COLLEGE CODE : 9527

COLLEGE NAME : SARDAR RAJA COLLEGE OF

ENGINEERING DEPARTMENT : CSE

STUDENT NM-ID : au952723104005

ROLL NO : 952723104005

DATE : 17-10-2025

Project names as

INTERACTIVE FORM VALIDATION

Submitted by,

A. ANUMERCY

9843750241

Phase 5 — Project Demonstration & Documentation

1. Final Demo Walkthrough

The final demo demonstrates the functionality and user interaction of the **Interactive Form Validation** system.
The demo covers:

- **Form Design:** Interactive form with fields such as Name, Email, Password, Confirm Password, and Phone Number.
- **Real-Time Validation:**
    - ○ Email validation (format check using regex).
    - ○ Password strength indicator (checks for length, uppercase, digits, and special characters).
    - ○ Confirm password match verification.
    - ○ Phone number pattern validation.
    - ○ Mandatory field detection with visual feedback (red border and error message).
- **Dynamic Feedback:** Instant error/success messages below each field.
- **Form Submission:** Submit button becomes active only when all validations pass.
- **Responsive UI:** Works across devices (desktop, tablet, mobile).

🧿 **Demo Flow:**

1. User opens the form.
2. Inputs are validated as the user types.
3. Error messages and icons appear dynamically.
4. When all fields are valid, the submit button activates.
5. On submission, a success message appears or data is sent to the backend (if applicable).

---

2. Project Report

Project Title: Interactive Form Validation **Objective:**
To design an interactive web form with real-time client-side validation to improve data accuracy and user experience.

Technologies Used:

- Frontend: HTML5, CSS3, JavaScript (ES6)
- Optional Backend: Node.js / Django (if used for data storage)
- Tools: VS Code, GitHub Pages (deployment)

Modules:

1. Input Field Components

2. Validation Logic (Regex-based)
3. UI Feedback (Color, Icons, Messages)
4. Submission Handling

Outcomes:

- Enhanced user experience through interactive validation.
- Reduction in incorrect form submissions.
- Reusable validation logic for multiple forms.

Source Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Interactive Form Validation</title>
<style>     body {       fontfamily:
Arial, sans-serif;       max-width:
```

```
600px;       margin: 20px auto;
padding: 20px;       border: 1px solid
#ccc;
   }       label {
display: block;
margin-top: 15px;
   }
      input, select
{       width:
100%;
padding: 8px;
margin-top: 5px;

   }

   .error {       color:
red;       font-size:
0.9em;
   }
```

```
    .success {       color: green;
font-size:
1em;       margin-top:
15px;
    }

    button {       margintop:
20px;       padding:
10px 20px;
    }
  </style>
</head>
<body>    <h2>Registration Form</h2>
  <form id="registrationForm" novalidate>

    <label>Full Name:
      <input type="text" id="name" name="name" />   <span class="error"
id="nameError"></span>
```

```
</label>

<label>Email:
  <input type="email" id="email" name="email" />
  <span class="error" id="emailError"></span>
</label>

<label>Password:
  <input type="password" id="password" name="password" />
  <span class="error" id="passwordError"></span>
</label>

<label>Confirm Password:
  <input type="password" id="confirmPassword" name="confirmPassword" />
  <span class="error" id="confirmPasswordError"></span>
</label>

<label>Phone Number:
  <input type="text" id="phone" name="phone" />
```

```
  <span class="error" id="phoneError"></span>
</label>

<label>Age:
  <input type="number" id="age" name="age" />
  <span class="error" id="ageError"></span>
</label>

<label>Gender:
  <input type="radio" name="gender" value="male"> Male
  <input type="radio" name="gender" value="female"> Female
  <input type="radio" name="gender" value="other"> Other
  <span class="error" id="genderError"></span>
</label>

<label>Country:
  <select id="country" name="country">
    <option value="">-- Select Country --</option>
```

```
    <option value="us">United States</option>
    <option value="uk">United Kingdom</option>
    <option value="in">India</option>
    <option value="other">Other</option>
  </select>
  <span class="error" id="countryError"></span>
 </label>

 <label>
  <input type="checkbox" id="terms" name="terms" />
  I agree to the terms and conditions
  <span class="error" id="termsError"></span>
 </label>

 <button type="submit">Submit</button>
 <div class="success" id="successMessage"></div>     </form>

<script>     const form = document.getElementById('registrationForm');
```

```
    form.addEventListener('submit', function (e) {
      e.preventDefault(); // Prevent actual submission
let isValid = validateForm();       if
(isValid) {
        document.getElementById('successMessage').textContent = "Form submitted
successfully!";         form.reset();
      } else {
document.getElementById('successMessage').textContent = "";
      }
    });

    function validateForm() {       let
isValid = true;

      // Helper functions       const setError = (id, message) =>
{         document.getElementById(id).textContent = message;
        isValid = false;
      };
```

```
    const clearError = (id) => {         document.getElementById(id).textContent
= ""; };


    // Full Name       const name =
document.getElementById('name').value.trim();  if (!name) {
setError('nameError', 'Name is required.');       } else if (!/^[a-zAZ\s]+$/.test(name))
{         setError('nameError', 'Name must contain only
letters.');
} else {
clearError('nameError');
    }

    // Email       const email = document.getElementById('email').value.trim();
if (!email) {         setError('emailError', 'Email is required.');       } else if (!/^[\w.-
]+@[\w.-]+\.[azAZ]{2,}$/.test(email)) {         setError('emailError',
'Invalid email format.');
```

```
    } else {
clearError('emailError');
    }

    // Password       const password =
document.getElementById('password').value;
if (!password) {        setError('passwordError',
'Password is required.');
    } else if (password.length < 8 || !/[A-Z]/.test(password) ||
            !/[a-z]/.test(password) || !/[0-9]/.test(password) ||
            !/[!@#$%^&*]/.test(password)) {
      setError('passwordError', 'Password must be 8+ chars with uppercase,  lowercase,
number, and symbol.');
    } else {
clearError('passwordError');
    }
```

```
    // Confirm Password       const confirmPassword =
document.getElementById('confirmPassword').value; if (!confirmPassword) {
setError('confirmPasswordError', 'Please confirm your password.');
    } else if (password !== confirmPassword) {        setError('confirmPasswordError',
'Passwords do not match.');  } else {

clearError('confirmPasswordError');
    }


    // Phone       const phone =
document.getElementById('phone').value.trim();       if
(!phone) {        setError('phoneError', 'Phone number
is required.');
    } else if (!/^\d{10}$/.test(phone)) {        setError('phoneError',
'Enter a valid 10-digit phone number.');
    } else {
clearError('phoneError');
    }
```

```
    // Age       const age = parseInt(document.getElementById('age').value);
if (!age) {         setError('ageError', 'Age is required.');
} else if (age < 18 || age > 100) {         setError('ageError',
'Age must be between 18 and 100.');
    } else {         clearError('ageError');
    }

    // Gender       const gender =
form.querySelector('input[name="gender"]:checked');        if
(!gender) {         setError('genderError', 'Please select your
gender.');
    } else {
clearError('genderError');
    }

    // Country       const country = document.getElementById('country').value;
    if (!country) {         setError('countryError', 'Please select
a country.');
    } else {
```

```
clearError('countryError');
      }

      // Terms       const terms =
document.getElementById('terms').checked;         if (!terms) {
setError('termsError', 'You must agree to the terms.');         } else {
clearError('termsError');
      }

      return isValid;
    }
  </script>

</body>
</html>
```

---

3. API Documentation

Include the following:

1. Form initial view (empty).
2. Invalid email input example.
3. Weak password feedback.
4. Passwords do not match warning.
5. All valid fields with active Submit button.
6. Successful submission message.

---

4. Challenges & Solutions

Challenge	Solution

Handling multiple validations simultaneously	Implemented modular JavaScript functions for each field
Providing real-time feedback without page reload	Used event listeners (`input`, `keyup`, `blur`) for live validation
Ensuring mobile responsiveness	Used CSS Flexbox and media queries
Managing password strength detection	Created regex-based scoring system for strength indicator
Preventing form submission on invalid data	Disabled submit button until all checks passed

---

5. GitHub README & Setup

https://github.com/mercy2006anu/Interactive-form-validation-.git

Validation Repository **README Includes:**

- Project overview

- Features list
- Screenshots
- Technologies used
- How to run the project
- Author information

Output:

Registration Form

Full Name:

324

Name must contain only letters.

Email:

wer

Invalid email format.

Password:

•••

Password must be 8+ chars with uppercase, lowercase, number, and symbol.

Confirm Password:

•••

Phone Number:

were

Enter a valid 10-digit phone number.

Age:

e

Age is required.

Gender:

Male

Female

Other Please select your gender.

Country:

-- Select Country --

Please select a country.

I agree to the terms and conditions You must agree to the terms.

Submit

6. Final Submission

- GitHub **Repository:**

https://github.com/mercy2006anu/Interactive-form-validation-

- **Deployed Link:**

https://mercy2006anu.github.io/Interactive-form-validation-/