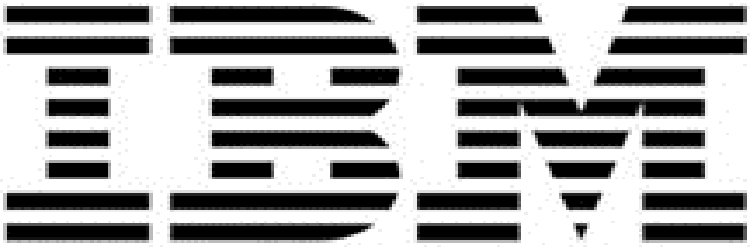

COLLEGE CODE COLLEGE NAME DEPARTMENT STUDENT NM-ID

ROLL NO : 953023104013

DATE : 15/09/2025

NM-ID : EA39DF4FD84A6DA921347567F2C2E7C2

Completed the project Phase : 1

Project: USERS REGISTRATION WITH VALIDATION

SUBMITTED BY,

NAME : ANTRO RIYAS S

MOBILE NO : 6374500519

1. Problem Statement

In today's digital world, almost every web application requires users to register and create an account before accessing certain features. A proper **user registration system** ensures that only valid and authentic users can use the platform.

However, if registration is implemented without validation, it leads to several problems:

- **Invalid Data**: Users may leave fields empty or enter incorrect formats (e.g., wrong email structure).
- **Weak Security**: Without password validation, users may create weak credentials, making accounts vulnerable to attacks.
- **Duplicate Accounts**: Multiple accounts with the same email cause confusion and unnecessary data storage.
- **Bad User Experience**: Lack of feedback on incorrect input frustrates users.

Therefore, the project aims to build a **User Registration Module with Validation** using **Frontend + Node.js REST API**. The system will enforce rules (like strong password policy, unique email, proper formats) to ensure clean, secure, and reliable data entry.

2. Users & Stakeholders

Users

- **New Users**: People who want to register by providing their name, email, and password.
- **Returning Users**: Users who already have accounts and may attempt to register again (system must handle duplicates).

Stakeholders

- **Application Owners**: Want reliable and secure registration to maintain trust.
- **Developers**: Build and maintain the frontend form and backend API.
- **Database Administrators**: Ensure stored user data is consistent and optimized.
- **Security Team**: Responsible for enforcing password policies, encryption, and data privacy compliance.

Example: In an e-commerce system, customers are the users, while the store owners, developers, and IT team are stakeholders.

3. User Stories

User stories describe how the system should behave from different perspectives.

- **As a new user**, I want to register with my email and password so that I can create an account.
- **As a user**, I want the system to validate my email so that I don't enter invalid information.
- **As a user**, I want to be informed if my password is weak so that I can choose a stronger one.
- **As a user**, I want to confirm my password during registration so that I don't make typing errors.
- **As a user**, I want to see error messages instantly on the frontend so that I can fix mistakes quickly.
- **As an admin**, I want duplicate registrations to be prevented so that data remains clean.
- **As a developer**, I want clear API error codes so that frontend and backend remain consistent.

These stories ensure that both usability and security are addressed.

4. MVP Features

The **Minimum Viable Product (MVP)** for the registration module should include the following:

1. **Registration Form**
    - Fields: Name, Email, Password, Confirm Password.
    - Client-side validation (required fields, email format, password length).
2. **Server-Side Validation (Node.js API)**
    - Duplicate email check.
    - Strong password enforcement (minimum 8 characters, uppercase, lowercase, number, special character).
    - Secure password hashing (e.g., bcrypt).
3. **Database Storage**
    - User table with fields: id, name, email, password_hash, created_at.
4. **Error Handling**

- Return meaningful error messages for invalid inputs.
- Handle cases like empty fields, wrong formats, or existing accounts.

5. **Success Confirmation**
    - If all validations pass, save user details and return success message.

5. Wireframes / API Endpoint List

Wireframe (UI Sketch)

| User Registration Form |

Name: [_____]

Email: [_____]

Password: [_____]

Confirm Password:[_____]

[Register]

Error messages appear below each field.

API Endpoints

1. **POST /api/register**

Request Body (JSON)

{

"name": "John Doe",

"email": "john@example.com",

"password": "Strong@123",

"confirmPassword": "Strong@123"

}

- **Response (Success)**

{

"message": "User registered successfully",

"status": 201

}

- **Response (Error)**

{

"error": "Email already exists",

"status": 400

}

2. **GET /api/users** (Admin only)

-
    - Fetch registered users.

6. Acceptance Criteria

- The registration system will be considered successful if it meets the following:

3. **Input Validation**
    - All fields (name, email, password, confirm password) are mandatory.
    - Email must be in correct format and unique.
    - Password must meet strength requirements.
    - Confirm password must match password.
4. **Security Requirements**
    - Passwords are hashed before storing in the database.
    - No plain text password is stored.
5. **System Behavior**
    - Successful registration returns a success message and stores user details.
    - Invalid registration shows clear error messages without crashing the app.
    - API follows REST principles with proper status codes (200, 201, 400, 409).

6. **User Experience**

- Clear form layout.
- Error messages displayed near the input fields.
- Response time should be less than 2 seconds

Conclusion

The **User Registration with Validation** system is an essential foundation for any web application that requires user accounts. By clearly defining the **problem statement, users & stakeholders, user stories, MVP features, wireframes, API endpoints, and acceptance criteria**, this phase establishes a solid roadmap for the project.

A properly validated registration process ensures **data accuracy, security, and a smooth user experience**. It prevents issues such as duplicate accounts, weak passwords, and invalid inputs, while also meeting the expectations of both end-users and stakeholders.

This phase confirms that the registration module will not only allow new users to create accounts but will also maintain system integrity and security through strict validation rules. The outlined **MVP and acceptance criteria** provide a measurable way to evaluate success, ensuring that the project moves into the next development phases with clarity and confidence.