IT342-Section
SYSTEMS INTEGRATION AND ARCHITECTURE 1

FUNCTIONAL REQUIREMENTS
SPECIFICATION (FRS)

Project Title: Mini App – User Registration & Authentication System

Prepared By: Axcel O. Macansantos

Date of Submission: 2/11/26

Version: 2.0

Table of Contents

1. Introduction

1.1. Purpose

This document strives to explain the functional and non-functional aspects of requirements for the User Registration & Authentication System of the Mini App. This paper is meant for educators, students, and developers who will create, evaluate, and then use sql(XAMPP), Springboot, and ReactJS to create and build the system.

1.2. Scope

The system delivers basic and usable user account management features including account registration, login and logout, access to a protected user dashboard. The system ensures that only authenticated users can access protected pages.

1.3. Definitions, Acronyms, and Abbreviations

- FRS – Functional Requirements Specification
- UI – User Interface
- API – Application Program Interface
- ERD – Entity Relationship Diagram
- JWT – JSON Web Token
- CRUD – Create, Read, Update, Delete

2. Overall Description

2.1. System Perspective

The system is a client-server program made up of a front-end built with ReactJS, a SpringBoot back-end and a SQL XAMPP database. XAMPP saves user credentials and profiles, while the backend controls authentication logic.

2.2. User Classes and Characteristics

Guest User:

- No account created yet.
- Can register.
- Can login.

Authenticated User:

- Successfully logged in.
- Can visit profile and/or dashboard.
- Can logout..

2.3. Operating Environment

- Front-end: ReactJS.
- Back-end: SpringBoot REST API.
- Database: SQL XAMPP.
- Tools: draw.io.

2.4. Assumptions and Dependencies

- Stable internet connection.
- SQL XAMPP configured.
- Front-end connected to Back-end API.
- Password hashed.

3. System Features and Functional Requirements

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

3.1. Feature 1:

Description: Enables a visitor to register for a new account by supplying legitimate personal data login details.

Functional Requirements:
-A guest user will be able to register on the system with their email address and password.
-The system will verify the necessary input areas.
-After a successful registration, the system will save user data in the database.

3.2. Feature 2:

Description: Enables users who have registered to safely log in and view protected pages.

Functional Conditions:
-The system will use legitimate credentials to authenticate users.
-After a successful login, the system will produce an authentication token.
-The system will prevent unauthenticated users from accessing protected pages. When a user logs out, the system will invalidate the session or token.
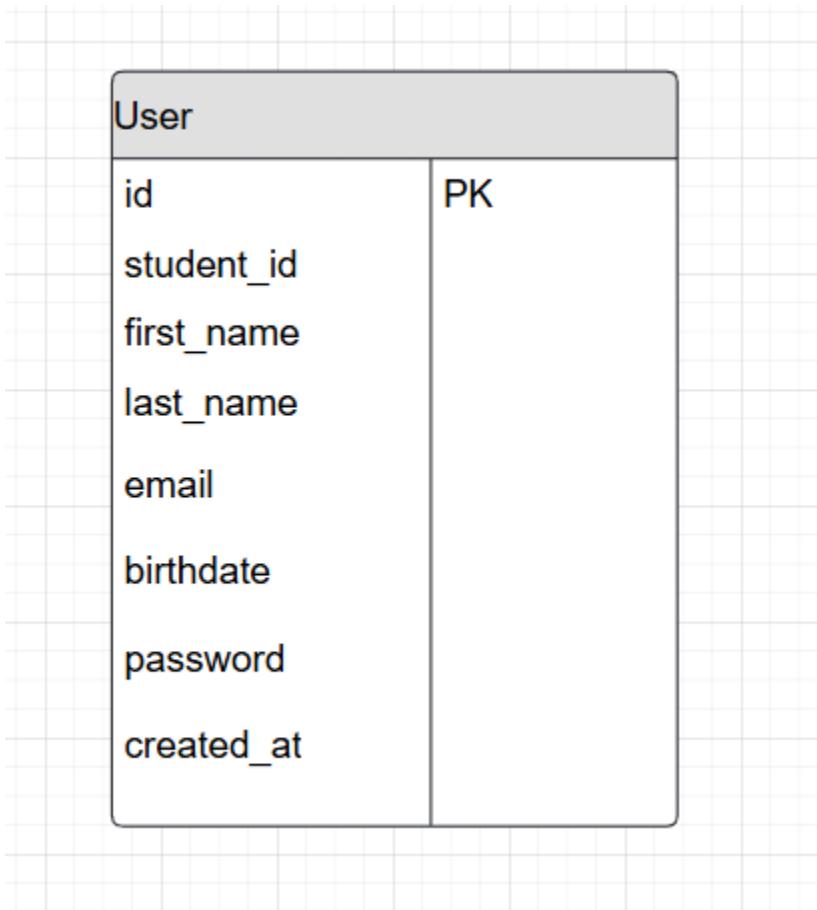
4. Non-Functional Requirements

- Security: Password must be handled securely, and passwords must be encrypted.
- Performance: Responses to login and registration should take place within reasonable time constraints
- Usability: The user interface ought to be easy to use.
- Reliability: Incorrect input and authentication problems should be handled properly.
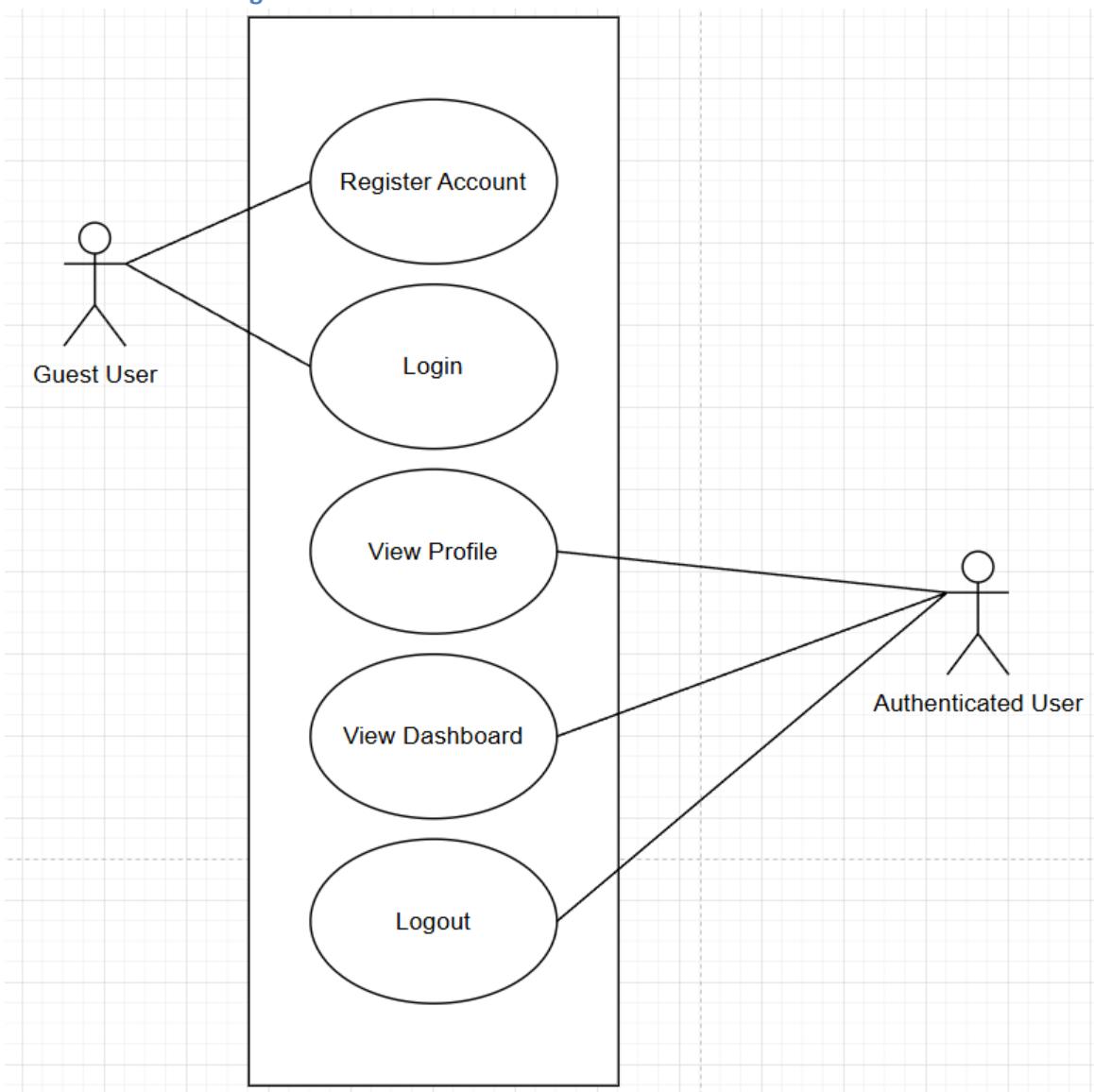
5. System Models (Diagrams)

Insert the necessary diagrams for the system:

5.1. ERD

User	
id	PK
student_id	
first_name	
last_name	
email	
birthdate	
password	
created_at	
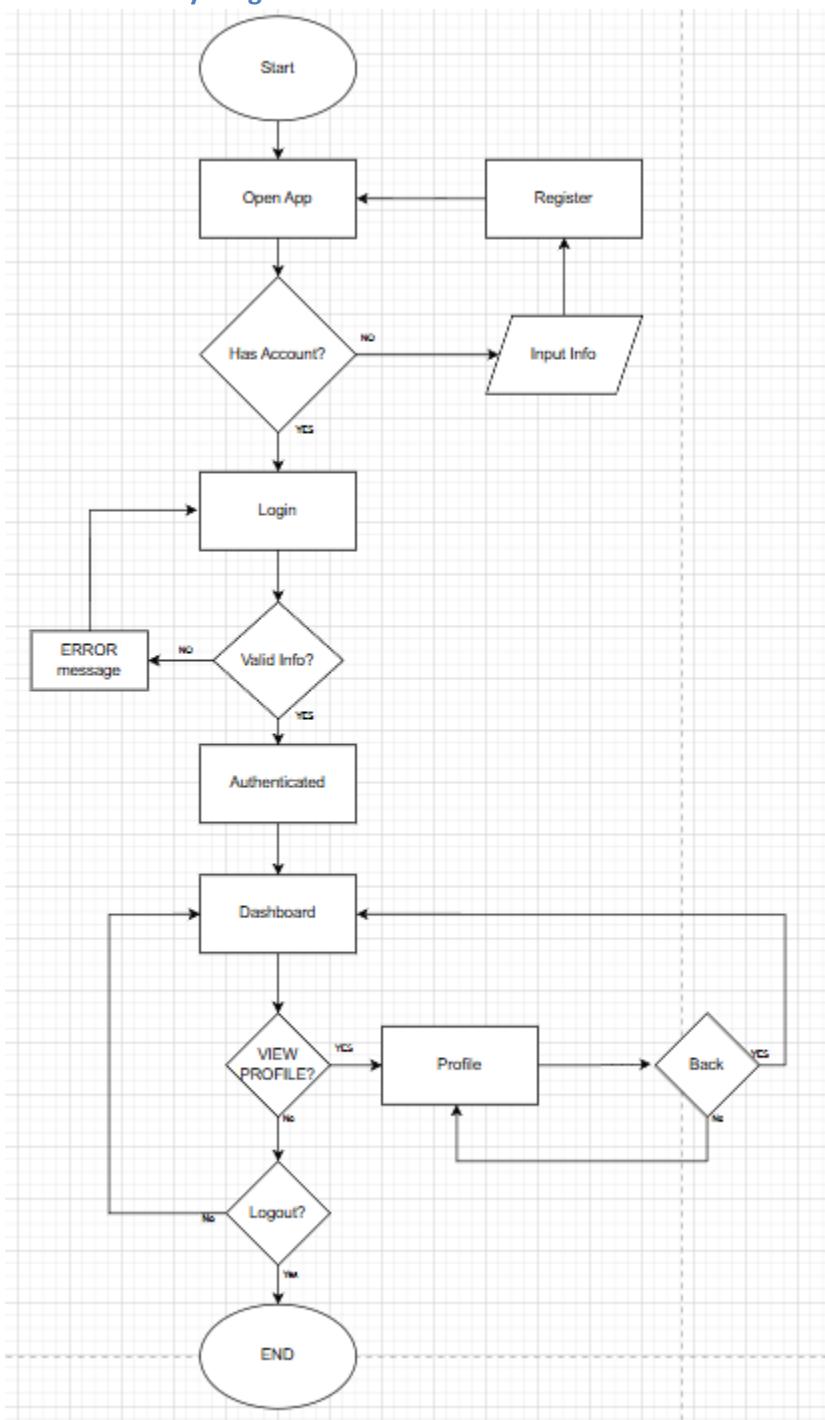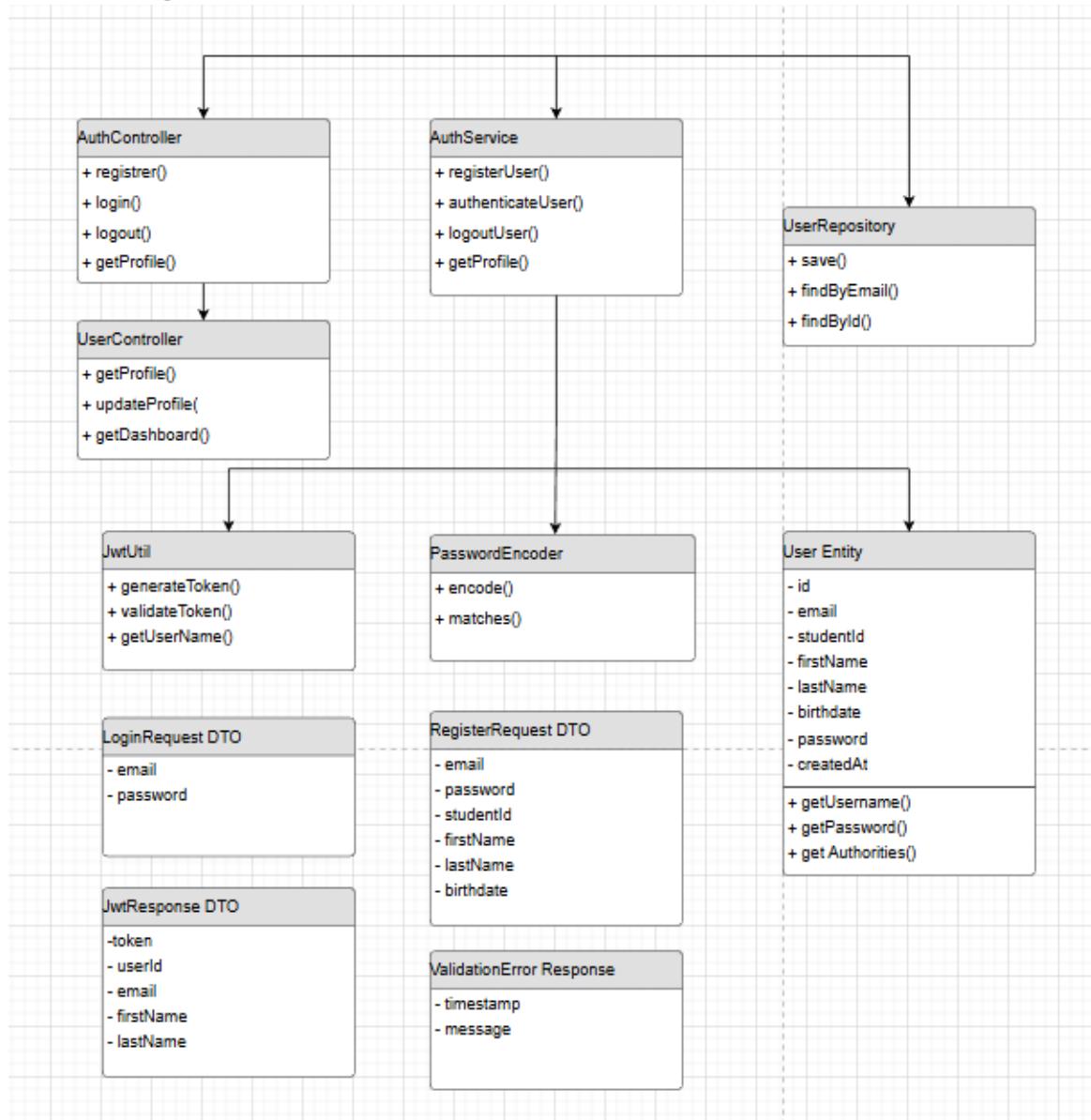
5.2. Use Case Diagram

5.3. Activity Diagram

```
                        ┌─────────┐
                        │  Start  │
                        └────┬────┘
                             │
                             ▼
              ┌──────────┐        ┌──────────┐
              │ Open App │◄───────│ Register │
              └────┬─────┘        └────▲─────┘
                   │                   │
                   ▼                   │
            ◇ Has Account? ◇──NO──► [ Input Info ]
                   │
                  YES
                   │
                   ▼
              ┌─────────┐
              │  Login  │
              └────┬────┘
                   │
                   ▼
   ┌───────┐      ◇ Valid Info? ◇──NO──► ┌───────────┐
   │ ERROR │                             │  message  │  (ERROR message)
                   │
                  YES
                   │
                   ▼
            ┌──────────────┐
            │ Authenticated│
            └──────┬───────┘
                   │
                   ▼
            ┌────────────┐
            │ Dashboard  │◄──────────────────┐
            └─────┬──────┘                   │
                  │                          │
                  ▼                          │
            ◇ VIEW      ◇──YES──►┌─────────┐──►◇ Back ◇──YES──┐
              PROFILE?          │ Profile │       │           │
                  │             └─────────┘      No           │
                 No                                           │
                  │                                           │
                  ▼                                           │
            ◇ Logout? ◇──NO──────────────────────────────────┘
                  │
                 YES
                  │
                  ▼
            ┌─────────┐
            │   END   │
            └─────────┘
```

5.4. Class Diagram

AuthController
- + registrer()
- + login()
- + logout()
- + getProfile()

AuthService
- + registerUser()
- + authenticateUser()
- + logoutUser()
- + getProfile()

UserRepository
- + save()
- + findByEmail()
- + findById()

UserController
- + getProfile()
- + updateProfile(
- + getDashboard()

JwtUtil
- + generateToken()
- + validateToken()
- + getUserName()

PasswordEncoder
- + encode()
- + matches()

User Entity
- - id
- - email
- - studentId
- - firstName
- - lastName
- - birthdate
- - password
- - createdAt
- + getUsername()
- + getPassword()
- + get Authorities()

LoginRequest DTO
- - email
- - password

RegisterRequest DTO
- - email
- - password
- - studentId
- - firstName
- - lastName
- - birthdate

JwtResponse DTO
- -token
- - userId
- - email
- - firstName
- - lastName

ValidationError Response
- - timestamp
- - message

## 5.5. Sequence Diagram

User Register

6. Appendices

Coding session will begin next week during laboratory hours.