CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342-G1
SYSTEMS INTEGRATION AND ARCHITECTURE 1

FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

Project Title: User Registration and Authentication

Prepared By: Lichael Yashua E. Ursulo

Date of Submission: 02/14/2026

Version: 3

Table of Contents

1. Introduction

1.1. Purpose

This document specifies the requirements for a User Registration and Authentication system that controls access to the application through secure login and session handling. It is intended for developers, instructors, and stakeholders who will review the system documentation and implement the solution during the face-to-face coding session.

1.2. Scope

The system will support account registration, user login, viewing of a protected dashboard/profile, and logout. It enforces access control so that protected pages cannot be viewed while logged out. This scope focuses only on the basic authentication flow and does not include features such as password recovery, email verification, multi-factor authentication, or third-party login integrations.

1.3. Definitions, Acronyms, and Abbreviations

- **FRS – Functional Requirements Specification:** A document that explains what the system should do, including its behaviors and key features.
- **UI – User Interface:** The part of the application where the user interacts with the system (screens, forms, buttons).
- **API – Application Programming Interface:** Rules and endpoints that let the React frontend communicate with the Spring Boot backend.
- **ERD – Entity Relationship Diagram:** A diagram that shows the database tables and how the data entities are connected.
- **JWT – JSON Web Token:** A secure token format used to represent an authenticated session and transmit verified information.
- **BCrypt:** A password-hashing method used to securely encrypt passwords before saving them in the database.
- **ReactJS:** A JavaScript library used to build the frontend interface and handle client-side logic.
- **Spring Boot:** A Java framework used to build REST APIs and manage backend processing.
- **MySQL:** A relational database management system used to store user credentials and profile data.

2. Overall Description

2.1. System Perspective

The authentication system functions as a core module that validates user identity before allowing access to protected pages. It follows a typical three-tier web architecture: React handles the user interface, Spring Boot processes requests and applies business rules, and the database stores user credentials and profile information.

2.2. User Classes and Characteristics

- **Guest User (Unauthenticated):** Can access public pages, register an account, and attempt to log in.
- **Authenticated User:** Can access the dashboard/profile page and perform logout.

2.3. Operating Environment

The system will operate in modern web browsers for the frontend interface. The backend will run in a Java Spring Boot REST environment, connected to a relational database for storing user data.

2.4. Assumptions and Dependencies

- Users are expected to have stable internet access to use the web system.
- The frontend depends on the backend API being available and reachable.
- The system depends on the database being active and properly configured with the required users table.
- The system assumes passwords are stored in hashed form (not plain text) for security.

3. System Features and Functional Requirements

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

3.1. Feature 1: Account Registration

Description: Allows a guest user to create an account by submitting valid credentials.
Functional Requirements:

- The system shall allow guest users to register using a unique username and/or email.
- The system shall validate required fields (e.g., username, email, password) before account creation.
- The system shall reject registration if the username or email already exists.
- The system shall hash the user's password (e.g., BCrypt) before saving it to the database.
- The system shall store the new user record only after successful validation.

3.2. Feature 2: Login, Session Handling, and Logout

Description: Allows registered users to log in, access protected pages, and end their session securely.
Functional Requirements:

- The system shall verify login credentials against stored user records in the database.
- The system shall deny access and display an error message when credentials are invalid.
- The system shall issue an authentication session/token (e.g., JWT) after successful login.

- The system shall block unauthenticated access to protected pages (dashboard/profile).
- The system shall redirect unauthenticated users to the login page when they attempt to access protected pages.
- The system shall terminate the session (clear token/session data) when the user logs out.

4. Non-Functional Requirements

- **Security:** Passwords must be stored using secure hashing (e.g., BCrypt) and protected pages must require authentication checks.
- **Performance:** Login and registration requests should respond within a reasonable time under normal usage.
- **Usability:** Forms and messages should be clear and easy to understand for guest and authenticated users.
- **Reliability:** The system should handle invalid input and failed authentication attempts with proper error feedback.
- **Availability:** The system requires working network connectivity and operational backend/database services to function correctly.

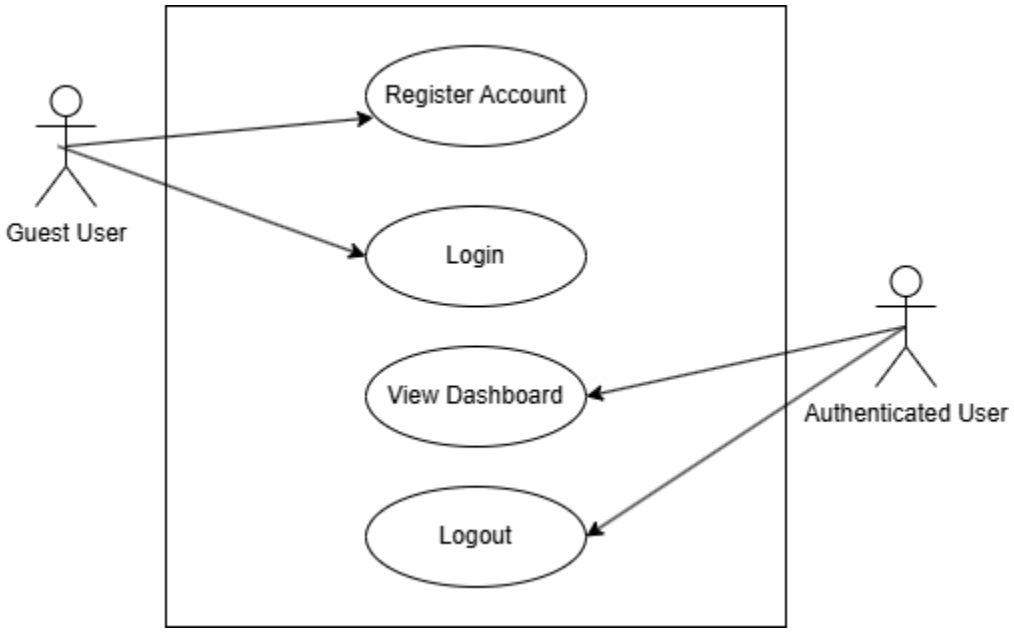
5. System Models (Diagrams)

Insert the necessary diagrams for the system:

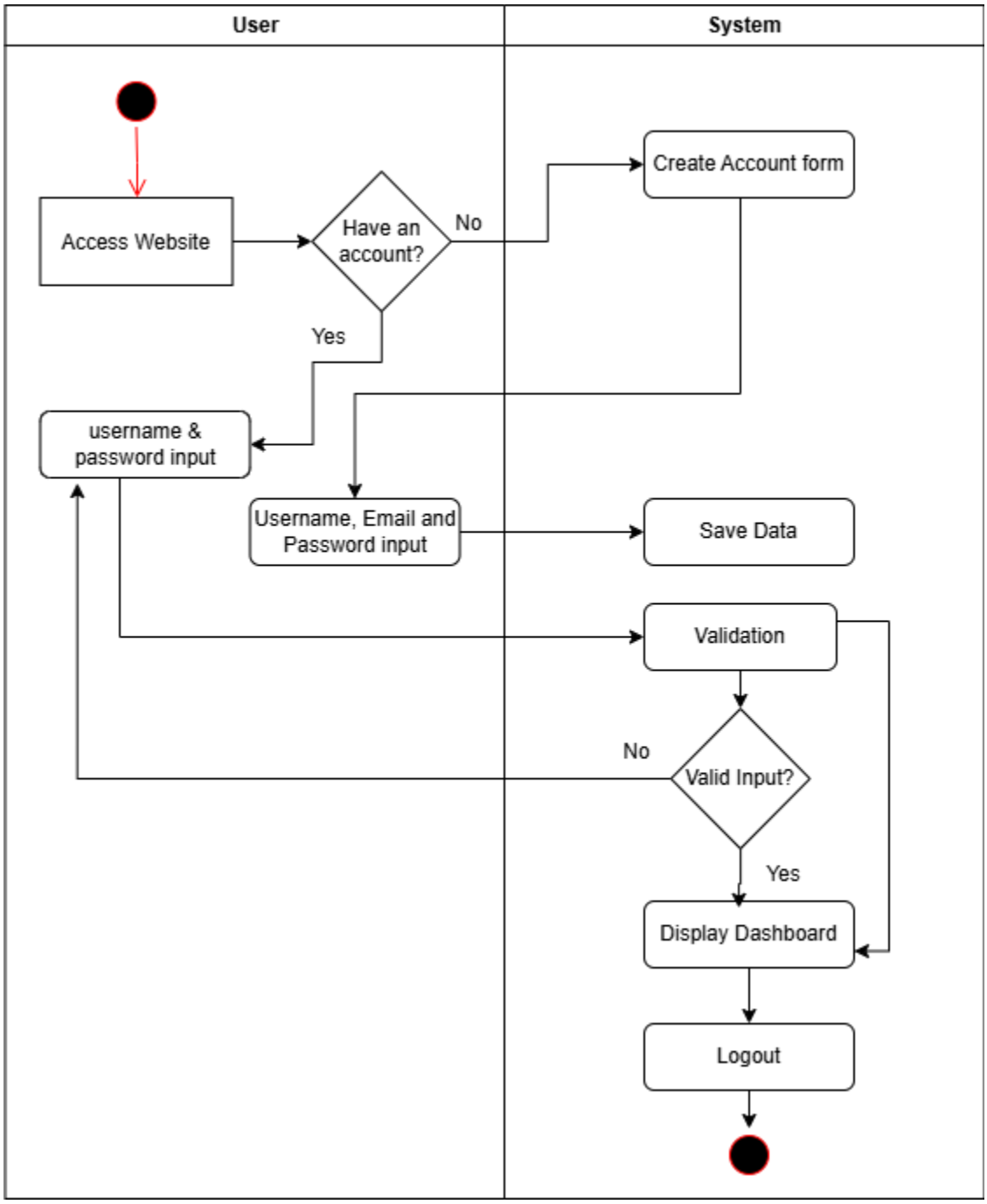
5.1. ERD

User	
PK	user_id
	username
	email
	password
	created_at

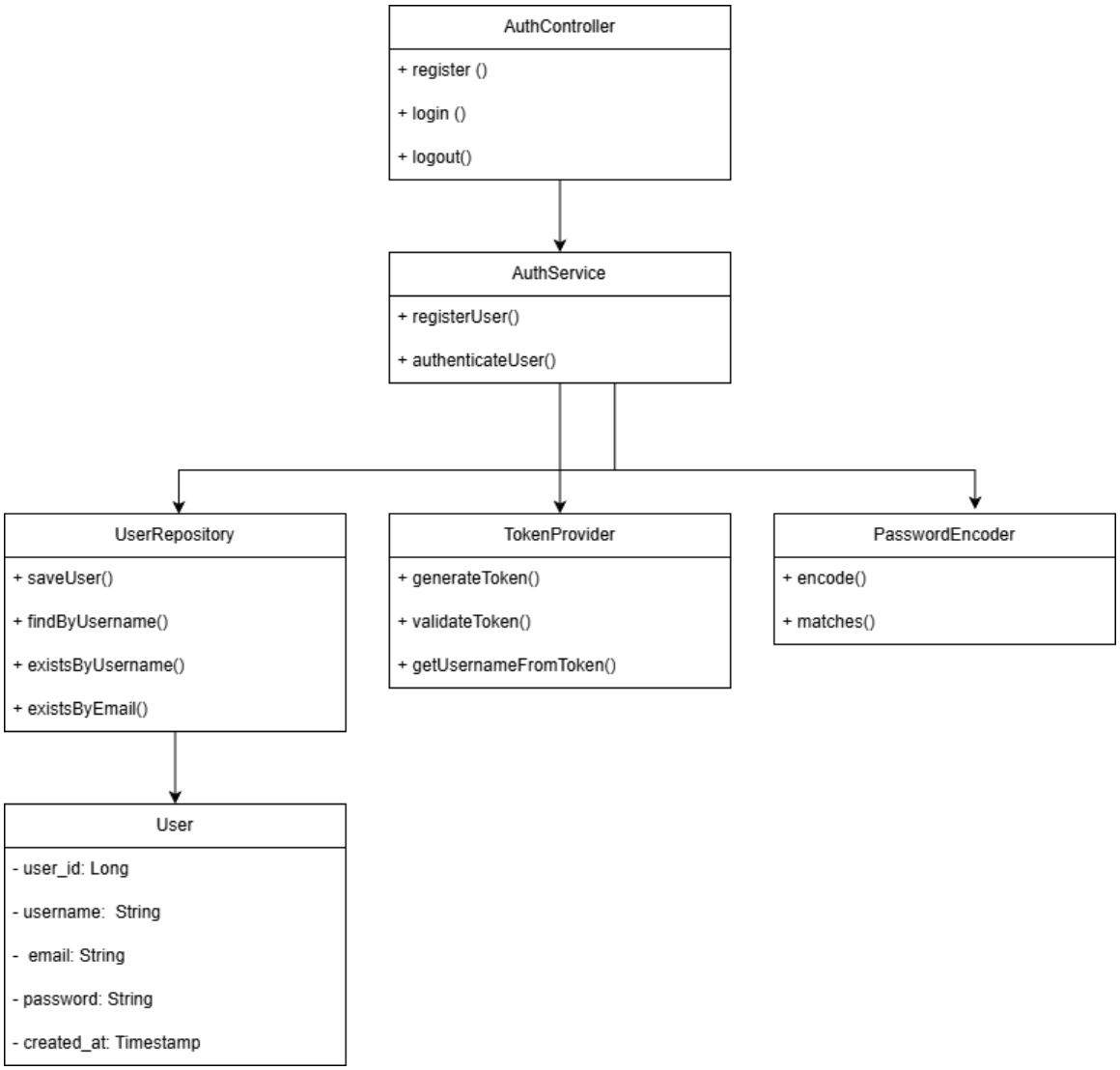
5.2. Use Case Diagram



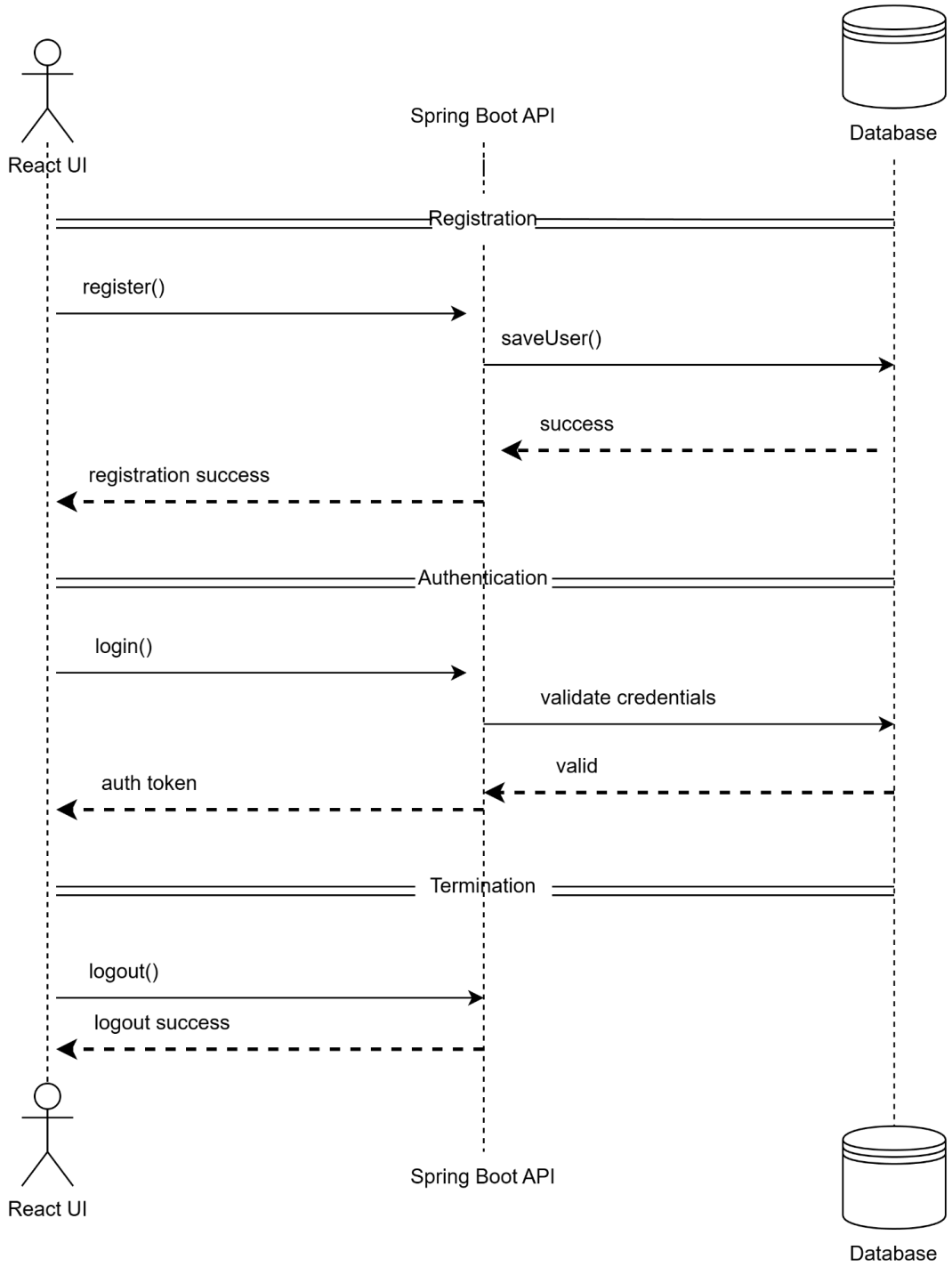
5.3. Activity Diagram



5.4. Class Diagram



5.5. Sequence Diagram

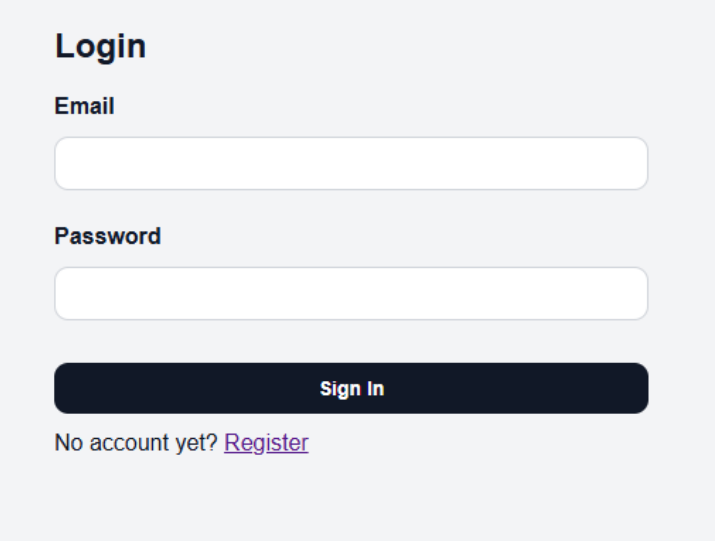


6. Appendices

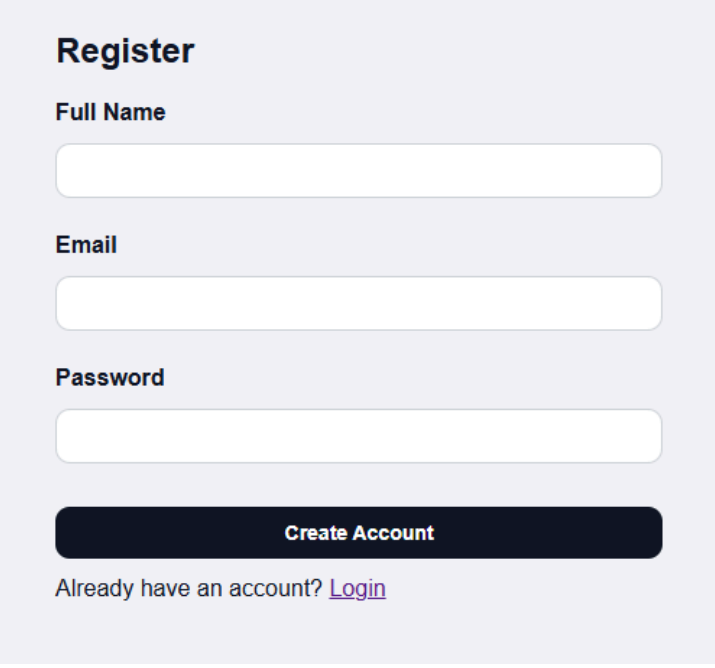
- **Technology Stack:** Built using ReactJS for the user interface, Spring Boot for the backend API, and MySQL as the database for storing user information.
- **Security Notes:** Passwords are protected using BCrypt hashing, and user sessions are handled using JSON Web Tokens (JWT).
- **Diagramming Tools:** All system diagrams were created using diagrams.net / draw.io.
- **External Requirements:** The system assumes stable internet access and properly configured backend and database services for correct operation.

Screenshots

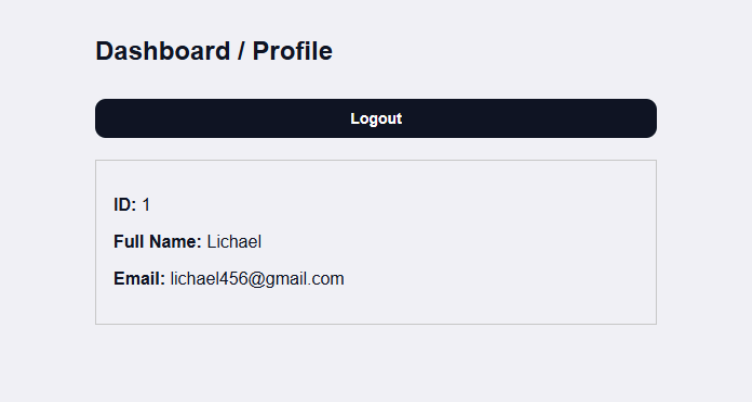
Login Page Web:



Register Page Web:



Dashboard/Profile Page:



Android Dashboard/Profile, Login, and Register pages: