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: 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 outline the functional and non-functional requirements for the **Mini-App - User Registration & Authentication**. This system serves as a foundational authentication platform intended for developers or students to demonstrate secure user session management.

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

1.2. Scope

The system is a web and mobile based application focused exclusively on **User Authentication** and **Session Control**. It allows users to create accounts, authenticate their identity, and access a protected dashboard. It does not include external integrations or complex data processing beyond profile visualization.

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
SRS	System Requirements Specification
Auth	Short for Authentication; the process of verifying a user's identity
Session	The period during which a user is logged in and recognized by the system
Protected Route	A page or view that requires a valid session token to access

2. Overall Description

2.1. System Perspective

The Mini-App acts as a standalone portal. It follows a classic Client-Server architecture where the frontend handles user input and the backend manages the logic for credential verification and session persistence.

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

3.1. Feature 1: User Registration

Description:

Allows a guest to create a unique identity within the system using a name, email, and password.

Functional Requirements:

- **FR1.1:** The system shall provide a registration form requiring Full Name, Email, and Password.
- **FR1.3:** The system shall encrypt passwords using a one-way hashing algorithm before database storage.

3.2. Feature 2: User Authentication (Login)

Description:

Validates user identity to initiate a secure session.

Functional Requirements:

- **FR2.1:** The system shall verify credentials against the stored database records.
- **FR2.2:** The system shall issue a session token (JWT) upon successful authentication.
- **FR2.3:** The system shall display an error message for incorrect email or password combinations.

3.3. Feature 3: Protected Dashboard & Profile

Description:

A restricted area displaying user-specific data.

Functional Requirements:

- **FR3.1:** The system shall display the user's name on the Dashboard.
- **FR3.2:** The system shall verify the presence of a valid session token before rendering this page.

3.4. Feature 3: Secure Logout

Description

Terminates the active session.

Functional Requirements

- **FR4.1:** The system shall invalidate the session token on the server/client side.
- **FR4.2:** The system shall redirect the user to the Login page immediately 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

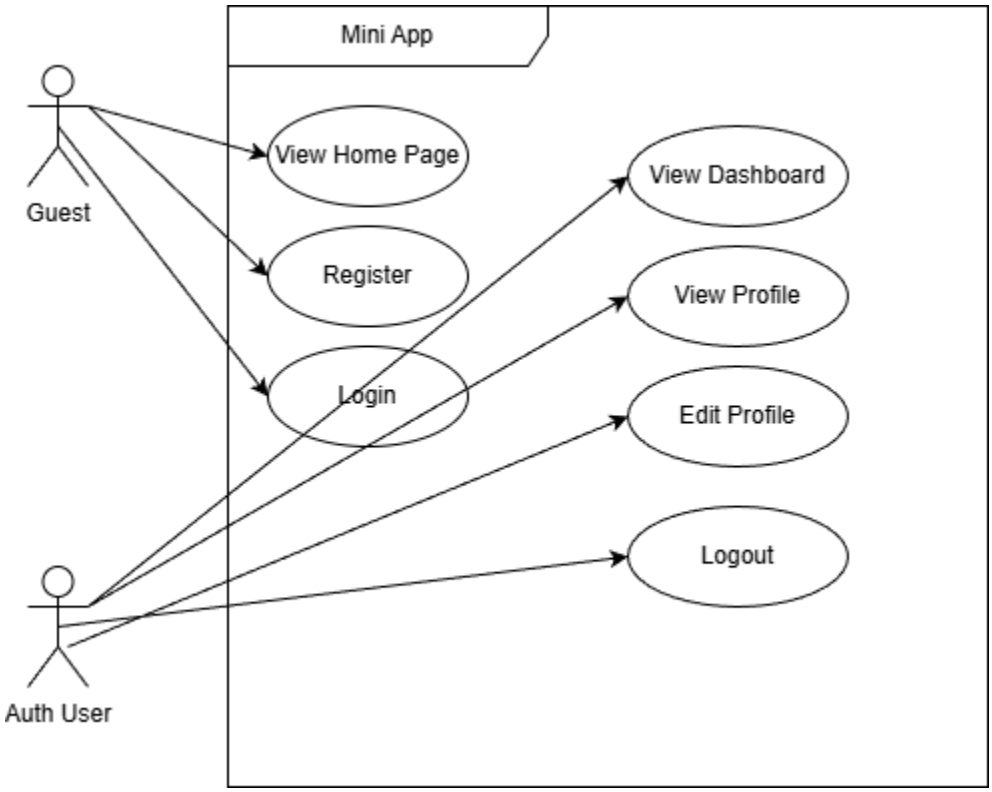
- The system shall be available during normal operating hours
- The system shall handle invalid inputs without crashing

5. System Models (Diagrams)

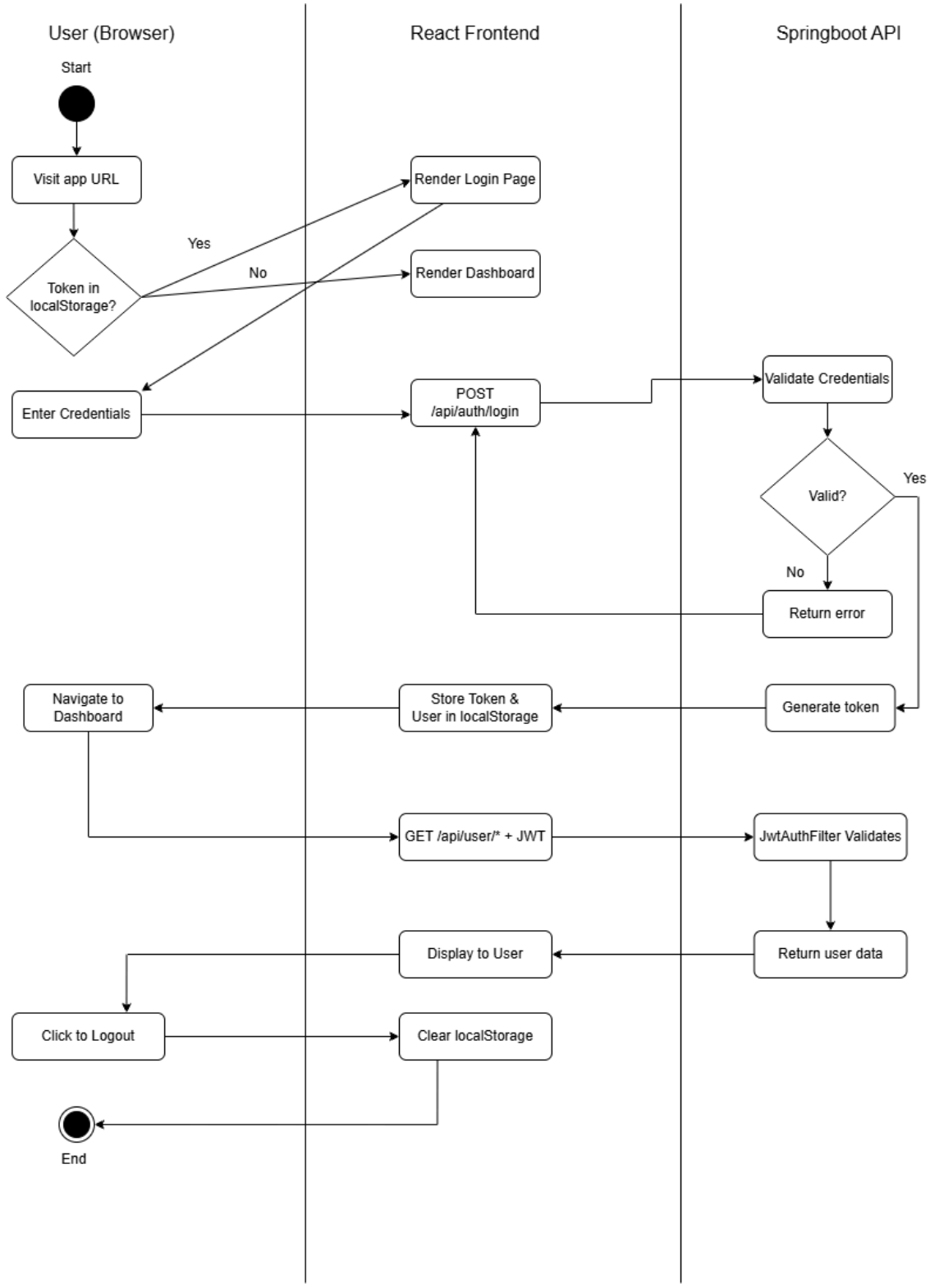
5.1. ERD

User	
PK	id : bigInt(20)
	fullName : varchar(255) email : varchar(255) password : varchar(255)

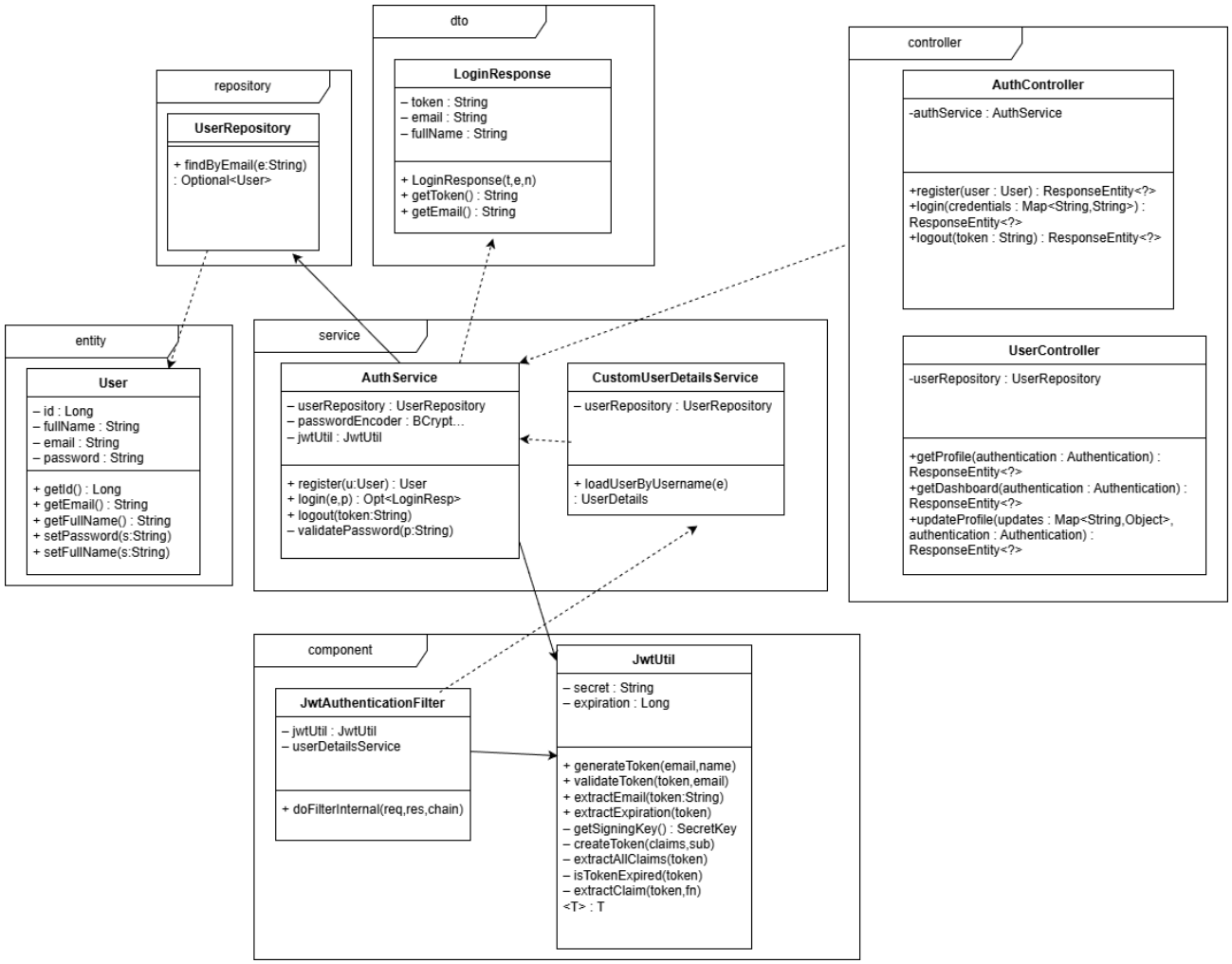
5.2. Use Case Diagram



5.3. Activity Diagram

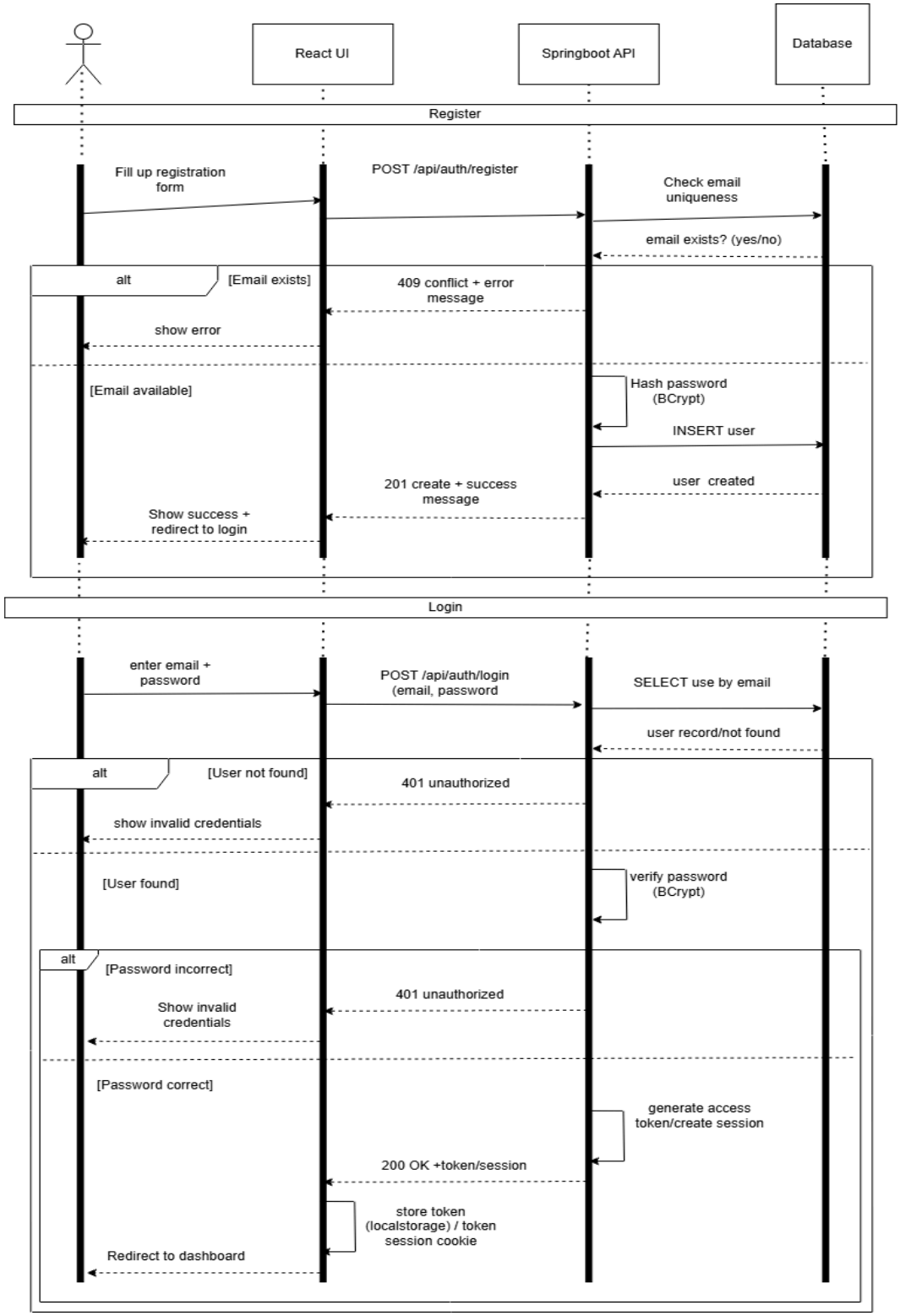


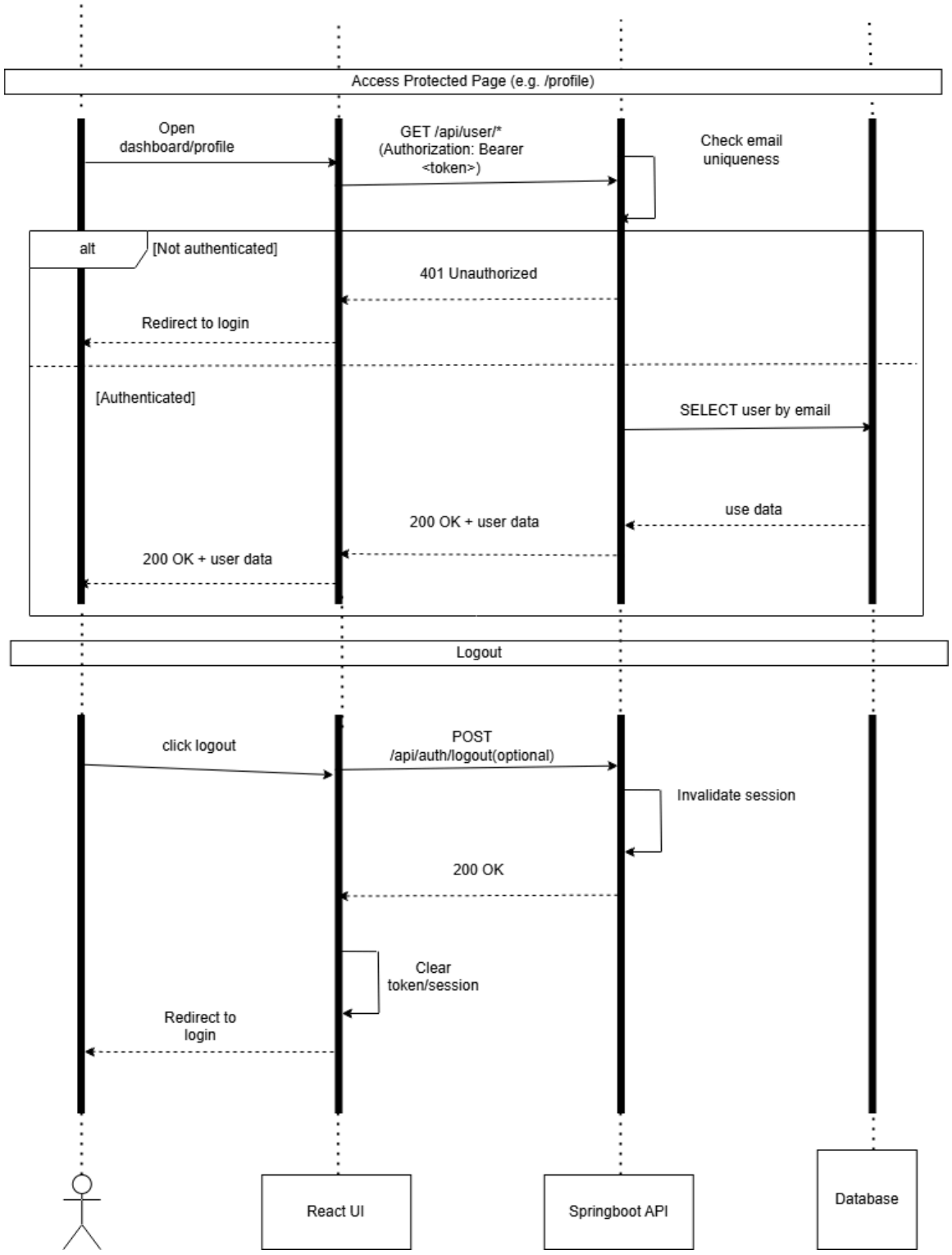
5.4. Class Diagram



5.5. Sequence Diagram

Register, Login, ProtectedRoute, and Logout

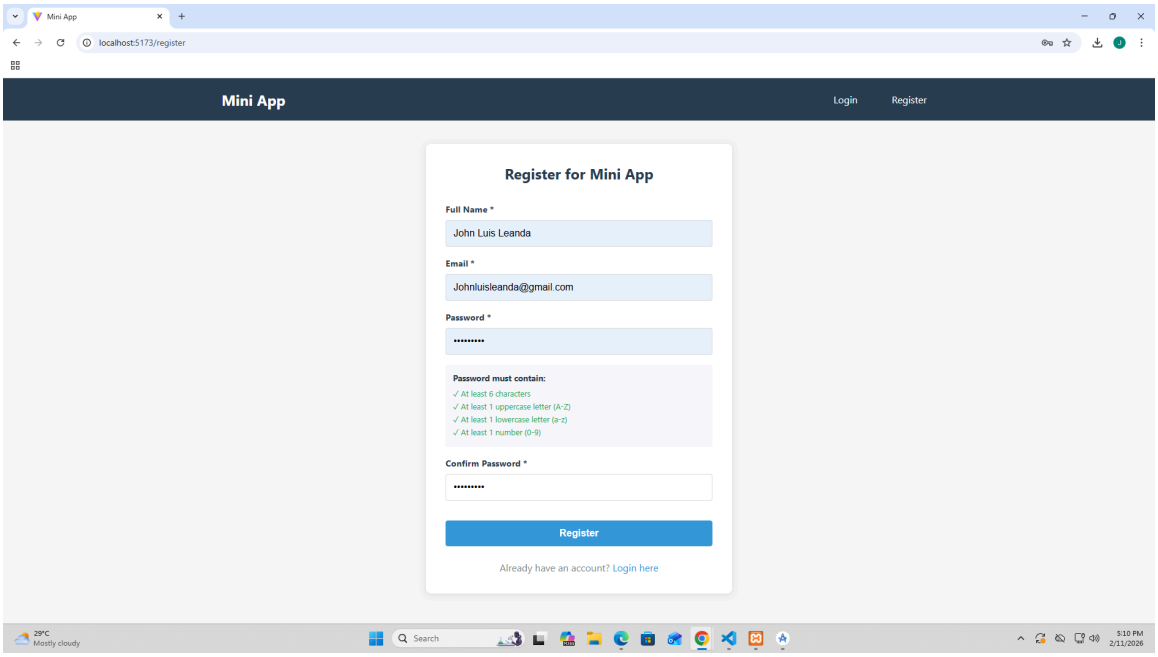



Access Protected Page (e.g. /profile)
Open dashboard/profile
GET /api/user/* (Authorization: Bearer <token>)
Check email uniqueness
alt
[Not authenticated]
401 Unauthorized
Redirect to login
[Authenticated]
SELECT user by email
use data
200 OK + user data
200 OK + user data
Logout
click logout
POST /api/auth/logout(optional)
Invalidate session
200 OK
Clear token/session
Redirect to login
React UI
Springboot API
Database

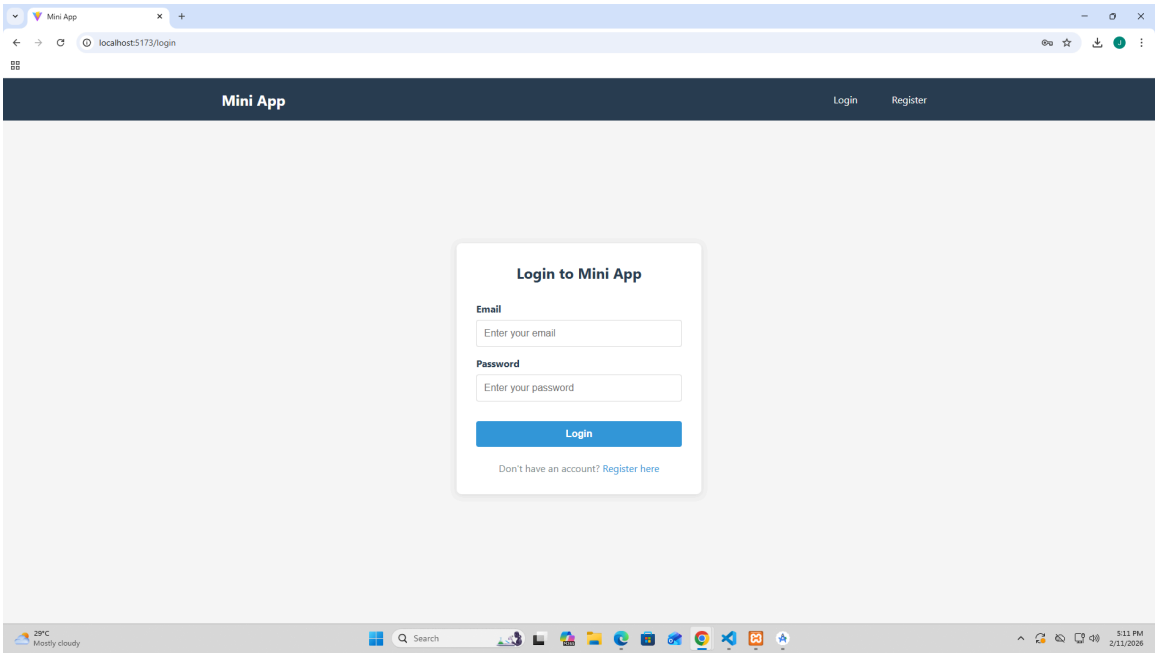
6. Appendices

Web Screenshots:

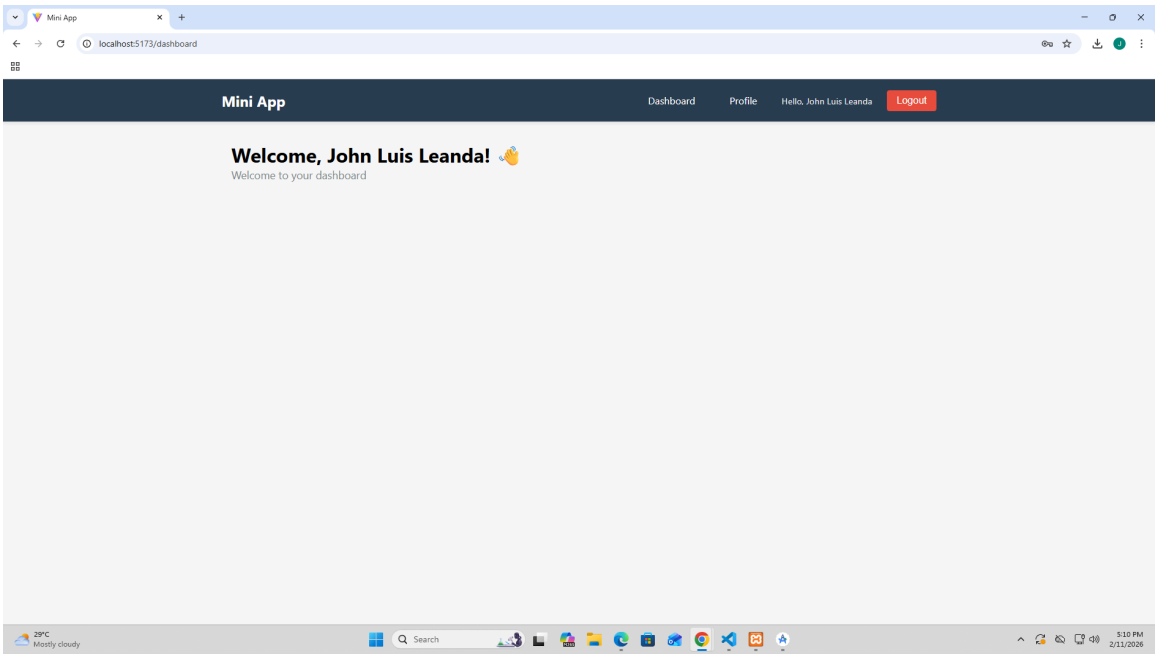
1. Register



2. Login



3. Dashboard



4. Logout Result (Redirected to Login)