**BHARATIYA VIDYA BHAVAN’S**

**SARDAR PATEL INSTITUTE OF TECHNOLOGY**

**Munshi nagar, Andheri (W), Mumbai - 400058**

**(Autonomous Institute Affiliated to University of Mumbai)**

**[Department of Computer Science and Engineering (Master of Computer Applications)]**

**CLASS: F.Y. MCA SEM: I AY:24-25**

**COURSE CODE: MC504 SUBJECT NAME: WEB TECHNOLOGY LAB**

**ROLL NO.: 2024510001 BATCH: A**

**NAME: Atharva Vasant Angre**

**EXPERIMENT NO: 06**

**EXPERIMENT TITLE:** Create a React application with a user-friendly form that includes validation, submission, and reset functionalities.

**Problem Statement**: Create a React application with a user-friendly form that includes validation, submission, and reset functionalities. The form should capture three fields: Username, Email, and Password. The application should display error messages if any field is invalid or left empty. Additionally, the form should notify the user upon successful submission. Reset functionality should clear all fields and error messages.

**Requirements**:

**Form Fields**:

1. Username: This field is required.

2. Email: This field is required and must follow a valid email format.

3. Password: This field is required and must be at least 6 characters long and must be a strong password as a combination of uppercase &lowercase letters, digits, special symbols etc.

**Validation**:

1. Each field must be validated before the form can be submitted.

2. If a field is invalid, an appropriate error message should be displayed below the respective field.

**Form Submission**:

1. Upon clicking the Submit button, the form should validate all fields.

2. If there are no errors, the form data should be "submitted" (logged to the console in this case) and a success message should be displayed.

3. If any field has an error, the form should display the error messages without submitting.

**Form Reset**:

1. The Reset button should clear all form fields, reset error messages, and remove any success message.

**Success Message**:

1. When the form is submitted successfully (with no errors), display success message at the top of the form.

**Example User Flow**

1. Initial State: The form is empty, with no error messages displayed.

2. Input and Validation:

1. The user types a username but leaves the email field empty.

2. On submission, an error message displays below the email field stating, "Email is required."

3. The user enters an invalid email format, and an error message shows "Email address is invalid."

4. The user enters a valid email but leaves the password field empty. Anerror displays, "Password is required."

3. Successful Submission:

1. The user enters valid data for all fields and clicks Submit.

2. A success message is displayed, and the form data is logged in the console.

4. Reset:

1. The user clicks Reset, and the form fields, error messages, and success

message are cleared.

**Constraints**

* Implement validation checks to ensure the form is not submitted with empty or invalid fields.
* Use React hooks (useState) to manage form data and validation state.
* Provide a responsive layout and clear error messages to enhance usability.

This form will allow users to interact in a straightforward way, ensuring data accuracy through validation while providing feedback on their actions.

**Code**:

App.jsx

import React from "react";

import Form from "./form";

function App() {

  return (

    <div className="App">

      <Form />

    </div>

  );

}

export default App;

-----------------------------------------

ToDoApp.jsx

import React, { useState } from "react";

import "./form.css";

const Form = () => {

  const [formData, setFormData] = useState({

    username: "",

    email: "",

    password: "",

  });

  const [errors, setErrors] = useState({});

  const [successMessage, setSuccessMessage] = useState("");

  // Validate email format

  const validateEmail = (email) => {

    const emailRegex = /^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$/;

    return emailRegex.test(email);

  };

  // Validate password strength

  const validatePassword = (password) => {

    const strongPasswordRegex =

      /^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[@$!%\*?&])[A-Za-z\d@$!%\*?&]{6,}$/;

    return strongPasswordRegex.test(password);

  };

  const handleChange = (e) => {

    setFormData({

      ...formData,

      [e.target.name]: e.target.value,

    });

    setErrors({

      ...errors,

      [e.target.name]: "",

    });

    setSuccessMessage("");

  };

  const handleSubmit = (e) => {

    e.preventDefault();

    let validationErrors = {};

    if (!formData.username) {

      validationErrors.username = "Username is required.";

    }

    if (!formData.email) {

      validationErrors.email = "Email is required.";

    } else if (!validateEmail(formData.email)) {

      validationErrors.email = "Email address is invalid.";

    }

    if (!formData.password) {

      validationErrors.password = "Password is required.";

    } else if (!validatePassword(formData.password)) {

      validationErrors.password =

        "Password must be at least 6 characters long, with uppercase, lowercase, digit, and special symbol.";

    }

    if (Object.keys(validationErrors).length === 0) {

      console.log("Form Data Submitted:", formData);

      setSuccessMessage("Form submitted successfully!");

      setFormData({ username: "", email: "", password: "" });

    } else {

      setErrors(validationErrors);

    }

  };

  const handleReset = () => {

    setFormData({ username: "", email: "", password: "" });

    setErrors({});

    setSuccessMessage("");

  };

  return (

    <div className="form-container">

      <h2>User Registration Form</h2>

      {successMessage && <p className="success-message">{successMessage}</p>}

      <form onSubmit={handleSubmit}>

        <div className="form-field">

          <label>Username</label>

          <input

            type="text"

            name="username"

            value={formData.username}

            onChange={handleChange}

          />

          {errors.username && <span className="error">{errors.username}</span>}

        </div>

        <div className="form-field">

          <label>Email</label>

          <input

            type="text"

            name="email"

            value={formData.email}

            onChange={handleChange}

          />

          {errors.email && <span className="error">{errors.email}</span>}

        </div>

        <div className="form-field">

          <label>Password</label>

          <input

            type="password"

            name="password"

            value={formData.password}

            onChange={handleChange}

          />

          {errors.password && <span className="error">{errors.password}</span>}

        </div>

        <div style={{ textAlign: "center" }}>

          <button type="submit">Submit</button>

          <button type="button" onClick={handleReset}>

            Reset

          </button>

        </div>

      </form>

    </div>

  );

};

export default Form;

--------------------------------------------------

App.css

.form-container {

  max-width: 600px;

  margin: 20 auto;

  padding: 50px;

  border: 1px solid #ccc;

  border-radius: 5px;

  position: absolute;

  top: 50%;

  left: 50%;

  transform: translate(-50%, -50%);

}

.form-container h2 {

  text-align: center;

  margin-bottom: 20px;

}

.form-field {

  margin-bottom: 15px;

}

.form-field label {

  display: block;

  margin-bottom: 5px;

  font-weight: bold;

}

.form-field input {

  width: 100%;

  padding: 8px;

  border: 1px solid #ccc;

  border-radius: 4px;

}

button {

  padding: 10px 15px;

  margin-right: 10px;

  border: none;

  background-color: rgb(61, 183, 61);

  color: #fff;

  cursor: pointer;

  border-radius: 4px;

}

button[type="button"] {

  background-color: #6c757d;

}

.error {

  color: red;

  font-size: 0.9em;

}

.success-message {

  background: rgb(61, 183, 61);

  height: 30px;

  border-radius: 5px;

  text-align: center;

  place-content: center;

  font-size: 1em;

  text-align: center;

  margin-bottom: 10px;

}

Outputs: 







