CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342-G1
SYSTEMS INTEGRATION AND ARCHITECTURE 1

FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

Project Title: User Registration & Authentication

Prepared By: Franz Raven R. Sanchez

Date of Submission: 2/7/2026

Version: 3

Table of Contents

1. Introduction

1.1. Purpose

The goal of this specification is to establish a clear framework for the User and Authentication System by defining its core functionalities and architectural design. It provides a thorough description of the system's operational constraints and requirements to support effective implementation. Furthermore, it acts as a foundational reference for evaluating current performance and planning future system iterations.

1.2. Scope

The User and Authentication System is built to facilitate secure account creation, authentication, profile administration, and session termination. To maintain system integrity, access control mechanisms are implemented to ensure that protected assets are available exclusively to verified users. This document encompasses the full analytical and architectural design of the module, serving as the foundational blueprint for all upcoming implementation and development stages.

1.3. Definitions, Acronyms, and Abbreviations

Term/Acronym	Definition
FRS	Functional Requirements Specification. A document that describes the system's functional and non-functional requirements.
UI	User Interface. The part of the system that allows users to interact with the application.
API	Application Programming Interface. A set of rules that enables communication between system components.
UML	Unified Modeling Language. A standard modeling language used to create system diagrams
JWT	JSON Web Token. A security token used for user authentication.
DB	Database. A structured collection of data used to store user information.
ReactJS	A JavaScript framework used to develop the system's frontend interface.
Springboot	A Java-based framework used to develop

	the system's backend services
Client-Server Architecture	A system structure where the client communicates with the server to request services.
Relational Database	A database system that stores data in tables with defined relationships
Authentication Token	A digital credential generated after login to verify user identity.
Encryption	The process of securing sensitive data such as passwords

2. Overall Description

2.1. System Perspective

The User and Authentication System is a web-based application developed using a client-server architecture. The system will use React-based front-end, a Spring Boot based back-end, and a relational database for data persistence. Communication between system components is handled through RESTful APIs.

2.2. User Classes and Characteristics

User Class	**Description**
Guest User	Users who have not logged in and may register
Authenticated User	Users who have successfully logged in and can access system features

2.3. Operating Environment

The system is intended to operate in the following environment:

- Operating System: Windows, macOS, or Linux
- Frontend Framework: ReactJS
- Backend Framework: Spring Boot (Java)
- Database Management System: MySQL or PostgreSQL
- Web Browser: Chrome, Edge, Firefox
- Development Tools: Visual Studio Code and IntelliJ IDEA

2.4. Assumptions and Dependencies

The system's effective deployment and functionality rely on the following conditions:

- **Connectivity**: Users are expected to have a consistent connection to the internet.\
- **Infrastructure**: Both the application and database servers must remain active and accessible.
- **Dependencies**: All necessary libraries for secure authentication and data encryption must be integrated.
- **Environment**: The hosting environment must be configured to support the Java and Node.js runtime environments.
- **Maintenance**: Routine system updates and maintenance are required to ensure long-term stability.

3. System Features and Functional Requirements

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

3.1. Feature 1: User Registration

Description: This module facilitates the onboarding of new users by capturing essential credentials and establishing a secure record within the database.
Functional Requirements:

- The system shall verify the integrity and format of all user-provided data during the registration process.
- The system shall permit account creation upon the submission of valid personal identifiers.
- The system shall utilize the password encoder to hash credentials before they are committed to the user table.
- The system shall trigger an email verification process as a prerequisite for full account activation.

3.2. Feature 2: User Login

Description: This feature validates the identity of returning users, allowing them to bypass access controls and engage with restricted system components.
Functional Requirements:

- The system shall provide an interface for users to submit their registered email and password for verification.
- The AuthService shall perform a comparison between provided credentials and stored database records.
- The system shall employ the TokenProvider to issue a unique authentication token upon successful identity confirmation.
- The system shall reject access attempts and display error messages for any unauthorized or invalid login requests.
- The system shall grant access to the "View Dashboard" use case following a verified login event.

3.3. Feature 3: User Profile

Description: This functionality empowers verified users to manage their public-facing data and internal account settings through a dedicated interface.

Functional Requirements:

- The system shall enable authenticated users to retrieve and view their specific profile attributes, such as bio, avatar, and cover image
- The system shall allow for the modification of account details, including the username, email and profile imagery.
- The system shall run validation checks on all modified data to ensure it meets system constraints before persistence.
- The system shall update the User entity in the database via the UserRepository once changes are confirmed.
- The system shall provide visual confirmation to the user upon the successful update of their information.

3.4. Feature 4: Logout

Description: This feature provides a secure exit point for users, effectively ending their authenticated state to protect account privacy.

Functional Requirements:

- The system shall offer a global logout command accessible to all authenticated users.
- The system shall invalidate the current session token to prevent subsequent unauthorized requests.
- The system shall remove active user data from the client-side session storage.
- The system shall immediately redirect the user to the public login interface upon session closure.
- The system shall ensure that any attempt to access "View Profile" or "Update Profile" after logout is blocked.

4. Non-Functional Requirements

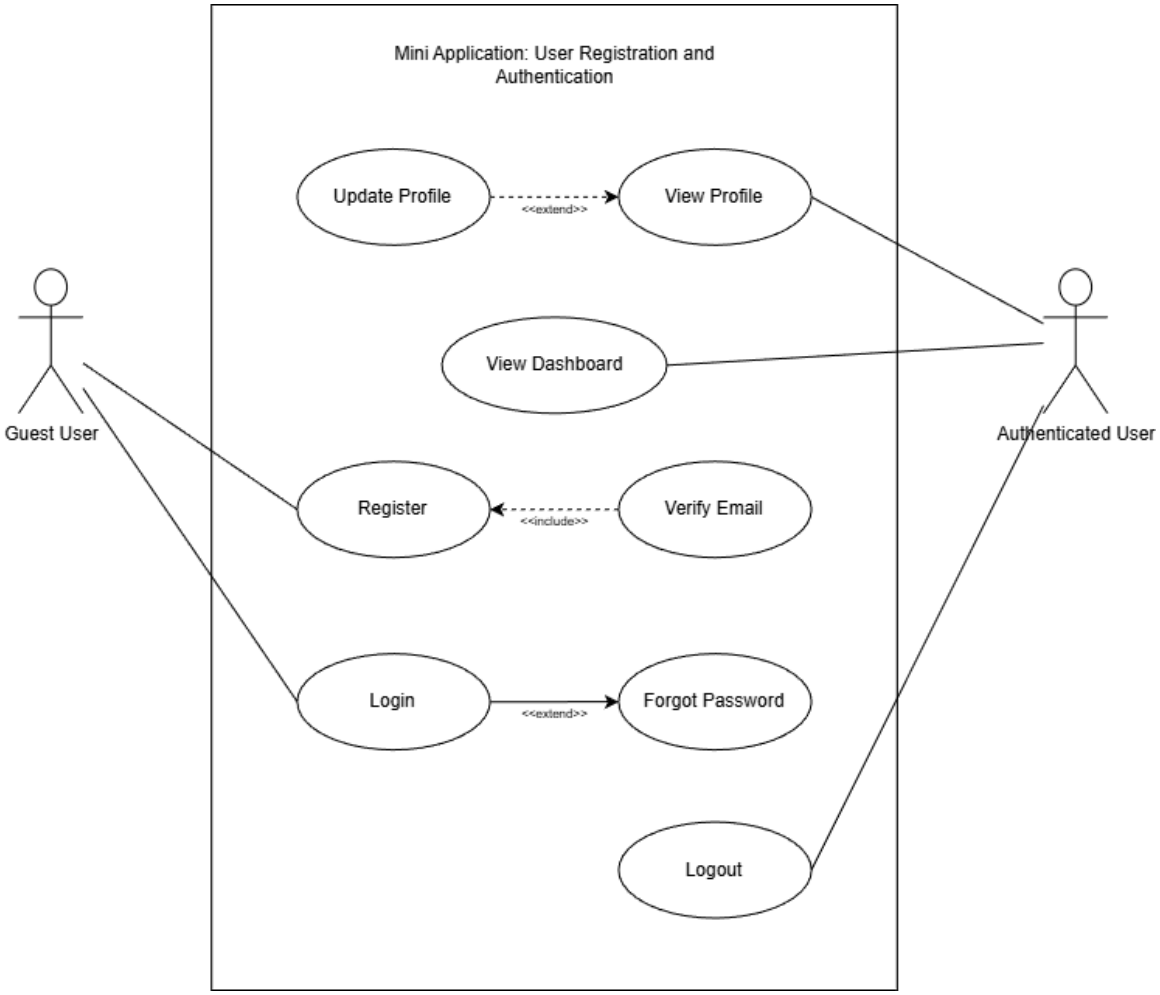
Category	Requirement
Security	User credentials shall be encrypted and protected
Scalability	The system shall support future user growth
Maintanability	The system shall follow modular and documented design
Reliability	The system shall operate consistently under normal conditions

5. System Models (Diagrams)

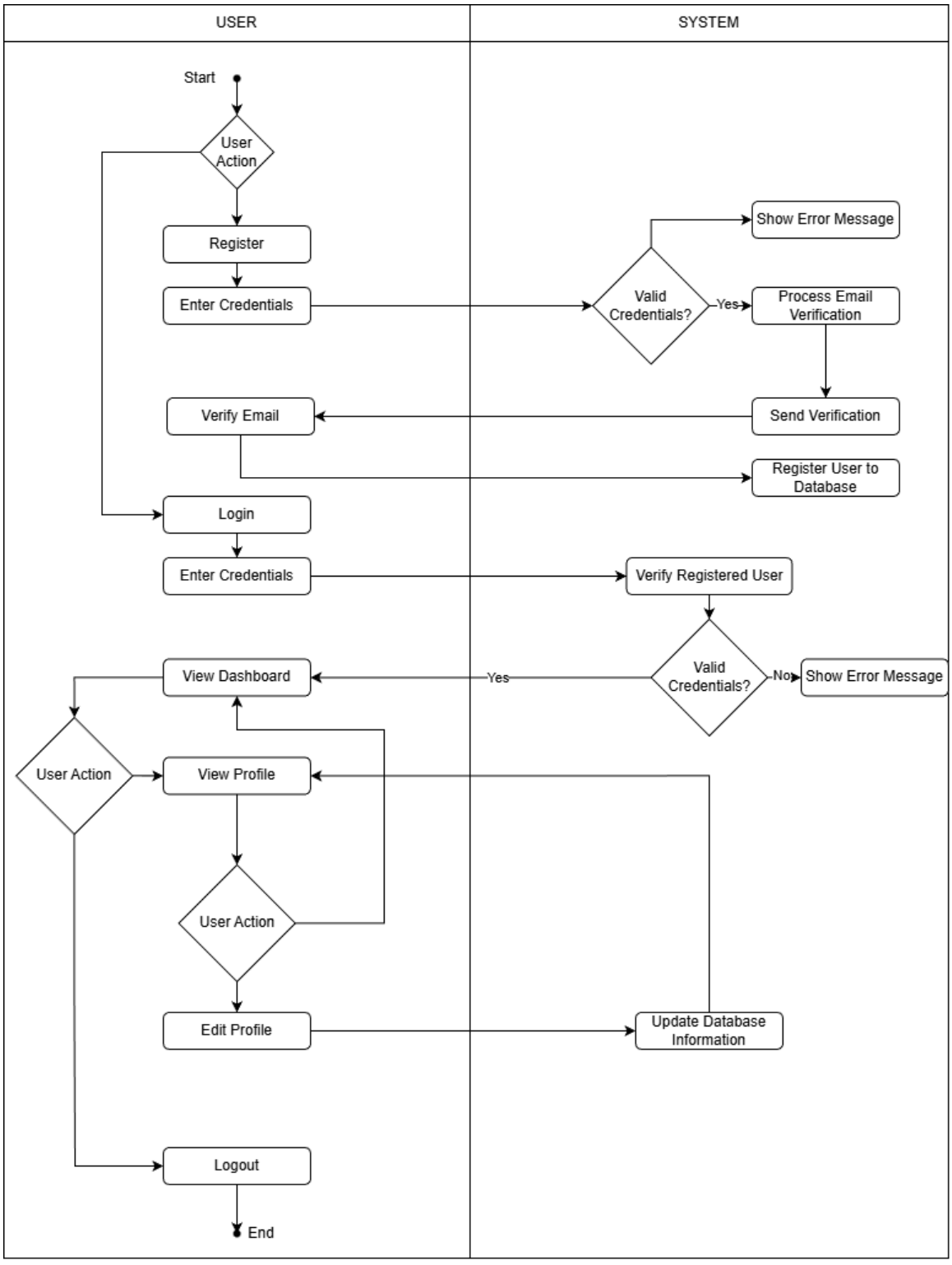
5.1. ERD

User		
PK	id	string
	avatar	string
	bio	string
	cover_image	string
	email	string
	password	string
	username	string
	created_at	timestamp

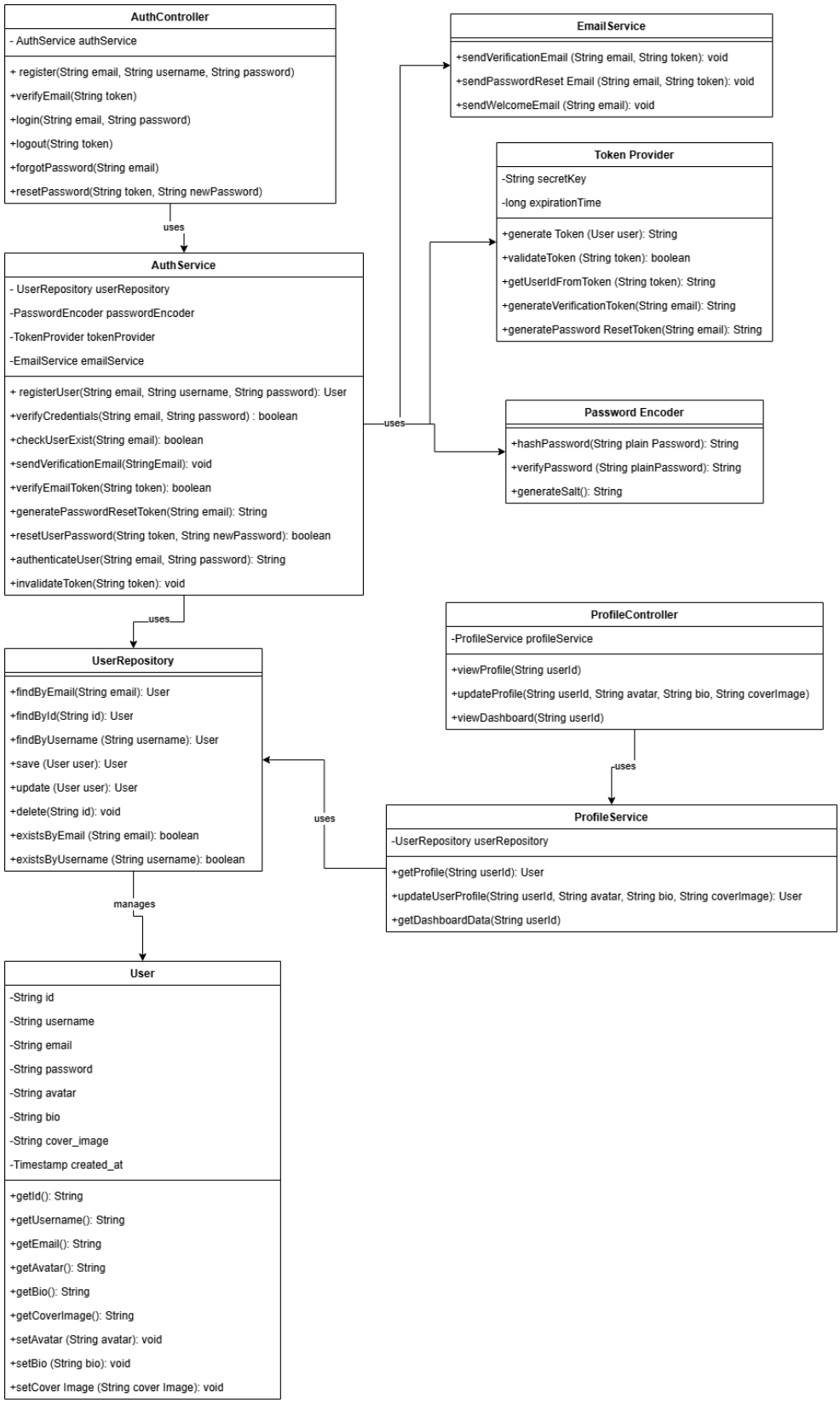
5.2. Use Case Diagram



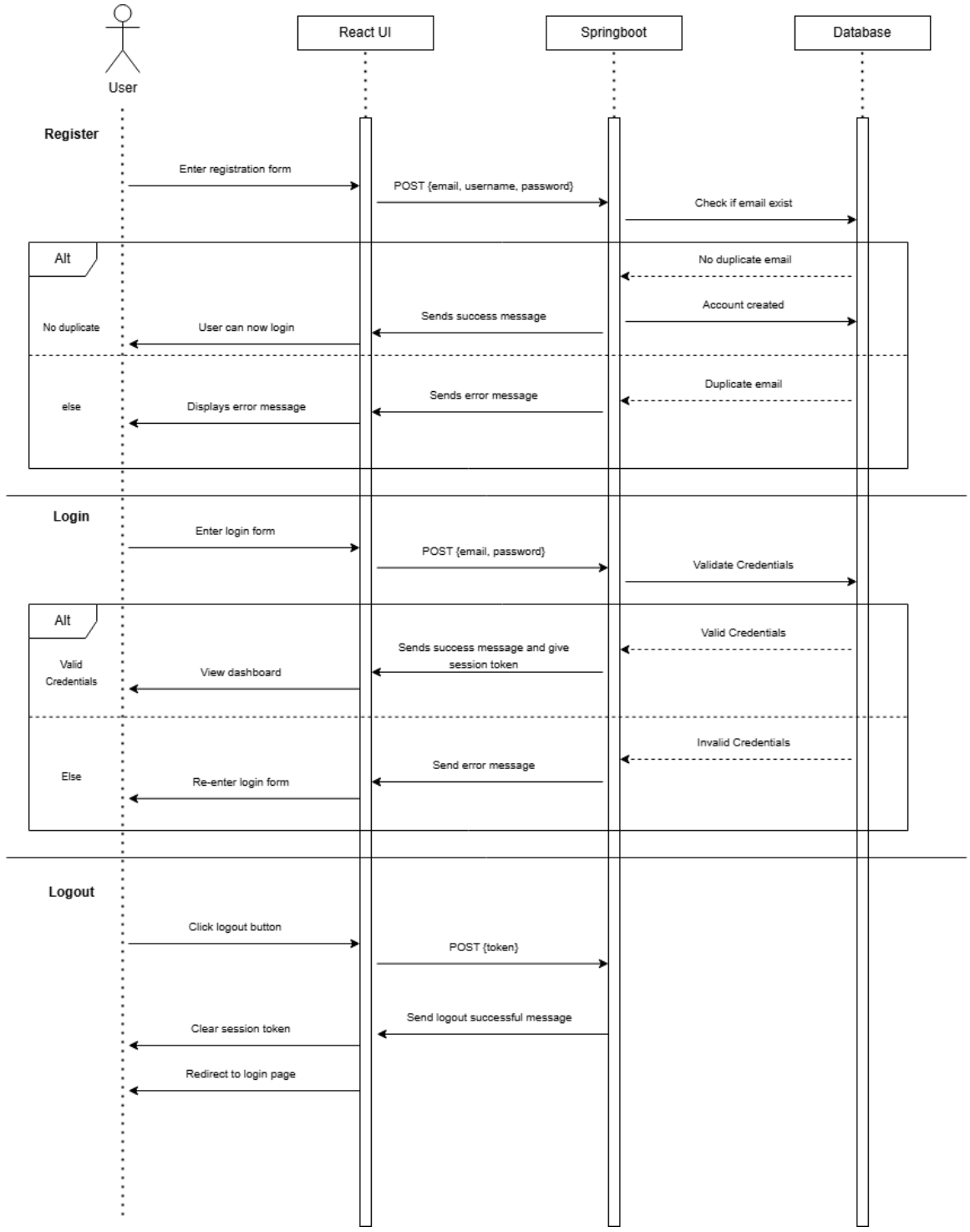
5.3. Activity Diagram



5.4. Class Diagram



5.5. Sequence Diagram



6. Appendices

References

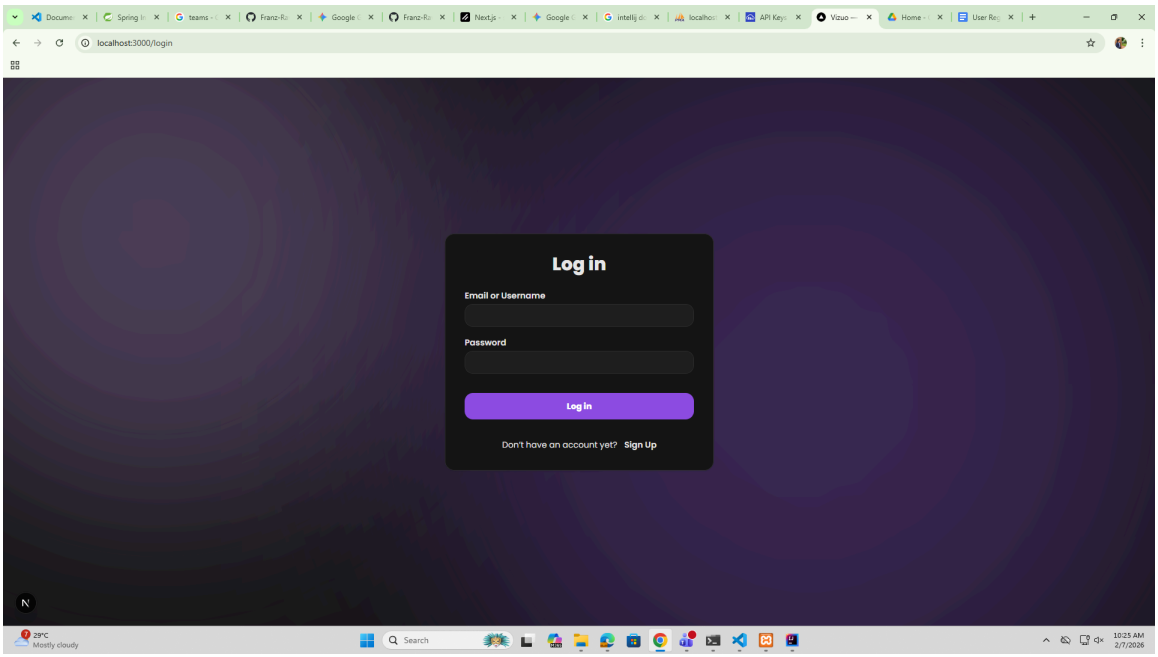
- Spring Boot Technical Documentation
- ReactJS Development Guide
- UML Modeling Standards
- JWT Security Specifications

Supporting Materials

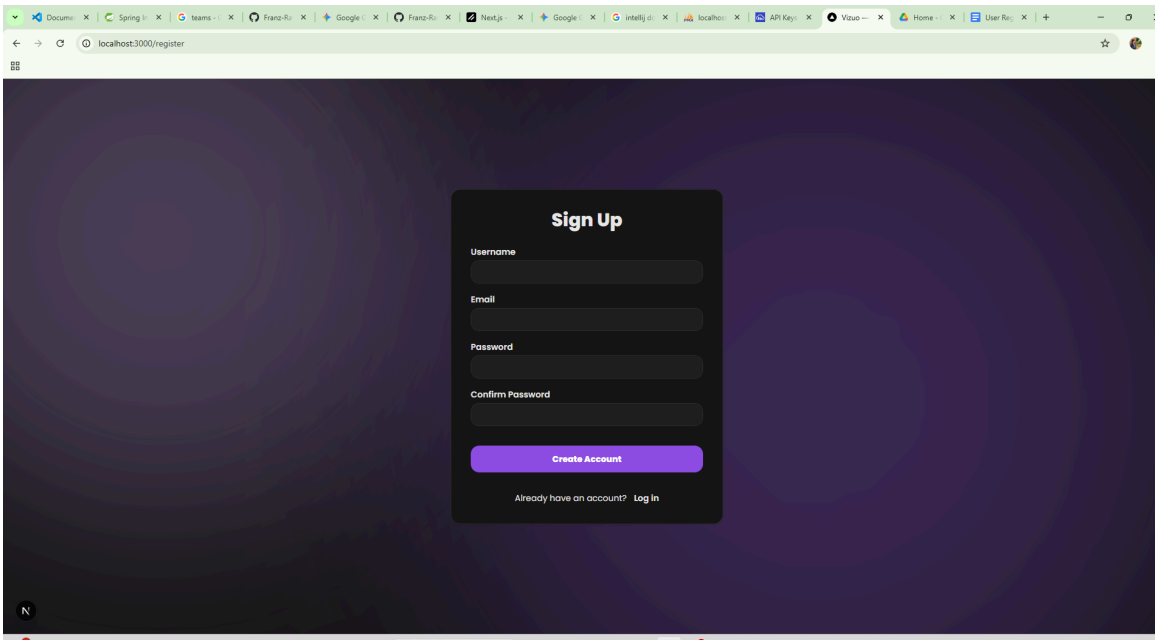
- System diagrams created using draw.io
- Development notes and planning documents
- Implementation guidelines for future phases

7. User Interface

LOGIN



REGISTER


localhost:3000/register
Sign Up
Username
Email
Password
Confirm Password
Create Account
Already have an account? Log in