**4. ReactJS-HOL**

**Post.js**

import React from 'react';

class Post extends React.Component {

render() {

return (

<div style={{ border: '1px solid black', margin: '10px', padding: '10px' }}>

<h3>{this.props.title}</h3>

<p>{this.props.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: [],

hasError: false

};

}

loadPosts = () => {

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

.then((response) => response.json())

.then((data) => {

this.setState({ posts: data });

})

.catch((error) => {

throw new Error('Failed to fetch posts');

});

};

componentDidMount() {

this.loadPosts();

}

componentDidCatch(error, info) {

alert('An error occurred while rendering the component.');

console.log('Error:', error);

console.log('Info:', info);

this.setState({ hasError: true });

}

render() {

if (this.state.hasError) {

return <h2>Something went wrong.</h2>;

}

return (

<div>

<h1>Blog Posts</h1>

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

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

))}

</div>

);

}

}

export default Posts;

**App.js**

import React from 'react';

import Posts from './Posts';

function App() {

return (

<div>

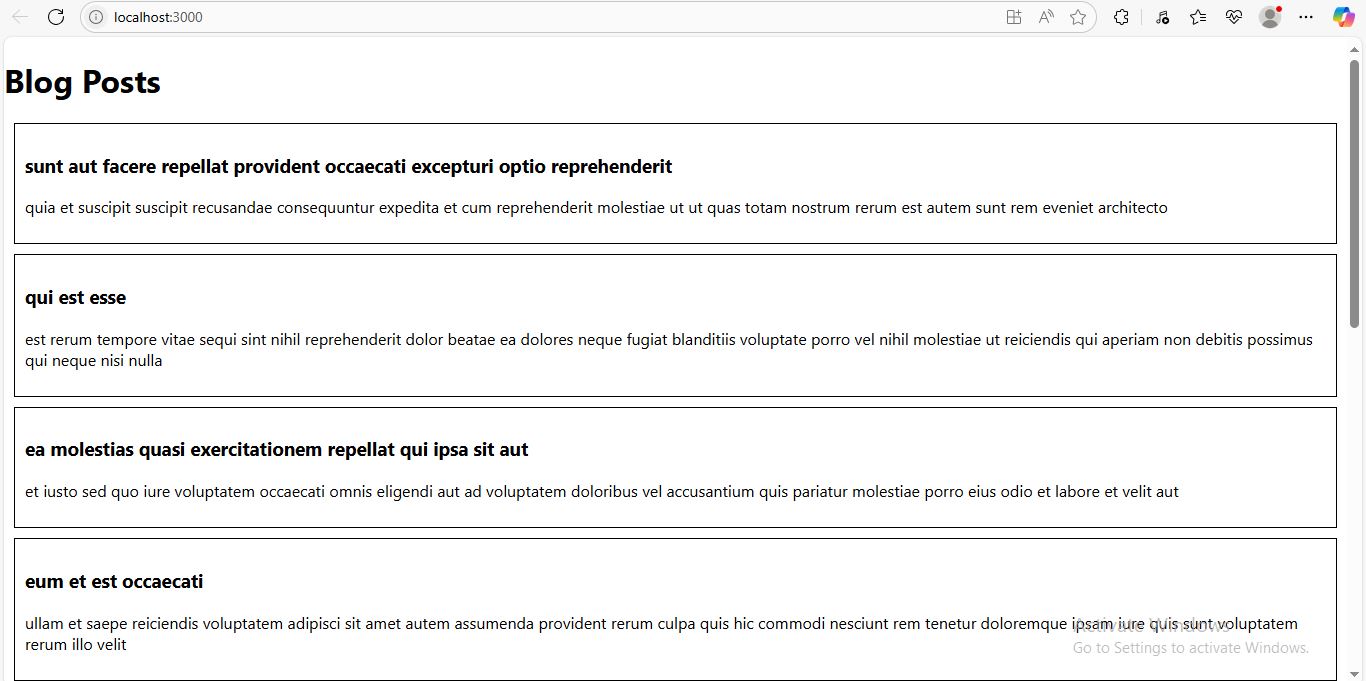
<Posts />

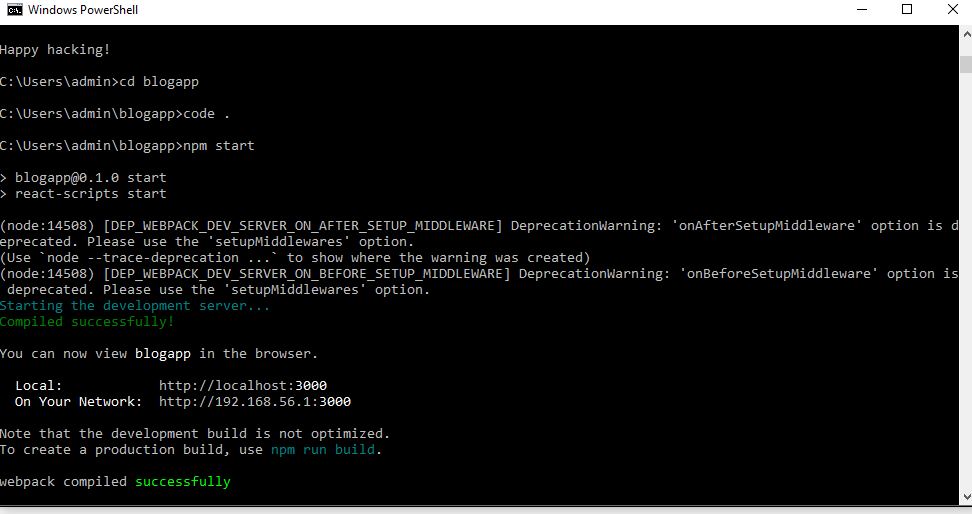
</div>

);

}

export default App;

****

****

**1. Explain the need and benefits of component life cycle**

React components go through a lifecycle from creation to destruction. The **component lifecycle** allows you to:

* Control **when and how** components mount, update, and unmount
* Fetch data after the component is inserted into the DOM
* Handle errors gracefully
* Perform clean-up tasks

**Benefits:**

* Improved resource management (e.g., API calls)
* Better debugging and error handling
* Controlled rendering and re-rendering

**2. Identify various life cycle hook methods**

**Common Class Component Lifecycle Methods:**

| **Phase** | **Lifecycle Methods** |
| --- | --- |
| Mounting | constructor, componentDidMount |
| Updating | shouldComponentUpdate, componentDidUpdate |
| Unmounting | componentWillUnmount |
| Error Handling | componentDidCatch, getDerivedStateFromError |

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

1. constructor()
2. render()
3. componentDidMount()

On update:

1. shouldComponentUpdate()
2. render()
3. componentDidUpdate()

On error:

1. getDerivedStateFromError()
2. componentDidCatch()