NM1116 – FRONT-END TECHNOLOGIES

A PROJECT REPORT

Submitted by

GOWRI K (950523104017)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Dr. SIVANTHI ADITANAR COLLEGE OF ENGINEERING

TIRUCHENDUR-628215

ANNA UNIVERSITY : CHENNAI 600 025

NOVEMBER-2025

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

This is to certify that the Naan Mudhalvan project report titled "**INTERACTIVE FORM VALIDATION**" is the bonafide work of Gowri K (Reg no: 950523104017) who has successfully carried out the project as a part of the Naan Mudhalvan program under the course "Front-end technologies", conducted in collaboration with IBM.

This project was completed under the guidance of trainer Mr.R.Kishor from IBM and was supervised by Mrs.K.Easwari, as part of the academic and practical training framework of the program.

STAFF IN CHARGE **HEAD OF THE DEPARTMENT**

The project report submitted for the viva voce held on .....

INTERNAL EXAMINER **EXTERNAL EXAMINER**

INTERACTIVE FORM VALIDATION

Project Created By : Aashika N (950523104001)

Abhinaya S (950523104003)

Gowri K (950523104017)

Project Reviewed By : Kishor R

Project Created Date : 03/November/2025

College code : 9505

Team Name : SACOECSE1

Executive Summary

This project focuses on developing an **Interactive Real-Time Form Validation System** that enhances the accuracy, accessibility, and overall user experience of online forms. Traditional web forms often lack instant feedback, leading to user frustration, higher error rates, and poor data quality. This project aims to solve these issues by implementing real-time validation, responsive design, and accessibility features that provide instant, meaningful feedback to users while typing.

The system was developed using **React.js, Node.js, HTML5, CSS3, and JavaScript** . A shared validation schema that ensures consistent rules between the frontend and backend. Core features includeinline error messages**,** apassword strength meter**,** asynchronous checks for email or username availability. The responsive, mobile-friendly design ensures smooth performance across different devices and browsers.

Strong focus was placed on **security and performance**, with input sanitization, encrypted password storage, and protection against SQL injection and XSS attacks. Enhancements such as multi-step forms**,** dark mode**,** reCAPTCHA integration, and email notifications were added to improve usability and reliability

The final system delivers a **secure, user-friendly, and scalable form validation experience** suitable for websites, student registration portals, and business applications. By integrating modern web technologies with accessibility and real-time interactivity, the project provides a robust solution that minimizes user errors, increases form completion rates, and enhances overall satisfaction.

TABLE OF CONTENTS

INTERACTIVE FORM VALIDATION

1.Problem Understanding& Requirements

1.Problem Statement:

- Web forms often lack real-time feedback, leading to higher error rates.
- Users waste time correcting mistakes only after submitting.
- Delayed validation causes frustration and poor user experience.
- Lack of accessibility makes it hard for differently-abled users.
- Inconsistent client-side and server-side rules create confusion.
- Generic error messages reduce clarity and increase support needs.
- Weak security checks allow invalid or malicious input.
- Non-responsive forms fail on mobile devices and slow networks.
- Poor validation increases form abandonment rates and reduces conversions.

2.Users & Stakeholders:

- **End Users**: People filling forms (students, customers, employees).
- **Product Owners/Instructors**: Define goals and success criteria.
- **Frontend Developers**: Build interactive validation on UI.
- **Backend Developers**: Ensure server-side validation and APIs.
- **QA/Test Engineers:** Verify validation logic and usability.
- **UX/ QA/Test Engineers Accessibility Designers**: Improve clarity and inclusivity.
- **DevOps**:Manage deployment and versioning.

3.User Stories:

- **As a new user**, I want instant validation (e.g., email format, password strength) while typing so I can correct errors early.

- **As a returning user**, I want email or username availability checked asynchronously so I don’t waste time filling a form that will be rejected.
- **As a user with accessibility needs**, I want validation messages announced clearly by screen readers so I can understand and fix issues.
- **As a user on mobile**, I want a responsive form layout so validation messages fit neatly on small screens.
- **As a developer**, I want a shared validation schema between frontend and backend so rules are consistent across platforms.
- **As a tester**, I want clear validation scenarios and test cases so I can ensure reliable functionality.
- **As a product owner**, I want meaningful and simple error messages so userscomplete forms without confusion.
- **As a security engineer**, I want strong input validation to prevent malicious entries like SQL injection or XSS.
- **As a business stakeholder**, I want fewer abandoned forms so conversion rates improve.

4.MVP Features:

- Real-time validation with inline messages.
- Password strength meter and show/hide option.
- Async checks (email, username availability).
- Shared validation rules (client + server).
- Accessibility support with ARIA roles.
- Mobile-friendly design.

5.Wireframes

  - **Signup Form**: Fields for Name, Email, Password, Confirm Password, Phone. Inline error messages below each field. Password strength meter below password field.
  - **Profile Update Form**: Editable fields for Email, Phone, and Address with inline validation.

- **Contact Form**:Fields for Name, Email, and Message with inline error and success states.
- **Field Component Layout**: Input box + validation icon (✓/✕) + error/success message area.
- **Responsive Design**: Mobile-first layout with stacked fields, larger touch targets, and clear contrast.

6.Acceptance Criteria:

- Form cannot submit with invalid data.
- Inline feedback within 300ms.
- Accessible error messages (screen reader support).
- Shared schema for consistent validation.
- Async checks return within 1 second.
- Minimum 80% test coverage for validation logic.
- Works on major devices and browsers.

2. Solution Design & Architecture

---

1. Tech Stack Selection:

- **Frontend:**
  - HTML5, CSS3, JavaScript (ES6+) for base form and validation logic.
  - React.js for modular, reusable form components with real-time validation.
- **Backend:**
  - Node.js with Express.js for secure server-side validation and API endpoints.
- **Database:**
  - MongoDB or MySQL for user records and async checks (e.g., email availability).
- **Validation Tools/Libraries:**
  - Yup / Joi for schema-based validation.
  - Regex for email, password, and phone number formats.
- **Accessibility & Testing Tools:**
  - ARIA roles for assistive technologies.
  - Jest / Mocha for automated validation testing.

2. UI Structure:

- **Signup Form:** Name, Email, Password, Confirm Password, Phone with inline validation + password strength meter.
- **Profile Update Form:** Email, Phone, Address with real-time validation.
- **Contact Form:** Name, Email, Message with inline error/success messages.
- **Field Layout:** Input + validation icon (✓ / ✕) + error message.

- **Responsive Design:** Mobile-first layout, larger touch targets, adaptive feedback messages.
- **Accessibility Features:** Screen-reader-friendly error labels and ARIA support.

3. Data Handling Approach:

- **Client-Side Validation:**
  - Immediate feedback for required fields, email format, password strength.
  - Inline messages appear within 300ms.
- **Server-Side Validation:**
  - Re-checks all inputs for security and consistency.
  - Protects against SQL injection, XSS, and malicious entries.
- **Async Checks:**
  - Email/username availability verified against database within 1 second.
- **Error Handling:**
  - Clear, user-friendly messages.
  - Standardized JSON error responses for frontend consumption.

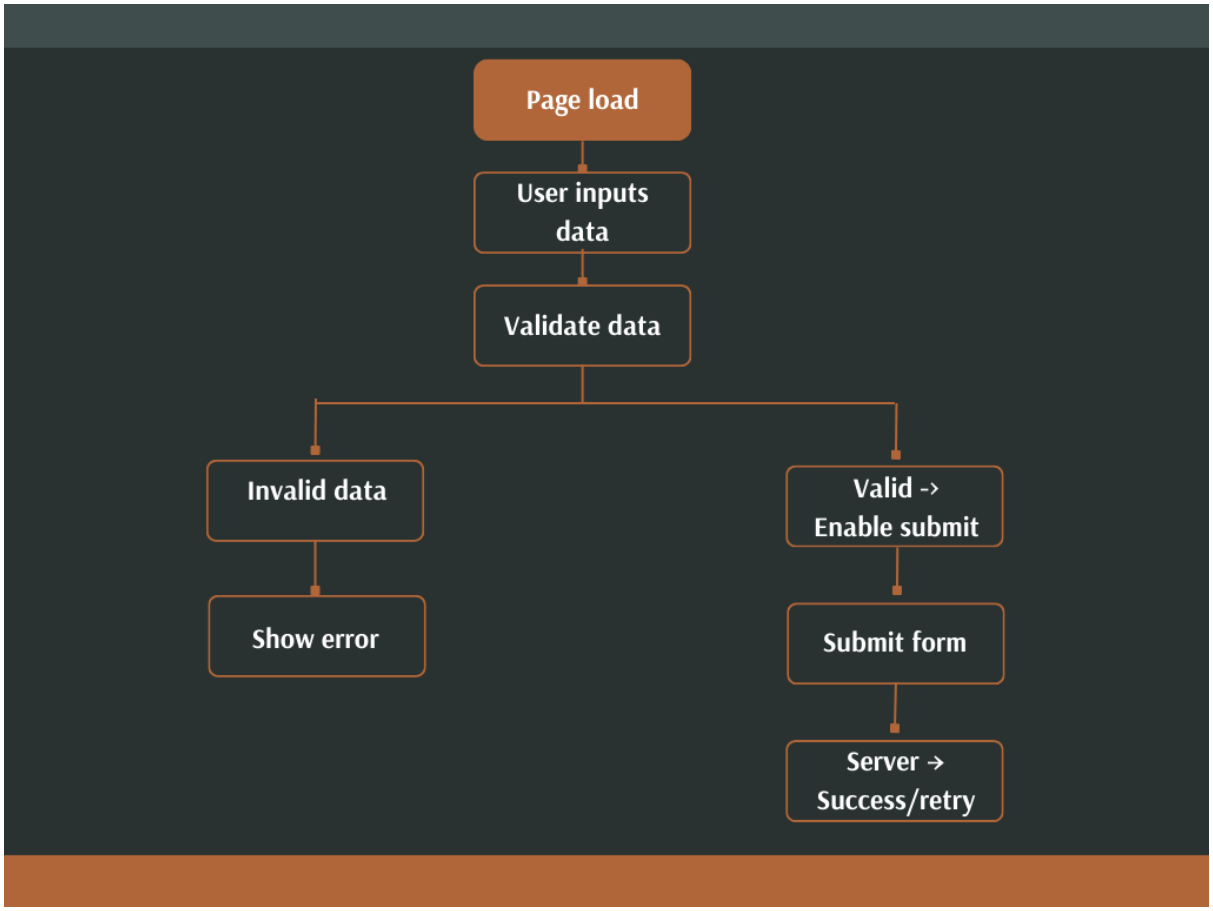
4. Component / Module Diagram:

Modules:

- **InputField Component** – encapsulates field + validation logic.
- **ValidationService** – schema-based rules, shared between frontend & backend.
- **ErrorDisplay Component** – shows inline and ARIA-friendly messages.
- **PasswordStrength Meter** – provides visual feedback.
- **Async Validator** – checks availability from backend.
- **FormController** – manages submission, reset, and API calls.

5. Basic Flow Diagram:

1. User enters input.
2. Client-side validation checks input instantly.
    - If invalid → show inline error message.
    - If valid → mark field with ✓ and proceed.
3. Async validation (username/email availability) triggered automatically.
4. Once all fields are valid, submit button enabled.
5. On submission, server-side validation re-verifies inputs.
6. If valid → data stored in database.
7. If invalid → return error response with suggestions.
8. User receives feedback and can correct mistakes immediately.



3. MVP Implementation

---

1. Project Setup

- **Step 1: Create a Project Folder**
  Make a folder named *Interactive Form Validation*.
- **Step 2: Arrange Files**
  - **Frontend** → HTML, CSS, JavaScript files for forms.
  - **Backend** → If needed, Node.js/Express files.
  - **Documents** → Reports, diagrams, and screenshots.
- **Step 3: Tools Used**
  - **VS Code** → For writing code.
  - **Browser (Chrome/Edge)** → For testing the forms.
  - **Database (MySQL/MongoDB)** → For storing user details.
  - **GitHub** → To save versions and share project.
- **Step 4: Basic Setup**
  - Create simple form pages (Signup, Profile Update, Contact).
  - Add CSS styling for a clean look.
  - Connect to a small database (optional).
  - Write simple validation rules (email format, password rules, phone number check).

2. Core Features Implementation

- **Forms Included**
  1. **Signup Form** → Name, Email, Password, Confirm Password, Phone.
  2. **Profile Update Form** → Email, Phone, Address.
  3. **Contact Form** → Name, Email, Message.

- **Main Features**
    - Real-time validation → Errors show immediately while typing.
    - Password strength meter → Shows Weak, Medium, or Strong.
    - Confirm password → Must match with password.
    - Async checks → Email or username availability checked.
    - Error & success messages → Highlight wrong fields in red, correct fields in green.
    - Submit button → Enabled only if all fields are correct.
    - Mobile-friendly → Works properly on small screens also.

3. Data Storage

- **Frontend**: Data first stored temporarily in the form (local state).
- **Backend**: If used, details are stored in a database like MySQL or MongoDB.
- **What is Stored**: Name, Email, Password, Phone, and Address.
- **Safety Check**: Duplicate emails are not allowed, and passwords are stored securely.

4. Testing Core Features

- **Manual Testing**
    - Leave a field empty → Error should appear.
    - Enter wrong email → Show error.
    - Weak password → Show message to improve.
    - Password mismatch → Prevent form submission.
    - All valid inputs → Form should submit successfully.
- **Accessibility Testing**
    - Screen readers announce error messages clearly.
    - Forms work smoothly on mobile phones and tablets.

- **Performance Testing**
    - Validation messages appear within 1 second.
    - Form should work without slowing down even with multiple fields.

5. Version Control (GitHub)

- **Step 1**: Initialize Git in the project folder.
- **Step 2**: Save changes often with clear commit messages. Example:
    - *“Added signup form validation”*
    - *“Improved password strength check”*
    - *“Connected database for storing user info”*
- **Step 3**: Upload project to GitHub for backup and sharing.
- **Step 4**: Use branches for new features (e.g., signup-form, contact-form) before merging into the main project.

4. Enhancements & Deployment

---

1. Additional Features

- **Multi-Step Form** → Break long forms into steps (Personal Info → Academic Info → Account Setup).
- **Save and Continue Later** → Users can save form progress and complete it later.
- **Captcha Integration** → Add reCAPTCHA to prevent bot entries.
- **Email Notification** → Send confirmation mail after successful registration.
- **Dark Mode Option** → User can switch between light and dark themes.
- **Form Summary Page** → Show all entered details before final submission.

2. UI/UX Improvements

- **Mobile-Friendly Layout** → Improved design for different screen sizes.
- **Progress Bar** → Shows how much of the form is completed (Step 1 of 3, Step 2 of 3).
- **Clear Messages** → Instead of technical errors, show user-friendly messages (e.g., “Please enter a valid email” instead of “Invalid format”).
- **Accessibility Features** → Better screen reader support, larger buttons, and proper color contrast.
- **Animations** → Smooth transitions for error messages and success popups.

3. API Enhancements

- **Improved Validation APIs** → Faster response for email and username availability.
- **Standard JSON Responses** → All errors and success messages follow a common format.

- **Authentication** → Secure login using JWT (JSON Web Token) for future expansion.
- **Third-Party Integration** → Can be connected with student management systems or Google login.
- **API Documentation** → Clear explanation of available APIs for future developers.

4. Performance & Security Checks

- **Performance Improvements**
  - Minified CSS and JavaScript files for faster loading.
  - Optimized database queries to reduce response time.
  - Caching frequently used data for quicker access.
- **Security Enhancements**
  - Passwords stored in encrypted form.
  - Strong input validation to stop SQL Injection and XSS attacks.
  - HTTPS used for secure data transfer.
  - Rate limiting added to prevent multiple wrong login attempts.
- **Testing & Monitoring**
  - Load tested with multiple users at the same time.
  - Regular penetration testing to find security weaknesses.
  - Error logs monitored for quick fixing of problems.

5. Project Demonstration & Documentation

---

1. Final Demo Walkthrough

- Open **index.html** in a web browser.
- Fill the form: Name, Student ID, Department, Year, Email, and Password.
- **Invalid inputs:** Error messages, when a field is empty or formatted incorrectly.
    - Example: Incorrect email → "Please enter a valid email."
    - Weak password → "Password must be 8+ characters, uppercase, number & symbol."
    - Student ID invalid → "Student ID must be 6–10 alphanumeric characters."
- **Valid inputs:** Errors disappear and a confirmation message/alert indicates successful registration.

2. Project Report

Project Title: Interactive Form Validation

Team Members:

- N. Aashika
- S. Abhinaya
- K. Gowri

Problem Statement:

Many users enter wrong details in web forms. Our goal is to build a form that gives **instant error or success.**

Objective:

- Validate user inputs in real time
- Show error messages immediately
- Design a clean and responsive form

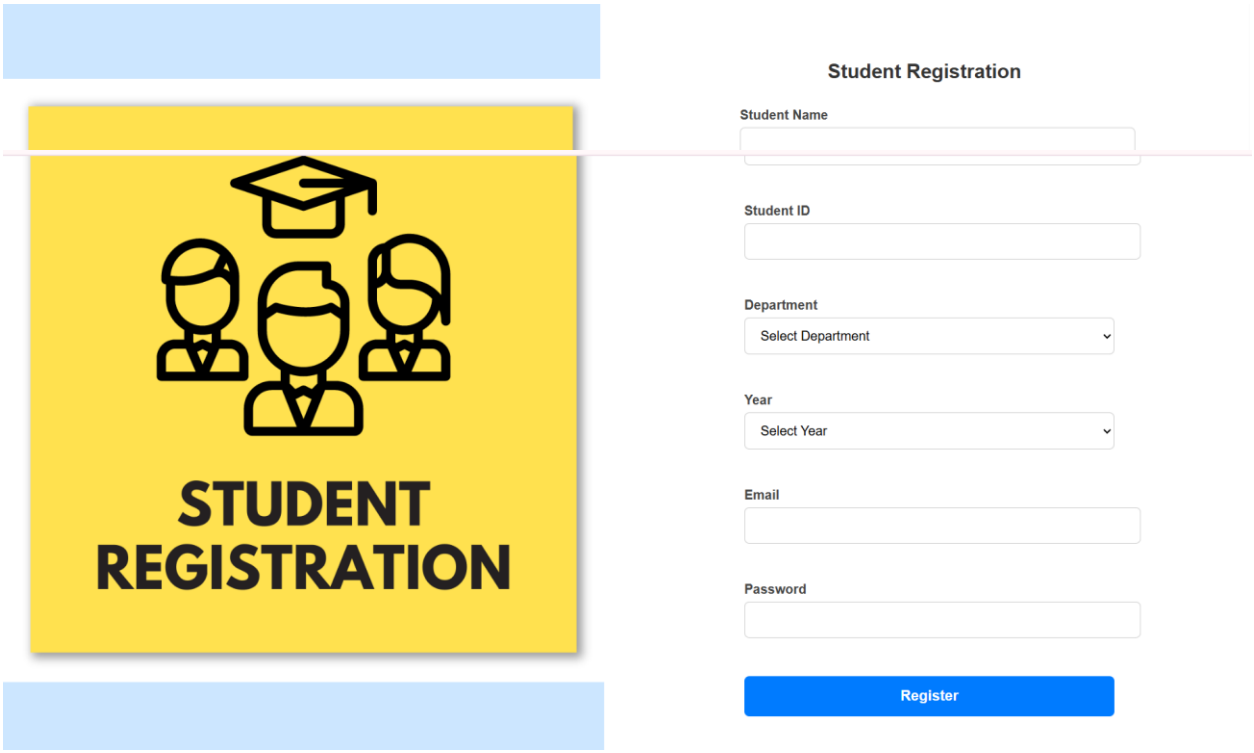
Features:

- Real-time validation for name, email, password
- Inline error and success messages
- Responsive and user-friendly design

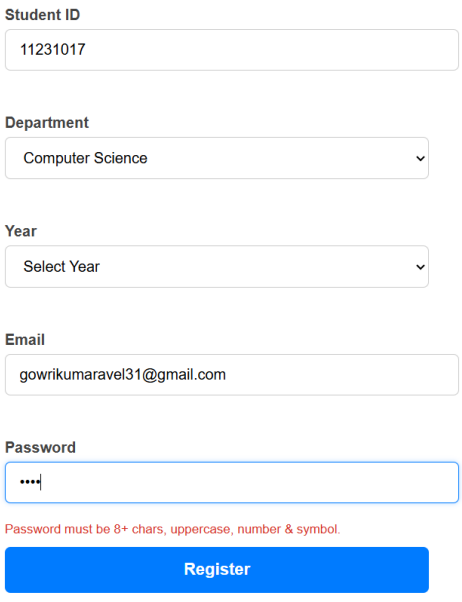
3. Screenshots / API Documentation

- Form Interface: Shows all input fields and Register button
- Error Message: Shows inline error when invalid data entered
- Success Message: Confirms successful registration when all inputs are valid

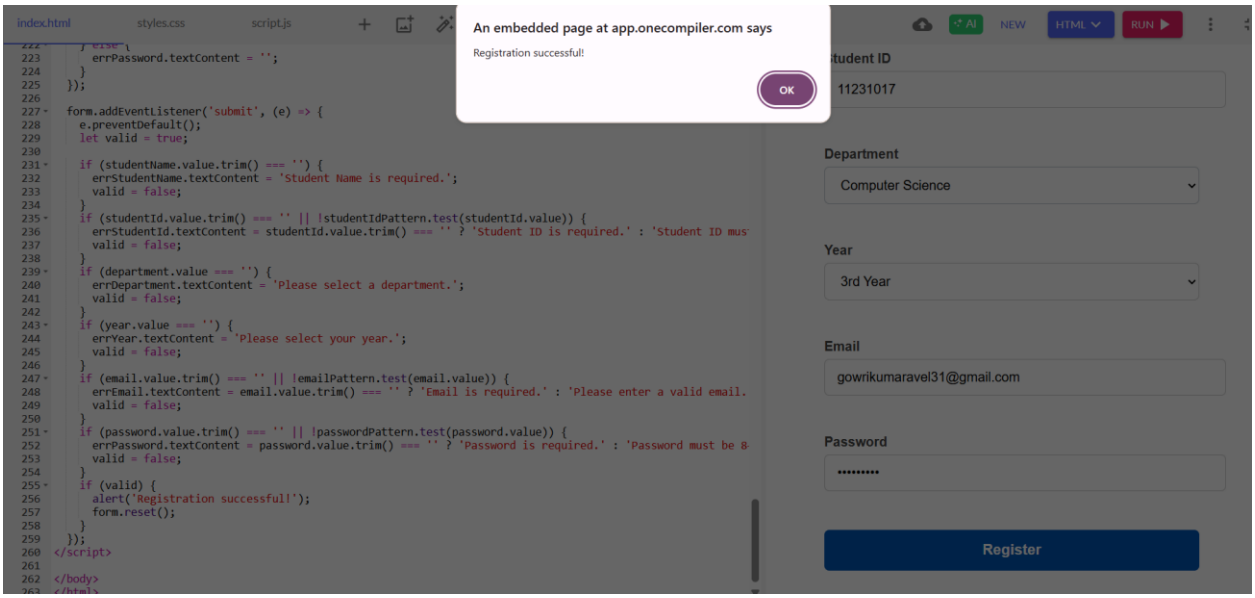
Screenshot form → form interface:



Screenshot error → error message example:



Screenshot success → success message example:



4. Challenges & Solutions

Challenge	Solution
Email validation	Used regex to check format
Showing instant feedback	Used JavaScript event listeners
Aligning design	Used CSS for layout
Password validation rules	Added simple JS conditions for uppercase, number, and symbol

5. GitHub README & Setup Guide

The project includes a **README.md** file that provides a complete overview of the Interactive Form Validation system. It contains the project description, main features, technologies used, and detailed setup instructions.

Setup Instructions:

1. Download or clone the repository from GitHub.
2. Open the **index.html** file in any web browser.
3. Fill the form fields to experience real-time validation and instant feedback.

The README serves as a clear and structured guide to help users understand, install, and execute the project efficiently.

The form layout is aligned, clean and responsive for desktop and mobile. Left-side is logo appears on larger screens. Form is centered and easy to navigate.

6. Final Submission (Repo + Deployed link)

The completed project is hosted on **GitHub** for version control and easy accessibility.

GitHub Repository Link:
https://github.com/GowriCherry31/interactive-form-validation

Deployed Link:
https://gowricherry31.github.io/interactive-form-validation/

The project is hosted using **GitHub .**Users can open the deployed link to view and test the form directly in their web browser, observing real-time validation features, responsive layout, and live feedback messages.

The project repository contains all necessary source files, documentation, and setup details, ensuring it is complete, functional, and ready for submission.

7. Conclusion

This project focuses on developing aformthat enhances the accuracy, accessibility and overall user experience of online forms. Traditional web forms often lack instant feedback, leading to user frustration, higher error rates, and poor data quality. This project aims to solve these issues by implementing real-time validation, responsive design, and accessibility features that provide instant, meaningful feedback to users while typing.