CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342-Section
SYSTEMS INTEGRATION AND ARCHITECTURE 1

FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

Project Title: Mini App – User Registration & Authentication

Prepared By: Kylene J. Arong

Date of Submission: 02/03/2026

Version: 1.0

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this document is to describe the design and functionality of a mini user authentication system. This document is intended for students and instructors and will serve as a reference for system implementation.

1.2. Scope

The system allows users to register, log in, view their profile, and log out. It includes basic authentication features and restricts access to protected pages for unauthenticated users.

1.3. Definitions, Acronyms, and Abbreviations

User – A person who interacts with the system
Authentication – The process of verifying a user's identity
JWT – JSON Web Token used for authentication
UI – User Interface
API – Application Programming Interface

2. Overall Description

2.1. System Perspective

The system is a standalone authentication module that can be integrated into a larger web or mobile application. It follows a client-server architecture.

2.2. User Classes and Characteristics

Guest User – Can register and log in
Authenticated User – Can view profile and log out

2.3. Operating Environment

Web browser (for React UI)

Backend server using Spring Boot

Relational database for user data storage

2.4. Assumptions and Dependencies

Users have internet access

The system depends on a backend server and database

Authentication tokens are securely stored on the client side

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 an account.

Functional Requirements:

- The system shall allow users to register using a username, email, and password

- The system shall prevent duplicate usernames or emails
- The system shall store passwords in encrypted form

3.2. Feature 2: User Login

Description: Allows registered users to log in to the system.

Functional Requirements:

- The system shall authenticate users using valid credentials
- The system shall deny access for invalid login attempts
- The system shall generate an authentication token upon successful login

4. Non-Functional Requirements

Security: Passwords must be encrypted

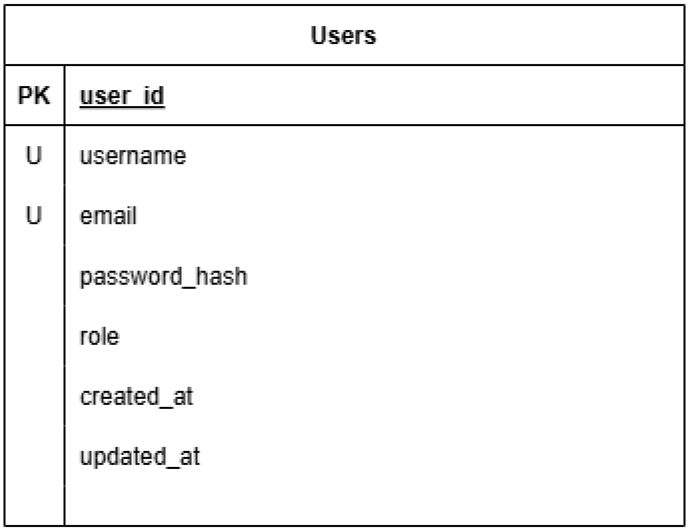
Usability: The system should be easy to use

Performance: Authentication responses should be fast

5. System Models (Diagrams)

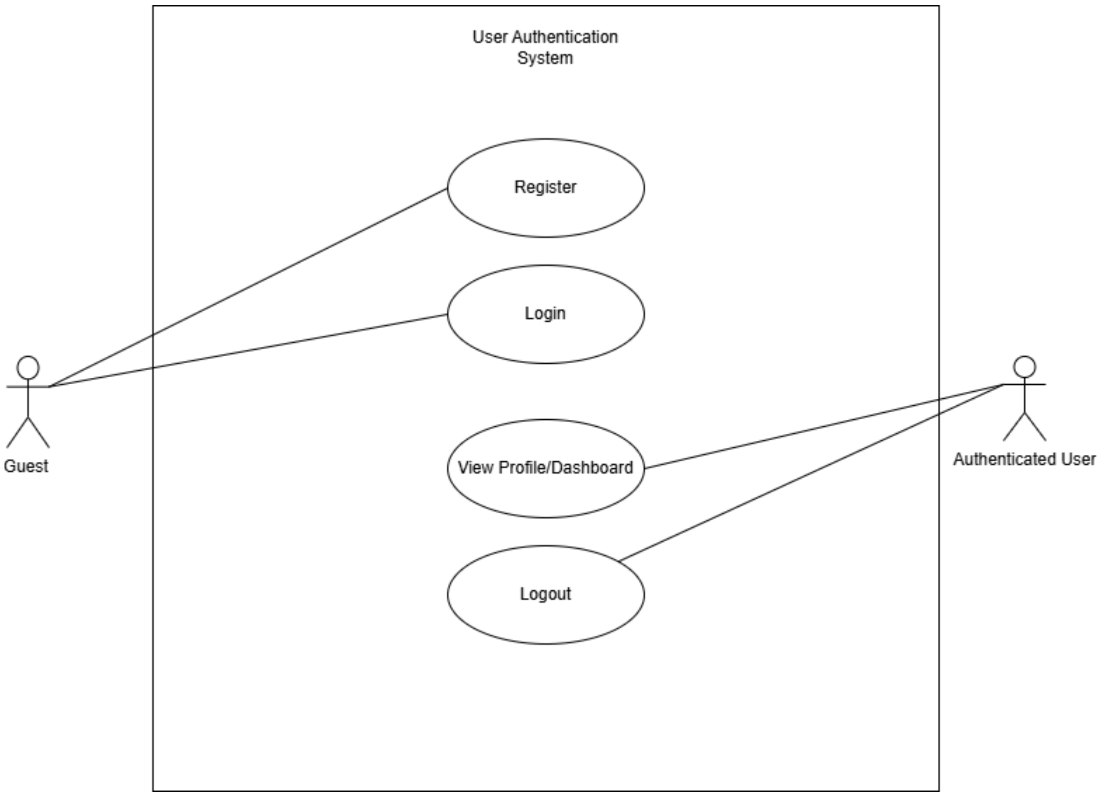
Insert the necessary diagrams for the system:

5.1. ERD

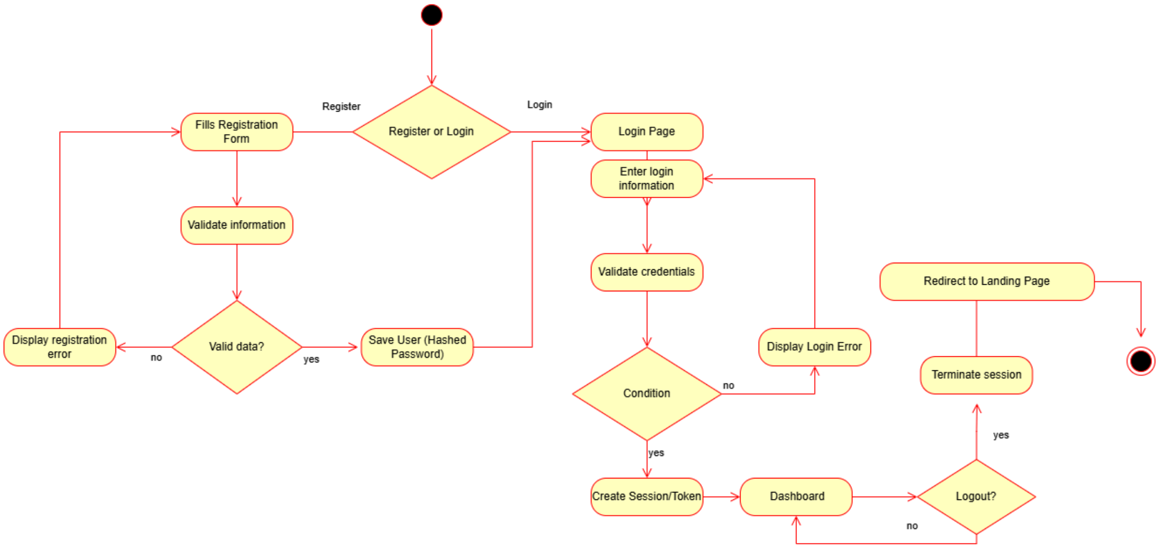


Users	
PK	user_id
U	username
U	email
	password_hash
	role
	created_at
	updated_at

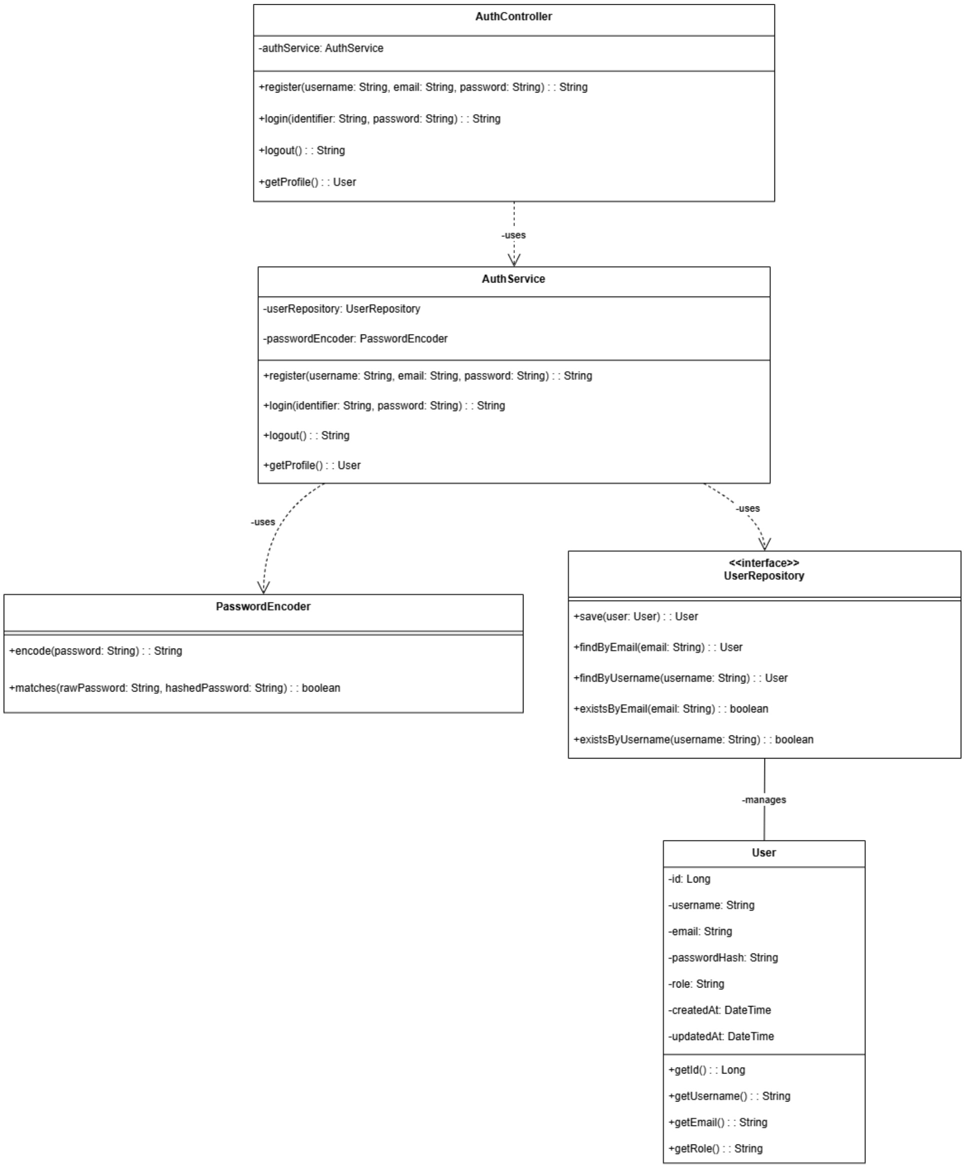
5.2. Use Case Diagram



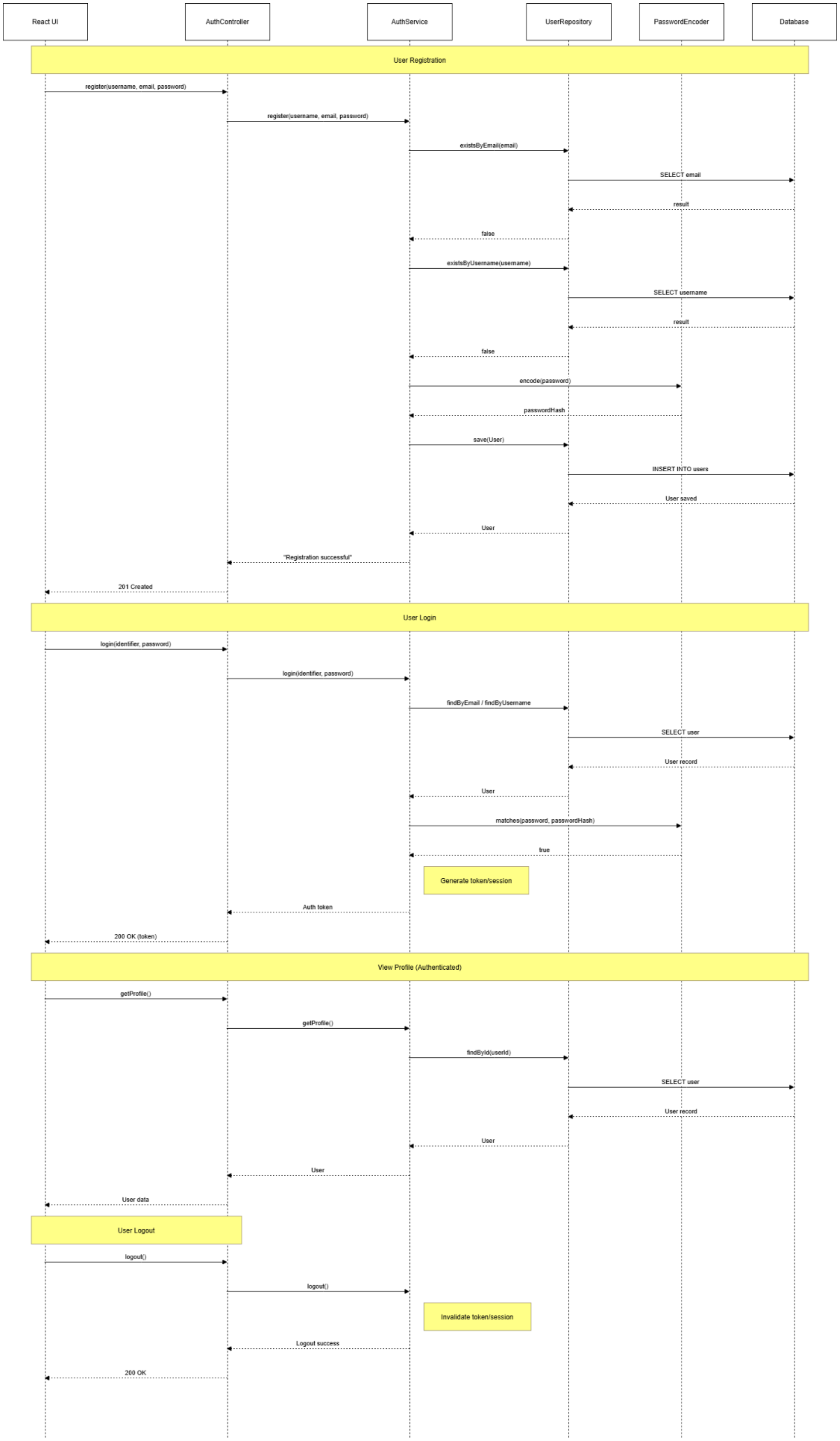
5.3. Activity Diagram



5.4. Class Diagram


AuthController
-authService: AuthService
+register(username: String, email: String, password: String) : : String
+login(identifier: String, password: String) : : String
+logout() : : String
+getProfile() : : User
-uses
AuthService
-userRepository: UserRepository
-passwordEncoder: PasswordEncoder
+register(username: String, email: String, password: String) : : String
+login(identifier: String, password: String) : : String
+logout() : : String
+getProfile() : : User
-uses
-uses
PasswordEncoder
+encode(password: String) : : String
+matches(rawPassword: String, hashedPassword: String) : : boolean
<<interface>>
UserRepository
+save(user: User) : : User
+findByEmail(email: String) : : User
+findByUsername(username: String) : : User
+existsByEmail(email: String) : : boolean
+existsByUsername(username: String) : : boolean
-manages
User
-id: Long
-username: String
-email: String
-passwordHash: String
-role: String
-createdAt: DateTime
-updatedAt: DateTime
+getId() : : Long
+getUsername() : : String
+getEmail() : : String
+getRole() : : String

5.5. Sequence Diagram


React UI
AuthController
AuthService
UserRepository
PasswordEncoder
Database
User Registration
register(username, email, password)
register(username, email, password)
existsByEmail(email)
SELECT email
result
false
existsByUsername(username)
SELECT username
result
false
encode(password)
passwordHash
save(User)
INSERT INTO users
User saved
User
"Registration successful"
201 Created
User Login
login(identifier, password)
login(identifier, password)
findByEmail / findByUsername
SELECT user
User record
User
matches(password, passwordHash)
true
Generate token/session
Auth token
200 OK (token)
View Profile (Authenticated)
getProfile()
getProfile()
findById(userId)
SELECT user
User record
User
User
User data
User Logout
logout()
logout()
Invalidate token/session
Logout success
200 OK

6. Appendices

Include any additional information, references, or support materials.