CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342-G1
SYSTEMS INTEGRATION AND ARCHITECTURE 1

FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

Project Title: Laboratory Activity: Mini App – User Registration & Authentication

Prepared By: Canonigo, Johndaniel A.

Date of Submission: 2/06/2026

Version: 1.1

Table of Contents

1. Introduction

1.1. Purpose

This document outlines the requirements for a **simple user authentication and dashboard system** designed to enable users to register, log in, access a personalized dashboard, and securely log out. The intended audience includes:

- **Stakeholders** (product owners, project managers) to validate system scope and functionality.
- **Developers** (frontend/backend engineers) to implement features based on defined requirements.
- **QA testers** to design test cases aligned with functional and non-functional criteria.

1.2. Scope

The system will provide:

- **User registration** with data validation (e.g., email format, password strength).
- **User authentication** (login) with secure credential verification.
- **Dashboard access** for authenticated users.
- **Logout functionality** to terminate user sessions.

Out of Scope:

- Password reset/recovery workflows.
- Social media login integrations (e.g., Google, Facebook).
- Advanced user role management (e.g., admin privileges).
- Multi-factor authentication (MFA).

1.3. Definitions, Acronyms, and Abbreviations

Term	Definition
PK	Primary Key (e.g., user_ID in the User entity diagram).
Auth	Short for "authentication" (verifying user identity).
Dashboard	A personalized interface for authenticated users post-login.
Guest User	A user who has not registered or logged in (from the use case diagram).

2. Overall Description

2.1. System Perspective

The system operates as a **standalone authentication module** for a web application. It integrates with a larger application (e.g., an e-commerce platform or SaaS tool) to manage user identity and session control. The system's boundaries include:

- **Input**: User-submitted registration/login data (e.g., email, password).
- **Output**: Dashboard access, session tokens, or error messages.

- **Integration Points**: Database (for user storage) and the parent application (for dashboard content).

2.2. User Classes and Characteristics

User Class	Characteristics
Guest User	- Cannot access the dashboard. - Can register or attempt to log in.
Authenticated User	- Has a valid session. - Can view the dashboard and log out.

2.3. Operating Environment

- **Frontend**: Web browser (Chrome, Firefox, Safari) with HTML/CSS/JavaScript (e.g., React).
- **Backend**: Server-side framework (e.g., Node.js, Spring Boot) for API endpoints.
- **Database**: Relational database (e.g., PostgreSQL, MySQL) to store user data
- **Security**: HTTPS for encrypted data transmission.

2.4. Assumptions and Dependencies

- Users have basic internet and technical literacy.
- The database is pre-configured with the schema shown in the *User entity diagram*.
- Passwords are hashed using a secure algorithm (e.g., bcrypt)
- The parent application provides the dashboard UI (this system only handles authentication).

3. System Features and Functional Requirements

Describe each major feature of the system and its functional requirements.

3.1. Feature 1: User Registration

Description:
Allows a *Guest User* to create a new account by submitting personal details (from the *User entity diagram*).

Functional Requirements:

- **FR1.1**: Collect user data: first_name, last_name, age, gender, address, email, and password.
- **FR1.2**: Validate inputs (e.g., email format, password length ≥ 8 characters).
- **FR1.3**: Hash the password using PasswordEncoder (see *Class diagram*) before storing in the database.
- **FR1.4**: Redirect to the login page after successful registration.

3.2. Feature 2: User Authentication (Login)

Description:
Verifies user credentials and grants access to the dashboard for *Authenticated Users*.

Functional Requirements:

- **FR2.1**: Accept email and password inputs.
- **FR2.2**: Validate credentials against the database using AuthService.validateUser() (see *Class diagram*).
- **FR2.3**: Display error messages for invalid credentials (e.g., "Wrong Email or Password").
- **FR2.4**: Create a session token via TokenProvider (see *Class diagram*) upon successful login.

3.3. Feature 3: Dashboard Access

Description:
Provides authenticated users with access to a personalized dashboard.

Functional Requirements:

- **FR3.1**: Redirect to the dashboard only after successful authentication (see *Activity diagram*).
- **FR3.2**: Restrict dashboard access to *Authenticated Users* (block unauthorized access).
- **FR3.3**: Display user-specific data (e.g., name, email) on the dashboard.

3.4. Feature 4: Logout

Description:
Terminates the user's session and returns them to the login screen.

Functional Requirements:

- **FR4.1**: Invalidate the session token via AuthController.logout().
- **FR4.2**: Redirect to the login page after logout (see *Activity diagram*).
- **FR4.3**: Clear client-side session data (e.g., cookies, local storage).

4. Non-Functional Requirements

Specify system quality attributes such as performance, security, usability, reliability, etc.

Category	Requirement
Security	- Passwords must be hashed using PasswordEncoder (see *Class diagram*). - All data transmitted via HTTPS.
Performance	- Login/registration requests must respond in < 5 seconds.
Usability	- Intuitive UI with clear error messages (e.g., “Email already exists”).
Reliability	- 99.9% uptime for authentication endpoints. - No data loss during registration/login.
Scalability	- Design to scale horizontally (e.g., load balancing for authentication services).

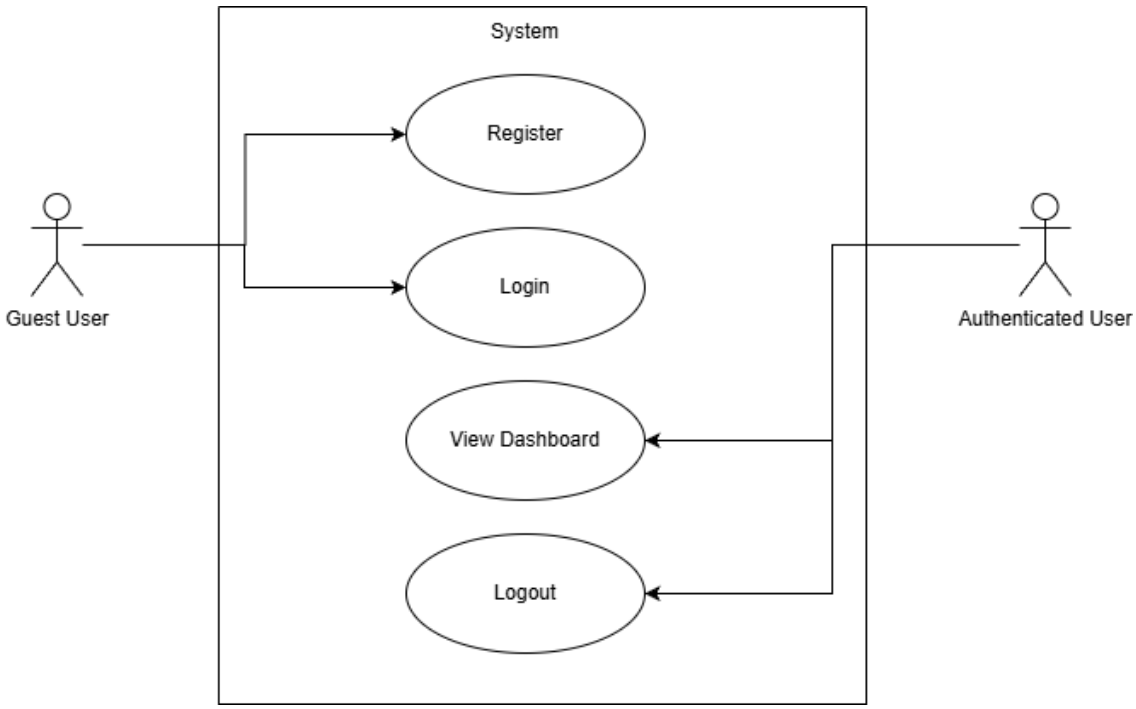
5. System Models (Diagrams)

Insert the necessary diagrams for the system:

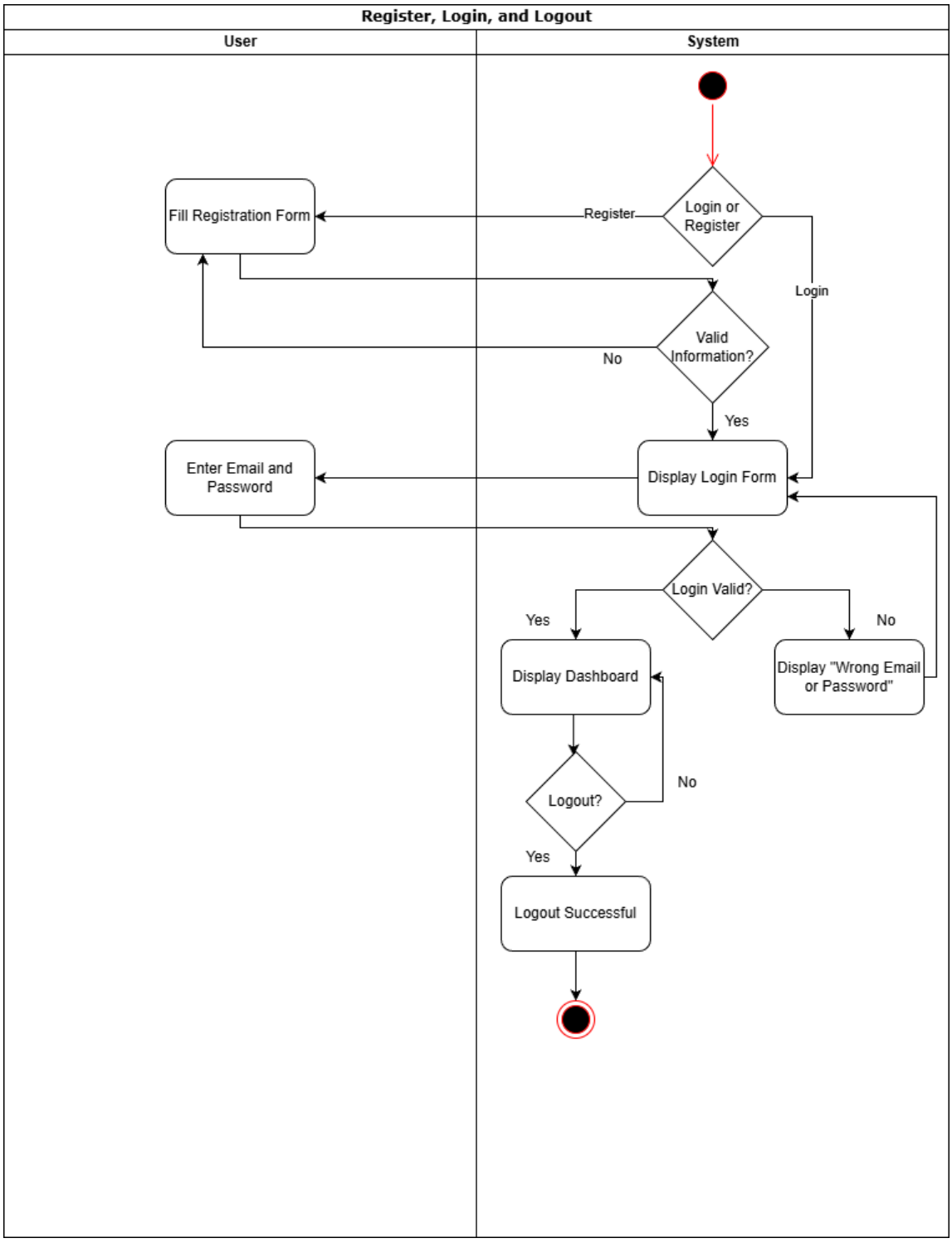
5.1. ERD

User		
PK	user_ID	Data Type
	first_name	varchar(255)
	last_name	varchar(255)
	age	int(3)
	gender	varchar(20
	address	varchar(255)
	email	varchar(255)
	password	varchar(255)

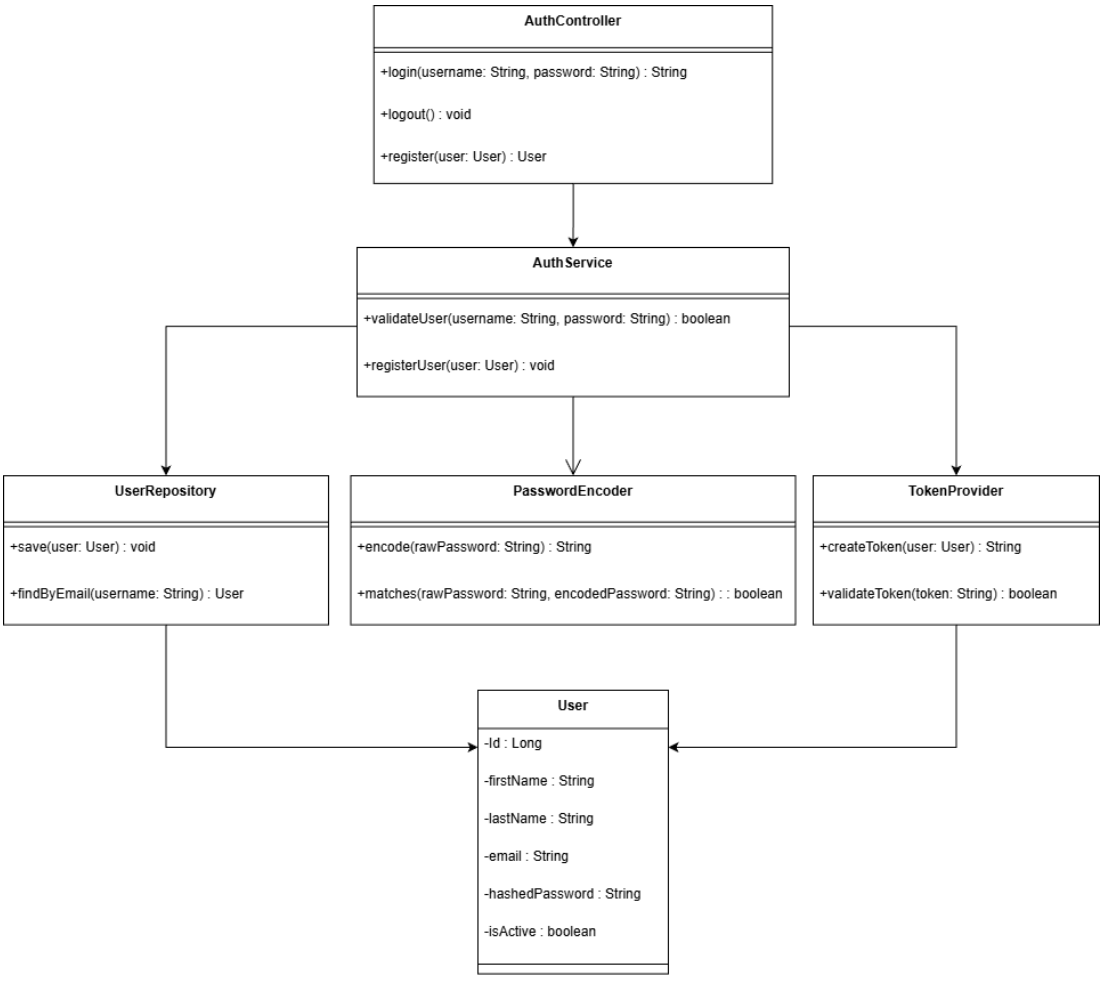
5.2. Use Case Diagram



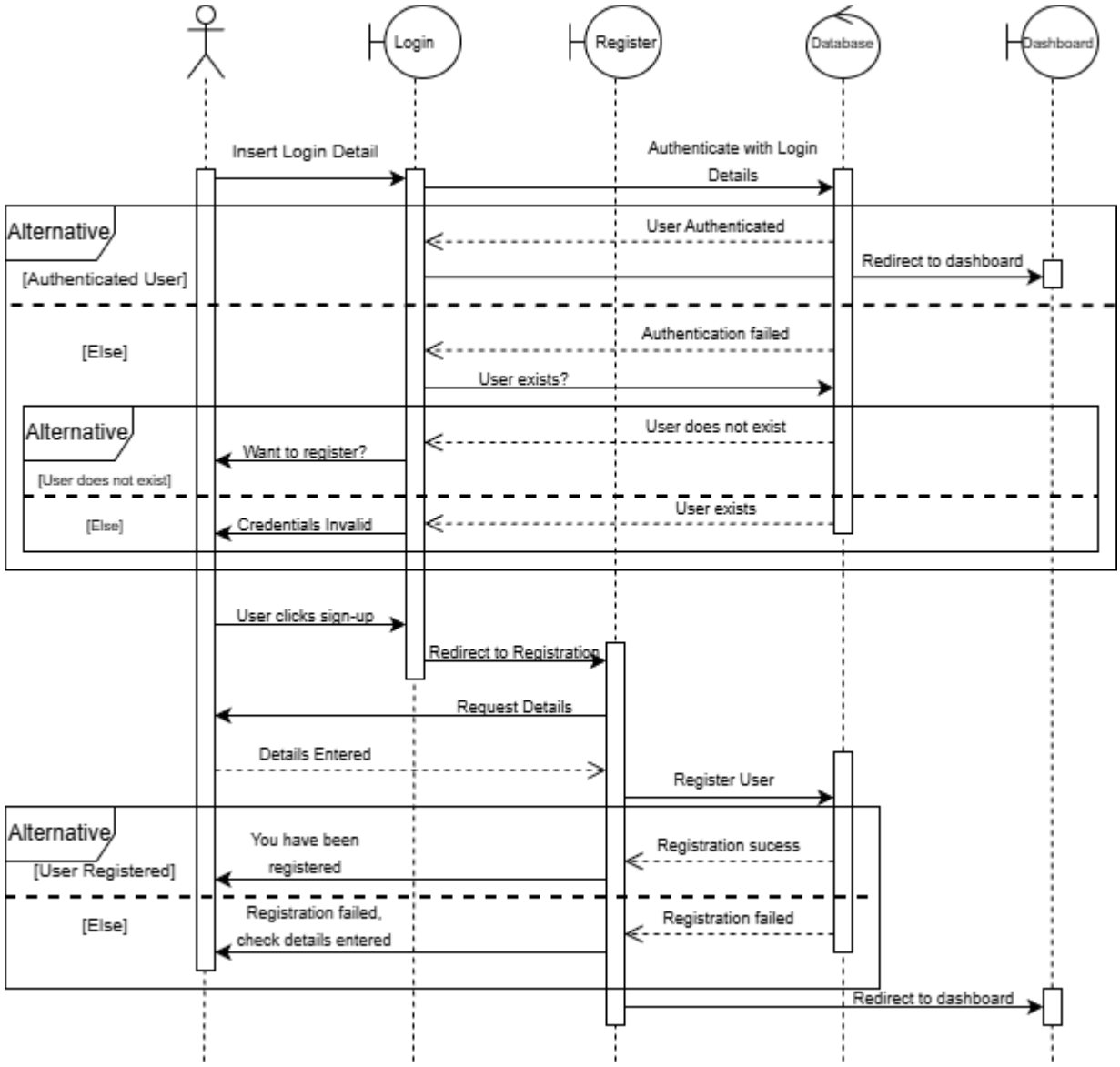
5.3. Activity Diagram



5.4. Class Diagram

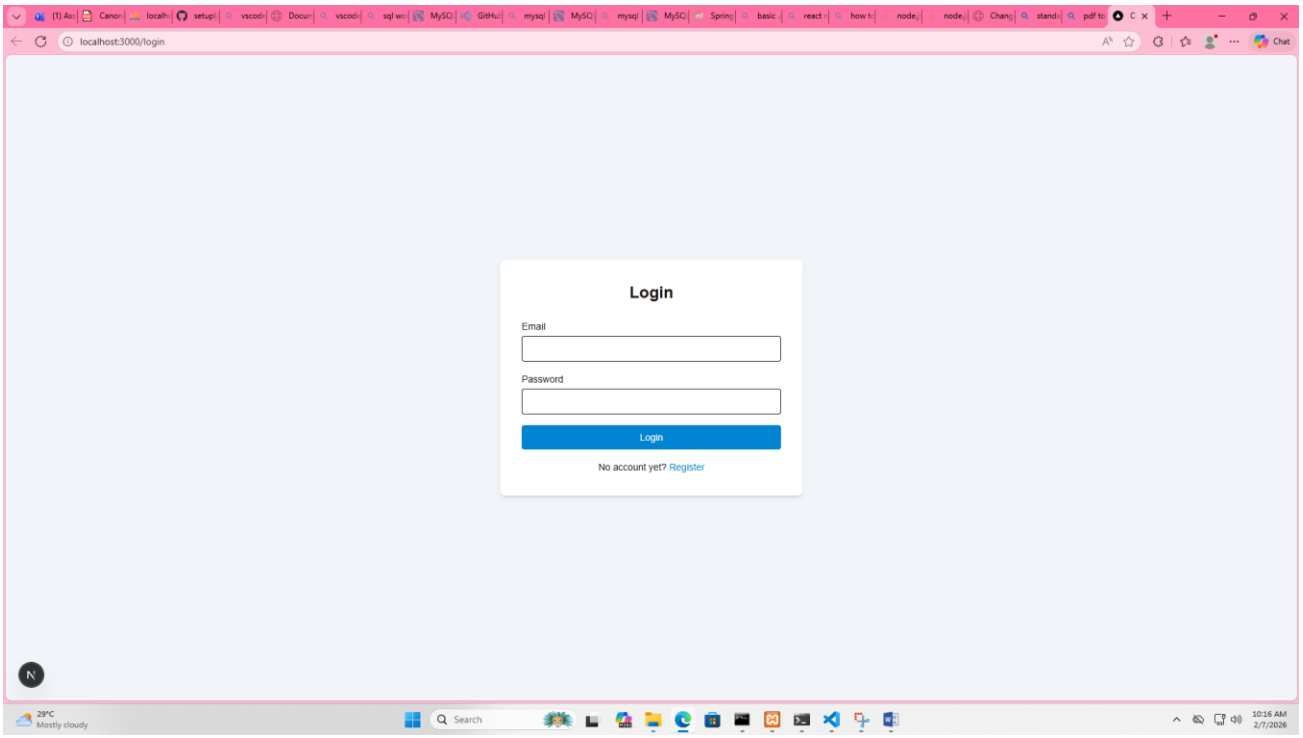


5.5. Sequence Diagram

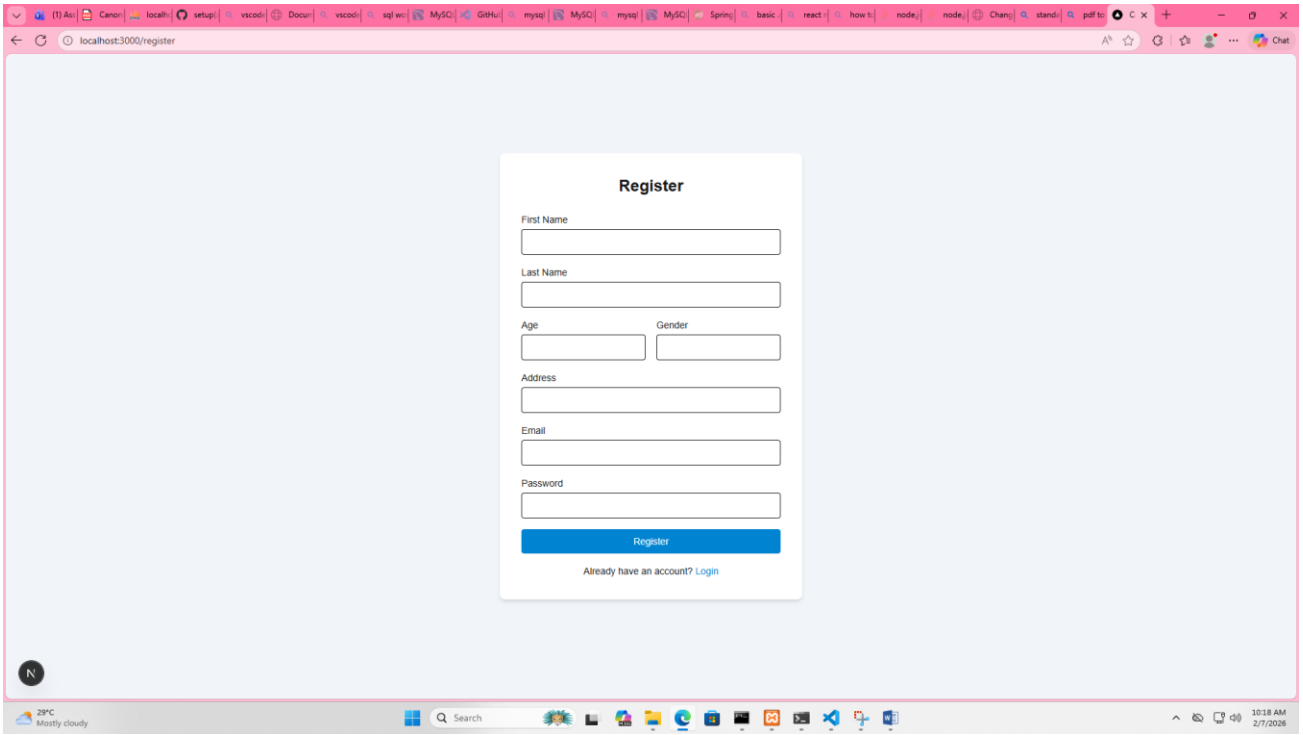


Screenshots of Web UI:

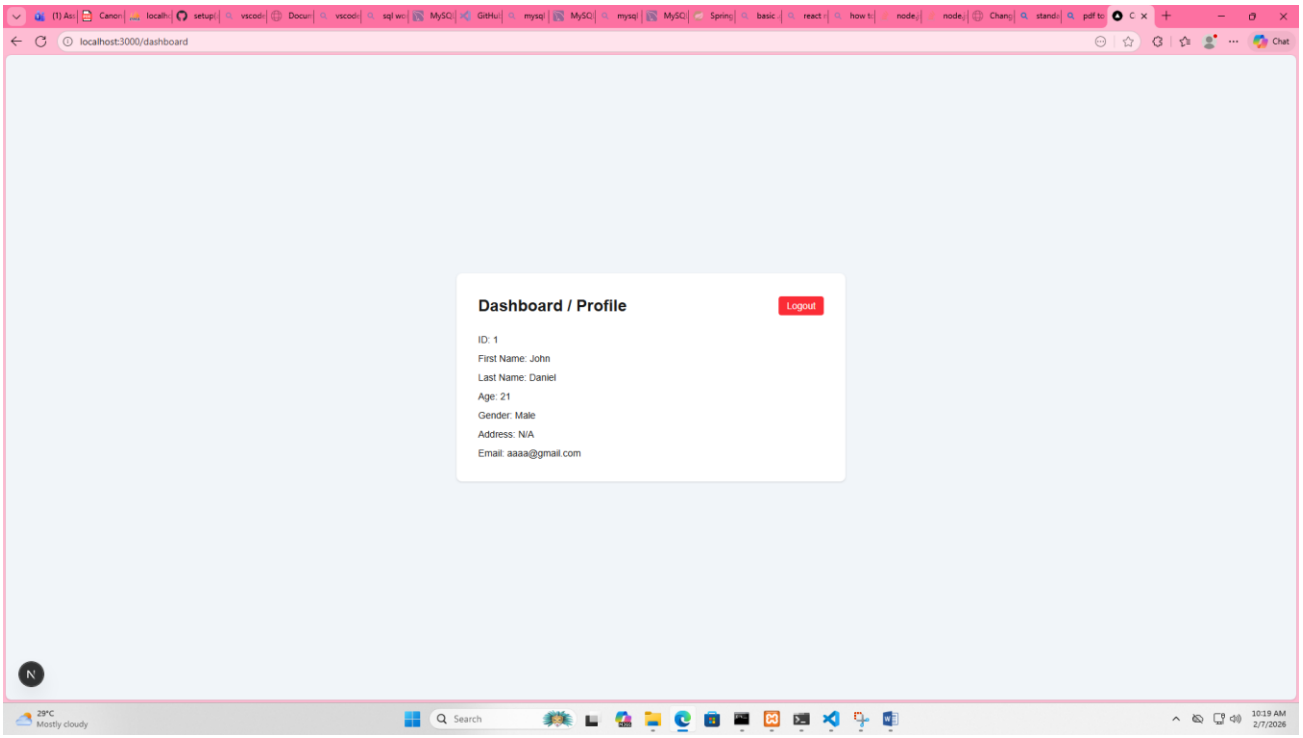
Login:



Register:



Dashboard/Profile with Logout:



6. Appendices

Include any additional information, references, or support materials.

Support Materials:

https://www.geeksforgeeks.org/sql/how-to-draw-entity-relationship-diagrams/

https://www.geeksforgeeks.org/system-design/use-case-diagram/

https://www.geeksforgeeks.org/system-design/unified-modeling-language-uml-activity-diagrams/

https://www.geeksforgeeks.org/system-design/unified-modeling-language-uml-class-diagrams/

https://www.geeksforgeeks.org/system-design/unified-modeling-language-uml-class-diagrams/