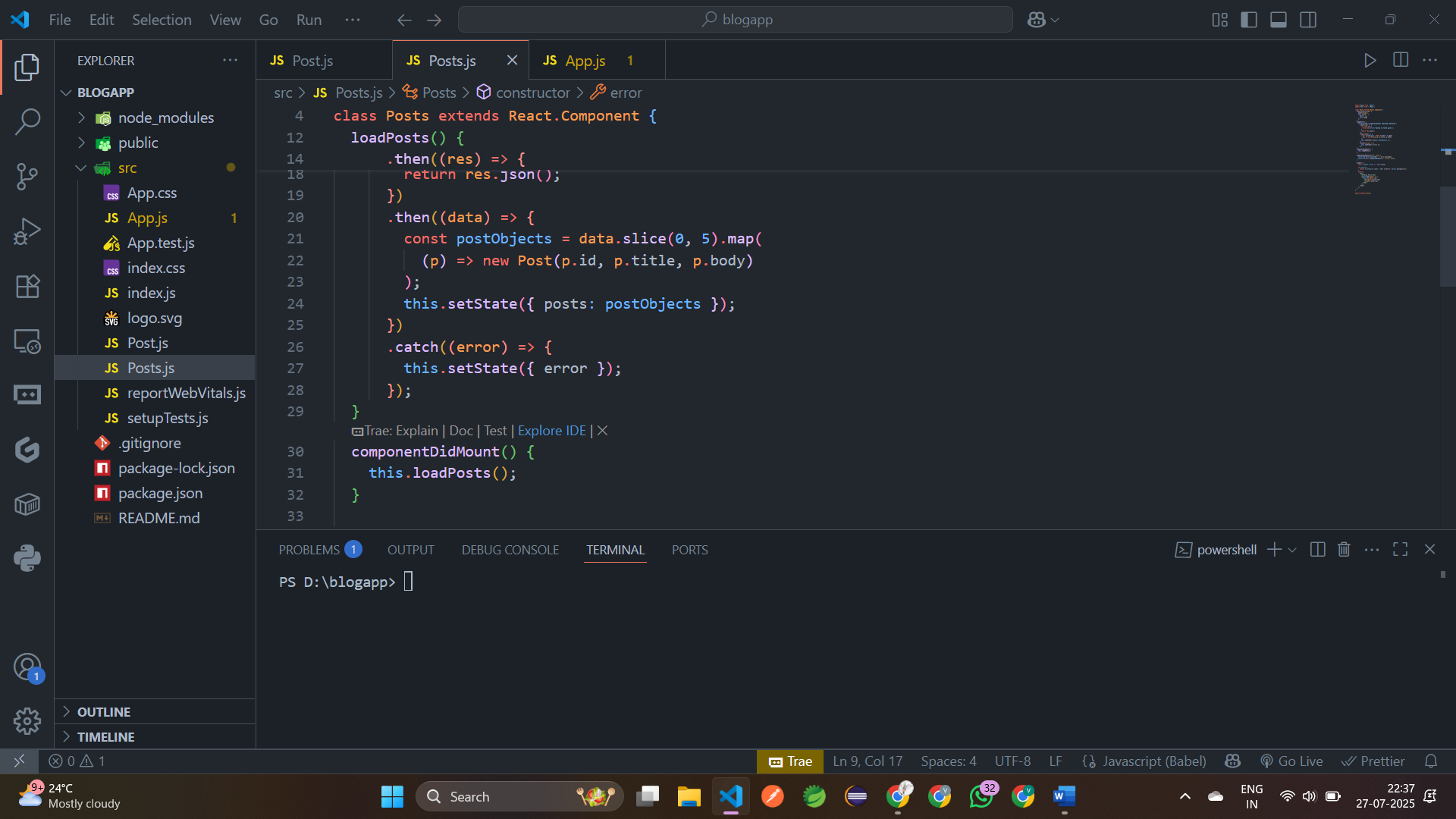


step 7: Create a new method in component with the name as **loadPosts()**

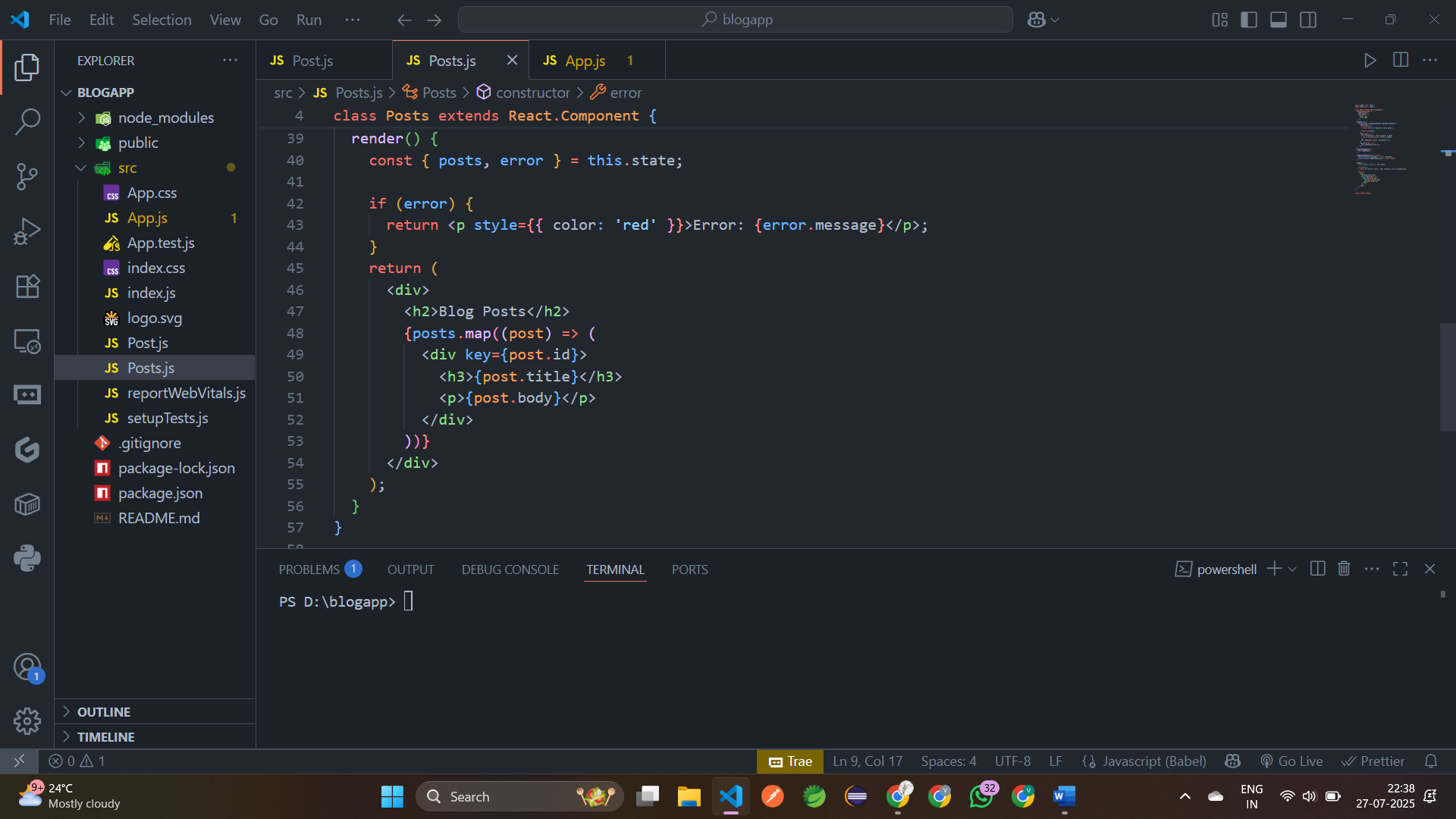
class Posts extends React.Component{  
     constructor(props){  
         super(props);    }  
     loadPosts(){     
 }  
 }



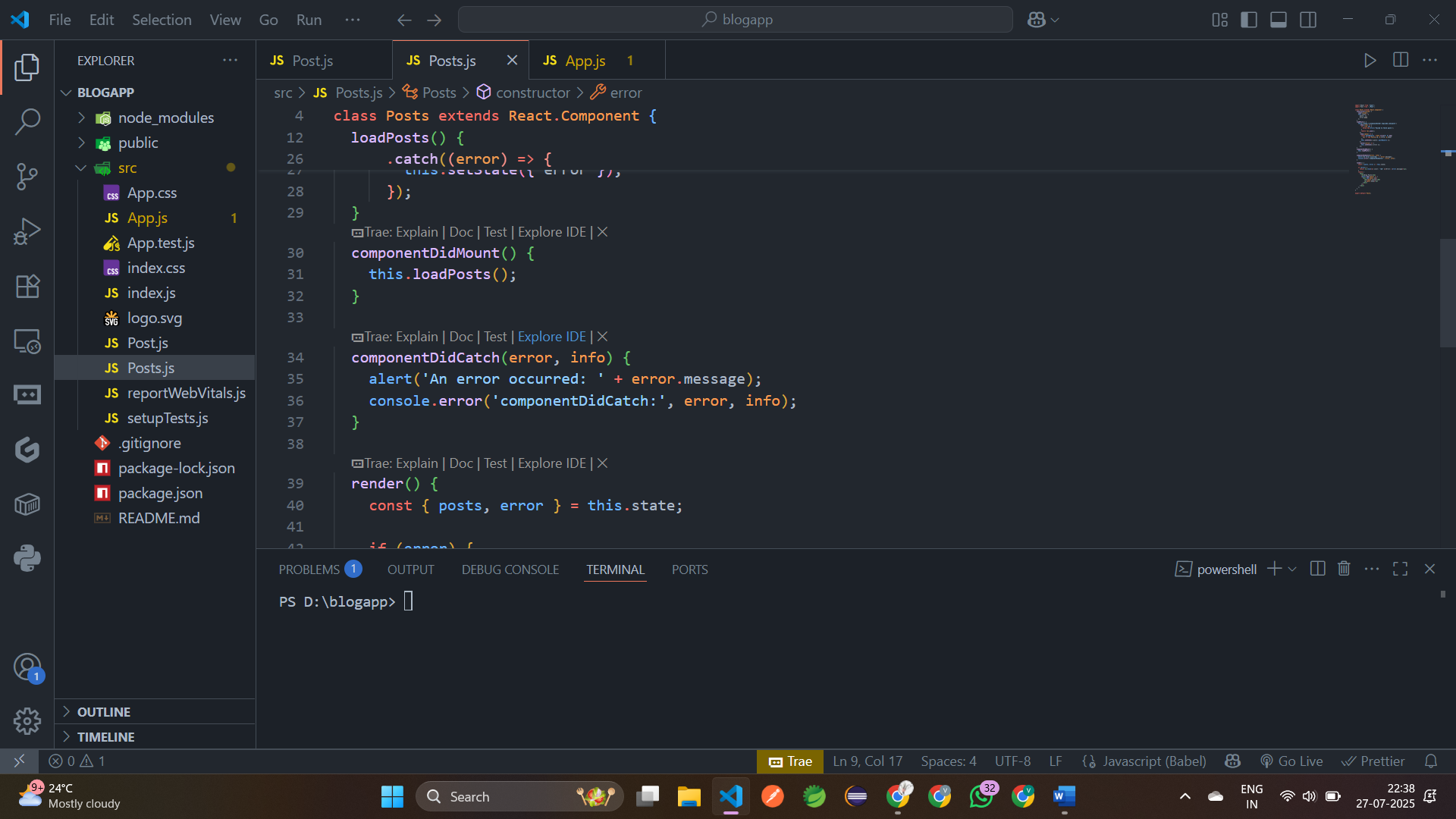
step 8: Implement the **componentDidMount()** hook to make calls to **loadPosts()** which will fetch the posts



step 9: Implement the **render()** which will display the title and post of posts in html page using heading and paragraphs respectively.



step 10: Define a **componentDidCatch()** method which will be responsible for displaying any error happing in the component as alert messages.



step 11: **Post.js**

import React from 'react';  
 import Post from './Post';  
 class Posts extends React.Component {  
   constructor(props) {  
     super(props);  
     this.state = {  
       posts: [],  
       error: null     };   }  
   loadPosts() {  
     fetch('https://jsonplaceholder.typicode.com/posts')  
       .then((res) => {  
         if (!res.ok) {  
           throw new Error('Failed to fetch posts');         }  
         return res.json();      })  
      .then((data) => {  
         const postObjects = data.slice(0, 5).map(  
           (p) => new Post(p.id, p.title, p.body)        );  
         this.setState({ posts: postObjects });      })  
       .catch((error) => {  
         this.setState({ error });       });}  
   componentDidMount() {  
     this.loadPosts();   }   
 componentDidCatch(error, info) {  
         alert('An error occurred: ' + error.message);  
 console.error('componentDidCatch:', error, info);   }  
   render() {  
      const { posts, error } = this.state;  
     if (error) {  
       return <p style={{ color: 'red' }}>Error: {error.message}</p>;    }  
     return (  
       <div>         <h2>Blog Posts</h2>  
       {posts.map((post) => (  
           <div key={post.id}>  
             <h3>{post.title}</h3>  
             <p>{post.body}</p>          </div>

      </div>    );  }}

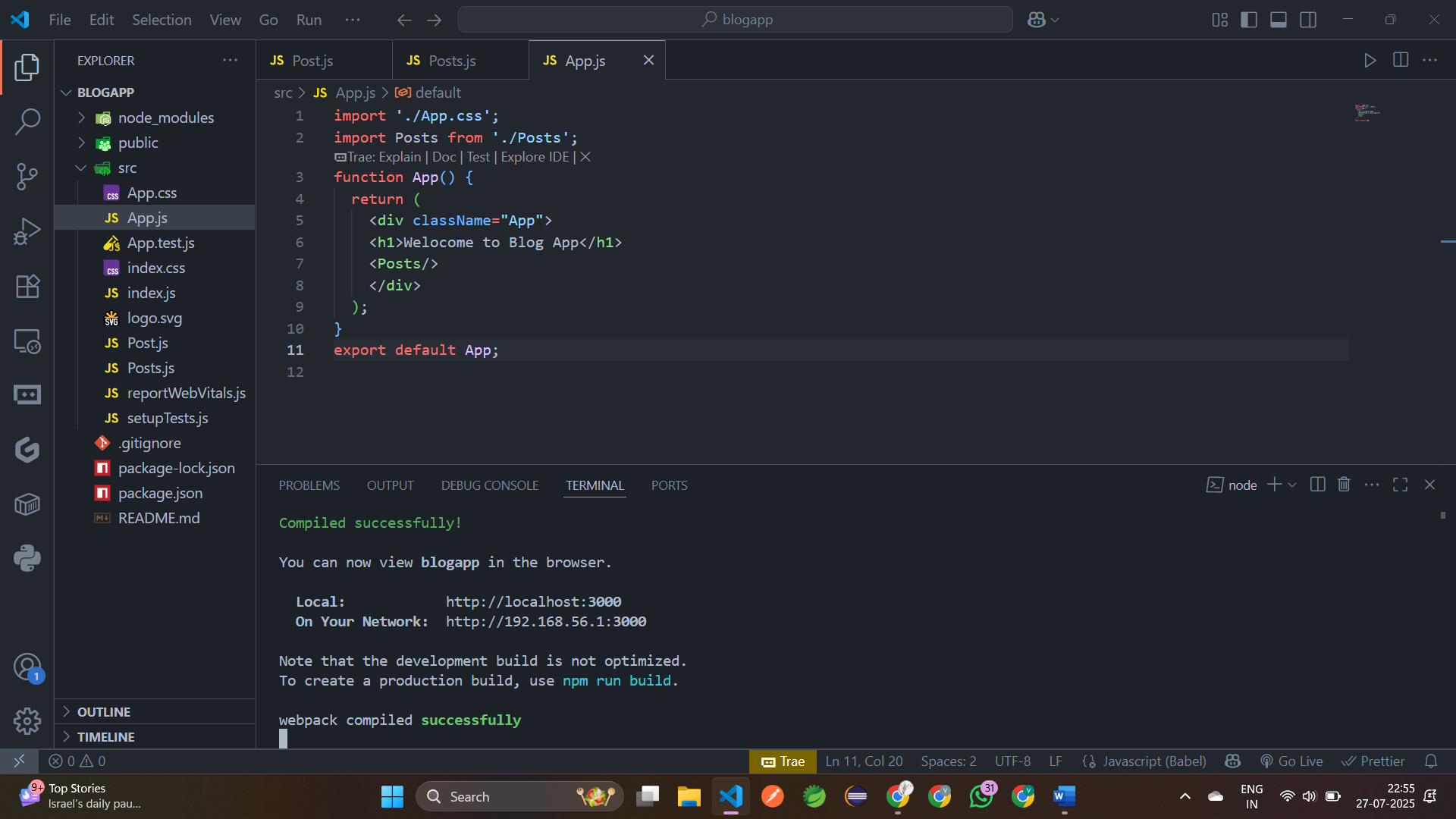
export default Posts;

step 12: Adding the Posts component to App component.

import './App.css';  
 import Posts from './Posts';  
 function App() {  
   return (  
     <div className="App">  
     <h1>Welocome to Blog App</h1>  
     <Posts/>  
     </div>  );  
 }export default App;

step 13: Build and Run the application using **npm start** command.

URL: <http://localhost:3000>



**Output:**



