CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342-Section SYSTEMS INTEGRATION AND ARCHITECTURE 1

FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

Project Title: Mini App

Prepared By: John Kenneth Devibar

Date of Submission: 02/10/2026

Version: 3

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this document is to define the functional and non-functional requirements for a secure authentication system.

1.2. Scope

The system will handle user identity management that allows new and existing users to access protected pages such as dashboards.

1.3. Definitions, Acronyms, and Abbreviations

API - Application Programming Interface

JWT - JSON Web Token

PK / FK - Primary Key / Foreign Key

SRS - Software Requirements Specification

2. Overall Description

2.1. System Perspective

This system acts as the "Security Layer" of a larger web application. It follows a **Client-Server architecture** where the React frontend is completely separated from the Spring Boot backend, communicating exclusively through JSON-based REST APIs.

2.2. User Classes and Characteristics

Guest User: Has access to public routes (Register, Login).

Authenticated User: Has successfully logged in and holds a valid token; can access the Dashboard and Logout.

2.3. Operating Environment

Hardware: Standard PC/Server with internet access.

Software: Web Browser (Chrome/Edge/Firefox), Java 17+, Node.js, and a Relational Database (MySQL/PostgreSQL).

2.4. Assumptions and Dependencies

- Users have a unique email address.
- The system assumes the browser supports LocalStorage or Cookies for storing session tokens.

3. System Features and Functional Requirements

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

3.1. Feature 1:

Description: User Registration
Functional Requirements:
- The system shall provide a form to collect username, email, password, and full_name.

- The system shall validate that the email is in a valid format.

- The system shall hash the password before saving it to the database.

3.2. Feature 2:

Description: User Authentication
Functional Requirements:
- The system shall verify credentials against the database.

- The system shall return a JWT upon successful authentication.

- The system shall invalidate the client-side session upon Logout.

4. Non-Functional Requirements

Security

Password Hashing: The system must never store passwords in plain text; all passwords must be hashed using a strong algorithm like BCrypt.

Token-Based Auth: The system shall use JWT (JSON Web Tokens) for session management to ensure stateless and secure communication between React and Spring Boot.

Data Integrity: Sensitive data transmitted between the client and server should ideally be encrypted (via HTTPS/TLS).

Performance

Response Time: The authentication API (Login/Register) should respond within 500ms under normal load conditions.

Efficiency: The system should handle concurrent login requests without a significant increase in latency.

Usability

Feedback Mechanism: The system must provide clear, user-friendly error messages (e.g., "Invalid username or password" or "Email already registered") rather than raw technical stack traces.

Responsiveness: The React UI must be responsive, ensuring the login and registration forms are easily accessible on both desktop and mobile browsers.

Availability

Data Persistence: Once a user registers, their data must be reliably stored in the database and be retrievable for subsequent logins.

Error Handling: The system should gracefully handle database connection failures by displaying a "Service Temporarily Unavailable" message to the user.

Scalability

Decoupled Architecture: By separating the React frontend and Spring Boot backend, the system should allow for independent scaling of the UI and the API.

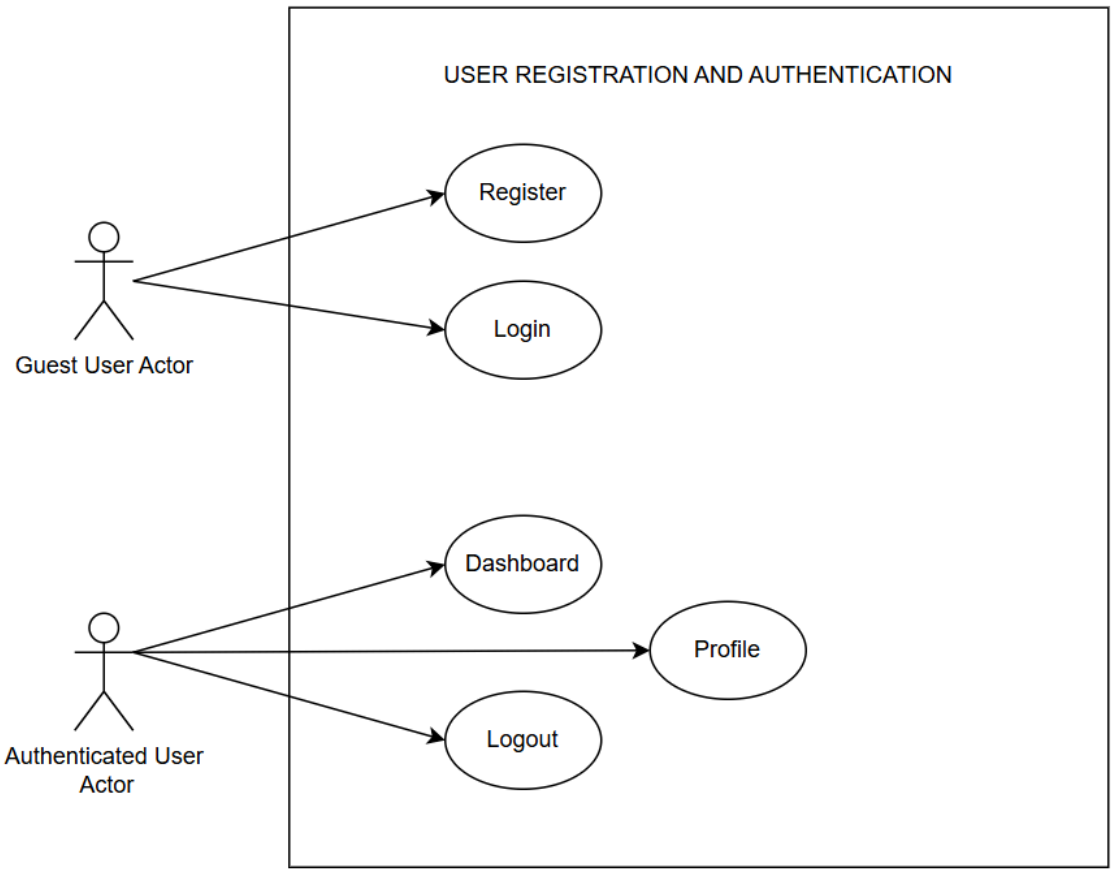
5. System Models (Diagrams)

Insert the necessary diagrams for the system:

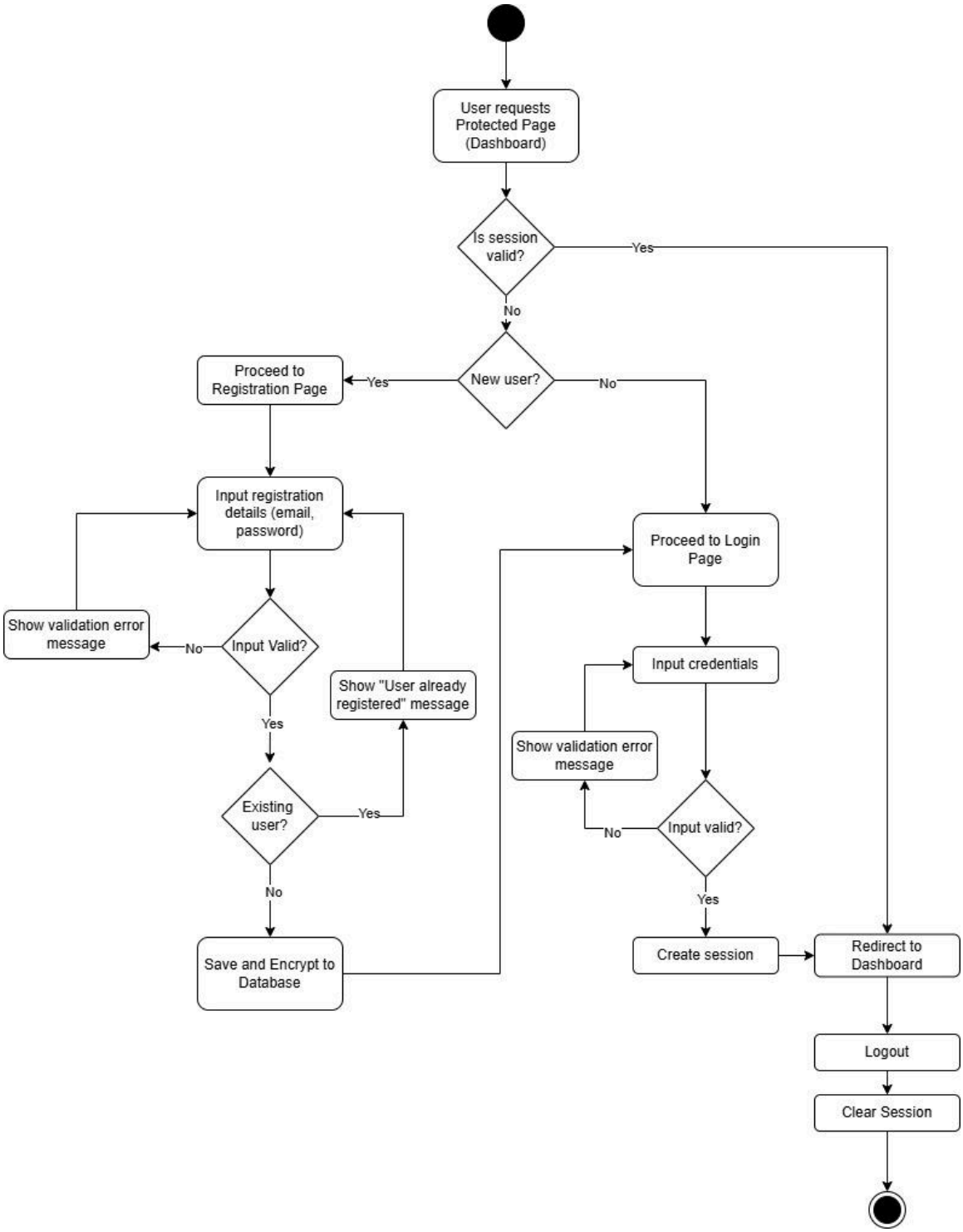
5.1. ERD

USER		
PK	userID	VARCHAR(15)
	username	VARCHAR(50)
	email	VARCHAR(100)
	password	VARCHAR(255)
	full_name	VARCHAR(100)
	created_at	TIMESTAMP

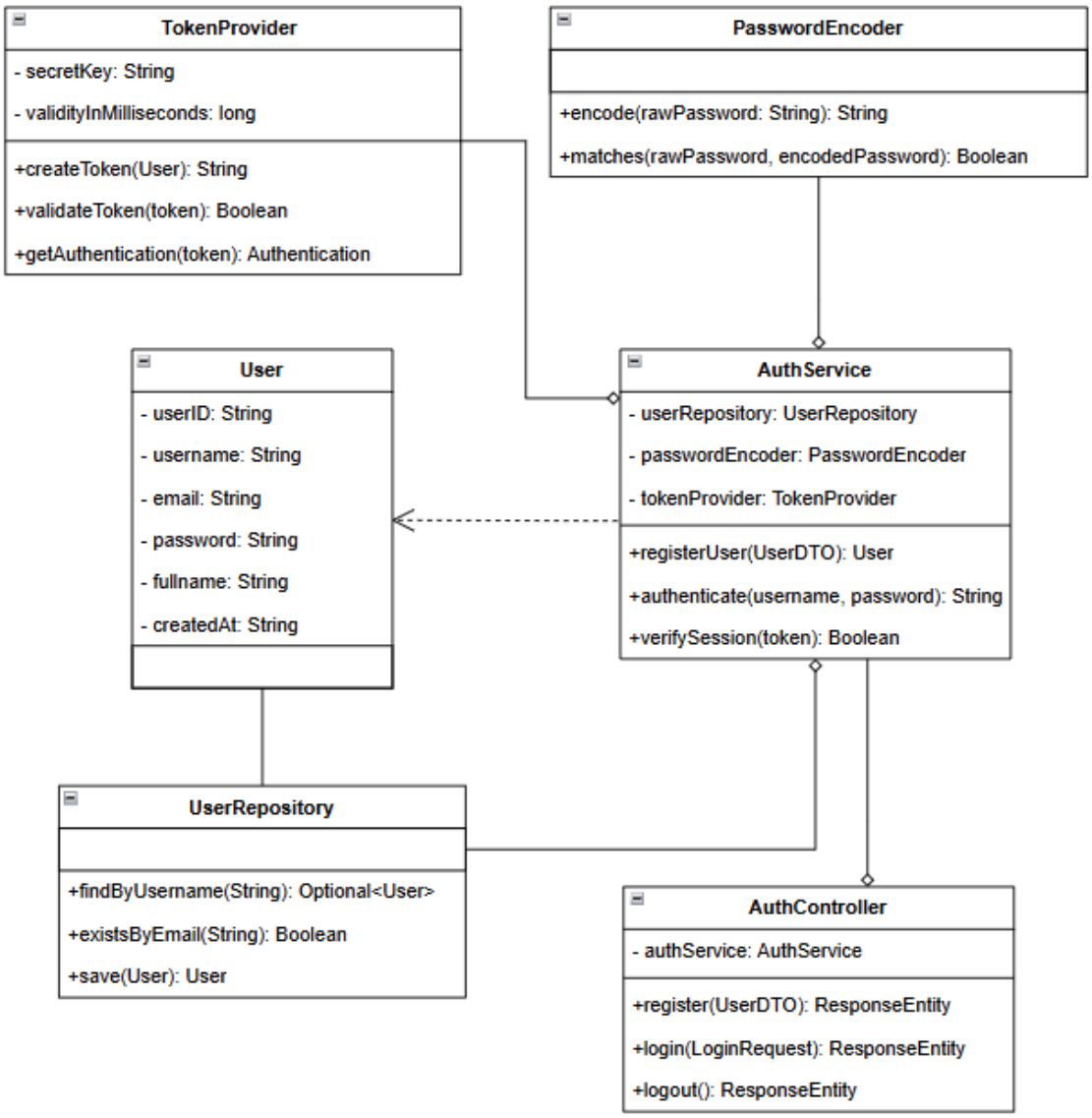
5.2. Use Case Diagram



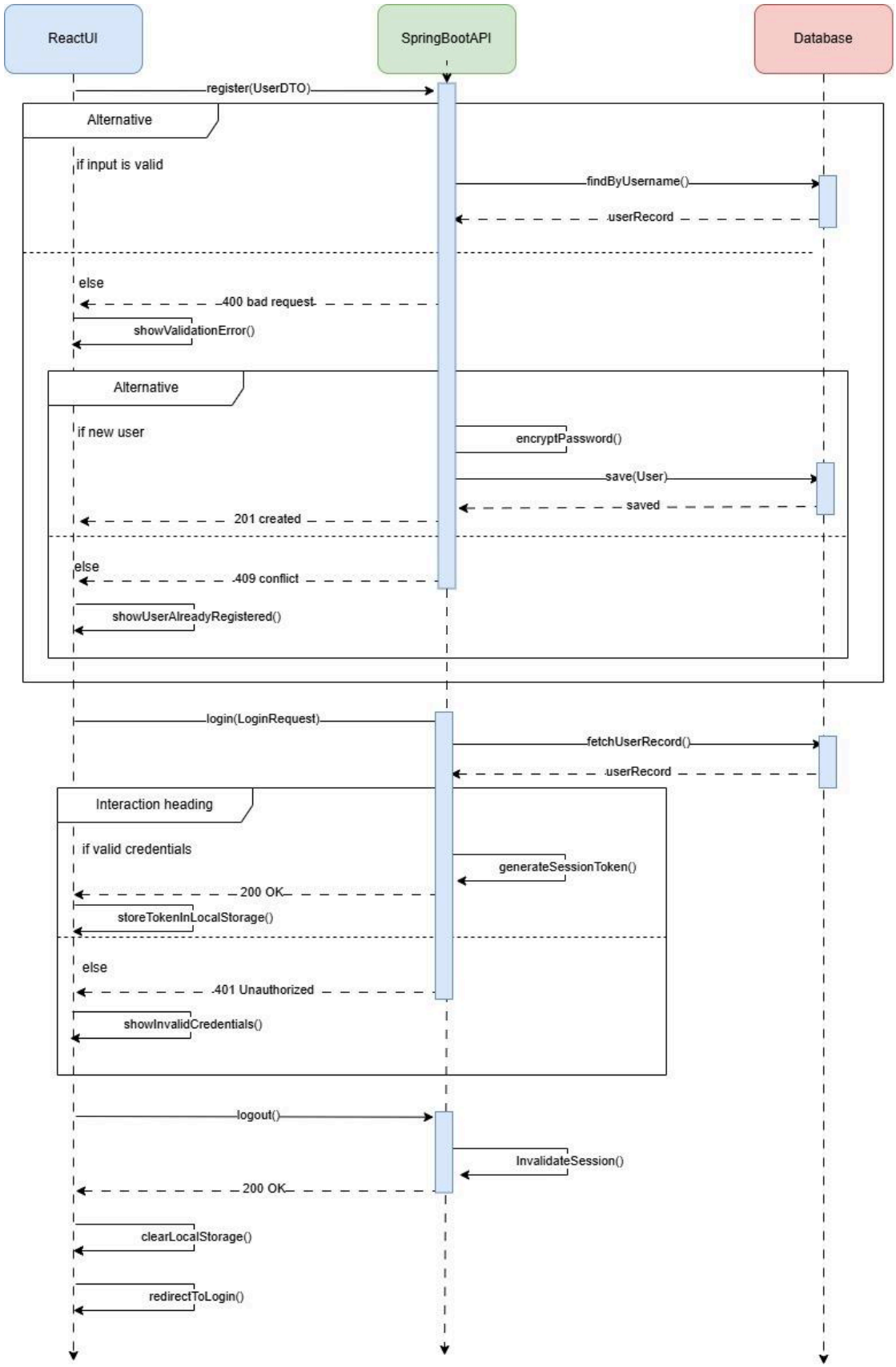
5.3. Activity Diagram



5.4. Class Diagram



5.5. Sequence Diagram



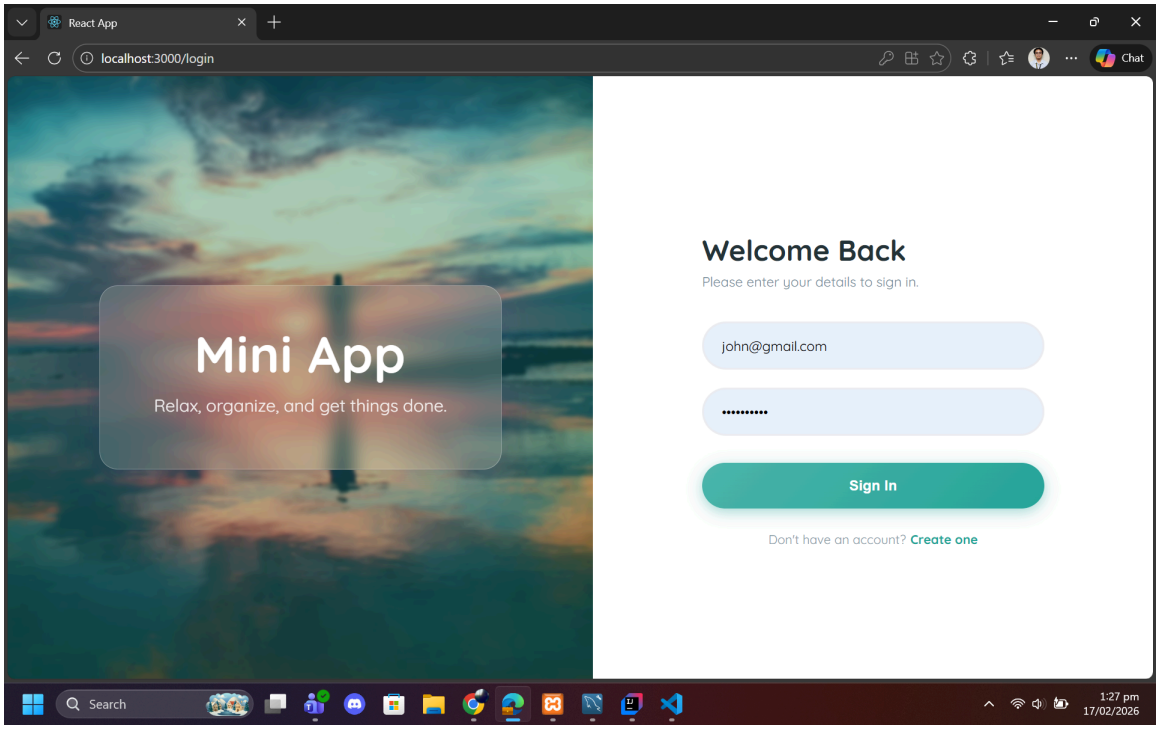
6. Appendices

Tools used: Draw.io for diagrams.

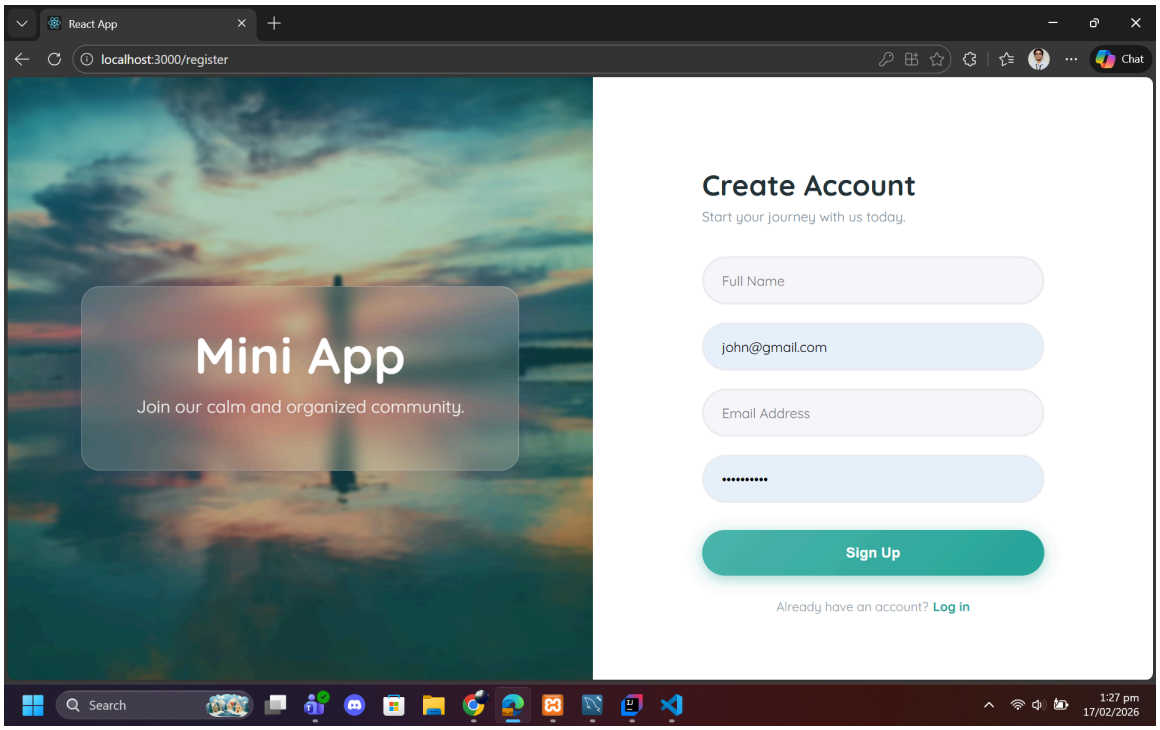
Tech Stack: React (Frontend), Spring Boot (Backend), Spring Security (Auth), MySQL/PostgreSQL (DB).

7. Web Screenshots

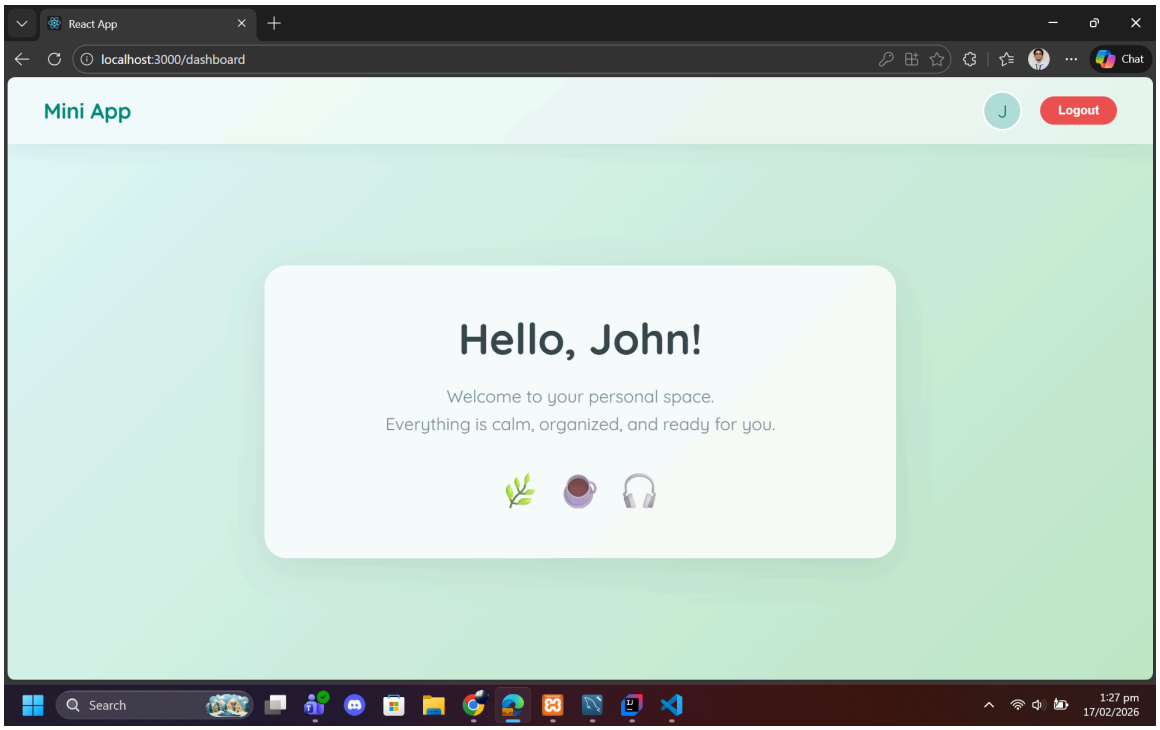
LOGIN



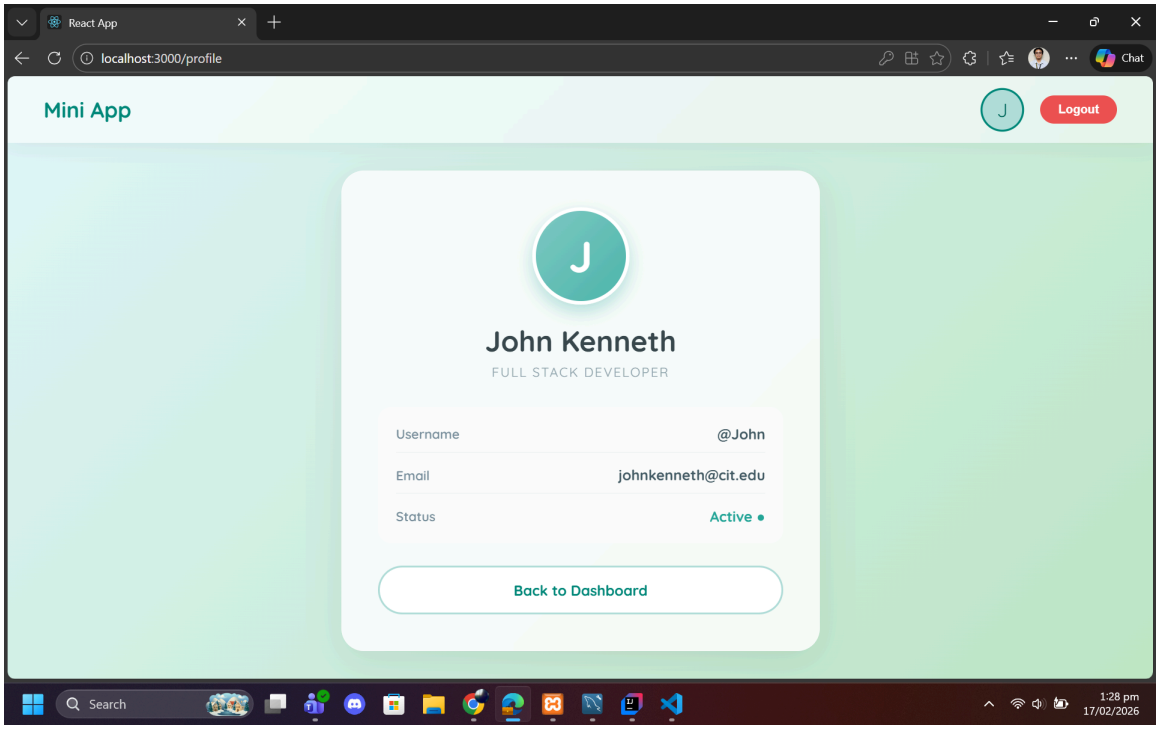
REGISTER



DASHBOARD



PROFILE



LOGOUT