CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342-Section SYSTEMS INTEGRATION AND ARCHITECTURE 1

FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

Project Title: User Registration and Authentication Prepared By: Reyes,

Christian Andrey V.

Date of Submission: February 3, 2026

Version: 1.0

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this document is to define the system architecture and functional flow for a standard **Authentication and Authorization Module**. This serves as a blueprint for the development of secure user registration, session management (login/logout), and protected route handling. The intended audience includes the development team and project stakeholders.

1.2. Scope

The system provides a secure gateway for users to create accounts and access a personalized dashboard.

- **Included:** Database persistence for user records, password encryption, JWT (JSON Web Token) generation, and frontend route guarding.
- **Excluded:** Password recovery (forgot password), OAuth2 (Social Login), and multi-factor authentication.

1.3. Definitions, Acronyms, and Abbreviations

JWT (JSON Web Token): A compact, URL-safe means of representing claims to be transferred between two parties.

BCrypt: A password-hashing function used to securely store passwords.

SPA: Single Page Application (React).

Endpoint: A specific URL where an API can be accessed (e.g., /api/auth/login).

2. Overall Description

2.1. System Perspective

This module acts as the "Front Door" of a larger web application. It sits between the public internet and the application's private data, ensuring that only identified users can interact with the backend resources.

2.2. User Classes and Characteristics

Guest User: Unauthenticated individuals who can only access the registration and login pages.

Authenticated User: Users who have successfully logged in and have permission to view their profile and perform system-specific tasks.

2.3. Operating Environment

- **Frontend:** React.js, Node.js environment, Axios for API calls.
- **Backend:** Spring Boot (Java), Spring Security, Hibernate/JPA.
- **Database:** PostgreSQL or MySQL.
- **Tools:** Draw.io (for diagrams), Postman (for API testing).

2.4. Assumptions and Dependencies

- Users have a valid email address for registration.
- The browser supports local storage or cookies for saving the authentication token.
- The Spring Boot server is reachable via REST API calls from the React frontend.

3. System Features and Functional Requirements

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

3.1. Feature 1: User Identity Management (Registration)

Description: Allows new users to create a permanent account in the system database.
Functional Requirements:
- The system shall provide a form to collect username, email, and password.
- The system shall validate that the email format is correct and not already registered.
- The system shall encrypt the password using BCrypt before saving it to the database.

3.2. Feature 2: Authentication and Session Control (Login/Logout) Description:

Validates user credentials and manages the "logged-in" state.
Functional Requirements:
- The system shall compare the provided credentials against the stored hashed password.
- The system shall issue a JWT upon successful authentication.
- The system shall allow the user to terminate the session (Logout), which clears the token from the client-side storage.

4. Non-Functional Requirements

Security: Passwords must never be stored in plain text. All API communication must be stateless (via JWT).

Performance: The authentication process (from login click to dashboard view) should take less than 2 seconds under normal network conditions.

Usability: The UI must provide clear feedback for errors (e.g., "Invalid credentials" or "Email already taken").

Reliability: The system should handle database connection failures gracefully by showing a "Service Unavailable" message.

5. System Models (Diagrams)

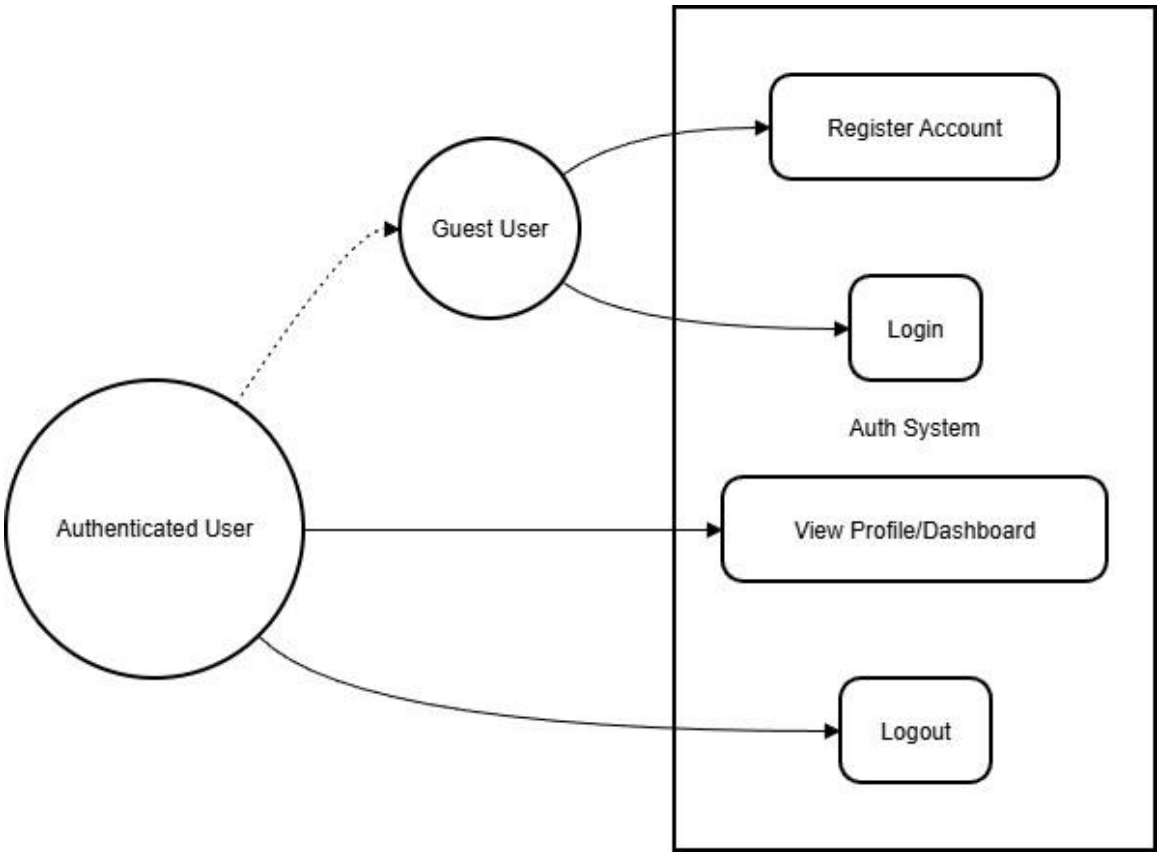
Insert the necessary diagrams for the system:

5.1. ERD

USERS		
Long	id	PK
String	email	UK
String	password_hash	
String	first_name	
String	last_name	
LocalDateTime	created_at	
LocalDateTime	last_login	
Boolean	is_active	

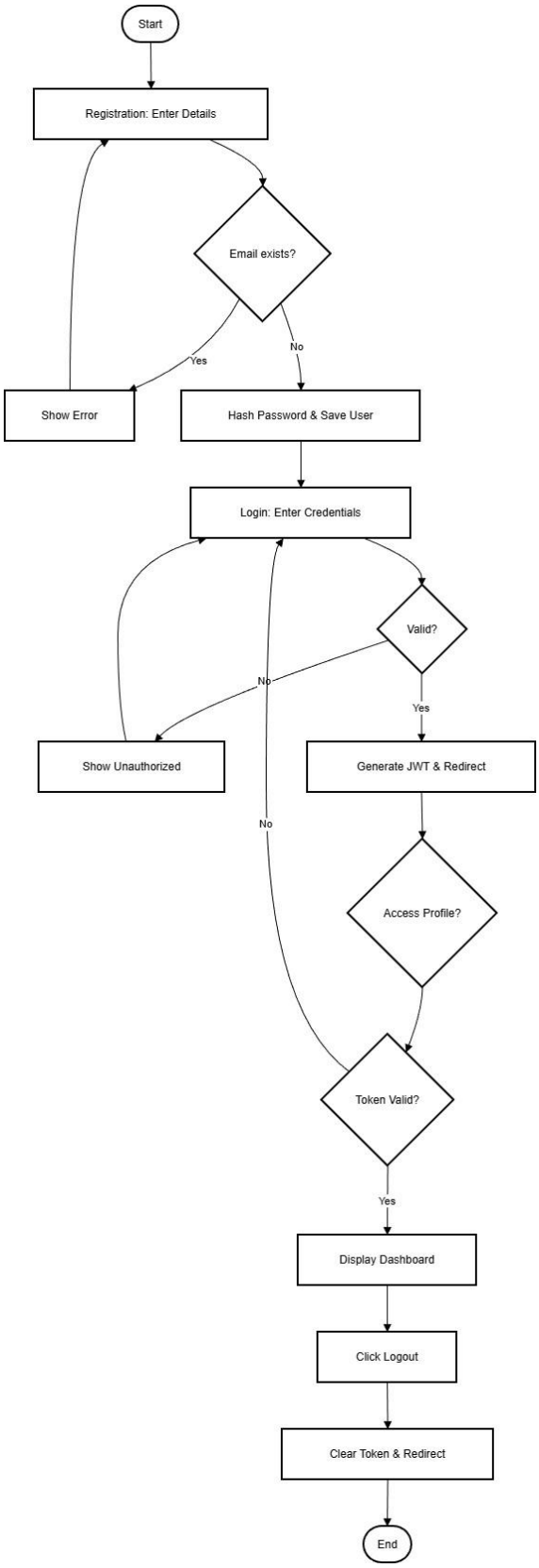
5.2.

Use Case Diagram



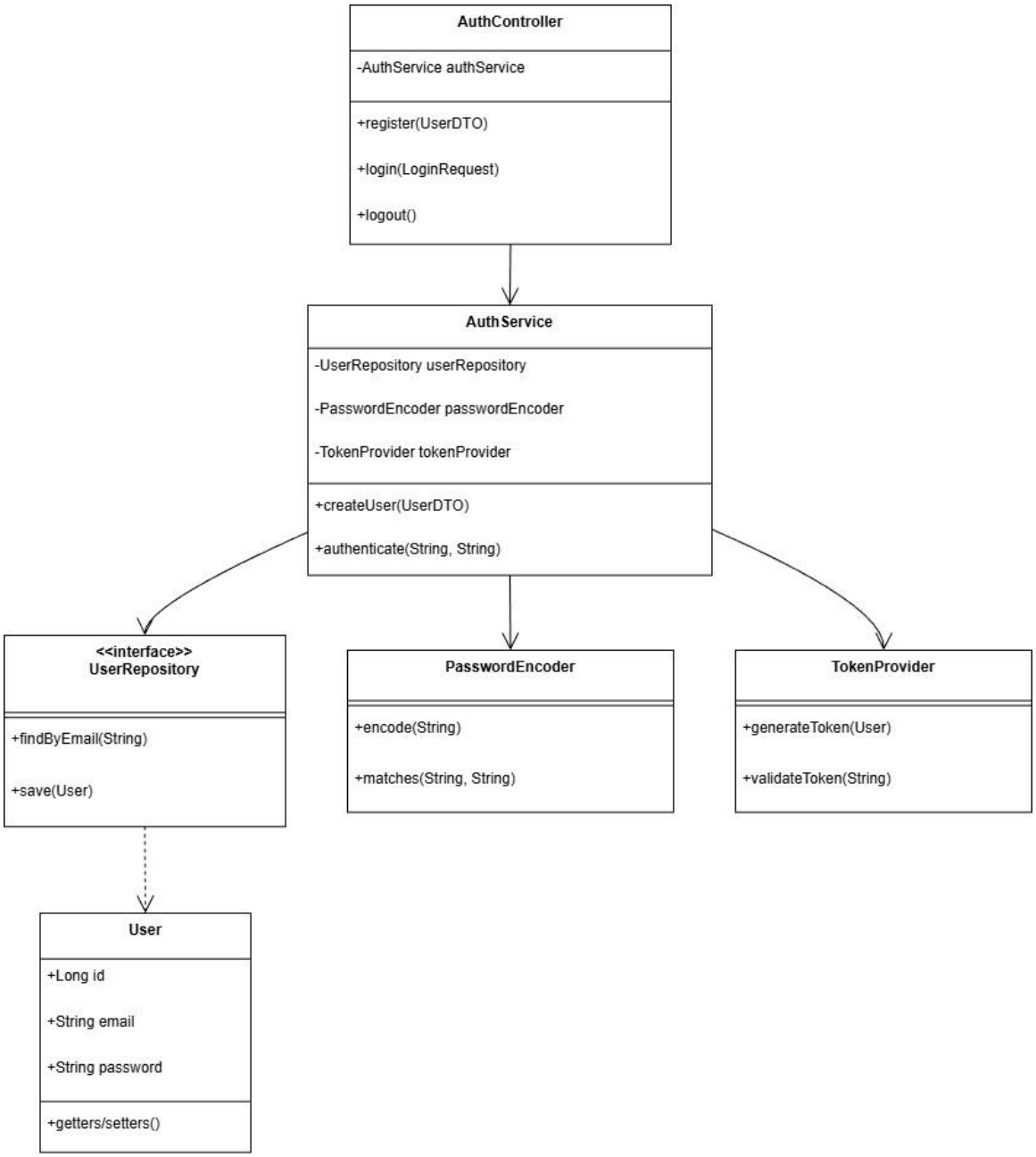
Activity Diagram

5.3.


Start
Registration: Enter Details
Email exists?
Yes
No
Show Error
Hash Password & Save User
Login: Enter Credentials
Valid?
No
Yes
Show Unauthorized
Generate JWT & Redirect
No
Access Profile?
Token Valid?
Yes
Display Dashboard
Click Logout
Clear Token & Redirect
End

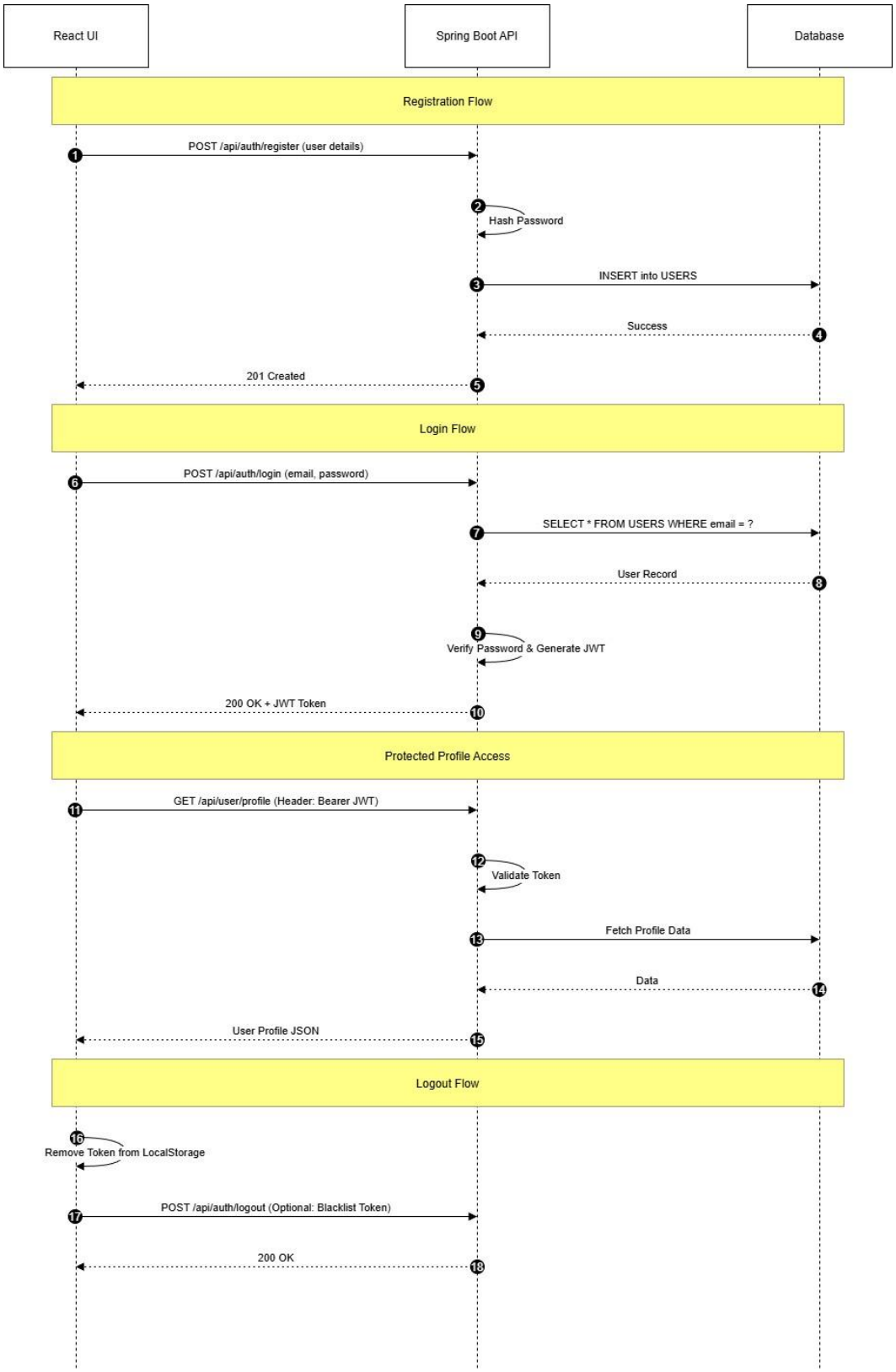
5.4.

Class Diagram



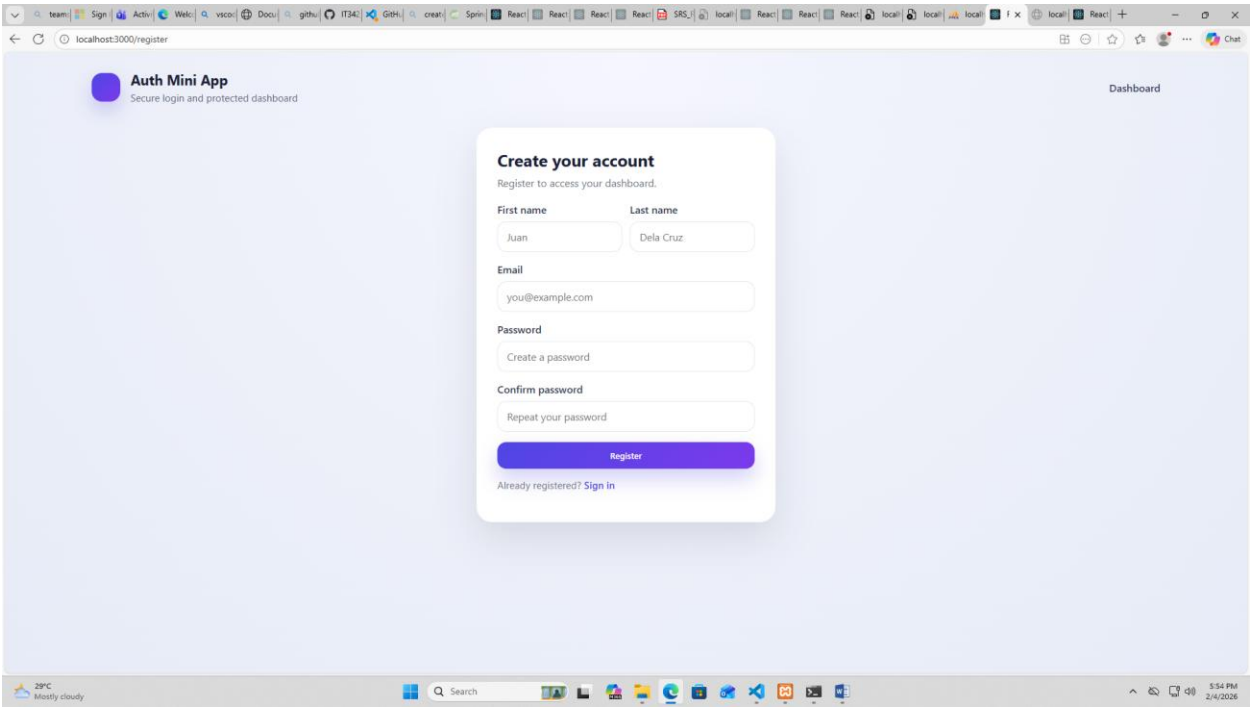
Sequence Diagram

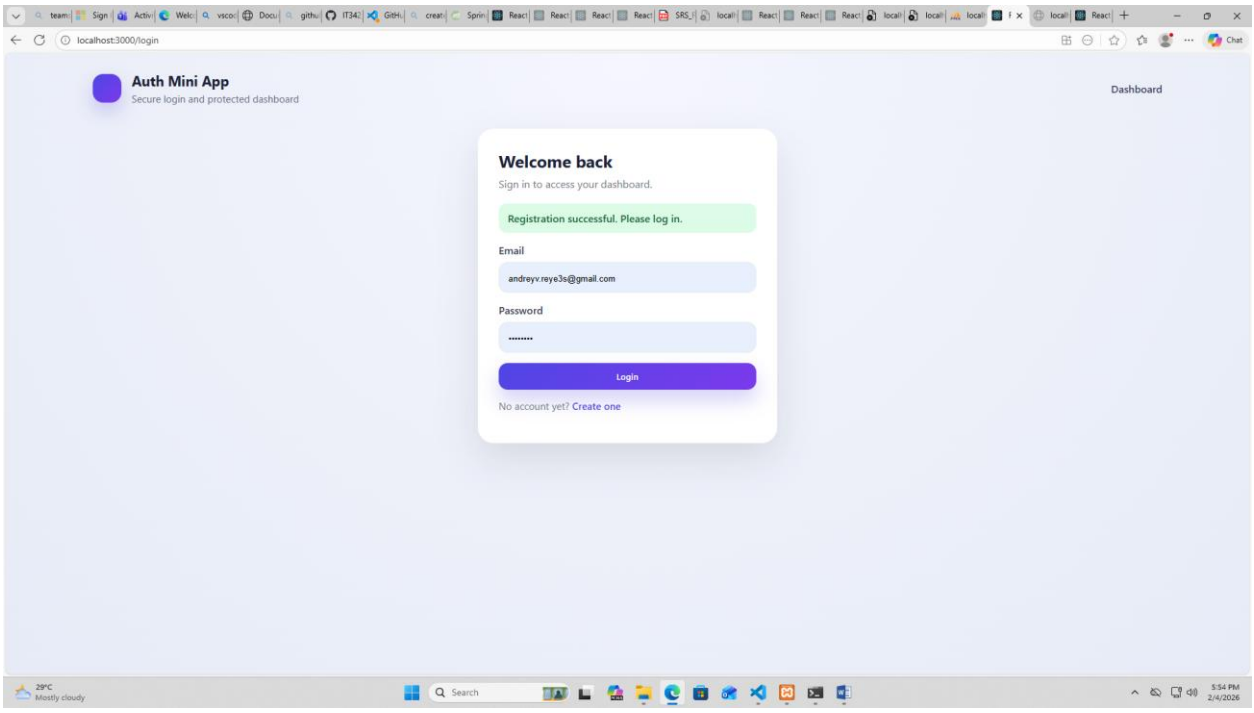
5.5.

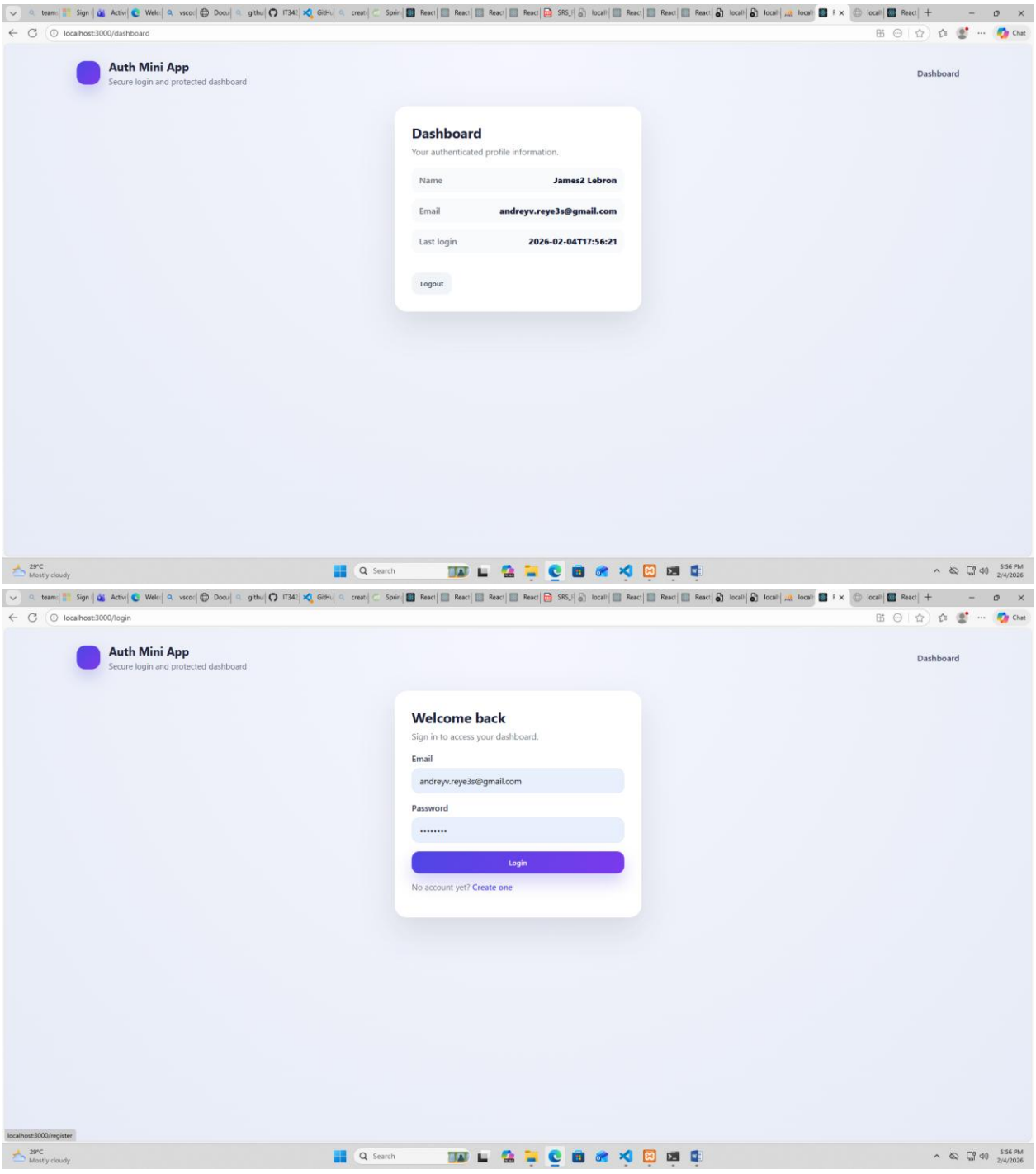

React UI
Spring Boot API
Database
Registration Flow
1 POST /api/auth/register (user details)
2 Hash Password
3 INSERT into USERS
4 Success
5 201 Created
Login Flow
6 POST /api/auth/login (email, password)
7 SELECT * FROM USERS WHERE email = ?
8 User Record
9 Verify Password & Generate JWT
10 200 OK + JWT Token
Protected Profile Access
11 GET /api/user/profile (Header: Bearer JWT)
12 Validate Token
13 Fetch Profile Data
14 Data
15 User Profile JSON
Logout Flow
16 Remove Token from LocalStorage
17 POST /api/auth/logout (Optional: Blacklist Token)
18 200 OK

6. Screenshots of UI

WEB







6. Appendices

Include any additional information, references, or support materials