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: Joseph James Banico

Date of Submission: February 3, 2026

Version: 1.0

Table of Contents

1. Introduction

1.1. Purpose

This paper outlines the Mini App's User Registration & Authentication System's functional and non-functional needs. It is intended to help students, teachers, and developers understand, create, and implement the system.

1.2. Scope

The system will allow users to create an account, log in, view a personal profile or dashboard, and safely log out. It is going to enforce limits on access, preventing unauthorized individuals from accessing restricted pages. This document's focus is restricted to system analysis, design, and supporting diagrams.

1.3. Definitions, Acronyms, and Abbreviations

- API (Application Programming Interface) – A set of protocols that allows the ReactJS frontend to communicate with the Spring Boot backend.
- ERD (Entity Relationship Diagram) – A visual diagram used to show the structure and relationships of the data within the database.
- Firebase – A platform provided by Google used in this project for user authentication and cloud data storage.
- FRS (Functional Requirements Specification) – The formal document that describes the intended features and behavior of the system.
- JDK (Java Development Kit) – The necessary tools and environment required to run and develop the Spring Boot backend.
- ReactJS – A JavaScript library used for building the user interface and frontend components of the application.
- Spring Boot – A Java-based framework used to build the backend REST API that handles the system's logic.
- UML (Unified Modeling Language) – A standard way to visualize system designs, specifically used for the Class and Sequence diagrams in this report.

2. Overall Description

2.1. System Perspective

The Mini App - User Registration & Authentication System is part of a bigger web application network. It acts as the user management gateway, allowing for secure registration, login, and logout. The system communicates with a ReactJS-based frontend, a Spring Boot backend API, and a Firebase database for authentication and data storage. In this environment, only authorized users may access protected resources, while guest users have restricted access to basic functionality. This modular architecture enables the system to be extended with other application components in the future, such as new dashboards, services, or third-party interfaces.

2.2. User Classes and Characteristics

Guest User:

- Has no account or is not logged in.

- Can register or log in to the system.

Authenticated User

-Has successfully logged in.

-Can view their dashboard/profile and log out.

2.3. Operating Environment

Frontend: ReactJS application, compatible with modern web browsers such as Chrome, Firefox, or Edge.

Backend: Spring Boot REST API running on a system with Java JDK installed.

Database: Firebase for authentication and data storage (Firestore or Realtime Database).

Hardware Requirements: Desktop or laptop with at least 4GB RAM, 2 GHz processor, and stable internet connection.

Tools: draw.io and smartdraw.com to create diagrams.

2.4. Assumptions and Dependencies

- Users have internet access and provide valid registration information.
- ReactJS, Spring Boot, Firebase, Node.js, and JDK for development and operation.

3. System Features and Functional Requirements

3.1. Feature 1: User Registration

Description: Allows a new user to create an account.

Functional Requirements:

- Users can enter required information
- Input data is validated for correctness.
- User data is securely stored in Firebase.

3.2. Feature 2: User Login

Description: Enables registered users to log in.

Functional Requirements:

- Users can enter credentials to access their account.
- The system authenticates users and starts a secure session.
- Incorrect credentials display an error message.

4. Non-Functional Requirements

Performance: The system should load pages and process requests quickly.

Security: All user data must be protected and only accessible to authorized users.

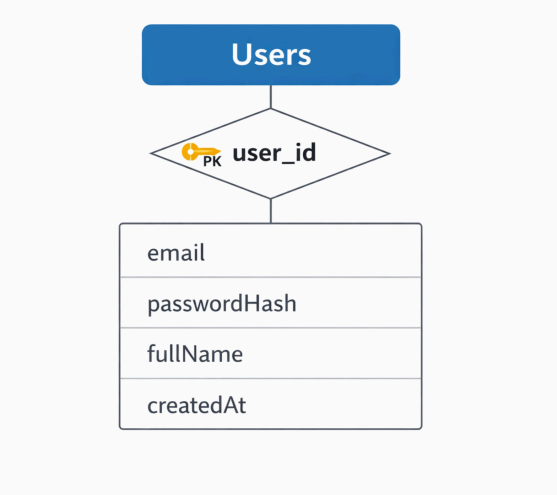
Usability: The system should be easy to navigate and understand.

Reliability: The system should run consistently without unexpected failures.

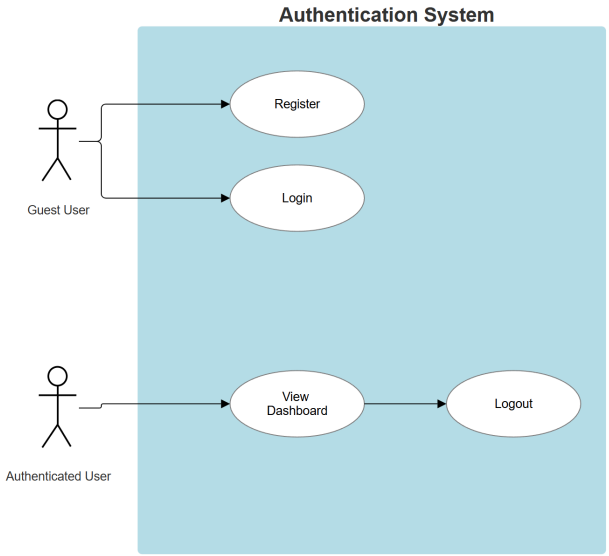
Maintainability: The code and system should be easy to update and manage.

5. System Models (Diagrams)

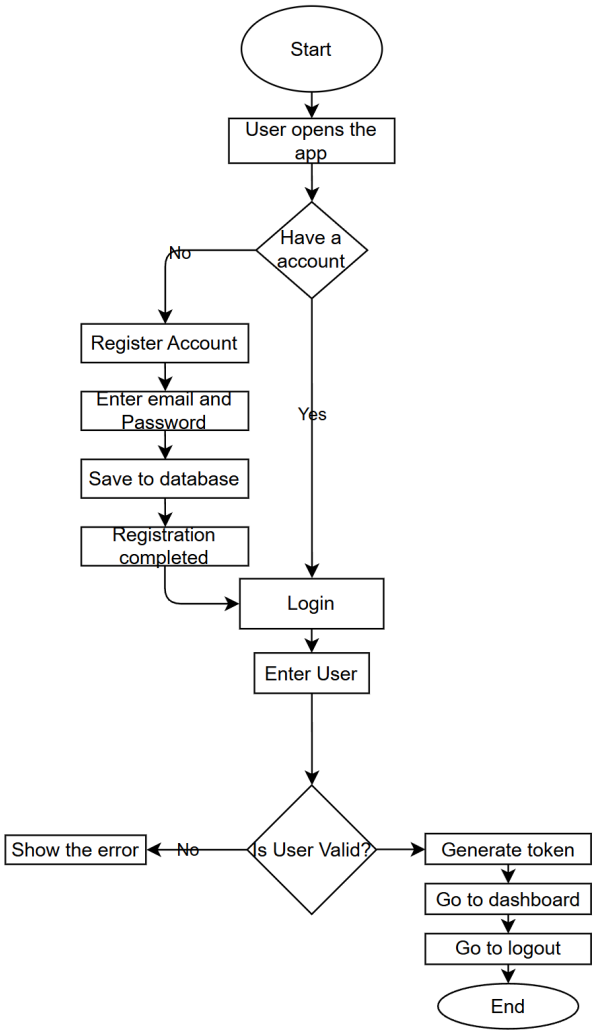
5.1. ERD



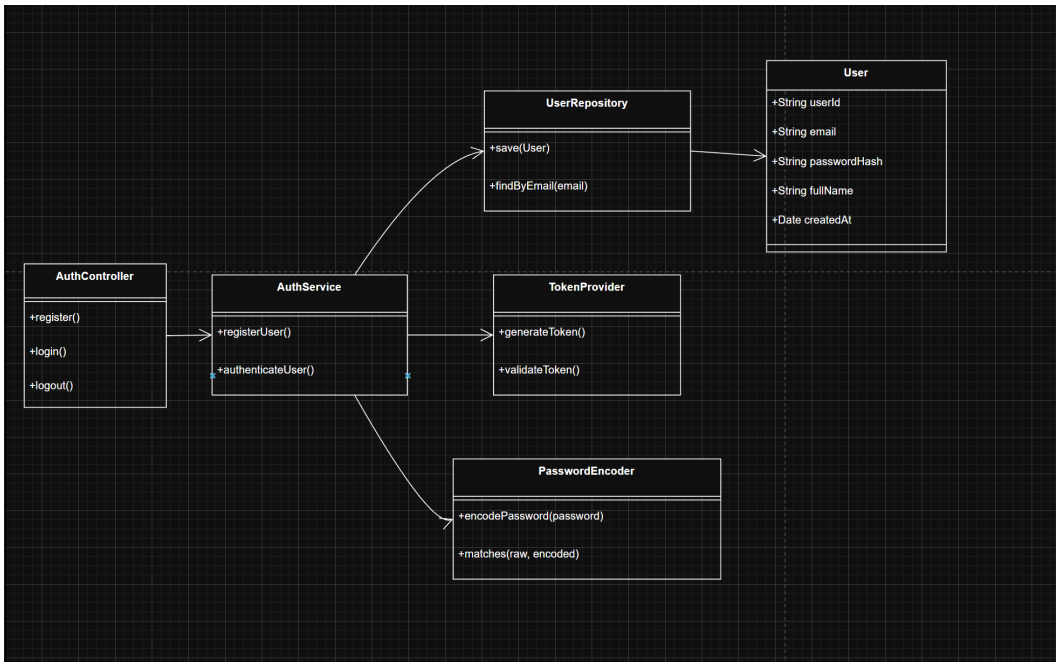
5.2. Use Case Diagram



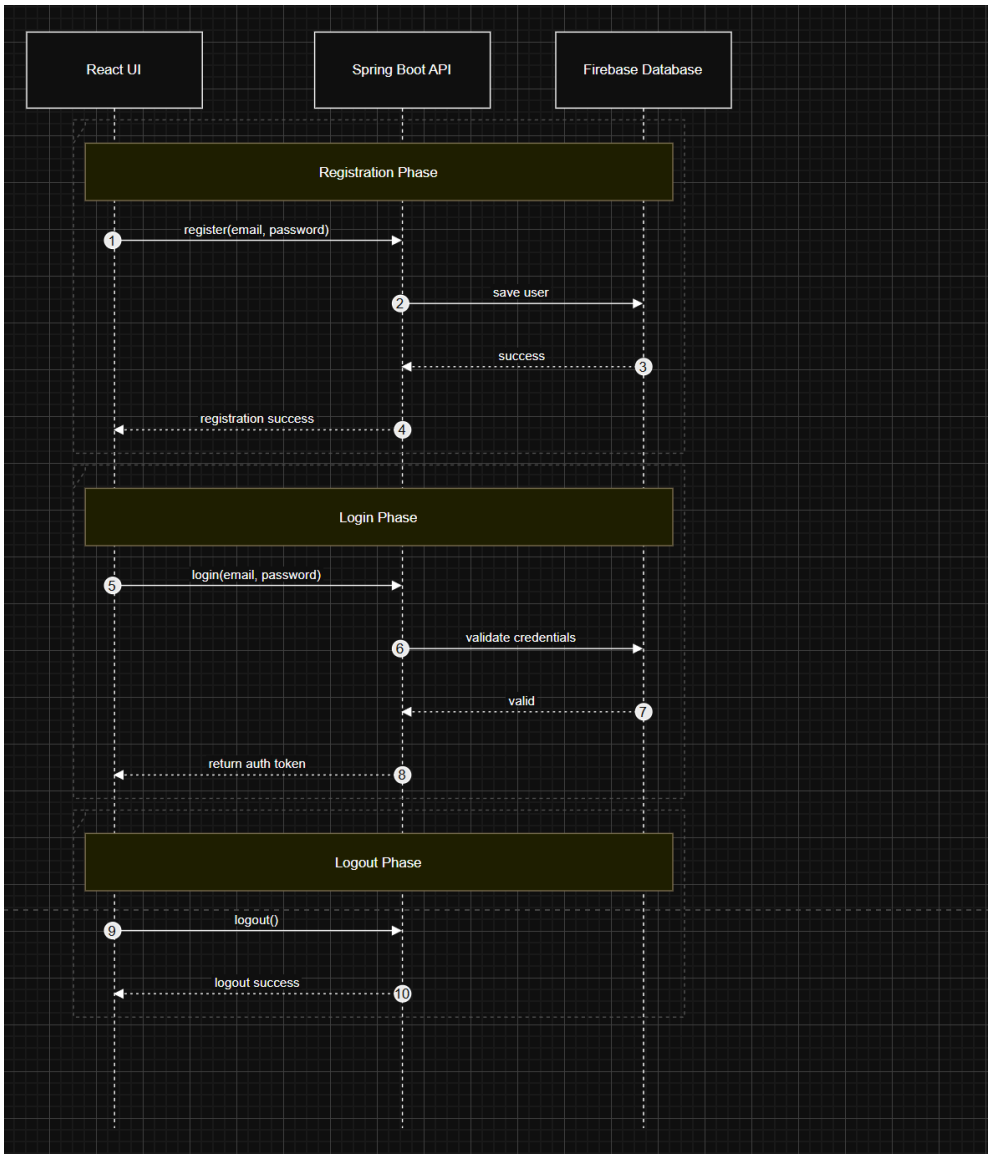
5.3. Activity Diagram



5.4. Class Diagram



5.5. Sequence Diagram



6. Appendices

This section contains the list of software tools, hardware specifications, and technical documentation used to complete this lab activity.