CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342-Section
SYSTEMS INTEGRATION AND ARCHITECTURE 1

FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

Project Title: Mini App – User Registration & Authentication System

Prepared By: Ralph Keane A. Maestrado

Date of Submission: February 3, 2026

Version: 1.0

Table of Contents

1. Introduction

1.1. Purpose

This system provides a secure gateway for users to access the application by managing their credentials and personal profiles. It is intended for developers, project stakeholders, and system administrators.

1.2. Scope

The system will handle Sign-ups, Login, Password Recovery, and Profile Management. It serves as the foundational security layer for the "Mini App" but does not include third-party payment processing or complex social media integrations at this stage.

1.3. Definitions, Acronyms, and Abbreviations

- **FRS** – Functional Requirements Specification: A document that describes how a system must behave and its specific features.
- **UI** – User Interface: The space where interactions between humans and the application occur.
- **API** – Application Programming Interface: A set of rules that allow the React frontend and Spring Boot backend to communicate.
- **ERD** – Entity Relationship Diagram: A visual representation of the database structure and how data entities relate to one another.
- **JWT** – JSON Web Token: A secure method for transmitting information between parties as a JSON object, used here for authentication tokens.
- **BCrypt** – A password-hashing function used to securely store credentials in the database.
- **ReactJS** – A JavaScript library used for building the user interface and handling the frontend logic.
- **Spring Boot** – A Java-based framework used to create the REST API and handle the backend business logic.
- **MySQL** – A relational database management system used to store user credentials and profile data.

2. Overall Description

2.1. System Perspective

The authentication system acts as a standalone module that integrates with the main "Mini App" database to verify user identity before granting access to protected features. A three-tier client-server architecture. The Frontend ReactJS handles the UI; the Backend Spring Boot API processes business logic; and the Database MySQL stores persistence data.

2.2. User Classes and Characteristics

- **Unregistered User:** Can view the landing page and register.

- **Registered User:** Can log in, manage their profile, and access app features.

2.3. Operating Environment

Modern web browsers for the front end, a Java-based Spring Boot REST environment for the backend, and MySQL for data storage.

2.4. Assumptions and Dependencies

- Users are assumed to have stable internet access to facilitate communication between the ReactJS frontend and Spring Boot backend.
- The system depends on the backend REST API being correctly always configured and accessible to the frontend.
- The application assumes that the MySQL database (or Firebase) is active and properly structured to store user credentials.
- It is assumed that users access the application through a modern web browser capable of executing ReactJS.

3. System Features and Functional Requirements

3.1. Feature 1: Account Creation (Registration)

Description: Enables guests to join the system by providing valid credentials.
Functional Requirements:

- The system will enable guest users to register using a unique email address and a secure password.
- The system shall validate all required input fields to ensure data integrity.
- The system shall encrypt the user's password before storing it in the MySQL database.
- The system shall save user information to the database only after successful validation and account creation.

3.2. Feature 1: User Authentication and Session Management

Description: Enables registered users to log in, access secure areas of the system, and safely log out.
Functional Requirements:

- The system shall verify user identity using valid login credentials stored in the database.
- The system shall issue a secure authentication token (e.g., JWT) after a successful login.
- The system shall prevent unauthenticated users from accessing protected pages or the dashboard.
- The system shall terminate the user session and invalidate the authentication token upon a logout request.

4. Non-Functional Requirements

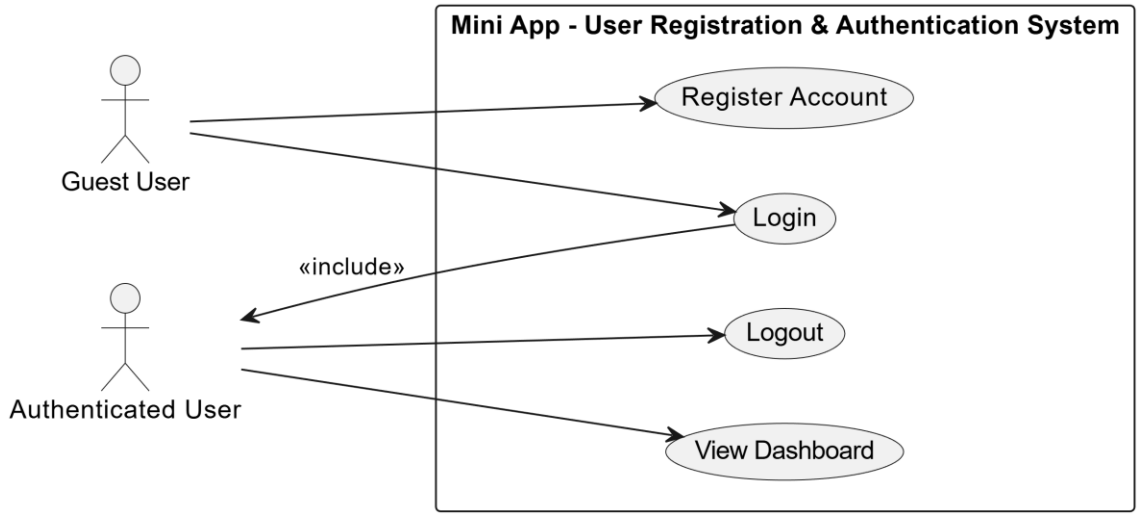
- **Security**: All user passwords must be securely hashed using strong encryption algorithms like BCrypt before being stored in the database.
- **Performance**: The system shall process login and registration requests within an acceptable timeframe to ensure smooth user experience.
- **Usability**: The user interface shall be designed to be simple, intuitive, and easy for all user classes to navigate.
- **Reliability**: The system shall gracefully manage invalid inputs and authentication errors by providing clear feedback to the user.
- **Availability**: The system is dependent on the availability of the internet and the correct configuration of backend services to function properly.

5. System Models (Diagrams)

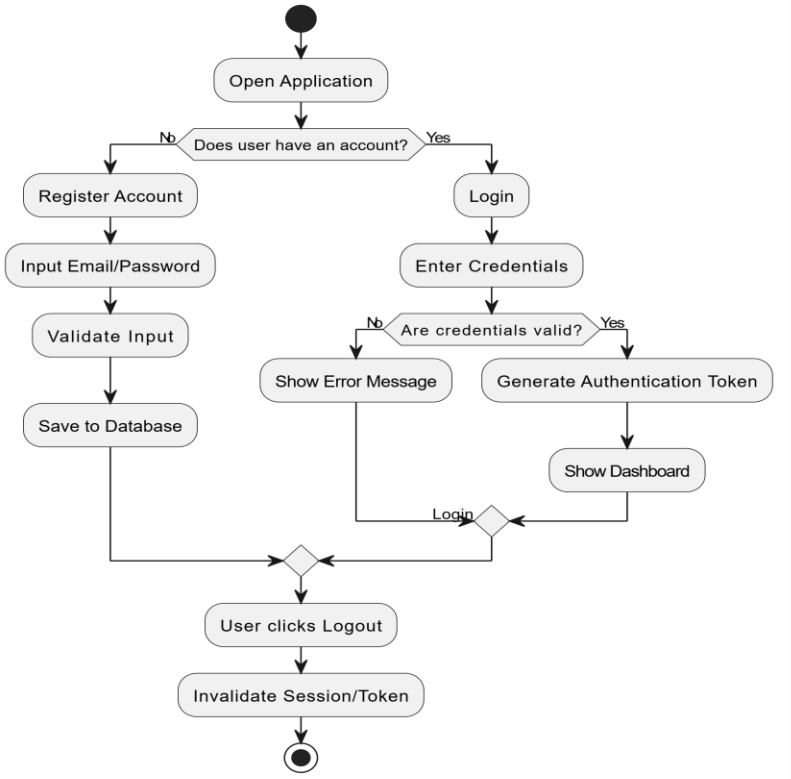
5.1. ERD

Table	
PK	user_id
	email
	password_hash
	first_name
	last_name
	created_at

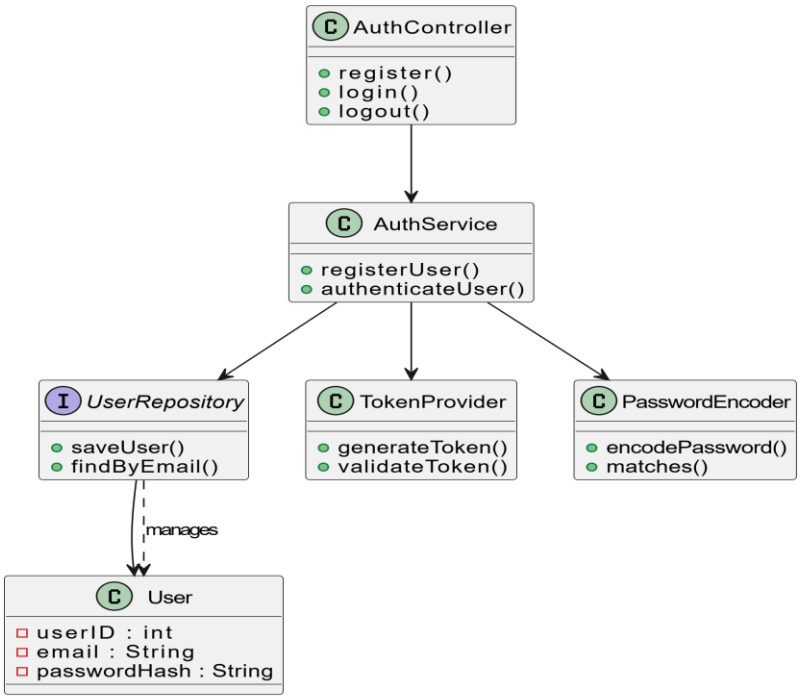
5.2. Use Case Diagram



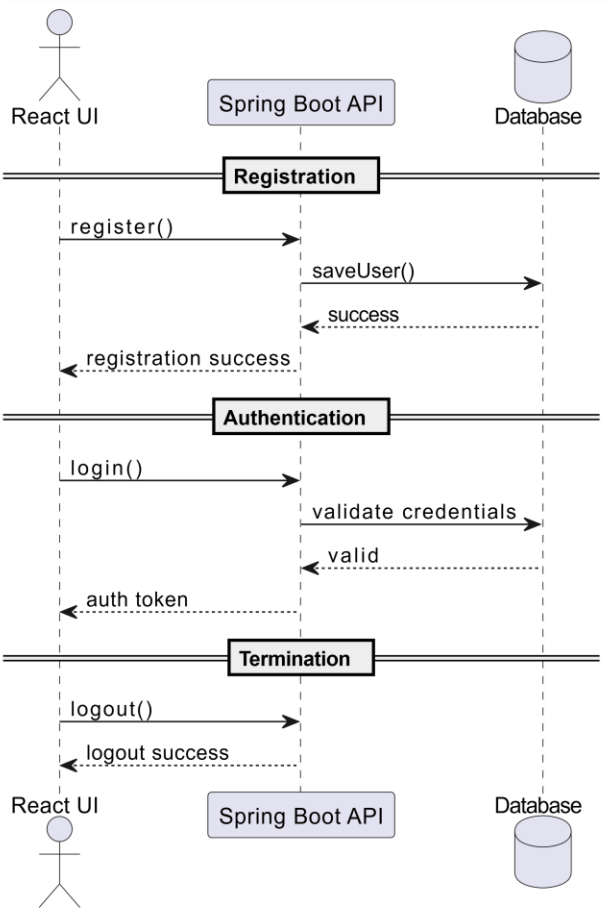
5.3. Activity Diagram



5.4. Class Diagram



5.5. Sequence Diagram



6. Appendices

- Technological References: Developed using ReactJS for the UI, Spring Boot for the API, and MySQL for data storage.
- Security: Uses BCrypt for password hashing and JSON Web Tokens (JWT) for secure user sessions.
- Tools: All diagrams were created using draw.io / diagrams.net.
- External Services: Assumes access to internet connectivity and correctly configured backend/database services.