CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342-Section
SYSTEMS INTEGRATION AND ARCHITECTURE 1

FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

Project Title: StudentPortal Lite

Prepared By: John Luis C. Leanda

Date of Submission: 2/3/2026

Version: 1.0

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this document is to describe the functional and non-functional requirements of the StudentPortal Lite System.

This system allows users to register, log in, view their profile dashboard, and log out securely.

This document is intended for:

- Developers
- Instructors
- Students involved in designing and implementing the system

The diagrams and descriptions in this document will serve as the basis for the system's implementation during the next development phase.

1.2. Scope

The Student Portal Lite System is a web-based application that provides basic user authentication and profile management features.

The system will:

- Allow users to register an account
- Allow registered users to log in
- Display a user dashboard/profile after login
- Prevent access to protected pages when logged out
- Allow users to log out of the system

The system will not include advanced features such as:

- Password recovery
- Role-based access control
- Data analytics
- Third-party authentication (e.g., Google, Facebook)

1.3. Definitions, Acronyms, and Abbreviations

Term	Definition
User	A person who interacts with the system
Guest User	A user who is not logged in
Authenticated User	A user who has successfully logged in

Authentication	The process of verifying user identity
ERD	Entity Relationship Diagram
UI	User Interface
API	Application Programming Interface
JWT	JSON Web Token

2. Overall Description

2.1. System Perspective

The Student Portal Lite System is a standalone web application consisting of:

- A React-based frontend for user interaction
- A Spring Boot backend API for handling authentication logic
- A relational database for storing user information

The system follows a client-server architecture where the frontend communicates with the backend via HTTP requests.

2.2. User Classes and Characteristics

Guest User

- Has no account or is not logged in
- Can register a new account
- Can log in using valid credentials
- Cannot access protected pages

Authenticated User

- Has a registered account
- Can log in and log out
- Can view their profile/dashboard
- Can access protected pages

2.3. Operating Environment

Hardware

- Desktop or laptop computer
- Internet connection

Software

- Web browser (Google Chrome, Mozilla Firefox, Microsoft Edge)
- React (Frontend)
- Spring Boot (Backend)
- MySQL (Database)

Tools

- draw.io / diagrams.net (for diagrams)
- MS Word / PDF reader (for documentation)

2.4. Assumptions and Dependencies

- Users have a stable internet connection
- The system is accessed through a modern web browser
- The backend server is running and reachable
- The database is available and properly configured
- Passwords are stored in encrypted form
- Authentication uses tokens or sessions

3. System Features and Functional Requirements

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

3.1. Feature 1:

Description:

This feature allows a guest user to create a new account by providing the required personal and login information.

Functional Requirements:

- The system shall allow users to register using a valid email and password
- The system shall validate input fields before account creation
- The system shall store user information securely in the database

3.2. Feature 2:

Description:

This feature allows registered users to authenticate and access protected areas of the system.

Functional Requirements:

- The system shall authenticate users using email and password
- The system shall prevent login with invalid credentials
- The system shall grant access to the dashboard upon successful login

3.3. Feature 3:

Description:

This feature allows authenticated users to view their personal profile information.

Functional Requirements:

- The system shall display the user’s profile information
- The system shall restrict access to authenticated users only
- The system shall redirect unauthenticated users to the login page

3.4. Feature 3:

Description

This feature allows authenticated users to log out of the system securely.

Functional Requirements

- The system shall terminate the user session or invalidate the token
- The system shall redirect the user to the login page after logout
- The system shall prevent access to protected pages after logout

4. Non-Functional Requirements

Performance

- The system shall respond to user requests within a reasonable time
- Login and registration should complete within a few seconds

Security

- Passwords shall be stored in encrypted form
- Unauthorized users shall not access protected pages
- User sessions or tokens shall be securely managed

Usability

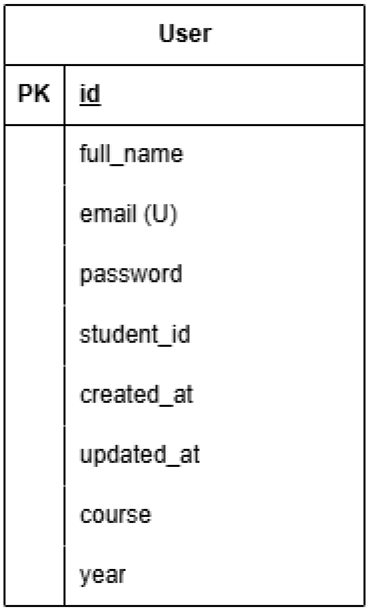
- The user interface shall be simple and easy to navigate
- Error messages shall be clear and user-friendly

Reliability

- The system shall be available during normal operating hours
- The system shall handle invalid inputs without crashing

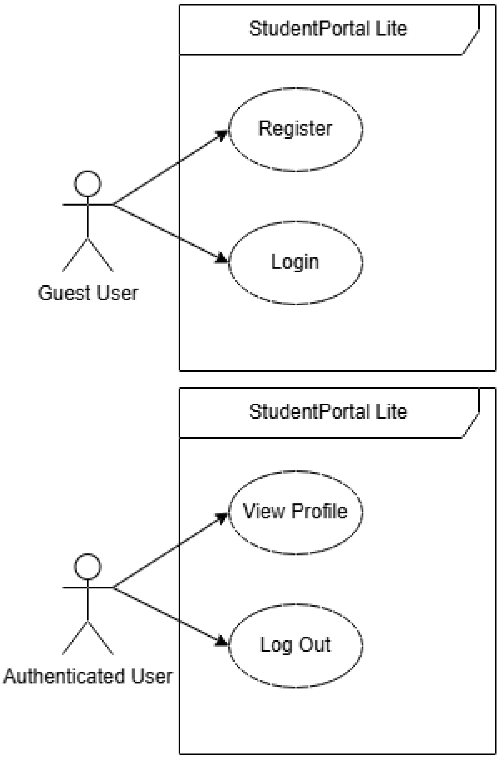
5. System Models (Diagrams)

5.1. ERD

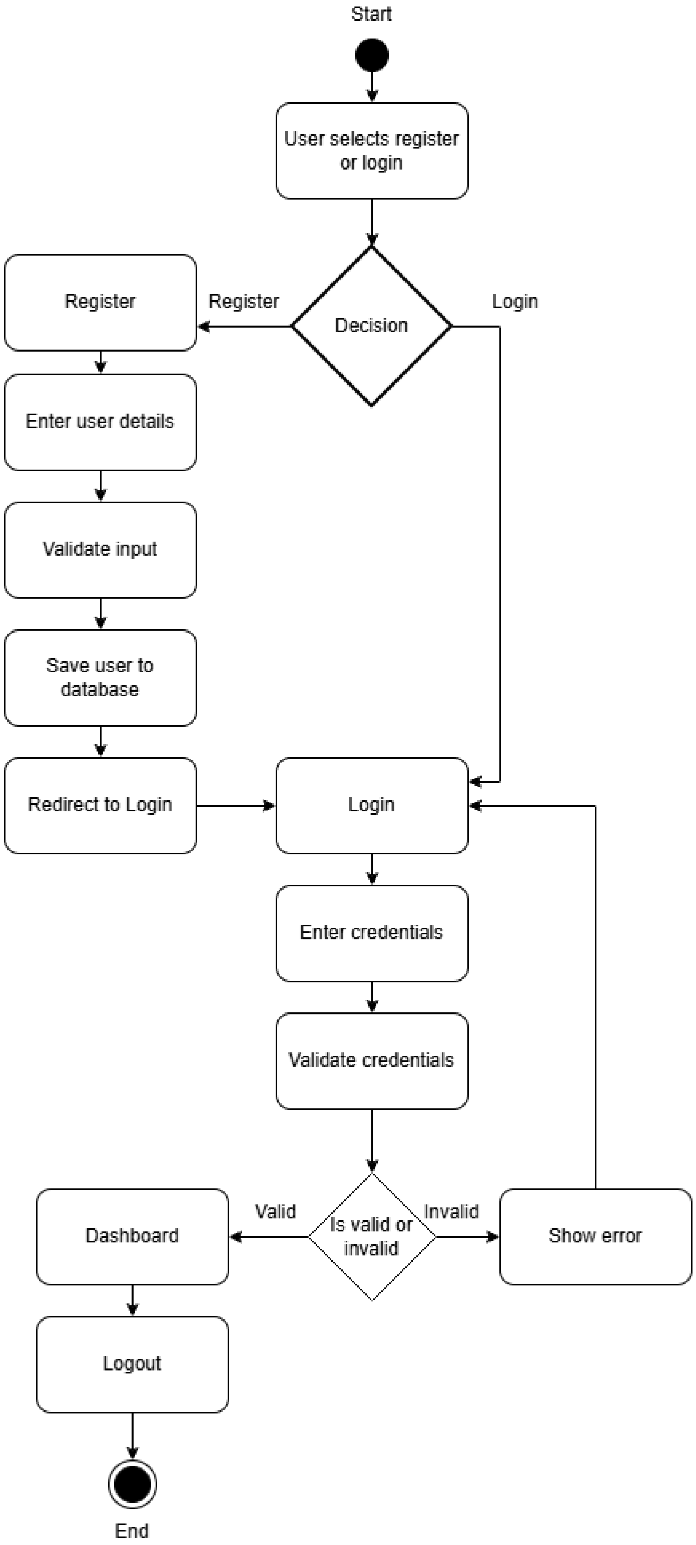


User	
PK	id
	full_name
	email (U)
	password
	student_id
	created_at
	updated_at
	course
	year

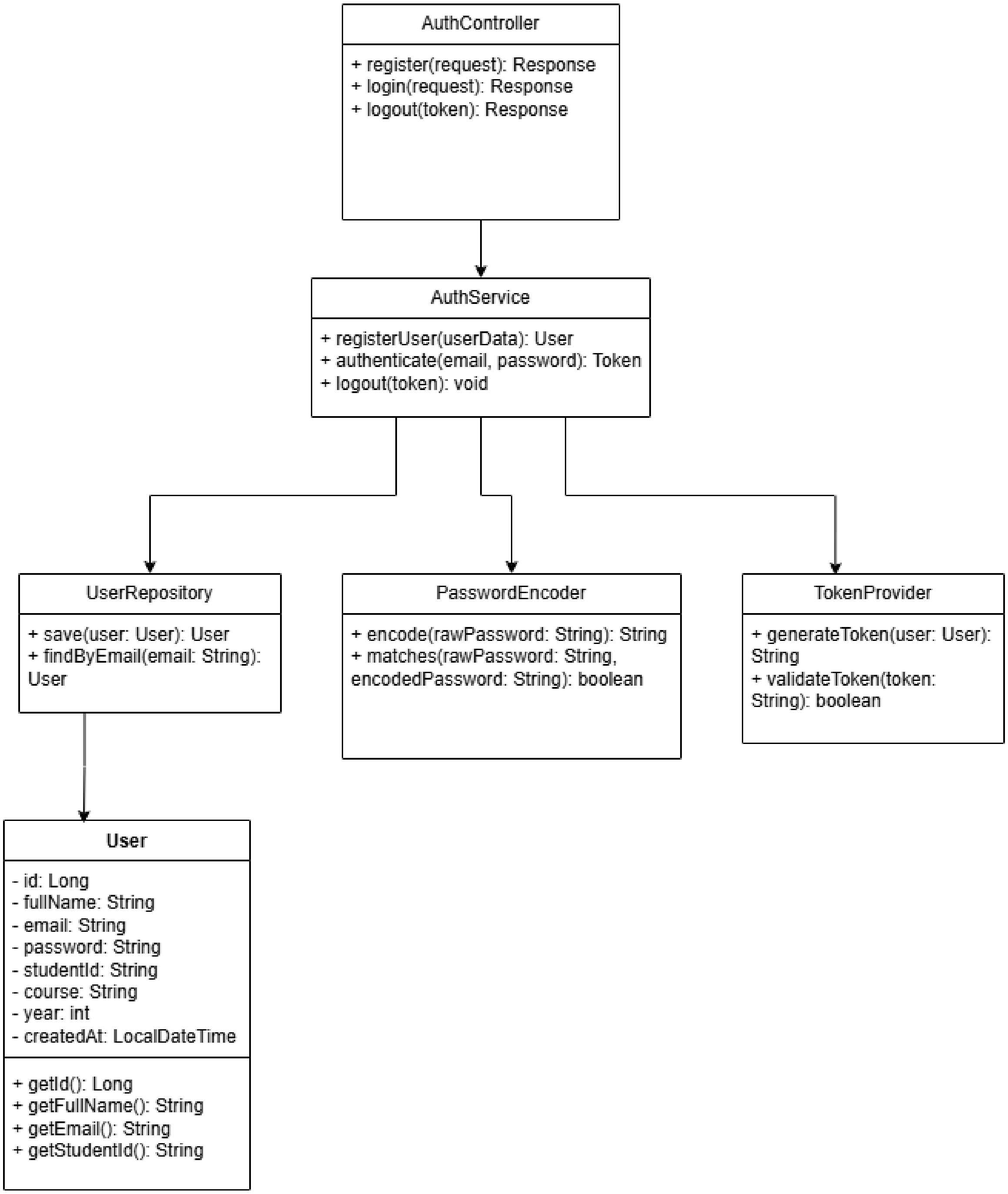
5.2. Use Case Diagram



5.3. Activity Diagram



5.4. Class Diagram



AuthController

+ register(request): Response
+ login(request): Response
+ logout(token): Response

AuthService

+ registerUser(userData): User
+ authenticate(email, password): Token
+ logout(token): void

UserRepository

+ save(user: User): User
+ findByEmail(email: String): User

PasswordEncoder

+ encode(rawPassword: String): String
+ matches(rawPassword: String, encodedPassword: String): boolean

TokenProvider

+ generateToken(user: User): String
+ validateToken(token: String): boolean

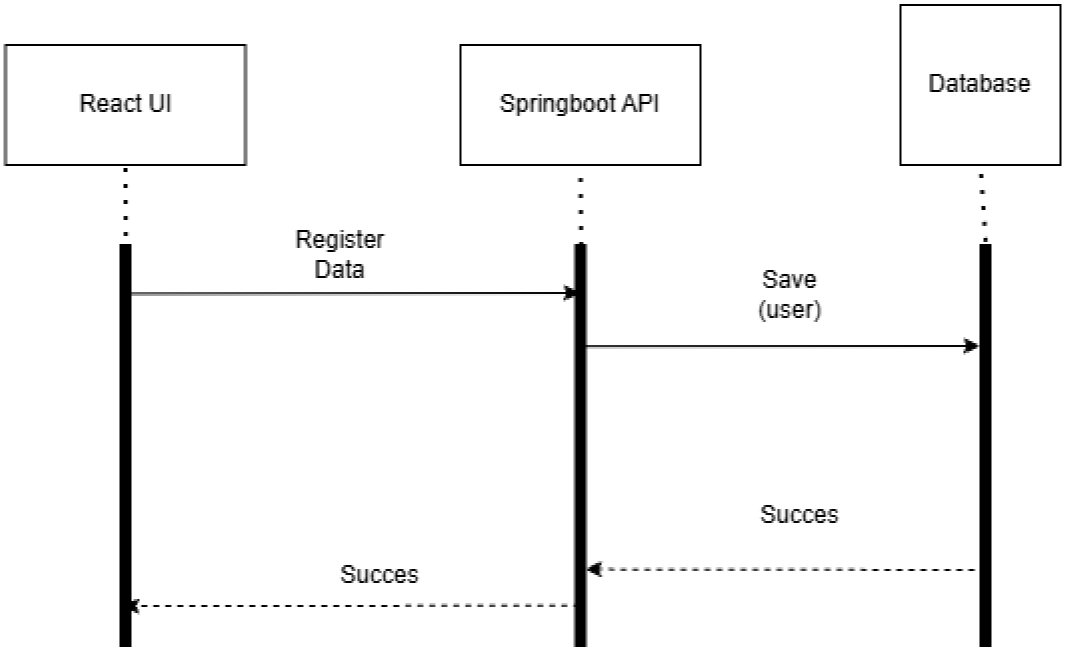
User

- id: Long
- fullName: String
- email: String
- password: String
- studentId: String
- course: String
- year: int
- createdAt: LocalDateTime

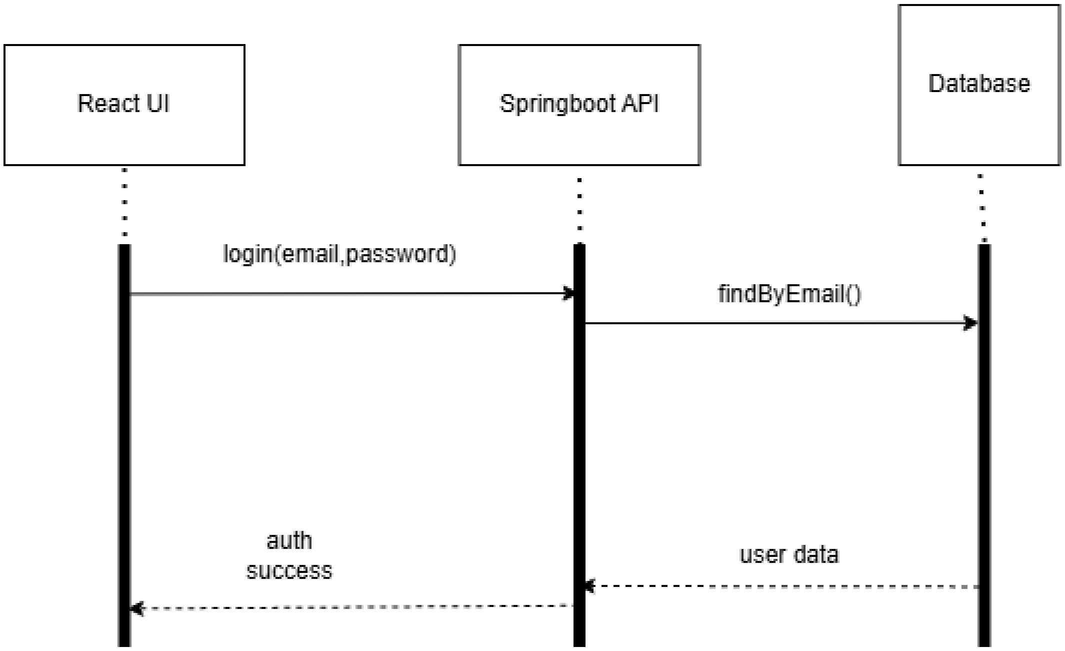
+ getId(): Long
+ getFullName(): String
+ getEmail(): String
+ getStudentId(): String

5.5. Sequence Diagram

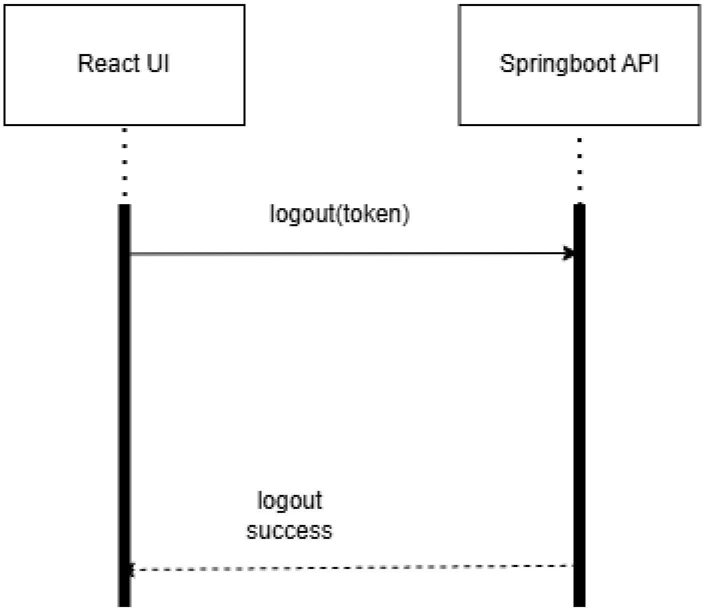
Register



Login



Logout


React UI
Springboot API
logout(token)
logout
success