CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342-Section SYSTEMS INTEGRATION AND ARCHITECTURE 1

FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

Project Title: User Registration and Authentication

Prepared By: Jeremiah T. Ramos

Date of Submission: January 31, 2026

Version: 1.0.0

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this document is to describe the authentication system of the application, including its features, functional behavior, and constraints.

1.2. Scope

The system provides **user authentication and access control only**.

Its scope is limited to:

- User registration
- User login
- User logout
- Viewing authenticated user account information

The system **does not** include business logic, role management, content management, or any domain-specific features beyond authentication.

1.3. Definitions, Acronyms, and Abbreviations

Term	Definition
Authentication	The process of verifying a user's identity
JWT	JSON Web Token, used for stateless authentication
API	Application Programming Interface
UI	User Interface
ERD	Entity Relationship Diagram
AuthLogs	Database table that records authentication-related actions
BlacklistedTokens	Database table that stores invalidated JWTs
REST	Representational State Transfer

2. Overall Description

2.1. System Perspective

The system is a **standalone authentication module** implemented as a **Spring Boot REST API** with a **React-based frontend**.

The frontend communicates with the backend via HTTP requests, and the backend persists authentication data in a relational database.

This system can function independently or be integrated into a larger application as an authentication service.

2.2. User Classes and Characteristics

User Type	Description
Guest User	An unauthenticated user who can register or log in
Authenticated User	A logged-in user who can view their profile and log out

2.3. Operating Environment

- **Frontend:** React (web browser-based)
- **Backend:** Spring Boot (Java)
- **Database:** Relational database (e.g., MySQL or PostgreSQL)
- **Authentication:** JWT-based stateless authentication
- **Client Platform:** Modern web browsers
- **Server Platform:** Any OS capable of running Java (Windows, Linux, macOS)

2.4. Assumptions and Dependencies

- Users have access to a modern web browser.
- Java Runtime Environment is available on the server.
- A relational database is properly configured and accessible.
- Network connectivity is stable between client and server.
- JWT secrets and security configurations are properly set.

3. System Features and Functional Requirements

3.1. Feature 1: User Registration

Description: Allows a guest user to create a new account by providing required credentials.
Functional Requirements:

- The system shall allow users to register using a username, email, and password.
- The system shall validate that the username and email are unique.
- The system shall securely store the user password in hashed form.

3.2. Feature 2: User Login

Description: Allows a registered user to authenticate and obtain access to protected resources.
Functional Requirements:
- The system shall allow users to log in using valid credentials.
- The system shall validate credentials against stored user data.
- The system shall issue a JWT upon successful authentication.

3.3. Feature 3: User Logout

Description: Allows an authenticated user to terminate their session.
Functional Requirements:
- The system shall allow authenticated users to log out.
- The system shall invalidate the JWT by storing it in a blacklist until it expires.
- The system shall record the logout action in the logs.

3.4. Feature 2: View Profile and Account Details

Description: Allows an authenticated user to view their account information.
Functional Requirements:
- The system shall restrict access to authenticated users only.
- The system shall retrieve user profile data from the database.
- The system shall return user information without exposing sensitive fields such as passwords.

4. Non-Functional Requirements

Security:

- Passwords shall be hashed using a secure algorithm.
- JWTs shall be validated on every protected request.
- Logged-out tokens shall be rejected using token blacklisting.

Performance:

- Authentication requests shall be processed within acceptable response times under normal load.

Usability:

- The user interface shall provide clear feedback for authentication actions (success or failure).

Reliability:

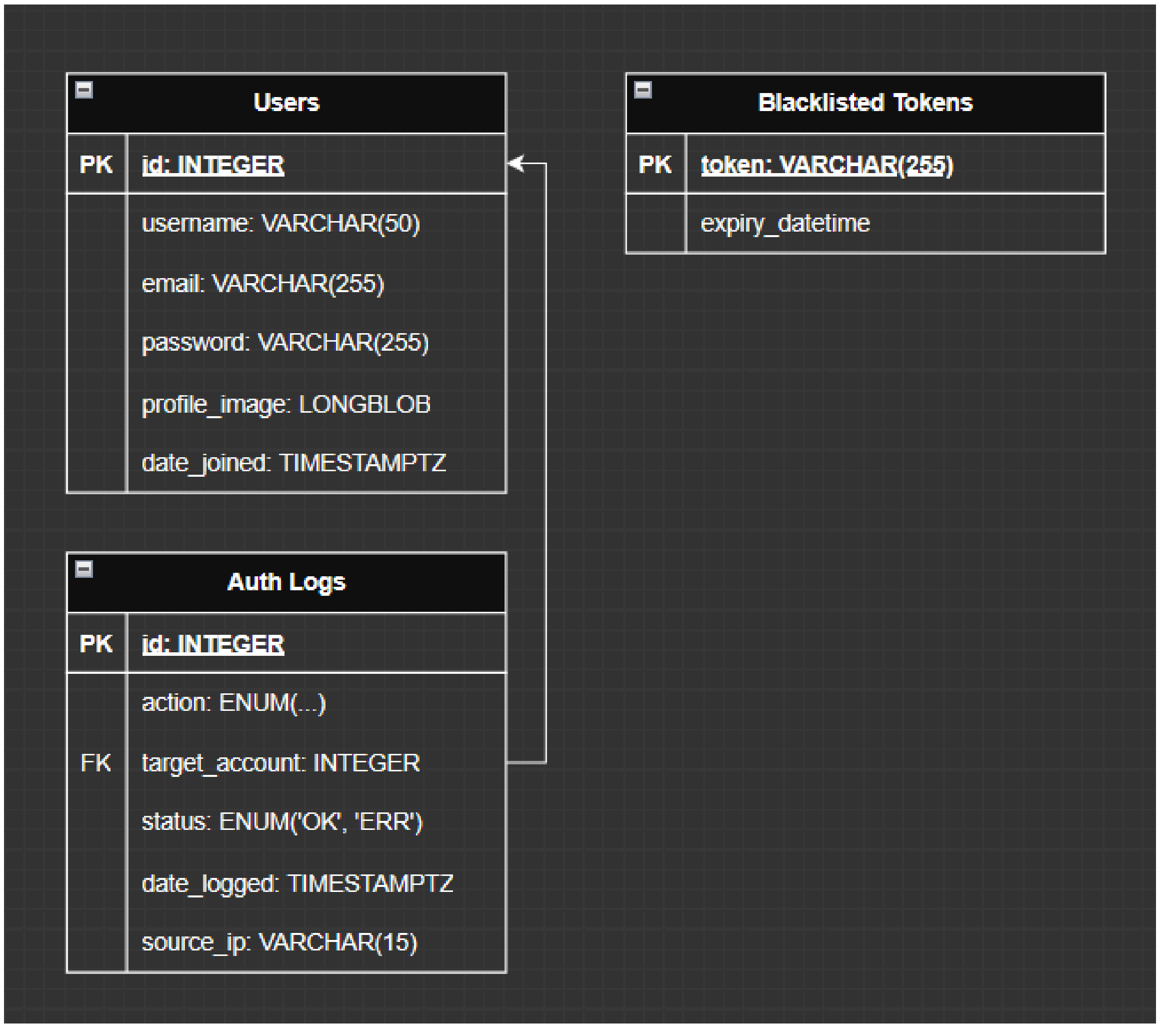
- Authentication logs shall be recorded consistently for auditing purposes.

Maintainability:

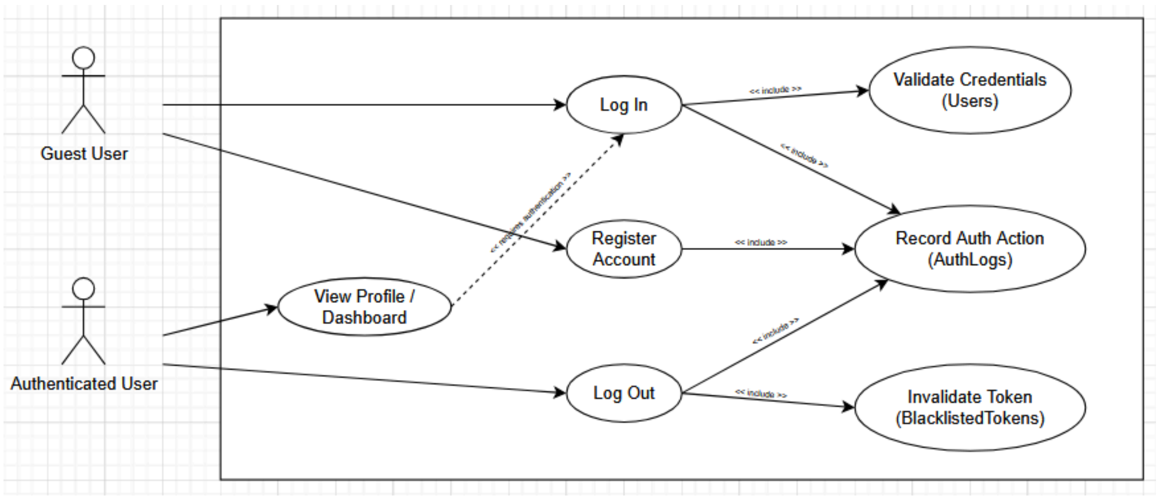
5. The system shall follow a layered architecture (Controller, Service, Repository) to allow easy updates and extensions.

6. System Models (Diagrams)

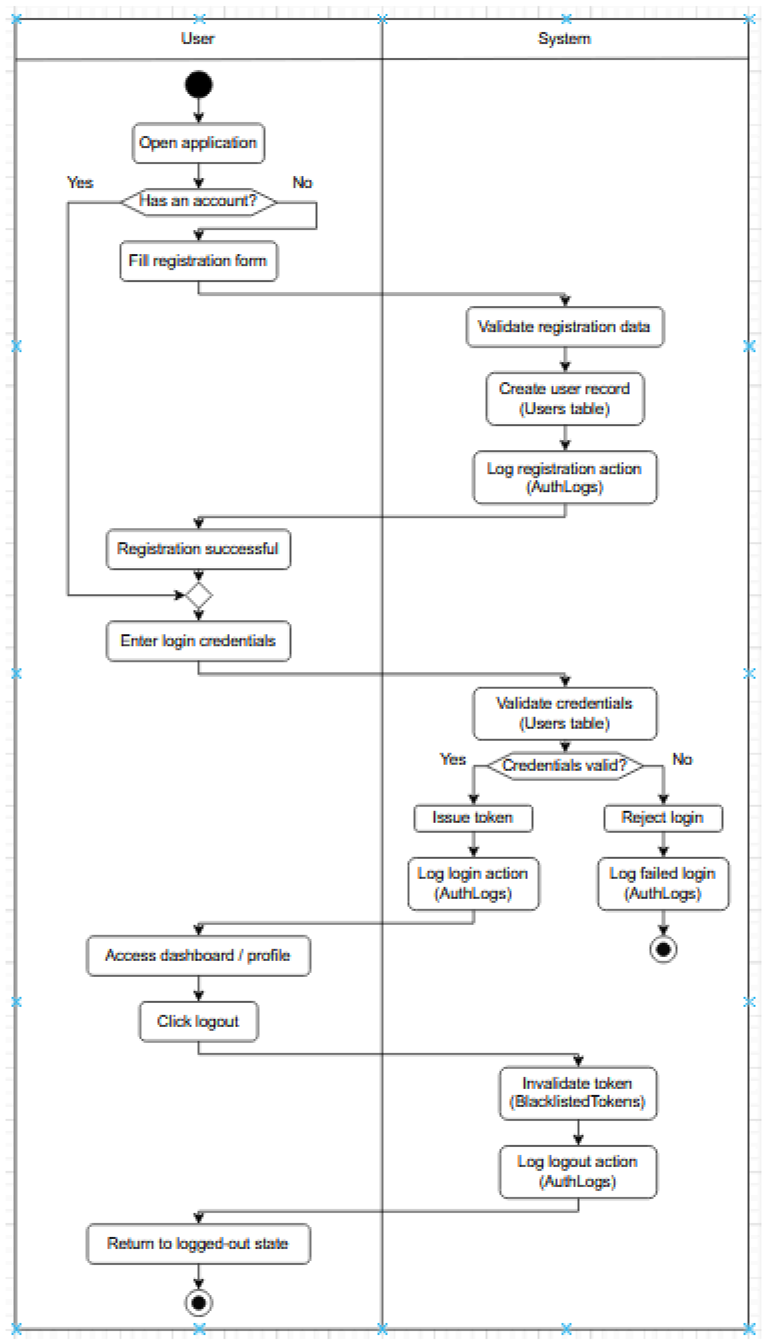
6.1. ERD



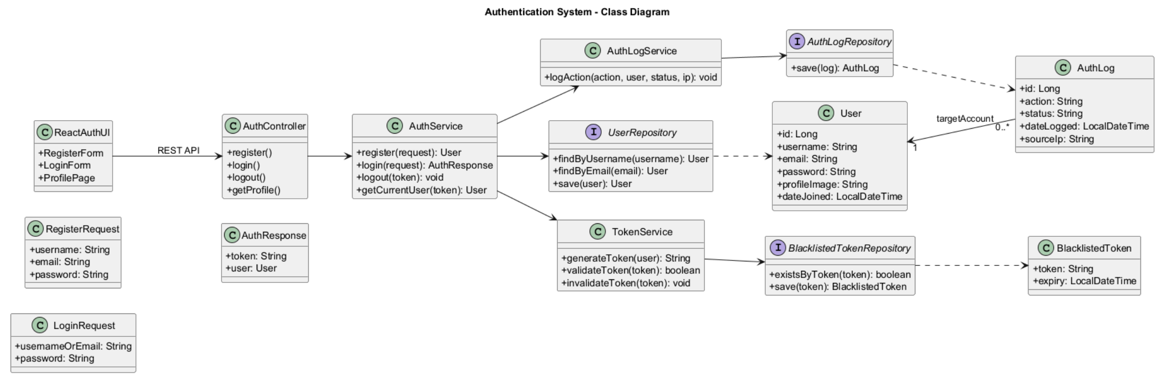
6.2. Use Case Diagram


Guest User
Authenticated User
Log In
Register Account
View Profile / Dashboard
Log Out
Validate Credentials (Users)
Record Auth Action (AuthLogs)
Invalidate Token (BlacklistedTokens)
<< include >>
<< requires authentication >>

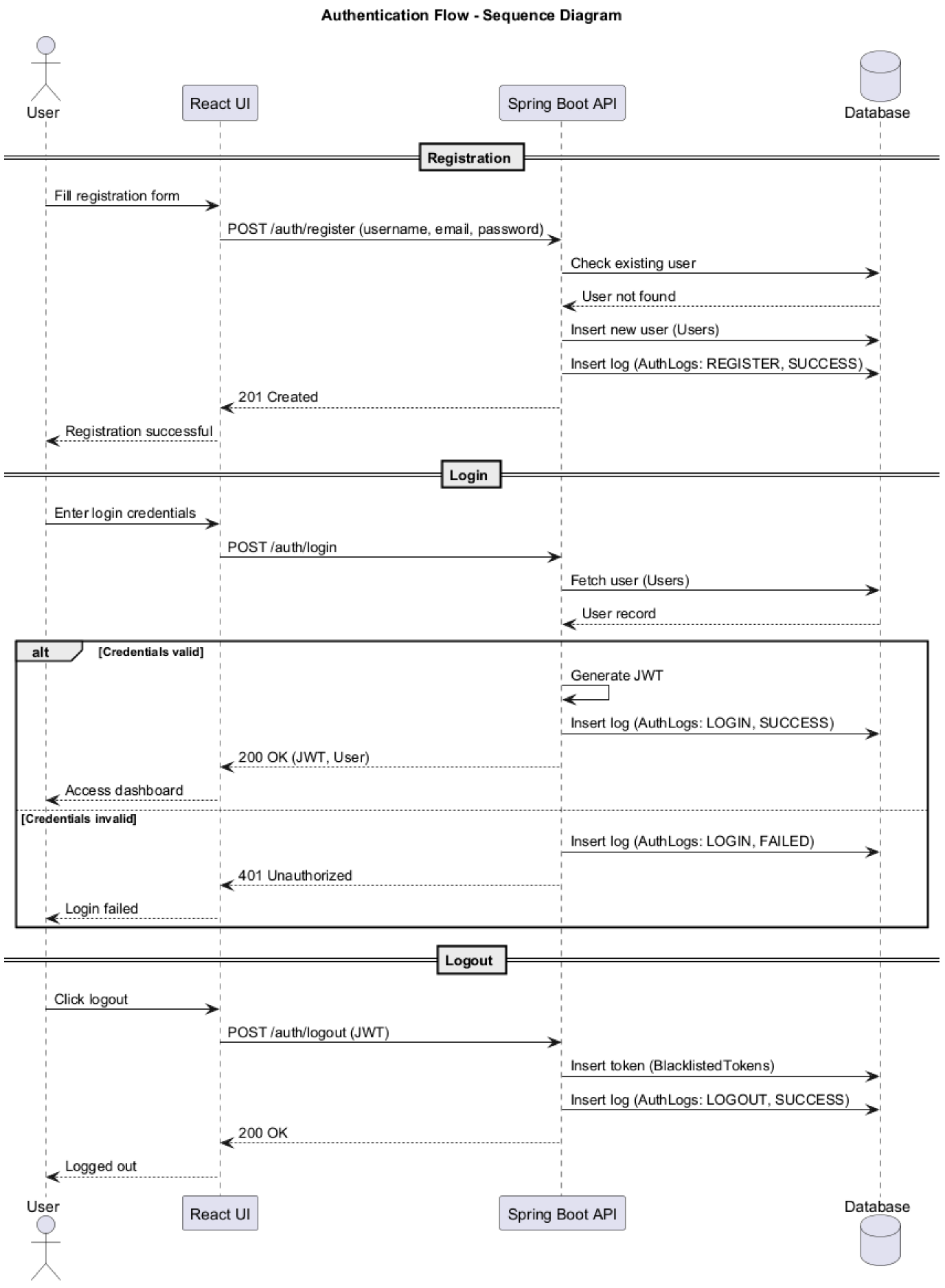
6.3. Activity Diagram


User
System
Open application
Yes
Has an account?
No
Fill registration form
Validate registration data
Create user record (Users table)
Log registration action (AuthLogs)
Registration successful
Enter login credentials
Validate credentials (Users table)
Yes
Credentials valid?
No
Issue token
Reject login
Log login action (AuthLogs)
Log failed login (AuthLogs)
Access dashboard / profile
Click logout
Invalidate token (BlacklistedTokens)
Log logout action (AuthLogs)
Return to logged-out state

6.4. Class Diagram



6.5. Sequence Diagram


Authentication Flow - Sequence Diagram
User
React UI
Spring Boot API
Database
Registration
Fill registration form
POST /auth/register (username, email, password)
Check existing user
User not found
Insert new user (Users)
Insert log (AuthLogs: REGISTER, SUCCESS)
201 Created
Registration successful
Login
Enter login credentials
POST /auth/login
Fetch user (Users)
User record
alt
[Credentials valid]
Generate JWT
Insert log (AuthLogs: LOGIN, SUCCESS)
200 OK (JWT, User)
Access dashboard
[Credentials invalid]
Insert log (AuthLogs: LOGIN, FAILED)
401 Unauthorized
Login failed
Logout
Click logout
POST /auth/logout (JWT)
Insert token (BlacklistedTokens)
Insert log (AuthLogs: LOGOUT, SUCCESS)
200 OK
Logged out
User
React UI
Spring Boot API
Database