CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342-G2
SYSTEMS INTEGRATION AND ARCHITECTURE 1

FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

Project Title: User Registration Authentication

Prepared By: Rod Gabrielle M. Cañete

Date of Submission: 02/16/2026

Version: v3

Table of Contents

1. Introduction

1.1. Purpose

This System Requirements Document (SRD) defines the functional and non-functional requirements for the User Authentication module of the application. The primary purpose of this system is to provide a secure and reliable mechanism through which users can register new accounts, authenticate their identity via login, and safely terminate their sessions via logout.

This document is intended for the following audience: software developers responsible for implementation, project managers overseeing the development lifecycle, quality assurance engineers designing test cases, and stakeholders reviewing scope and feasibility prior to the scheduled face-to-face coding session.

1.2. Scope

The system under development is a User Authentication Module that will be integrated into the broader application platform. The module is responsible for three core operations: (1) allowing new users to create an account by providing required credentials, (2) enabling existing users to verify their identity and gain access to the system, and (3) permitting authenticated users to securely end their session.

The scope of this document is limited to the registration, login, and logout flows. Features such as password reset, two-factor authentication, OAuth third-party login, role-based access control, and user profile management fall outside the current scope and may be addressed in future iterations.

1.3. Definitions, Acronyms, and Abbreviations

Term / Acronym	Definition
SRD	System Requirements Document — this document.
Authentication	The process of verifying that a user is who they claim to be, typically via credentials such as an email and password.
Registration	The process by which a new user creates an account in the system by supplying required personal and credential information.
Session	A time-bound, server-side record that tracks an authenticated user's state after a successful login.
Session Token	A unique cryptographic string generated upon login and stored client-side (e.g., in a cookie or local storage) to identify an active session.

Credential	A combination of user-supplied data (e.g., email and password) used to verify identity during login.
Logout	The process of invalidating the current user session, thereby ending the authenticated state.
Input Validation	Server-side and/or client-side checks that ensure user-supplied data meets format and security requirements before processing.
API	Application Programming Interface — the set of endpoints the front-end communicates with to perform backend operations.
HTTPS	Hypertext Transfer Protocol Secure — ensures encrypted communication between the client and server.

2. Overall Description

2.1. System Perspective

The User Authentication Module operates as a self-contained component within a larger web application architecture. It sits between the user-facing front-end (browser or mobile client) and the application's back-end services. The front-end sends registration, login, and logout requests over HTTPS to the authentication API endpoints. The module, in turn, interacts with a user database to store new accounts, verify credentials, and manage session records.

The module does not operate in isolation. It depends on the availability of a relational database for persistent user storage, a secure transport layer (HTTPS) for all client-server communication, and — in future phases — an email service for verification. For this initial scope, however, the focus is on the core request-response flows between the client and the authentication API.

2.2. User Classes and Characteristics

User Class	Description	Characteristics
New User	An individual who does not yet have an account and intends to register.	Has no prior interaction with the system. May have limited technical literacy. Requires clear, guided registration form.

Registered User	An individual who has an existing account and intends to log in to access the system.	Familiar with the login interface. Expected to remember or retrieve their credentials. Primary interaction point for the system.
Authenticated User	A registered user who has successfully logged in and currently holds an active session.	Has access to protected resources. May initiate a logout at any time. Session is bound to a session token.

2.3. Operating Environment

The system is designed to operate within the following environment:

Client-Side: A modern web browser (Chrome, Firefox, Safari, Edge — latest two major versions) on desktop or mobile. The front-end interface will be built using standard web technologies (HTML, CSS, JavaScript).

Server-Side: A back-end application server capable of processing API requests, executing business logic, and enforcing input validation rules. The specific framework will be determined during the implementation phase.

Database: A relational database system to store user records (email, hashed password, account metadata) and active session information. The specific DBMS will be confirmed prior to coding.

Network: All communication between the client and the server must occur over HTTPS to ensure data is encrypted in transit.

2.4. Assumptions and Dependencies

The following assumptions and external dependencies have been identified for this module:

- It is assumed that the database server will be available and accessible to the back-end application at all times during development and production.
- It is assumed that all users will access the system through a web browser with JavaScript enabled.
- It is assumed that email verification during registration is out of scope for this phase and will be addressed in a subsequent iteration.
- The system depends on a functioning HTTPS certificate being properly configured on the server to secure all data in transit.
- It is assumed that password hashing and storage will follow industry-standard best practices (e.g., bcrypt or argon2). The specific hashing algorithm will be selected during implementation.
- The coding and testing phases are dependent on this requirements document being reviewed and approved prior to the face-to-face session.

3. System Features and Functional Requirements

3.1. Feature 1: User Registration

Description:

User Registration is the process by which a new user creates an account in the system. The user is presented with a registration form that collects the required information — specifically, a unique email address and a password. Upon successful submission, the system validates the input, checks for duplicate accounts, securely stores the new user's credentials, and confirms successful account creation to the user.

Functional Requirements:

Requirement ID	Requirement Description
FR-REG-01	The system shall present the user with a registration form containing fields for email address and password.
FR-REG-02	The system shall validate that the email field is not empty and conforms to a standard email format (e.g., user@domain.com) before submission is processed.
FR-REG-03	The system shall validate that the password field is not empty and meets a minimum length requirement of at least 8 characters.
FR-REG-04	The system shall check whether the submitted email address is already associated with an existing account. If a duplicate is found, the system shall reject the registration and display an appropriate error message.
FR-REG-05	The system shall securely hash the user's password using an industry-standard algorithm before storing it in the database. The plain-text password shall never be stored.
FR-REG-06	Upon successful registration, the system shall store the new user's account record (email and hashed password) in the database.

FR-REG-07	Upon successful registration, the system shall display a confirmation message indicating that the account has been created successfully.
FR-REG-08	If any validation error occurs during registration, the system shall display a clear and specific error message to the user indicating which field requires correction.

3.2. Feature 2: User Login

Description:

User Login is the process by which an existing, registered user verifies their identity and gains access to the system. The user is presented with a login form that collects their email address and password. The system validates the input, checks the provided credentials against the stored account records, and — upon successful verification — establishes an authenticated session for the user.

Functional Requirements:

Requirement ID	Requirement Description
FR-LOG-01	The system shall present the user with a login form containing fields for email address and password.
FR-LOG-02	The system shall validate that neither the email nor the password field is empty before processing the login request.
FR-LOG-03	The system shall retrieve the stored account record matching the submitted email address and compare the hashed version of the submitted password against the stored hash.
FR-LOG-04	If the email address does not match any existing account, the system shall reject the login attempt and display a generic error message (e.g., “Invalid email or password”) without revealing which specific field is incorrect.

FR-LOG-05	If the password does not match the stored hash for the given email, the system shall reject the login attempt and display the same generic error message as described in FR-LOG-04.
FR-LOG-06	Upon successful credential verification, the system shall create a new session record and generate a unique session token.
FR-LOG-07	The system shall transmit the session token to the client in a secure manner (e.g., via an HttpOnly cookie or an equivalent secure mechanism) to maintain the authenticated state.
FR-LOG-08	Upon successful login, the system shall redirect the user or display a confirmation indicating that they are now authenticated.

3.3. Feature 3: User Logout

Description:

User Logout is the process by which an authenticated user terminates their current session. When the user initiates a logout action, the system invalidates the active session on the server side, clears the session token from the client, and returns the user to an unauthenticated state. This ensures that no further authenticated requests can be made using the terminated session.

Functional Requirements:

Requirement ID	Requirement Description
FR-LGO-01	The system shall provide a clearly visible and accessible logout option to any authenticated user within the interface.
FR-LGO-02	Upon the user initiating a logout action, the system shall send a logout request to the server, including the current session token for identification.

FR-LGO-03	The system shall invalidate the session record associated with the submitted session token on the server side, ensuring the session can no longer be used for authentication.
FR-LGO-04	The system shall instruct the client to clear or remove the session token from its storage (e.g., clear the cookie or remove the token from local storage).
FR-LGO-05	Upon successful logout, the system shall redirect the user to the login page or a public landing page and display a message confirming that the user has been logged out.
FR-LGO-06	If a user attempts to access a protected resource after logout (i.e., with an invalidated or missing session token), the system shall deny access and redirect the user to the login page.

4. Non-Functional Requirements

Non-functional requirements define the quality attributes and operational constraints that the authentication system must satisfy. These requirements govern how the system performs, not just what it does.

ID	Attribute	Requirement
NFR-01	Security	All user credentials must be transmitted exclusively over HTTPS. Passwords must be hashed using a strong, industry-standard algorithm (e.g., bcrypt, argon2) before storage. Plain-text passwords must never be persisted. Session tokens must be cryptographically secure and sufficiently random to prevent guessing or brute-force attacks.
NFR-02	Usability	The registration and login forms must be intuitive and clearly labeled. Error messages must be specific enough for users to understand and correct their input without revealing sensitive system details. The interface must be responsive and usable on both desktop and mobile devices.

NFR-03	Performance	The system shall process a login or registration request and return a response within 2 seconds under normal load conditions. The system shall be capable of handling concurrent user sessions without noticeable degradation in response time.
NFR-04	Reliability	The authentication module shall maintain consistent behavior across supported browsers. If the server or database becomes temporarily unavailable, the system shall display a user-friendly error message rather than an unhandled exception or crash.
NFR-05	Maintainability	The authentication logic (validation, hashing, session management) shall be modular and separated from the front-end presentation layer. Code and configurations shall follow established project conventions to facilitate future updates and debugging.
NFR-06	Scalability	The session management design shall support a growing number of concurrent users without requiring significant architectural changes. The database schema for user accounts and sessions shall accommodate future expansion (e.g., additional user fields or authentication methods).

5. System Models (Diagrams)

5.1. ERD

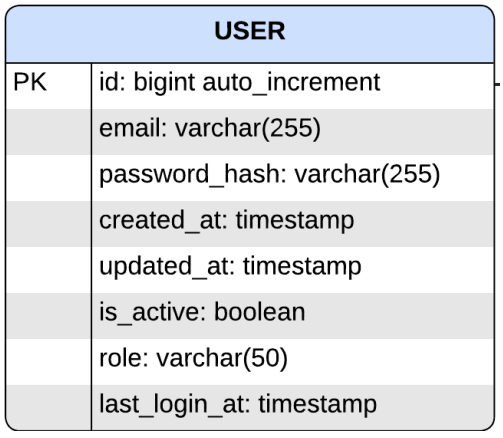


Figure 5.1 — *Database Schema — Users*

5.2. Use Case Diagram

The Use Case Diagram identifies two actor types and the actions available to each within the Authentication System boundary. A Guest User — someone without an existing account — can Register or Login. An Authenticated User — someone who has successfully logged in — can View Profile and Logout. The «include» relationships indicate that viewing a profile and logging out are accessible only after a successful login establishes an active session.

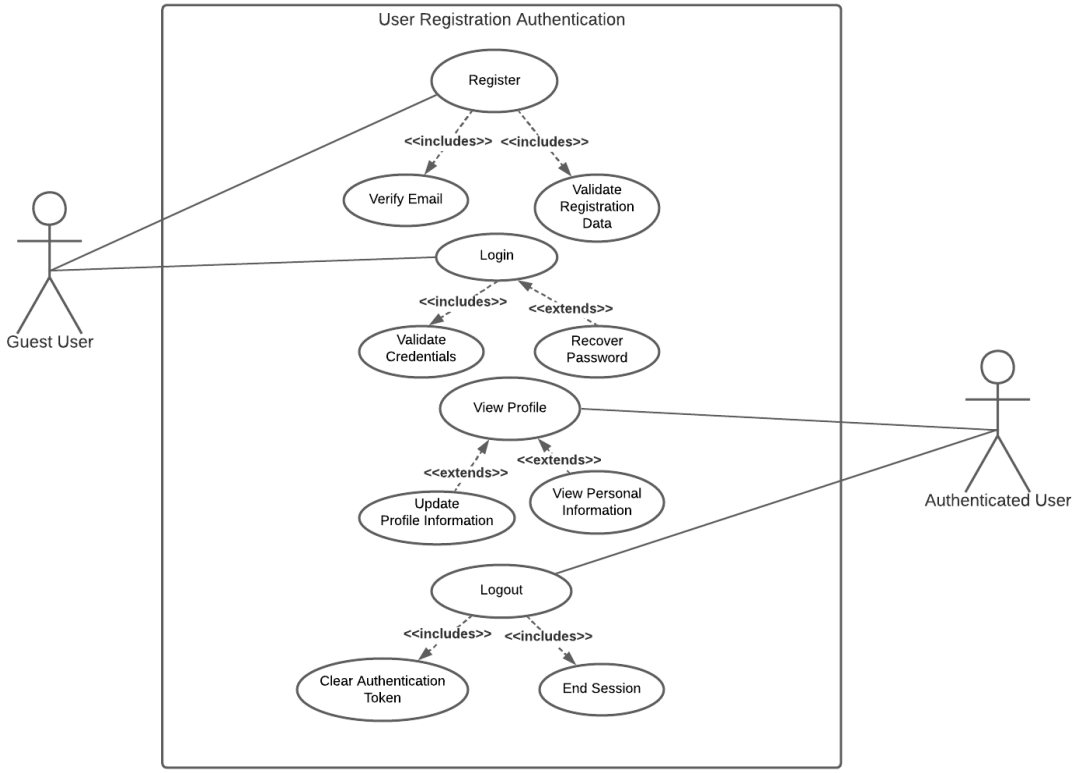


Figure 5.2 — *Use Case Diagram — Guest User & Authenticated User interactions*

5.3. Activity Diagram

The Activity Diagram presents all three flows — Registration, Login, and Logout — side by side in colour-coded swim lanes. Each lane traces the complete sequence of actions, decision points, and possible error states from start to end. Decision diamonds indicate validation gates: if input is invalid or credentials fail, the flow branches to an error state and loops back. Only paths that clear every validation gate proceed to the final success outcome.

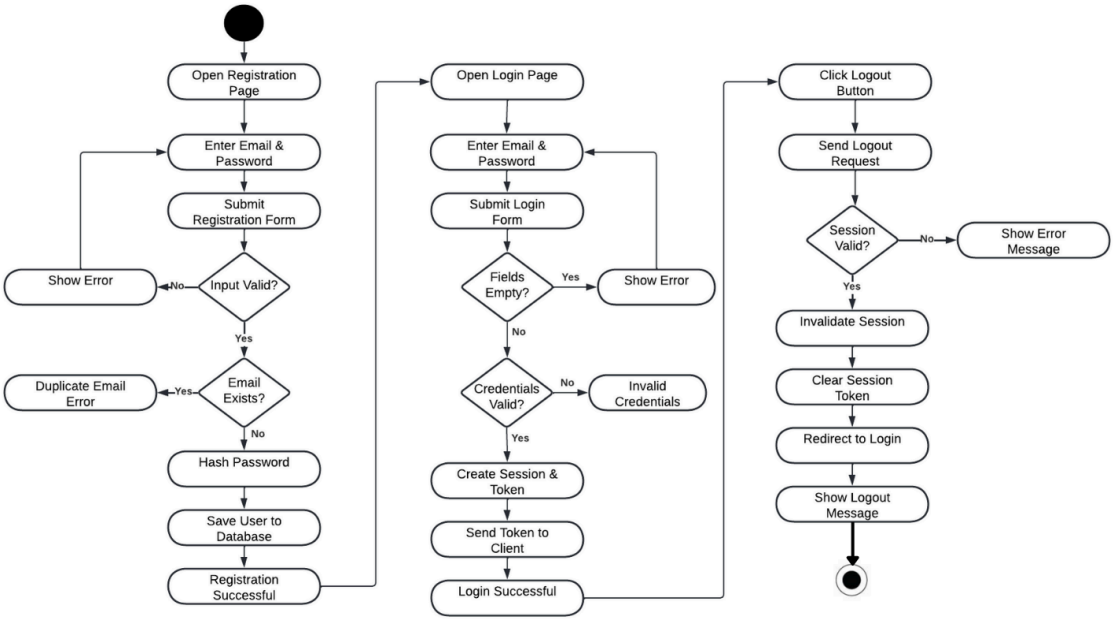


Figure 5.3 — *Activity Diagram — Registration, Login & Logout flows with decision points*

5.4. Class Diagram

The Class Diagram maps the expected backend class structure following a layered architecture pattern typical of Spring Boot applications. The Controller layer (AuthController) receives HTTP requests and delegates to the Service layer (AuthService), which encapsulates all business logic. The Service depends on the Repository layer (UserRepository interface) for database operations, and on two utility classes — PasswordEncoder for secure password hashing via bcrypt, and TokenProvider for session token generation and validation. The User entity represents the database-mapped model class.

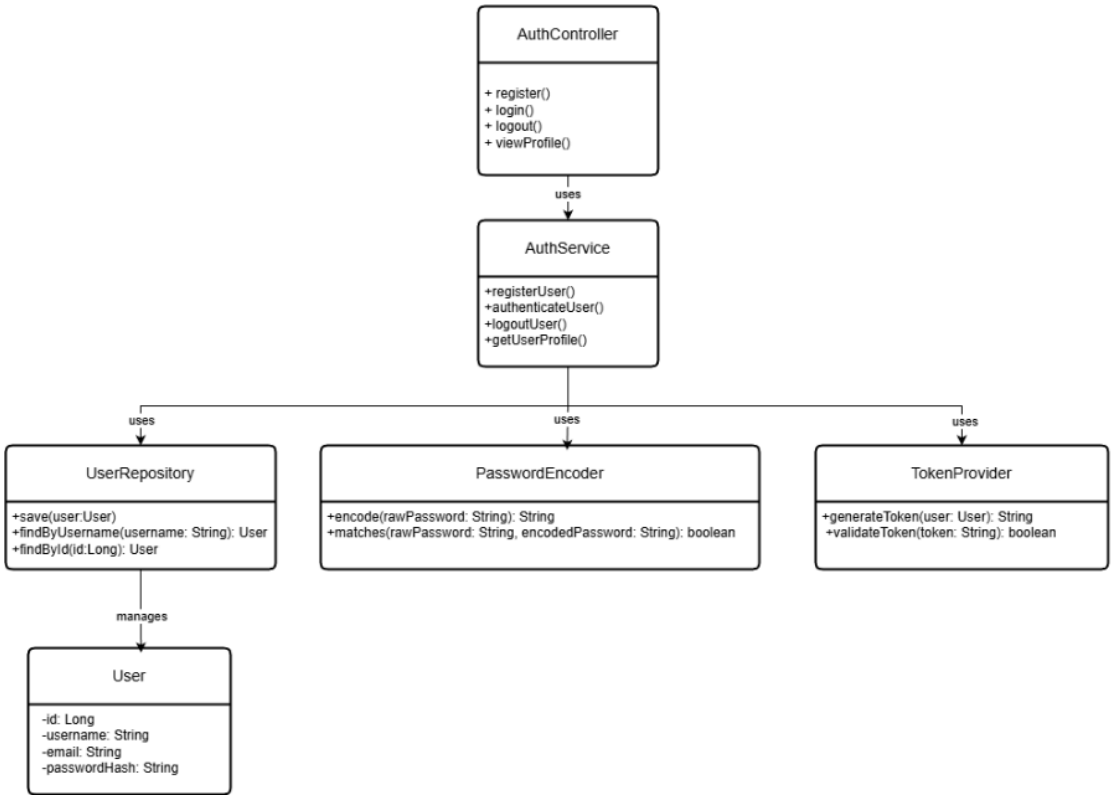


Figure 5.4 — *Class Diagram — Controller → Service → Repository + Utility classes*

5.5. Sequence Diagram

The Sequence Diagram illustrates the step-by-step interaction between the three system participants — React UI (client), Spring Boot API (server), and the Database — for each of the three flows. Solid arrows represent outgoing requests; dashed arrows represent responses. Self-calls on the Spring Boot lifeline indicate internal method invocations such as input validation, password hashing, and token generation. Each flow is enclosed in a labelled section boundary for clarity.

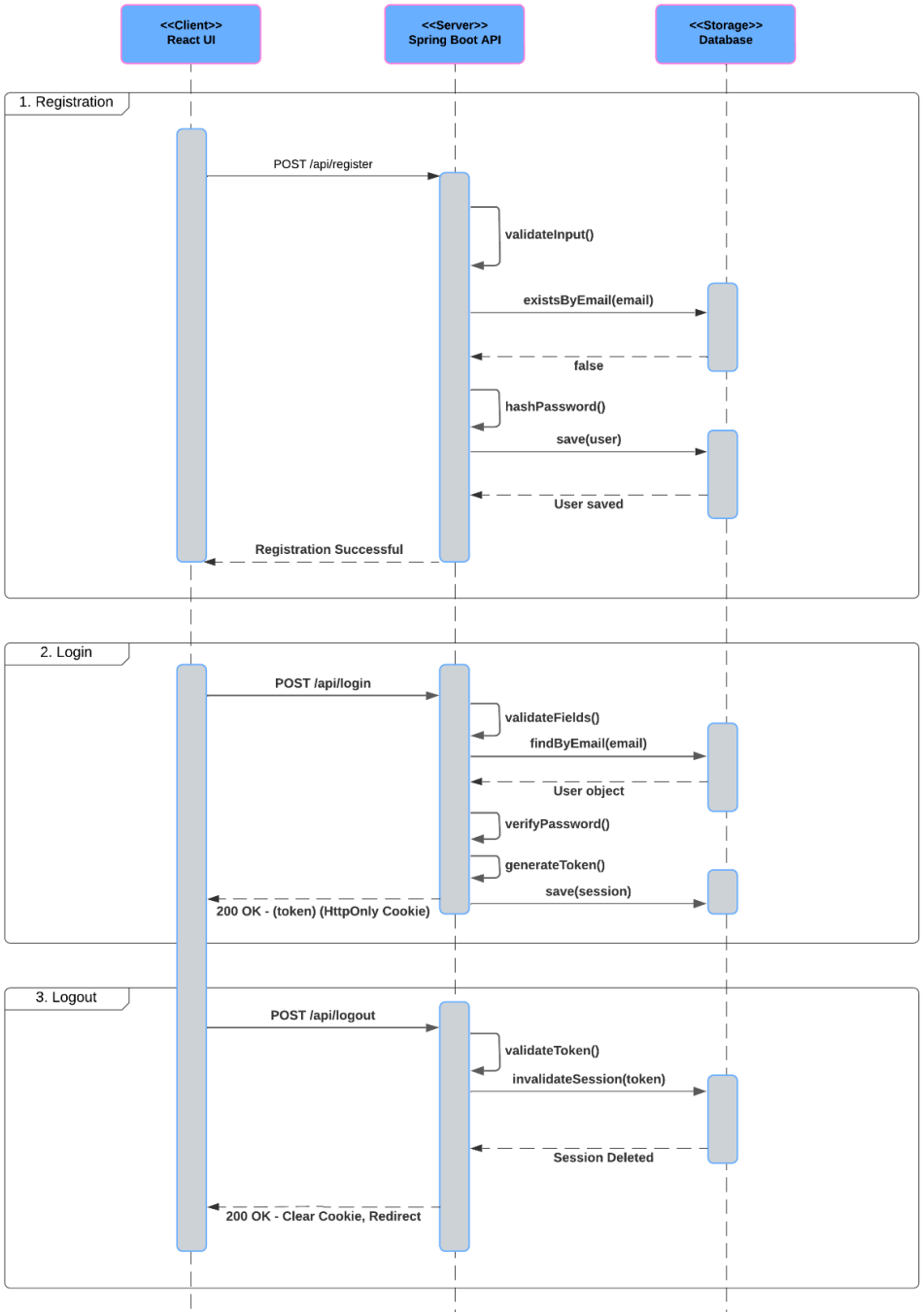


Figure 5.5 — *Sequence Diagram — Register, Login & Logout interactions across all layers*

6. Appendices

Appendix A: API Endpoint Specifications

- **POST /api/register**
  - Request body schema (JSON)
  - Response codes (200, 400, 409, 500)
  - Example request/response
- **POST /api/login**
  - Request/response format
  - Cookie/token structure
- **POST /api/logout**
  - Authentication requirements
  - Response format

Appendix B: Error Code Reference

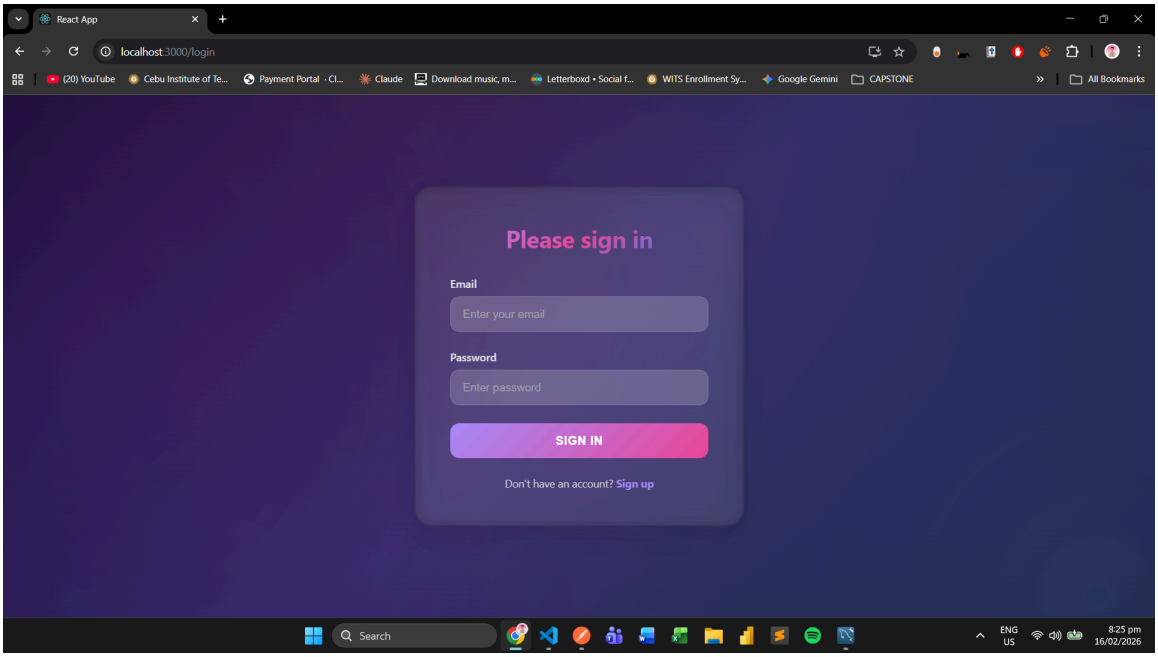
Code	Message	Cause
400	Invalid email format	Email validation failed
409	Email already exists	Duplicate registration
401	Invalid credentials	Login failed

Appendix C: Technology Stack

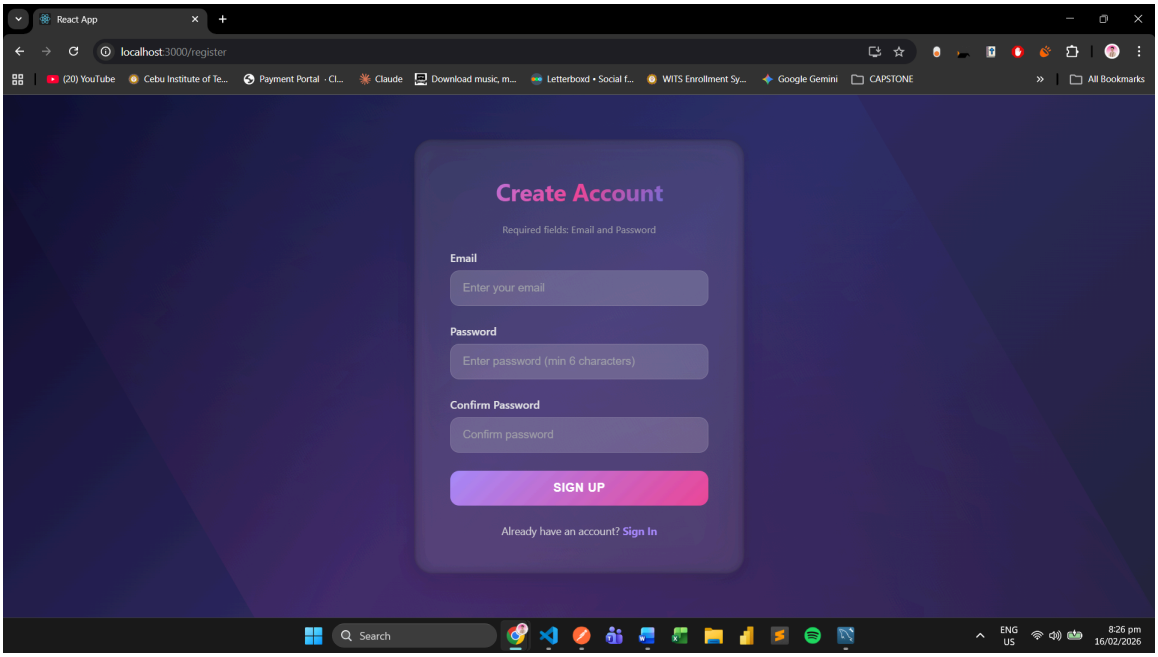
- Frontend: React
- Backend: Spring Boot
- Database: MySQL / Supabase
- Tools: Maven/Gradle, Postman

7. Web Screenshots:

Login Form:



Registration Form:



Dashboard and Logout: