**WEEK – 7(Additional HandsOn)**

**React**

**19. ReactJS-HOL**

**Objectives**

**Understanding need for isolation in testing**

**Isolation in testing** means testing a **unit/component independently**, without relying on:

* External systems (APIs, DBs)
* Other components
* Network responses
* User input

**Why isolation is important:**

| **Benefit** | **Explanation** |
| --- | --- |
| Reliable tests | Failures only occur due to issues in the component |
| Fast execution | No need for API or server connections |
| Easy debugging | You know exactly what failed |
| Encourages best design | Makes components more modular and testable |

**Understanding the concept of mocking**

**Mocking** means creating **fake functions, data, or modules** to simulate the behavior of real ones.

This helps you:

* Test components **in isolation**
* Avoid real API calls
* Simulate various responses (success, failure, delay)

**Types of Mocking:**

| **Type** | **Example** |
| --- | --- |
| Function mocking | jest.fn() to mock any function |
| Module mocking | jest.mock('axios') |
| API mocking | Fake server responses |

**Using Jest for unit testing and mocking**

**Jest** is the official testing framework used with React. It comes with:

* Assertions (expect)
* Mocks (jest.fn)
* Snapshot testing
* Built-in test runner

1. **Install Jest (if not already):**

npm install --save-dev jest

1. **Create a simple test file**

**Component:**

// Welcome.js

export default function Welcome({ name }) {

return <h1>Hello, {name}</h1>;

}

**Test:**

// Welcome.test.js

import { render, screen } from '@testing-library/react';

import Welcome from './Welcome';

test('renders the Welcome component', () => {

render(<Welcome name="Priya" />);

expect(screen.getByText('Hello, Priya')).toBeInTheDocument();

});

1. **Run the tests**

npm test

1. **Using Mocks in Jest**

**Example: Mock an API call**

import axios from 'axios';

jest.mock('axios');

test('fetches users', async () => {

axios.get.mockResolvedValue({ data: [{ name: 'Anu' }] });

// Your fetchUsers() function logic

const result = await fetchUsers();

expect(result[0].name).toBe('Anu');

});

**CODE:**

**src/RepoList.js:**

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

function RepoList() {

  // Set the username to TechieSyed for your output

  const username = "TechieSyed";

  const [repos, setRepos] = useState([]);

  const [loading, setLoading] = useState(true);

  const [error, setError] = useState(null);

  useEffect(() => {

    fetch(`https://api.github.com/users/${username}/repos`)

      .then(response => {

        if (!response.ok) throw new Error("Network error");

        return response.json();

      })

      .then(data => {

        setRepos(data);

        setLoading(false);

      })

      .catch(err => {

        setError(err.message);

        setLoading(false);

      });

  }, []);

  if (loading) return <div style={{textAlign: 'center'}}>Loading...</div>;

  if (error) return <div style={{color: 'red',textAlign: 'center'}}>{error}</div>;

  return (

    <div style={{ textAlign: 'center', marginTop: '40px' }}>

      <h1 style={{fontWeight:'bold', fontSize:'2rem', marginBottom:'24px'}}>

        Git repositories of User - {username}

      </h1>

      <div>

        {repos.map(repo => (

          <div key={repo.id} style={{margin: '5px 0'}}>{repo.name}</div>

        ))}

      </div>

    </div>

  );

}

export default RepoList;

**src/App.js:**

import React from 'react';

import RepoList from './RepoList';

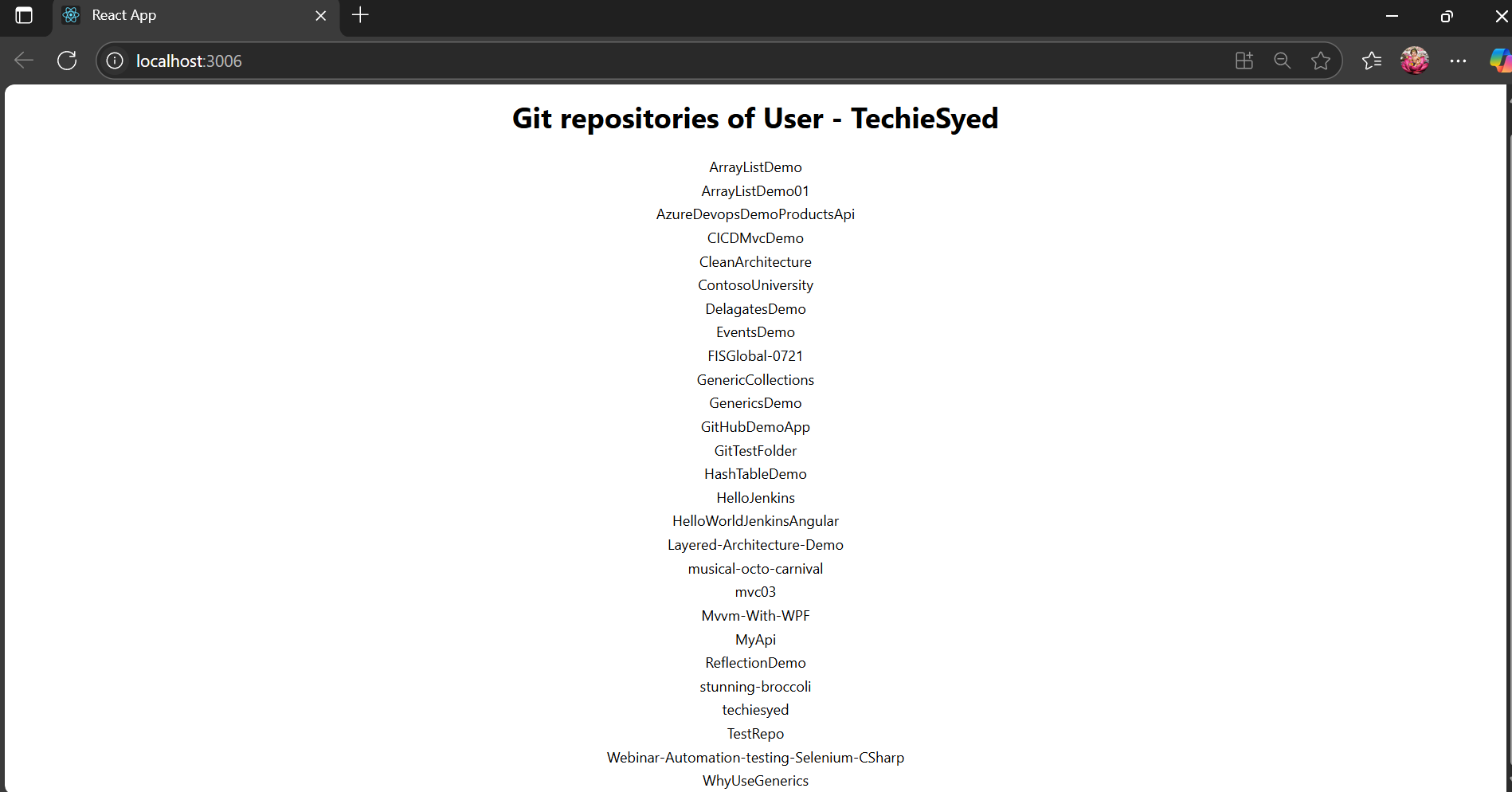
function App() {

  return <RepoList />;

}

export default App;

**OUTPUT:**

****