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: Loy, Andrei Sam P.

Date of Submission: 1/31/2026

Version: 1.1

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this system is to provide a secure and user-friendly platform for managing user accounts. It enables users to register, log in, access protected areas like their profile or dashboard, and log out safely.

1.2. Scope

The system will let users:

- Register new accounts.
- Log in using their credentials.
- Access a personal dashboard.
- Log out securely.

What it won't do:

- No third-party login features
- Advanced features like password recovery, email verification, or multi-factor authentication (unless required later)

1.3. Definitions, Acronyms, and Abbreviations

- **UI (User Interface):** The front-end part of the system used by the user (React UI).
- **API (Application Programming Interface):** Backend endpoints used by the UI to communicate with the system (Spring Boot API).
- **Authentication:** Verifying a user's identity (login process).
- **Authorization:** Controlling access to protected pages based on login status.
- **Session/Token:** Data used to keep the user logged in after successful authentication.
- **Database (DB):** Storage for user records (e.g., Users table).
- **Password Hashing:** Securing passwords by storing an encrypted/hashed version instead of plain text.

2. Overall Description

2.1. System Perspective

This system is a basic authentication module that can be part of a larger web application. It connects a React UI to a Spring Boot backend API, which interacts with a database to store and retrieve user data. The system enforces access control so protected pages cannot be opened when a user is logged out.

2.2. User Classes and Characteristics

- **Guest User:** Not logged in. Can access registration and login pages only.
- **Authenticated User:** Logged in user. Can access protected pages like dashboard/profile and can log out.

2.3. Operating Environment

- **Client:** Web browser (Chrome/Edge/Firefox) running the React UI
- **Server:** Spring Boot backend application running on a server/local machine
- **Database:** MySQL (or any relational DB) storing the Users table
- **Tools/Tech:** React, Spring Boot, REST API, MySQL, Postman (optional for testing)

2.4. Assumptions and Dependencies

- Users have internet/local network access to reach the system.
- The database server is running and accessible by the Spring Boot API.
- The backend will handle password hashing and authentication logic.
- The UI depends on backend API responses to allow/deny access to protected pages.

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 by providing required details.
Functional Requirements:

- The system shall allow a guest user to submit registration details (e.g., username/email and password).
- The system shall validate required fields and prevent empty/invalid inputs.
- The system shall store the new user record in the database with a hashed password.

3.2. Feature 2: User Login and Access Control

Description: Allows users to log in and access protected pages
Functional Requirements:

- The system shall allow users to submit login credentials.
- The system shall verify credentials against stored user data in the database.
- The system shall block access to protected pages if the user is not authenticated.

3.3. Feature 2: User Logout

Description: Allows authenticated users to safely end their session.
Functional Requirements:

- The system shall provide a logout action for authenticated users.
- The system shall clear the user’s session/token after logout.
- The system shall redirect the user back to the login page after logout.

4. Non-Functional Requirements

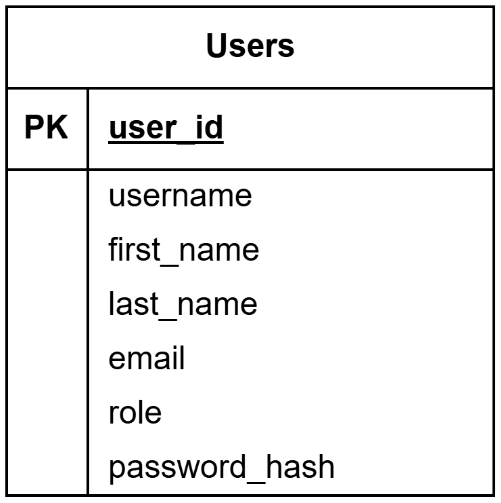
- **Security:** Passwords must be stored as hashed values (not plain text). Protected pages must require authentication.
- **Usability:** The UI should be simple and easy to navigate (clear register/login/logout buttons).
- **Performance:** Login and registration should respond quickly under normal use.

- **Reliability:** The system should handle invalid input gracefully (show error messages without crashing).
- **Maintainability:** Code should be modular (separate controller, service, repository classes).

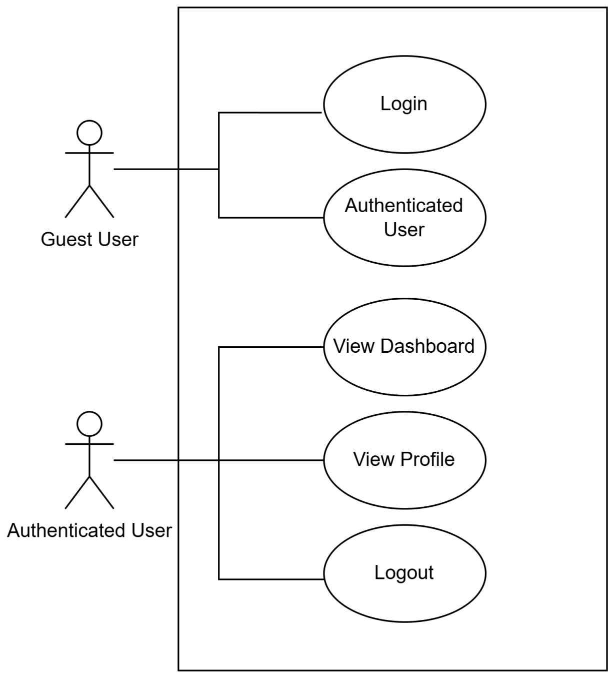
5. System Models (Diagrams)

Insert the necessary diagrams for the system:

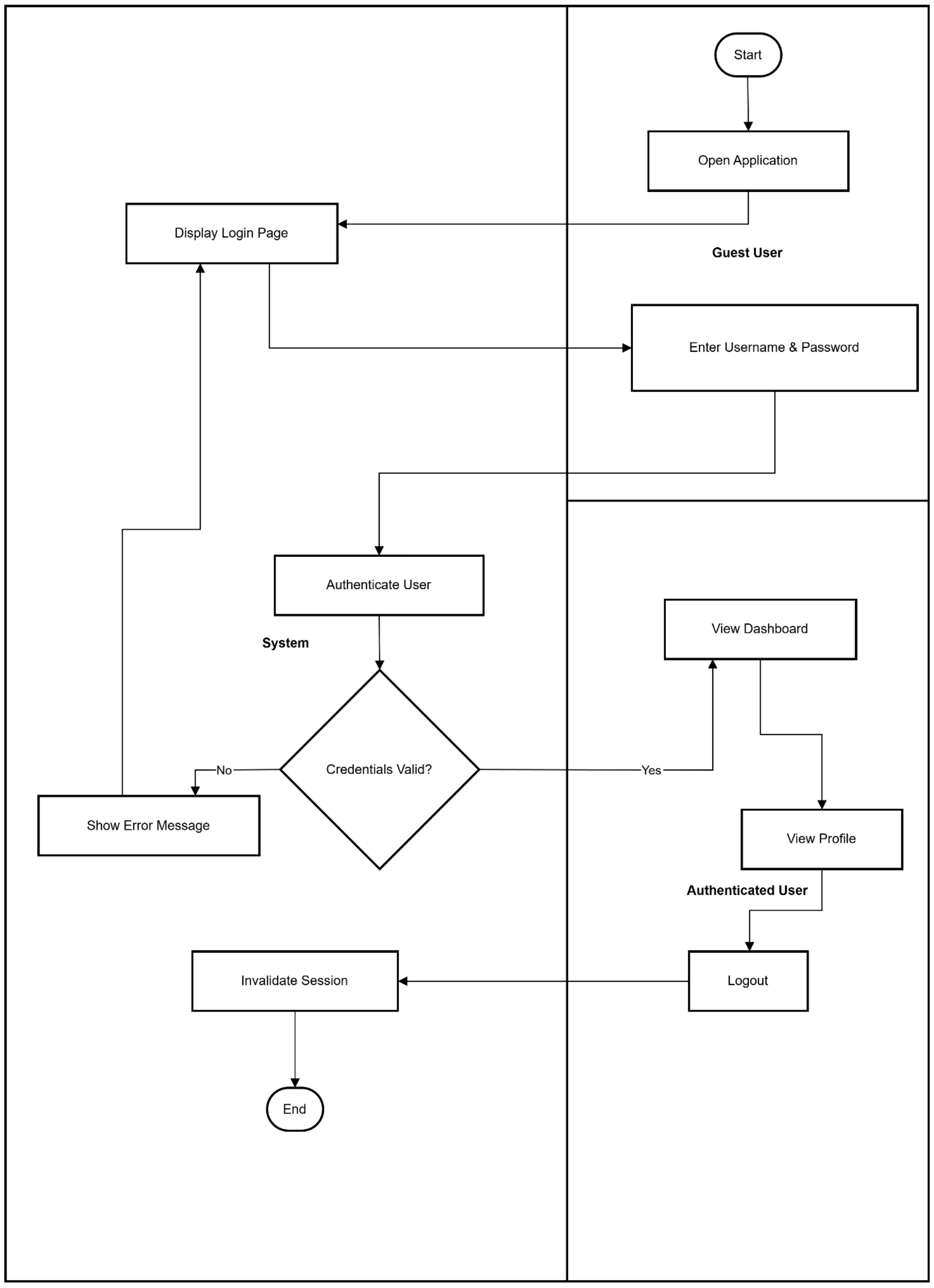
5.1. ERD



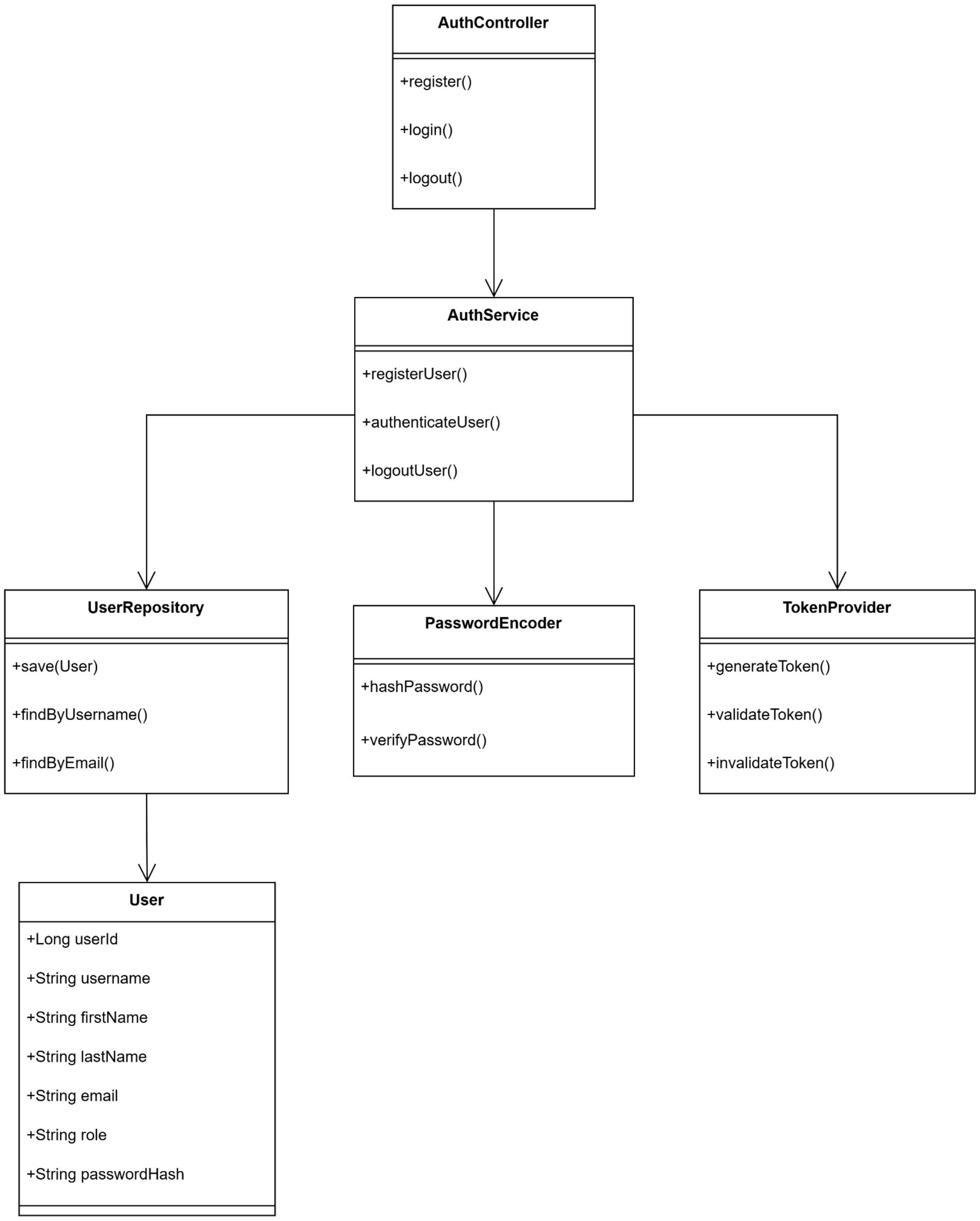
5.2. Use Case Diagram



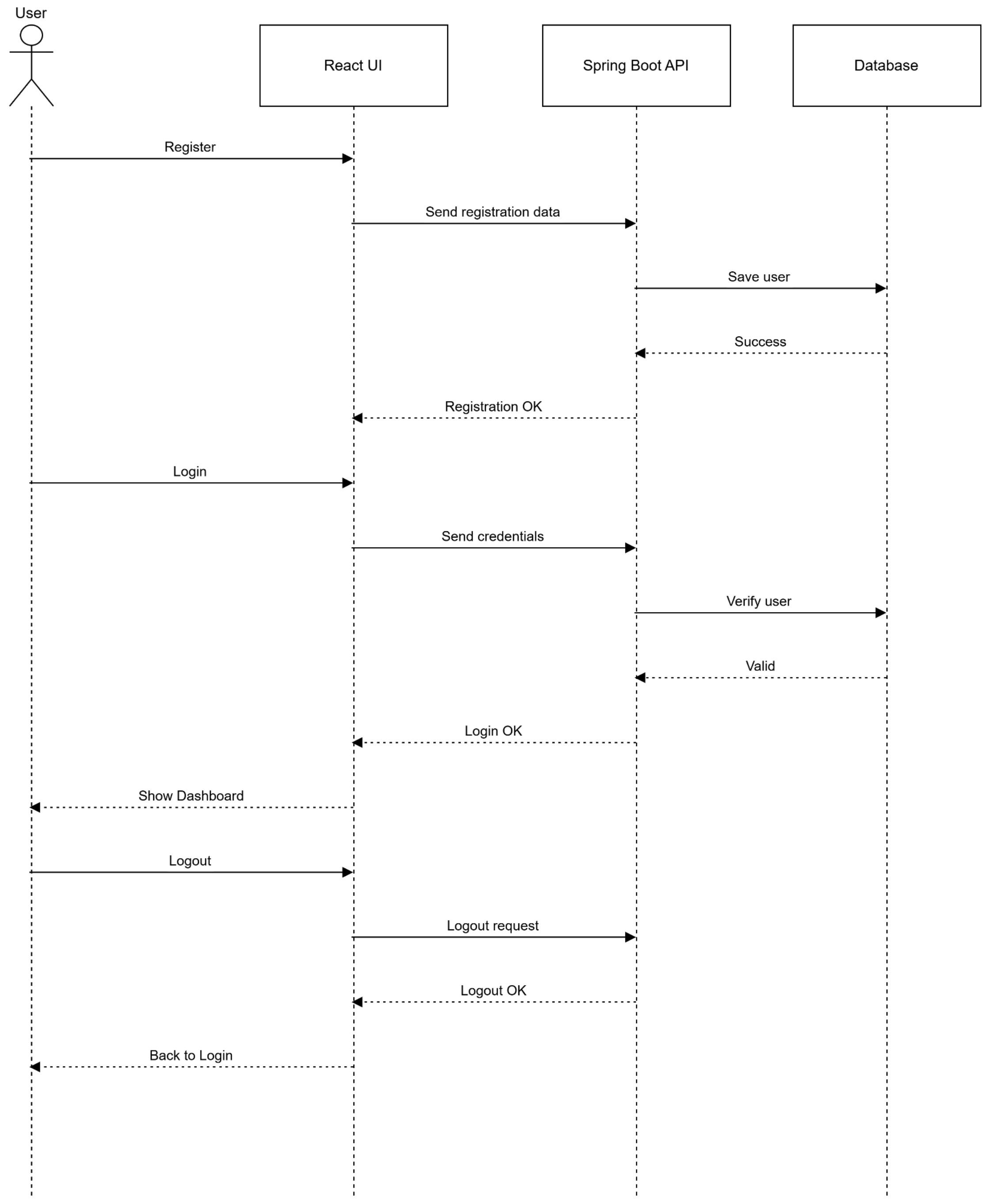
5.3. Activity Diagram



5.4. Class Diagram



5.5. Sequence Diagram



6. Appendices

Include any additional information, references, or support materials.