Experiment: Create Login Form with React State Management

Aim

To create a login form in React using the `useState` hook to manage local component state, capture and handle user input, and display validation messages for empty fields.

Tools Used

- React.js (JavaScript library for building UI)
- Node.js (for running React application)
- Visual Studio Code (for code editing)
- Browser (for testing the application)

Procedure

1. Setup React App:

```
npx create-react-app login-form
cd login-form
npm start
```

2. Create Login Component: In `src` folder, create a file `LoginForm.js`.
3. Import React and useState Hook:

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

4. Initialize State for Form Fields:

```
const [username, setUsername] = useState("");
const [password, setPassword] = useState("");
const [error, setError] = useState("");
```

5. Handle Input Changes:

```
const handleUsernameChange = (e) => setUsername(e.target.value);
const handlePasswordChange = (e) => setPassword(e.target.value);
```

6. Handle Form Submission with Validation:

```
const handleSubmit = (e) => {
    e.preventDefault();
    if (!username || !password) {
        setError("Both fields are required!");
    } else {
        console.log("Username:", username);
        console.log("Password:", password);
        setError("");
    }
};
```

7. Render Form in JSX:

```
return (
    <div style={{ width: "300px", margin: "50px auto" }}>
        <h2>Login Form</h2>
        <form onSubmit={handleSubmit}>
            <div>
                <label>Username:</label>
                <input
                    type="text"
                    value={username}
                    onChange={handleUsernameChange}
                />
            </div>
            <div>
                <label>Password:</label>
                <input
                    type="password"
                    value={password}
                    onChange={handlePasswordChange}
                />
            </div>
            {error && <p style={{ color: "red" }}>{error}</p>}
            <button type="submit">Login</button>
        </form>
    </div>
);
```

8. Use Login Component in App.js:

```
import React from "react";
import LoginForm from "./LoginForm";

function App() {
    return (
        <div>
            <LoginForm />
        </div>
    );
}

export default App;
```

9. Test the Application: Run `npm start`, enter values, click Login, and observe console logs and validation messages.

Code (Complete `LoginForm.js`)

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

const LoginForm = () => {
    const [username, setUsername] = useState("");
    const [password, setPassword] = useState("");
    const [error, setError] = useState("");

    const handleUsernameChange = (e) => setUsername(e.target.value);
    const handlePasswordChange = (e) => setPassword(e.target.value);

    const handleSubmit = (e) => {
        e.preventDefault();
        if (!username || !password) {
            setError("Both fields are required!");
        } else {
            console.log("Username:", username);
            console.log("Password:", password);
            setError("");
        }
    };

    return (
        <div style={{ width: "300px", margin: "50px auto" }}>
            <h2>Login Form</h2>
            <form onSubmit={handleSubmit}>
                <div>
                    <label>Username:</label>
                    <input
                        type="text"
                        value={username}
                        onChange={handleUsernameChange}
                    />
                </div>
                <div>
                    <label>Password:</label>
                    <input
                        type="password"
                        value={password}
                        onChange={handlePasswordChange}
                    />
                </div>
                {error && <p style={{ color: "red" }}>{error}</p>}
                <button type="submit">Login</button>
            </form>
        </div>
    );
};

export default LoginForm;
```

Output

1. Login form with Username and Password fields is displayed.
2. Username and Password are logged in console upon submission.
3. Validation message "Both fields are required!" appears if fields are empty.
4. State updates in real-time as user types.

Learning Outcomes

1. Understand `useState` for managing component state.
2. Capture and handle form inputs dynamically.
3. Implement simple form validation.
4. Handle form submission events in React.
5. Learn console logging for debugging and verification.