COLLEGE CODE: 9504
COLLEGE NAME: DR.G.U.POPE COLLEGE OF ENGINEERING
DEPARTMENT: CSE
STUDENT NM-ID: 02BC087296AF144A84D42EECC2343D59
ROLL NO: 950423104017
DATE: 15/09/2025

COMPLETED THE PHASE 1 INTERACTIVE FORM VALIDATION

SUBMITTED BY:
NAME: K.Jovina
MOBILE NO: 9361007290

PROBLEM STATEMENT

The project aims to develop an interactive form validation system that ensures user inputs are accurate, complete, and meet predefined criteria before form submission. The system will provide real-time, user-friendly feedback on errors and guide users to correct mistakes through clear, actionable messages. It will enhance the user experience by preventing invalid data entry, reducing form submission errors, and improving accessibility by communicating validation feedback effectively both visually and for assistive technologies.

This problem addresses the common issues where forms accept incomplete or incorrect data, causing delays, misunderstandings, and degraded user satisfaction. The solution focuses on integrating accessibility best practices, including in-line error messages and alerts for screen readers, flexible validation rules, and dynamic error handling during user input.

USERS AND STAKEHOLDERS

Users

User Type	Description	Needs and Goals
Website Visitors	Individuals completing forms for registration, login, checkout, feedback, etc.	Quick form completion, instant feedback, clarity
Mobile App Users	People using forms on tablets or smartphones	Responsive design, touch-friendly, fast feedback
Users with Accessibility Needs	Individuals using assistive technologies (screen readers, keyboard navigation, etc.)	Accessible error messages, logical form flow

Stakeholders

Stakeholder Role	Description	Interest/Responsibility
Project Owner	Sponsors or managers overseeing the project	User satisfaction, business objectives
Developers	Frontend and backend engineers implementing validation	Code quality, error handling, maintainability
UI/UX Designers	Designers crafting form and feedback visuals	User experience, intuitive interface
QA Testers	Testers ensuring robustness and usability	Find edge cases, reduce bugs
Business/Marketing	Teams tracking conversions and feedback	High completion rates, valuable user data

User Story Table

Story ID	Title	User Persona	Goal/Need	Benefit/Reason	Acceptance Criteria
US01	Real-time Field Validation	Website Visitor	See instant, clear guidance when entering form data	Avoid mistakes and submit information correctly	Validation occurs as data is typed; errors/success are promptly visible
US02	Instant Error Feedback	Registered User	Get alerts for missing/ incorrect fields without reloads	Resolve issues quickly and complete forms smoothly	User sees actionable messages for required/incorrect inputs
US03	Input Security for Forms	Administrator	Prevent invalid or malicious inputs	Keep website/ database secure and protected	Form blocks script/SQL inputs and auto-sanitizes fields
US04	Visual Clarity for Validation Messages	Designer	Have distinct, user-friendly error/status messages	Ease form completion and minimize frustration	Validation/error messages use clear icons, colors, and plain language

US05	Mobile Friendly Validation	Mobile User	Validation displays are optimized for small screens	Get seamless experience on all devices	Validation overlays and prompts adapt to mobile screen size

MVP Structure

1. Form Layout & Fields

- Use HTML <form> with fields: Text, Email, Number, Password, Checkbox, etc.[2][1]
- Each field identified with relevant name and id attributes.[2]

2. Built-in Validation Attributes

- Apply attributes directly to fields:
    - required for mandatory inputs
    - type="email" or type="number" for input formats
    - minlength, maxlength for text
    - pattern for custom rules (e.g., regex).

3. CSS Styling for Validity

- Use pseudo-classes:
    - :valid for green border or checkmark
    - :invalid for red border or error icon
- Immediate visual feedback as user interacts.

4. Real-Time Error Messaging

- Show error beside/under field if invalid (e.g., "Please enter a valid email")
- Hide error when fixed.

5. JavaScript Validation Logic

- Supplement built-in rules with custom checks:

- On submit, check every field's validity in a validateForm() function
- Block submission if any invalid.

- Example logic:

```
function validateForm() {

  // Check all required fields

  // Validate email with regex

  // Show/hide error messages

  // If all pass, allow submission

}
```

6. Submit Button Control

- Disable submit button until all inputs are valid and the form passes validation.

7. Accessibility & Usability

- Use clear label text, required indications, and ARIA attributes where needed
- Error feedback is screen-reader friendly.

Structured Workflow Table

Stage	Element/Action	Purpose	Key Technologies

Layout	HTML form and input fields	Structure for user input	HTML5
Validation Attributes	required, type, pattern	Basic constraints	HTML5 attributes
CSS Feedback	:valid, :invalid classes	Visual cues for valid/ invalid input	CSS
Error Messaging	Inline/adjacent text elements	Guide user to fix issues	HTML, JavaScript
JS Logic	validateForm() function	Custom validation, checks on submit	JavaScript
Submit Control	Enable/disable submit button	Prevent wrong submission	JS, HTML

Each stage is essential for building a robust, interactive MVP that ensures reliable, real-time form validation.

WIREFRAMES/ API ENDPOINT LIST

Here is a detailed list of Wireframes and an API Endpoint List for the Interactive Form Validation project.

Wireframes for Interactive Form Validation

Below are the recommended components and layout ideas for wireframes to implement interactive form validation effectively:

- Form Structure & Layout
  - Fields grouped logically (e.g., Personal Details, Account Info).
  - Labels above input fields for clarity.
  - Separate required and optional fields visually.
  - Submit button disabled until all validations pass.
- Inline Validation Feedback
  - Real-time error messages appear immediately after input.
  - Use clear, plain language error messages next to the relevant field.
  - Use color codes: red for errors, green for success, subtle color for info.
  - Show success icons or indicators (like checkmarks) in fields with valid input.
- Error Message Design
  - Error boxes around fields with invalid data.
  - Guidance text within the error message offering solutions.
  - Do not overburden messages; keep confirmation messages brief and subtle.
- Responsive Design
  - Layout adapts for desktop and mobile.
  - Input elements and messages rearrange for smaller screens.
- Example Wireframe Flow

- User starts typing in the Email field.
- Inline validation checks format; if invalid, immediate red message appears.
- If email is taken, specialized error message suggests login or password reset.
- Password field shows strength meter and validation rules dynamically.
- Submit button only enabled when all fields are valid.

These elements represent a user-friendly, anti-confusion validated form experience like Twitter's or Pinterest's forms.

API Endpoint List for Interactive Form Validation

Endpoint	Method	Description	Request Payload	Response
/api/forms/{form_id}/validate	POST	Validates form data server-side before final submission.	JSON object with form field values	{ is_valid: bool, validation_messages: {field: message}, field_status?: {field: status}}
/api/forms/{form_id}/submit	POST	Submits the full form data after client and server validation.	JSON object with all form data	Success confirmation or error report

- Usage
  - /validate endpoint is called often during user input for real-time server-side checks (e.g., checking if username/email is already taken).

- /submit is called once after client-side validations pass.

Together, the wireframes and APIs provide a blueprint for designing and implementing a robust interactive form validation system that balances user experience with backend verification.

ACCEPTANCE CRITERIA FOR INTERACTIVE FORM VALIDATION

- Inline validation feedback is shown immediately after user input upon field focus loss or relevant character count reached, avoiding premature validation.
- Error messages are clear, specific, non-technical, and provide actionable guidance for correction next to the respective input field.
- Validation covers all required fields, correct input formats (e.g., email, phone number), password strength, and value ranges where applicable.
- The submit button remains disabled until all inputs pass client-side validation successfully.
- The system also performs server-side validation via API, and validation errors returned are properly displayed inline in the form.
- Success feedback (e.g., green check icons or success messages) is shown when inputs are valid to encourage completion.
- Validation logic does not interrupt or frustrate users with premature error messages during typing but validates on field exit or when appropriate input length is reached.
- The form validation must be accessible, with error messages and field states announced properly for assistive technologies.
- No data is submitted unless all client- and server-side validations are cleared.

- All validations support responsive design, functioning correctly on desktop and mobile devices.
- User can easily identify and fix errors without needing technical knowledge.
- The validation system appropriately handles edge cases and unexpected input safely, preventing injection or malicious data submission.

These criteria ensure a robust, user-friendly, and secure interactive form validation experience that meets both functional and usability goals.