CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342-Section SYSTEMS INTEGRATION AND ARCHITECTURE 1

FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

Project Title: Mini App - User Registration and Authentication

Prepared By: Ervin Louis B. Villas

Date of Submission: 02/07/2026

Version: 1.0

Table of Contents

1. Introduction

1.1. Purpose

The system is designed as a way to demonstrate the user registration and authentication flow and how to modularize different systems that can be inserted into other systems. This document is intended for students, instructors, and developers that will evaluate, implement, and/or integrate the system.

1.2. Scope

The system will allow users to register, login, view and edit their own profile, and logout. For profile view and edit, users can only do this action if they have an account and have logged in to their account. The system does not include password change feature into the system. The system's functionality is focused on user registration, authentication, and profile management.

1.3. Definitions, Acronyms, and Abbreviations

- API - Application Programming Interface
- SQL - Structured Query Language
- Session - A temporary interaction between a user and the system that maintains state across multiple requests
- Authentication - The process of verifying the identity of a user
- Authorization - The process of determining what an authenticated user is allowed to do
- Profile - A user's personal information page containing details such as name, email, and other account information

2. Overall Description

2.1. System Perspective

The system is designed to demonstrate user management capabilities. The website's frontend will be built using ReactJS and shadcn components while the backend and business logic will be handled and built using Java Spring Boot with MySQL or PostgreSQL as the database. Since it is only focused on the authentication, authorization, and profile management, this can be integrated into other systems where it requires user registration and authentication functionality. This can serve as the module that will handle the security of the system.

2.2. User Classes and Characteristics

- Guest User
    - User that has visited the website but does not have an account or is not logged in
    - Can access the registration page to create a new account
    - Can access the login page to authenticate
    - Cannot access profile-related pages
- Authenticated User
    - User that is logged in and is able to visit and edit their profile
    - Has successfully completed registration and logged in with valid credentials
    - Can logout from the system

2.3. Operating Environment

Fronted - ReactJS, shadcn

Backend - Java Spring Boot, Spring Security

Database - MySQL / PostgreSQL

Tools - VS Code, Node.js, npm, MySQL Workbench

2.4. Assumptions and Dependencies

- ReactJS framework and its ecosystem of libraries for frontend functionality
- Spring Boot framework for backend application structure and session management
- Database system (MySQL or PostgreSQL) for data persistence
- shadcn/ui library for consistent UI components
- Java Runtime Environment for backend execution
- Node.js and npm for frontend build and dependency management
- Database JDBC drivers for database connectivity

3. System Features and Functional Requirements

3.1. Feature 1: User Registration

Description: Allows a guest to create an account which is then saved into the database
Functional Requirements:
- The system should provide a registration interface for Name, Email, and Password, and Confirm Password
- The system should validate that all required fields are filled before allowing form submission
- The system should validate that the email address is in a valid email format (contains @ symbol and domain)
- The system should verify that the email address is unique and not already registered in the system
- The system should verify that the password and confirm password fields match exactly
- The system should display appropriate error messages for validation failures

- Upon successful registration, the system should store the user information in the database.
- After successful registration, the system should redirect the user to the profile page

3.2. Feature 2: User Authentication

Description: Validates user identity and provides access to protected pages
Functional Requirements:
- The system should provide a login interface for Email and Password
- The system should validate that both email and password fields are filled before allowing form submission
- The system should verify the provided email exists in the database
- The system should compare the provided password with the stored hashed password for the given email
- The system should display an error message "Invalid email or password" if authentication fails
- Upon successful login, the system should redirect the authenticated user to their profile page

3.3. Feature 3: User Logout

Description: Allows an authenticated user to logout from the website
Functional Requirements:
- The system should provide a logout button/link visible to authenticated users on all protected pages
- When the user clicks logout, the system should invalidate the current session on the server
- After logout, the system should redirect the user to the login page with a message indicating "You have been logged out successfully."

3.4. Feature 4: User Profile Management

Description: Allows authenticated users to view and modify their account details
Functional Requirements:
- The system should restrict access to profile page to authenticated users only
- When an unauthenticated user attempts to access a profile page, the system should redirect them to the login page with a message "Please login to view your profile."
- The system should provide a profile view page displaying the user information with default profile picture
- The system should provide an "Edit Profile" button on the profile view page
- The system should provide an edit profile form to edit the user's Name, Email, and Profile Picture
- The system should validate that the email address is in a valid email format when updating the profile
- If the user changes their email address, the system should verify that the new email address is unique and not already registered by another user
- The system should provide a "Save Changes" button to submit the profile updates

- The system should provide a "Cancel" button or link to discard changes and return to the profile view page
- Upon successful profile update, the system should save the changes to the database
- After successful update, the system should redirect the user to the profile view page with a success message "Profile updated successfully."

4. Non-Functional Requirements

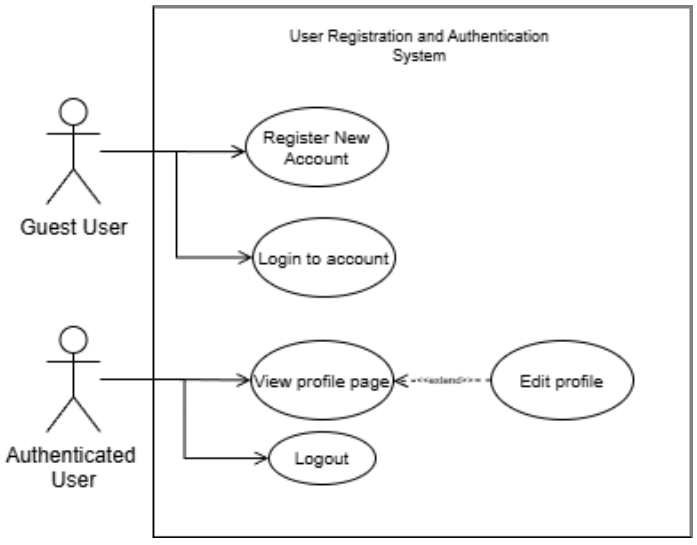
- The system should have a response time of less than 10 seconds
- The system should enable only authenticated users to view and edit their own profile
- The system should display clear and meaningful error messages for all validation failures and system errors
- Form fields should have clearly visible labels indicating what information is required
- Navigation between pages should be intuitive with clearly labeled links and buttons

5. System Models (Diagrams)

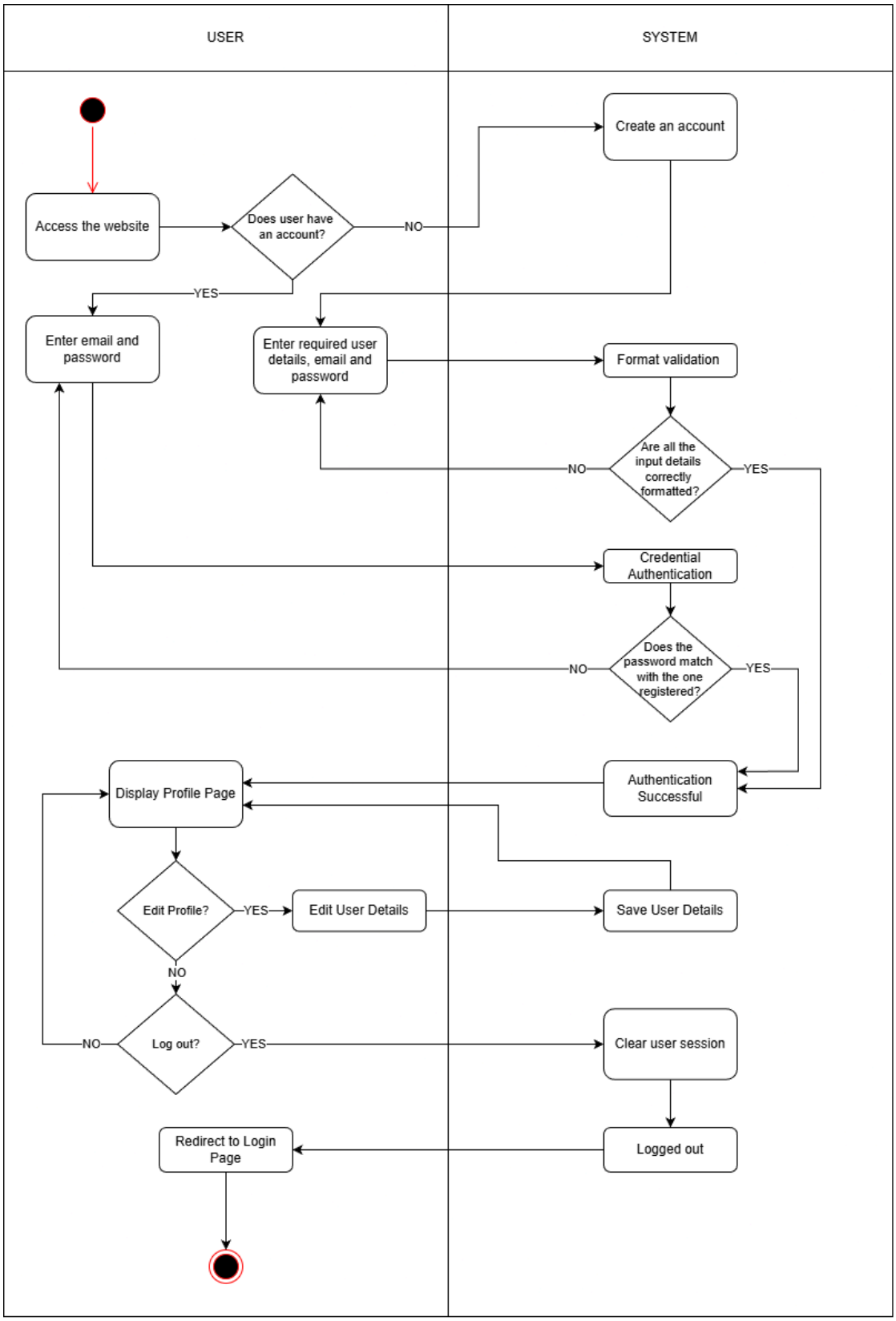
5.1. ERD

User	
PK	userID
	email
	password
	profile_picture
	first_name
	last_name
	created_at
	updated_at

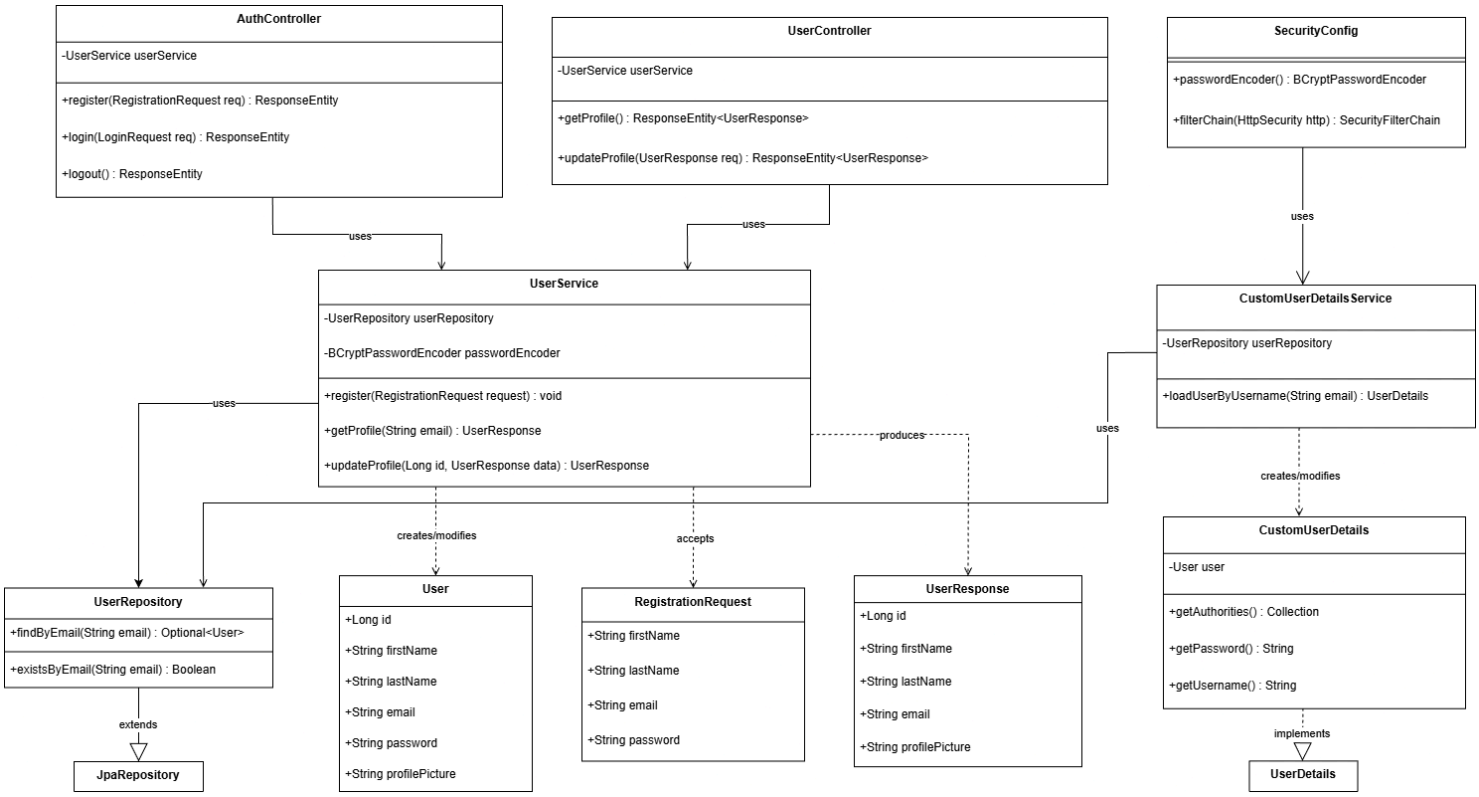
5.2. Use Case Diagram



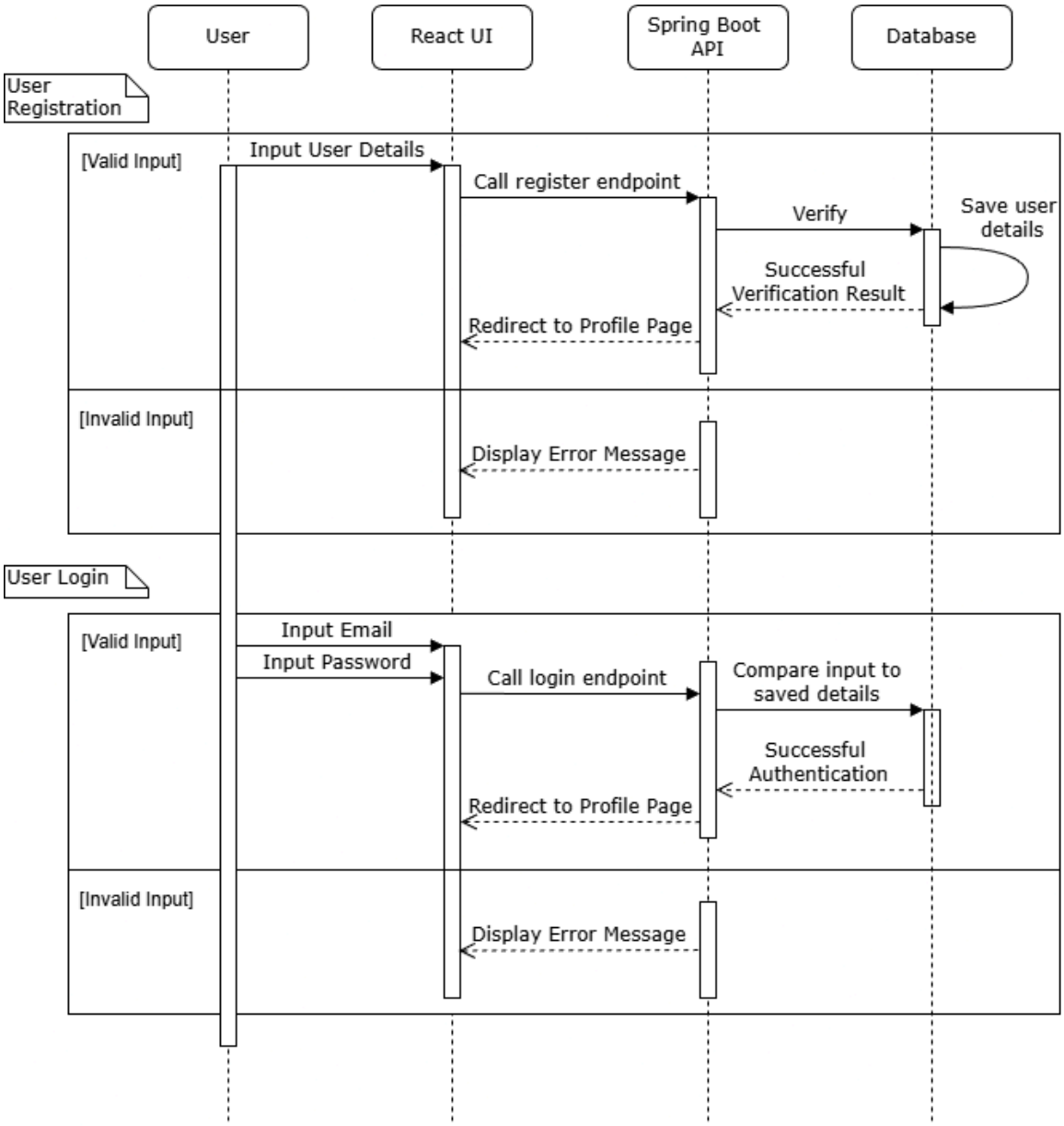
5.3. Activity Diagram



5.4. Class Diagram



5.5. Sequence Diagram



6. Appendices

References:

https://docs.spring.io/spring-boot/reference/web/spring-security.html

https://www.geeksforgeeks.org/system-design/unified-modeling-language-uml-activity-diagrams/

https://www.geeksforgeeks.org/system-design/unified-modeling-language-uml-sequence-diagrams/

https://www.geeksforgeeks.org/system-design/unified-modeling-language-uml-class-diagrams/

https://www.geeksforgeeks.org/advance-java/authentication-and-authorization-in-spring-boot-3-0-with-spring-security/