

**Shram Sadhana Bombay Trust Sanchlit
Arts, Commerce and Science College, Bambhori, Jalgaon
Bachelor of Computer Application (B.C.A.)**



Bachelor of Computer Applications

**LAB MANUAL ON
BCA 607 - Web Development Technology IV**

Name:

Class:

Sem:

Roll No:

Seat No:

Name of faculty: Ms. H. S. Jadhvani

Academic Year: 20 - 20

Shram Sadhana Bombay Trust Sanchlit
Arts, Commerce and Science College, Bambhori, Jalgaon
Bachelor of Computer Application (B.C.A.)
Practical: 01

DOP:

DOC:

Title: Write ReactJS code to use all the states in in the created Application.

Objective: Demonstrate the use of different types of states in React, including local state, reducer state, and global state using context. The application also showcases the use of hooks like useState, useReducer, useEffect, and useContext to manage and update state in a functional React

Theory: Here Steps to do this Practical

Step1: Install NodeJS (LTS FILE)

<https://nodejs.org/en>

Step2: Install Visual Studio Code

<https://code.visualstudio.com/download>

Step3: After Installing the NodeJS and VS code

(Setup the Extensions of NodeJS and VS code)

Step4: Open VS code and install the required Extension

- i. npm
- ii. prettier
- iii. ESLint
- iv. Bracket Pair Colorize
- v. ReactJs Snippets
- vi. Live Server

Step5: Open Terminal in VS code to check the Versions of NodeJS and npm packages

- i. node -v (To Check the NodeJS Version)
- ii. npm -v (To Check the NPM Version)
- iii. npx -v (To Check the NodeJS Package Execute Version)

Step6: After Checking the versions of NodeJS packages we can create a ReactJS app

npx create-react-app 1prac

This command is used to create a ReactJS app

Step7: After creating a ReactJS app, the **cd** command is used in the terminal to navigate into the app's directory. Alternatively, (**code .**)

Step8: After redirecting, The ReactJS app's editor will open, where the right side will show a folder named **1prac**. Inside this folder, there will be a subfolder named **src**, and within **src**, there will be a file named **App.js** where you will write the code.

Step9: In the App.js file, need to install some packages, such as:

1. npm install
2. npm install @babel/plugin-private-property-in-object

These commands will install the necessary dependencies for React application.

Step10: Code to Write in **App.js** File

```
import React, { useState, useEffect, useReducer, createContext, useContext } from 'react';
const GlobalContext = createContext();
const reducer = (state, action) => action.type === 'increment' ? { count: state.count + 1 } : action.type ===
'decrement' ? { count: state.count - 1 } : state;

function App() {
  const [name, setName] = useState("John"), [age, setAge] = useState(25), [state, dispatch] =
  useReducer(reducer, { count: 0 }), [globalState, setGlobalState] = useState({ theme: "light" });

  useEffect(() => console.log('Component updated'), [name, age]);

  const toggleTheme = () => setGlobalState(prev => ({ theme: prev.theme === "light" ? "dark" : "light"
  }));

  return (
    <GlobalContext.Provider value={{ globalState, toggleTheme }}>
      <div style={{ backgroundColor: globalState.theme === "light" ? "white" : "darkgray", color:
globalState.theme === "light" ? "black" : "white" }}>
        <h1>React States Example</h1>
        <h2>Local State</h2>
        <p>{name} {age}</p>
        <button onClick={() => setName(name === "John" ? "Jane" : "John")}>Toggle Name</button>
        <button onClick={() => setAge(age === 25 ? 30 : 25)}>Toggle Age</button>

        <h2>Reducer State</h2>
        <p>{state.count}</p>
        <button onClick={() => dispatch({ type: 'increment' })}>Increment</button>
        <button onClick={() => dispatch({ type: 'decrement' })}>Decrement</button>

        <h2>Global State</h2>
        <p>{globalState.theme}</p>
        <button onClick={toggleTheme}>Toggle Theme</button>

        <ChildComponent name={name} age={age} />
        <ChildComponentWithContext />

      </div>
    </GlobalContext.Provider>
  );
}
```

```

</GlobalContext.Provider>
);
}

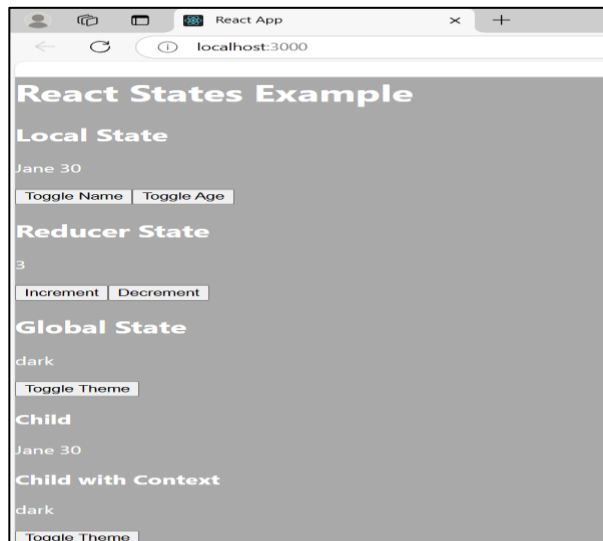
function ChildComponent({ name, age }) {
  return <div><h3>Child</h3><p>{name} {age}</p></div>;
}

function ChildComponentWithContext() {
  const { globalState, toggleTheme } = useContext(GlobalContext);
  return <div><h3>Child with Context</h3><p>{globalState.theme}</p><button
onClick={toggleTheme}>Toggle Theme</button></div>;
}
export default App;

```

Step11: Run this file (npm start)

Output:



Conclusion: The app showcases managing local, reducer, and global states using React hooks like useState, useReducer, and useContext, enabling dynamic and efficient state handling.

Submitted By:

Checked By:
Ms. H. S. Jadhvani

Sign:

Name:

Roll No:

Shram Sadhana Bombay Trust Sanchlit
Arts, Commerce and Science College, Bambhori, Jalgaon
Bachelor of Computer Application (B.C.A.)
Practical: 02

DOP:

DOC:

Title: Write ReactJS code for Client-side form validation.

Objective: To implement client-side form validation in ReactJS, ensuring that the name, email, and password fields meet specific requirements before form submission. Validations include:

- i. **Name:** Must not be empty.
- ii. **Email:** Must follow a valid email format.
- iii. **Password:** Must be at least 6 characters long.

Theory: Here Steps to do this Practical

Step1: Install NodeJS (LTS FILE)

<https://nodejs.org/en>

Step2: Install Visual Studio Code

<https://code.visualstudio.com/download>

Step3: After Installing the NodeJS and VS code
(Setup the Extensions of NodeJS and VS code)

Step4: Open VS code and install the required Extension

- i. npm
- ii. prettier
- iii. ESLint
- iv. Bracket Pair Colorize
- v. ReactJs Snippets
- vi. Live Server

Step5: Open Terminal in VS code to check the Versions of NodeJS and npm packages

- i. node -v (To Check the NodeJS Version)
- ii. npm -v (To Check the NPM Version)
- iii. npx -v (To Check the NodeJS Package Execute Version)

Step6: After Checking the versions of NodeJS packages we can create a ReactJS app

npx create-react-app 2prac

This command is used to create a ReactJS app

Step7: After creating a ReactJS app, the **cd** command is used in the terminal to navigate into the app's directory. Alternatively, (**code .**)

Step8: After redirecting, The ReactJS app's editor will open, where the right side will show a folder named **2prac**. Inside this folder, there will be a subfolder named **src**, and within **src**, there will be a file named **App.js** where you will write the code.

Step9: In the App.js file, need to install some packages, such as:

1. npm install
2. npm install @babel/plugin-private-property-in-object

These commands will install the necessary dependencies for React application.

Step10: Code to Write in **App.js** File

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

```
function App() {
  const [formValues, setFormValues] = useState({ name: "", email: "", password: "" });
  const [errors, setErrors] = useState({ name: "", email: "", password: "" });
  const [isSubmit, setIsSubmit] = useState(false);

  const handleInputChange = (e) => setFormValues({ ...formValues, [e.target.name]: e.target.value });

  const validate = () => {
    let tempErrors = {}, isValid = true;
    if (!formValues.name) { tempErrors.name = 'Name is required'; isValid = false; }
    if (!formValues.email || !/^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/.test(formValues.email)) {
      tempErrors.email = 'Email is invalid'; isValid = false;
    }
    if (!formValues.password || formValues.password.length < 6) { tempErrors.password = 'Password must be 4+ chars'; isValid = false; }
    setErrors(tempErrors);
    return isValid;
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    if (validate()) { setIsSubmit(true); console.log('Form submitted', formValues); }
  };

  return (
    <div>
      <h1>Form Validation</h1>

      <form onSubmit={handleSubmit}>

        <input type="text" name="name" value={formValues.name} onChange={handleInputChange}
placeholder="Name" />
        {errors.name && <span style={{ color: 'red' }}>{errors.name}</span>}


```

```

    <input type="email" name="email" value={formValues.email} onChange={handleInputChange}
placeholder="Email" />
    {errors.email && <span style={{ color: 'red' }}>{errors.email}</span>}

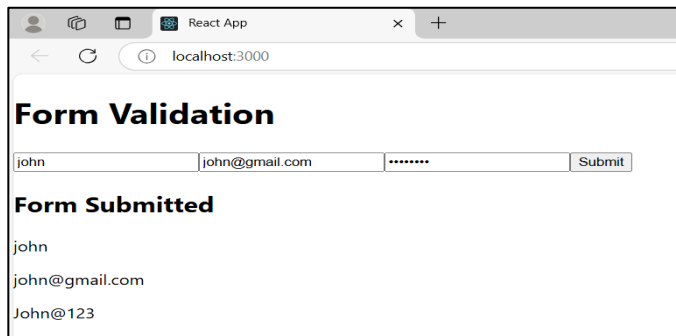
    <input type="password" name="password" value={formValues.password}
onChange={handleInputChange} placeholder="Password" />
    {errors.password && <span style={{ color: 'red' }}>{errors.password}</span>}

    <button type="submit">Submit</button>
  </form>
  {isSubmit && <div><h2>Form
Submitted</h2><p>{formValues.name}</p><p>{formValues.email}</p><p>{formValues.password}</p
></div>}
  </div>
);
}
export default App;

```

Step11: Run this file (npm start)

Output:



Conclusion: This ReactJS form validation ensures proper input before submission, providing real-time feedback and preventing invalid submissions.

Submitted By:

Sign:

Name:

Roll No:

Checked By:
Ms. H. S. Jadhvani

Shram Sadhana Bombay Trust Sanchlit
Arts, Commerce and Science College, Bambhori, Jalgaon
Bachelor of Computer Application (B.C.A.)
Practical: 03

DOP:

DOC:

Title: Write ReactJS code for Applying form components.

Objective: To create a ReactJS job application form with text inputs, radio buttons, and a checkbox, capturing user data and displaying the submitted information.

Theory: Here Steps to do this Practical

Step1: Install NodeJS (LTS FILE)

<https://nodejs.org/en>

Step2: Install Visual Studio Code

<https://code.visualstudio.com/download>

Step3: After Installing the NodeJS and VS code
(Setup the Extensions of NodeJS and VS code)

Step4: Open VS code and install the required Extension

- i. npm
- ii. prettier
- iii. ESLint
- iv. Bracket Pair Colorize
- v. ReactJs Snippets
- vi. Live Server

Step5: Open Terminal in VS code to check the Versions of NodeJS and npm packages

- i. node -v (To Check the NodeJS Version)
- ii. npm -v (To Check the NPM Version)
- iii. npx -v (To Check the NodeJS Package Execute Version)

Step6: After Checking the versions of NodeJS packages we can create a ReactJS app

npx create-react-app 3prac

This command is used to create a ReactJS app

Step7: After creating a ReactJS app, the **cd** command is used in the terminal to navigate into the app's directory. Alternatively, (**code .**)

Step8: After redirecting, The ReactJS app's editor will open, where the right side will show a folder named **3prac**. Inside this folder, there will be a subfolder named **src**, and within **src**, there will be a file named **App.js** where you will write the code.

Step9: In the App.js file, need to install some packages, such as:

1. npm install
2. npm install @babel/plugin-private-property-in-object

These commands will install the necessary dependencies for React application.

Step10: Code to Write in **App.js** File

```
import React, { useState } from 'react';
function App() {
  const [formData, setFormData] = useState({ name: "", email: "", phone: "", gender: "", jobInterest: false });
  const [submittedData, setSubmittedData] = useState(null);

  const handleChange = (e) => {
    const { name, value, type, checked } = e.target;
    setFormData(prev => ({ ...prev, [name]: type === 'checkbox' ? checked : value }));
  };

  const handleSubmit = (e) => { e.preventDefault(); setSubmittedData(formData); };

  return (
    <div className="App">
      <h1>Job Application</h1>

      <form onSubmit={handleSubmit}>

        <input type="text" name="name" value={formData.name} onChange={handleChange}
placeholder="Full Name" required />
        <input type="email" name="email" value={formData.email} onChange={handleChange}
placeholder="Email" required />
        <input type="tel" name="phone" value={formData.phone} onChange={handleChange}
placeholder="Phone" required />

        <div>

          <label><input type="radio" name="gender" value="male" checked={formData.gender === 'male'}
onChange={handleChange} /> Male</label>
          <label><input type="radio" name="gender" value="female" checked={formData.gender ===
'female'} onChange={handleChange} /> Female</label>
          <label><input type="radio" name="gender" value="other" checked={formData.gender === 'other'}
onChange={handleChange} /> Other</label>
        </div>

        <label><input type="checkbox" name="jobInterest" checked={formData.jobInterest}
onChange={handleChange} /> Interested in a Job</label>

      </form>
    </div>
  );
}
```

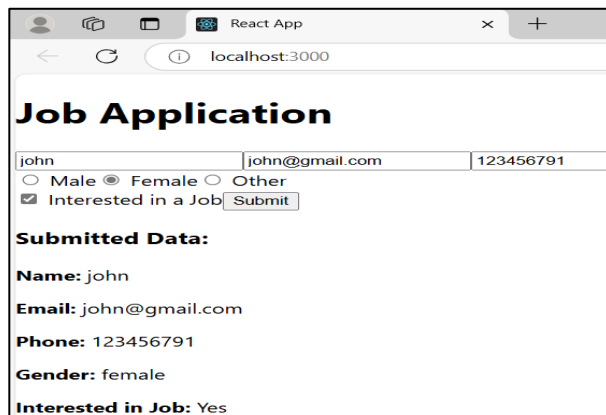
```

<button type="submit">Submit</button>
</form>
{submittedData && (
  <div>
    <h3>Submitted Data:</h3>
    <p><strong>Name:</strong> {submittedData.name}</p>
    <p><strong>Email:</strong> {submittedData.email}</p>
    <p><strong>Phone:</strong> {submittedData.phone}</p>
    <p><strong>Gender:</strong> {submittedData.gender}</p>
    <p><strong>Interested in Job:</strong> {submittedData.jobInterest ? 'Yes' : 'No'}</p>
  </div>
)}
</div>
);
}
export default App;

```

Step11: Run this file (npm start)

Output:



Job Application

john | john@gmail.com | 123456791

☐ Male ☒ Female ☐ Other

☒ Interested in a Job

Submitted Data:

Name: john

Email: john@gmail.com

Phone: 123456791

Gender: female

Interested in Job: Yes

Conclusion: The code demonstrates managing form data with useState, handling various input types, and displaying the submitted information.

Submitted By:

Checked By:
Ms. H. S. Jadhvani

Sign:

Name:

Roll No:

Shram Sadhana Bombay Trust Sanchlit
Arts, Commerce and Science College, Bambhori, Jalgaon
Bachelor of Computer Application (B.C.A.)
Practical: 04

DOP:

DOC:

Title: Write ReactJS code to create student Registration Form.

Objective: To create a student registration form in ReactJS where users can input their name, email, password, and select a course from a dropdown. The form validates input fields and displays an alert upon successful registration

Theory: Here Steps to do this Practical

Step1: Install NodeJS (LTS FILE)

<https://nodejs.org/en>

Step2: Install Visual Studio Code

<https://code.visualstudio.com/download>

Step3: After Installing the NodeJS and VS code
(Setup the Extensions of NodeJS and VS code)

Step4: Open VS code and install the required Extension

- i. npm
- ii. prettier
- iii. ESLint
- iv. Bracket Pair Colorize
- v. ReactJs Snippets
- vi. Live Server

Step5: Open Terminal in VS code to check the Versions of NodeJS and npm packages

- i. node -v (To Check the NodeJS Version)
- ii. npm -v (To Check the NPM Version)
- iii. npx -v (To Check the NodeJS Package Execute Version)

Step6: After Checking the versions of NodeJS packages we can create a ReactJS app

npx create-react-app 4prac

This command is used to create a ReactJS app

Step7: After creating a ReactJS app, the **cd** command is used in the terminal to navigate into the app's directory. Alternatively, (**code .**)

Step8: After redirecting, The ReactJS app's editor will open, where the right side will show a folder named **4prac**. Inside this folder, there will be a subfolder named **src**, and within **src**, there will be a file named **App.js** where you will write the code.

Step9: In the App.js file, need to install some packages, such as:

1. npm install
2. npm install @babel/plugin-private-property-in-object

These commands will install the necessary dependencies for React application.

Step10: Code to Write in **App.js** File

```
import React, { useState } from "react";
function StudentRegistrationForm() {
  const [formData, setFormData] = useState({ name: "", email: "", password: "", course: "" });

  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData(prev => ({ ...prev, [name]: value }));
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    alert("Registration Successful!");
    console.log(formData);
    setFormData({ name: "", email: "", password: "", course: "" });
  };

  return (
    <div style={{ maxWidth: "400px", margin: "auto", padding: "20px" }}>
      <h2>Student Registration</h2>

      <form onSubmit={handleSubmit}>
        <div><label>Name:</label><input type="text" name="name" value={formData.name}
onChange={handleChange} required /></div>
        <div><label>Email:</label><input type="email" name="email" value={formData.email}
onChange={handleChange} required /></div>
        <div><label>Password:</label><input type="password" name="password"
value={formData.password} onChange={handleChange} required /></div>
        <div>
          <label>Course:</label>
          <select name="course" value={formData.course} onChange={handleChange} required>
            <option value="">Select a course</option>
            <option value="Science">Science</option>
            <option value="Commerce">Commerce</option>
            <option value="Arts">Arts</option>
          </select>
        </div>
        <button type="submit">Register</button>
      </form>
    </div>
  );
}
export default StudentRegistrationForm;
```

Step11: Run this file (npm start)

Output:

Student Registration

Name:

Email:

Password:

Course:

Conclusion: This ReactJS code handles form input, manages form data using useState, and processes form submission by resetting the form and showing a success message.

Submitted By:

Checked By:
Ms. H. S. Jadhvani

Sign:

Name:

Roll No:

Shram Sadhana Bombay Trust Sanchlit
Arts, Commerce and Science College, Bambhori, Jalgaon
Bachelor of Computer Application (B.C.A.)
Practical: 05

DOP:

DOC:

Title: Write ReactJS code to create Simple Login Form.

Objective: To create a simple login form in ReactJS where users can input their username and password. The form performs validation and displays an error message for any missing or incorrect fields, and shows a success message upon successful submission.

Theory: Here Steps to do this Practical

Step1: Install NodeJS (LTS FILE)

<https://nodejs.org/en>

Step2: Install Visual Studio Code

<https://code.visualstudio.com/download>

Step3: After Installing the NodeJS and VS code
(Setup the Extensions of NodeJS and VS code)

Step4: Open VS code and install the required Extension

- i. npm
- ii. prettier
- iii. ESLint
- iv. Bracket Pair Colorize
- v. ReactJs Snippets
- vi. Live Server

Step5: Open Terminal in VS code to check the Versions of NodeJS and npm packages

- i. node -v (To Check the NodeJS Version)
- ii. npm -v (To Check the NPM Version)
- iii. npx -v (To Check the NodeJS Package Execute Version)

Step6: After Checking the versions of NodeJS packages we can create a ReactJS app

npx create-react-app 5prac

This command is used to create a ReactJS app

Step7: After creating a ReactJS app, the **cd** command is used in the terminal to navigate into the app's directory. Alternatively, (**code .**)

Step8: After redirecting, The ReactJS app's editor will open, where the right side will show a folder named **5prac**. Inside this folder, there will be a subfolder named **src**, and within **src**, there will be a file named **App.js** where you will write the code.

Step9: In the App.js file, need to install some packages, such as:

1. npm install
2. npm install @babel/plugin-private-property-in-object

These commands will install the necessary dependencies for React application.

Step10: Code to Write in **App.js** File

```
import React, { useState } from 'react';
function LoginForm() {
  const [formData, setFormData] = useState({ username: "", password: "" });
  const [errors, setErrors] = useState({});
  const [isSubmit, setIsSubmit] = useState(false); // Track successful submission

  const handleChange = (e) => setFormData({ ...formData, [e.target.name]: e.target.value });

  const validate = () => {
    const tempErrors = {};
    if (!formData.username) tempErrors.username = 'Username is required';
    if (!formData.password) tempErrors.password = 'Password is required';
    setErrors(tempErrors);
    return Object.keys(tempErrors).length === 0;
  };

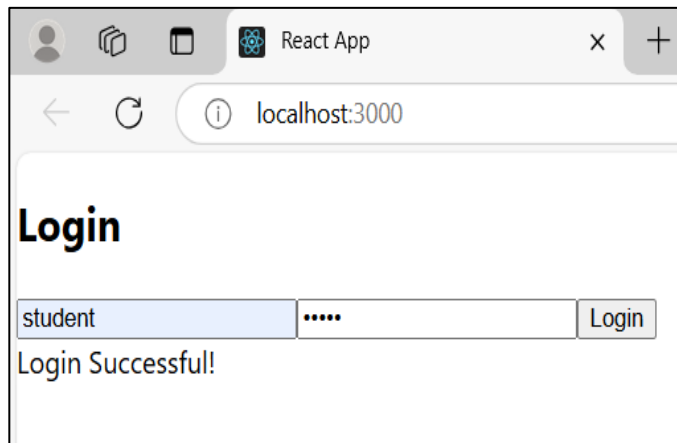
  const handleSubmit = (e) => {
    e.preventDefault();
    if (validate()) {
      setIsSubmit(true); // Set to true when form is valid
      console.log('Login successful', formData); // You can replace this with actual login logic
    }
  };

  return (
    <div>
      <h2>Login</h2>
      <form onSubmit={handleSubmit}>
        <input type="text" name="username" value={formData.username} onChange={handleChange}
placeholder="Username" />
        {errors.username && <span>{errors.username}</span>}
        <input type="password" name="password" value={formData.password} onChange={handleChange}
placeholder="Password" />
        {errors.password && <span>{errors.password}</span>}
        <button type="submit">Login</button>
      </form>
      {isSubmit && <div>Login Successful!</div>} { /* Show success message */}
    </div>
  );
}

export default LoginForm;
```

Step11: Run this file (npm start)

Output:



Conclusion: This ReactJS code handles user input with useState, performs form validation, and provides feedback by displaying error messages for invalid fields and a success message after a successful login.

Submitted By:

Checked By:
Ms. H. S. Jadhvani

Sign:

Name:

Roll No:

Shram Sadhana Bombay Trust Sanchlit
Arts, Commerce and Science College, Bambhori, Jalgaon
Bachelor of Computer Application (B.C.A.)
Practical: 06

DOP:

DOC:

Title: Write ReactJS Create a Single Page Application.

Objective: To create a Single Page Application (SPA) in ReactJS using React Router for navigation. This app includes three pages: Home, About, and Contact, with routing enabled to switch between pages without refreshing the browser.

Theory: Here Steps to do this Practical

Step1: Install NodeJS (LTS FILE)

<https://nodejs.org/en>

Step2: Install Visual Studio Code

<https://code.visualstudio.com/download>

Step3: After Installing the NodeJS and VS code
(Setup the Extensions of NodeJS and VS code)

Step4: Open VS code and install the required Extension

- i. npm
- ii. prettier
- iii. ESLint
- iv. Bracket Pair Colorize
- v. ReactJs Snippets
- vi. Live Server

Step5: Open Terminal in VS code to check the Versions of NodeJS and npm packages

- i. node -v (To Check the NodeJS Version)
- ii. npm -v (To Check the NPM Version)
- iii. npx -v (To Check the NodeJS Package Execute Version)

Step6: After Checking the versions of NodeJS packages we can create a ReactJS app

npx create-react-app 6prac

This command is used to create a ReactJS app

Step7: After creating a ReactJS app, the **cd** command is used in the terminal to navigate into the app's directory. Alternatively, (**code .**)

Step8: After redirecting, The ReactJS app's editor will open, where the right side will show a folder named **6prac**. Inside this folder, there will be a subfolder named **src**, and within **src**, there will be a file named **App.js** where you will write the code.

Step9: In the App.js file, need to install some packages, such as:

1. npm install
2. npm install @babel/plugin-private-property-in-object
3. npm install react-router-dom

These commands will install the necessary dependencies for React application.

Step10: Code to Write in **App.js** File

```
import React from 'react';
import { BrowserRouter as Router, Routes, Route, Link } from 'react-router-dom';
function Home() {
  return <h1>Welcome to the SSBT WORLD</h1>;
}

function About() {
  return <h1>We Are The Students of TYBCA</h1>;
}

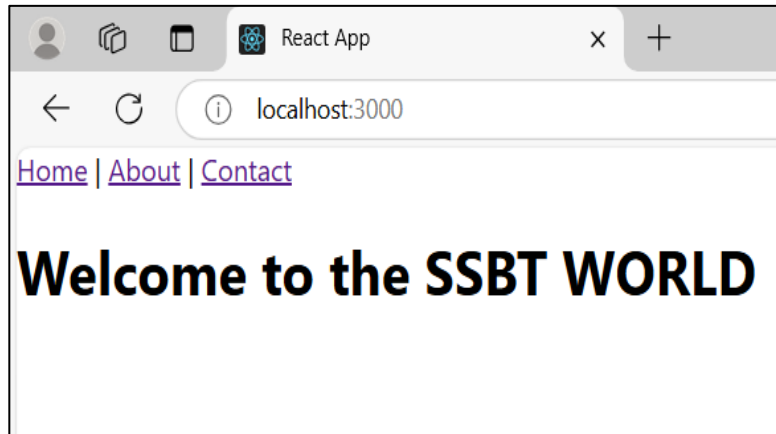
function Contact() {
  return <h1>Please contact us When Any Jobs And Internships Are Availabel</h1>;
}

function App() {
  return (
    <Router>
      <div>
        <nav>
          <Link to="/">Home</Link> | <Link to="/about">About</Link> | <Link
to="/contact">Contact</Link>
        </nav>
        <Routes>
          <Route path="/" element={ <Home /> } />
          <Route path="/about" element={ <About /> } />
          <Route path="/contact" element={ <Contact /> } />
        </Routes>
      </div>
    </Router>
  );
}

export default App;
```

Step11: Run this file (npm start)

Output:



Conclusion: This ReactJS code demonstrates building a Single Page Application using React Router. It allows seamless navigation between different components (Home, About, and Contact) without reloading the page, enhancing user experience with dynamic routing.

Submitted By:

Checked By:
Ms. H. S. Jadhvani

Sign:

Name:

Roll No:

Shram Sadhana Bombay Trust Sanchlit
Arts, Commerce and Science College, Bambhori, Jalgaon
Bachelor of Computer Application (B.C.A.)
Practical: 07

DOP:

DOC:

Title: Write ReactJS /NodeJS code to Applying Routing

Objective: To implement routing in a ReactJS app integrated with a Node.js backend, enabling navigation between pages and fetching course data via a REST API.

Theory: Here Steps to do this Practical

Step1: Install NodeJS (LTS FILE)

<https://nodejs.org/en>

Step2: Install Visual Studio Code

<https://code.visualstudio.com/download>

Step3: After Installing the NodeJS and VS code
(Setup the Extensions of NodeJS and VS code)

Step4: Open VS code and install the required Extension

- i. npm
- ii. prettier
- iii. ESLint
- iv. Bracket Pair Colorize
- v. ReactJs Snippets
- vi. Live Server

Step5: Open Terminal in VS code to check the Versions of NodeJS and npm packages

- i. node -v (To Check the NodeJS Version)
- ii. npm -v (To Check the NPM Version)
- iii. npx -v (To Check the NodeJS Package Execute Version)

Step6: After Checking the versions of NodeJS packages we can create a ReactJS app and a Express (Backend app)

`mkdir 7prac`

`cd 7prac`

After creating and navigating the folder we need to create a file named (server.js) for backend
npx create-react-app 7practical for frontend

This command is used to create a ReactJS app

Step7: After creating a ReactJS app, the **cd** command is used in the terminal to navigate into the app's directory. Alternatively, (**code .**)

Step8: After redirecting, The ReactJS app's editor will open, where the right side will show a folder named **7practical**. Inside this folder, there will be a subfolder named **src**, and within **src**, there will be a file named **App.js** where you will write the code.

Step9: In the App.js file, need to install some packages, such as

- i. npm install
- ii. npm install @babel/plugin-private-property-in-object
- iii. npm install react-router-dom
- iv. npm init -y

These commands will install the necessary dependencies for React application.

Step10: Code to Write in **server.js** File

```
const express = require('express');
const cors = require('cors');
const app = express();
const PORT = 5000;
app.use(cors());
const courses = [
  { id: 1, name: "Mathematics", description: "Learn algebra, geometry, and calculus." },
  { id: 2, name: "Physics", description: "Explore the laws of nature." },
  { id: 3, name: "Computer Science", description: "Understand programming and algorithms." },
];
app.get('/api/courses', (req, res) => {
  res.json(courses);
});
app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});
```

Step11: Run this file (node server.js)

App.js

```
import React, { useState, useEffect } from "react";
import { BrowserRouter as Router, Routes, Route, Link } from "react-router-dom";
import axios from "axios";
```

```
function App() {
  return (
    <Router>
      <nav>
        <Link to="/">Home</Link> | <Link to="/courses">Courses</Link>
      </nav>
      <Routes>
        <Route path="/" element={<div><h1>Welcome to Education</h1></div>} />
        <Route path="/courses" element={<Courses />} />
      </Routes>
    </Router>
  );
}
```

```

function Courses() {
  const [courses, setCourses] = useState([]);
  useEffect(() => {

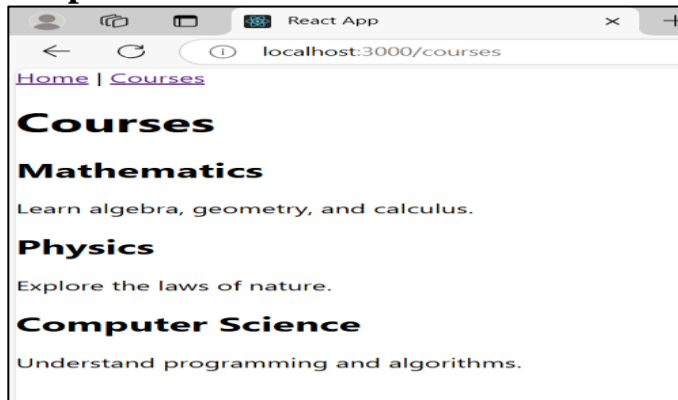
    axios.get("http://localhost:5000/api/courses")
      .then(res => setCourses(res.data))
      .catch(console.error);
  }, []);

  return (
    <div>
      <h1>Courses</h1>
      {courses.length ? courses.map(c => (
        <div key={c.id}><h2>{c.name}</h2><p>{c.description}</p></div>
      )) : <p>Loading...</p>}
    </div>
  );
}
export default App;

```

Step11: Run this file (npm start)

Output:



Conclusion: The code demonstrates seamless integration of React Router for navigation, Axios for data fetching, and Express.js for backend API handling. It provides a scalable and modular approach for client-server communication.

Submitted By:

Checked By:
Ms. H. S. Jadhvani

Sign:

Name:

Roll No:

Shram Sadhana Bombay Trust Sanchlit
Arts, Commerce and Science College, Bambhori, Jalgaon
Bachelor of Computer Application (B.C.A.)
Practical: 08

DOP:

DOC:

Title: Write ReactJS /NodeJS code to demonstrate the use of POST Method.

Objective: To demonstrate the use of the POST method in a ReactJS and Node.js application, where data from a form is sent to a backend server, processed, and a response is returned.

Theory: Here Steps to do this Practical

Step1: Install NodeJS (LTS FILE)

<https://nodejs.org/en>

Step2: Install Visual Studio Code

<https://code.visualstudio.com/download>

Step3: After Installing the NodeJS and VS code

(Setup the Extensions of NodeJS and VS code)

Step4: Open VS code and install the required Extension

- i. npm
- ii. prettier
- iii. ESLint
- iv. Bracket Pair Colorize
- v. ReactJs Snippets
- vi. Live Server

Step5: Open Terminal in VS code to check the Versions of NodeJS and npm packages

- i. node -v (To Check the NodeJS Version)
- ii. npm -v (To Check the NPM Version)
- iii. npm -v (To Check the NodeJS Package Execute Version)

Step6: After Checking the versions of NodeJS packages we can create a ReactJS app and a Express (Backend app)

mkdir 8prac

cd 8prac

After creating and navigating the folder we need to create a file named (server.js) for backend

npx create-react-app 8practical for frontend

This command is used to create a ReactJS app

Step7: After creating a ReactJS app, the **cd** command is used in the terminal to navigate into the app's directory. Alternatively, (**code .**)

Step8: After redirecting, The ReactJS app's editor will open, where the right side will show a folder named **8practical**. Inside this folder, there will be a subfolder named **src**, and within **src**, there will be a file named **App.js** where you will write the code.

Step9: In the App.js file, need to install some packages, such as:

- i. npm install
- ii. npm install @babel/plugin-private-property-in-object
- iii. npm install react-router-dom
- iv. npm init -y
- v. npm install web-vitals

These commands will install the necessary dependencies for React application.

Step10: Code to Write in **server.js** File

```
const express = require("express");
const bodyParser = require("body-parser");
const cors = require("cors");
const app = express();
app.use(cors());
app.use(bodyParser.json());
app.post("/api/data", (req, res) => {
  const { name, email } = req.body;
  console.log("POST request received:", name, email);
  res.json({ message: "Data received successfully!" });
});
app.listen(5000, () => console.log("Server running on http://localhost:5000));
```

Step11: Run this file (node server.js)

App.js

```
import React, { useState } from "react";
import axios from "axios";
function App() {
  const [formData, setFormData] = useState({ name: "", email: "" });
  const [responseMessage, setResponseMessage] = useState("");

  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData({ ...formData, [name]: value });
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    axios.post("http://localhost:5000/api/data", formData)
      .then(res => setResponseMessage(res.data.message))
      .catch(err => console.error(err));
  };

  return (
    <div>
```



```

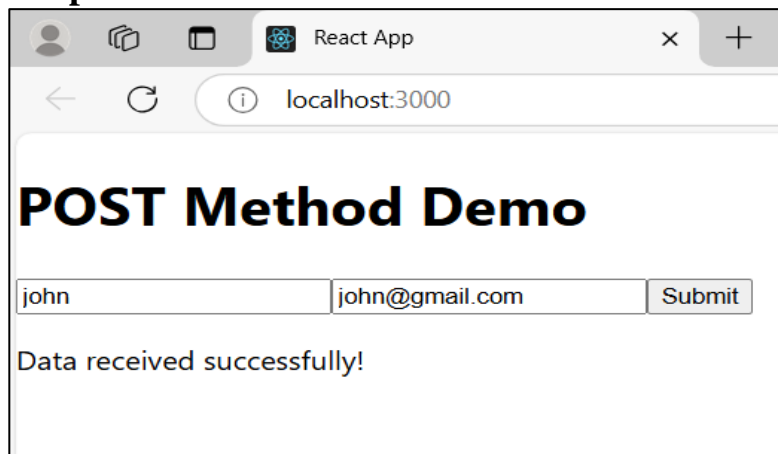
<h1>POST Method Demo</h1>
  <form onSubmit={handleSubmit}>
    <input type="text" name="name" placeholder="Name" value={formData.name}
onChange={handleChange} required />

    <input type="email" name="email" placeholder="Email" value={formData.email}
onChange={handleChange} required />
    <button type="submit">Submit</button>
  </form>
  {responseMessage && <p>{responseMessage}</p>}
</div>
);
}
export default App;

```

Step11: Run this file (npm start)

Output:



Conclusion: The code effectively shows how to handle POST requests using Axios in React and Express.js on the backend. It enables secure and seamless data submission, making it suitable for real-world applications like form handling and API integration.

Submitted By:

Sign:

Name:

Roll No:

Checked By:
Ms. H. S. Jadhvani

Shram Sadhana Bombay Trust Sanchlit
Arts, Commerce and Science College, Bambhori, Jalgaon
Bachelor of Computer Application (B.C.A.)
Practical: 09

DOP:

DOC:

Title: Write ReactJS/NodeJS code to demonstrate the use of GET Method.

Objective: To demonstrate the use of the GET method in a ReactJS and Node.js application, where data is fetched from a server API and displayed dynamically in the frontend.

Theory: Here Steps to do this Practical

Step1: Install NodeJS (LTS FILE)

<https://nodejs.org/en>

Step2: Install Visual Studio Code

<https://code.visualstudio.com/download>

Step3: After Installing the NodeJS and VS code
(Setup the Extensions of NodeJS and VS code)

Step4: Open VS code and install the required Extension

- i. npm
- ii. prettier
- iii. ESLint
- iv. Bracket Pair Colorize
- v. ReactJs Snippets
- vi. Live Server

Step5: Open Terminal in VS code to check the Versions of NodeJS and npm packages

- i. node -v (To Check the NodeJS Version)
- ii. npm -v (To Check the NPM Version)
- iii. npm -v (To Check the NodeJS Package Execute Version)

Step6: After Checking the versions of NodeJS packages we can create a ReactJS app and a Express (Backend app)

mkdir 9prac

cd 9prac

After creating and navigating the folder we need to create a file named (server.js) for backend

npx create-react-app 9practical for frontend

This command is used to create a ReactJS app

Step7: After creating a ReactJS app, the **cd** command is used in the terminal to navigate into the app's directory. Alternatively, (**code .**)

Step8: After redirecting, The ReactJS app's editor will open, where the right side will show a folder named **9practical**. Inside this folder, there will be a subfolder named **src**, and within **src**, there will be a file named **App.js** where you will write the code.

Step9: In the App.js file, need to install some packages, such as

- i. npm install
- ii. npm install @babel/plugin-private-property-in-object
- iii. npm install react-router-dom
- iv. npm init -y
- v. npm install web-vitals

These commands will install the necessary dependencies for React application.

Step10: Code to Write in **server.js** File

```
const express = require("express");
const cors = require("cors");
const app = express();
app.use(cors());

// Mock data to simulate server response
const mockData = [
  { id: 1, name: "John Doe", email: "john@example.com" },
  { id: 2, name: "Jane Smith", email: "jane@example.com" },
];

app.get("/api/data", (req, res) => {
  console.log("GET request received");
  res.json(mockData);
});

app.listen(5000, () => console.log("Server running on http://localhost:5000));
```

Step11: Run this file (node server.js)

App.js

```
import React, { useState, useEffect } from "react";
import axios from "axios";
function App() {
  const [data, setData] = useState([]);

  useEffect(() => {
    axios.get("http://localhost:5000/api/data")
      .then(res => setData(res.data))

    .catch(err => console.error(err));
  }, []);

  return (
    <div>
```

```

<h1>GET Method Demo</h1>
<h3>Data from Server:</h3>
<ul>
  {data.map((item) => (
    <li key={item.id}>
      {item.name} - {item.email}
    </li>
  ))}
</ul>
</div>
);
}
export default App;

```

Step11: Run this file (npm start)

Output:



Conclusion: The code effectively illustrates fetching data using the GET method with Axios in React and serving mock data via Express.js. It highlights efficient client-server interaction for data retrieval, making it ideal for applications requiring real-time data display.

Submitted By:

Sign:

Name:

Roll No:

Checked By:
Ms. H. S. Jadhvani

Shram Sadhana Bombay Trust Sanchlit
Arts, Commerce and Science College, Bambhori, Jalgaon
Bachelor of Computer Application (B.C.A.)
Practical: 10

DOP:

DOC:

Title: To demonstrate REST API in Node JS

Objective: The objective of this project is to create a RESTful API using Node.js and Express that performs basic CRUD operations (Create, Read, Delete) on user data. This API demonstrates how to handle requests such as fetching all users, adding a new user, and deleting a user, with data being stored in a mock in-memory array.

Theory: Here Steps to do this Practical

Step1: Install NodeJS (LTS FILE)

<https://nodejs.org/en>

Step2: Install Visual Studio Code

<https://code.visualstudio.com/download>

Step3: After Installing the NodeJS and VS code

(Setup the Extensions of NodeJS and VS code)

Step4: Open VS code and install the required Extension

- i. npm
- ii. prettier
- iii. ESLint
- iv. Bracket Pair Colorize
- v. ReactJs Snippets
- vi. Live Server

Step5: Open Terminal in VS code to check the Versions of NodeJS and npm Packages

- i. node -v (To Check the NodeJS Version)
- ii. npm -v (To Check the NPM Version)

Step6: You need to create a folder using **mkdir** in the VS Code terminal, then navigate to that folder using **cd**. After that, you should create a file named index.js in that folder and write your code in it.

mkdir 10prac

cd 10prac

Step7: In the index.js file, need to install some packages, such as

- i. npm install
- ii. npm install @babel/plugin-private-property-in-object
- iii. npm install react-router-dom
- iv. npm init -y
- v. npm install web-vitals

These commands will install the necessary dependencies for React application.

Step10: Code to Write in **index.js** File

```

const express = require('express');
const app = express();
const port = 3000; // You can change this if needed

// Middleware to parse JSON
app.use(express.json());

// Root route
app.get('/', (req, res) => {
  res.send('Welcome to SSBT College!');
});

// Mock database
let users = [
  { id: 1, name: 'John Doe' },
  { id: 2, name: 'Jane Doe' }
];

// Routes for users
app.get('/api/users', (req, res) => {
  res.json(users);
});

app.get('/api/users/:id', (req, res) => {
  const user = users.find(u => u.id === parseInt(req.params.id));
  if (!user) return res.status(404).send('User not found');
  res.json(user);
});

app.post('/api/users', (req, res) => {
  const newUser = {
    id: users.length + 1,
    name: req.body.name
  };
  users.push(newUser);
  res.status(201).json(newUser);
});

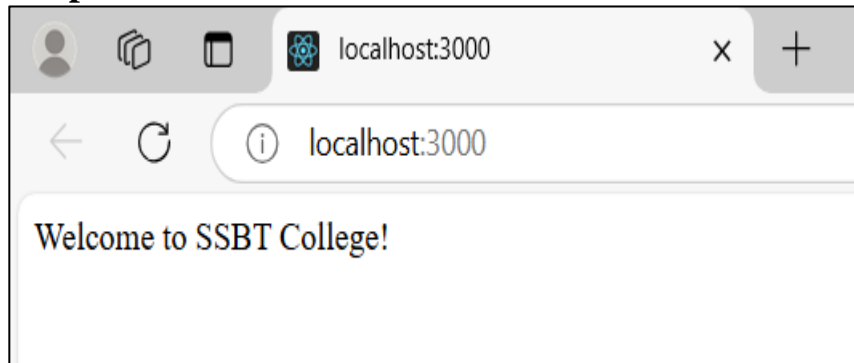
app.delete('/api/users/:id', (req, res) => {
  users = users.filter(u => u.id !== parseInt(req.params.id));
  res.send('User deleted');
});

// Start the server
app.listen(port, () => {
  console.log(`Server is running at http://localhost:${port}`);
});

```

Step11: Run this file (node server.js)

Output:



Conclusion: The API successfully handles HTTP requests (GET, POST, DELETE) and allows interaction with a mock database. It demonstrates how to build and manage routes, parse JSON data, and return appropriate responses. This serves as a foundation for more complex API development.

Submitted By:

Sign:

Name:

Roll No:

Checked By:
Ms. H. S. Jadhvani

Shram Sadhana Bombay Trust Sanchlit
Arts, Commerce and Science College, Bambhori, Jalgaon
Bachelor of Computer Application (B.C.A.)
Practical: 11

DOP:

DOC:

Title: Create Node JS Application for to stored student information in database.

Objective: To create a Node.js application that stores and manages student information in a database. The app will support CRUD operations (Create, Read, Update, Delete) to manage student records, such as name, age, and class.

Theory: Here Steps to do this Practical

Step1: Install NodeJS (LTS FILE)

<https://nodejs.org/en>

Step2: Install Visual Studio Code

<https://code.visualstudio.com/download>

Step3: After Installing the NodeJS and VS code
(Setup the Extensions of NodeJS and VS code)

Step4: Open VS code and install the required Extension

- i. npm
- ii. prettier
- iii. ESLint
- iv. Bracket Pair Colorize
- v. ReactJs Snippets
- vi. Live Server

Step5: Open Terminal in VS code to check the Versions of NodeJS and npm packages

- i. node -v (To Check the NodeJS Version)
- ii. npm -v (To Check the NPM Version)
- iii. npm -v (To Check the NodeJS Package Execute Version)

Step6: After Checking the versions of NodeJS packages we can create a ReactJS app and a Express (Backend app)

`mkdir 11prac`

`cd 11prac`

After creating and navigating the folder we need to create a file named (server.js) for backend

`npx create-react-app 11practical` for frontend

This command is used to create a ReactJS app

Step7: After creating a ReactJS app, the **cd** command is used in the terminal to navigate into the app's directory. Alternatively, (**code .**)

Step8: After redirecting, The ReactJS app's editor will open, where the right side will show a folder

named **11practical**. Inside this folder, there will be a subfolder named **src**, and within **src**, there will be a file named **App.js** where you will write the code.

Step9: In the App.js file, need to install some packages, such as

- i. npm install
- ii. npm install @babel/plugin-private-property-in-object
- iii. npm install react-router-dom
- iv. npm init -y
- v. npm install web-vitals
- vi. npm install axios
- vii. npm install express mongoose cors

These commands will install the necessary dependencies for React application.

Step10: Code to Write in **server.js** File

```
const express = require("express");
const mongoose = require("mongoose");
const cors = require("cors");
const app = express();
app.use(express.json());
app.use(cors());
```

// MongoDB Connection (Database name: studentsDB)

```
mongoose.connect("mongodb://localhost:27017/studentsDB", {
  useNewUrlParser: true,
  useUnifiedTopology: true,
}).then(() => console.log("MongoDB connected"))
.catch((err) => console.error(err));
```

// Schema and Model (Collection name: Student)

```
const Student = mongoose.model("Student", new mongoose.Schema({
  name: String,
  age: Number,
  grade: String,
}));
```

```
app.post("/add-student", async (req, res) => {
  try {
    await new Student(req.body).save();
    res.send("Added!");
  } catch (err) {
    res.status(500).send(err.message);
  }
});
```

```
app.get("/students", async (req, res) => {
  try {
```

```

res.json(await Student.find());
} catch (err) {
  res.status(500).send(err.message);
}
});

app.listen(5000, () => console.log("Server on http://localhost:5000"));

```

Step11: Run this file (node server.js)

App.js

```

import React, { useState, useEffect } from "react";
import axios from "axios";
const App = () => {
  const [students, setStudents] = useState([]);
  const [student, setStudent] = useState({ name: "", age: "", grade: "" });

  useEffect(() => {
    axios.get("http://localhost:5000/students")
      .then((res) => setStudents(res.data))
      .catch((err) => console.error(err));
  }, []);

  const handleSubmit = (e) => {
    e.preventDefault();
    axios.post("http://localhost:5000/add-student", student)
      .then(() => {
        setStudent({ name: "", age: "", grade: "" });
        axios.get("http://localhost:5000/students")
          .then((res) => setStudents(res.data));
      });
  };

  return (
    <div>
      <h1>Student Management</h1>
      <form onSubmit={handleSubmit}>
        <input type="text" placeholder="Name" required
          value={student.name} onChange={(e) => setStudent({ ...student, name: e.target.value })} />
        <input type="number" placeholder="Age" required
          value={student.age} onChange={(e) => setStudent({ ...student, age: e.target.value })} />
        <input type="text" placeholder="Grade" required
          value={student.grade} onChange={(e) => setStudent({ ...student, grade: e.target.value })} />
        <button type="submit">Add</button>
      </form>
    </div>
  );
};

```

```

<ul>
  {students.map((s) => (
    <li key={s._id}>{s.name} - {s.age} years - Grade: {s.grade}</li>
  ))}
</ul>
</div>
);
};
export default App;

```

Step11: Run this file (npm start)

Output:

Student Management

Name	Age	Grade	Add
------	-----	-------	-----

- john - 20 years - Grade: A
- jack - 21 years - Grade: B
- Marry - 25 years - Grade: C
- sam - 27 years - Grade: A

Conclusion: The app uses Node.js, React.js, and MongoDB to manage student data:

- Backend: Handles data storage and APIs.
- Frontend: Collects and displays data dynamically.
- Integration: Ensures smooth communication with axios.

Submitted By:

Checked By:
Ms. H. S. Jadhvani

Sign:

Name:

Roll No:

Shram Sadhana Bombay Trust Sanchlit
Arts, Commerce and Science College, Bambhori, Jalgaon
Bachelor of Computer Application (B.C.A.)
Practical: 12

DOP:

DOC:

Title: Create Node JS Application for login credentials.

Objective: The objective of this Node.js application is to create a simple and functional login system. It demonstrates the basics of handling HTTP requests, parsing form data, and validating user credentials using Express.js. The application serves as a foundation for understanding how to build authentication systems in web applications.

Theory: Here Steps to do this Practical

Step1: Install NodeJS (LTS FILE)

<https://nodejs.org/en>

Step2: Install Visual Studio Code

<https://code.visualstudio.com/download>

Step3: After Installing the NodeJS and VS code

(Setup the Extensions of NodeJS and VS code)

Step4: Open VS code and install the required Extension

- i. npm
- ii. prettier
- iii. ESLint
- iv. Bracket Pair Colorize
- v. ReactJs Snippets
- vi. Live Server

Step5: Open Terminal in VS code to check the Versions of NodeJS and npm Packages

- i. node -v (To Check the NodeJS Version)
- ii. npm -v (To Check the NPM Version)

Step6: You need to create a folder using **mkdir** in the VS Code terminal, then navigate to that folder using **cd**. After that, you should create a file named index.js in that folder and write your code in it.

mkdir 12prac

cd 12prac

Step7: In the index.js file, need to install some packages, such as

- i. npm install
- ii. npm install @babel/plugin-private-property-in-object
- iii. npm install react-router-dom
- iv. npm init -y
- v. npm install web-vitals
- vi. npm install express body-parser

These commands will install the necessary dependencies for React application.

Step10: Code to Write in **index.js** File

```
const express = require('express');
const bodyParser = require('body-parser');

const app = express();
const PORT = 3000;

// Middleware to parse form data
app.use(bodyParser.urlencoded({ extended: true }));

// Serve static files (optional, for HTML)
app.use(express.static('public'));

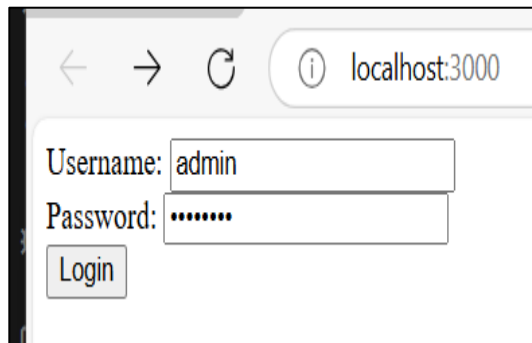
// Route for the login form
app.get('/', (req, res) => {
  res.send(`
    <form action="/login" method="post">
      <label for="username">Username:</label>
      <input type="text" id="username" name="username" required>
      <br>
      <label for="password">Password:</label>
      <input type="password" id="password" name="password" required>
      <br>
      <button type="submit">Login</button>
    </form>
  `);
});

// Route to handle login
app.post('/login', (req, res) => {
  const { username, password } = req.body;

  // Example validation (you can replace this with database validation)
  if (username === 'admin' && password === 'password') {
    res.send(`<h1>Welcome, ${username}!</h1>`);
  } else {
    res.send(`<h1>Invalid credentials. Please try again.</h1>`);
  }
});

// Start the server
app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});
```

Step11: Run this file (node index.js)

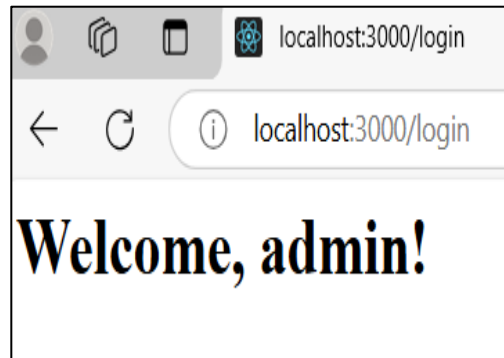
Output:

← → ↻ ⓘ localhost:3000

Username:

Password:

Login



Conclusion: This practical application showcases the essential steps to set up a Node.js server and implement basic user login functionality. By completing this, you gain insights into handling form submissions, validating credentials, and serving dynamic responses. This knowledge is a stepping stone to building more secure and robust authentication systems.

Submitted By:**Sign:****Name:****Roll No:****Checked By:**
Ms. H. S. Jadhvani

Shram Sadhana Bombay Trust Sanchlit
Arts, Commerce and Science College, Bambhori, Jalgaon
Bachelor of Computer Application (B.C.A.)
Practical: 13

DOP:

DOC:

Title: Create Node JS Application to display student information.

Objective: The objective of this Node.js application is to create a simple web server using Express.js to display student information. The application will serve student data, such as the student's name, age, and course, on the homepage of the web application.

Theory: Here Steps to do this Practical

Step1: Install NodeJS (LTS FILE)

<https://nodejs.org/en>

Step2: Install Visual Studio Code

<https://code.visualstudio.com/download>

Step3: After Installing the NodeJS and VS code
(Setup the Extensions of NodeJS and VS code)

Step4: Open VS code and install the required Extension

- i. npm
- ii. prettier
- iii. ESLint
- iv. Bracket Pair Colorize
- v. ReactJs Snippets
- vi. Live Server

Step5: Open Terminal in VS code to check the Versions of NodeJS and npm Packages

- i. node -v (To Check the NodeJS Version)
- ii. npm -v (To Check the NPM Version)

Step6: You need to create a folder using **mkdir** in the VS Code terminal, then navigate to that folder using **cd**. After that, you should create a file named index.js in that folder and write your code in it.

mkdir 13prac

cd 13prac

Step7: In the index.js file, need to install some packages, such as

- i. npm install
- ii. npm install @babel/plugin-private-property-in-object
- iii. npm install react-router-dom
- iv. npm init -y
- v. npm install web-vitals
- vi. npm install express body-parser
- vii. npm install express

These commands will install the necessary dependencies for React application.

Step10: Code to Write in **index.js** File

```
// Step 1: Import the Express module
```

```
const express = require('express');
```

```
// Step 2: Create an instance of Express app
```

```
const app = express();
```

```
// Step 3: Set the port where the app will run
```

```
const port = 3000;
```

```
// Step 4: Define student data
```

```
const students = [
```

```
  { id: 1, name: 'John', age: 20, course: 'Mathematics' },
```

```
  { id: 2, name: 'Jane', age: 22, course: 'Physics' },
```

```
  { id: 3, name: 'Sam ', age: 19, course: 'Computer Science' } 
```

```
];
```

```
// Step 5: Set up the route to display student information
```

```
app.get('/', (req, res) => {
```

```
  let studentList = '<h1>Student Information</h1><ul>';
```

```
  students.forEach(student => {
```

```
    studentList += `<li>${student.name}, ${student.age} years old, studying ${student.course}</li>`;
```

```
  });
```

```
  studentList += '</ul>';
```

```
  res.send(studentList); // Send the student data as HTML
```

```
});
```

```
// Step 6: Start the server
```

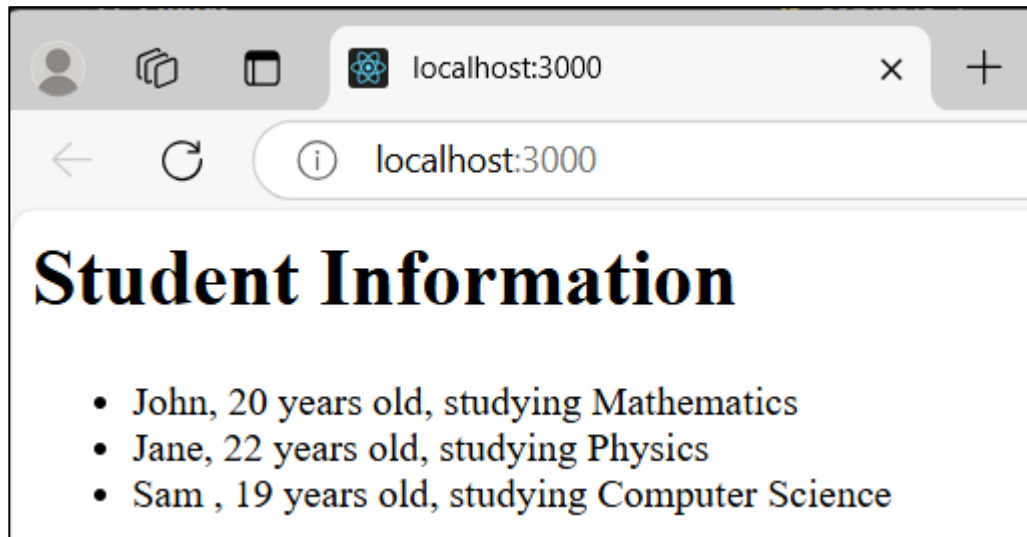
```
app.listen(port, () => {
```

```
  console.log(`Server is running on http://localhost:${port}`);
```

```
});
```

Step11: Run this file (node index.js)

Output:



Conclusion: This Node.js application demonstrates how to create a simple web application using Express to serve dynamic data. By following the above steps, you can understand how to:

1. Set up a basic server with Express.
2. Serve dynamic content (like student information) in an HTML format.
3. Use Node.js and Express to handle HTTP requests and send responses to clients.

The application serves as a foundational exercise in working with Node.js and Express, providing a basis for building more complex web applications with dynamic data handling and routing.

Submitted By:

Checked By:
Ms. H. S. Jadhvani

Sign:

Name:

Roll No:

Shram Sadhana Bombay Trust Sanchlit
Arts, Commerce and Science College, Bambhori, Jalgaon
Bachelor of Computer Application (B.C.A.)
Practical: 14

DOP:

DOC:

Title: Create Node JS Application to update, display and delete student information.

Objective: Create a backend API using Node.js to manage student data, providing functionality to update, display, and delete student information. The application will connect to a MongoDB database and expose RESTful routes for each operation.

Theory: Here Steps to do this Practical

Step1: Install NodeJS (LTS FILE)

<https://nodejs.org/en>

Step2: Install Visual Studio Code

<https://code.visualstudio.com/download>

Step3: After Installing the NodeJS and VS code
(Setup the Extensions of NodeJS and VS code)

Step4: Open VS code and install the required Extension

- i. npm
- ii. prettier
- iii. ESLint
- iv. Bracket Pair Colorize
- v. ReactJs Snippets
- vi. Live Server

Step5: Open Terminal in VS code to check the Versions of NodeJS and npm packages

- i. node -v (To Check the NodeJS Version)
- ii. npm -v (To Check the NPM Version)
- iii. npm -v (To Check the NodeJS Package Execute Version)

Step6: After Checking the versions of NodeJS packages we can create a ReactJS app and a Express (Backend app)

`mkdir 11prac`

`cd 11prac`

After creating and navigating the folder we need to create a file named (server.js) for backend
npx create-react-app 11practical for frontend

This command is used to create a ReactJS app

Step7: After creating a ReactJS app, the `cd` command is used in the terminal to navigate into the app's directory. Alternatively, (`code .`)

Step8: After redirecting, The ReactJS app's editor will open, where the right side will show a folder

named **11practical**. Inside this folder, there will be a subfolder named **src**, and within **src**, there will be a file named **App.js** where you will write the code.

Step9: In the App.js file, need to install some packages, such as

- i. npm install
- ii. npm install @babel/plugin-private-property-in-object
- iii. npm install react-router-dom
- iv. npm init -y
- v. npm install web-vitals
- vi. npm install axios
- vii. npm install express mongoose cors

These commands will install the necessary dependencies for React application.

Step10: Code to Write in **server.js** File

```
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const app = express();
const PORT = 5000;
// Middleware
app.use(express.json());
app.use(cors());
// MongoDB Connection
mongoose.connect('mongodb://localhost:27017/studentDB')
  .then(() => console.log('Connected to MongoDB'))
  .catch(err => console.error(err));
// Mongoose Schema & Model
const Student = mongoose.model('Student', { name: String, grade: String });
// Routes
app.get('/students', async (req, res) => res.json(await Student.find()));
app.post('/students', async (req, res) => res.json(await new Student(req.body).save()));
app.delete('/students/:id', async (req, res) => res.json(await Student.findByIdAndDelete(req.params.id)));
// Start Server
app.listen(PORT, () => console.log(`Server running at http://localhost:${PORT}`));
```

Step11: Run this file (node server.js)

App.js

```
import React, { useState, useEffect } from 'react';
const App = () => {
  const [students, setStudents] = useState([]);
  const [name, setName] = useState("");
  const [grade, setGrade] = useState("");

  const fetchStudents = async () => {
    const res = await fetch('http://localhost:5000/students');
    setStudents(await res.json());
  };

  const addStudent = async () => {
    await fetch('http://localhost:5000/students', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ name, grade }),
    });
    fetchStudents();
  };

  const deleteStudent = async (id) => {
    await fetch(`http://localhost:5000/students/${id}`, { method: 'DELETE' });
    fetchStudents();
  };

  useEffect(() => {
    fetchStudents();
  }, []);

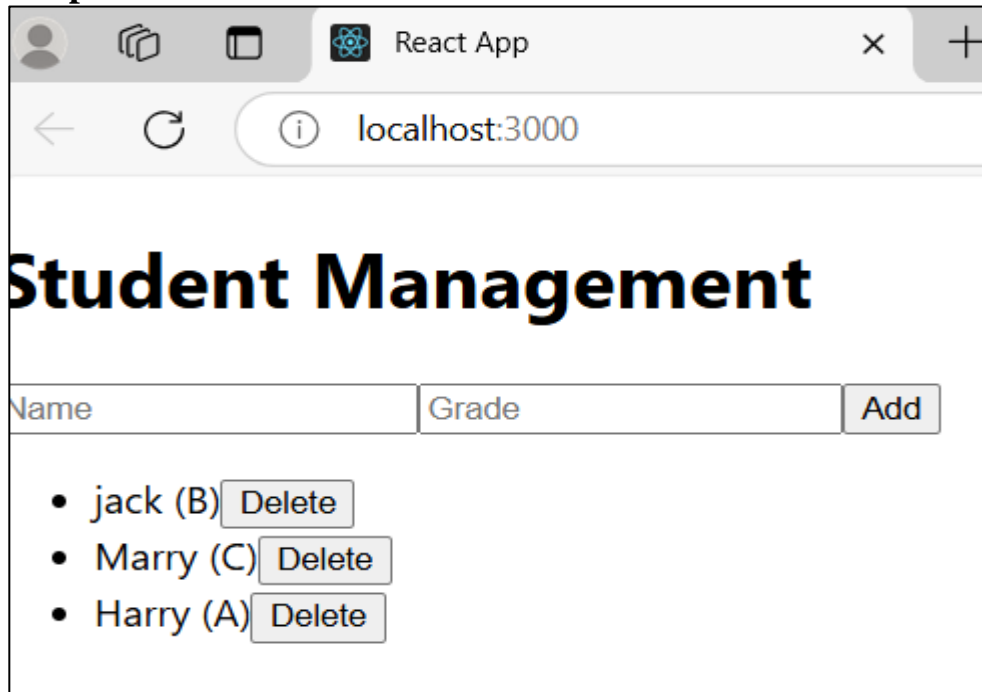
  return (
    <div>
      <h1>Student Management</h1>
      <input value={name} onChange={e => setName(e.target.value)} placeholder="Name" />
      <input value={grade} onChange={e => setGrade(e.target.value)} placeholder="Grade" />
      <button onClick={addStudent}>Add</button>
      <ul>
        {students.map(student => (
          <li key={student._id}>
            {student.name} ({student.grade})
            <button onClick={() => deleteStudent(student._id)}>Delete</button>
          </li>
        ))}
      </ul>
    </div>
  )
}
```

```
</div>  
);  
};
```

export default App;

Step11: Run this file (node server.js)

Output:



Conclusion: The Node.js application provides a robust backend for managing student information with operations to add, update, display, and delete student records. By integrating MongoDB as the database and using Express for handling HTTP requests, the application offers a simple yet powerful way to manage data. This setup can be extended for more advanced features like authentication, validation, and error handling, making it a scalable solution for real-world applications.

Submitted By:

Checked By:
Ms. H. S. Jadhvani

Sign:

Name:

Roll No: