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: Salgado, Craig Zachary

Date of Submission: 2/2/2026

Version:

Table of Contents

1. Introduction 3
   1.1. Purpose 3
   1.2. Scope 3
   1.3. Definitions, Acronyms, and Abbreviations 3
2. Overall Description 3
   2.1. System Perspective 3
   2.2. User Classes and Characteristics 3
   2.3. Operating Environment 3
   2.4. Assumptions and Dependencies 3
3. System Features and Functional Requirements 3
   3.1. Feature 1: 3
   3.2. Feature 2: 3
4. Non-Functional Requirements 3
5. System Models (Diagrams) 4
   5.1. ERD 4
   5.2. Use Case Diagram 4
   5.3. Activity Diagram 4
   5.4. Class Diagram 4
   5.5. Sequence Diagram 4
6. Appendices 4

1. Introduction

1.1. Purpose

The system enables secure user registration, login, and session management for web applications built with React frontend and Spring Boot backend.

1.2. Scope

What the System Will Do

- Allow new users to create accounts with email and password
- Authenticate existing users through a secure login process
- Generate and manage JWT (JSON Web Tokens) for session handling
- Provide access to protected resources (user profile/dashboard) for authenticated users
- Enable users to securely log out and invalidate their sessions
- Prevent unauthorized access to protected pages
- Hash passwords using BCrypt before storage
- Support role-based access control (USER/ADMIN roles)

System Boundaries

In Scope:

- User registration with validation
- Login/logout functionality
- JWT-based authentication
- Protected route access control
- Basic user profile viewing

Out of Scope:

- Password reset/recovery functionality
- Email verification

- Two-factor authentication (2FA)
- Social media login integration (OAuth)
- User profile editing capabilities
- Account Deletion

1.3. Definitions, Acronyms, and Abbreviations

JWT - JSON Web Token - A compact, URL-safe means of representing claims to be transferred between two parties for authentication

Bcrypt - A password hashing function designed to be computationally expensive to resist brute-force attacks

API - Application Programming Interface - A set of endpoints that allow the frontend to communicate with the backend

2. Overall Description

2.1. System Perspective

The User Authentication System operates as a foundational security layer within a modern web application architecture. It follows a three-tier architecture pattern:

- **Presentation Layer:** React-based single-page application providing the user interface
- **Application Layer:** Spring Boot RESTful API handling business logic and authentication processes
- **Data Layer:** MySQL database storing user credentials and related information

The system integrates with the larger application ecosystem by providing authentication services that protect other application features. All protected resources in the application rely on this system to verify user identity before granting access.

2.2. User Classes and Characteristics

Guest Users

- **Characteristics:** Unauthenticated visitors to the application
- **Technical Expertise:** Basic computer literacy; no technical knowledge required
- **Primary Activities:** Create new accounts, log in to existing accounts
- **Access Level:** Limited to public pages and authentication forms

Registered Users (USER role)

- **Characteristics:** Authenticated users with standard privileges
- **Technical Expertise:** Basic to intermediate computer skills
- **Primary Activities:** Access personal dashboard, view profile, use application features
- **Access Level:** Full access to standard user features and protected resources

2.3. Operating Environment

Client-Side Requirements

- **Browser:** Modern web browsers (Chrome 90+, Firefox 88+, Safari 14+, Edge 90+)
- **JavaScript:** Enabled (required for React application)
- **Cookies:** Enabled for session management
- **Network:** Stable internet connection

Server-Side Requirements

- **Runtime:** Java 17 or higher
- **Framework:** Spring Boot 3.x
- **Database:** MySQL 8.0 or higher
- **Operating System:** Linux (Ubuntu 20.04+), Windows Server 2019+, or macOS 11+

Development Tools

- **Frontend:** Node.js 18+, npm/yarn, React 18+
- **Backend:** Maven 3.8+, Spring Boot CLI
- **Database:** MySQL Workbench or similar database management tool

2.4. Assumptions and Dependencies

Assumptions

- Users have valid email addresses for registration
- The application will be deployed over HTTPS in production
- Database connection is reliable and available
- Users will not share credentials with others
- Server infrastructure can handle expected concurrent user load

External Dependencies

- **Spring Security:** For authentication and authorization framework
- **jjwt (Java JWT):** For JWT token generation and validation
- **BCrypt:** For password hashing (included in Spring Security)
- **React Router:** For client-side routing and protected route implementation
- **Axios:** For HTTP requests from React to Spring Boot API

MySQL JDBC Driver: For database connectivity

3. System Features and Functional Requirements

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

3.1. Feature 1: User Registration

Description: Allows new users to create an account by providing required information. The system validates input, ensures email uniqueness, securely hashes passwords, and stores user data in the database.

Functional Requirements:

- The system shall provide a registration form with fields for email, password, first name, and last name.
- The system shall validate that the email field contains a properly formatted email address.
- The system shall require passwords to be at least 8 characters long.
- The system shall check if the email address is already registered and display an error if it exists.
- The system shall hash passwords using BCrypt before storing them in the database.
- The system shall assign the default role of USER to newly registered accounts.
- The system shall set is_active to TRUE for new accounts by default.
- The system shall automatically record created_at and updated_at timestamps.
- The system shall return a success message with HTTP 201 status upon successful registration.
- The system shall display appropriate error messages for validation failures.

3.2. Feature 2: User Login

Description: Authenticates users by verifying their credentials against stored data. Upon successful authentication, the system generates a JWT token that enables access to protected resources.

Functional Requirements:

- The system shall provide a login form with fields for email and password.
- The system shall verify that a user account exists for the provided email address.
- The system shall compare the provided password against the stored BCrypt hash.

- The system shall return HTTP 401 Unauthorized if the email does not exist.
- The system shall return HTTP 401 Unauthorized if the password does not match.
- The system shall generate a JWT access token containing user ID, email, and role upon successful authentication.
- The system shall set JWT expiration time (e.g., 24 hours from generation).
- The system shall return the JWT token in an httpOnly cookie with the response.
- The system shall return HTTP 200 OK with user information upon successful login.
- The system shall redirect authenticated users to the dashboard/profile page.

3.3. Feature 3:

Description: Enforces authentication requirements on protected pages and resources. Only users with valid JWT tokens can access the user profile, dashboard, and other protected features.

Functional Requirements:

- The system shall implement a security filter to intercept requests to protected endpoints.
- The system shall extract and validate the JWT token from the request cookie.
- The system shall verify JWT signature using the secret key.
- The system shall check JWT expiration time and reject expired tokens.
- The system shall return HTTP 401 Unauthorized if no token is present.
- The system shall return HTTP 401 Unauthorized if the token is invalid or expired.
- The system shall extract user information from valid tokens and make it available to the application.
- The system shall grant access to protected resources if the token is valid.
- The system shall redirect unauthenticated users attempting to access protected pages to the login page.
- The system shall display user-specific data (name, email) on the profile/dashboard page.

3.4. Feature 4: User Logout

Description: Allows authenticated users to securely end their session. The system invalidates the current token and clears client-side authentication state.

Functional Requirements:

- The system shall provide a logout button/link on all authenticated pages.
- The system shall extract the JWT token from the logout request.
- The system shall add the token to a blacklist/invalidation mechanism.
- The system shall clear the httpOnly cookie containing the JWT.
- The system shall clear all client-side authentication state (localStorage, sessionStorage).
- The system shall return HTTP 200 OK upon successful logout.
- The system shall redirect users to the login page after logout.
- The system shall handle logout gracefully even if the token is already invalid.
- The system shall prevent reuse of invalidated tokens.
- The system shall display a confirmation message after successful logout.

4. Non-Functional Requirements

4.1 Security

- The system shall use BCrypt with a minimum cost factor of 10 for password hashing.
- The system shall never store passwords in plain text.
- The system shall use HTTPS for all communications in production.
- The system shall set httpOnly and secure flags on authentication cookies.
- The system shall use a cryptographically secure secret key for JWT signing.
- The system shall implement CORS policies to prevent unauthorized cross-origin requests.
- The system shall sanitize all user inputs to prevent SQL injection and XSS attacks.
- The system shall implement rate limiting on authentication endpoints to prevent brute force attacks.

4.2 Performance

- The system shall complete login authentication within 2 seconds under normal load.

- The system shall complete user registration within 3 seconds under normal load.
- The system shall validate JWT tokens within 100 milliseconds.
- The system shall support at least 100 concurrent users without performance degradation.
- The system shall cache user session data to minimize database queries.

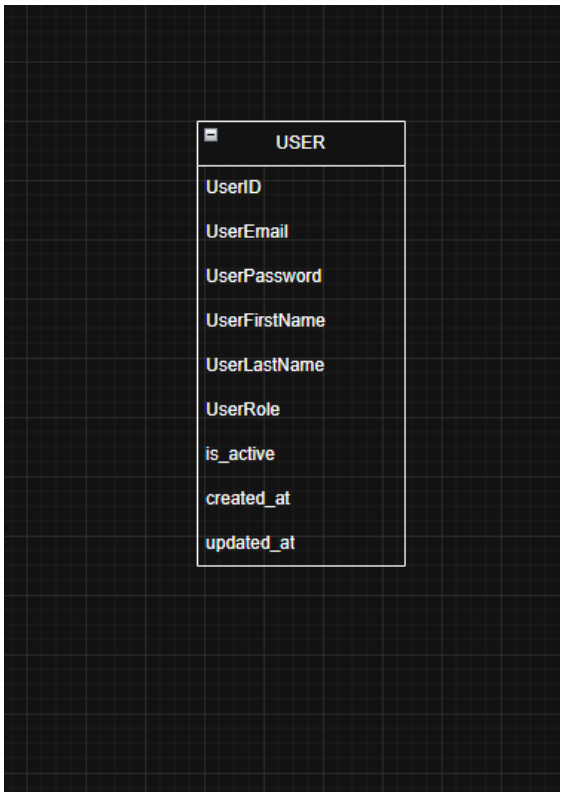
4.3 Usability

- The system shall provide clear, user-friendly error messages for all validation failures.
- The system shall display password strength indicators during registration.
- The system shall use responsive design that works on desktop, tablet, and mobile devices.
- The system shall require no more than 3 steps to complete registration or login.
- The system shall provide visual feedback for all user actions (loading spinners, success messages).

5. System Models (Diagrams)

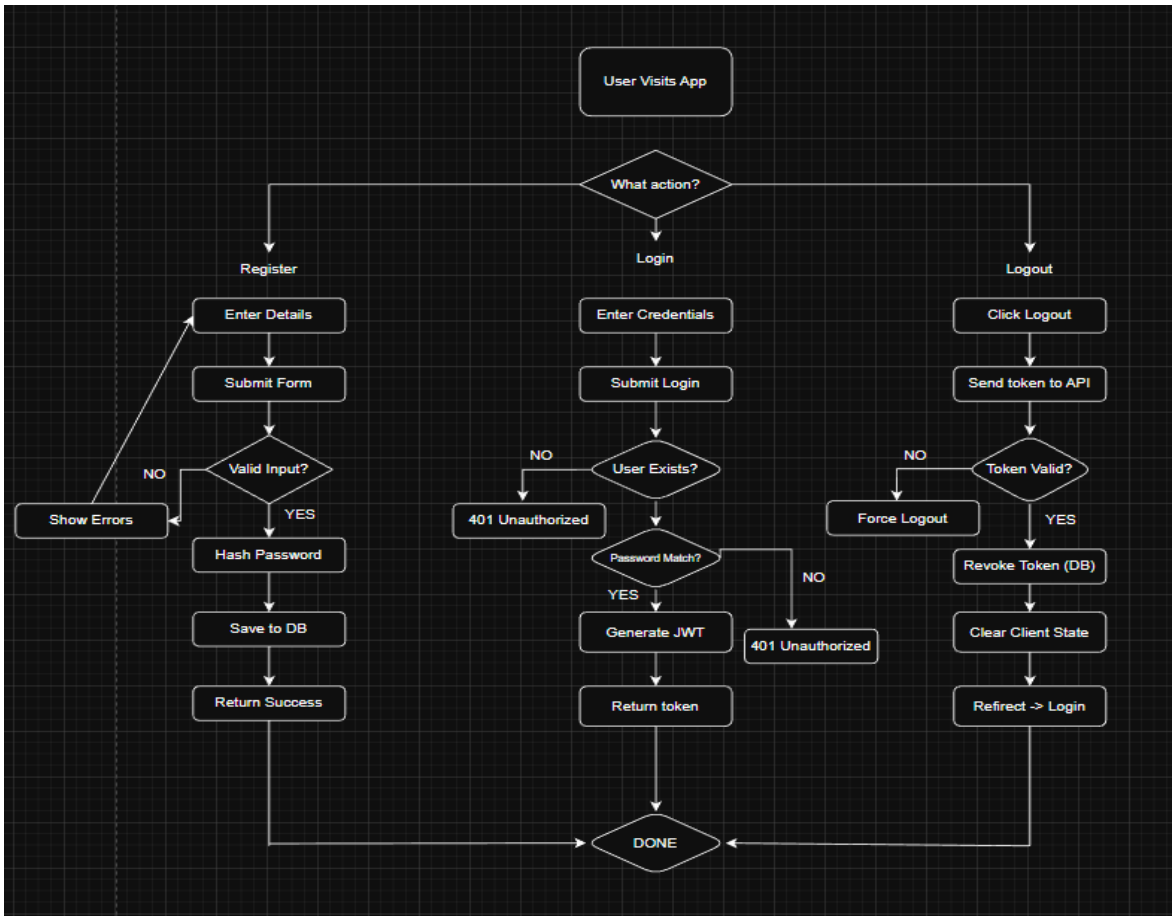
Insert the necessary diagrams for the system:

5.1. ERD

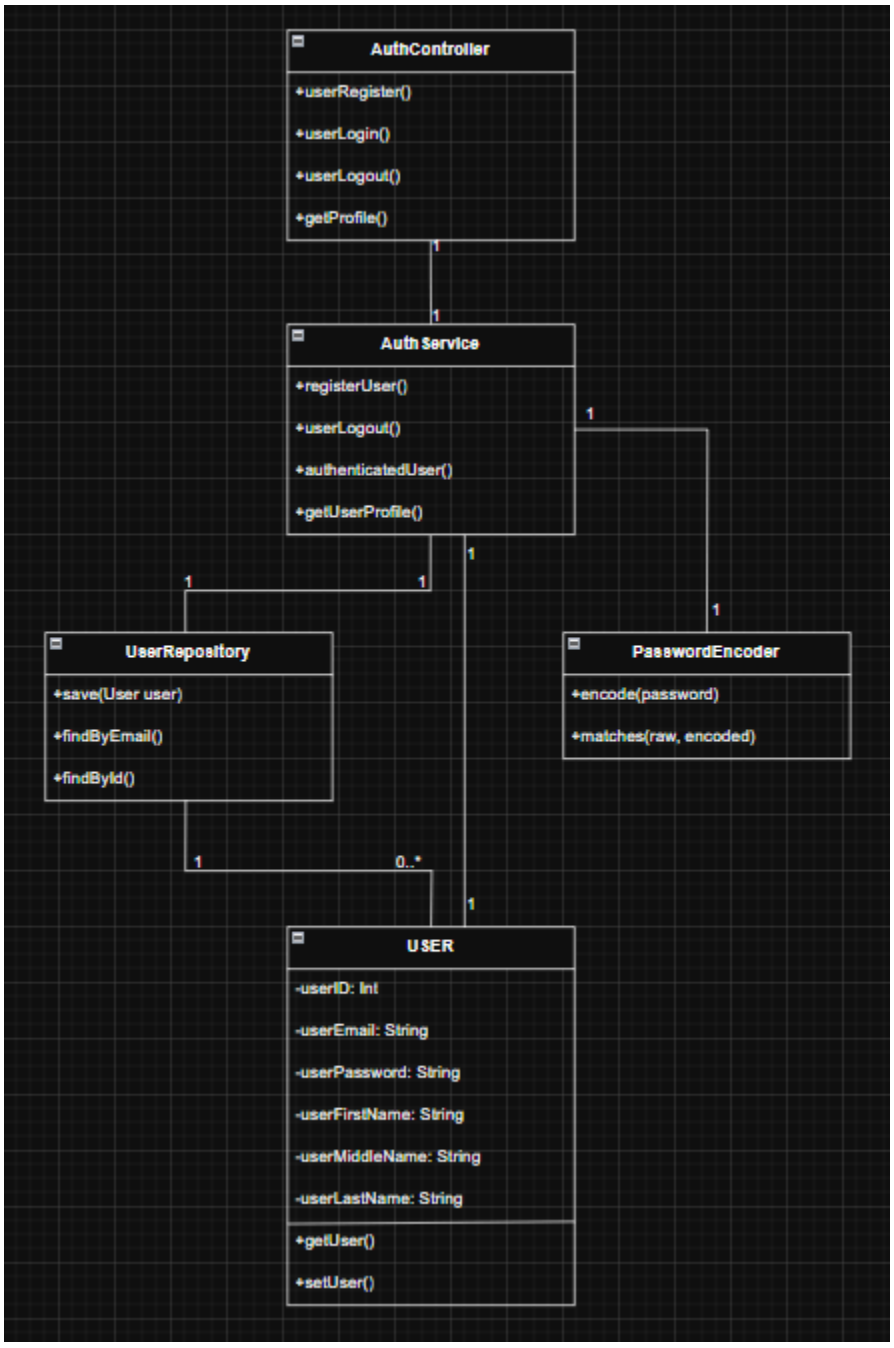


5.2. Use Case Diagram

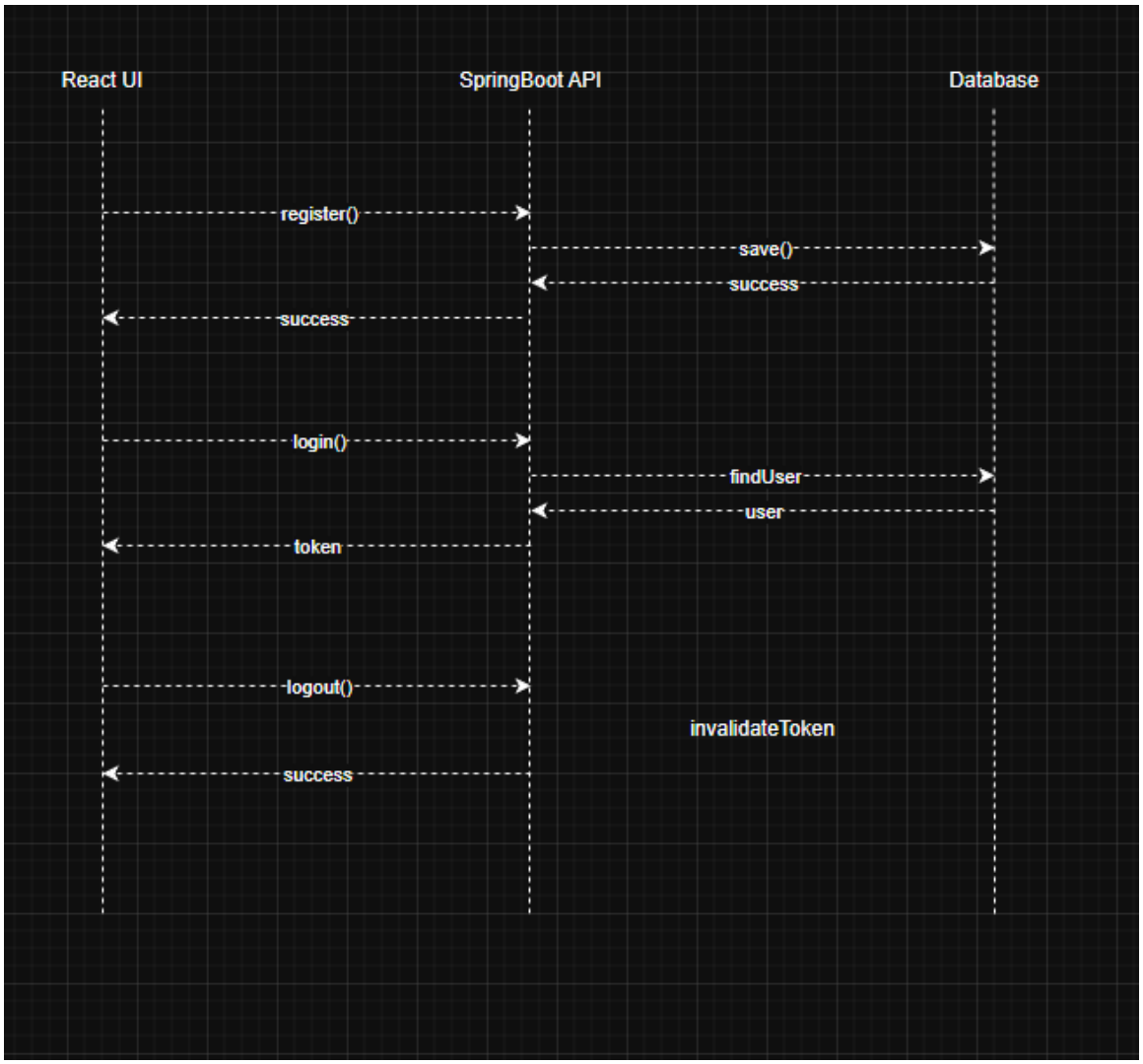
5.3. Activity Diagram



5.4. Class Diagram

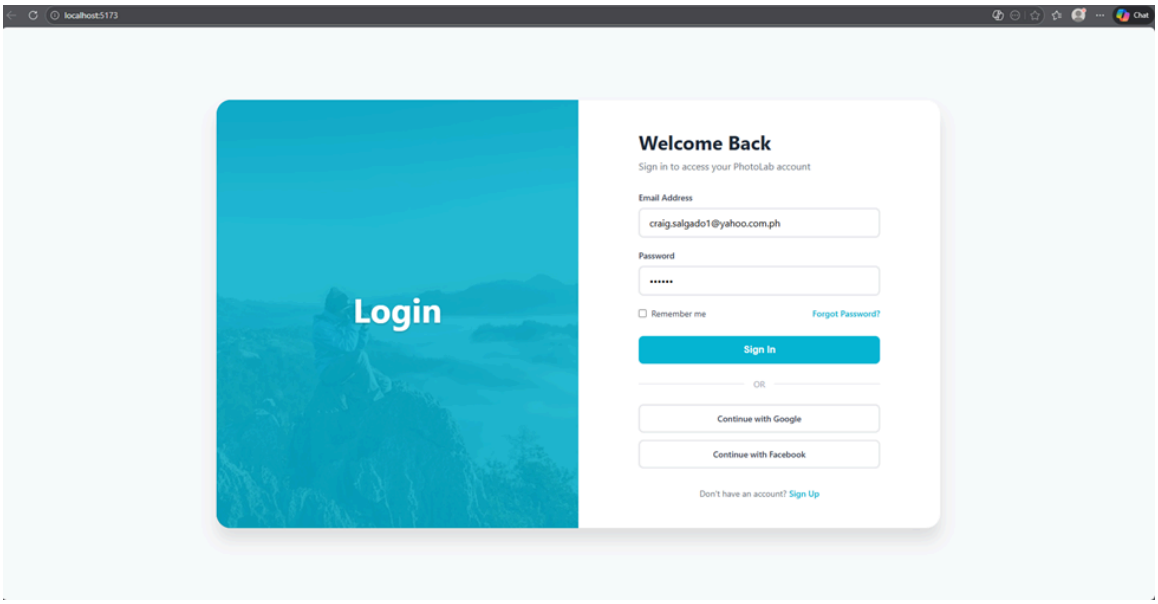


5.5. Sequence Diagram



6. Appendices

LOGIN PAGE



REGISTER