**4. Creating blogapp using React**

**Objective:**

To build a React application using class component that fetches blog post data from a public API and displays it, while handling lifecycle events and errors.

**Software Requirements**:

* Node.js and npm
* Visual Studio Code
* Web browser

**Steps to create React Application:**

Step1: Create React App.

Open the command prompt and execute the below command

|  |
| --- |
| npx create-react-app blogapp |

Step2: Open the project directory in VSCode.

i.e.,

|  |
| --- |
| cd blogapp  code . |

Step3: Create a class that defines the structure of post object.

i.e.,

Post.js

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

Step4: create a class component Posts along with lifecycle events and errors.

i.e.,  
Posts.js

|  |
| --- |
| import React from 'react';  import Post from './Post';  class Posts extends React.Component {  constructor(props) {  super(props);  this.state = {  posts: [],  hasError: false,  errorMessage: ''  };  } |

Now, please modify the class component by adding **theloadPosts(**) method

|  |
| --- |
| loadPosts() {  fetch('https://jsonplaceholder.typicode.com/posts')  .then(response => response.json())  .then(data => {  const postObjects = data.map(item => new Post(item.id, item.title, item.body));  this.setState({ posts: postObjects });  })  .catch(error => {  this.setState({ hasError: true, errorMessage: error.message });  });  } |

Then, include the hook i.e., **componentDidMount()** inside the Posts.js

|  |
| --- |
| componentDidMount() {  this.loadPosts();  } |

Later, place the **render()** method as the content needs to be displayed i.e.,

|  |
| --- |
| render() {  const { posts, hasError, errorMessage } = this.state;  if (hasError) {  return <div>Error: {errorMessage}</div>;  }  return (  <div>  <h1>Blog Posts</h1>  {posts.map (post => (  <div key={post.id}>  <h3>{post.title}</h3>  <p>{post.body}</p>  </div>  ))}  </div>  );  } |

Then, add the hook **componentDidCatch()** to get notified if an error occurs.

|  |
| --- |
| componentDidCatch(error, info) {  alert("An error occurred in Posts component!");  console.error("Caught error:", error, info);  }  }  export default Posts; |

Step5: Modify the App.js

Inside the App.js import and use the Posts component in App.js

|  |
| --- |
| import React from 'react';  import Posts from './Posts';  function App() {  return (  <div className="App">  <Posts />  </div>  );  }  export default App; |

Step6: Run the Application.

Open the terminal in vscode and execute the below command. So that the development server starts automatically.

**Expected outcome** is in below format with multiple posts.

