CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342-G1
SYSTEMS INTEGRATION AND ARCHITECTURE 1

FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

Project Title: Mini App – User Registration & Authentication

Prepared By: Melody Ann M. Garbo

Date of Submission: February 14, 2026

Version: 1.2

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this document is to provide a detailed description of the User Authentication System. It outlines the functional and non-functional requirements to guide the development and testing phases. The intended audience includes project stakeholders and developers.

1.2. Scope

This system manages user registration, login, and logout processes. It provides a secure way for users to create accounts and access a private profile dashboard. The system boundaries are limited to account management and session control. It does not include password recovery or third party social logins at this stage.

1.3. Definitions, Acronyms, and Abbreviations

API: Application Programming Interface used for communication between the frontend and backend.

DTO: Data Transfer Object used to pass data between the UI and the server.

JWT: JSON Web Token used for secure session management.

UI: User Interface where the person interacts with the application.

Bcrypt: A password hashing function used to secure user credentials.

2. Overall Description

2.1. System Perspective

The system acts as a security gateway for a web application. It functions as a standalone module that connects a React frontend to a Spring Boot backend and a relational database. It ensures that only registered users can access specific protected resources within the application environment.

2.2. User Classes and Characteristics

- **Guest User**: An unidentified visitor who can only access the registration and login pages.
- **Authenticated User**: A person who has successfully logged in and can view their profile and perform a logout.

2.3. Operating Environment

- **Hardware**: Standard computer with internet connectivity.
- **Software**: Modern web browsers such as Chrome, Firefox, or Edge.
- **Backend**: Java 17 or higher with the Spring Boot framework.
- **Database**: PostgreSQL or MySQL for data storage.

2.4. Assumptions and Dependencies

It is assumed that users have a valid email address for registration. The system depends on the availability of a database server and a stable network connection to process requests. It is also assumed that the browser used supports local storage for keeping the session token.

3. System Features and Functional Requirements

3.1. Feature 1: User Registration

Description: Allows new users to create a unique account by providing their personal information.

Functional Requirements:

- The system shall allow users to input their first name, last name, email, password, and gender.
- The system shall verify if the email address is already in use before creating a new account.
- The system shall encrypt the user password using Bcrypt before saving it to the database.

3.2. Feature 2: User Authentication (Login and Logout)

Description: Verifies user identity and manages active sessions.

Functional Requirements:

- The system shall allow a user to log in using a registered email and password.

- The system shall generate a secure JWT token upon successful authentication.

- The system shall restrict access to the Profile Dashboard if a valid token is not present.

- The system shall invalidate the session and redirect the user to the login page when the logout button is clicked.

3.3. Feature 3: Profile Dashboard

Description: Provides a secure area for authenticated users to view their account details.

Functional Requirements:

- The system shall retrieve and display the user's first name, last name, email, and gender upon successful login.

- The system shall verify the validity of the JWT token before granting access to the dashboard.

- The system shall redirect unauthenticated users to the login page if they attempt to access the dashboard URL directly.

- The system shall ensure that a user can only view their own personal information and not the data of other users.

4. Non-Functional Requirements

Security: All passwords must be hashed. The system must use JWT to protect private routes from unauthorized access.

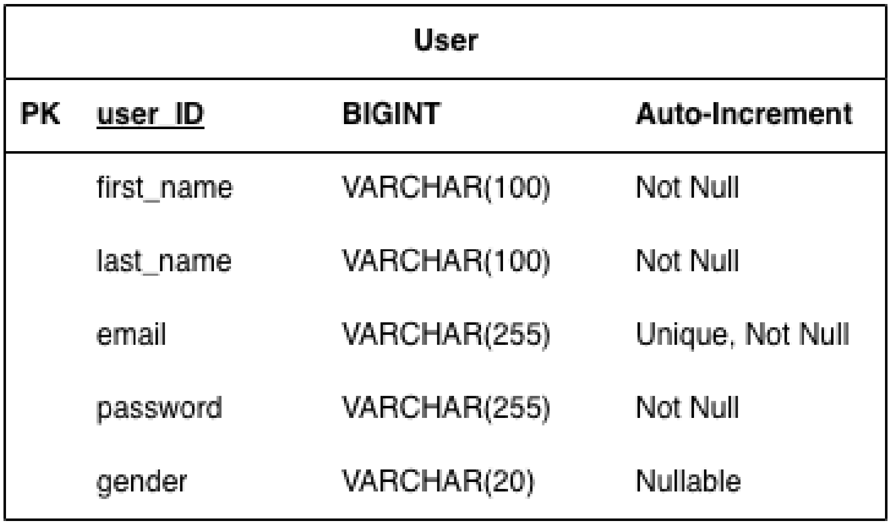
Usability: The registration and login forms must provide clear error messages for incorrect inputs.

Performance: The system should complete the authentication process and load the profile dashboard in less than two seconds.

Reliability: The database must maintain data integrity and ensure that user records are not lost during server restarts.

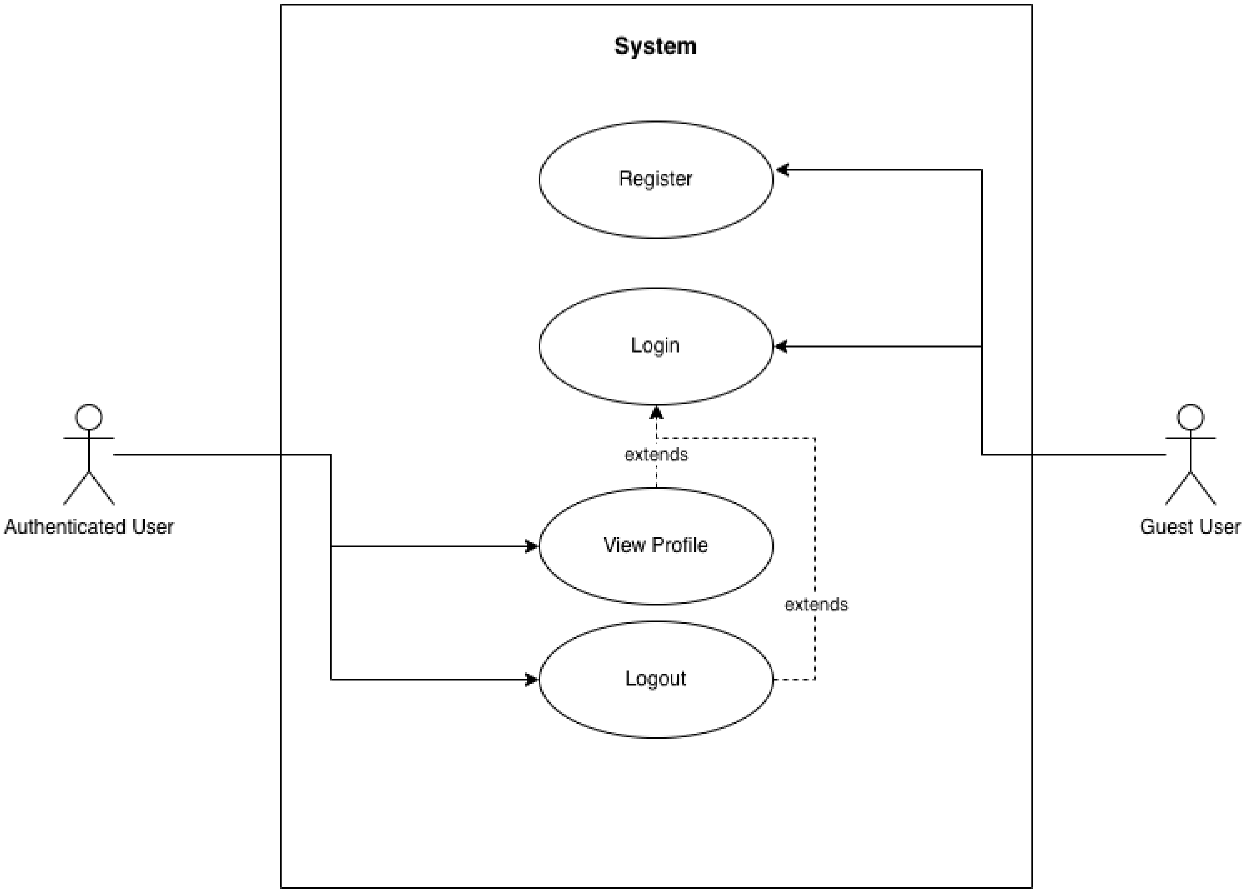
5. System Models (Diagrams)

5.1. ERD

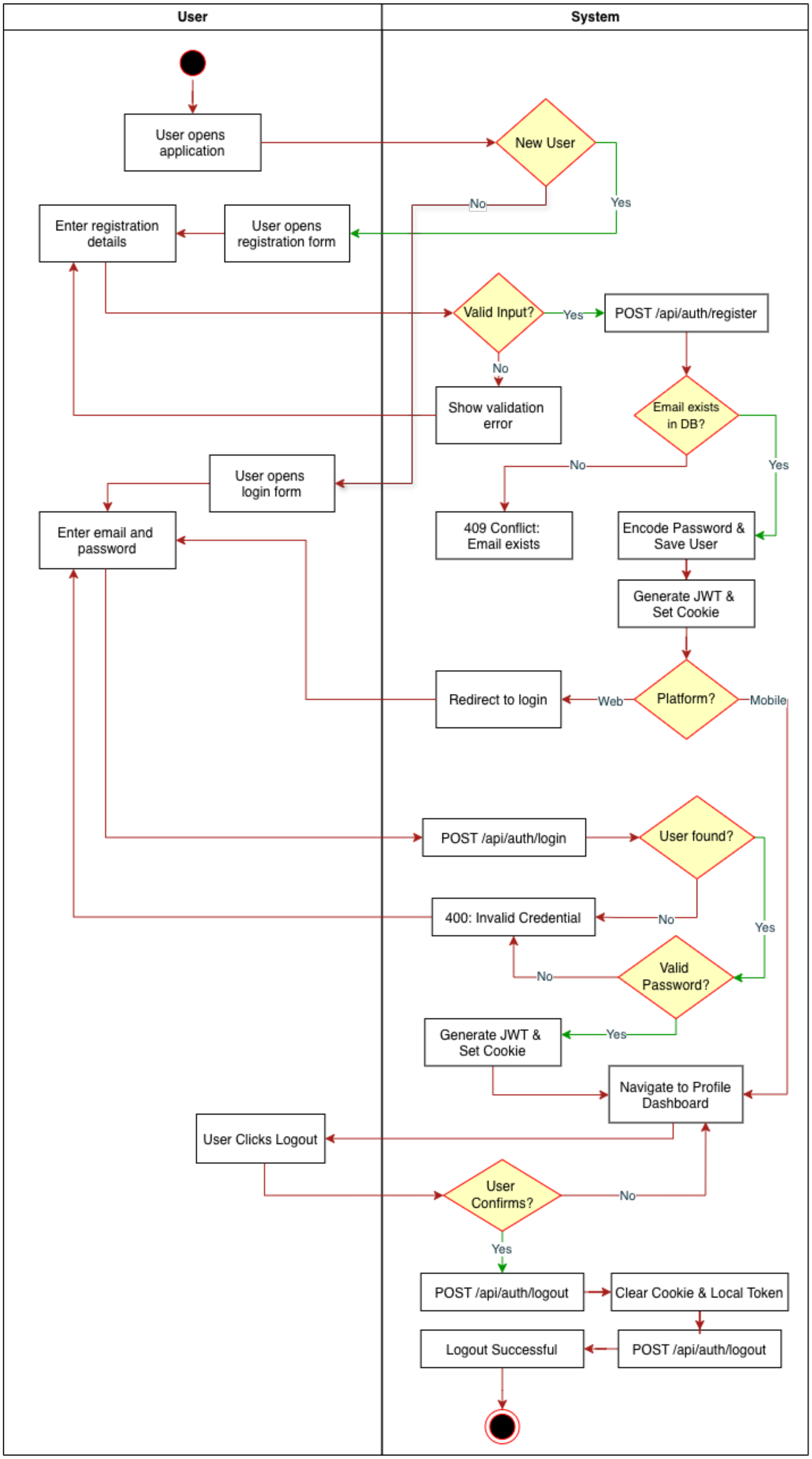


	User		
PK	**user_ID**	**BIGINT**	**Auto-Increment**
	first_name	VARCHAR(100)	Not Null
	last_name	VARCHAR(100)	Not Null
	email	VARCHAR(255)	Unique, Not Null
	password	VARCHAR(255)	Not Null
	gender	VARCHAR(20)	Nullable

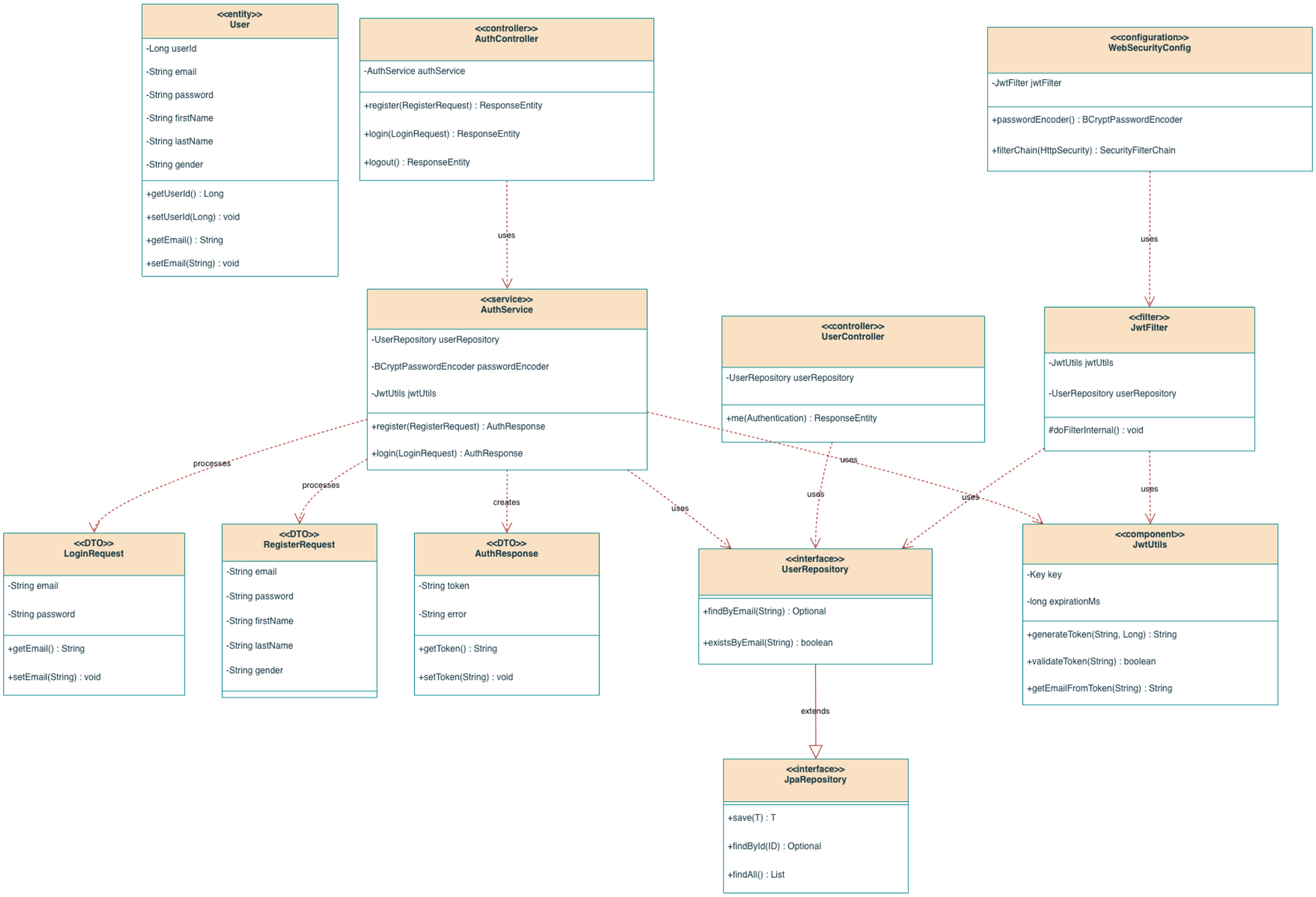
5.2. Use Case Diagram



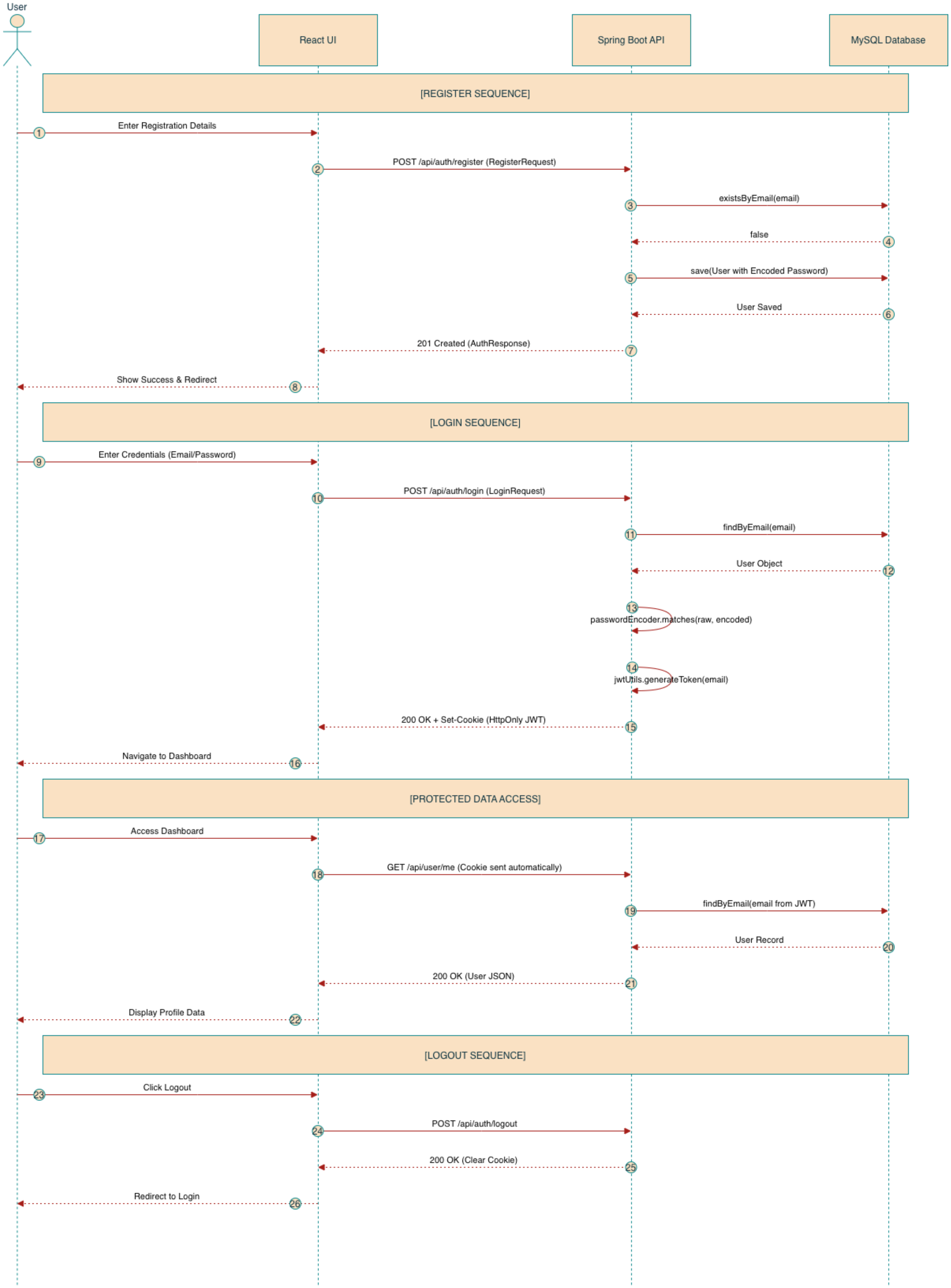
5.3. Activity Diagram



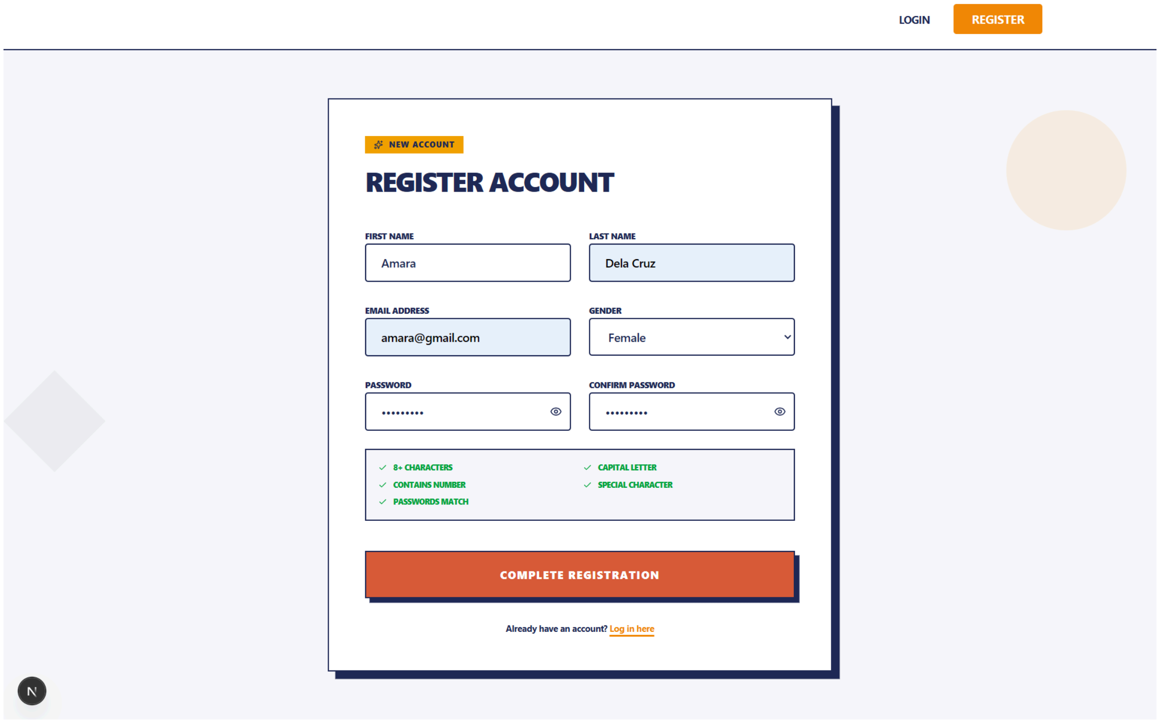
5.4. Class Diagram



5.5. Sequence Diagram


User
React UI
Spring Boot API
MySQL Database
[REGISTER SEQUENCE]
1 Enter Registration Details
2 POST /api/auth/register (RegisterRequest)
3 existsByEmail(email)
4 false
5 save(User with Encoded Password)
6 User Saved
7 201 Created (AuthResponse)
8 Show Success & Redirect
[LOGIN SEQUENCE]
9 Enter Credentials (Email/Password)
10 POST /api/auth/login (LoginRequest)
11 findByEmail(email)
12 User Object
13 passwordEncoder.matches(raw, encoded)
14 jwtUtils.generateToken(email)
15 200 OK + Set-Cookie (HttpOnly JWT)
16 Navigate to Dashboard
[PROTECTED DATA ACCESS]
17 Access Dashboard
18 GET /api/user/me (Cookie sent automatically)
19 findByEmail(email from JWT)
20 User Record
21 200 OK (User JSON)
22 Display Profile Data
[LOGOUT SEQUENCE]
23 Click Logout
24 POST /api/auth/logout
25 200 OK (Clear Cookie)
26 Redirect to Login

5.6. Web - Register Page



REGISTER ACCOUNT

FIRST NAME: Amara

LAST NAME: Dela Cruz

EMAIL ADDRESS: amara@gmail.com

GENDER: Female

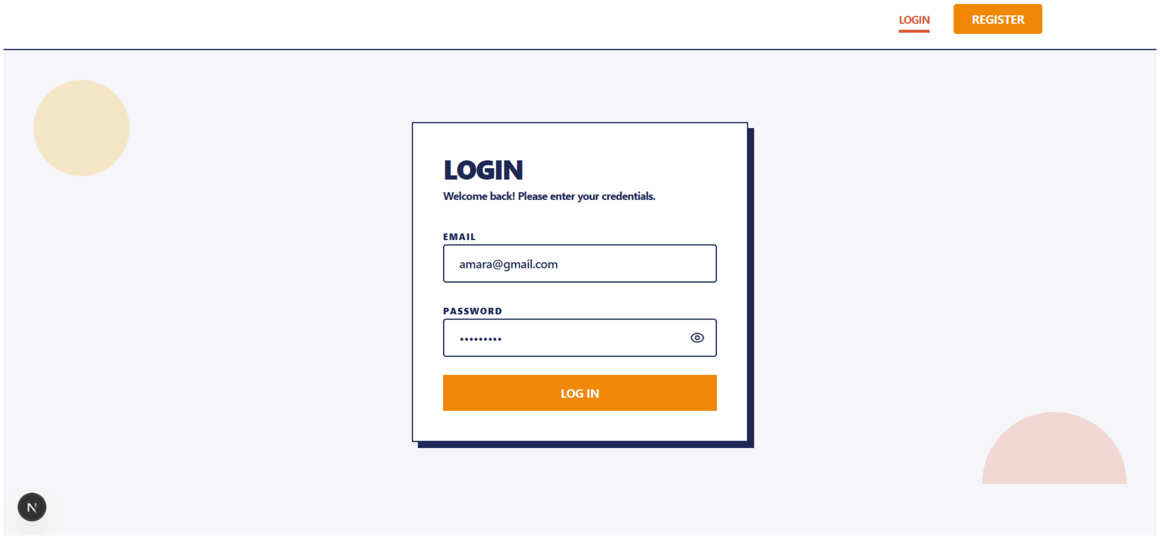
PASSWORD: •••••••••

CONFIRM PASSWORD: •••••••••

✓ 8+ CHARACTERS
✓ CAPITAL LETTER
✓ CONTAINS NUMBER
✓ SPECIAL CHARACTER
✓ PASSWORDS MATCH

Already have an account? Log in here

5.7. Web - Login Page



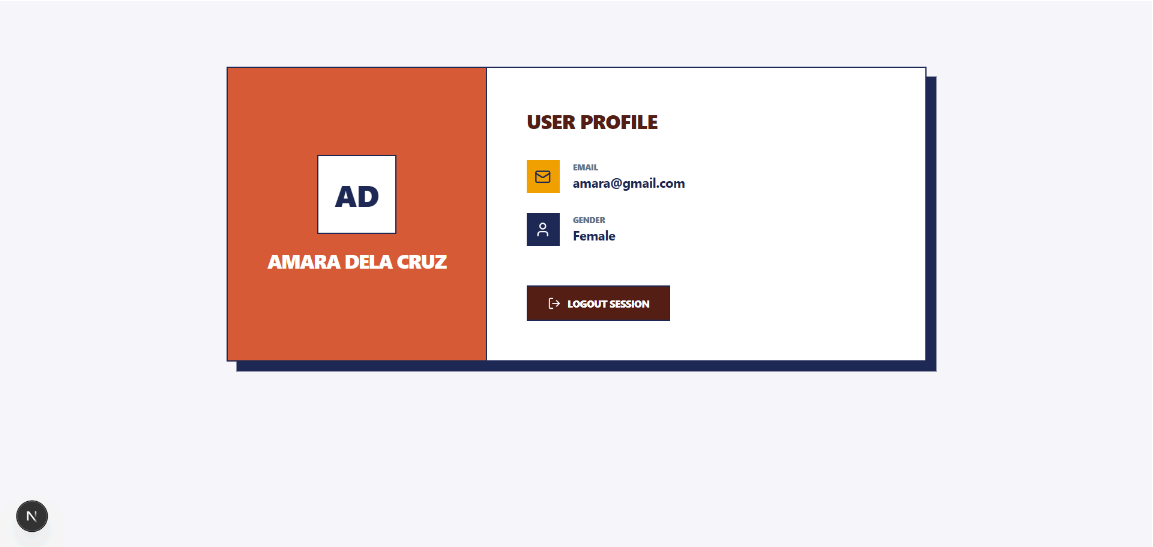
LOGIN

Welcome back! Please enter your credentials.

EMAIL: amara@gmail.com

PASSWORD: •••••••••

5.8. Web - Profile Dashboard



AD

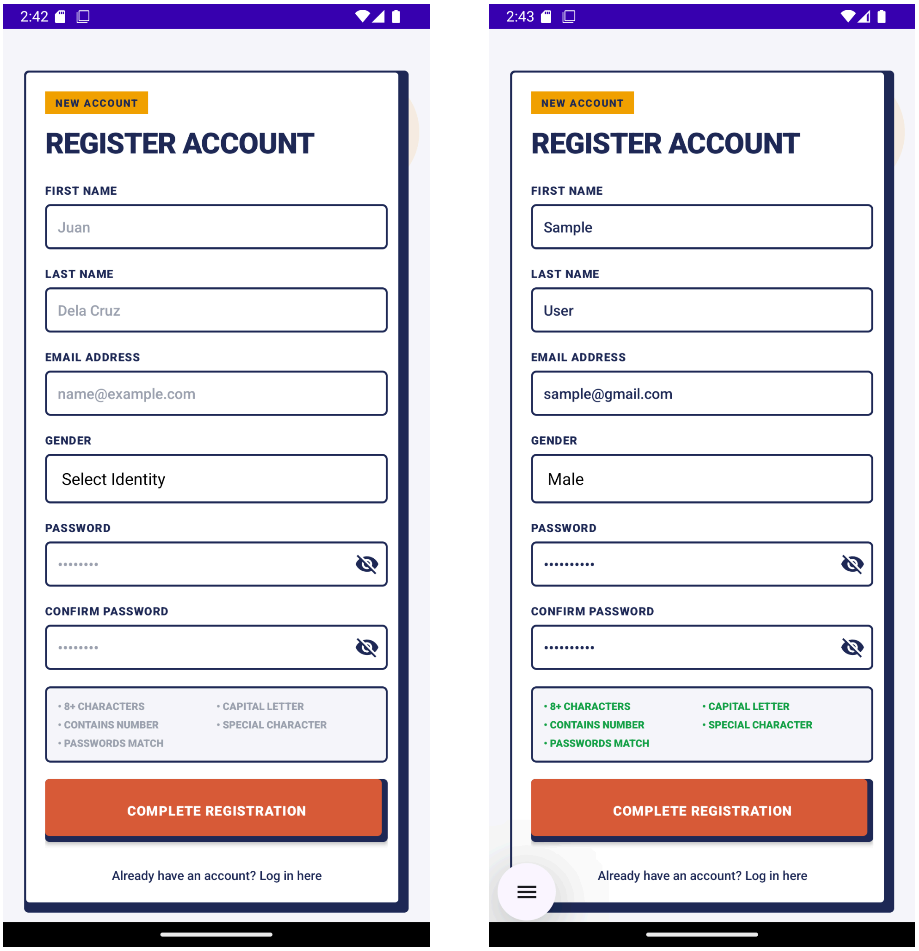
AMARA DELA CRUZ

USER PROFILE

EMAIL
amara@gmail.com

GENDER
Female

5.9. Mobile - Register Page



REGISTER ACCOUNT

FIRST NAME
Juan

LAST NAME
Dela Cruz

EMAIL ADDRESS
name@example.com

GENDER
Select Identity

PASSWORD
••••••••

CONFIRM PASSWORD
••••••••

• 8+ CHARACTERS • CAPITAL LETTER
• CONTAINS NUMBER • SPECIAL CHARACTER
• PASSWORDS MATCH

Already have an account? Log in here

REGISTER ACCOUNT

FIRST NAME
Sample

LAST NAME
User

EMAIL ADDRESS
sample@gmail.com

GENDER
Male

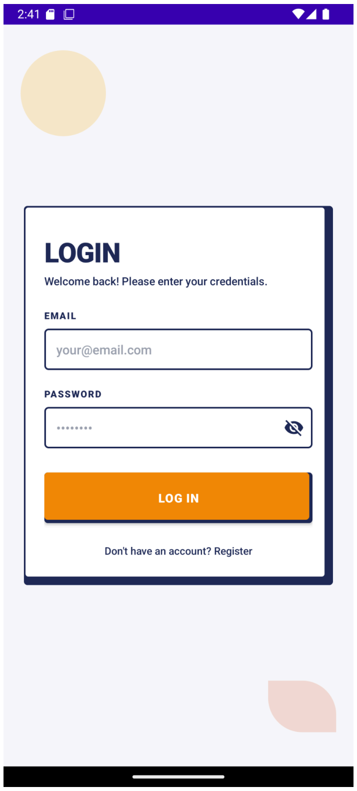
PASSWORD
••••••••••

CONFIRM PASSWORD
••••••••••

• 8+ CHARACTERS • CAPITAL LETTER
• CONTAINS NUMBER • SPECIAL CHARACTER
• PASSWORDS MATCH

Already have an account? Log in here

5.10. Mobile - Login Page



LOGIN

Welcome back! Please enter your credentials.

EMAIL

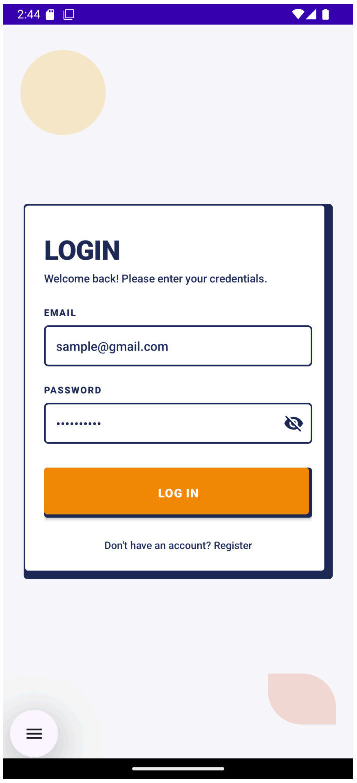
your@email.com

PASSWORD

••••••••

LOG IN

Don't have an account? Register



LOGIN

Welcome back! Please enter your credentials.

EMAIL

sample@gmail.com

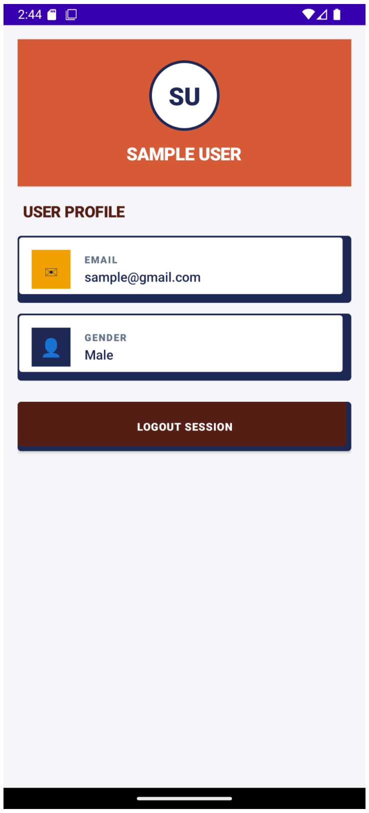
PASSWORD

••••••••••

LOG IN

Don't have an account? Register

5.11. Mobile - Profile Dashboard



SU

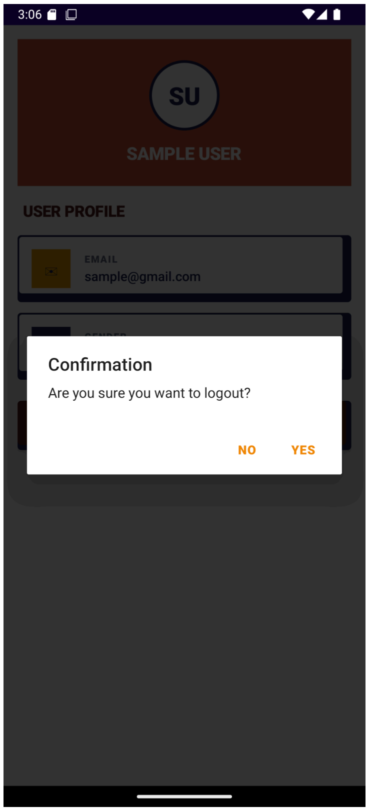
SAMPLE USER

USER PROFILE

EMAIL
sample@gmail.com

GENDER
Male

LOGOUT SESSION



Confirmation

Are you sure you want to logout?

NO YES

6. Appendices

Appendix A: Technical Diagrams

The following diagrams serve as the visual architectural guide for the system implementation:

- **Database Schema (ERD)**: Defines the User table structure with a BIGINT userId primary key and unique constraints for the email field.
- **Activity Diagram**: Illustrates the dynamic flow of the system, showing the decision logic for registration and login, and the path toward accessing the protected Profile Dashboard.
- **Class Diagram**: Illustrates the 3-tier Spring Boot architecture including AuthController, AuthService, and UserRepository, as well as the specialized TokenProvider and PasswordEncoder components.
- **Sequence Diagram**: Outlines the step-by-step communication between the React UI, Spring Boot API, and the Database for Registration, Login, and Logout events.

Appendix B: API Endpoint Summary

The backend will expose the following REST endpoints for the React frontend to consume:

- **POST /api/auth/register**: Accepts a RegisterRequest object and returns the created user or an error message.
- **POST /api/auth/login**: Accepts a LoginRequest object and returns a JWT token upon successful credentials verification.
- **POST /api/auth/logout**: Clears the authentication context for the current session.
- **GET /api/user/profile**: A protected endpoint that returns the authenticated user's data; requires a valid JWT in the Authorization header.