**React Blog Post App**

**REACT**

Vaishnavi

26/7/2025

This React application demonstrates how to fetch and display data from an external API using class-based components. It uses fetch() to load blog posts from the JSONPlaceholder API and renders each post using reusable components.

**Objective:**

* To fetch data from a public API using fetch().
* To manage component state and lifecycle using class-based components.
* To display blog posts dynamically using reusable components (Post).
* To implement basic error handling with componentDidCatch.

**Implementation:**

**Step 1: Create a React App**



**Step 2: Create Components**

**Post.js**

import React from 'react';

class Post extends React.Component {

render() {

const { title, body } = this.props;

return (

<div style={{ marginBottom: '20px' }}>

<h3>{title}</h3>

<p>{body}</p>

</div>

);

}

}

export default Post;

**Posts.js**

import React from 'react';

import Post from './Post';

class Posts extends React.Component {

constructor(props) {

super(props);

this.state = {

posts: [],

error: null

};

}

componentDidMount() {

this.loadPosts();

}

loadPosts() {

fetch('https://jsonplaceholder.typicode.com/posts')

.then(response => {

if (!response.ok) throw new Error('Failed to fetch posts');

return response.json();

})

.then(data => this.setState({ posts: data }))

.catch(error => this.setState({ error }));

}

componentDidCatch(error, info) {

alert('An error occurred: ' + error.message);

}

render() {

const { posts, error } = this.state;

if (error) {

return <h2>Error loading posts.</h2>;

}

return (

<div style={{ padding: '20px' }}>

<h2>Blog Posts</h2>

{posts.slice(0, 10).map(post => (

<Post key={post.id} title={post.title} body={post.body} />

))}

</div>

);

}

}

export default Posts;

**Step 3: Update App.js**

import './App.css';

import Posts from './Posts';

function App() {

return (

<div className="App">

<Posts />

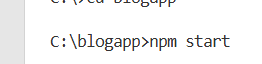
</div>

);

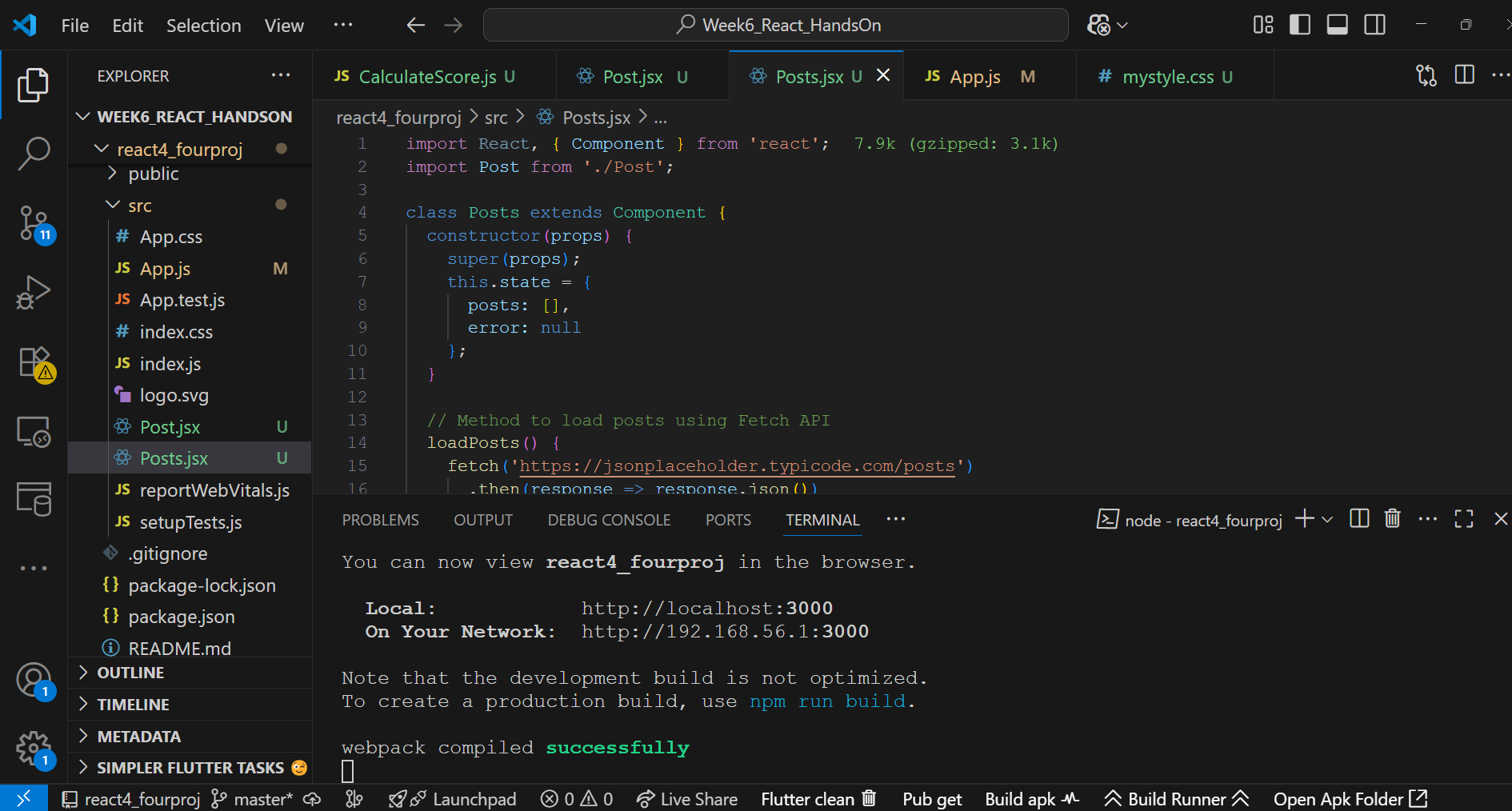
}

export default App;

**Step 4: Run the App**



You will see the first 10 blog posts loaded dynamically from the API and rendered to the screen.



**Output:**

