



**9530**

**ST.MOTHER THERESA ENGINEERING COLLEGE**

**COMPUTER SCIENCE ENGINEERING**

**NM-ID:1CDC1B6BC979C8786FE491EC7E908042**

**REG NO:953023104057**

**DATE:14-09-2025**

**Completed the project named as**

**Phase-2**

**INTERACTIVE FORM VALIDATION TOOL**

**SUBMITTED BY,**

**M.MARI ANU**

**PH NO:6374153920**

# TITLE:INTERACTIVE FORM VALIDATION TOOLS

## :HTML,CSS,JS FUNCTIONALITY

### Problem statement:

In many web applications, user input is collected through online forms for activities such as account creation, login, and data submission. However, users often provide incomplete, incorrect, or weak input, such as invalid email addresses, empty fields, or insecure passwords. This leads to : Data inaccuracy in the system.

Poor user experience due to delayed error detection after submission.

Security risks from weak password choices.

Increased server load, as invalid data must be handled at the backend.

Therefore, there is a need for an interactive form validation mechanism that checks user inputs in real-time, provides instant feedback, and prevents the submission of invalid data before reaching the server

### Objective:

The objective of this project is to design and implement an interactive form validation system that ensures user input accuracy and enhances usability. The system validates form fields such as name, email, and password in real-time using JavaScript and regular expressions. By providing immediate feedback through dynamic error messages and visual indicators (CSS styling), the project aims to: Improve data accuracy and prevent invalid submissions.Enhance user experience with instant feedback. Ensure strong security practices by enforcing password strength. Demonstrate the integration of HTML, CSS, and JavaScript for frontend validation. Reduce backend load by filtering invalid data at the client side.

### AIM 1:

To develop an interactive form that validates user inputs such as name, email, and password in real-time using JavaScript and Regular Expressions.

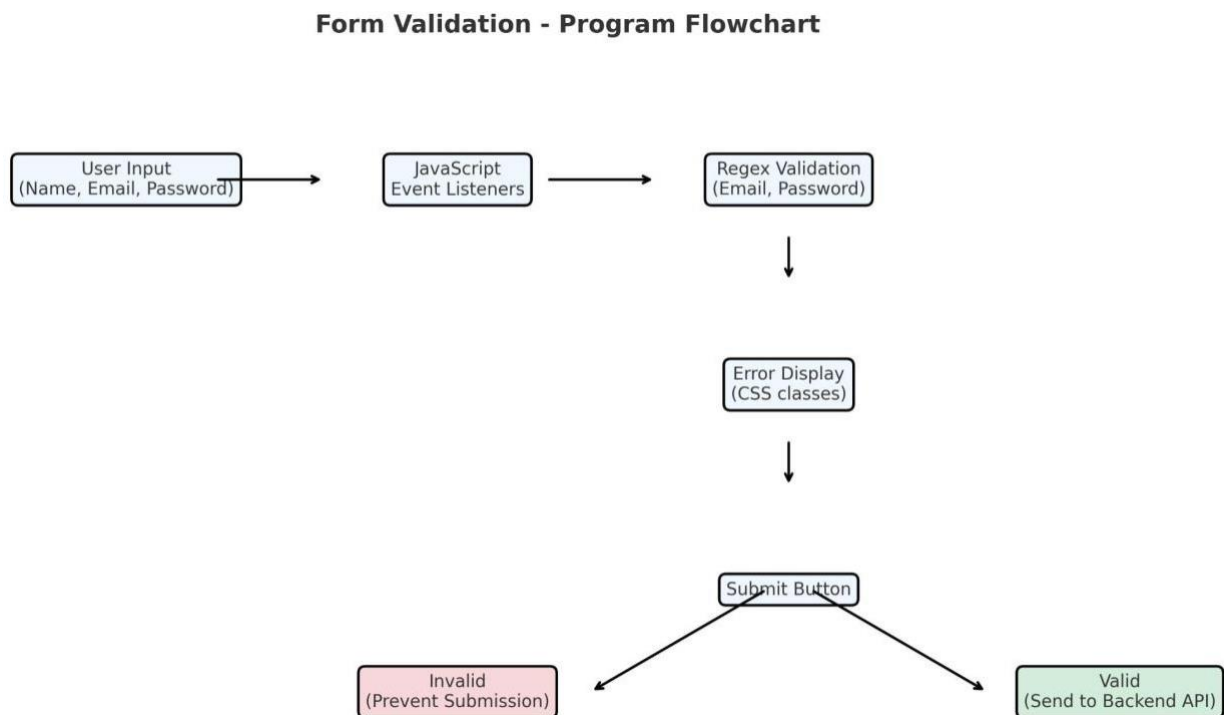
### AIM 2:

To enhance user experience and data accuracy by providing instant feedback through dynamic CSS styling and preventing submission of invalid data.

DATE : 14/06/2025

To create an interactive web form that validates user inputs in real-time using HTML, CSS, and JavaScript. To prevent invalid data submission and enhance user experience with dynamic error feedback.

Flowchart:



# PROGRAM

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<title>Interactive Form Validation</title>
```

**<style>**

**Body {**

**Font-family: Arial, sans-serif;**

**Margin: 50px;**

**}**

**Form {**

**Max-width: 400px;**

**}**

**Input {**

**Display: block;**

**Width: 100%;**

**Padding: 10px;**

**Margin-bottom: 10px;**

**Border: 1px solid #ccc;**

**Border-radius: 5px;**

**}**

**Input.invalid {**

**Border-color: red;**

**}**

**.error {**

**Color: red;**

**Font-size: 0.3em;**

**Display: none;**

**}**

**Button {**

**Padding: 10px 20px;**

```
    Background-color: #4CAF50;
    Color: white;
    Border: none;
    Border-radius: 5px;
    Cursor: pointer;
}
Button:hover {
    Background-color: #45a043;
}
</style>
</head>
<body>

<h2>Sign Up Form</h2>
<form id="signupForm" novalidate>
    <label for="name">Name:</label>
    <input type="text" id="name" name="name" required>
    <div class="error" id="nameError">Name is required.</div>

    <label for="email">Email:</label>
    <input type="email" id="email" name="email" required>
    <div class="error" id="emailError">Enter a valid email.</div>

    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required>
```

```
<div class="error" id="passwordError">Password must be at least C
characters.</div>
```

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

```
<script>
```

```
  Const form = document.getElementById('signupForm');
```

```
  Const nameInput = document.getElementById('name');
```

```
  Const emailInput = document.getElementById('email');
```

```
  Const passwordInput = document.getElementById('password');
```

```
  Const nameError = document.getElementById('nameError');
```

```
  Const emailError = document.getElementById('emailError');
```

```
  Const passwordError = document.getElementById('passwordError');
```

```
  // Real-time validation
```

```
  nameInput.addEventListener('input', () => {
```

```
    if (nameInput.value.trim() === '') {
```

```
      nameInput.classList.add('invalid');
```

```
      nameError.style.display = 'block';
```

```
    } else {
```

```
      nameInput.classList.remove('invalid');
```

```
      nameError.style.display = 'none';
```

```
    }
```

```
  });
```

```
emailInput.addEventListener('input', () => {  
    const emailPattern = /^[^ ]+@[^ ]+\.[a-z]{2,3}$/;  
    if (!emailInput.value.match(emailPattern)) {  
        emailInput.classList.add('invalid');  
        emailError.style.display = 'block';  
    } else {  
        emailInput.classList.remove('invalid');  
        emailError.style.display = 'none';  
    }  
});
```

```
passwordInput.addEventListener('input', () => {  
    if (passwordInput.value.length < C) {  
        passwordInput.classList.add('invalid');  
        passwordError.style.display = 'block';  
    } else {  
        passwordInput.classList.remove('invalid');  
        passwordError.style.display = 'none';  
    }  
});
```

*// Form submission validation*

```
Form.addEventListener('submit', e => {  
    Let valid = true;
```

```
    If (nameInput.value.trim() === '') {
        nameInput.classList.add('invalid');
        nameError.style.display = 'block';
        valid = false;
    }
    Const emailPattern = /^[^ ]+@[^ ]+\.[a-z]{2,3}$/;
    If (!emailInput.value.match(emailPattern)) {
        emailInput.classList.add('invalid');
        emailError.style.display = 'block';
        valid = false;
    }
    If (passwordInput.value.length < 6) {
        passwordInput.classList.add('invalid');
        passwordError.style.display = 'block';
        valid = false;
    }

    If (!valid) {
        e.preventDefault(); // Prevent form submission
    }
});
</script>

</body>
</html>
```



### Interactive Form Validation

Name:

Email:

Password:  Invalid ☐

Confirm Password:  Weak ☐

#### Project hurdles :

1. **Regex Complexity** – Writing efficient regular expressions for email format and password strength can be tricky and may reject valid inputs.
2. **Browser Compatibility** – Some validation features may behave differently across browsers.
3. **User Experience** – Showing too many error messages at once can confuse users. Proper placement and styling of errors is necessary.
4. **Security** – Client-side validation alone is not secure; data must still be validated on the server.
5. **Accessibility** – Ensuring error messages and validation hints are usable for screen readers.

5.Password Strength – Balancing strict rules (security) with user convenience (usability).

## Phase-2:solution designC architecture

### Tech Stack Selection:

- 1 Frontend: HTML5, CSS3, JavaScript (Vanilla JS for validation).
- 2 Backend (Optional for storage): Node.js with Express.js.
- 3 Database (Optional): MongoDB (for saving user accounts).
- 4 Libraries/Tools: Regex for validation, Bootstrap (optional for styling).

### UI Structure / API Schema Design

#### UI Structure:

- 1 Input fields: Name, Email, Password, Confirm Password.
- 2 Real-time error messages below each input.
- 3 Submit button with validation check.

API Schema (if backend is used):

```
{  
  "name": "string",  
  "email": "string",  
  "password": "string"  
}
```

### 3. Data Handling Approach

- 1 Input data validated on client side using JavaScript + Regex.
- 2 Errors displayed dynamically via CSS classes (red border, error text).

3 On submit:

4 – If frontend only → Prevents form submission when invalid.

5 – If backend enabled → Valid data sent to Node.js API → Stored in DB.

#### 4. Component / Module Diagram

1 Frontend Module: Handles UI, validations, and error display.

2 Validation Module: Regex-based rules for email, password strength, empty fields.

3 Backend Module (optional): Node.js REST API for data persistence.

#### Conclusion :

The interactive form validation system enhances user experience by providing real-time feedback on input fields such as name, email, and password. Using HTML, CSS, and JavaScript, the form dynamically checks for valid inputs, highlights errors, and prevents submission of invalid data. This approach reduces user errors, improves data accuracy, and strengthens overall web application reliability.