CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342-Section SYSTEMS INTEGRATION AND ARCHITECTURE 1

FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

Project Title: Mini-App - User Registration and Authentication

Prepared By: Fernandez, Xavier A.

Date of Submission: 2/2/2026

Version: 1 part 2

Table of Contents

1. Introduction

1.1. Purpose

To familiarize and design a system flow for User Registration, Login, and Logout.

1.2. Scope

The documentation shows the system flow of User Registration, Login and Logout.

1.3. Definitions, Acronyms, and Abbreviations

List and define important terms used in this document.

1. **SRS**: Software Requirements Specification.
2. **ERD**: Entity Relationship Diagram.
3. **JWT / Session Token**: A string of characters used to identify a session after a user logs in.
4. **Hashing**: The process of converting a password into a fixed-length string of characters for security.
5. **RBAC**: Role-Based Access Control

2. Overall Description

2.1. System Perspective

The system serves as a **centralized Authentication and Profile Management module** within a larger web application ecosystem. It acts as a gatekeeper between the public internet and "Protected Pages" by managing user state through session tokens.

2.2. User Classes and Characteristics

- **Guest User:** Unauthenticated individuals who can only access the registration, and login pages. Their primary goal is to establish an identity within the system.
- **Authenticated User:** Guest User / users who have successfully provided valid credentials. They inherit the abilities of a guest but gain elevated permissions to access the Dashboard, view their Profile, and terminate their session via Logout.

2.3. Operating Environment

The system is designed to operate within a standard web-based ecosystem consisting of the following layers:

- **Server Side (Backend):**
    - **Runtime Environment:** Java Runtime Environment (JRE) or Node.js (depending on your final implementation of the AuthService).
    - **Framework:** Spring Boot or a similar MVC architecture to handle the AuthController and AuthService logic.
    - **Web Server:** An embedded server like Apache Tomcat or Nginx to handle incoming HTTP requests for registration and login.

- **Database Layer:**
    - **Management System:** A relational database (RDBMS) such as MySQL or PostgreSQL to manage the User, Sessions, and User Profiles tables.

    - **Persistence:** The environment must support persistent storage to ensure password_hash and session_token data survive system restarts.

- **Client Side (Frontend):**
    - **Browsers:** The application shall be compatible with modern web browsers (e.g., Google Chrome, Mozilla Firefox, Microsoft Edge).

    - **State Management:** The client environment must support cookie or local storage handling to store the session_token returned by the system.

- **Network:**
    - **Protocol:** The environment should ideally support HTTPS to ensure that sensitive data like passwords and session tokens are encrypted during transit.

2.4. Assumptions and Dependencies

List any assumptions and external dependencies that may affect the system.

1. **Assumption**: Users have access to a modern web browser with JavaScript enabled.
2. **Dependency**: The system relies on a persistent database being available and reachable.
3. **Dependency**: The backend server requires an environment capable of running the chosen framework.

3. System Features and Functional Requirements

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

3.1. Feature 1: User Identity and Profile Management

Description: This feature allows individuals to create a unique account and manage their personal information. It ensures that every user has a distinct record in the database and a corresponding profile for the dashboard.
Functional Requirements:
- Registration : The system shall allow Guest Users to create an account by providing an email and password.
- Password Encryption : The system must encrypt passwords using a secure algorithm before storing them in the *User* table.
- Profile Retrieval : The system shall fetch data from the *User Profiles* table to display the user's name and profile on the Dashboard upon successful authentication.

3.2. Feature 2: Session and Access Control

Description: This feature governs the login and logout lifecycle. It manages the creation and invalidation of session tokens to protect restricted pages from unauthorized access.
Functional Requirements:

- Authentication**:** The system shall verify user credentials against the database and generate a unique *session_token* for valid logins.
- Session Persistence: The system must store active sessions in the *Sessions* table, including an *expires_at* timestamp to handle automatic timeouts.
- Access Restriction: The system shall block access to "Protected Pages" if a valid, non-expired *session_token* is not present in the request.
- Logout (Invalidation): The system shall delete or invalidate the *session_token* in the database when a user triggers the logout action, transitioning them back to a "Guest" state.

4. Non-Functional Requirements

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

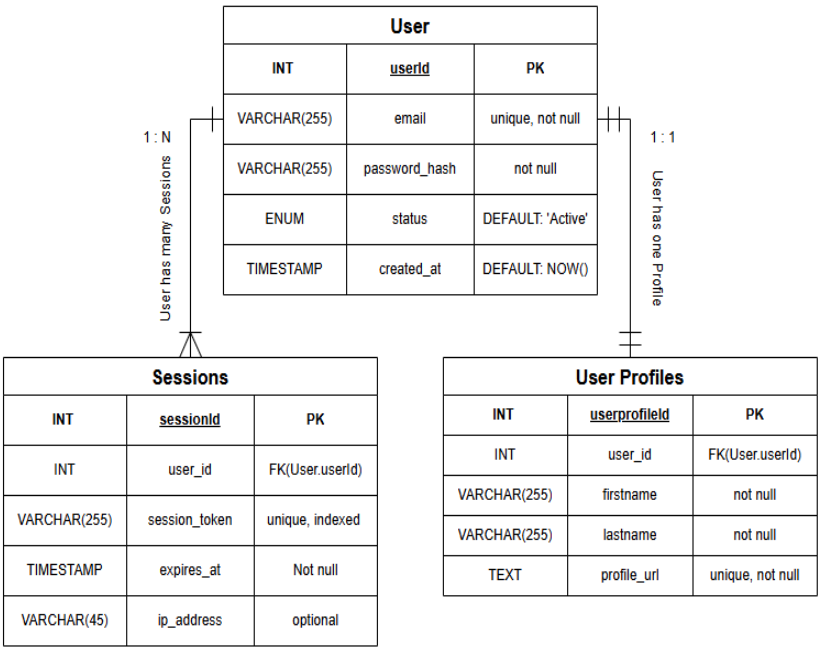
- **Security**: All passwords must be hashed using a strong algorithm like **BCrypt** or **Argon2** before storage.
- **Reliability**: The system should have a 99.9% uptime to ensure users can log in at any time.
- **Usability**: The registration and login forms should provide immediate feedback (e.g., "Invalid email format") to the user.
- **Performance**: Authentication requests should be processed in under 500ms.

5. System Models (Diagrams)

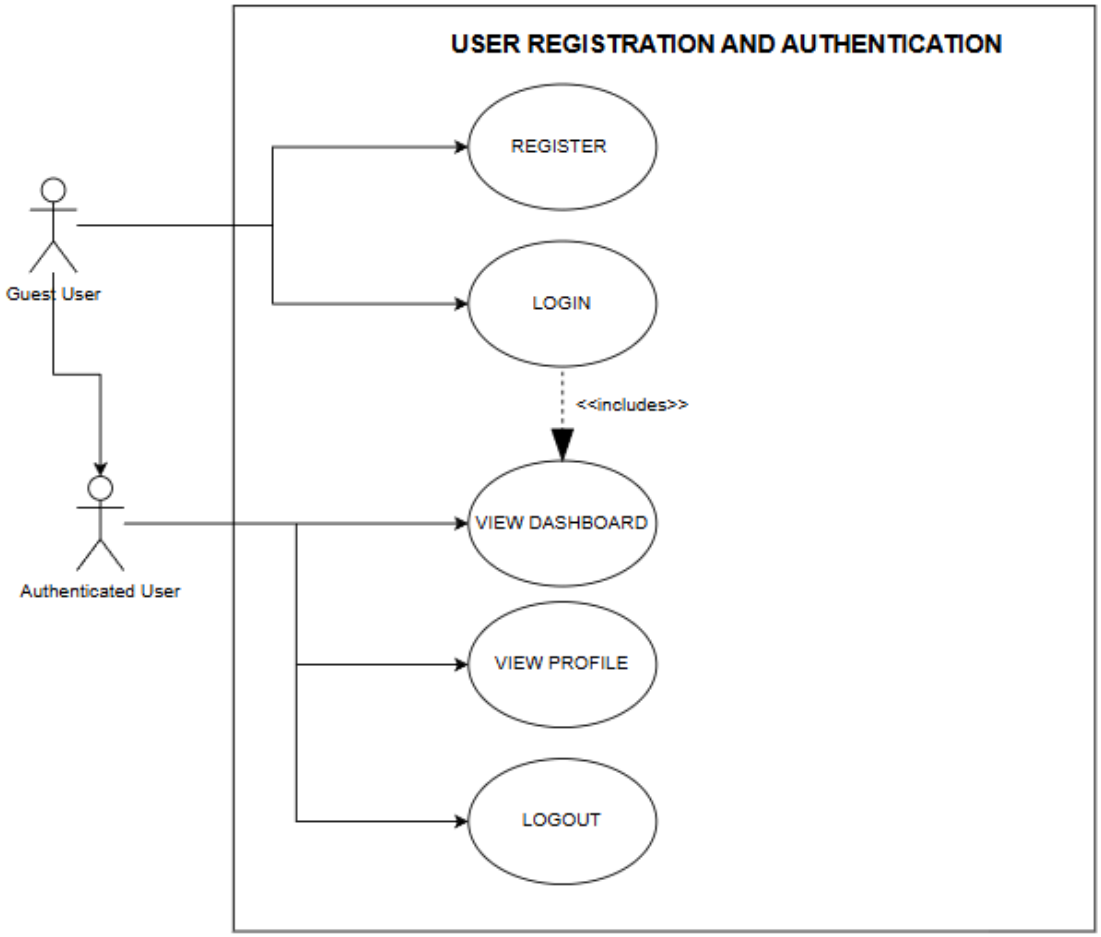
Insert the necessary diagrams for the system:

5.1. ERD

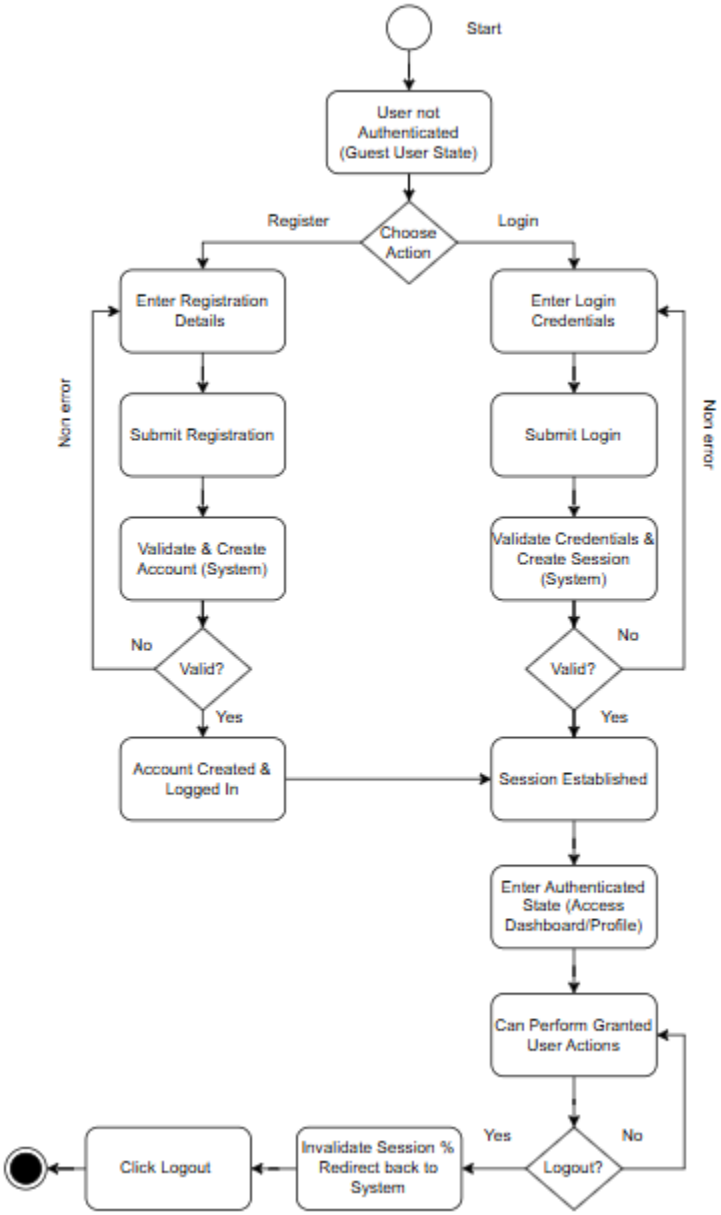
USER REGISTRATION, LOGIN and SESSION MANAGEMENT



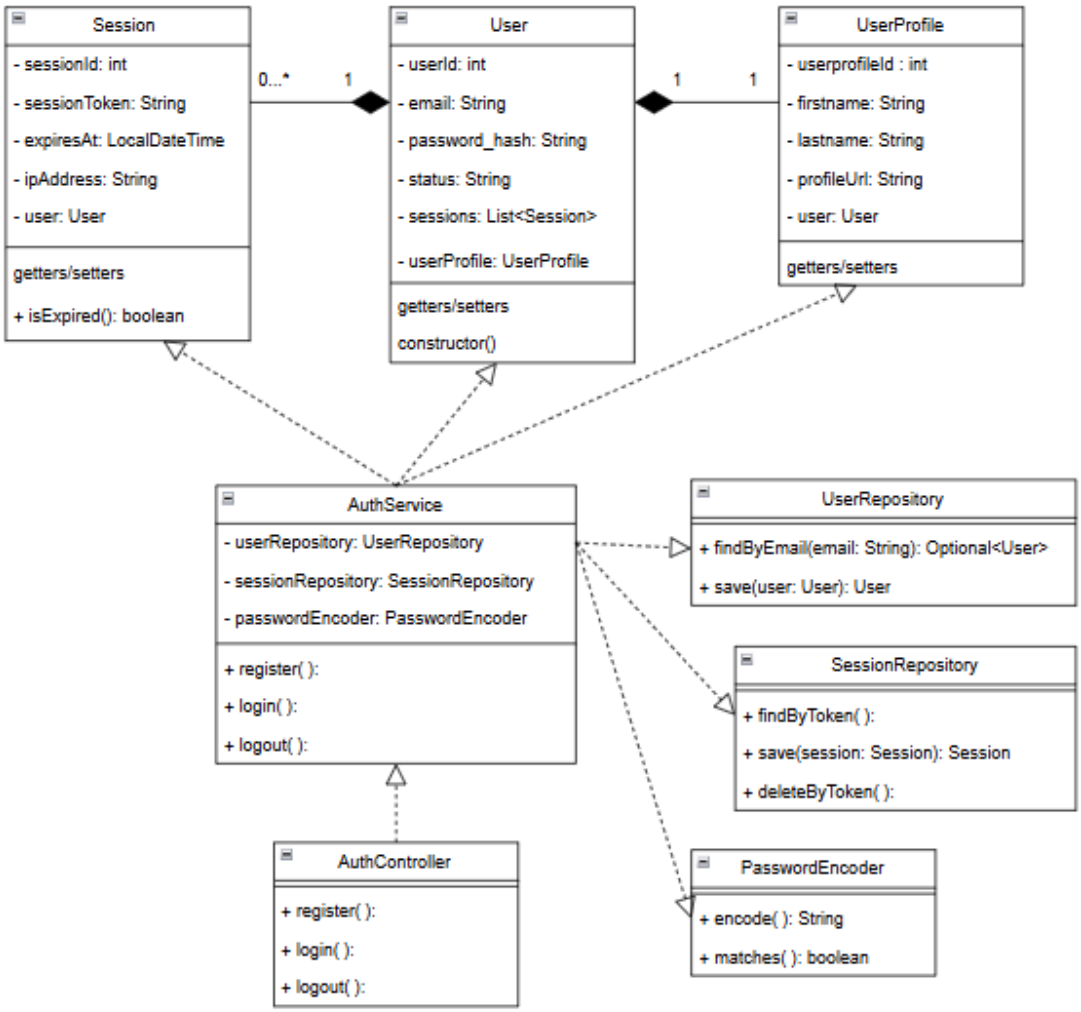
5.2. Use Case Diagram



5.3. Activity Diagram



5.4. Class Diagram



5.5. Sequence Diagram

6. Appendices

Include any additional information, references, or support materials.