# Exercise 4: ReactJS Lifecycle Methods - blogapp

This exercise demonstrates React component lifecycle methods using a blog app that fetches posts and handles errors.

## Step 1: Create React App

npx create-react-app blogapp  
cd blogapp  
code .

## Step 2: Create Post.js

// src/Post.js  
import React from 'react';  
  
class Post extends React.Component {  
 render() {  
 const { title, body } = this.props;  
 return (  
 <div>  
 <h2>{title}</h2>  
 <p>{body}</p>  
 </div>  
 );  
 }  
}  
  
export default Post;

## Step 3: Create Posts.js

// src/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.slice(0, 5) }))  
 .catch(error => {  
 console.error('Error fetching posts:', error);  
 this.setState({ hasError: true });  
 });  
 };  
  
 componentDidMount() {  
 this.loadPosts();  
 }  
  
 componentDidCatch(error, info) {  
 alert("An error occurred: " + error.toString());  
 console.error("Error boundary caught:", error, info);  
 }  
  
 render() {  
 if (this.state.hasError) {  
 return <h2>Something went wrong while loading posts.</h2>;  
 }  
  
 return (  
 <div>  
 {this.state.posts.map(post => (  
 <Post key={post.id} title={post.title} body={post.body} />  
 ))}  
 </div>  
 );  
 }  
}  
  
export default Posts;

## Step 4: Modify App.js

// src/App.js  
import React from 'react';  
import Posts from './Posts';  
  
function App() {  
 return (  
 <div className="App">  
 <h1>Blog Posts</h1>  
 <Posts />  
 </div>  
 );  
}  
  
export default App;

Run the application using `npm start`. The posts will be displayed using componentDidMount and errors handled using componentDidCatch.

# Exercise 5: Styling React Components with CSS Modules

This exercise focuses on styling React components using CSS Modules and inline styles.

## Step 1: Create CSS Module

/\* src/CohortDetails.module.css \*/  
.box {  
 width: 300px;  
 display: inline-block;  
 margin: 10px;  
 padding: 10px 20px;  
 border: 1px solid black;  
 border-radius: 10px;  
}  
  
dt {  
 font-weight: 500;  
}

## Step 2: Create CohortDetails.js

// src/CohortDetails.js  
import React from 'react';  
import styles from './CohortDetails.module.css';  
  
function CohortDetails({ name, status, startDate, endDate }) {  
 const titleStyle = {  
 color: status === 'ongoing' ? 'green' : 'blue'  
 };  
  
 return (  
 <div className={styles.box}>  
 <h3 style={titleStyle}>{name}</h3>  
 <dl>  
 <dt>Status:</dt>  
 <dd>{status}</dd>  
 <dt>Start Date:</dt>  
 <dd>{startDate}</dd>  
 <dt>End Date:</dt>  
 <dd>{endDate}</dd>  
 </dl>  
 </div>  
 );  
}  
  
export default CohortDetails;

## Step 3: Modify App.js

// src/App.js  
import React from 'react';  
import CohortDetails from './CohortDetails';  
  
function App() {  
 return (  
 <div>  
 <CohortDetails name="React Bootcamp" status="ongoing" startDate="01 July 2025" endDate="15 August 2025" />  
 <CohortDetails name="Node.js Track" status="completed" startDate="01 May 2025" endDate="15 June 2025" />  
 </div>  
 );  
}  
  
export default App;