CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342-Section SYSTEMS INTEGRATION AND ARCHITECTURE 1

FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

Project Title: User Registration & Authentication

Prepared By: Abelgas, Francis Carl D.

Date of Submission: February 8, 2026

Version: 1

Table of Contents

1. Introduction

1.1. Purpose

This document specifies requirements for the User Authentication System (Registration, Login, Logout). It serves as a blueprint for developers during implementation

1.2. Scope

Included:

- User registration with field validation (username, email, password, first name, last name)
- Secure login with JWT (JSON Web Token) authentication
- Profile and dashboard access for authenticated users
- Profile update functionality (username, first name, last name)
- Secure logout with token clearing
- Route protection to prevent unauthorized access
- Password hashing using BCrypt
- Session management using JWT tokens
- Web application (React frontend)
- Mobile application (Android native app)
- REST API backend (Spring Boot)
- MySQL database for user data storage

Excluded (Future Phases):

- Password recovery/reset functionality
- Email verification during registration
- Multi-factor authentication (MFA)
- Social login (Google, Facebook)
- Advanced role-based access control (RBAC) beyond basic authentication

1.3. Definitions, Acronyms, and Abbreviations

Term	Definition
JWT	JSON Web Token - A secure way to transmit information between parties as a JSON object
BCrypt	Password hashing algorithm used to securely store passwords
API	Application Programming Interface - Allows different software systems to communicate

REST	Representational State Transfer - An architectural style for designing web services
CRUD	Create, Read, Update, Delete - Basic database operations
Authentication	The process of verifying a user's identity
Authorization	The process of determining what an authenticated user can access
Session	A period of user activity on the system
Token	A piece of data used to verify user authentication
Route Protection	Preventing access to certain pages without authentication

2. Overall Description

2.1. System Perspective

The User Authentication System is a standalone web and mobile application that provides secure user registration and login functionality. It consists of three main components:

1. Frontend (Web): React-based web application that provides the user interface for registration, login, dashboard, and profile management
2. Frontend (Mobile): Android native application that provides the same functionality as the web app on mobile devices
3. Backend (API): Spring Boot REST API that handles business logic, authentication, and database operations
4. Database: MySQL database that stores user information securely

Both the web and mobile applications connect to the same Spring Boot backend API and share the same MySQL database. This ensures data consistency across all platforms.

2.2. User Classes and Characteristics

1. Guest User (Unauthenticated)

- Characteristics: Has not registered or logged in
- Needs: Create an account or login to existing account
- Technical Expertise: Basic (can fill forms)
- Access Level: Can only access registration and login pages

2. Authenticated User (Registered and Logged In)

- Characteristics: Has registered and successfully logged in
- Needs: View personal information, update profile, logout
- Technical Expertise: Basic
- Access Level: Can access dashboard, profile page, and logout functionality

2.3. Operating Environment

Web Application:

- Client-side: Modern web browsers (Chrome 90+, Firefox 88+, Safari 14+, Edge 90+)
- Frontend Framework: React 18.x
- HTTP Client: Axios

Mobile Application:

- Platform: Android 7.0 (API 24) or higher
- Development: Kotlin with XML layouts
- HTTP Client: Retrofit 2.9.0

Backend:

- Server: Spring Boot 3.2.0
- Java Version: 17
- Build Tool: Maven
- Port: 8080

Database:

- MySQL 8.0 or higher
- Database Name: user_auth_db
- Connection: JDBC

Network:

- Web: Standard HTTP/HTTPS connection to backend
- Mobile: HTTP connection via emulator (10.0.2.2:8080) or physical device IP

2.4. Assumptions and Dependencies

Assumptions:

1. Users have access to a stable internet connection
2. Users have a valid email address for registration
3. Backend server is running and accessible on port 8080

4. MySQL database is properly configured and running
5. Users understand basic form filling and navigation

Dependencies:

1. Spring Boot framework for backend API
2. React framework for web frontend
3. Android SDK for mobile app development
4. MySQL database server
5. JWT library for token generation and validation
6. BCrypt library for password hashing
7. Retrofit library for Android HTTP requests
8. Axios library for React HTTP requests

3. System Features and Functional Requirements

3.1. Feature 1: Username Validation

Description: New users can create an account by providing their personal information. The system validates all input fields, checks for duplicate emails, hashes the password, and stores the user information in the database.

Functional Requirements:

FR1.1: Username Validation

- The system shall require a username field
- The username must be at least 3 characters long
- The username must not exceed 50 characters
- The system shall display an error message if validation fails

FR1.2: Email Validation

- The system shall require a valid email address
- The email must match standard email format (user@domain.com)
- The system shall check if the email already exists in the database
- The system shall display an error message "Email already exists" if the email is already registered
- The system shall display an error message "Invalid email format" if format is incorrect

FR1.3: Password Validation

- The system shall require a password field

- The password must be at least 6 characters long
- The system shall display an error message if password is too short
- The system shall mask password characters with dots (••••••)

FR1.4: Password Hashing

- The system shall hash passwords using BCrypt algorithm before storing
- The system shall never store plain-text passwords in the database
- The hashed password shall be stored in the database

FR1.5: First Name and Last Name Validation

- The system shall require both first name and last name fields
- Both fields must not be empty
- The system shall display an error message if either field is empty

FR1.6: User Storage

- The system shall save validated user information to MySQL database
- The system shall automatically set user role as "USER"
- The system shall automatically set is_active status as true
- The system shall automatically generate created_at and updated_at timestamps
- The system shall assign a unique user_id to each new user

FR1.7: Registration Response

- The system shall return success message "User registered successfully" upon successful registration
- The system shall return user data (userId, email, username) in the response
- The system shall redirect user to login page after successful registration
- The system shall display appropriate error messages for any validation failures

FR1.8: Cross-Platform Availability

- Registration functionality shall be available on both web and mobile platforms
- Both platforms shall use the same backend API endpoint: POST /api/auth/register
- Both platforms shall have identical validation rules

3.2. Feature 2: User Login

Description: Registered users can login to the system by providing their email and password. The system validates credentials, generates a JWT token, and grants access to protected resources.

Functional Requirements:

FR2.1: Credential Validation

- The system shall require email and password for login
- The system shall validate email format before checking credentials
- The system shall verify password matches the stored hashed password
- The system shall display "User not found" error if email does not exist
- The system shall display "Invalid credentials" error if password is incorrect

FR2.2: JWT Token Generation

- The system shall generate a JWT token upon successful login
- The token shall contain user email as the subject
- The token shall be used for subsequent authenticated requests
- The token shall be returned in the login response

FR2.3: Last Login Update

- The system shall update the last_login timestamp in the database
- The timestamp shall reflect the current date and time of login
- This information shall be visible in the user profile

FR2.4: Login Response

- The system shall return success message "Login successful" upon successful authentication
- The system shall return user data including:
  - JWT token
  - userId
  - email
  - username
  - firstName
  - lastName
  - role
- The system shall store token in browser localStorage (web) or SharedPreferences (mobile)

FR2.5: Session Creation

- The system shall create a user session upon successful login
- The session shall remain active until user logs out or token expires
- The system shall use the JWT token to maintain session state

FR2.6: Navigation After Login

- The system shall redirect user to dashboard page after successful login
- The system shall remain on login page and display errors if login fails

FR2.7: Cross-Platform Availability

- Login functionality shall be available on both web and mobile platforms
- Both platforms shall use the same backend API endpoint: POST /api/auth/login
- Both platforms shall store JWT token securely

3.3. Feature 3: Dashboard/Profile Access

Description:
Authenticated users can view their personal information on the dashboard and profile pages. The system verifies the JWT token before allowing access and displays user data retrieved from the database.

Functional Requirements:

FR3.1: Token Verification

- The system shall verify JWT token before granting access to dashboard or profile
- The system shall check if token is present in Authorization header
- The system shall validate token signature and expiration
- The system shall redirect to login page if token is missing or invalid

FR3.2: Profile Data Retrieval

- The system shall fetch user profile data from database using the token
- The system shall use the endpoint: GET /api/user/me
- The system shall include JWT token in Authorization header: "Bearer {token}"
- The system shall retrieve user information including:
  - userId
  - username
  - email
  - firstName
  - lastName
  - role
  - created_at
  - last_login

FR3.3: Dashboard Display

- The system shall display a welcome message with user's first name
- The dashboard shall show: "Welcome, {firstName}"
- The dashboard shall be centered on the page
- The dashboard shall provide navigation to profile page
- The dashboard shall provide logout functionality

FR3.4: Profile Display

- The system shall display user information in a card layout
- The profile shall show an avatar with user's first name initial
- The profile shall display:
  - Full name (firstName + lastName)
  - Username
  - Email
  - First name
  - Last name
  - Role (displayed as a badge)
- The profile shall provide navigation back to dashboard

FR3.5: Route Protection

- The system shall protect dashboard and profile routes from unauthorized access
- The system shall automatically redirect unauthenticated users to login page
- Web app shall use React ProtectedRoute component
- Mobile app shall check token in TokenManager before rendering screens

FR3.6: Cross-Platform Availability

- Dashboard and profile functionality shall be available on both web and mobile platforms
- Both platforms shall display identical user information
- Both platforms shall use the same backend API endpoint

3.4. Feature 4: User Logout

Description:
Authenticated users can logout from the system, ending their session and clearing authentication data from the client.

Functional Requirements:

FR4.1: Logout Initiation

- The system shall provide a logout button on the dashboard page (web and mobile)
- The logout button shall be easily accessible and clearly labeled
- Web: Logout button located at bottom right of dashboard, left of "View Profile"
- Mobile: Logout button located at bottom right of dashboard, left of "View Profile"

FR4.2: Session Termination

- The system shall call the backend logout endpoint: POST /api/auth/logout
- The system shall include JWT token in the Authorization header
- The system shall clear JWT token from client-side storage:
  - Web: Remove token from localStorage

    - Mobile: Remove token from SharedPreferences
- The system shall clear all user data from client-side storage

FR4.3: Logout Response

- The system shall return success message "Logout successful"
- The system shall confirm session has been terminated

FR4.4: Post-Logout Navigation

- The system shall redirect user to login page after logout
- The system shall prevent accessing protected pages after logout
- Any attempt to access dashboard or profile shall redirect to login

FR4.5: Cross-Platform Availability

- Logout functionality shall be available on both web and mobile platforms
- Both platforms shall use the same backend API endpoint
- Both platforms shall clear tokens using their respective storage mechanisms

3.5. Feature 5: Profile Update

Description:
Authenticated users can update their profile information (username, first name, last name). The system validates the updated information and saves changes to the database.

Functional Requirements:

FR5.1: Update Access

- The system shall provide an "Edit Profile" button on the web profile page
- The button shall only be visible to authenticated users
- Mobile app: Profile update feature is available for future implementation

FR5.2: Editable Fields

- The system shall allow users to update:
    - Username
    - First Name
    - Last Name
- The system shall NOT allow editing of:
    - Email (fixed)
    - User ID (fixed)
    - Role (fixed)
    - Timestamps (automatic)

FR5.3: Update Validation

- The system shall validate username (minimum 3 characters)
- The system shall validate first name and last name (required fields)
- The system shall display error messages for invalid inputs
- The system shall require JWT token for authorization

FR5.4: Update Processing

- The system shall use endpoint: PUT /api/user/profile
- The system shall include JWT token in Authorization header
- The system shall send updated data:
    - userId
    - username
    - firstName
    - lastName
- The system shall update the database with new values

FR5.5: Update Response

- The system shall display success message "Profile updated successfully"
- The system shall refresh profile display with updated information
- The system shall return to profile view mode after successful update
- The system shall display error messages if update fails

FR5.6: Platform Availability

- Profile update is currently implemented on web platform only
- Mobile platform displays profile information (read-only)
- Both platforms use the same backend API endpoint

4. Non-Functional Requirements

4.1. Performance Requirements

NFR1: Response Time

- The system shall respond to user requests within 2 seconds under normal load
- API endpoints shall process requests within 1 second
- Database queries shall execute within 500 milliseconds
- Page loading shall complete within 3 seconds

NFR2: Throughput

- The system shall support at least 100 concurrent users
- The system shall handle at least 1000 API requests per minute

- The database shall support multiple simultaneous connections

4.2. Security Requirements

NFR3: Password Security

- The system shall hash all passwords using BCrypt algorithm
- The system shall never store passwords in plain text
- The system shall never display or transmit passwords in plain text
- Password hashing shall use a minimum work factor of 10

NFR4: Authentication Security

- The system shall use JWT tokens for authentication
- The system shall include token expiration time
- The system shall validate token signature on each request
- The system shall use HTTPS for all API communication in production

NFR5: Data Protection

- The system shall validate all user inputs to prevent SQL injection
- The system shall sanitize user inputs to prevent XSS attacks
- The system shall use parameterized queries for database operations
- The system shall protect sensitive API endpoints with JWT authentication

4.3. Usability Requirements

NFR6: User Interface

- The mobile interface shall match the web interface design
- The system shall use consistent colors across platforms:
    - Primary gradient: `#667eea` to `#764ba2`
    - Background: `#f9fafb`
    - Text: `#111827`
    - Borders: `#d1d5db`
    - Error messages: `#DC2626`
- The system shall provide clear error messages for validation failures
- The system shall provide visual feedback during loading states

NFR7: Navigation

- The system shall provide intuitive navigation between pages
- The system shall clearly indicate the current page
- The system shall provide easy access to logout functionality

- The system shall redirect appropriately after actions (login → dashboard, logout → login)

NFR8: Form Design

- Input fields shall have clear labels
- Input fields shall have placeholder text
- Required fields shall be clearly marked
- Error messages shall appear below the relevant input field

4.4. Compatibility Requirements

NFR9: Web Browser Support

- The web application shall work on Google Chrome 90+
- The web application shall work on Mozilla Firefox 88+
- The web application shall work on Safari 14+
- The web application shall work on Microsoft Edge 90+

NFR10: Mobile Platform Support

- The mobile application shall run on Android 7.0 (API 24) or higher
- The mobile application shall work on both emulator and physical devices
- The mobile application shall support screen sizes from 4.7" to 6.8"

NFR11: Backend Compatibility

- The backend shall run on Java 17 or higher
- The backend shall be compatible with Spring Boot 3.2.0
- The backend shall connect to MySQL 8.0 or higher

4.5. Reliability Requirements

NFR12: Availability

- The system shall have 99% uptime during business hours
- The system shall handle errors gracefully without crashing
- The system shall provide meaningful error messages to users
- The system shall log errors for debugging purposes

NFR13: Data Integrity

- The system shall ensure email uniqueness in the database
- The system shall prevent duplicate user registrations
- The system shall maintain data consistency across all platforms
- The system shall use database constraints to enforce data rules

4.6. Maintainability Requirements

NFR14: Code Quality

- The code shall follow industry-standard naming conventions
- The code shall be well-commented for complex logic
- The code shall be modular and follow separation of concerns
- The code shall use consistent formatting

NFR15: API Documentation

- All API endpoints shall be documented with purpose and parameters
- Request and response formats shall be clearly specified
- Error codes and messages shall be documented

4.7. Scalability Requirements

NFR16: Database Scalability

- The database shall support growth to 10,000 users minimum
- Database indexes shall be used for frequently queried fields (email, username)

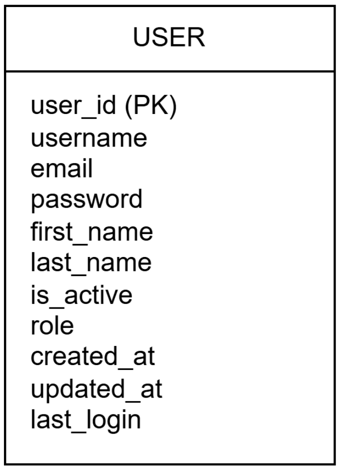
NFR17: Platform Scalability

- The system architecture shall support adding new frontend platforms
- The API shall be platform-agnostic
- The system shall support future feature additions without major refactoring

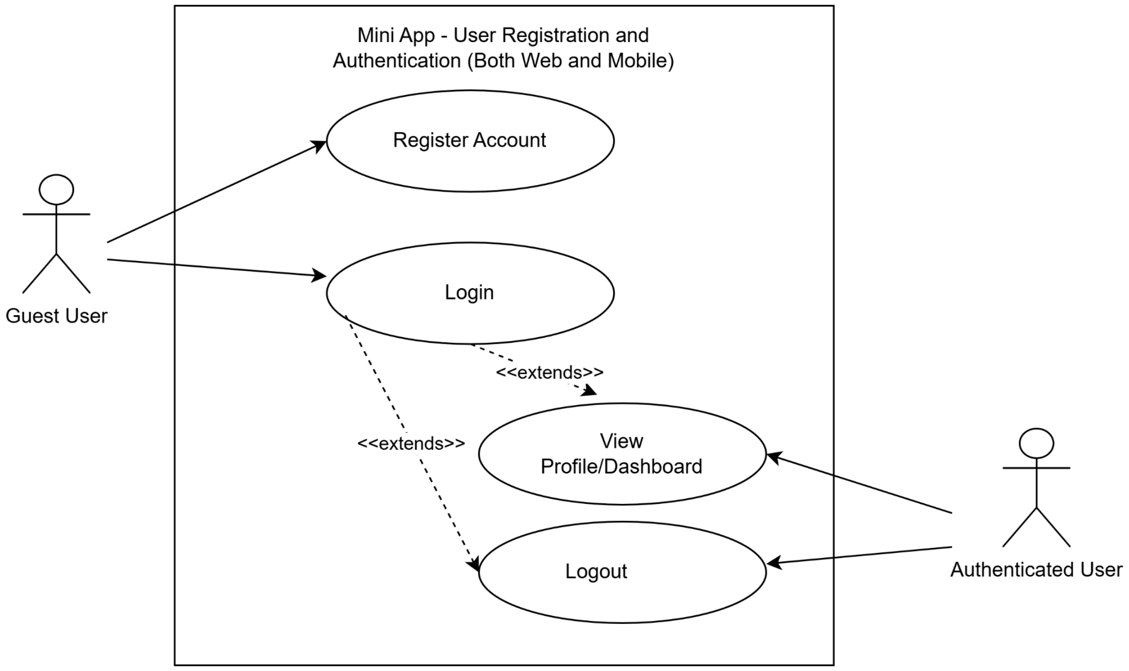
5. System Models (Diagrams)

Insert the necessary diagrams for the system:

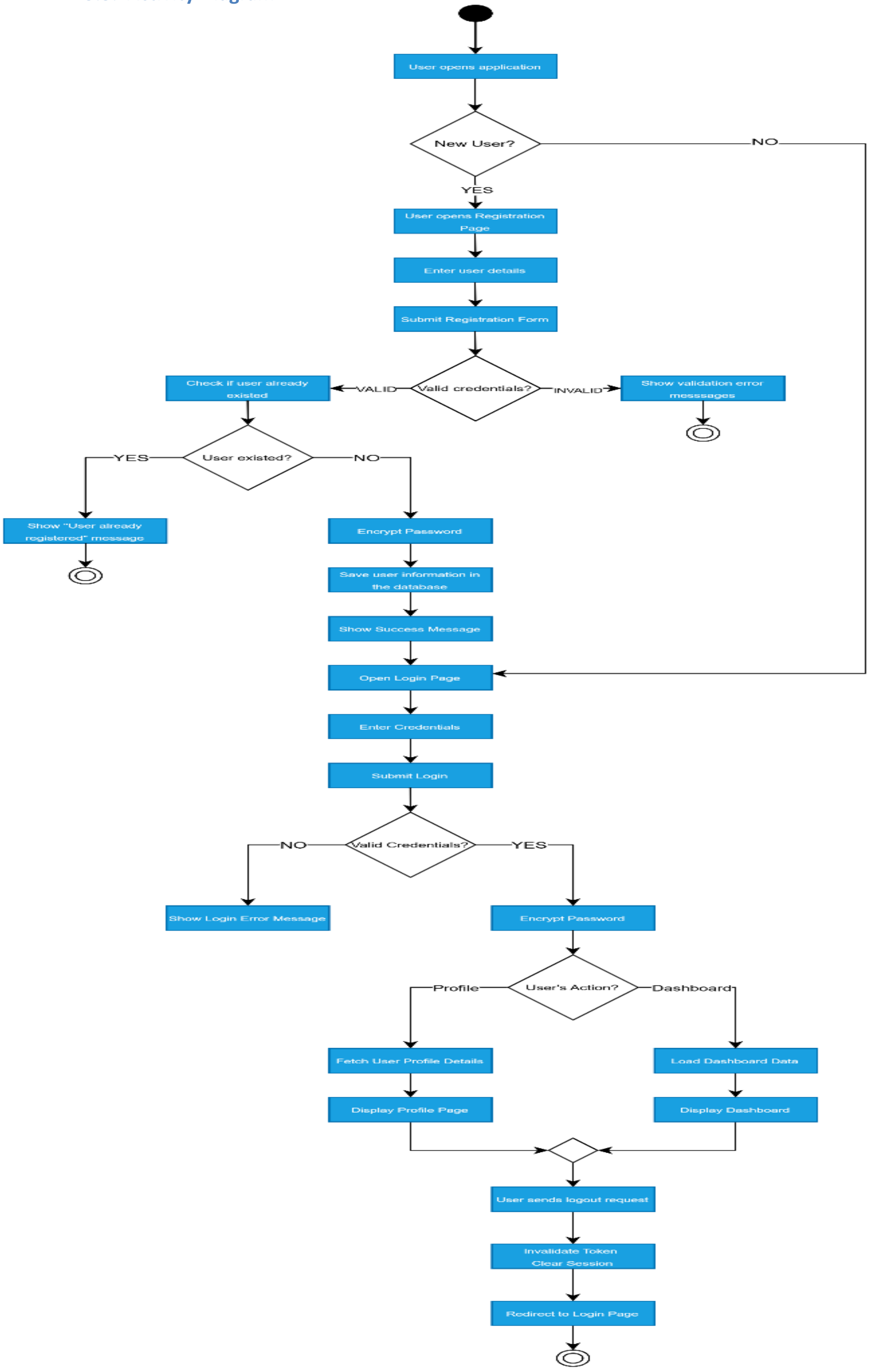
5.1. ERD



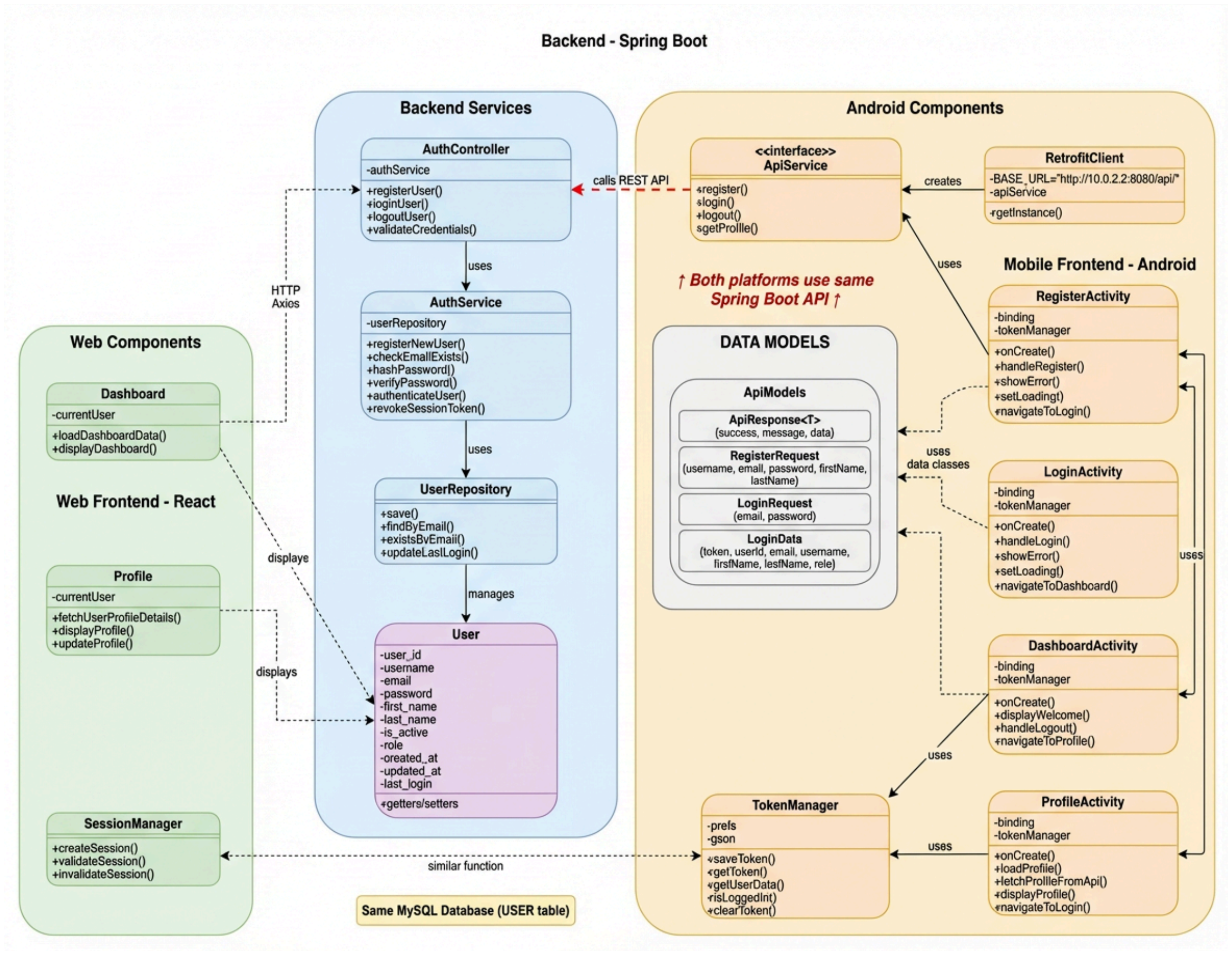
5.2. Use Case Diagram



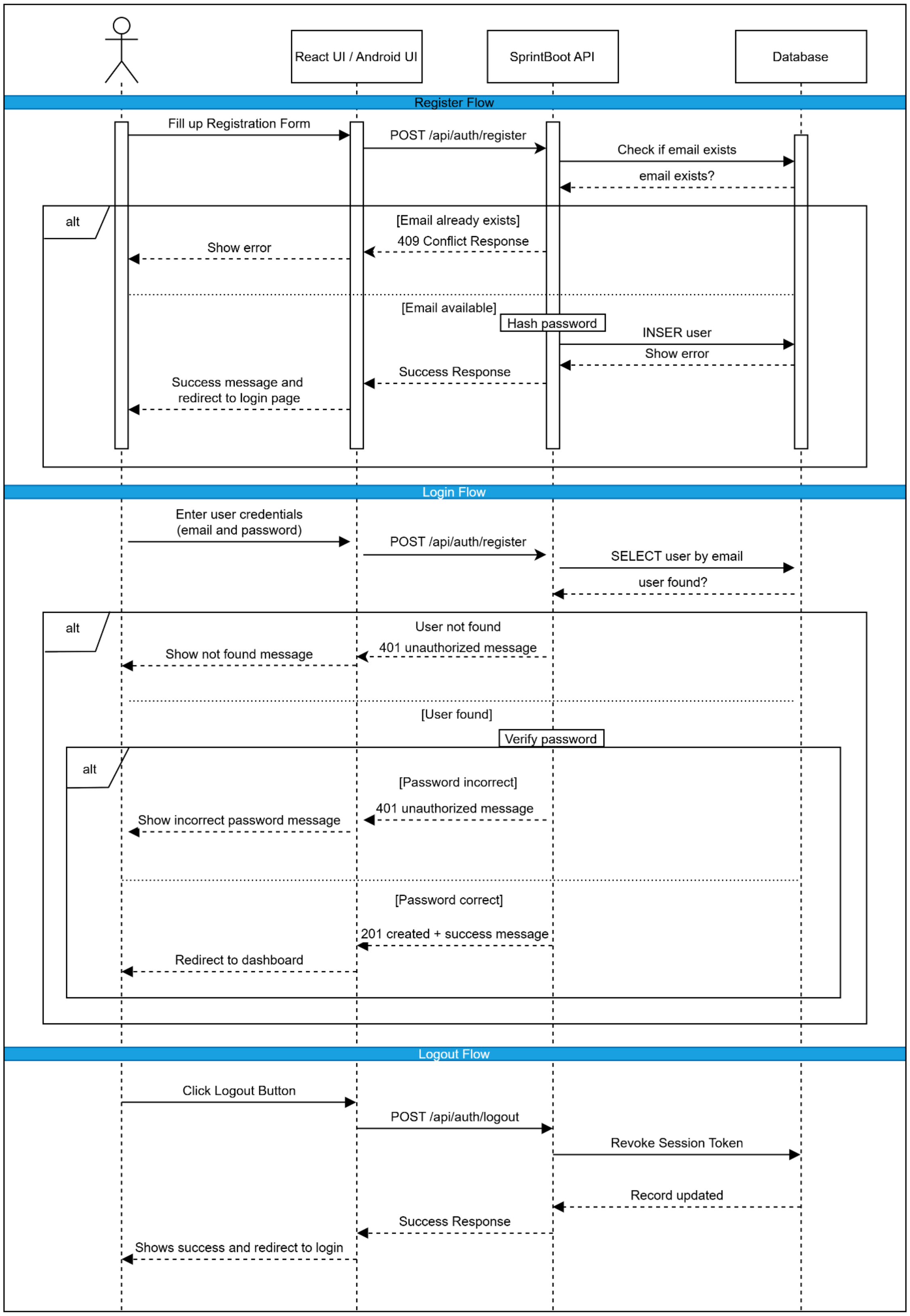
5.3. Activity Diagram



5.4. Class Diagram

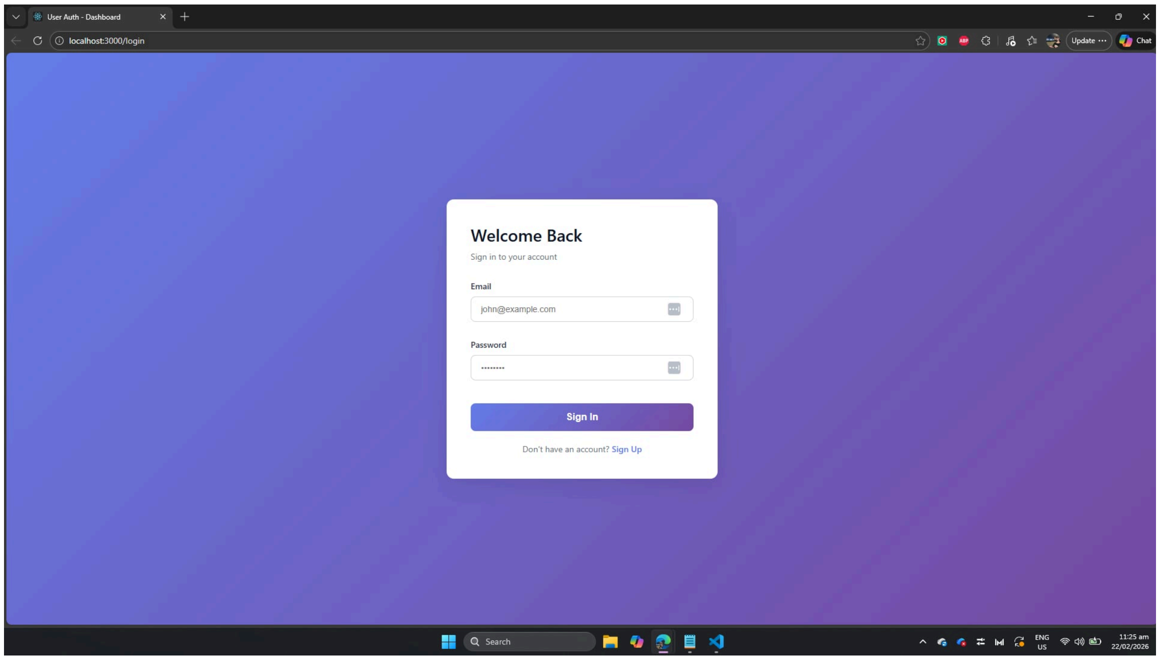


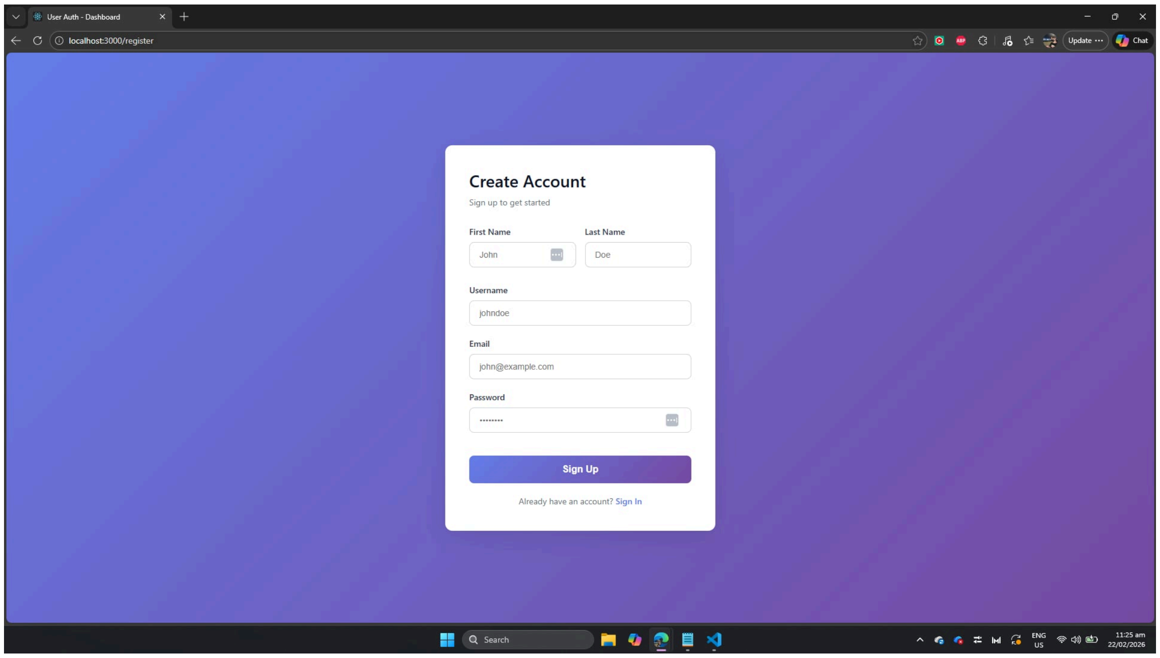
5.5. Sequence Diagram

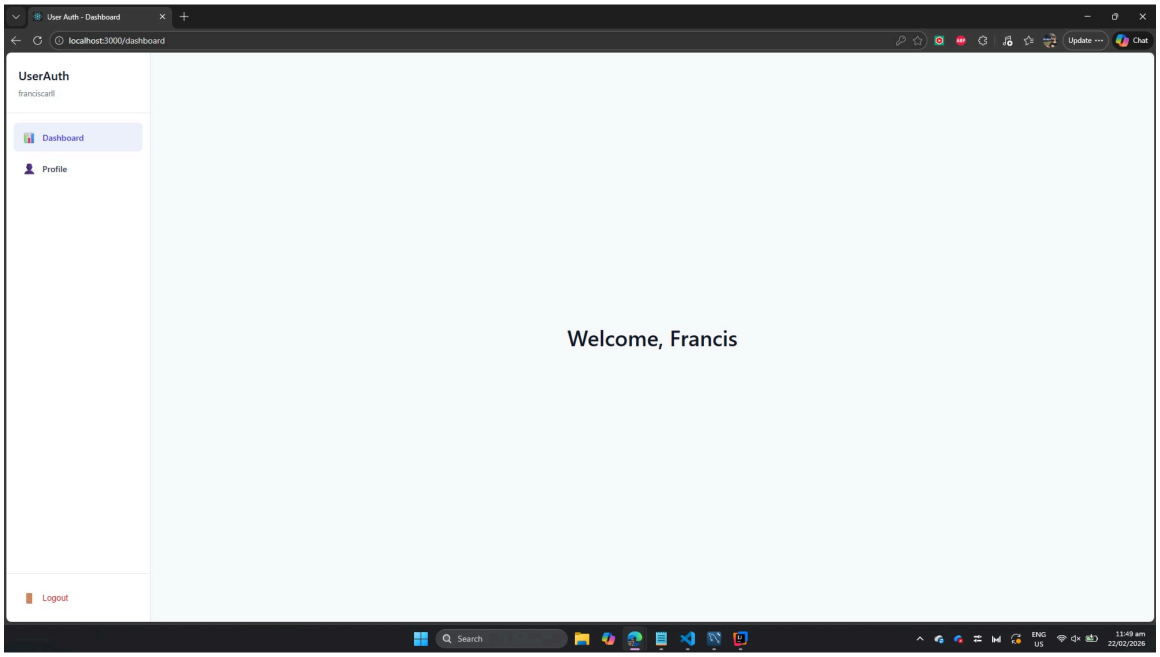


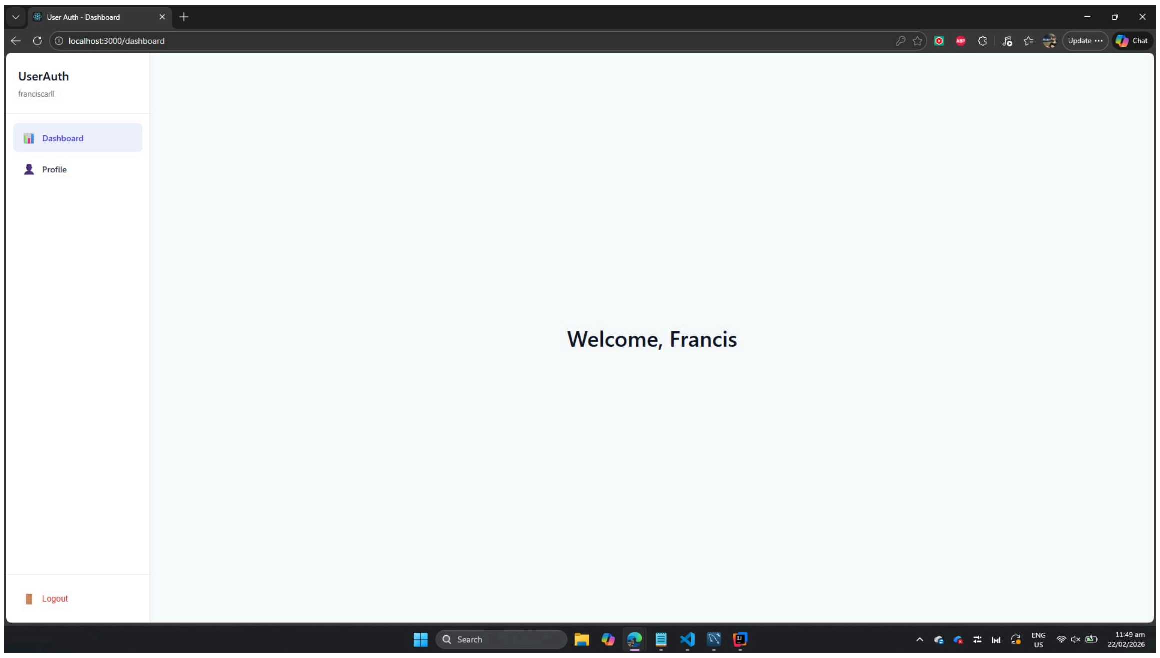
6. Appendices

Appendix A: Web UI

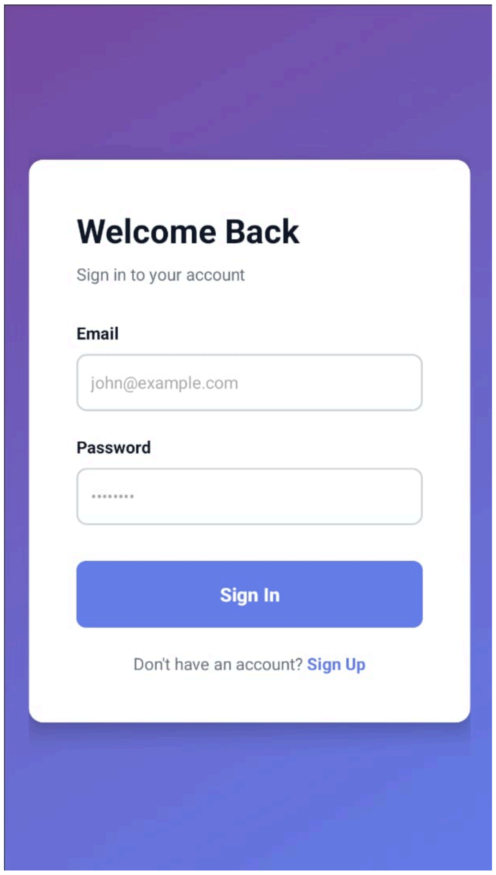





User Auth - Dashboard
localhost:3000/dashboard
UserAuth
franciscarll
Dashboard
Profile
Logout
Welcome, Francis


User Auth - Dashboard
localhost:3000/dashboard
UserAuth
franciscarll
Dashboard
Profile
Logout
Welcome, Francis

Appendix B: Mobile UI



Welcome Back

Sign in to your account

Email

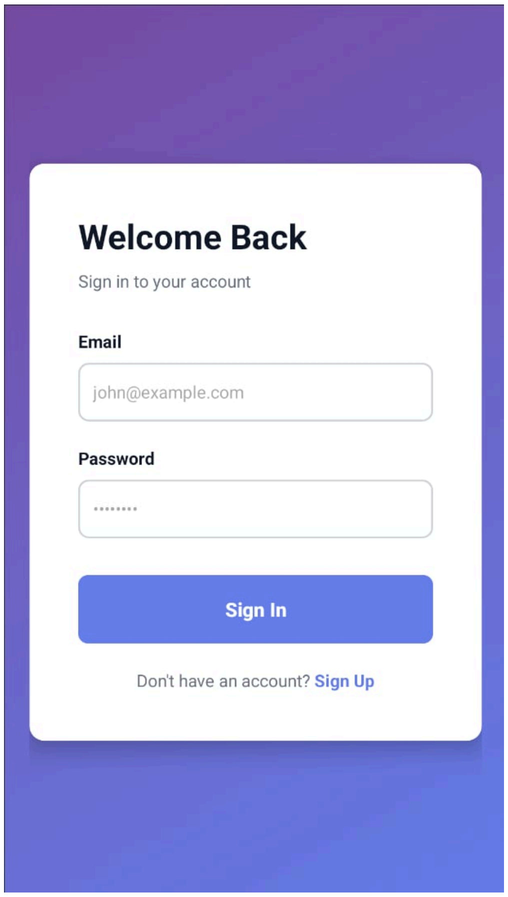
john@example.com

Password

••••••••

Sign In

Don't have an account? Sign Up



Welcome Back

Sign in to your account

Email

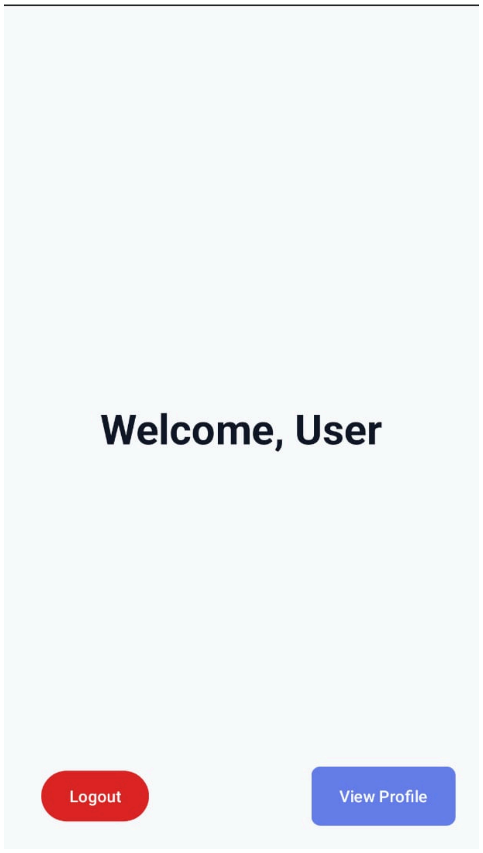
john@example.com

Password

••••••••

Sign In

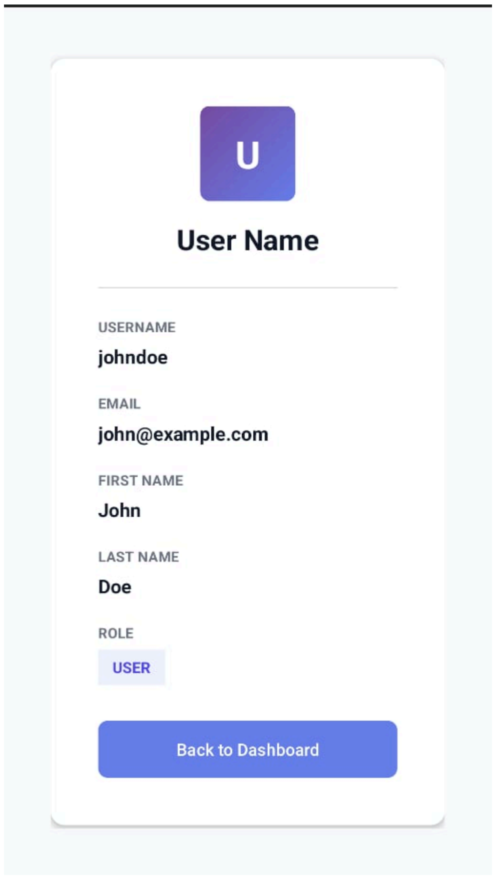
Don't have an account? Sign Up



Welcome, User

Logout

View Profile



U

User Name

USERNAME

johndoe

EMAIL

john@example.com

FIRST NAME

John

LAST NAME

Doe

ROLE

USER

Back to Dashboard