COLLEGE CODE : 9604

COLLEGE NAME : CSI INSTITUTE OF TECHNOLOGY DEPARTMENT : COMPUTER SCIENCE AND ENGINEERING

STUDENT NM- ID : D3AAE798B05A3FA9FEA49AA542B7E1E0

ROLL NO : 960423104072

DATE : 22/09/2025

SUBMITTED BY,

NAME : Sree Aswin.S

MOBILE NO: 9342338715

Phase 2 - USER REGISTRATION WITH VALIDATION

Tech stack selection:

1. User submits form on frontend → frontend validation runs.

1. Data sent to backend → backend validation runs.
2. Backend checks database for duplicate emails.
3. User saved in database → success response sent.

1. Errors handled and shown to user.

UI Structure & API Schema Design:

UI Structure:

1. Form Header → Title ("Create Account") + subtitle ("Register to get started").
2. Input Fields → Full Name, Email, Phone, Password, Confirm Password (with inline validation).
3. Validation Rules → Required fields, format checks (email/phone), password strength, confirm password match.

1. Extras → Terms & Conditions checkbox + link to Login page.
2. Submit Button → Disabled until all validations pass + error/success feedback shown.

API Schema Design:

idUUIDPK

username email

password_hash

VARCHAR(30) VARCHAR(100) TEXT

UNIQUE, NOT NULL UNIQUE, NOT NULL

NOT NULL (hashed with bcrypt/argon2)

phone dob created_at

VARCHAR(15) DATE

TIMESTAMP

NULL NULL

DEFAULT

CURRENT_TIMESTAMP

Data Handling Approach:

User Registration with Validation – Data Handling Approach in a **tabular format**:

Step

1

2

3

4

5

6

7

8

Process Input Capture

Client-side Validation

API Request

Backend Validation

Password Security

Database Storage

Response Handling

Post-Registration Actions

Details

User enters details (username, email, password, phone, etc.) in frontend form.

Regex & rules applied (email format, password strength, phone number length). Errors shown instantly.

Valid data sent via **HTTPS** as JSON (Content-Type: application/json).

Server checks duplicates, enforces business rules, sanitizes inputs.

Password hashed (e.g., bcrypt, Argon2) before storing.

User record stored in users table/collection with unique constraints.

API returns status codes (201 Created, 400 Bad Request, 409 Conflict, 500 Error).

Send email/SMS verification, generate JWT/session token if auto-login is enabled.

Component /ModuleDiagram:

++

| User Interface |

| (Registration UI) |

++

| v

++

| Input Validator |

| (Frontend checks)|

| • Empty fields |

| • Email format |

| • Password rules |

++

| v

++

| Registration API |

| (POST /register) |

| • Receives data |

| • Triggers|

|backend logic |

++

| v

++

| Backend Module |

| (Business Logic) |

| • Server-side|

|validation|

| • Duplicate user |

|check|

| • Hash password |

++

| v

++

| Database Module |

| (User Repository) |

| • Store user|

| • Enforce unique |

|constraints|

++

Key Points in Flow:

1. UI (Frontend) → Collects user details.
2. Validation (Frontend) → Prevents invalid input before sending to backend.

1. API Gateway → Routes registration requests.
2. Backend Validation → Ensures security, checks duplicates, hashes passwords.

1. Database → Stores only valid & secured user

Basic Flow Diagram: