CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342-Section SYSTEMS INTEGRATION AND ARCHITECTURE 1

FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

Project Title: Mini App

Prepared By: Richemmae V. Bigno

Date of Submission: 02/02/2026

Version: 1.0

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this system is to provide a secure, standardized gateway for user identity management. This document serves as a blueprint for developers to implement a functional Registration, Login, and Logout system using a React-Spring Boot stack. The intended audience includes the development team and course evaluators for the upcoming face-to-face coding session.

1.2. Scope

The system handles the lifecycle of a user session.

- Included: User account creation, credential validation via BCrypt, JWT/Session token generation, and protected route redirection.
- Boundaries: This system does not include password recovery (forgot password), multi-factor authentication, or third-party OAuth (like Google Login).

1.3. Definitions, Acronyms, and Abbreviations

- BCrypt: A password-hashing function used to secure credentials.
- JWT (JSON Web Token): A compact, URL-safe means of representing claims to be transferred between two parties.
- SPA: Single Page Application (React).
- Endpoint: A specific URL where the API can be accessed (e.g.,/api/auth/login).
- DTO (Data Transfer Object): Objects used to pass data between the frontend and backend
- Token Blacklist: A server-side store of invalidated tokens that have not yet expired.

2. Overall Description

2.1. System Perspective

This mini-app acts as the Identity Provider (IdP) layer. It sits between the user's browser (Client) and the application database. It follows a decoupled architecture where the React frontend communicates with the Spring Boot backend via RESTful APIs.

2.2. User Classes and Characteristics

- Guest (Unauthenticated): Users who can only access the landing, registration, and login pages.
- User (Authenticated): Users with a valid session who can access the private Dashboard and Profile pages.

2.3. Operating Environment

- Frontend: React.js (Node.js environment).
- Backend: Java 17+, Spring Boot 3.x.
- Database: H2 (In-memory) or MySQL.
- Tools: Visual Studio Code, IntelliJ IDEA, Postman (for API testing).

2.4. Assumptions and Dependencies

- Assumption: Users have a modern web browser with JavaScript enabled.
- Dependency: The system requires a persistent connection between the Client and the Server for token verification.

3. System Features and Functional Requirements

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

3.1. Feature 1: Identity Creation (Registration)

Description: Allows new users to create an account by providing unique credentials.
Functional Requirements:
- FR 1.1: The React UI shall provide separate input fields for username, email, and password.
- FR 1.2: The system shall trigger the registration process only when the "Register" button is clicked.
- FR 1.3: The Spring Boot API shall verify if the username/email exists in the Database.
- FR 1.4 (Alt - Success): If the user is unique, the API shall hash the password via BCrypt and return a 201 Created status
- FR 1.5 (Alt - Failure): If the user exists, the API shall return a 409 Conflict status, and the UI shall display an error message.

3.2. Feature 2: Authentication (Login)

Description: Validates user credentials to grant access to the application.
Functional Requirements:
- FR 2.1: The UI shall capture credentials and send a POST request to the API upon the "Login" click.
- FR 2.2: The AuthService shall perform a verifyPassword() self-check comparing input to the database record.
- FR 2.3 (Alt - Success): Upon successful match, the API shall return a JWT Token.
- FR 2.4: The React UI shall execute saveToken() to persist the session.
- FR 2.5 (Alt - Failure): If credentials fail, the API shall return 401 Unauthorized, and the UI shall prompt the user to try again.

3.3. Feature 3: Access Control (Protected Dashboard)

Description: Restricts access to sensitive pages based on token presence and validity.
Functional Requirements:
- FR 3.1: The system shall intercept navigation to /dashboard to check for a valid JWT.
- FR 3.2: If the token is missing or invalid, the system shall redirect the user to the /login route.

3.4. Feature 4: Session Termination (Logout)

Description: Ends the user session on both the client and server.
Functional Requirements:

- FR 4.1: Upon clicking "Logout," the React UI shall send a POST request to the /api/auth/logout endpoint.
- FR 4.2: The Spring Boot API shall add the current JWT to a Blacklist Store to prevent further use.
- FR 4.3: The React UI shall execute clearLocalStorage() to remove the token from the browser.
- FR 4.4: The system shall redirect the user to the Login page immediately after clearing the state.

4. Non-Functional Requirements

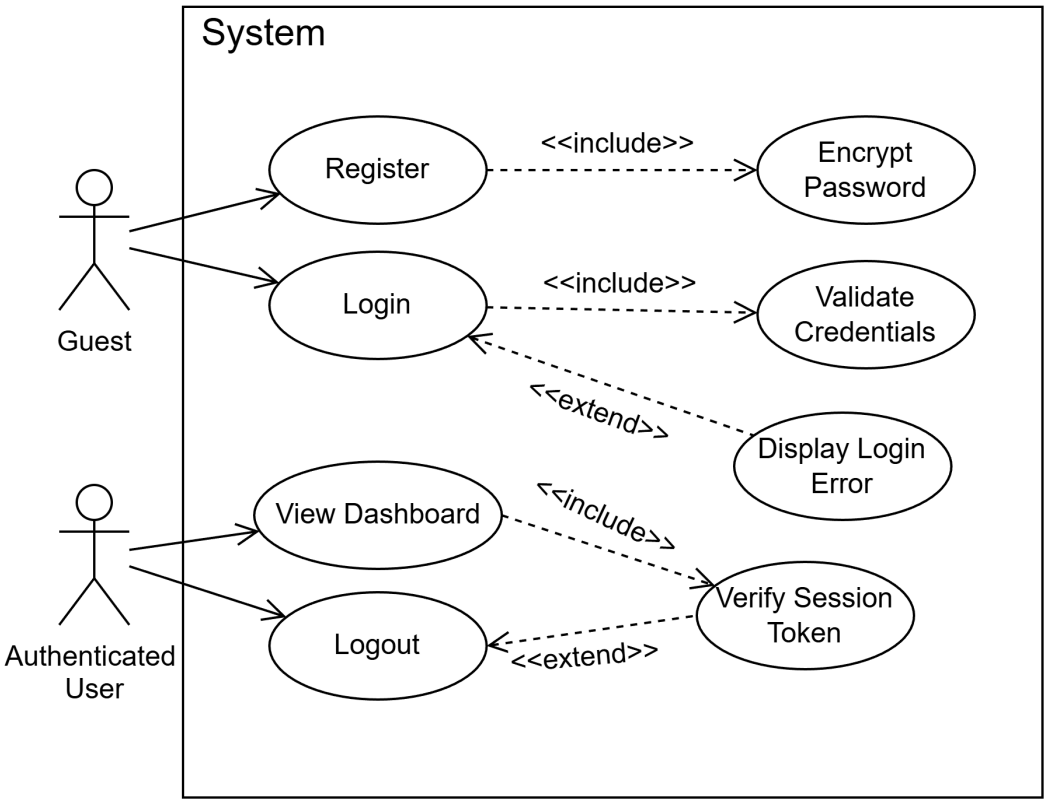
- Security: Passwords must be hashed via BCrypt; tokens must be blacklisted upon logout to prevent replay attacks.
- Performance: All authentication steps (hashing + DB lookup) should conclude in under 200ms.
- Usability: The UI must provide clear feedback (e.g., "Invalid username/password" or "Registration successful").
- Reliability: The system shall maintain state consistency between the React UI and the Spring Boot API during network interruptions.

5. System Models (Diagrams)

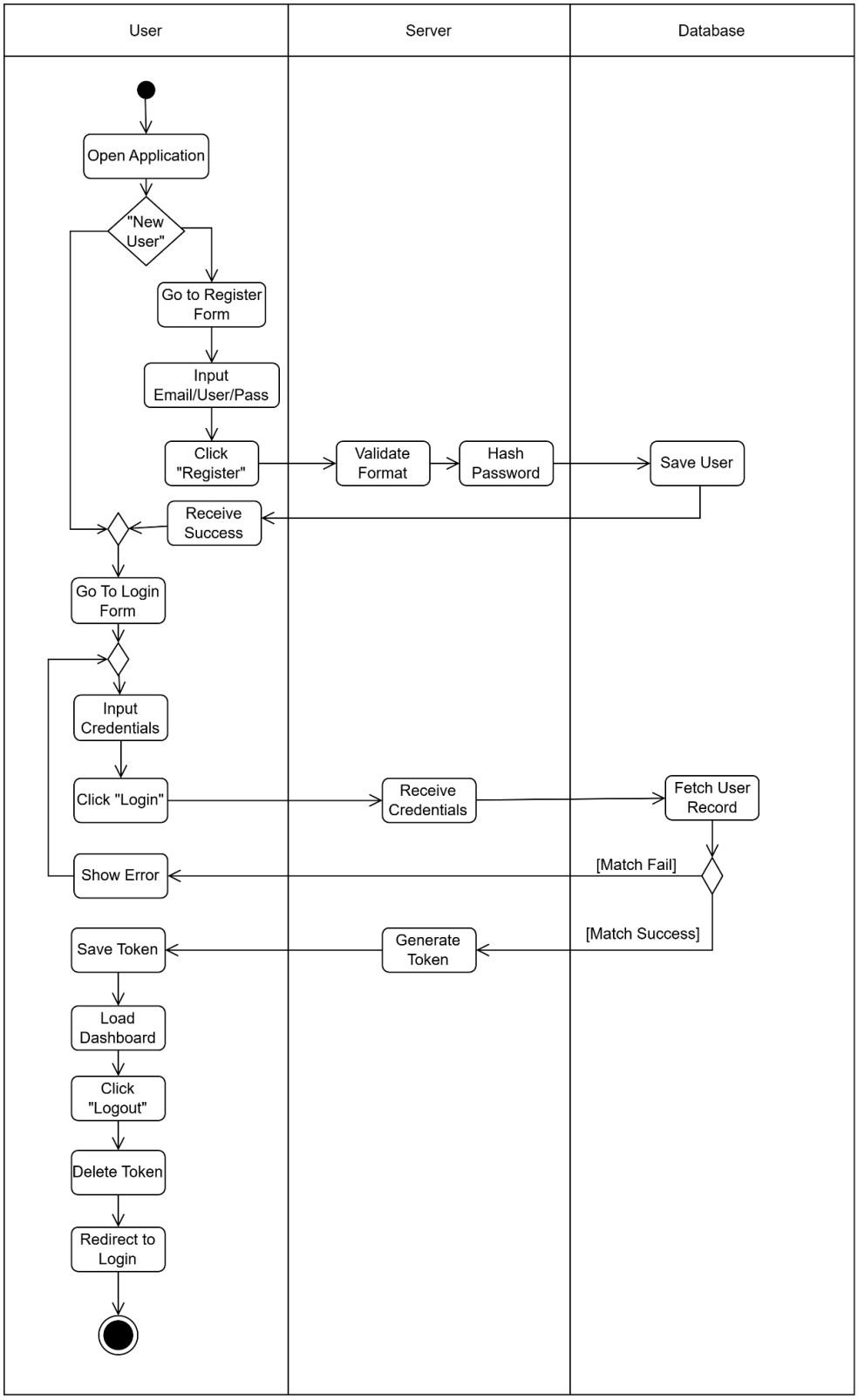
5.1. ERD

User		
PK	**user_id**	**bigint(20)**
	username	varchar(255)
	email	varchar(255)
	password	varchar(255)
	created_at	datetime(6)

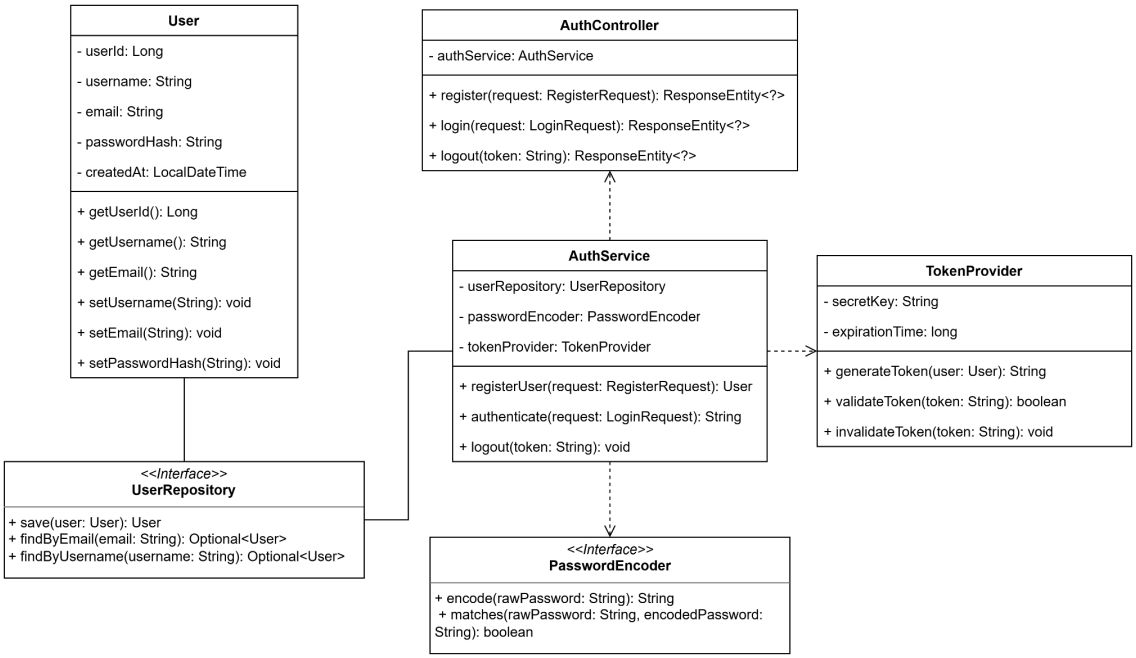
5.2. Use Case Diagram



5.3. Activity Diagram

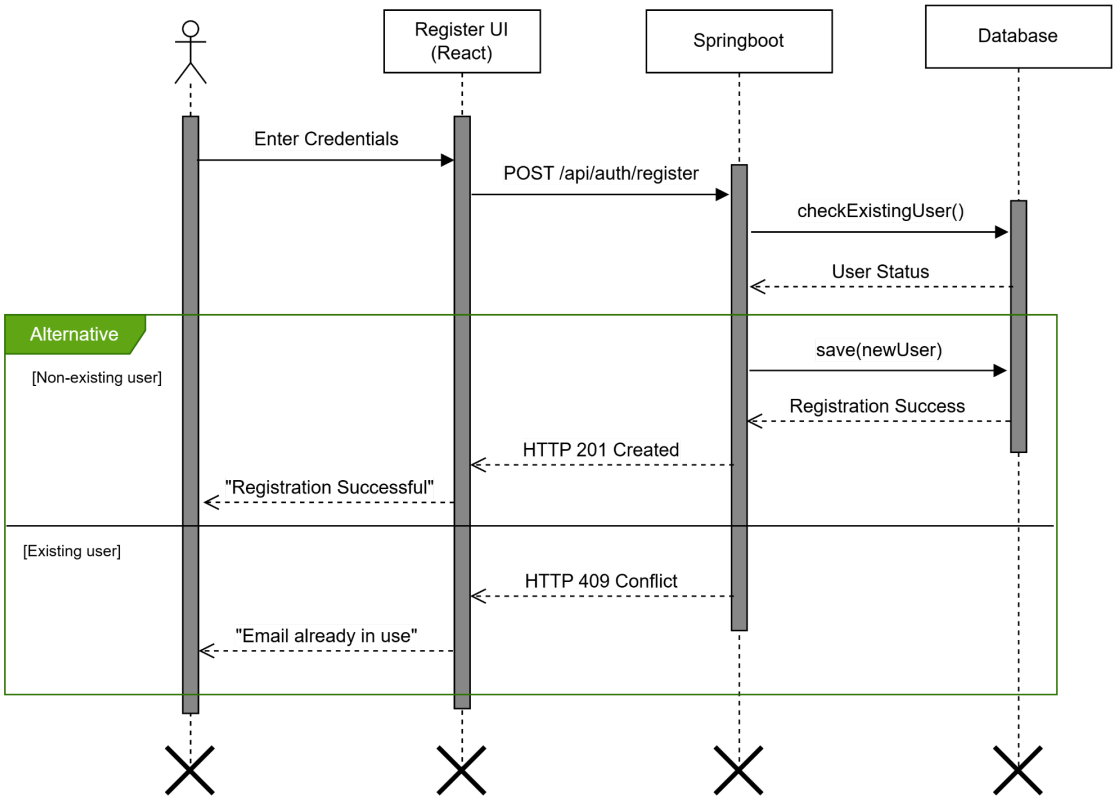


5.4. Class Diagram

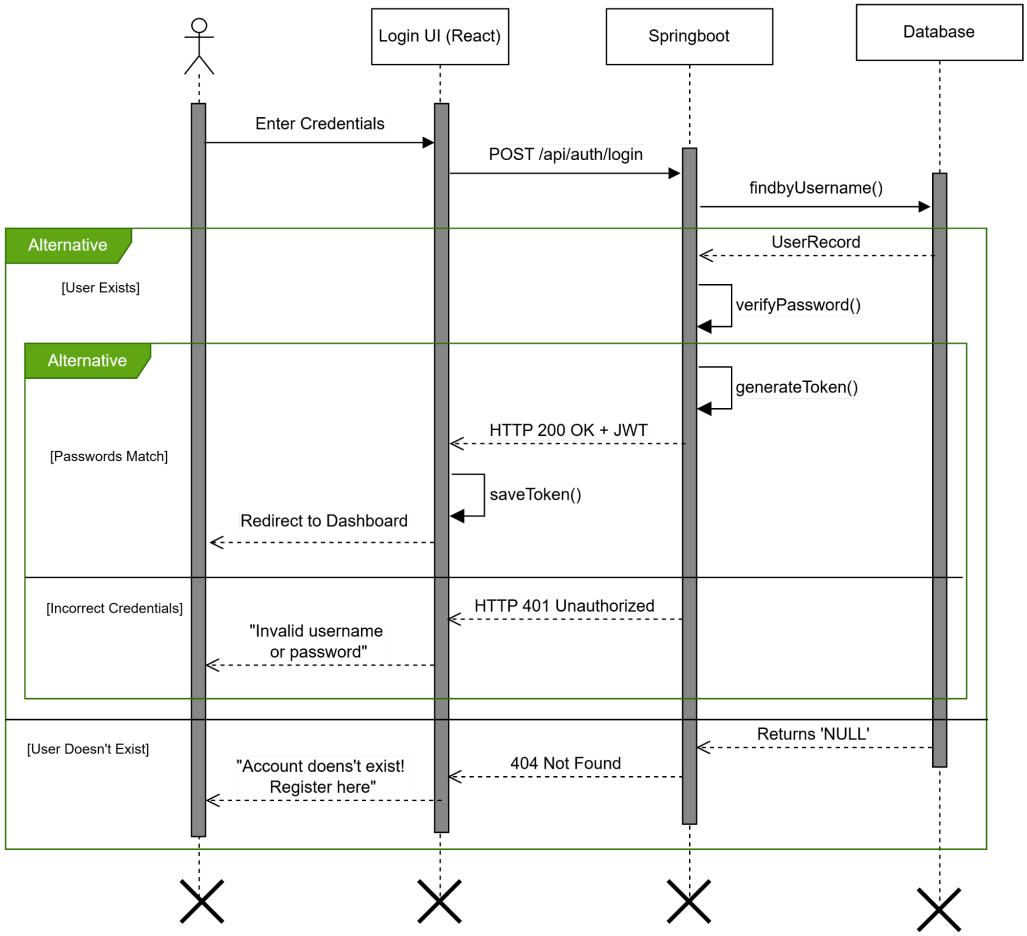


5.5. Sequence Diagram

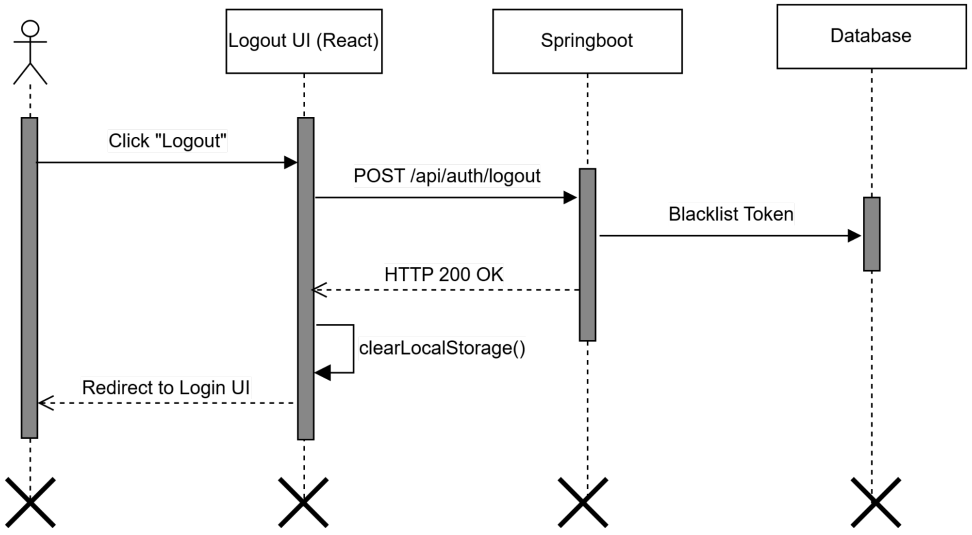
Register Sequence:



Login Sequence:



Logout Sequence



6. Appendices

Appendix A: Data Transfer Objects (DTOs)

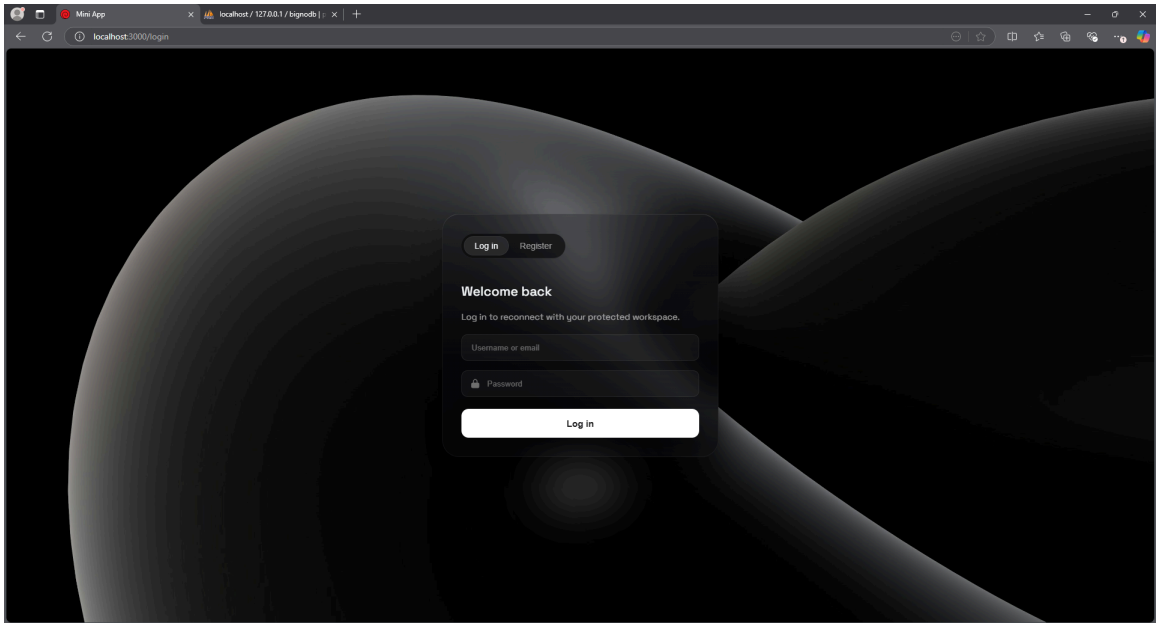
These are the Java classes that will represent the JSON data moving between React UI and Spring Boot API.

- RegistrationRequest: Contains username, email, and password.
- LoginRequest: Contains username and password.
- AuthResponse: Contains the accessToken (JWT) and username.

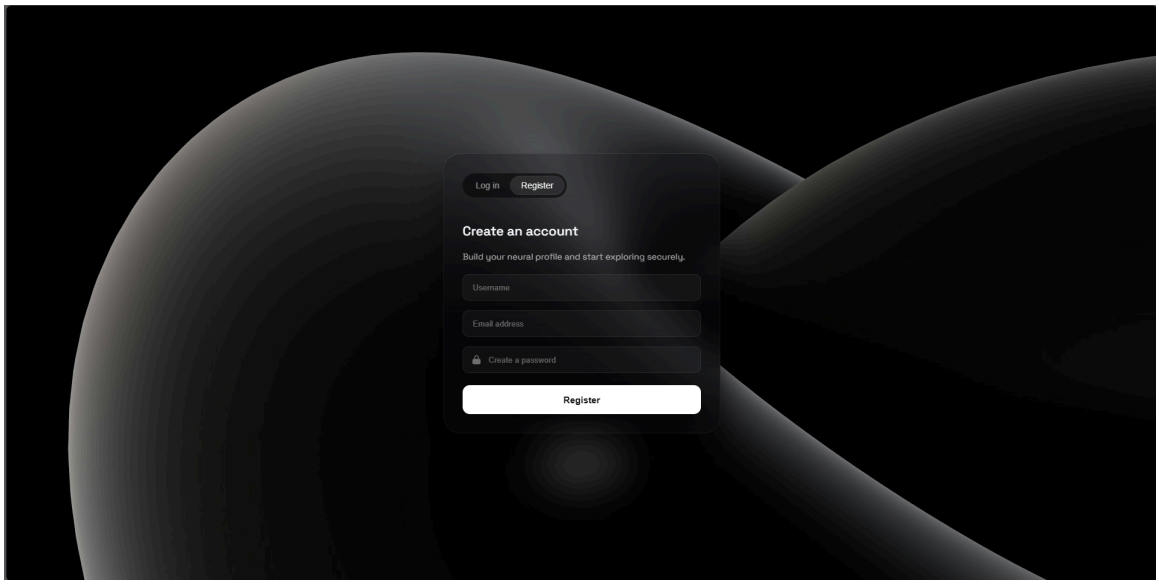
Appendix A: API Endpoints Table

Method	Endpoint	Description	Access
POST	/api/auth/register	Creates a new user account	Public
POST	/api/auth/login	Validates credentials and returns a token	Public
POST	/api/auth/logout	Adds the token to the blacklist	Authenticated
GET	/api/user/me	Returns details for the dashboard	Authenticated

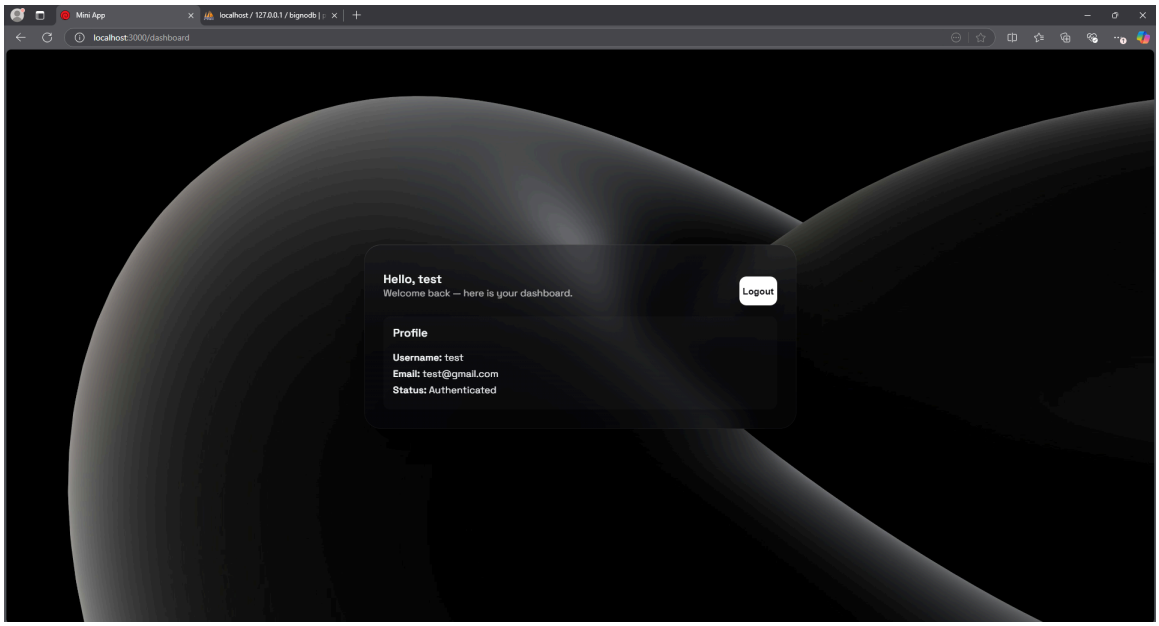
Appendix C: Login Page



Appendix D: Register Page



Appendix E: Dashboard Page



Appendix F: Log Out redirects to Log In Page