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

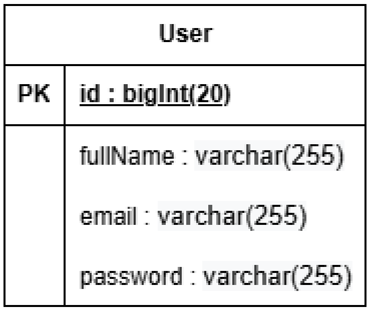
- The user interface shall be simple and easy to navigate
- Error messages shall be clear and user-friendly

Reliability

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

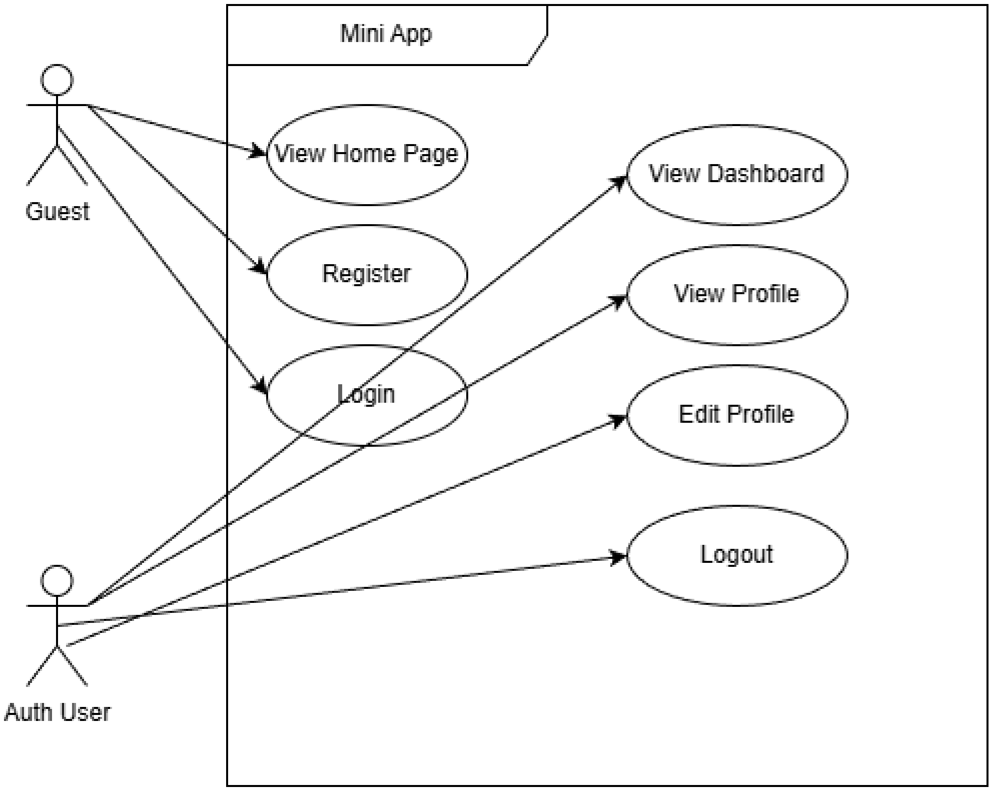
5. System Models (Diagrams)

5.1. ERD

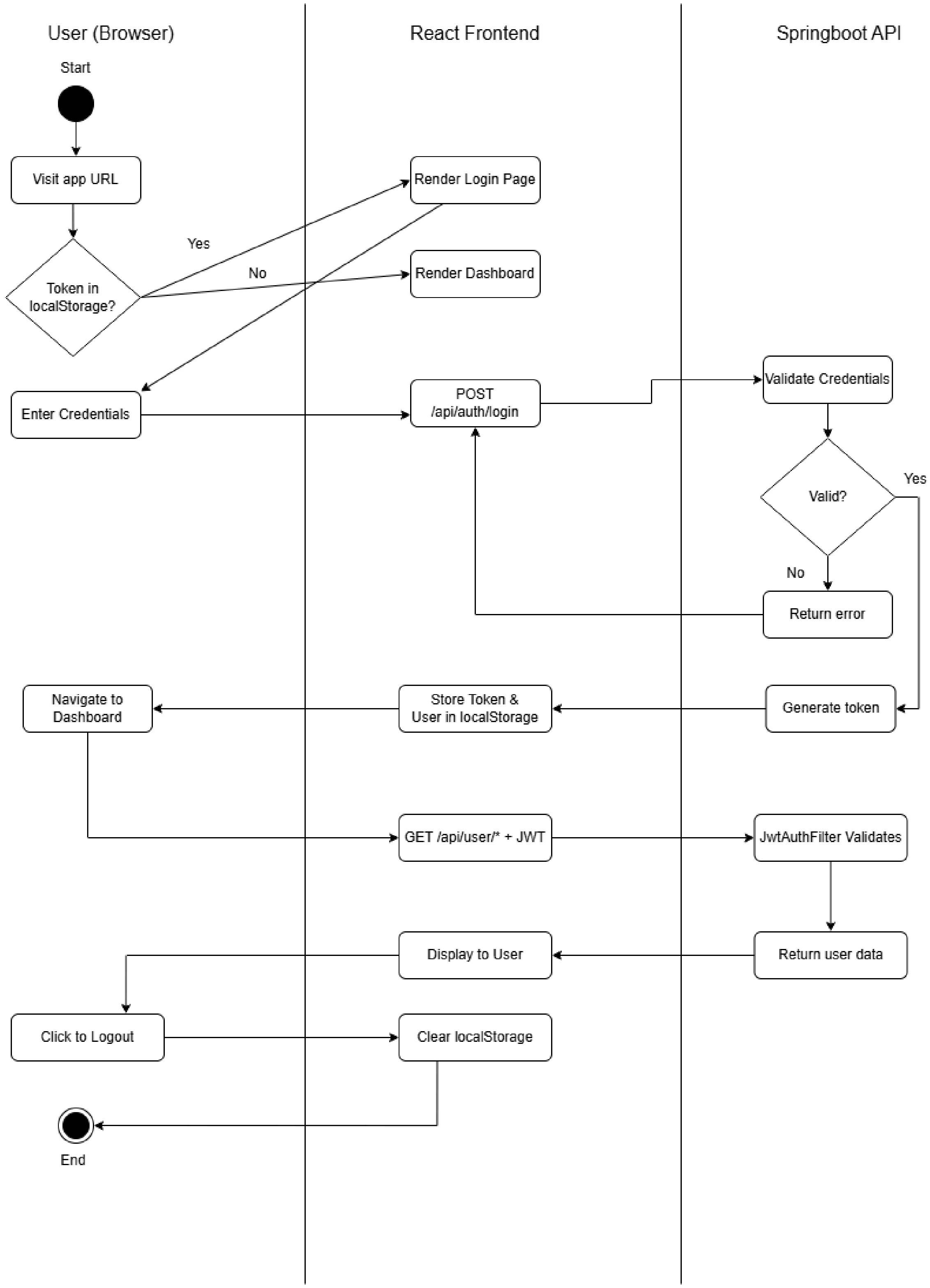


User	
PK	id : bigInt(20)
	fullName : varchar(255) email : varchar(255) password : varchar(255)

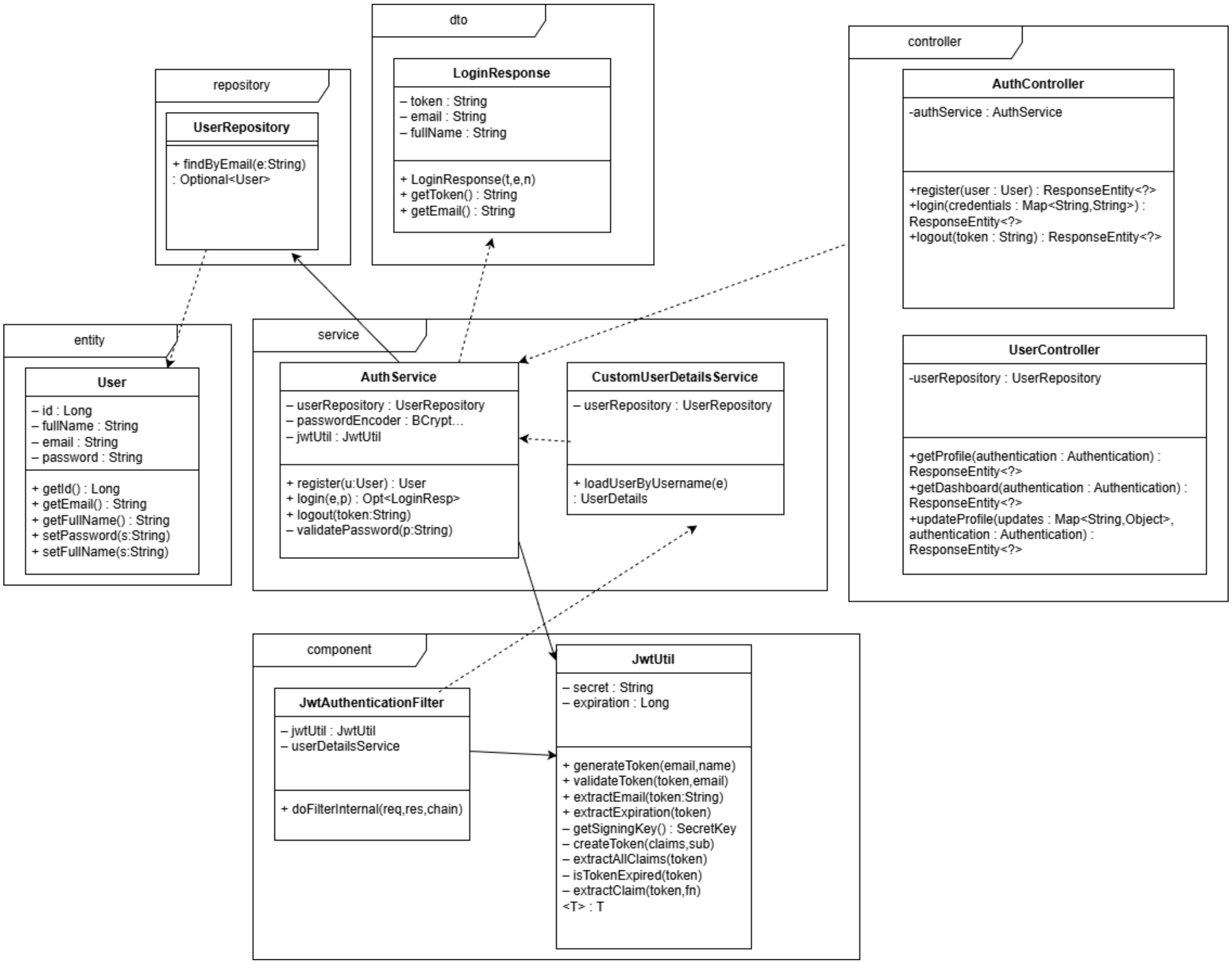
5.2. Use Case Diagram



5.3. Activity Diagram

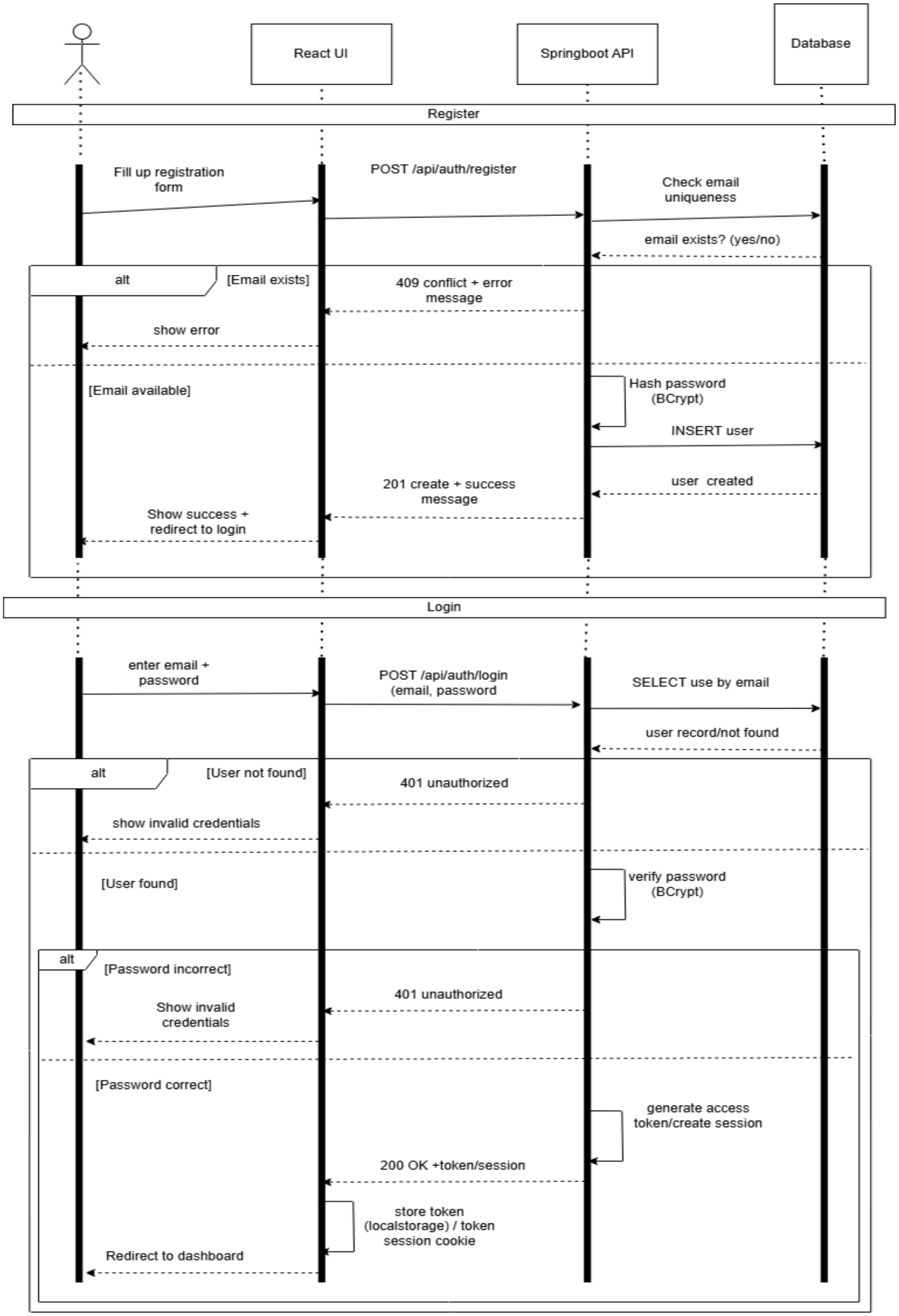


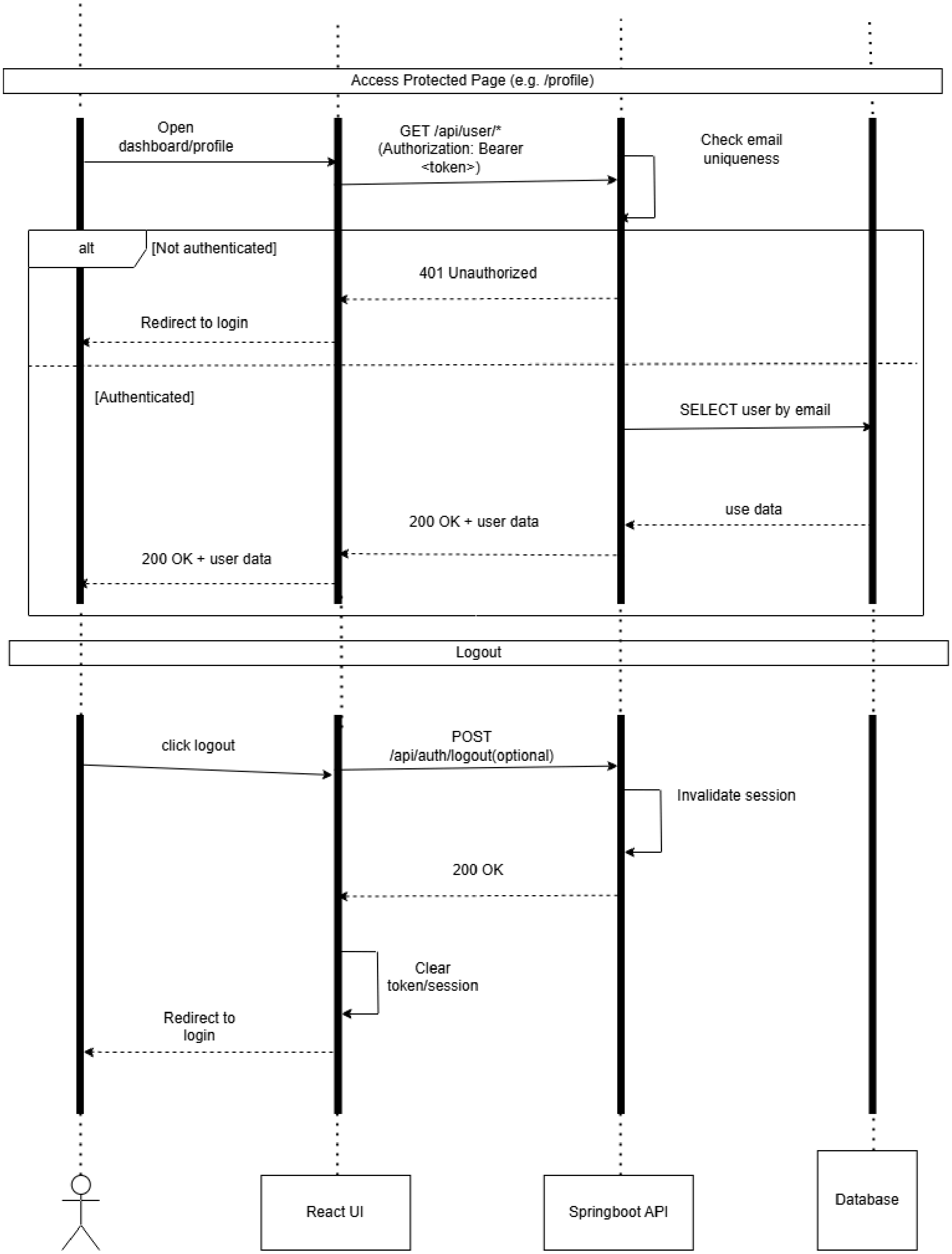
5.4. Class Diagram


dto
LoginResponse
– token : String
– email : String
– fullName : String
+ LoginResponse(t,e,n)
+ getToken() : String
+ getEmail() : String
controller
AuthController
-authService : AuthService
+register(user : User) : ResponseEntity<?>
+login(credentials : Map<String,String>) : ResponseEntity<?>
+logout(token : String) : ResponseEntity<?>
UserController
-userRepository : UserRepository
+getProfile(authentication : Authentication) : ResponseEntity<?>
+getDashboard(authentication : Authentication) : ResponseEntity<?>
+updateProfile(updates : Map<String,Object>, authentication : Authentication) : ResponseEntity<?>
repository
UserRepository
+ findByEmail(e:String) : Optional<User>
entity
User
– id : Long
– fullName : String
– email : String
– password : String
+ getId() : Long
+ getEmail() : String
+ getFullName() : String
+ setPassword(s:String)
+ setFullName(s:String)
service
AuthService
– userRepository : UserRepository
– passwordEncoder : BCrypt…
– jwtUtil : JwtUtil
+ register(u:User) : User
+ login(e,p) : Opt<LoginResp>
+ logout(token:String)
– validatePassword(p:String)
CustomUserDetailsService
– userRepository : UserRepository
+ loadUserByUsername(e) : UserDetails
component
JwtAuthenticationFilter
– jwtUtil : JwtUtil
– userDetailsService
+ doFilterInternal(req,res,chain)
JwtUtil
– secret : String
– expiration : Long
+ generateToken(email,name)
+ validateToken(token,email)
+ extractEmail(token:String)
+ extractExpiration(token)
– getSigningKey() : SecretKey
– createToken(claims,sub)
– extractAllClaims(token)
– isTokenExpired(token)
– extractClaim(token,fn) <T> : T

5.5. Sequence Diagram

Register, Login, ProtectedRoute, and Logout

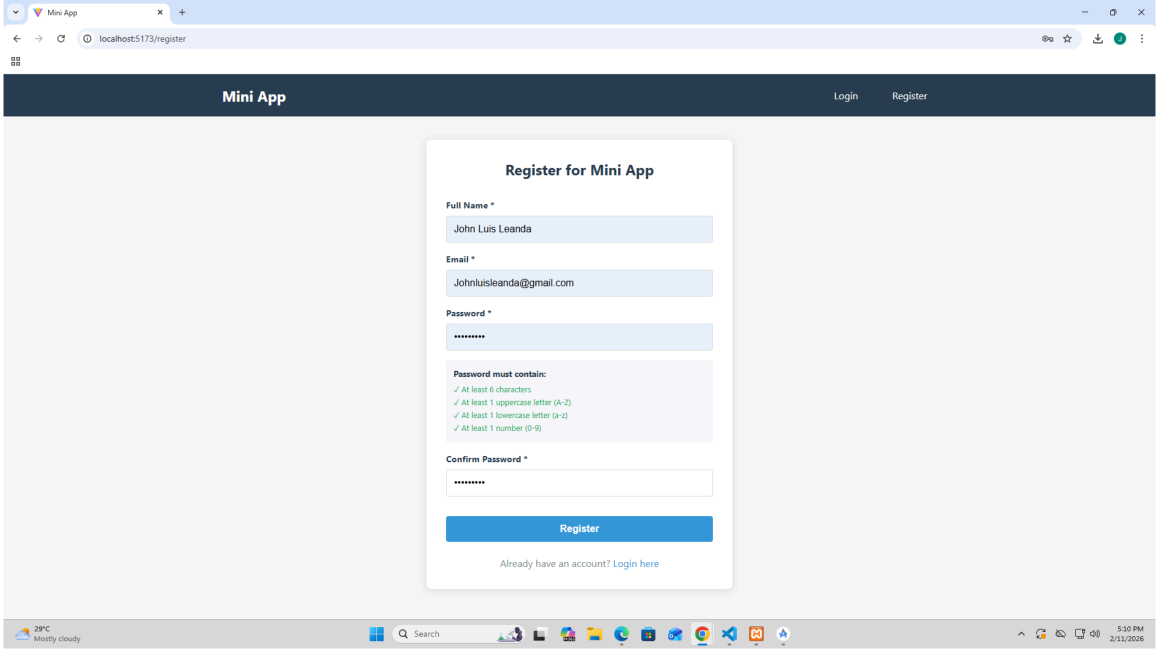



Access Protected Page (e.g. /profile)
Open dashboard/profile
GET /api/user/* (Authorization: Bearer <token>)
Check email uniqueness
alt
[Not authenticated]
401 Unauthorized
Redirect to login
[Authenticated]
SELECT user by email
use data
200 OK + user data
200 OK + user data
Logout
click logout
POST /api/auth/logout(optional)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

6. Appendices

Web Screenshots:

1. Register



Register for Mini App

Full Name *

John Luis Leanda

Email *

Johnluisleanda@gmail.com

Password *

•••••••••

Password must contain:

✓ At least 6 characters
✓ At least 1 uppercase letter (A-Z)
✓ At least 1 lowercase letter (a-z)
✓ At least 1 number (0-9)

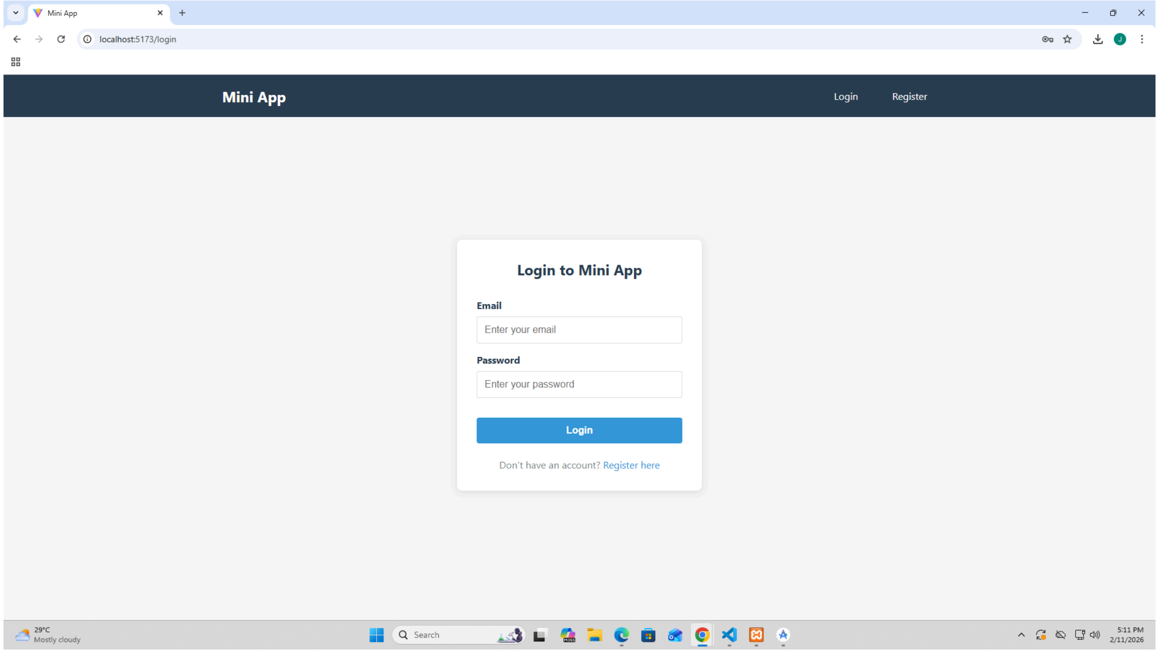
Confirm Password *

•••••••••

Register

Already have an account? Login here

2. Login



Login to Mini App

Email

Enter your email

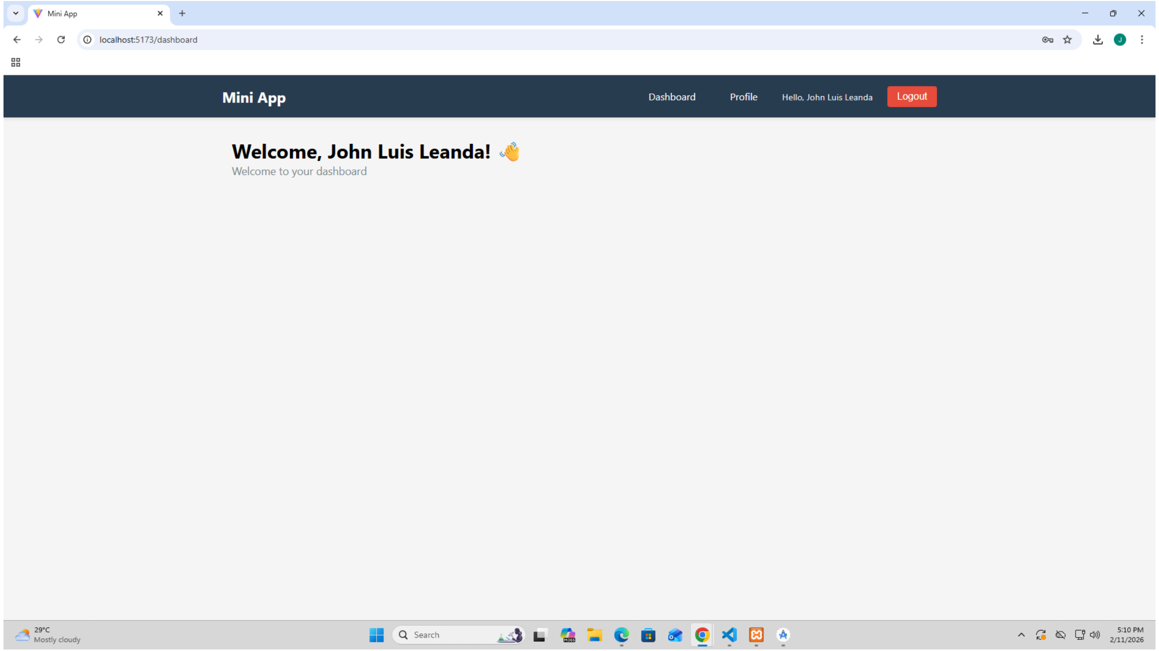
Password

Enter your password

Login

Don't have an account? Register here

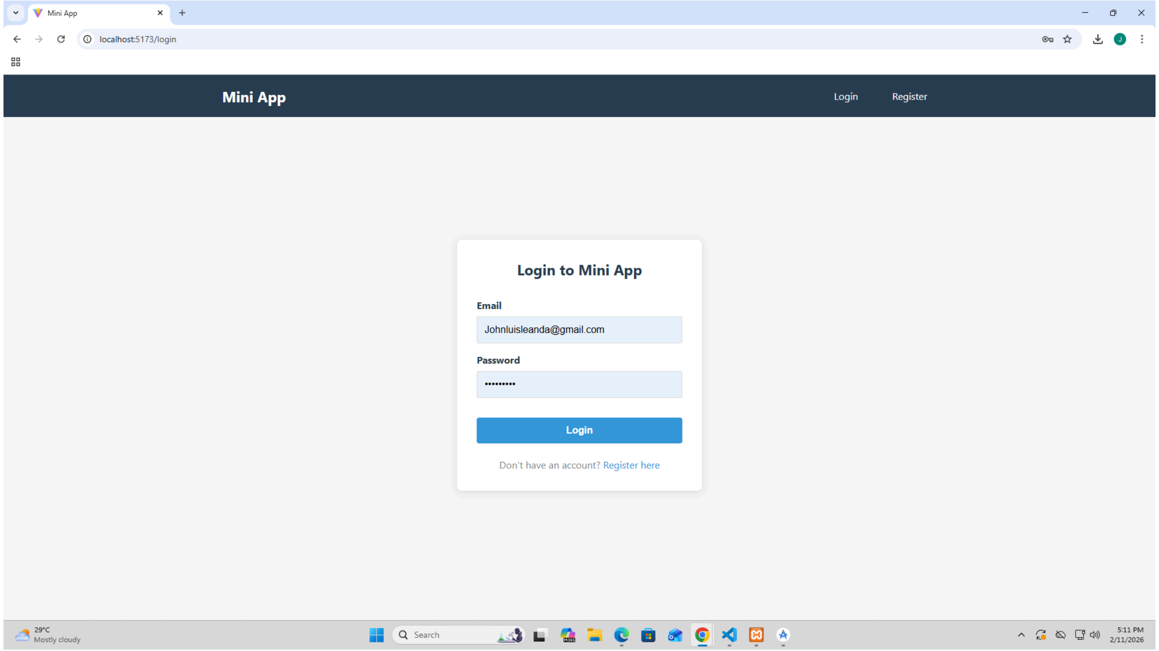
3. Dashboard



Welcome, John Luis Leanda! 👋

Welcome to your dashboard

4. Logout Result (Redirected to Login)



Login to Mini App

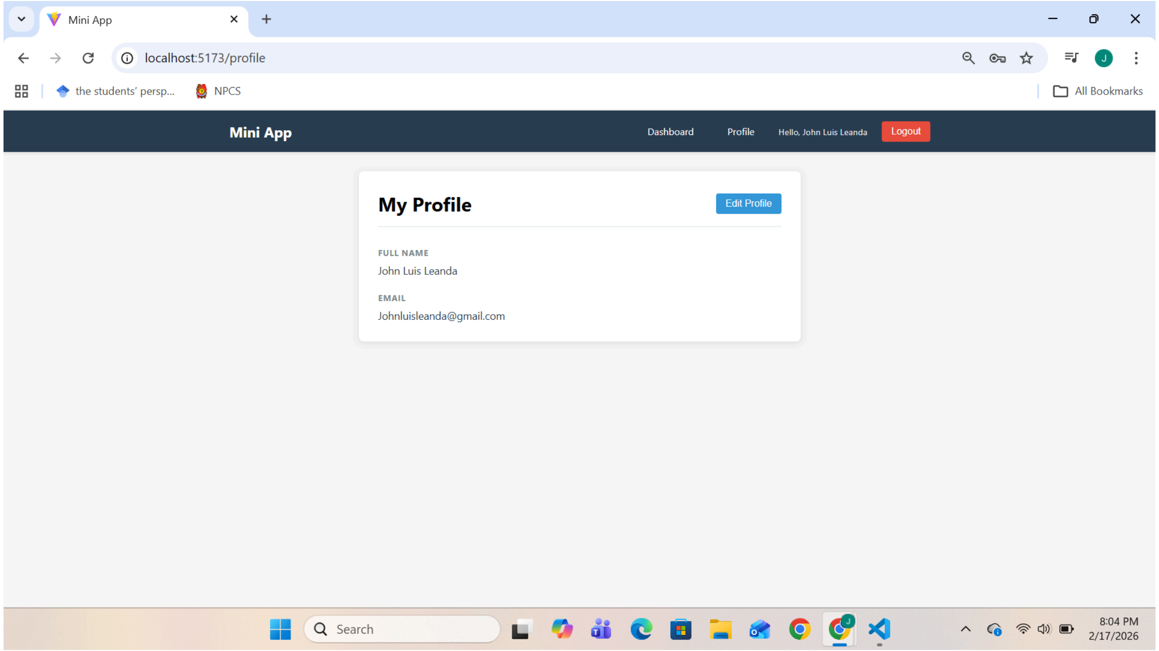
Email: Johnluisleanda@gmail.com

Password: •••••••••

Login

Don't have an account? Register here

5. Profile



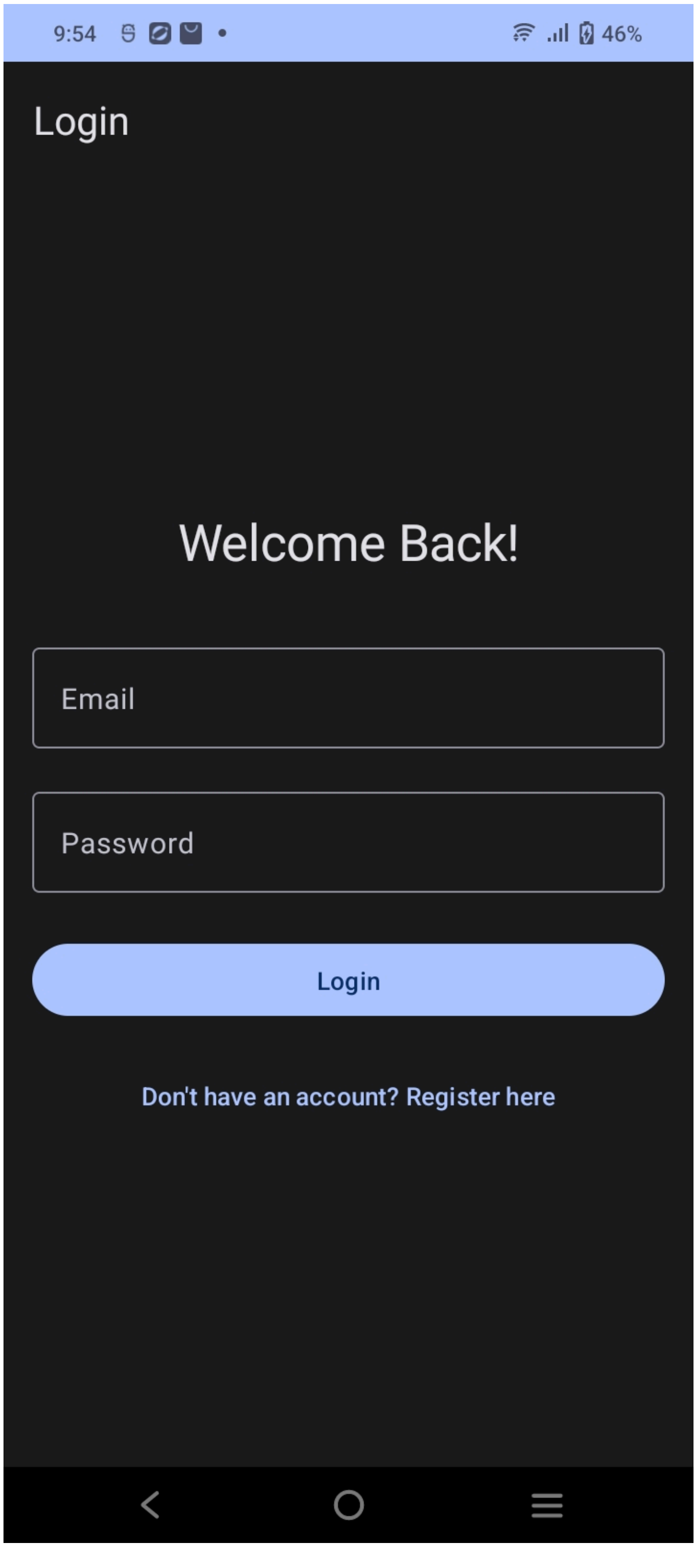
My Profile

FULL NAME
John Luis Leanda

EMAIL
Johnluisleanda@gmail.com

Mobile Screenshots

1. Login



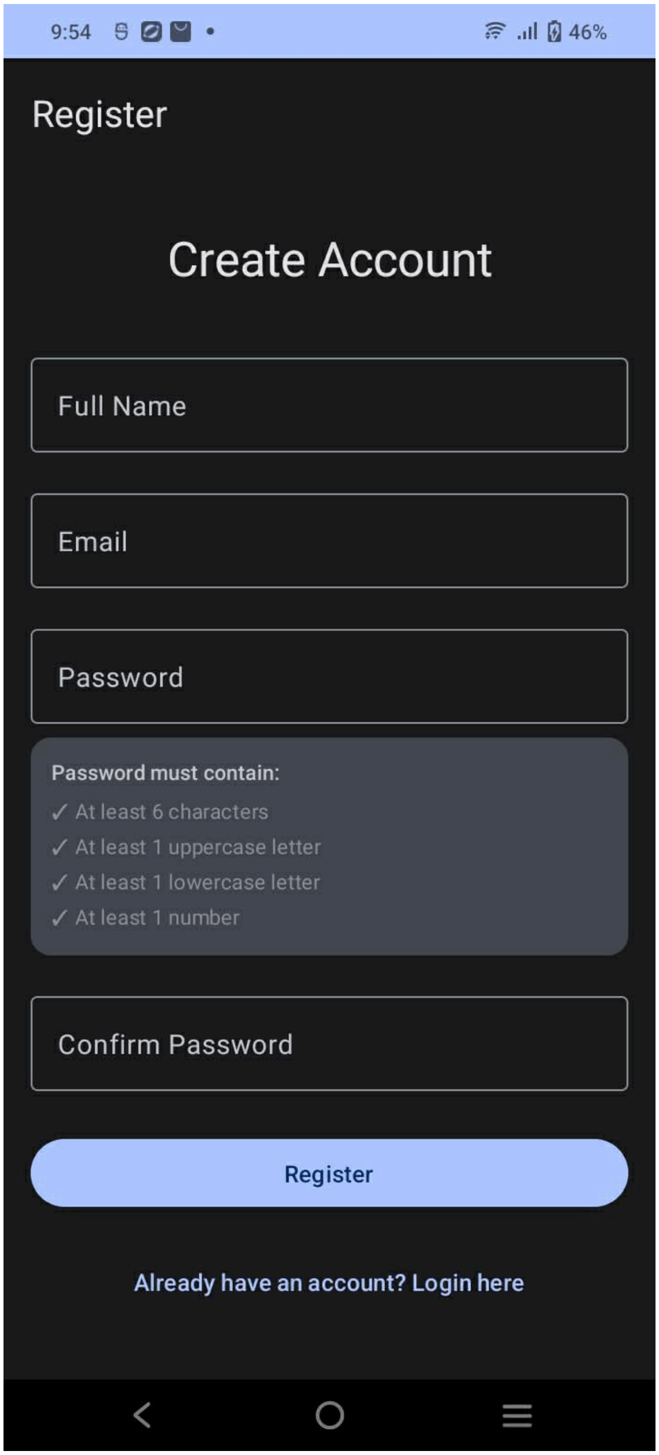
Welcome Back!

Email

Password

Don't have an account? Register here

2. Register



9:54 46%

Register

Create Account

Full Name

Email

Password

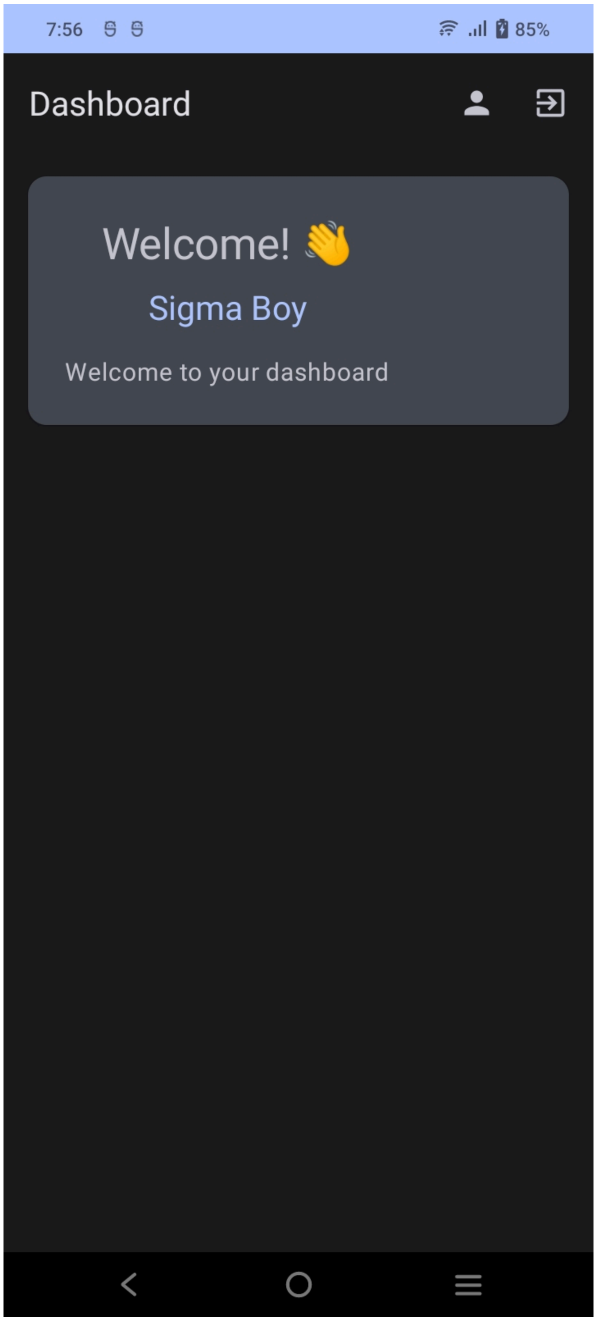
Password must contain:
✓ At least 6 characters
✓ At least 1 uppercase letter
✓ At least 1 lowercase letter
✓ At least 1 number

Confirm Password

Register

Already have an account? Login here

3. Dashboard



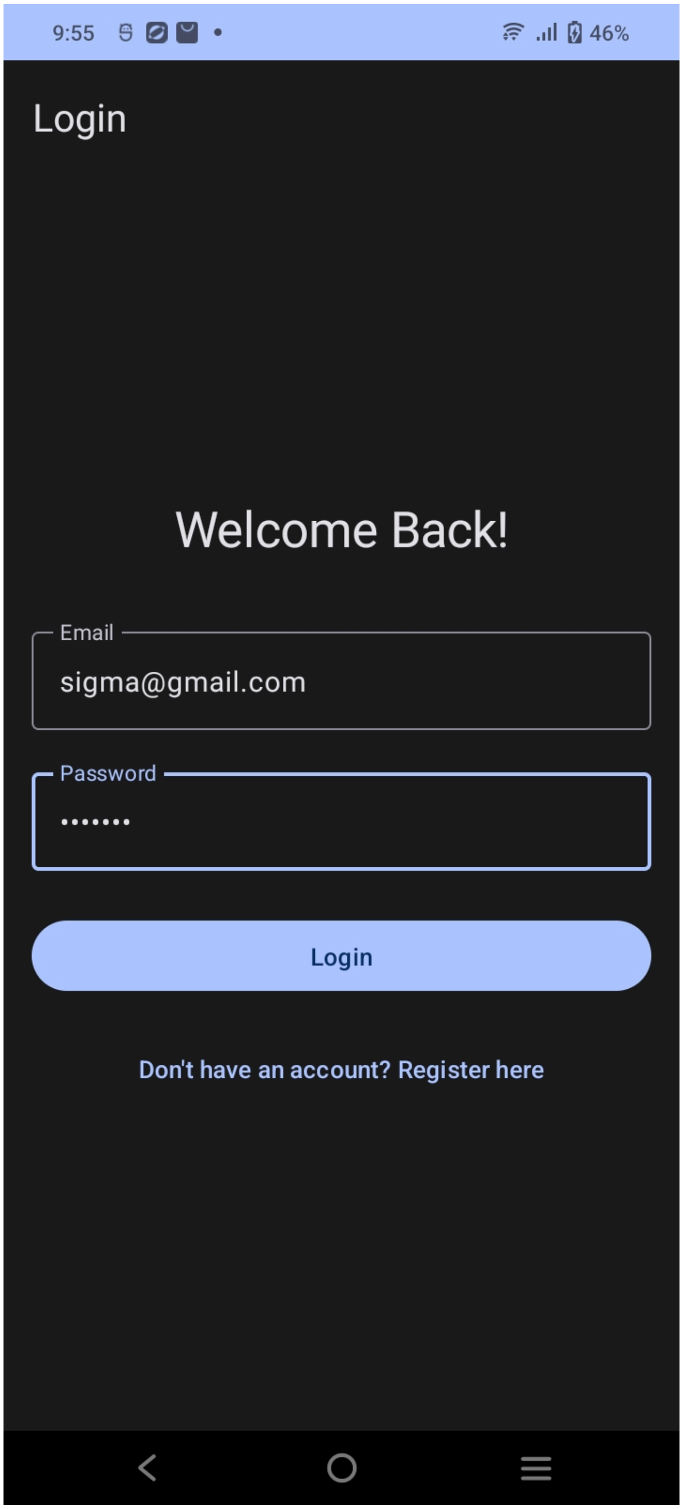
Dashboard

Welcome! 👋

Sigma Boy

Welcome to your dashboard

5. Logout result (login)



9:55 46%

Login

Welcome Back!

Email

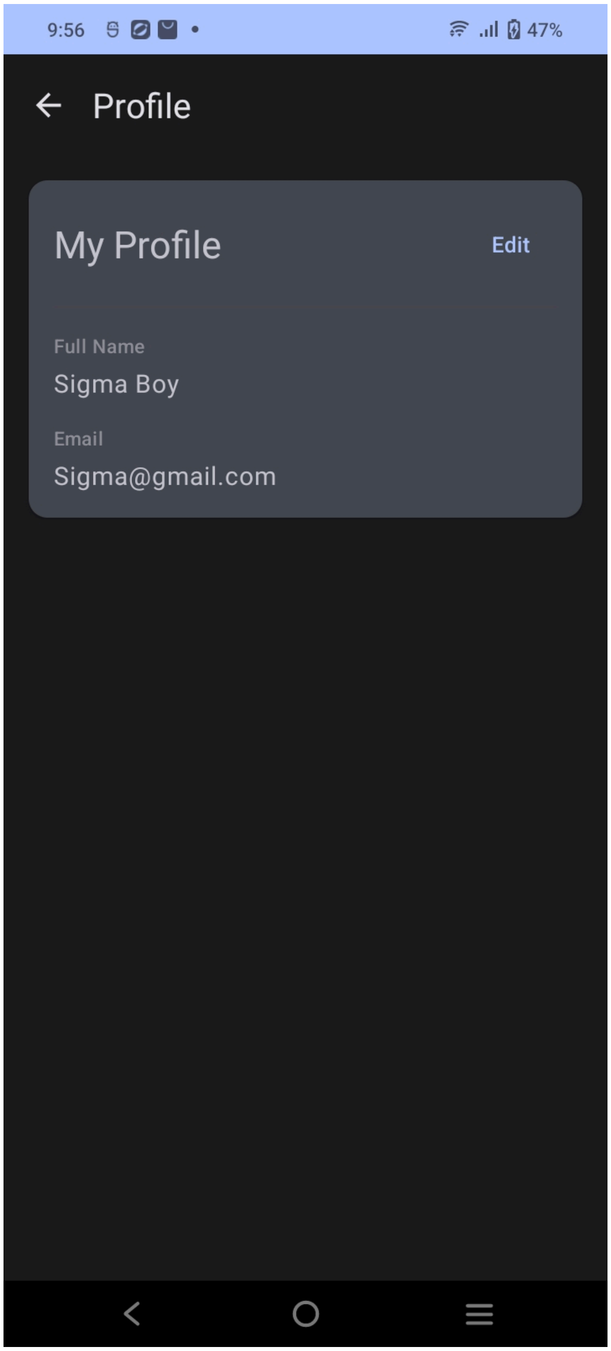
sigma@gmail.com

Password

•••••••

Don't have an account? Register here

5. Profile

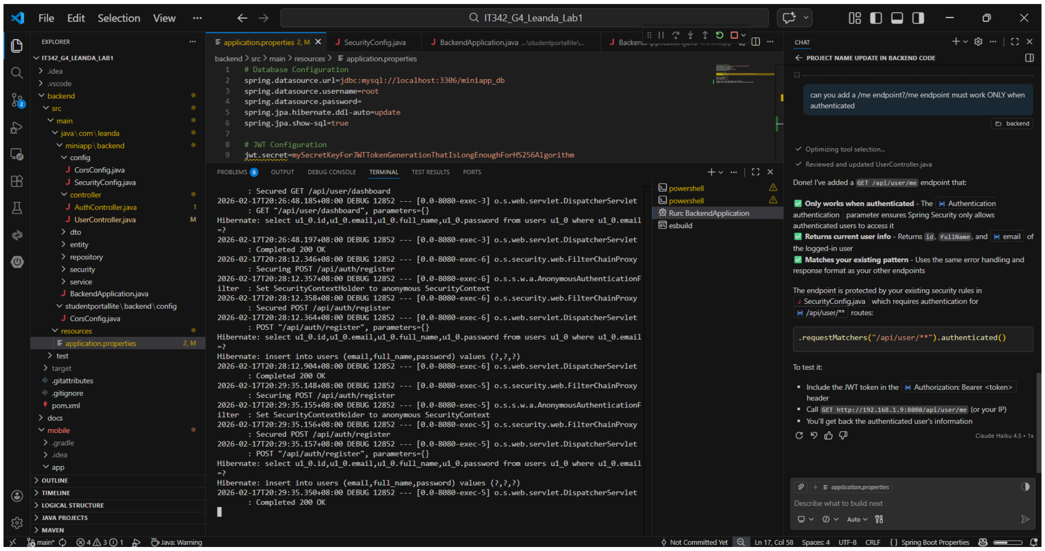


My Profile

Full Name
Sigma Boy

Email
Sigma@gmail.com

Proof of Integration



```
# Database Configuration
spring.datasource.url=jdbc:mysql://localhost:3306/miniapp_db
spring.datasource.username=root
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true

# JWT Configuration
jwt.secret=mySecretKeyForJWTTokenGenerationThatIsLongEnoughForHS256Algorithm
```

```
        : Secured GET /api/user/dashboard
2026-02-17T20:26:48.185+08:00 DEBUG 12852 --- [0.0-8080-exec-3] o.s.web.servlet.DispatcherServlet
        : GET "/api/user/dashboard", parameters={}
Hibernate: select u1_0.id,u1_0.email,u1_0.full_name,u1_0.password from users u1_0 where u1_0.email
=?
2026-02-17T20:26:48.197+08:00 DEBUG 12852 --- [0.0-8080-exec-3] o.s.web.servlet.DispatcherServlet
        : Completed 200 OK
2026-02-17T20:28:12.346+08:00 DEBUG 12852 --- [0.0-8080-exec-6] o.s.security.web.FilterChainProxy
        : Securing POST /api/auth/register
2026-02-17T20:28:12.357+08:00 DEBUG 12852 --- [0.0-8080-exec-6] o.s.s.w.a.AnonymousAuthenticationF
ilter   : Set SecurityContextHolder to anonymous SecurityContext
2026-02-17T20:28:12.358+08:00 DEBUG 12852 --- [0.0-8080-exec-6] o.s.security.web.FilterChainProxy
        : Secured POST /api/auth/register
2026-02-17T20:28:12.364+08:00 DEBUG 12852 --- [0.0-8080-exec-6] o.s.web.servlet.DispatcherServlet
        : POST "/api/auth/register", parameters={}
Hibernate: select u1_0.id,u1_0.email,u1_0.full_name,u1_0.password from users u1_0 where u1_0.email
=?
Hibernate: insert into users (email,full_name,password) values (?,?,?)
2026-02-17T20:28:12.904+08:00 DEBUG 12852 --- [0.0-8080-exec-6] o.s.web.servlet.DispatcherServlet
        : Completed 200 OK
2026-02-17T20:29:35.148+08:00 DEBUG 12852 --- [0.0-8080-exec-5] o.s.security.web.FilterChainProxy
        : Securing POST /api/auth/register
2026-02-17T20:29:35.155+08:00 DEBUG 12852 --- [0.0-8080-exec-5] o.s.s.w.a.AnonymousAuthenticationF
ilter   : Set SecurityContextHolder to anonymous SecurityContext
2026-02-17T20:29:35.156+08:00 DEBUG 12852 --- [0.0-8080-exec-5] o.s.security.web.FilterChainProxy
        : Secured POST /api/auth/register
2026-02-17T20:29:35.157+08:00 DEBUG 12852 --- [0.0-8080-exec-5] o.s.web.servlet.DispatcherServlet
        : POST "/api/auth/register", parameters={}
Hibernate: select u1_0.id,u1_0.email,u1_0.full_name,u1_0.password from users u1_0 where u1_0.email
=?
Hibernate: insert into users (email,full_name,password) values (?,?,?)
2026-02-17T20:29:35.350+08:00 DEBUG 12852 --- [0.0-8080-exec-5] o.s.web.servlet.DispatcherServlet
        : Completed 200 OK
```

can you add a /me endpoint?/me endpoint must work ONLY when authenticated

✓ Optimizing tool selection...

✓ Reviewed and updated UserController.java

Done! I've added a `GET /api/user/me` endpoint that:

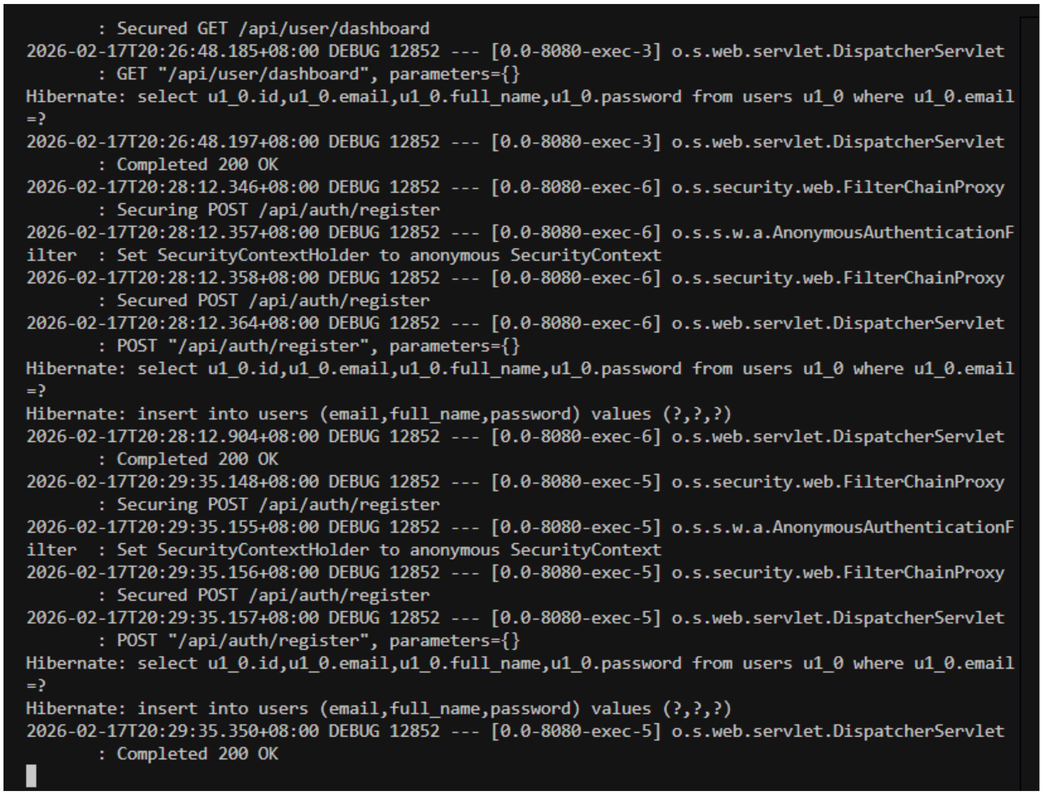
✅ **Only works when authenticated** - The Authentication authentication parameter ensures Spring Security only allows authenticated users to access it
✅ **Returns current user info** - Returns `id`, `fullName`, and email of the logged-in user
✅ **Matches your existing pattern** - Uses the same error handling and response format as your other endpoints

The endpoint is protected by your existing security rules in SecurityConfig.java which requires authentication for /api/user/** routes:

```
.requestMatchers("/api/user/**").authenticated()
```

To test it:

- Include the JWT token in the Authorization: Bearer <token> header
- Call `GET http://192.168.1.9:8080/api/user/me` (or your IP)
- You'll get back the authenticated user's information



```
        : Secured GET /api/user/dashboard
2026-02-17T20:26:48.185+08:00 DEBUG 12852 --- [0.0-8080-exec-3] o.s.web.servlet.DispatcherServlet
        : GET "/api/user/dashboard", parameters={}
Hibernate: select u1_0.id,u1_0.email,u1_0.full_name,u1_0.password from users u1_0 where u1_0.email
=?
2026-02-17T20:26:48.197+08:00 DEBUG 12852 --- [0.0-8080-exec-3] o.s.web.servlet.DispatcherServlet
        : Completed 200 OK
2026-02-17T20:28:12.346+08:00 DEBUG 12852 --- [0.0-8080-exec-6] o.s.security.web.FilterChainProxy
        : Securing POST /api/auth/register
2026-02-17T20:28:12.357+08:00 DEBUG 12852 --- [0.0-8080-exec-6] o.s.s.w.a.AnonymousAuthenticationF
ilter   : Set SecurityContextHolder to anonymous SecurityContext
2026-02-17T20:28:12.358+08:00 DEBUG 12852 --- [0.0-8080-exec-6] o.s.security.web.FilterChainProxy
        : Secured POST /api/auth/register
2026-02-17T20:28:12.364+08:00 DEBUG 12852 --- [0.0-8080-exec-6] o.s.web.servlet.DispatcherServlet
        : POST "/api/auth/register", parameters={}
Hibernate: select u1_0.id,u1_0.email,u1_0.full_name,u1_0.password from users u1_0 where u1_0.email
=?
Hibernate: insert into users (email,full_name,password) values (?,?,?)
2026-02-17T20:28:12.904+08:00 DEBUG 12852 --- [0.0-8080-exec-6] o.s.web.servlet.DispatcherServlet
        : Completed 200 OK
2026-02-17T20:29:35.148+08:00 DEBUG 12852 --- [0.0-8080-exec-5] o.s.security.web.FilterChainProxy
        : Securing POST /api/auth/register
2026-02-17T20:29:35.155+08:00 DEBUG 12852 --- [0.0-8080-exec-5] o.s.s.w.a.AnonymousAuthenticationF
ilter   : Set SecurityContextHolder to anonymous SecurityContext
2026-02-17T20:29:35.156+08:00 DEBUG 12852 --- [0.0-8080-exec-5] o.s.security.web.FilterChainProxy
        : Secured POST /api/auth/register
2026-02-17T20:29:35.157+08:00 DEBUG 12852 --- [0.0-8080-exec-5] o.s.web.servlet.DispatcherServlet
        : POST "/api/auth/register", parameters={}
Hibernate: select u1_0.id,u1_0.email,u1_0.full_name,u1_0.password from users u1_0 where u1_0.email
=?
Hibernate: insert into users (email,full_name,password) values (?,?,?)
2026-02-17T20:29:35.350+08:00 DEBUG 12852 --- [0.0-8080-exec-5] o.s.web.servlet.DispatcherServlet
        : Completed 200 OK
```

The above images are the logs of my backend. I tried to register one user in the web, and one user in the mobile app. The thread number **[0.0-8080-exec-6]** is the one from my web. While **[0.0-8080-exec-5]** is the one from the mobile app. They share the same backend.