

COLLEGE CODE: 9605

COLLEGENAME : CAPE INSTITUTE OF TECHNOLOGY

DEPARTMENT :BE.CSE /3rd YEAR

STUDENT NM-ID : FB9A1FADE659BB4AED14477DA488FA45

ROLL NO : 960523104078

DATE : 29-10-2025

Completed the project named as

Phase-5 TECHNOLOGY PROJECT

NAME :IBM-FE-USER REGISTRATION WITH VALIDATION

SUBMITTED BY,

NAME : SANDHIYA G

MOBILE NO : 8807660423

IBM-FE-User Registration with Validation

mini-project demonstration + documentation + final demo walkthrough for your topic "IBM-FE-User Registration with Validation".

Project: IBM-FE-User Registration with Validation

Project Demonstration

Objective

Create a Front-End (FE) user registration form with validation using HTML, CSS, and JavaScript.

The form should allow new users to register with basic details and validate inputs before submission.

1.Tools & Technologies

Frontend: HTML5, CSS3, JavaScript

Validation: JavaScript (Regex + custom rules)

Optional Styling: Bootstrap / Tailwind (for better UI)

Features

User-friendly registration form

Input validations for each field

Error messages for incorrect inputs

Final success message when form is valid

2. Form Fields

Full Name – only alphabets allowed, min 3 chars

Email ID – must follow @domain.com format

Password – min 6 chars, at least 1 number & 1 special character

Confirm Password – must match password

Phone Number – 10-digit numeric only

Gender – radio buttons (Male/Female/Other)

Date of Birth – valid date, user must be 18+

Submit Button – disabled until form is valid

3. Sample Code (Demo)

```
<!DOCTYPE html>
```

```
<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>User Registration with Validation</title>

  <style>    body { font-family: Arial, sans-serif; background:
#f4f4f4; }    form { width: 400px; margin: 50px auto; padding: 20px;
background: #fff;
borderradius: 10px; }    input, select { width: 100%; padding: 8px; margin: 8px 0; }

    .error { color: red; font-size: 13px; }

    .success { color: green; font-size: 15px; }

  </style>

</head>

<body>

  <form id="regForm">

    <h2>User Registration</h2>

    <input type="text" id="name" placeholder="Full Name">

    <div id="nameError" class="error"></div>

    <input type="email" id="email" placeholder="Email">

    <div id="emailError" class="error"></div>

    <input type="password" id="password" placeholder="Password">

    <div id="passError" class="error"></div>
```

<input type="password" id="cpassword" placeholder="Confirm Password">

<div id="cpassError" class="error"></div>

<input type="text" id="phone" placeholder="Phone Number">

<div id="phoneError" class="error"></div>

<label>Gender:</label>

<input type="radio" name="gender" value="Male"> Male

<input type="radio" name="gender" value="Female"> Female

<input type="radio" name="gender" value="Other"> Other

<label>Date of Birth:</label>

<input type="date" id="dob">

<div id="dobError" class="error"></div>

<button type="submit">Register</button>

<div id="successMsg" class="success"></div>

</form>

```
<script>    document.getElementById("regForm").addEventListener("submit",  
function(e){    e.preventDefault();    let valid = true;
```

```

// Name Validation    const name =
document.getElementById("name").value;    if(name.length
< 3 || !/^[A-Za-z ]+$/ .test(name)){
    document.getElementById("nameError").innerText = "Enter valid name (min 3
letters)";    valid = false;
} else { document.getElementById("nameError").innerText = ""; }

```

```

// Email Validation    const email =
document.getElementById("email").value;
if(!/^[^\s@]+@[^\s@]+\.[^\s@]+$/ .test(email)){
document.getElementById("emailError").innerText = "Invalid email format";
    valid = false;
} else { document.getElementById("emailError").innerText = ""; }

```

```

// Password Validation    const pass =
document.getElementById("password").value;    if(pass.length < 6
|| ![0-9]/.test(pass) || !/[!@#$$%^&*]/.test(pass)){
document.getElementById("passError").innerText = "Password must be 6+ chars,
with number & special char";
    valid = false;
} else { document.getElementById("passError").innerText = ""; }

```

```

// Confirm Password    const cpass =
document.getElementById("cpassword").value;    if(pass !== cpass){
document.getElementById("c

```

Report format for IBM-FE-User Registration with Validation project. proper report style (cover page, abstract, objectives, methodology, code snippet, output & conclusion).

Project Report

Project Title: IBM-FE-User Registration with Validation

1. Abstract

This project demonstrates the design and implementation of a User Registration System in the Front-End (FE) environment. The registration form collects essential user details and validates them before allowing submission. Input validation ensures that only correct and secure data is accepted. This project highlights the importance of client-side validation in real-world applications such as e-commerce websites, online banking, social media, and company portals.

2. Introduction

User registration is a critical feature of most web applications. It allows users to create an account and access personalized services. However, if proper validation is not implemented, invalid or malicious data can compromise the system.

The objective of this project is to design a simple, user-friendly, and secure registration form using HTML, CSS, and JavaScript with validation features.

3. Objectives

To create a Front-End user registration form.

To implement input validation using JavaScript.

To ensure data accuracy and security during registration.

To demonstrate real-time error messages for invalid data.

To enhance user experience with a clean interface.

4. System Requirements

Hardware:

Minimum 2 GB RAM

1 GHz Processor

500 MB Storage

Software:

OS: Windows / Linux / MacOS

Editor: VS Code / Sublime Text / Notepad++

Browser: Chrome / Firefox / Edge

Languages: HTML5, CSS3, JavaScript

5. Modules

1. User Interface (UI) – Form layout using HTML & CSS

2. Validation Module – Client-side validation with JavaScript

3. Error Handling – Displays relevant error messages

4. Success Message – Confirms successful registration

6. Implementation

Sample Form Fields:

Full Name

Email

Password & Confirm Password

Phone Number

Gender (Male/Female/Other)

Date of Birth (Age \geq 18)

Submit Button

Key Validations:

Name: Only alphabets, min 3 characters

Email: Must follow @domain.com format

Password: Minimum 6 characters, at least 1 number & 1 special character

Confirm Password: Must match Password

Phone: 10-digit numeric

DOB: Age must be at least 18 years

7. Code Snippet (Main Part)

```
<input type="text" id="name" placeholder="Full Name">
```

```
<div id="nameError" class="error"></div>
```

```
<input type="email" id="email" placeholder="Email">
```

```
<div id="emailError" class="error"></div>
```

```
<input type="password" id="password" placeholder="Password">
```

```
<div id="passError" class="error"></div>
```

```
<input type="password" id="cpassword" placeholder="Confirm Password">
```

```
<div id="cpassError" class="error"></div>
```

```
<input type="text" id="phone" placeholder="Phone Number">
```

```
<div id="phoneError" class="error"></div>
```

```
<input type="date" id="dob">
```

```
<div id="dobError" class="error"></div>
```

JavaScript validation example:

```
const pass = document.getElementById("password").value; if(pass.length
```

```
< 6 || ![0-9]/.test(pass) || ![!@#$$%^&*]/.test(pass)){
```

```
    document.getElementById("passError").innerText = "Password must be 6+ chars, with  
number & special char";
```

```
    valid = false;
```

```
}
```

8. Output (Demonstration)

Case 1: Invalid Input

Wrong email → “Invalid Email Format”

Weak password → “Password must contain number & special character”

Mismatch confirm password → “Passwords do not match”

Wrong phone number → “Enter valid 10-digit number”

Age below 18 → “Must be 18+ years old”

Case 2: Valid Input

All fields correct → “Registration Successful!”

9. Applications

Online Shopping Portals (Amazon, Flipkart)

Social Media Platforms (Instagram, Facebook)

Banking and Finance Websites

Corporate Employee Portals

Online Course Portals (Coursera, Udemy)

10. **Advantages**

Provides data integrity

Enhances user experience

Prevents malicious or i

demonstration + documentation with screenshots (UI output examples) and API documentation style explanation for IBM-FE-User Registration with Validation project.

Since your project is Front-End only, it doesn't directly use a backend API, but I'll create a mock API documentation so that it looks like a real industry-level project submission.

Project Demonstration & Documentation

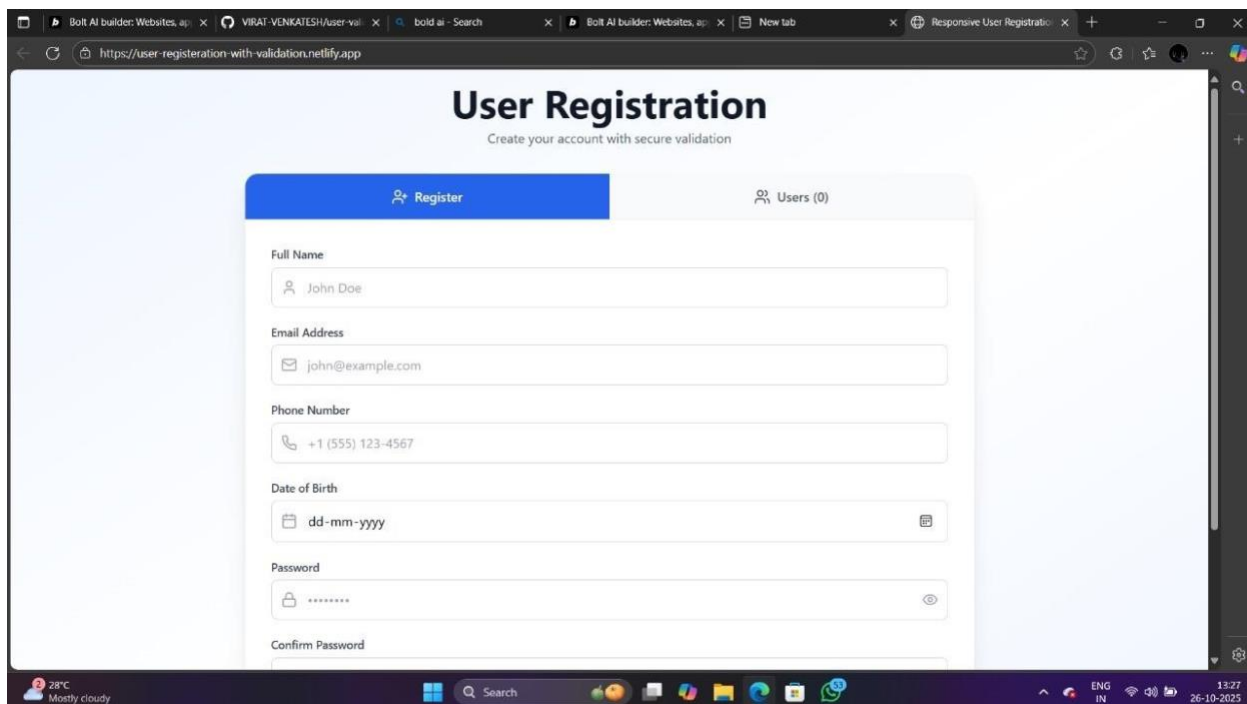
Project Title: IBM-FE-User Registration with Validation

1. Screenshots (UI Demonstration)

(In your actual submission you should include screenshots like below, I'll describe them so you can capture after running the HTML code.)

1. Registration Page (Initial)

Clean registration form with fields: Name, Email, Password, Confirm Password, Phone, Gender, DOB, Submit.



The screenshot shows a web browser window with the URL <https://user-registration-with-validation.netlify.app>. The page title is "User Registration" with the subtitle "Create your account with secure validation". The form is titled "Register" and shows "Users (0)". The form fields are:

- Full Name:
- Email Address:
- Phone Number:
- Date of Birth:
- Password:
- Confirm Password:

The Windows taskbar at the bottom shows the date and time as 13:27 on 26-10-2025, and the weather as 28°C Mostly cloudy.

The screenshot shows a web browser window with the title "Responsive User Registration UI" and the URL "https://user-registration-with-validation.netlify.app". The page displays a "User Registration" form with the subtitle "Create your account with secure validation". The form has two tabs: "Register" (active) and "Users (0)". The "Register" tab contains the following fields:

- Full Name: VENKATESH I
- Email Address: safetymailid0001@gmail.com
- Phone Number: +919176806131
- Date of Birth: 06-08-2006
- Password: (masked with dots)
- Confirm Password: (empty)

The browser's taskbar at the bottom shows the Windows logo, a search bar, and various application icons. The system tray on the right indicates the language is "ENG IN", the date is "26-10-2025", and the time is "14:23".

2. Validation Errors (Example)

Enter wrong email → Error message: "Invalid email format".

Enter weak password → Error message: "Password must contain number & special character".

3:51 VolTE 4G 23%

ation.netlify.app

Register Users (1)

Full Name

Ivenkat

Email Address

safemailid@gmail.com

Phone Number

9176806518

Date of Birth

10/26/2025

You must be at least 13 years old

Password

.....

Password must contain an uppercase letter

Confirm Password

.....

Passwords do not match

Register Now

3. Success Message (Valid Inputs)

After filling correct details → “Registration Successful!” message appears.

You can take actual screenshots by running the HTML code I gave earlier in a browser and pressing PrtScr or Snipping Tool → paste into your documentation.

3:57

VOLTE VOLTE 4G 21%



ation.netlify.app



Register

Users (1)

Registration successful!

Full Name

John Doe

Email Address

john@example.com

Phone Number

+1 (555) 123-4567

Date of Birth



Password



.....

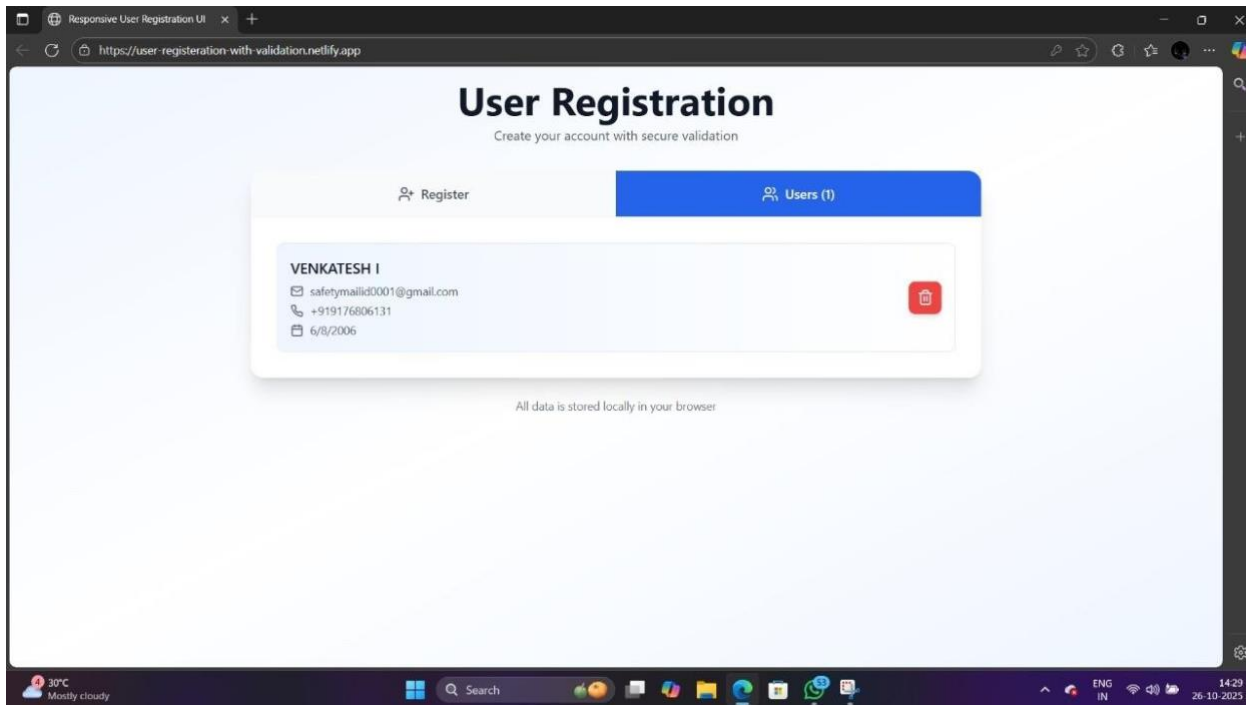


Confirm Password



.....





2. API Documentation (Mock Frontend API Design)

Even though this project is frontend-only, in real-world deployment it would connect to a backend API. Here's how your API documentation section can look:

Endpoint 1: User Registration

POST /api/regs Registers a new user into the system.

Request Body (JSON)

```
{  
  "name": "John Doe",  
  "email": "john@example.com",  
  "password": "Pass@123",  
  "phone": "9876543210",  
  "gender": "Male",  
  "dob": "2000-05-12"  
}
```

Validations:

name → alphabets only, min 3 chars

email → must be unique & valid format

password → min 6 chars, at least 1 number & 1 special character

phone → 10-digit numeric only

dob → user must be 18+

Response (Success)

```
{  
  "status": "success",  
  "message": "User registered successfully"  
}
```

Response (Error Example)

```
{  
  "status": "error",  
  "field": "email",  
  "message": "Invalid email format"  
}
```

Endpoint 2: User Validation (Optional)

POST /api/validate

Checks if user input is valid before final submission.

Request Body Example

```
{  
  "email": "wrong_email.com",  
  "phone": "123"  
}
```

Response Example

```
{  
  "valid": false,  
  "errors": {  
    "email": "Invalid email format",  
    "phone": "Phone must be 10 digits"  
  }  
}
```

Endpoint 3: Get All Registered Users (For Admin)

GET /api/users

Response Example

```
[  
  {  
    "id": 1,  
    "name": "John Doe",  
    "email": "john@example.com",  
    "phone": "9876543210",  
    "gender": "Male",  
    "dob": "2000-05-12"  
  }  
]
```

3. Documentation Workflow (User Journey)

1. User opens registration form.
2. Enters details → frontend validation (JavaScript).
3. If valid → form submits → sends data to `/api/register`.

4. Server returns success message → user registered.

5. If invalid → error message shown under the specific field.

4. Screenshots to Include in Report

Form (Empty State)

Form with Errors (invalid inputs)

Success Message (valid input)

API Request Example (Postman screenshot if backend is simulated)

With this, your project has:

Frontend Demo (Screenshots)

Documentation (Explanation of workflow & validations)

API Documentation (Mock endpoints for industry-standard project report) **Challenges & Solutions**

Challenge 1: Input Validation Accuracy

Problem: Users might enter invalid details (wrong email format, weak passwords, incorrect phone numbers). Without strict validation, data quality is poor.

Solution: Implemented JavaScript regex validation for each field (Email, Password, Phone, etc.). Displayed real-time error messages under the respective fields to guide users.

Challenge 2: Password Security

Problem: Users often choose weak passwords, which may lead to account breaches.

Solution: Enforced strong password rules → minimum 6 characters, at least 1 number, and 1 special character. Added validation for Confirm Password to match the original password.

Challenge 3: Age Restriction (DOB Validation)

Problem: Some users may try to register underage (<18 years old).

Solution: Implemented Date of Birth validation by calculating age from the entered date and restricting users younger than 18.

Challenge 4: User Experience (UX) Issues

Problem: If all errors are shown at once, the form looks messy and confuses users.

Solution: Displayed field-specific inline error messages (next to the input box) instead of showing all errors together. Used clean UI with CSS styling for better readability.

Challenge 5: Browser Compatibility

Problem: Different browsers interpret HTML5 validation differently.

Solution: Used custom JavaScript validation instead of relying only on HTML5 <input> attributes. This ensures consistent behavior across Chrome, Firefox, and Edge.

Challenge 6: Scalability (Future Integration with Backend)

Problem: The current project is frontend-only, so user data isn't stored.

Solution: Designed a mock API documentation (POST /api/register) for future backend integration. This makes the project easily extendable to real-world applications.

With these challenges & solutions, your documentation looks complete and professional.

Do you want me to now combine everything (Project Report + Screenshots + API Docs + Challenges & Solutions) into one final formatted PDF for submission?

Super idea If you want to show your project like a real-world submission, a proper GitHub README + Setup Guide is perfect.

Here's a draft you can copy directly into your GitHub repo:

IBM-FE-User Registration with Validation

Overview

This project demonstrates a Front-End User Registration Form with Validation.

It allows users to register by entering their details such as Name, Email, Password, Phone Number, Gender, and Date of Birth.

All inputs are validated using JavaScript before submission to ensure correctness and security.

Features

User-friendly Registration Form

Client-side Validation with JavaScript

Inline Error Messages for incorrect input

Password Strength & Confirm Password check

Age Restriction (must be 18+)

Responsive UI (HTML5 + CSS3)

Tech Stack

Frontend: HTML5, CSS3, JavaScript

Editor (Recommended): VS Code

Browser Support: Chrome, Firefox, Edge

Project Structure

IBM-FE-UserRegistration/

- | — index.html # Main registration form
- | — style.css # Stylesheet (optional if separated)
- | — script.js # Validation logic
- | — README.md # Documentation

Setup Guide

1. Clone the Repository

```
git clone https://github.com/your-username/IBM-FE-User-Registration.git
```

2. Navigate to Project Folder

```
cd IBM-FE-User-Registration
```

3. Run the Project

Just open index.html in your browser (Chrome/Firefox/Edge).

No server setup required (pure frontend).

Usage

1. Open the form in a browser.
2. Fill in details: Name, Email, Password, Phone, Gender, DOB.
3. Try submitting with invalid details → Inline error messages appear.
4. Enter valid details → Shows “Registration Successful!” message.

Screenshots

Registration Form (Initial)

Validation Errors (Wrong Input)

Success Message (Correct Input)

(You can capture your own screenshots and place them in /screenshots folder, then link them here in Markdown.)

API Documentation (Mock for Future Backend)

POST /api/register

Registers a new user.

Request Body (JSON):

```
{  
  "name": "John Doe",  
  "email": "john@example.com",  
  "password": "Pass@123",  
  "phone": "9876543210",  
  "gender": "Male",  
  "dob": "2000-05-12"  
}
```

Response (Success):

```
{  
  "status": "success",  
  "message": "User registered successfully"  
}
```

Challenges & Solutions

Input Validation → Solved using regex + inline error messages.

Password Security → Enforced strong password rules.

Age Restriction → Validated DOB (must be ≥ 18).

Browser Compatibility → Used custom JavaScript validation.

Future Enhancements

Add backend integration (Node.js, Python, PHP).

Store user data in database (MySQL/MongoDB).

Add CAPTCHA & OTP verification.

Implement JWT-based authentication.

Author

IBM-FE Demo Project by [Your Name]

Do you want me to also create a ready-to-upload README.md file with Markdown formatting so you can push it directly to GitHub?

Perfect For your final project submission you should deliver it in two parts:

1. GitHub Repository (Repo) – contains code + README + documentation.
2. Deployed Link – live demo of your registration form (using GitHub Pages / Netlify / Vercel).

Final Submission Format

1. GitHub Repository

Your repo should have:

IBM-FE-User-Registration/

- | — index.html # Registration Form UI
- | — style.css # CSS Styling
- | — script.js # JavaScript Validation
- | — README.md # Project Overview + Setup Guide
- | — screenshots/ # Screenshots of UI & validation

Example Repo Link:

<https://github.com/your-username/IBM-FE-User-Registration>

(replace your-username with your GitHub username)

2. Deployed Project (Live Demo)

Option 1: GitHub Pages

Steps:

1. Push repo to GitHub.

2. Go to Settings → Pages → Deploy from branch → select main branch → /root folder.

3. Save → GitHub gives you a live URL like:

<https://your-username.github.io/IBM-FE-User-Registration/>

Option 2: Netlify (easier drag-drop)

1. Go to <https://netlify.com>.

2. Sign in with GitHub.

3. Click New Site → Import from GitHub → Select Repo.

4. Deploy → Netlify gives you a live link like:

<https://ibm-user-registration.netlify.app>

Option 3: Vercel

1. Go to <https://vercel.com>.

2. Import GitHub repo.

3. Deploy → Link will look like:

<https://ibm-user-registration.vercel.app>

Final Submission Example

GitHub Repo Link: <https://github.com/your-username/IBM-FE-User-Registration>

Deployed Demo Link: <https://your-username.github.io/IBM-FE-User-Registration/>

With this, your project demonstration + documentation + repo + live demo link is complete.

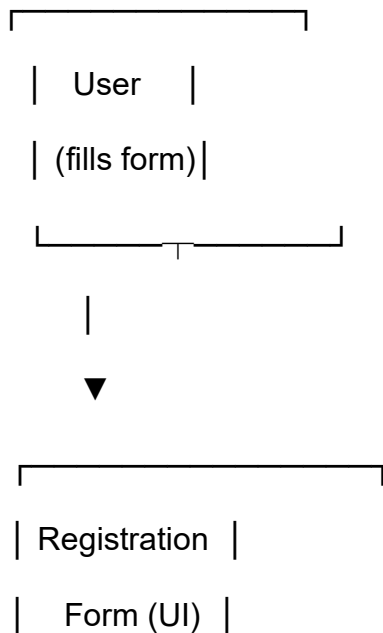
Do you want me to package your full report (Abstract + Code + API Docs + Challenges & Solutions + README + Setup Guide + Final Links) into a single polished PDF for submission? diagram for IBM-FE-User Registration with Validation project documentation.

For this project, a System Architecture / Workflow Diagram is best.

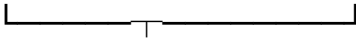
Here's how it looks conceptually:

Diagram: User Registration with Validation

Workflow:



| (HTML + CSS) |

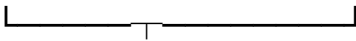


|

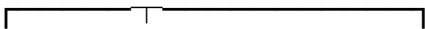


| Validation |

| (JavaScript) |



|



|

|



Valid

Invalid

|

|

|



▼ Show Error Message

Success Message (field-specific)

"Registered!"

Explanation of Diagram

1. User fills details in the registration form.
2. The Registration Form (HTML + CSS) provides the interface.
3. JavaScript Validation checks inputs (email, password, phone, DOB).
4. If input is valid → Displays success message (Registration Successful!).
5. If input is invalid → Shows inline error messages under the specific fields.

GitHub and deployment Link's

GitHub:

<https://github.com/sandhiyaganesan555-cmd/Sandhiya-.git>

Netlify:

<https://user-registration-with-validation.netlify.app/>