**Practical: 10**

**Aim:** Working with forms in React.

**Hardware Requirement:**

1. **Operating System:** Microsoft Windows 10 pro
2. **Processor:** 11th Gen Intel Core i5-1135G7 @2.40GHz, 2419 MHz, 4Core
3. **RAM:** 8GB DDR4
4. **Storage:** 256GB SSD 1TB HDD

**Software Requirement:**

1. **Download Link:** <https://nodejs.org/en/download>
2. **React.js:** React is a JavaScript library for building user interfaces. You'll need to set up a React project.
3. **Form Components:** Create React components to represent different parts of your form, such as input fields, buttons, and error messages.
4. **State Management:** Utilize React state to manage form data and validation state.
5. **Validation Library or Custom Validation Logic:** Choose a validation library like Formik or Yup, or implement custom validation logic to ensure data integrity.
6. **Styling:** Optionally, you may want to include CSS or a UI framework like Bootstrap for styling your form components.

**Knowledge Requirement:**

1. **React.js Fundamentals:** Understanding of React components, props, state, and lifecycle methods.
2. **HTML Forms:** Familiarity with HTML form elements and attributes like <input>, <select>, <textarea>, and form submission.
3. **JavaScript (ES6+):** Knowledge of modern JavaScript concepts such as arrow functions, destructuring, and object literals.
4. **Form Validation Techniques:** Understanding of form validation techniques including client-side and server-side validation.
5. **React Hooks (optional):** If using functional components, knowledge of React hooks like useState and useEffect can be helpful.

**Theory:**

* **Form Elements:**
* **Username:** Allows users to input their desired username with specified length and character restrictions.
* **Email:** Collects user email addresses with enforced email format validation.
* **Password:** Used for account security, validated for minimum length and password strength.
* **Confirm Password:** Ensures accuracy by asking users to re-enter their password, matching the previous input.
* **Validation:**
* **Password Length:** Ensure minimum length requirements are met.
* **Password Strength:** Evaluate the presence of uppercase, lowercase letters, numbers, and special characters.
* **Email Format:** Validate email addresses to ensure they follow the correct format.
* **Error Messaging:** Display appropriate error messages to guide users in correcting their input before submission.

**App.js**

import React, { useState } from 'react';

import './App.css';

function FormValidation() {

  const [formData, setFormData] = useState({

    username: '',

    email: '',

    password: '',

    confirmPassword: '',

  });

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

  const [submissionStatus, setSubmissionStatus] = useState(null);

  const [passwordStrength, setPasswordStrength] = useState('');

  const validateForm = () => {

    let errors = {};

    let isValid = true;

    if (!formData.username.trim()) {

      errors.username = 'Username is required';

      isValid = false;

    }

    if (!formData.email.trim()) {

      errors.email = 'Email is required';

      isValid = false;

    } else if (!/\S+@\S+\.\S+/.test(formData.email)) {

      errors.email = 'Email address is invalid';

      isValid = false;

    }

    if (!formData.password.trim()) {

      errors.password = 'Password is required';

      isValid = false;

    } else if (formData.password.length < 6) {

      errors.password = 'Password must be at least 6 characters';

      isValid = false;

    } else if (!/(?=.\*[!@#$%^&\*])/.test(formData.password)) {

      errors.password = 'Password must contain at least one special character';

      isValid = false;

    }

    if (formData.password !== formData.confirmPassword) {

      errors.confirmPassword = 'Passwords do not match';

      isValid = false;

    }

    setErrors(errors);

    return isValid;

  };

  const getPasswordStrength = (password) => {

    if (password.length < 6) {

      return 'Weak';

    } else if (!/(?=.\*[a-z])/.test(password) || !/(?=.\*[A-Z])/.test(password) || !/(?=.\*\d)/.test(password)) {

      return 'Medium';

    } else {

      return 'Strong';

    }

  };

  const handleSubmit = (e) => {

    e.preventDefault();

    if (validateForm()) {

      console.log('Form is valid, submitting...');

      setSubmissionStatus('success');

    } else {

      console.log('Form has errors, cannot submit');

      setSubmissionStatus('error');

    }

  };

  const handleChange = (e) => {

    const { name, value } = e.target;

    if (name === 'password') {

      setPasswordStrength(getPasswordStrength(value));

    }

    setFormData({

      ...formData,

      [name]: value,

    });

  };

  return (

    <div className="background">

      <div className="form-container">

        <h1>Form Validation</h1>

        <form onSubmit={handleSubmit}>

          <div>

            <label>Username</label>

            <input

              type="text"

              name="username"

              value={formData.username}

              onChange={handleChange}

            />

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

          </div>

          <div>

            <label>Email</label>

            <input

              type="email"

              name="email"

              value={formData.email}

              onChange={handleChange}

            />

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

          </div>

          <div>

            <label>Password</label>

            <input

              type="password"

              name="password"

              value={formData.password}

              onChange={handleChange}

            />

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

            {passwordStrength && <p className={`password-strength ${passwordStrength.toLowerCase()}`}>Password Strength: {passwordStrength}</p>}

          </div>

          <div>

            <label>Confirm Password</label>

            <input

              type="password"

              name="confirmPassword"

              value={formData.confirmPassword}

              onChange={handleChange}

            />

            {errors.confirmPassword && <span className="error-message">{errors.confirmPassword}</span>}

          </div>

          <button type="submit" className={`submit-button ${submissionStatus === 'success' ? 'success-button' : submissionStatus === 'error' ? 'error-button' : ''}`}>

            Submit

          </button>

        </form>

        {submissionStatus === 'success' && <p className="success-message">Form submitted successfully!</p>}

        {submissionStatus === 'error' && <p className="error-message">Form submission failed. Please check the form.</p>}

      </div>

    </div>

  );

}

export default FormValidation;

**App.css**

.form-container {

  max-width: 400px;

  margin: 60px auto;

  padding: 50px;

  border: 1px solid #ccc;

  border-radius: 5px;

  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

  background-color: #e5e5e5;

}

.form-container h1 {

  margin-bottom: 20px;

}

.form-container label {

  display: block;

  margin-bottom: 5px;

}

.form-container input[type="text"],

.form-container input[type="email"],

.form-container input[type="password"] {

  width: 100%;

  padding: 10px;

  margin-bottom: 15px;

  border: 1px solid #ccc;

  border-radius: 5px;

}

.form-container .submit-button {

  display: block;

  margin: 0 auto;

  width: 100px;

  padding: 10px;

  background-color: #282828;

  color: #fff;

  border: none;

  border-radius: 5px;

  cursor: pointer;

}

.form-container .submit-button:hover {

  background-color: #c0bbbb;

  color: black;

}

.error-message {

  color: red;

  margin-top: 5px;

  text-align: center;

}

.success-message {

  color: green;

  margin-top: 5px;

  text-align: center;

}

.password-strength {

  margin-top: 5px;

  font-size: 0.9em;

}

.weak {

  color: red;

}

.medium {

  color: orange;

}

.strong {

  color: green;

}

.success-button {

  background-color: green;

  color: white;

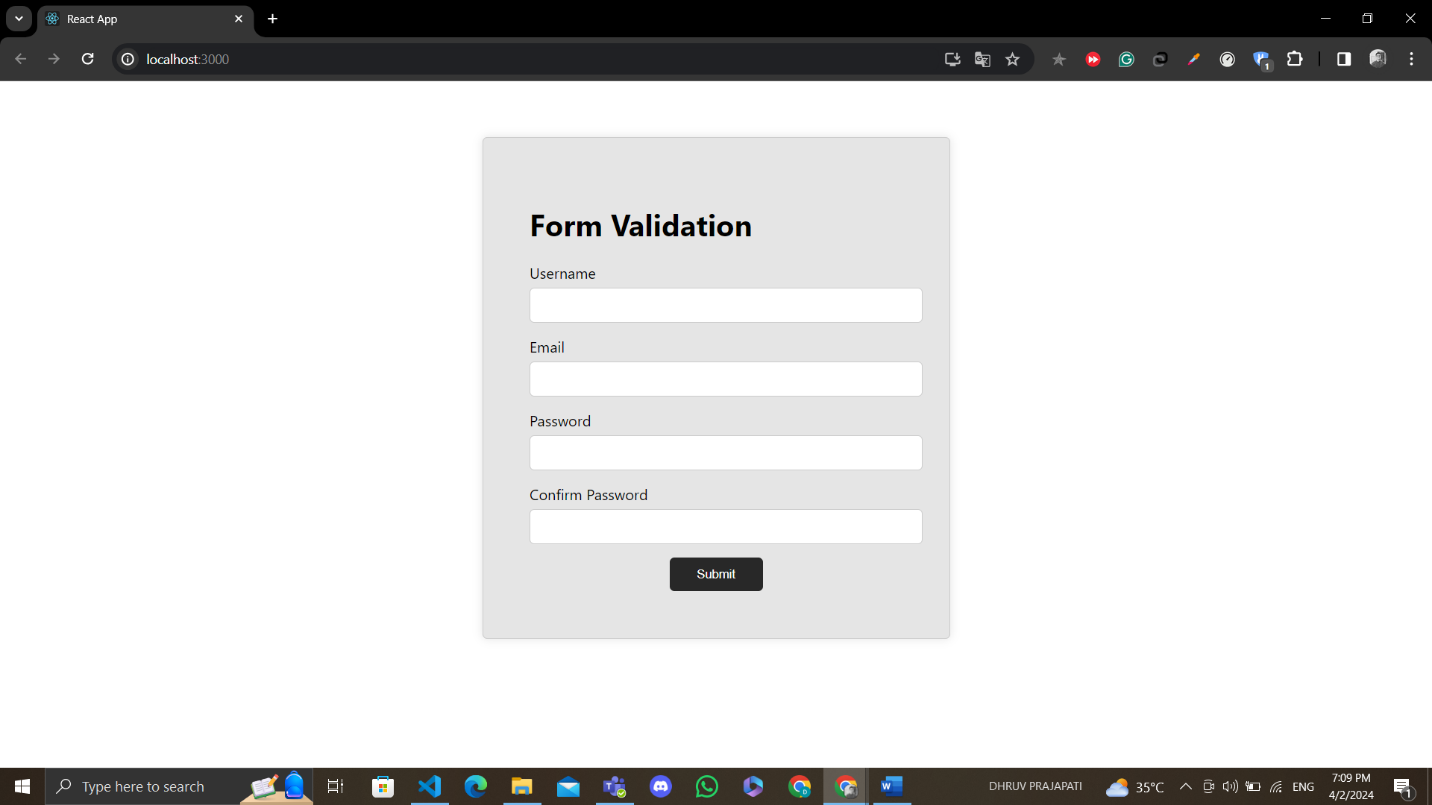
}

.error-button {

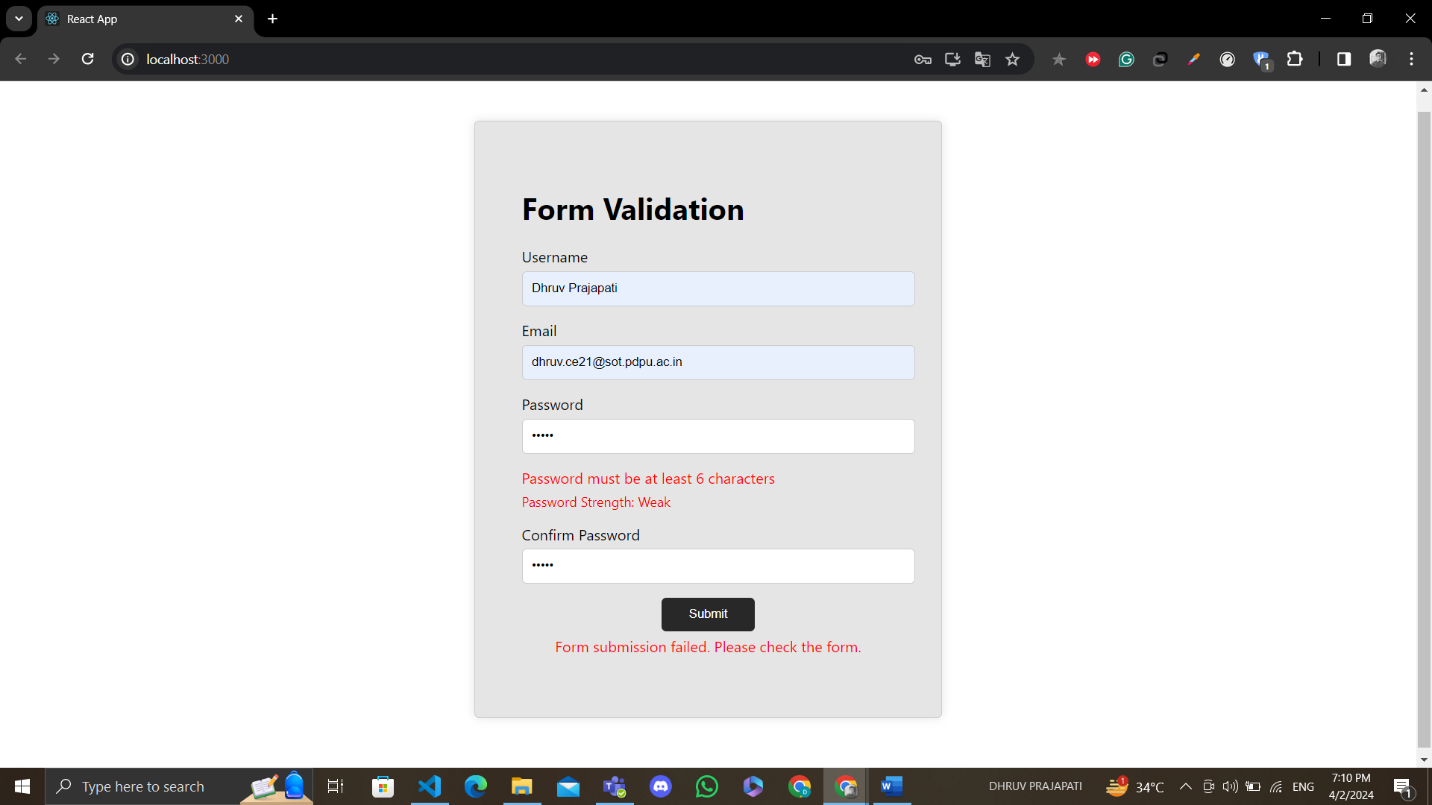
  background-color: red;

  color: white;

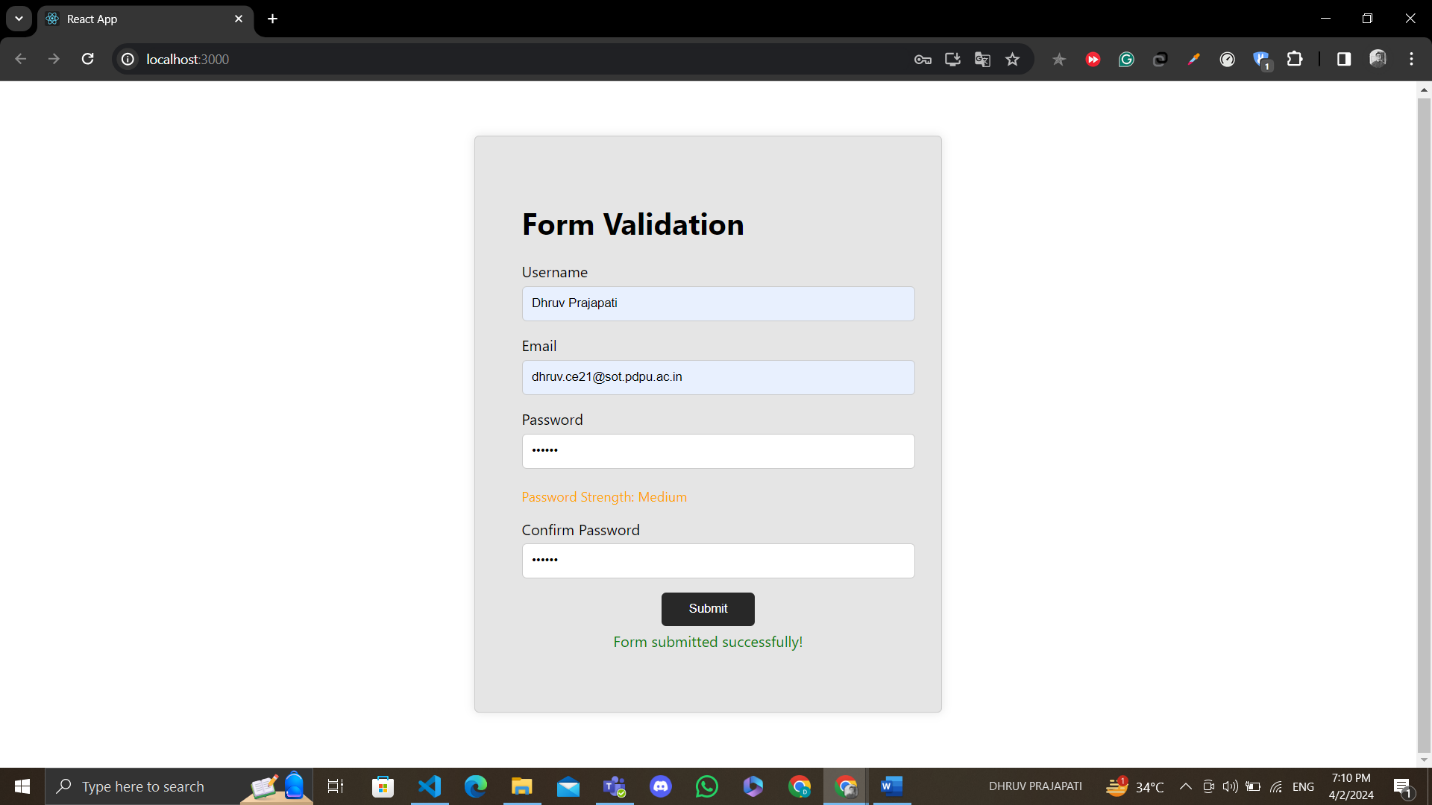
}



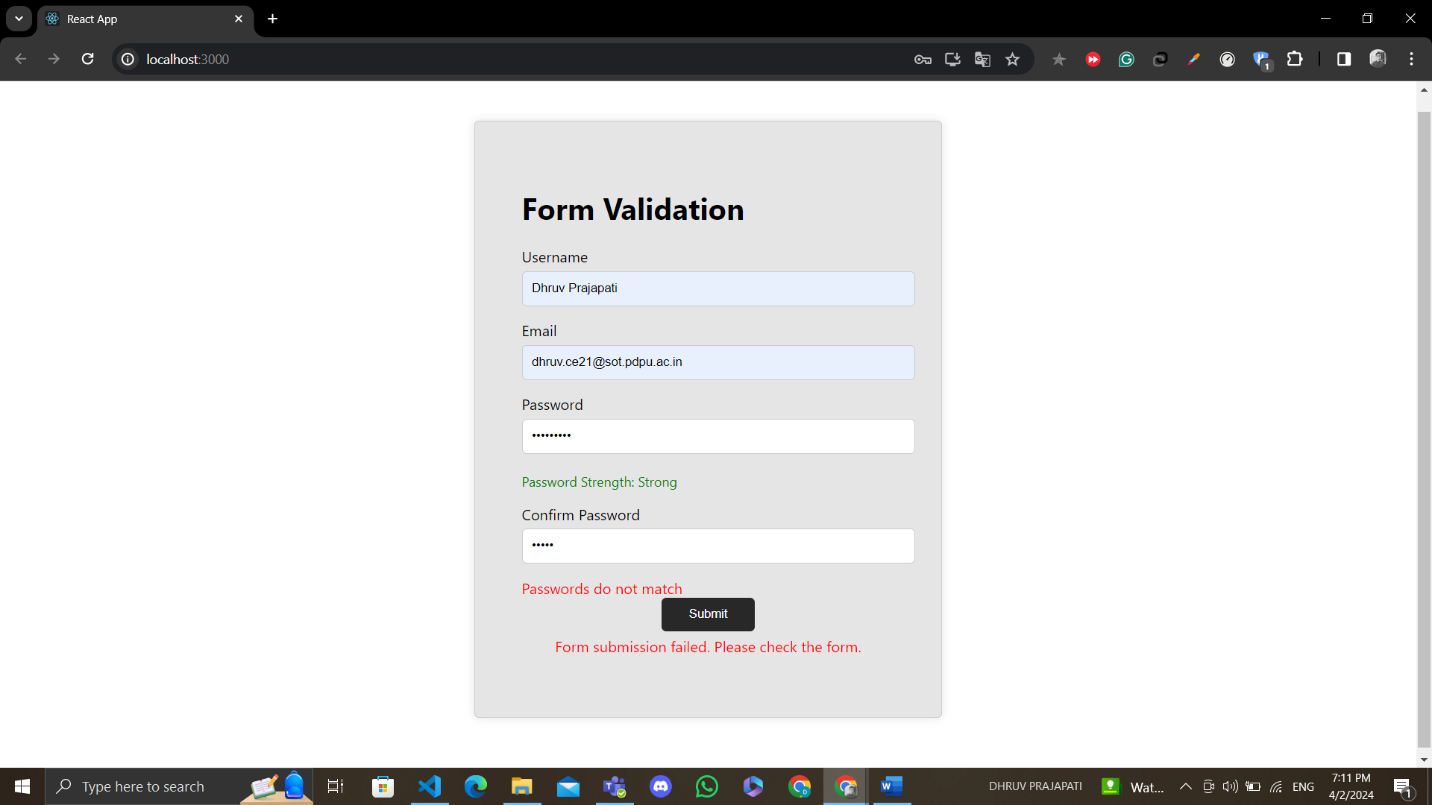
**Password must be at least 6 character.**



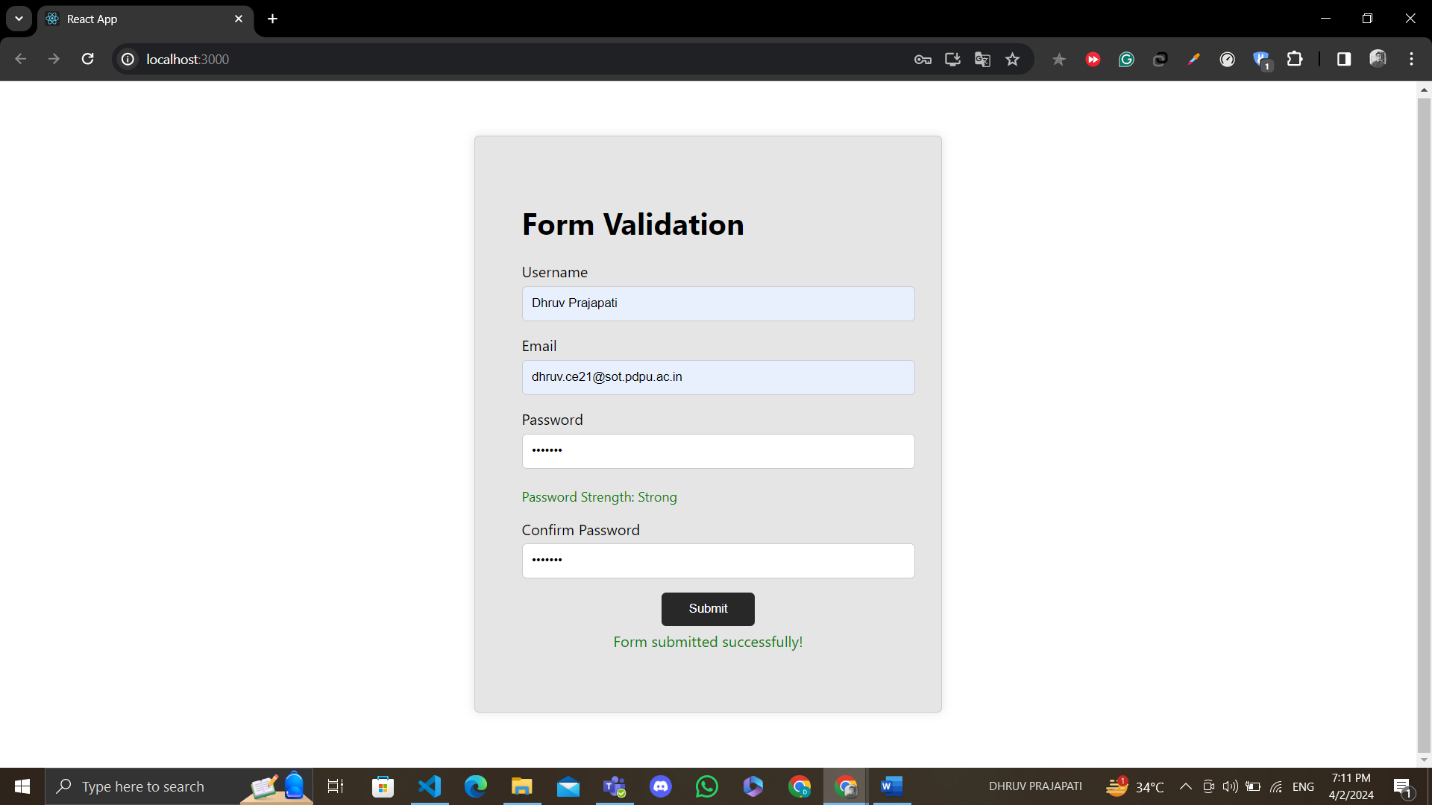
**Password is correct but password strength is medium.**



**Password strength is strong but password did not match.**



**Form validation successfully.**



**Conclusion:**

In conclusion, working with forms in React involves creating React components to represent form elements, managing their state, and implementing form validation to ensure data integrity. By leveraging React's component-based architecture and state management capabilities, developers can build interactive and user-friendly forms for collecting and validating user input. Whether using libraries like Formik and Yup or implementing custom validation logic, the goal is to create a seamless and intuitive user experience while ensuring data consistency and accuracy.

**References:**

<https://react.dev/learn>

<https://www.w3schools.com/REACT/DEFAULT.ASP>