**ReactJS-HOL- Exercise 4**

**1. Need and Benefits of Component Lifecycle**

Why lifecycle is needed:  
React components go through different phases: mounting, updating, and unmounting. Lifecycle methods (or hooks) help us:

* Perform actions at specific points during the component's life.
* Initialize data, like fetching from APIs when component loads.
* Respond to changes in props or state.
* Clean up before a component is removed (e.g., clear timers or unsubscribe listeners).

Benefits:

* Better control over rendering and performance.
* Ensures side effects (API calls, DOM updates) happen at the correct time.
* Helps in debugging and managing component behavior in large apps.

**2. Lifecycle Hook Methods in Class Components**

React provides several lifecycle hook methods grouped into phases:

A. Mounting (when the component is created and inserted into DOM):

* constructor()
* static getDerivedStateFromProps()
* render()
* componentDidMount()

B. Updating (when state or props change):

* static getDerivedStateFromProps()
* shouldComponentUpdate()
* render()
* getSnapshotBeforeUpdate()
* componentDidUpdate()

C. Unmounting (when component is removed from DOM):

* componentWillUnmount()

D. Error Handling:

* componentDidCatch()

**3. Sequence of Steps in Rendering a Component (Mounting Phase)**

Here is the typical order of method calls when a component is being rendered for the first time:

1. constructor() →  
   Used to initialize state and bind methods.
2. getDerivedStateFromProps() →  
   Syncs state from props if needed (rarely used).
3. render() →  
   Returns JSX to display on the screen.
4. componentDidMount() →  
   Called once after the component is rendered.  
   → Good place for API calls and DOM interactions.

**HANDSON:**

1. **Create a new react application using *create-react-app* tool with the name as “blogapp”**
2. **Open the application using VS Code**
3. **Create a new file named as Post.js in src folder with following code:**

import React from "react";

class Post extends React.Component {

  render() {

    const { title, body } = this.props;

    return (

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

        <h3>{title}</h3>

        <p>{body}</p>

      </div>

    );

  }

}

export default Post;

1. **Create a new class based component named as Posts inside Posts.js file:**

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((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 });

      });

  };

  componentDidMount() {

    this.loadPosts();

  }

  componentDidCatch(error, info) {

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

  }

  render() {

    const { posts } = this.state;

    return (

      <div>

        <h2>Blog Posts</h2>

        {posts.map((post) => (

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

        ))}

      </div>

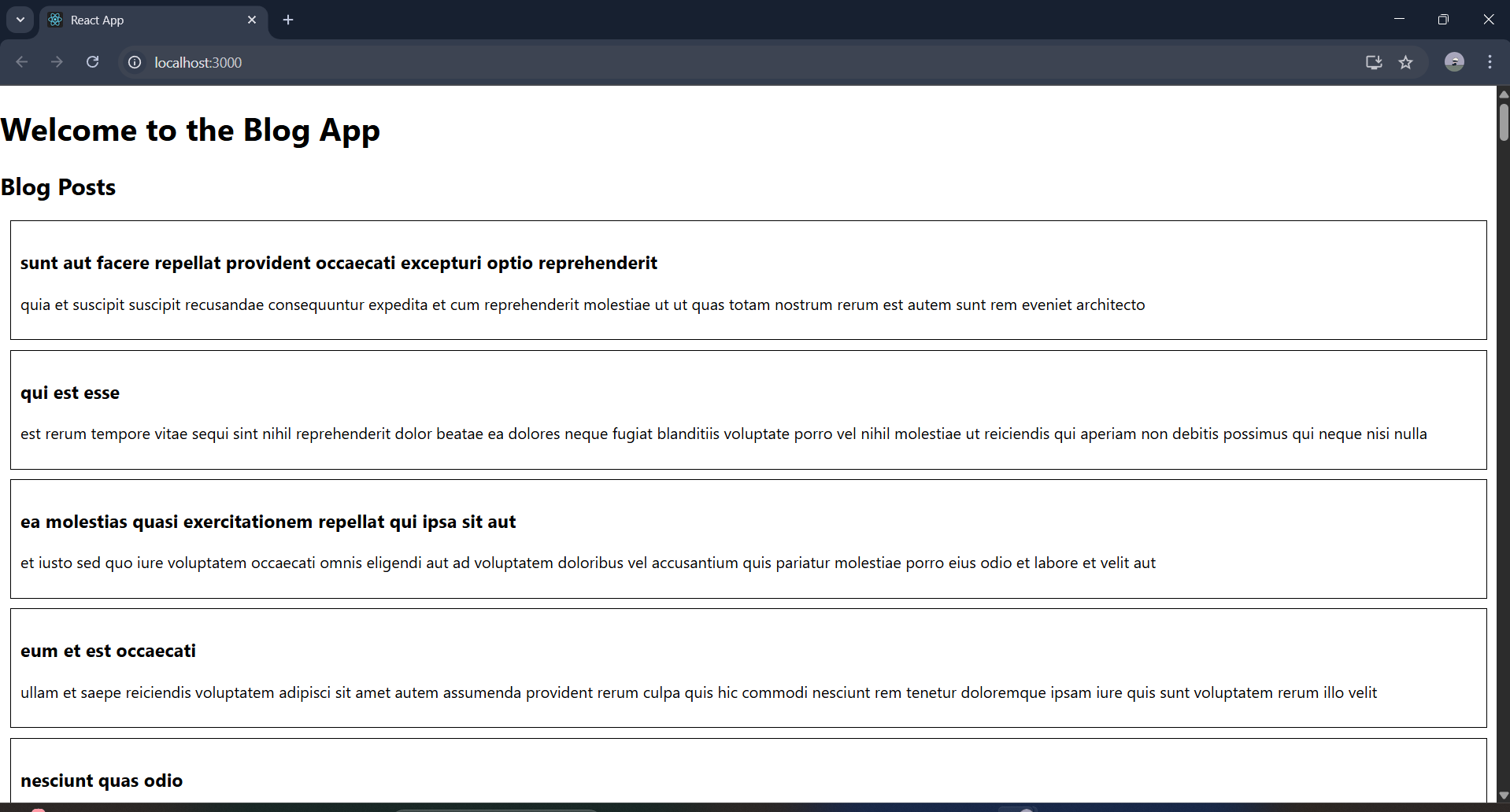
    );

  }

}

export default Posts;

1. **Build and Run the application using *npm start* command**.



**And like this there will be a long document.**