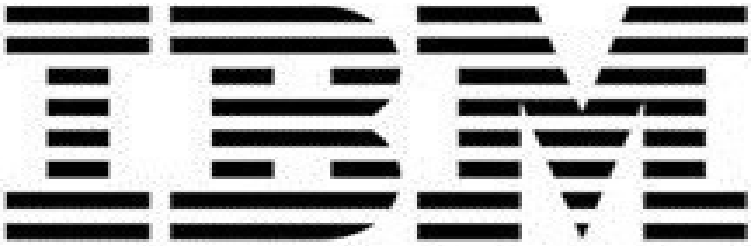

COLLEGE CODE COLLEGE NAME DEPARTMENT STUDENT NM-

ID ROLL NO : 953023104013

DATE : 06/10/2025

NM-ID : EA39DF4FD84A6DA921347567F2C2E7C2

Completed the project Phase : 5

Project: USERS REGISTRATION WITH VALIDATION

SUBMITTED BY,

NAME : ANTRO RIYAS

MOBILE NO : 6374500519

Project Demonstration & Documentation

Phase 5

1. Final Demo Walkthrough

Step 1: Open the registration form.

Step 2: Enter invalid inputs (e.g., empty email, short password) → Show real-time error messages.

Step 3: Enter correct details → Successful validation.

Step 4: Show that the user's data is stored in the database.

Step 5 (Optional): Redirect user to login page or dashboard after registration.

Demo Format:

- Live demo during presentation, or
- A **recorded screen walkthrough** (2–3 minutes) hosted on Google Drive / YouTube / Loom.

2. Project Report (Structure

A full academic-style report should contain the following:

a) **Title Page**

- Project Title: *User Registration with Validation*
- Student Name(s), Roll Number(s)
- Course / Subject, Semester, Institute

b) **Abstract**

A short overview of why secure user registration is necessary and what your project achieves.

c) **Introduction**

- Importance of registration in web applications.

- Why validation is necessary (to prevent errors, improve UX, ensure security).

d) **Objectives**

- To implement a secure and user-friendly registration system.
- To validate user inputs both on **client-side** (JavaScript) and **server-side** (backend).
- To ensure data security using hashing & input sanitization.

e) **System Design**

- **Architecture Diagram**: Client (frontend) ↔ Server (backend API) ↔ Database.
- **ER Diagram**: User table (fields: id, username, email, password_hash, created_at).
- **Validation Flow**: Input → Validation → Error/Success → Database Insert.

f) **Implementation**

- Technologies used (HTML, CSS, JS, Node.js/PHP/Java/Python, MySQL/MongoDB).
- Form fields: Username, Email, Password, Confirm Password.
- Validation checks:
  - Username: min 3 characters, no special characters.
  - Email: must follow valid email pattern.
  - Password: min 8 characters, at least 1 uppercase, 1 number, 1 special character.
  - Confirm Password: must match Password.

g) **Testing**

- Valid email vs invalid email.
- Short password vs strong password.
- Duplicate user entry.
- SQL injection attempt.
- Empty fields.

h) **Conclusion**

- Secure and reliable registration system achieved.
- Helps in reducing faulty user data.
- Provides better user experience.

3. Screenshots / API Documentation

POST /register

- **Request Body:** { "username": "john123", "email": "john@mail.com", "password": "Pass@123" }
- **Responses:**
  - ✅ Success: { "message": "User registered successfully" }

- ❌ Error: { "error": "Email already exists" }

4. Challenges & Solutions

Challenge	Solution
Handling invalid input	Used regex + server-side checks
Duplicate user registration	Checked for existing emails in DB
Password security	Implemented hashing (bcrypt/MD5/SHA256)
Preventing SQL injection	Used parameterized queries / ORM
User-friendly error handling	Inline error messages with JavaScript

5. GitHub README & Setup Guide

Repository Link:
👉 SMTEC-NM-IBM-USERS-REGISTRATION-WITH-VALIDATION

README should include:

- Project Title & Description.
- Features (validation rules, database integration).
- Tech Stack used.
- Setup Instructions:
  - Clone repo
  - Install dependencies (npm install or relevant)
  - Setup database (schema.sql)
  - Run server (npm start / php -S localhost:8000)

- Usage instructions (how to register a user).
- Screenshots.
- Live Demo Link (if deployed on Vercel/Netlify/Heroku).

6. Final Submission

GitHub Repo -

GitHub - Riyasantro/SMTEC-NM-IBM-USERS-REGISTRATION-WITH-VALIDATION

Conclusion

The project **User Registration with Validation** successfully implements a secure and reliable system for registering users in a web application. By combining client-side and server-side validation, it ensures that only valid and properly formatted data is stored in the database. This not only enhances data integrity but also improves the overall user experience by providing real-time feedback and meaningful error messages. Security measures such as password hashing and input sanitization further protect user data from potential threats like weak credentials and SQL injection.

Overall, the project achieves its objective of creating a robust registration system that is both user-friendly and secure. In the future, it can be extended with additional features such as email/OTP verification, CAPTCHA to prevent bots, and integration with third-party login providers (Google, GitHub, etc.), making it even more versatile and production-ready.