CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342-Section
SYSTEMS INTEGRATION AND ARCHITECTURE 1

FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

Project Title:

Prepared By: Gian B. Fernandez

Date of Submission: 05/02/2026

Version:

Table of Contents

1. Introduction

1.1. Purpose

The purpose of the Authentication System is to provide a secure and reliable way for users to register, log in, access protected resources, and log out. This document is intended for developers, testers, and project evaluators to understand the requirements and functionality of the system.

1.2. Scope

- Allow users to create accounts.
- Allow users to log in using credentials.
- Restrict access to protected pages for unauthenticated users.
- Allow users to log out securely.

1.3. Definitions, Acronyms, and Abbreviations

Term / Acronym	**Definition**
SRS	Software Requirement Specification
User	A person who registers and logs in to the system
Registration	The process of creating a new user account
Login	The process of authenticating a registered user
Logout	The process of ending a user session
Dashboard	Protected page visible only to authenticated users

2. Overall Description

2.1. System Perspective

The Authentication System operates as a standalone module responsible for validating users, maintaining session states, and controlling access to protected resources.

2.2. User Classes and Characteristics

User Class	**Characteristics / Privileges**
Guest User	Can register and attempt to log in. Cannot access protected pages.
Authenticated User	Can access protected pages, view profile/dashboard, and log out.

2.3. Operating Environment

- Web browser or mobile device for user access.
- Backend server handling authentication logic.
- Database to store user credentials securely.

2.4. Assumptions and Dependencies

- Users will provide valid and accurate credentials.
- Passwords will be encrypted and securely stored.
- System depends on a running database for storing user data.

3. System Features and Functional Requirements

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

3.1. Feature 1: User Registration

Description: Users can create an account to access protected pages.
Functional Requirements:
- Users provide username, email, and password.
- System validates input and prevents duplicate accounts.
- Passwords are securely hashed before storage.

3.2. Feature 2: User Login

Description: Registered users can log in to access protected pages.
Functional Requirements:
- Users enter email/username and password.
- System verifies credentials.
- Invalid credentials trigger an error and allow retry.

3.3. Feature 3: Profile / Dashboard Access

Description: Authenticated users can view their profile or dashboard.
Functional Requirements:
- Only authenticated users can access protected pages.
- Unauthorized users attempting access are redirected to login.

3.4. Feature 3: User Logout

Description: Users can safely end their session.
Functional Requirements:
- Logout invalidates the user session or token.
- Redirects the user to the login page.

4. Non-Functional Requirements

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

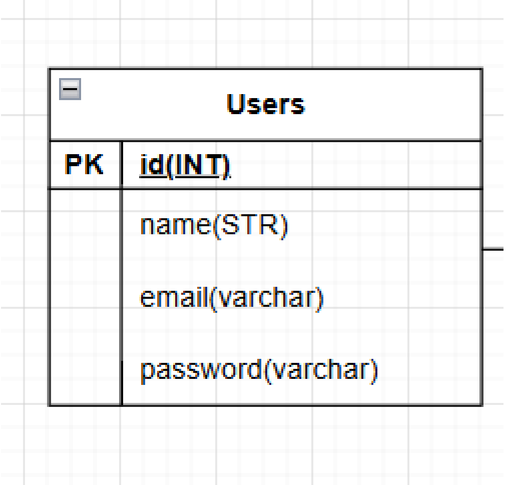
- **Performance:** System must authenticate users within 2–3 seconds.
- **Security:** Passwords must be encrypted; sessions protected from hijacking.
- **Usability:** Simple and clear forms for registration and login.
- **Reliability:** The system must remain available 99% of the time.
- **Maintainability:** Code should be modular for easy updates and fixes.

5. System Models (Diagrams)

Insert the necessary diagrams for the system:

5.1. ERD

Insert ERD here

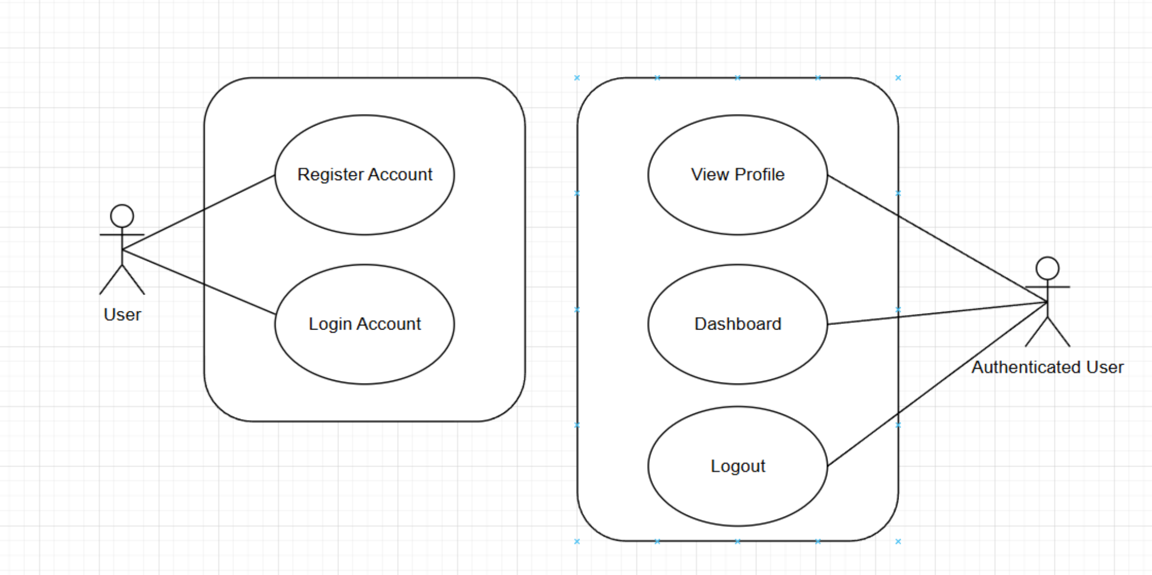


	Users
PK	id(INT)

	name(STR)
	email(varchar)
	password(varchar)

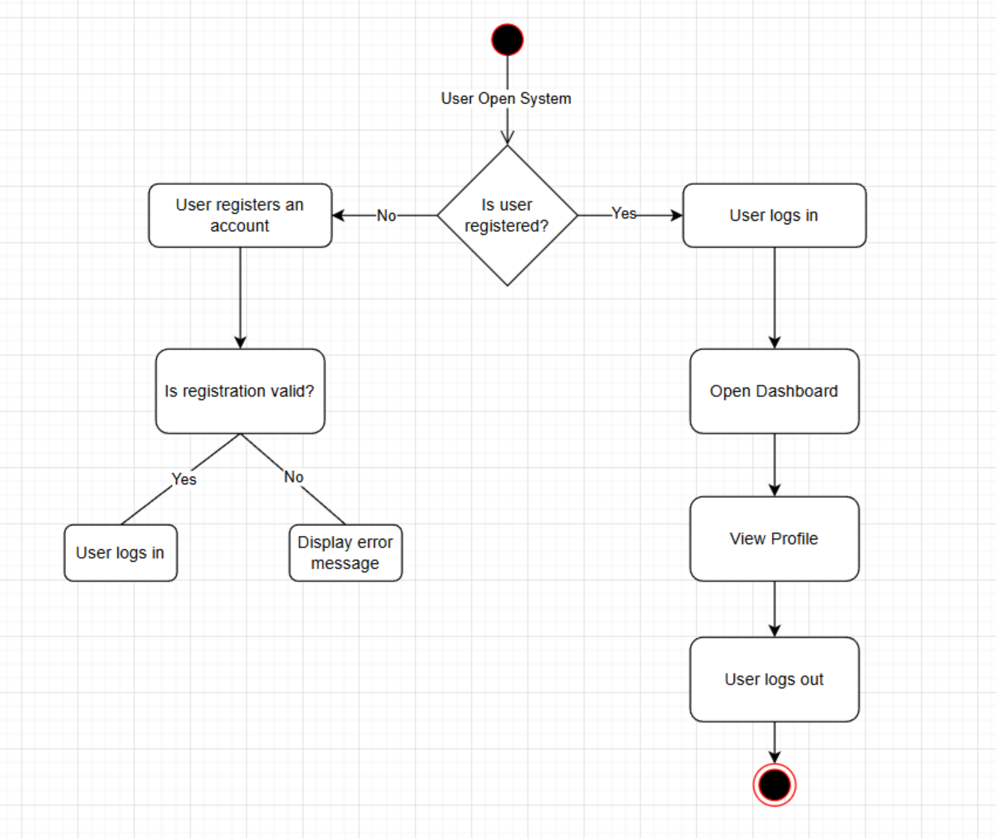
5.2. Use Case Diagram

Insert ERD here



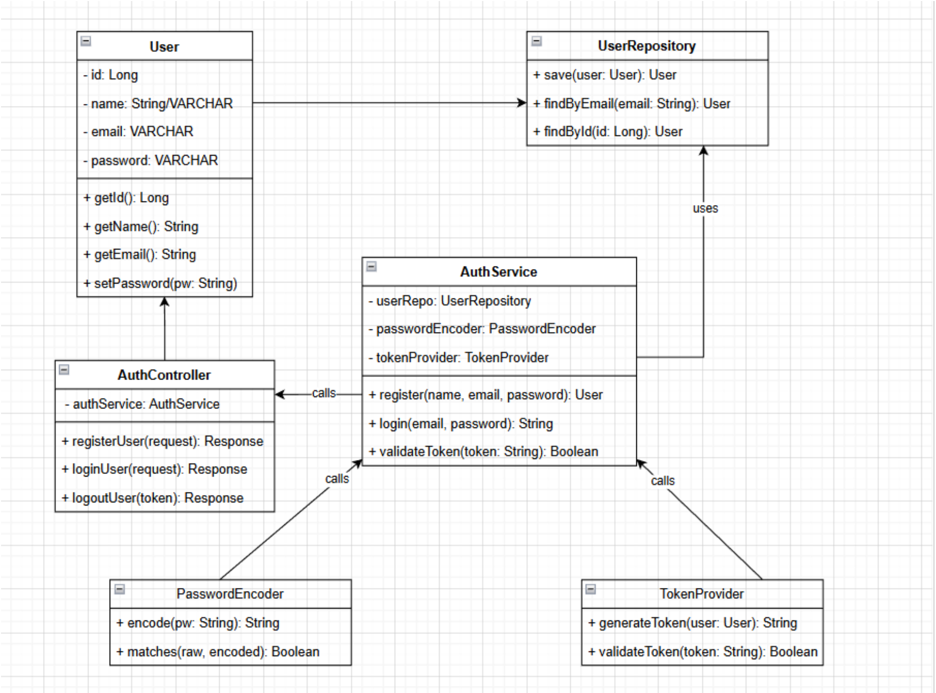
5.3. Activity Diagram

Insert ERD here



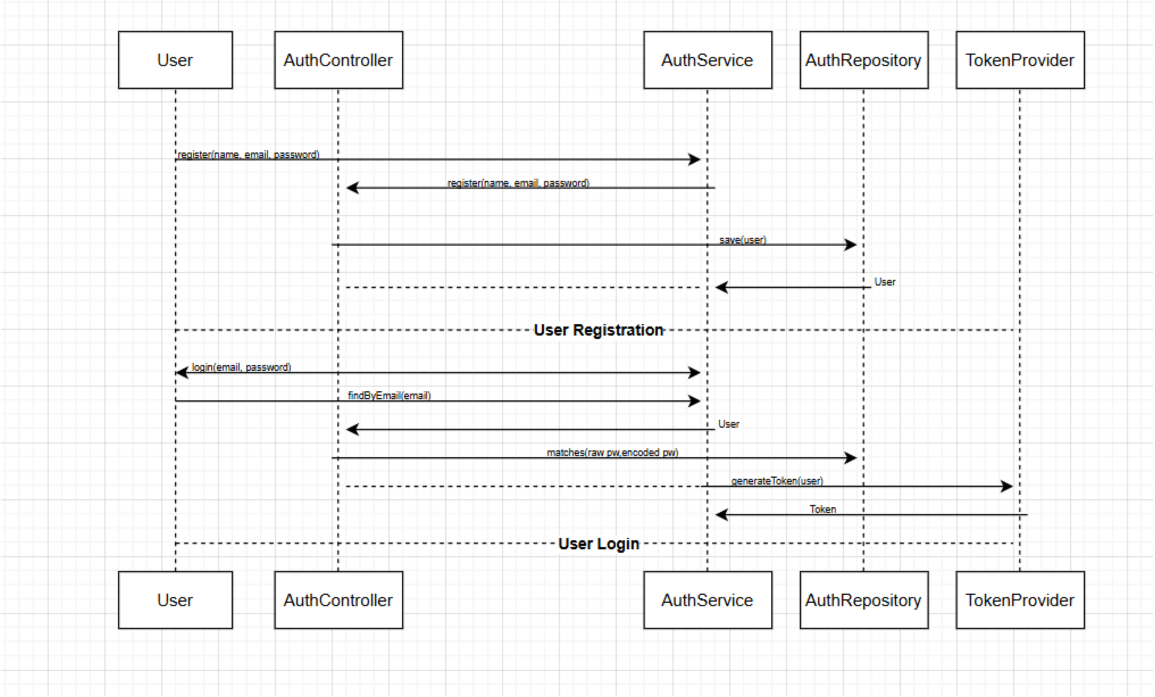
5.4. Class Diagram

Insert ERD here


User
- id: Long
- name: String/VARCHAR
- email: VARCHAR
- password: VARCHAR
+ getId(): Long
+ getName(): String
+ getEmail(): String
+ setPassword(pw: String)
UserRepository
+ save(user: User): User
+ findByEmail(email: String): User
+ findById(id: Long): User
uses
AuthService
- userRepo: UserRepository
- passwordEncoder: PasswordEncoder
- tokenProvider: TokenProvider
+ register(name, email, password): User
+ login(email, password): String
+ validateToken(token: String): Boolean
AuthController
- authService: AuthService
+ registerUser(request): Response
+ loginUser(request): Response
+ logoutUser(token): Response
calls
calls
calls
PasswordEncoder
+ encode(pw: String): String
+ matches(raw, encoded): Boolean
TokenProvider
+ generateToken(user: User): String
+ validateToken(token: String): Boolean

5.5. Sequence Diagram

Insert ERD here



6. Appendices

- Diagrams: ERD, Use Case Diagram, Activity Diagram for authentication flow
- References: User account data format, token/session structure