Create an application in ReactJS to implement component life cycle

LifecycleClassComponent.js

```
import React, { Component } from 'react';

class LifecycleClassComponent extends Component {
  constructor(props) {
    super(props);
    console.log('Constructor: Component is being created');
    this.state = {
      message: 'Hello, React Lifecycle!',
    };
  }

  static getDerivedStateFromProps(nextProps, nextState) {
    console.log('getDerivedStateFromProps: Called before every render');
    return null; // Returning null means no changes in state from props
  }

  shouldComponentUpdate(nextProps, nextState) {
    console.log('shouldComponentUpdate: Deciding if re-render is necessary');
    return true; // Return true to allow update, false to prevent re-render
  }

  getSnapshotBeforeUpdate(prevProps, prevState) {
    console.log('getSnapshotBeforeUpdate: Capture some data before DOM update');
    return null; // No snapshot data
  }

  componentDidUpdate(prevProps, prevState, snapshot) {
    console.log('componentDidUpdate: Component updated successfully');
```

```
  }

  componentDidMount() {
    console.log('componentDidMount: Component mounted (initial render completed)');
    // Perform actions after component is rendered, like fetching data
  }

  componentWillUnmount() {
    console.log('componentWillUnmount: Component is being removed');
    // Cleanup tasks like canceling API requests, timers
  }

  render() {
    console.log('render: Component rendering');
    return (
      <div>
        <h1>{this.state.message}</h1>
        <button onClick={() => this.setState({ message: 'Updated Message!' })}>
          Update Message
        </button>
      </div>
    );
  }
}

export default LifecycleClassComponent;
```

LifecycleFunctionComponent.js

```
import React, { useState, useEffect } from 'react';

const LifecycleFunctionComponent = () => {
```

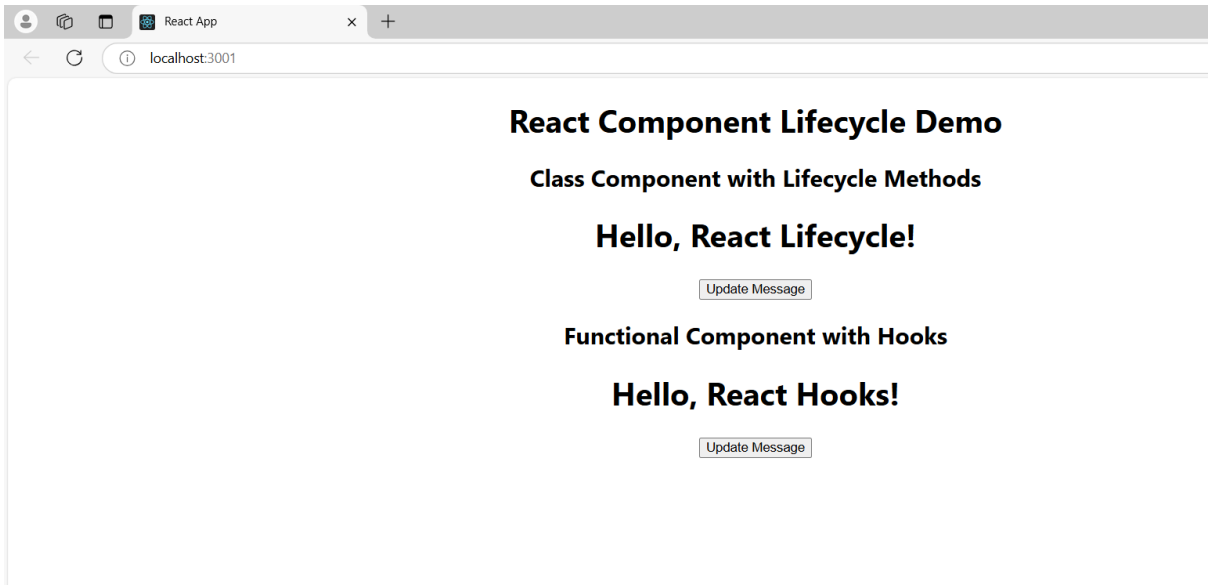
```
  const [message, setMessage] = useState('Hello, React Hooks!');

  // Equivalent to componentDidMount & componentDidUpdate (runs on every render)
  useEffect(() => {
    console.log('useEffect: Component mounted or updated');
    return () => {
      // This return function is equivalent to componentWillUnmount
      console.log('useEffect Cleanup: Component will unmount');
    };
  }, [message]); // The dependency array ensures it runs when "message" changes

  const updateMessage = () => {
    setMessage('Updated Message using Hooks!');
  };

  return (
    <div>
      <h1>{message}</h1>
      <button onClick={updateMessage}>Update Message</button>
    </div>
  );
};

export default LifecycleFunctionComponent;
```

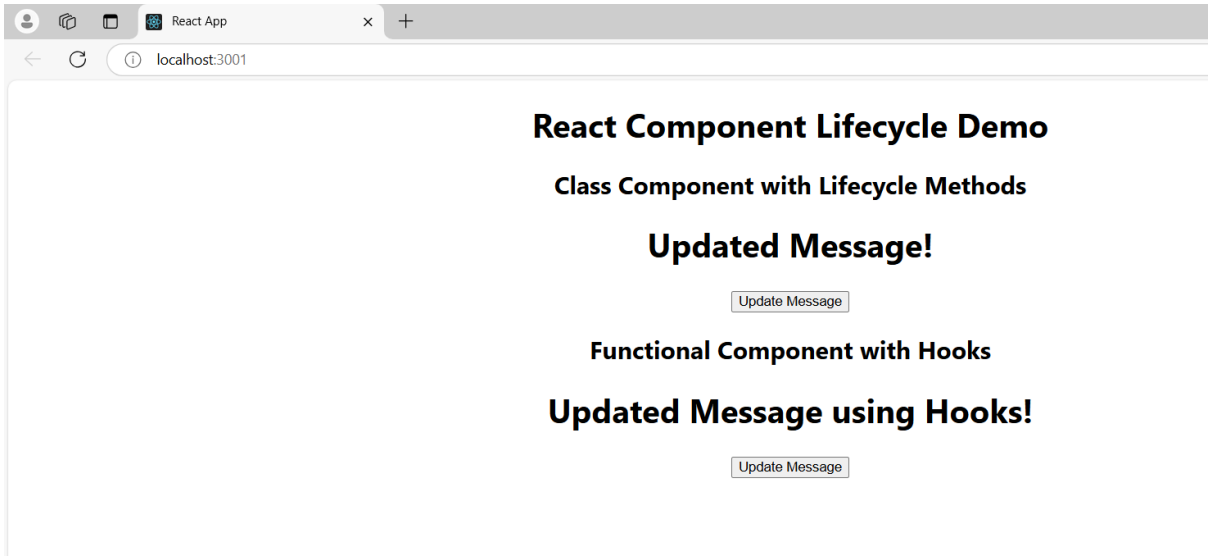
App.js

```
import React from 'react';
import './App.css';
import LifecycleClassComponent from './LifecycleClassComponent';
import LifecycleFunctionComponent from './LifecycleFunctionComponent';

function App() {
```

```
  return (
    <div className="App">
      <h1>React Component Lifecycle Demo</h1>

      <h2>Class Component with Lifecycle Methods</h2>
      <LifecycleClassComponent />

      <h2>Functional Component with Hooks</h2>
      <LifecycleFunctionComponent />
    </div>
  );
}

export default App;
```

output




React App
localhost:3001
React Component Lifecycle Demo
Class Component with Lifecycle Methods
Updated Message!
Update Message
Functional Component with Hooks
Updated Message using Hooks!
Update Message