|  |  |  |  |
| --- | --- | --- | --- |
| **Course Name:** | **Web Programming Laboratory 116U40L501** | **Semester:** | **V** |
| **Date of Performance:** | **7 / 10 / 2024** | **Batch No:** | **B - 1** |
| **Faculty Name:** | **Prof. Madhura Pednekar** | **Roll No:** | **16014022050** |
| **Faculty Sign & Date:** |  | **Grade/Marks:** | **\_\_\_ / 25** |

**Experiment No: 5**

**Title: Understanding Reactjs**

|  |
| --- |
| **Aim and Objective of the Experiment:** |
| Implementation of React Fundamentals and React Hooks. |

|  |
| --- |
| **COs to be achieved:** |
| Create interactive web content with JavaScript, including form validation and event handling. |

|  |
| --- |
| **Problem Statement:** |
| a) Demonstrate the   * React Fundamentals * Function Component * Styling/ Bootstrap * React JSX * Expressions in JSX * React Props * React state * React Component Lifecycle * React Events * Event Binding   b) Implementation of React Hooks with the following points   * useState * useEffect * useContext * useRef * useCallback * useReducer * UseMemo |

|  |
| --- |
| **Code:** |
| 1. **server.js**   const express = require('express');  const cors = require('cors'); // Import cors  const fs = require('fs');  const path = require('path');  const app = express(); // Initialize express app  const port = 5000; // Make sure this matches the port used in your fetch request  // Use CORS middleware  app.use(cors());  // Middleware to parse incoming JSON requests  app.use(express.json());  app.post('/submit', (req, res) => {    const formData = req.body;    // Format the data to be written to the file    const dataToWrite = `Name: ${formData.name}, Email: ${formData.email}, Password: ${formData.password}, Gender: ${formData.gender}, Age: ${formData.age}, College: ${formData.college}, Branch: ${formData.branch}, Year: ${formData.year}\n`;    // Define the path to formData.txt    const filePath = path.join(\_\_dirname, 'formData.txt');    try {      // Write the form data to the file synchronously      fs.writeFileSync(filePath, dataToWrite, { flag: 'a' }); // 'a' flag to append data      res.json({ message: 'Form data saved successfully!' });    } catch (error) {      console.error('Error writing to file:', error);      res.status(500).json({ message: 'Failed to save form data' });    }  });  app.listen(port, () => {    console.log(`Server is running on http://localhost:${port}`);  });   1. **app.js**   import './App.css';  import { useState } from 'react';  function App() {    const [formData, setFormData] = useState({      name: '',      email: '',      password: '',      gender: '',      age: '',      college: '',      branch: '',      year: ''    });    const handleChange = (e) => {      setFormData({        ...formData,        [e.target.name]: e.target.value      });    };    const handleSubmit = (e) => {      e.preventDefault();      fetch('http://localhost:5000/submit', { // Ensure the port matches your Express server        method: 'POST',        headers: { 'Content-Type': 'application/json' },        body: JSON.stringify(formData)      })      .then(response => response.json())      .then(data => {        console.log('Success:', data);        alert(data.message); // Display success message      })      .catch((error) => {        console.error('Error:', error);        alert('Failed to submit form');      });    };    return (      <div className="App">        <h1>Signup Form</h1>        <form onSubmit={handleSubmit}>          <div className="form-group">            <label htmlFor="name">Name:</label>            <input type="text" id="name" name="name" value={formData.name} onChange={handleChange} placeholder="Enter your name" required />          </div>            <div className="form-group">            <label htmlFor="email">Email:</label>            <input type="email" id="email" name="email" value={formData.email} onChange={handleChange} placeholder="Enter your email" required />          </div>            <div className="form-group">            <label htmlFor="password">Password:</label>            <input type="password" id="password" name="password" value={formData.password} onChange={handleChange} placeholder="Enter your password" required />          </div>          <div className="form-group">            <label htmlFor="gender">Gender:</label>            <select id="gender" name="gender" value={formData.gender} onChange={handleChange} required>              <option value="">Select gender</option>              <option value="male">Male</option>              <option value="female">Female</option>              <option value="other">Other</option>            </select>          </div>          <div className="form-group">            <label htmlFor="age">Age:</label>            <input type="number" id="age" name="age" value={formData.age} onChange={handleChange} placeholder="Enter your age" required />          </div>          <div className="form-group">            <label htmlFor="college">College:</label>            <input type="text" id="college" name="college" value={formData.college} onChange={handleChange} placeholder="Enter your college" required />          </div>          <div className="form-group">            <label htmlFor="branch">Branch:</label>            <input type="text" id="branch" name="branch" value={formData.branch} onChange={handleChange} placeholder="Enter your branch" required />          </div>          <div className="form-group">            <label htmlFor="year">Year:</label>            <select id="year" name="year" value={formData.year} onChange={handleChange} required>              <option value="">Select year</option>              <option value="first">First</option>              <option value="second">Second</option>              <option value="third">Third</option>              <option value="fourth">Fourth</option>            </select>          </div>          <button type="submit">Sign Up</button>        </form>      </div>    );  }  export default App;   1. **index.html**   <!DOCTYPE html>  <html lang="en">    <head>      <meta charset="utf-8" />      <link rel="icon" href="%PUBLIC\_URL%/favicon.ico" />      <meta name="viewport" content="width=device-width, initial-scale=1" />      <meta name="theme-color" content="#000000" />      <meta        name="description"        content="Web site created using create-react-app"      />      <link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />      <link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />      <title>React App</title>    </head>    <body>      <noscript>You need to enable JavaScript to run this app.</noscript>      <div id="root"></div>    </body>  </html> |

|  |
| --- |
| **Stepwise-Procedure / Algorithm:** |
| * Setup React Project: Create React app using npx create-react-app and install Bootstrap. * Import Bootstrap: Add Bootstrap import in index.js. * Create Functional Component: Create RegistrationForm.js for the form. * Set Up State: Use useState for managing form inputs like name, email, age, course. * Use Ref for Input Focus: Use useRef to focus on the name input. * Handle Form Submission: Implement form submission with validation using handleSubmit. * Create JSX Form Structure: Use JSX and Bootstrap for form layout and styling. * Add Event Handlers: Create onChange handlers for input fields. * Implement Form Validation: Validate form fields before submission. * Render Component in App: Import and render RegistrationForm in App.js. |

|  |
| --- |
| **Output** |
| node server.js    npm start |

|  |
| --- |
| **Post Lab Subjective/Objective type Questions:** |
| 1. **Explain the significance of React hooks like useState and useEffect in functional components. How do they improve component design compared to class-based components?**   React hooks like useState and useEffect significantly enhance the functionality of functional components by allowing developers to add state and lifecycle features without needing class-based components.   * **useState**: This hook allows functional components to manage state variables, which was previously only possible in class components. It provides a cleaner and more concise way to handle stateful logic. In the case of our student registration form, useState is used to manage the form inputs like name, email, and age. Every input field is tied to its corresponding state, and any change triggers an update in the UI. * **useEffect**: This hook mimics the lifecycle methods of class components, such as componentDidMount, componentDidUpdate, and componentWillUnmount. In the registration form, useEffect is used to focus the input field when the component mounts, offering a smooth user experience. It can also be used for tasks like data fetching or cleanup activities.   **Improvement over Class Components:** Hooks simplify code and improve readability. Previously, state and lifecycle logic in class components often resulted in complex and bloated code. Hooks enable developers to logically group related state and effects, making the code modular, reusable, and easier to maintain. |

|  |
| --- |
| **Conclusion:** |
| This approach demonstrates how to use React fundamentals and hooks to build an interactive student registration form. It includes state management, event handling, and basic validation, allowing for a clear understanding of React's capabilities. |

|  |
| --- |
| **Signature of faculty in-charge with Date:** |