CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342-G5 SYSTEMS INTEGRATION AND ARCHITECTURE 1

FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

Project Title: Mini App - User Registration & Authentication

Prepared By: Trisha Raye V. Cararag

Date of Submission: February 6, 2026

Version: 01

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this system is to design and document the user authentication flow for a mini application. This includes user registration, login, profile/dashboard access, and logout, with proper access control to protected pages.

1.2. Scope

In Scope:

The system will:

- Allow a guest user to register an account
- Allow a registered user to log in
- Allow authenticated users to view protected pages (profile/dashboard)
- Prevent unauthenticated users from accessing protected pages
- Allow authenticated users to log out

Out of Scope:

The system does not include:

- Actual deployment

1.3. Definitions, Acronyms, and Abbreviations

List and define important terms used in this document.

2. Overall Description

2.1. System Perspective

The mini application is a user authentication module that operates as part of a larger mini application.
It follows a client–server architecture, where:

- The frontend (React) handles user interaction.
- The backend (Spring Boot) processes authentication logic.
- The database stores user credentials and profile information.

2.2. User Classes and Characteristics

1. Guest User

- Has not logged in
- Can register a new account
- Can attempt to log in

2. Authenticated User

- Has successfully logged in
- Can access protected pages (profile/dashboard)
- Can log out

2.3. Operating Environment

- Frontend: React.js
- Backend: Spring Boot (Java)

2.4. Assumptions and Dependencies

List any assumptions and external dependencies that may affect the system.

3. System Features and Functional Requirements

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

3.1. Feature 1: User Registration

Description: Allows a guest user to create a new account by providing required personal and login information.

Functional Requirements:

- The system shall allow users to register using a unique username and email.
- The system shall validate user input before account creation.
- The system shall store passwords in encrypted form.
- The system shall prevent duplicate user accounts.

3.2. Feature 2: User Login

Description: Allows registered users to authenticate and access protected pages.

Functional Requirements:

- The system shall allow users to log in using valid credentials.
- The system shall verify credentials against stored data.
- The system shall deny access for invalid credentials.

3.3. Feature 3: View Profile / Dashboard

Description: Allows authenticated users to view their personal information and dashboard.

Functional Requirements:

- The system shall restrict access to authenticated users only.

- The system shall display user-specific information.
- The system shall redirect unauthenticated users to the login page.

3.4. Feature 4: Logout

Description: Allows authenticated users to end their session securely.

Functional Requirements:

- The system shall allow users to log out at any time.
- The system shall redirect users to the login page after logout.

4. Non-Functional Requirements

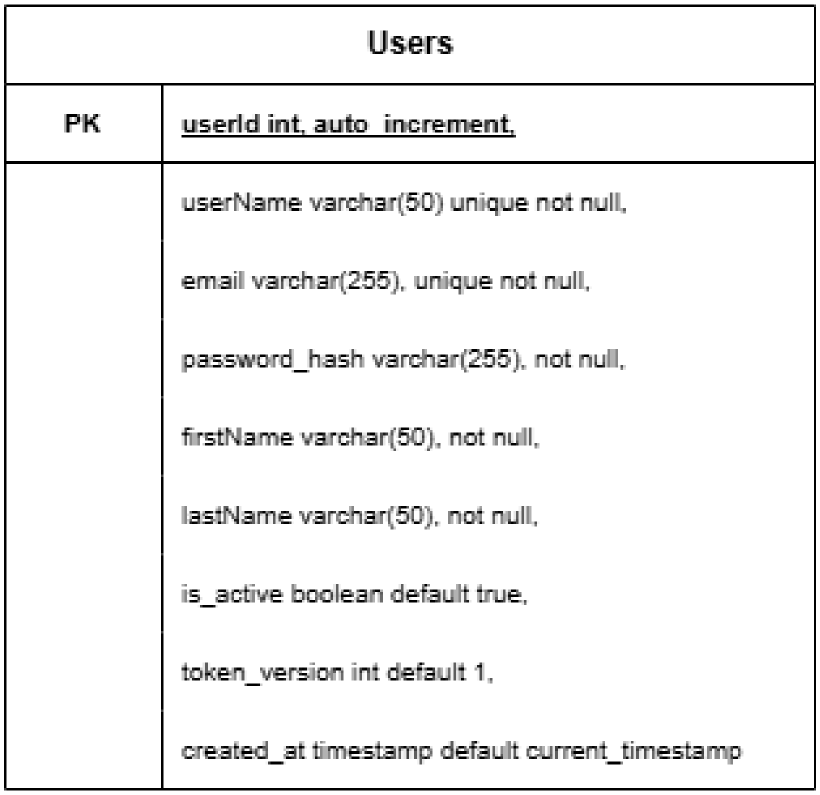
Specify system quality attributes such as performance, security, usability, reliability, etc.

5. System Models (Diagrams)

Insert the necessary diagrams for the system:

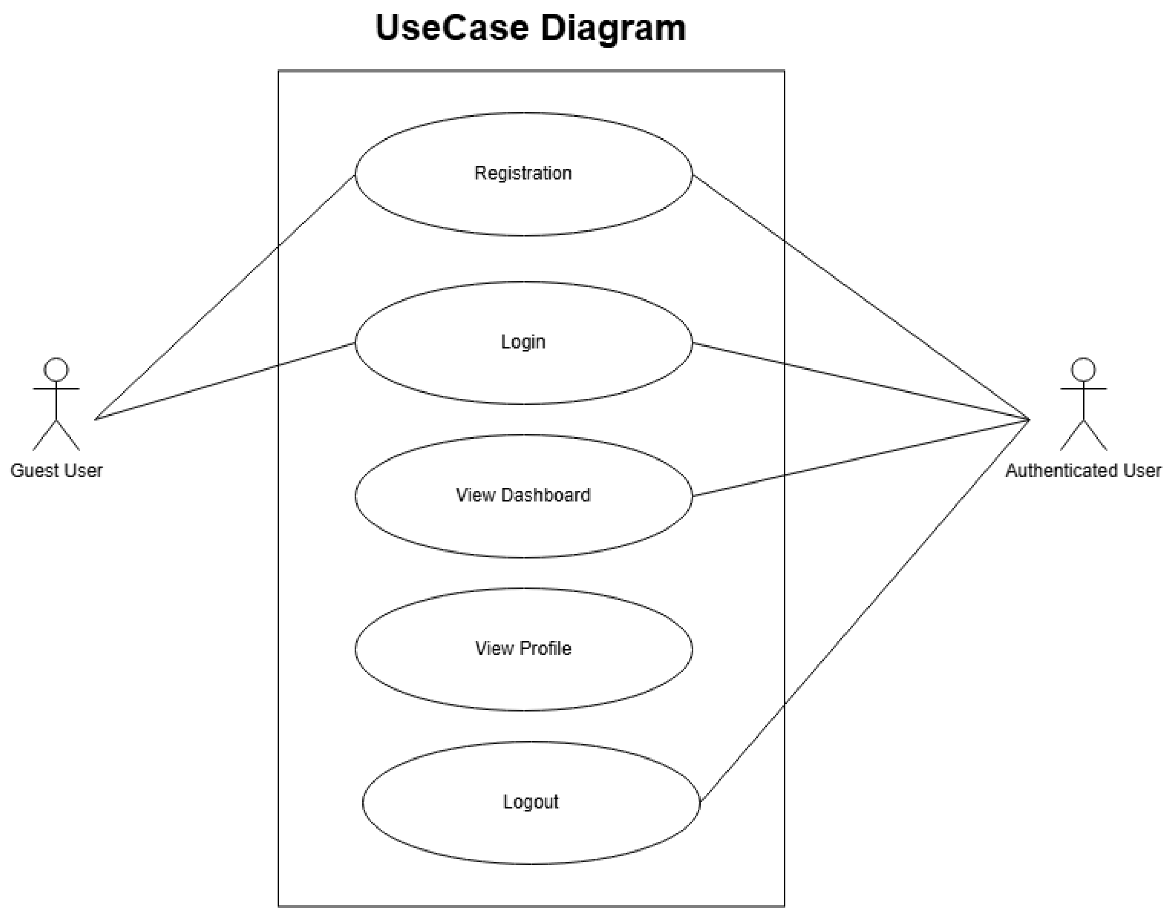
5.1. ERD

Insert ERD here

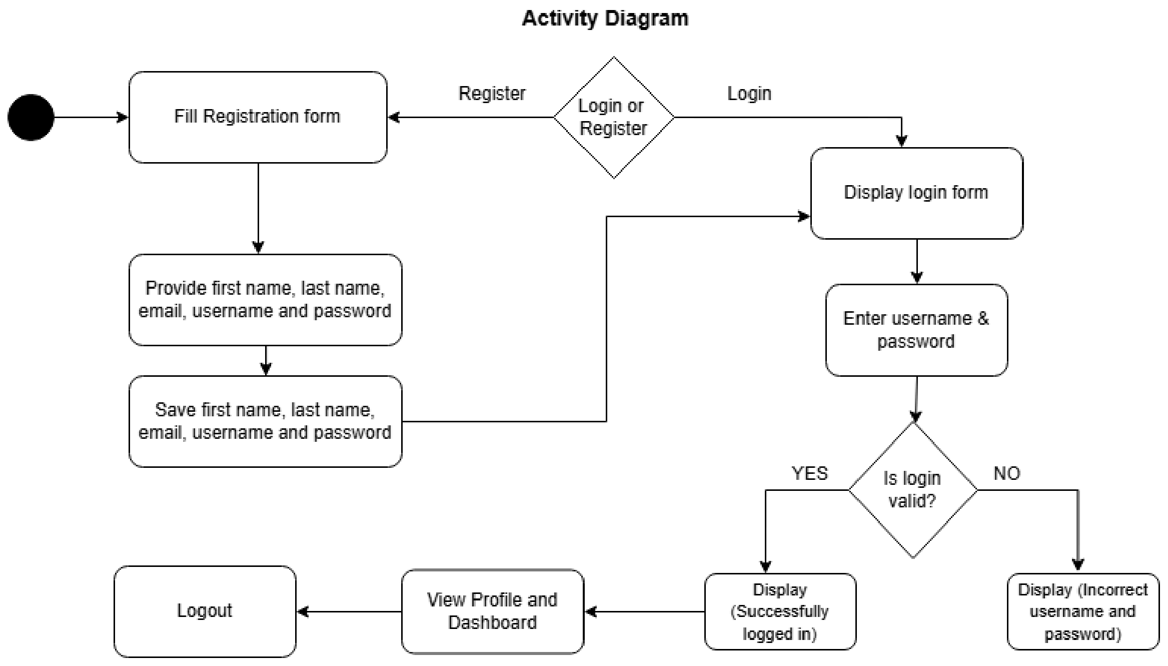


Users	
PK	userId int, auto_increment,
	userName varchar(50) unique not null,
	email varchar(255), unique not null,
	password_hash varchar(255), not null,
	firstName varchar(50), not null,
	lastName varchar(50), not null,
	is_active boolean default true,
	token_version int default 1,
	created_at timestamp default current_timestamp

5.2. Use Case Diagram

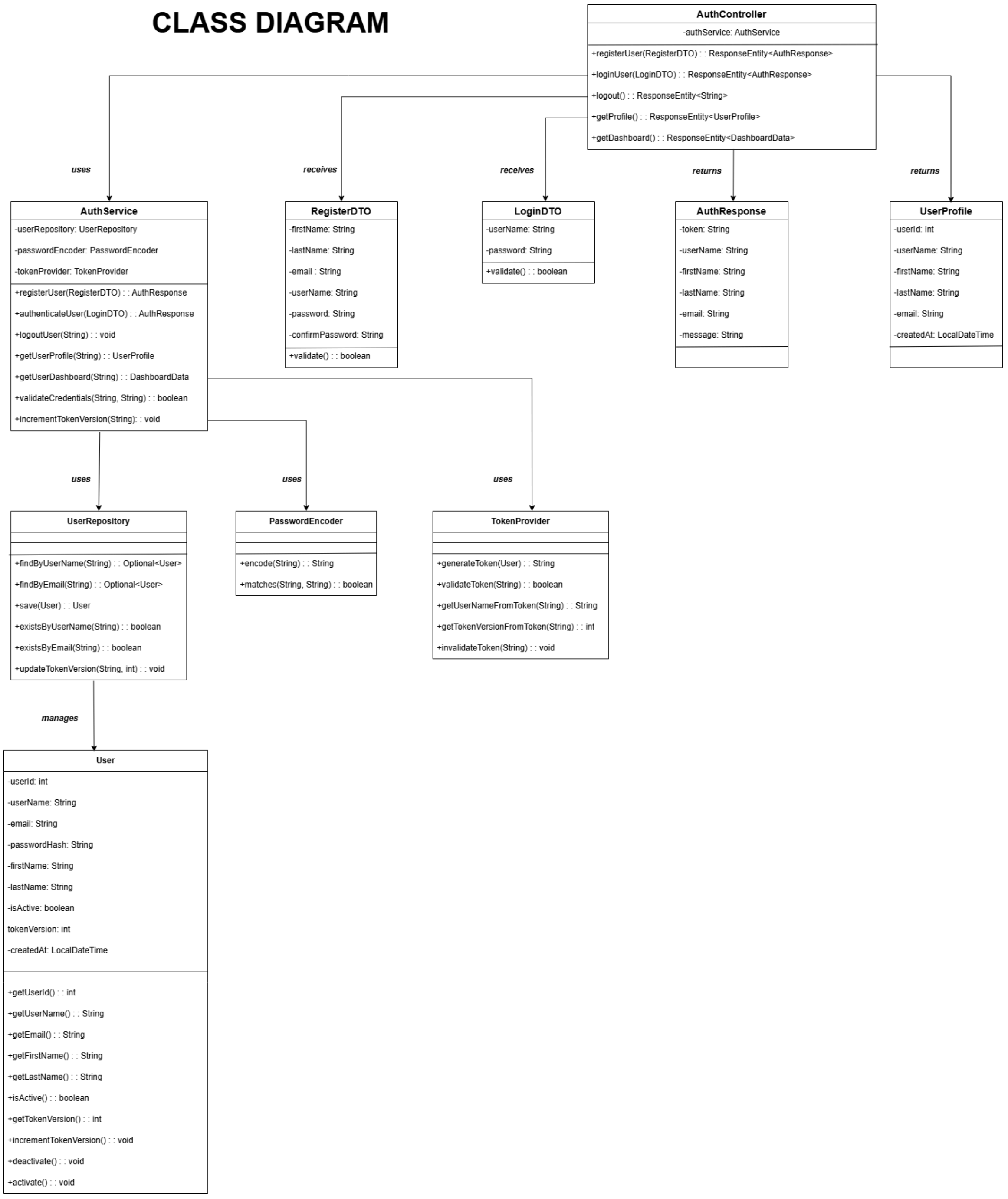


5.3. Activity Diagram


Activity Diagram
Fill Registration form
Register
Login or Register
Login
Display login form
Provide first name, last name, email, username and password
Save first name, last name, email, username and password
Enter username & password
YES
Is login valid?
NO
Logout
View Profile and Dashboard
Display (Successfully logged in)
Display (Incorrect username and password)

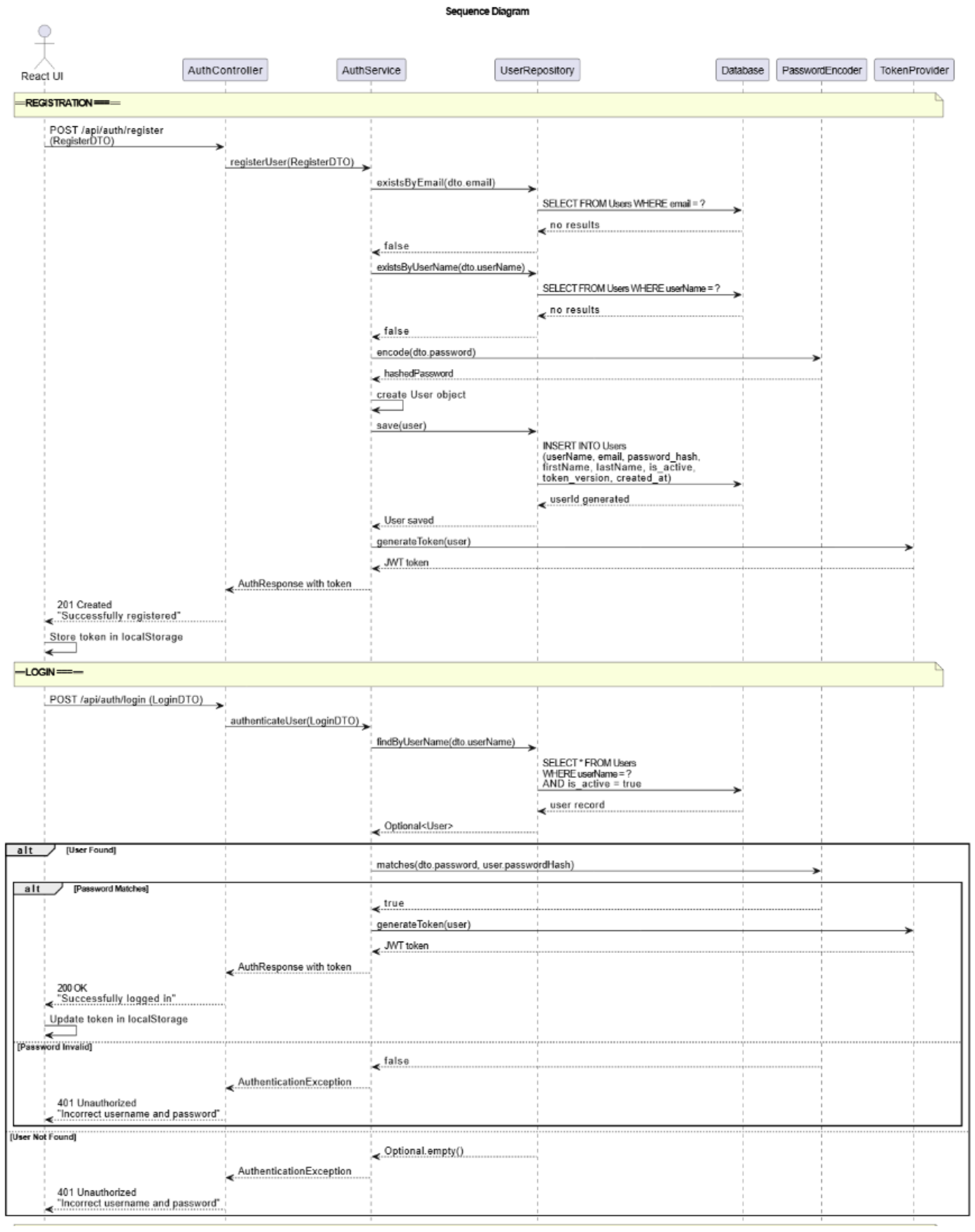
5.4. Class Diagram

Insert ERD here

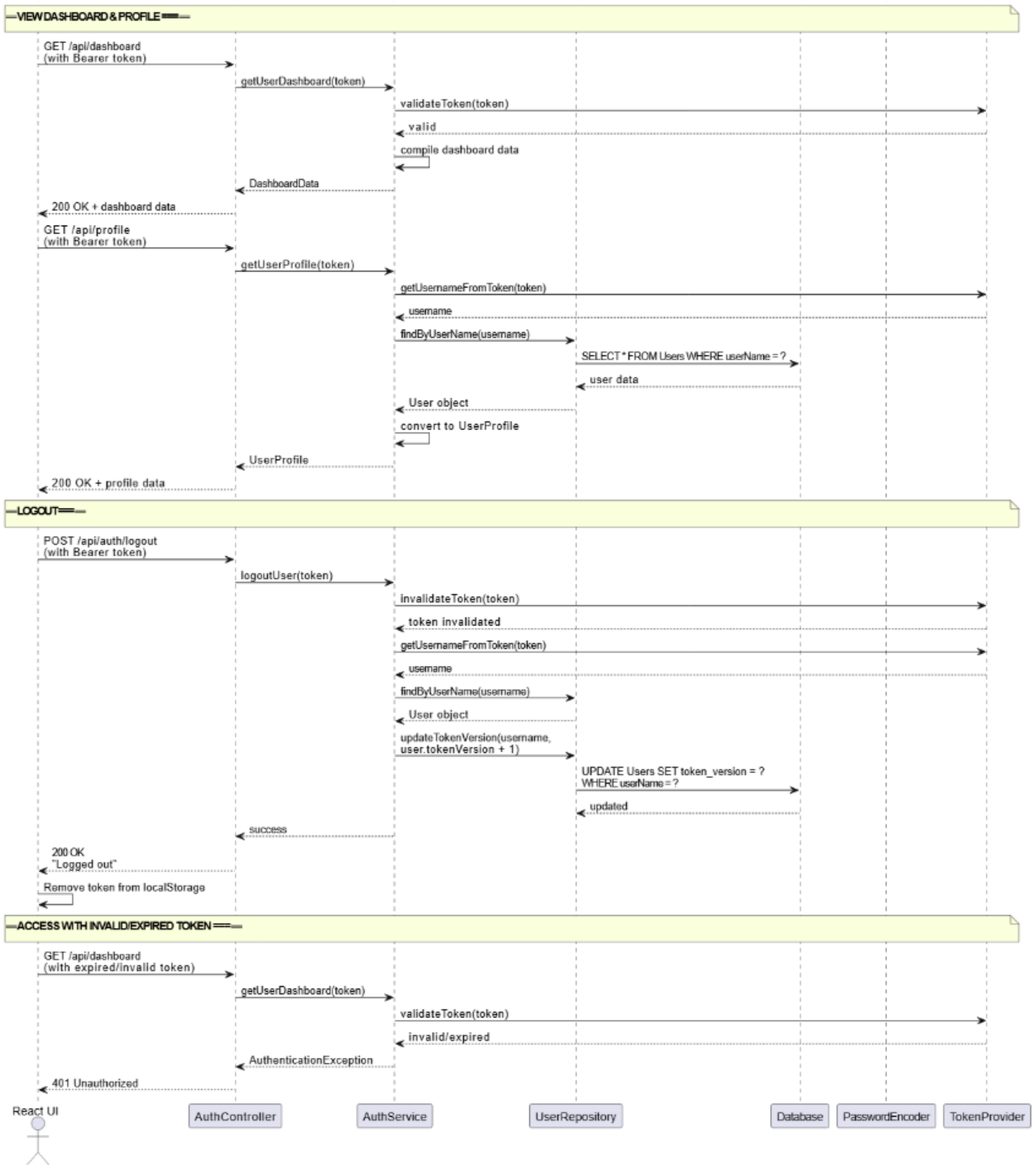

CLASS DIAGRAM
AuthController
-authService: AuthService
+registerUser(RegisterDTO) : : ResponseEntity<AuthResponse>
+loginUser(LoginDTO) : : ResponseEntity<AuthResponse>
+logout() : : ResponseEntity<String>
+getProfile() : : ResponseEntity<UserProfile>
+getDashboard() : : ResponseEntity<DashboardData>
uses
receives
receives
returns
returns
AuthService
-userRepository: UserRepository
-passwordEncoder: PasswordEncoder
-tokenProvider: TokenProvider
+registerUser(RegisterDTO) : : AuthResponse
+authenticateUser(LoginDTO) : : AuthResponse
+logoutUser(String) : : void
+getUserProfile(String) : : UserProfile
+getUserDashboard(String) : : DashboardData
+validateCredentials(String, String) : : boolean
+incrementTokenVersion(String): : void
RegisterDTO
-firstName: String
-lastName: String
-email : String
-userName: String
-password: String
-confirmPassword: String
+validate() : : boolean
LoginDTO
-userName: String
-password: String
+validate() : : boolean
AuthResponse
-token: String
-userName: String
-firstName: String
-lastName: String
-email: String
-message: String
UserProfile
-userId: int
-userName: String
-firstName: String
-lastName: String
-email: String
-createdAt: LocalDateTime
uses
uses
uses
UserRepository
+findByUserName(String) : : Optional<User>
+findByEmail(String) : : Optional<User>
+save(User) : : User
+existsByUserName(String) : : boolean
+existsByEmail(String) : : boolean
+updateTokenVersion(String, int) : : void
PasswordEncoder
+encode(String) : : String
+matches(String, String) : : boolean
TokenProvider
+generateToken(User) : : String
+validateToken(String) : : boolean
+getUserNameFromToken(String) : : String
+getTokenVersionFromToken(String) : : int
+invalidateToken(String) : : void
manages
User
-userId: int
-userName: String
-email: String
-passwordHash: String
-firstName: String
-lastName: String
-isActive: boolean
tokenVersion: int
-createdAt: LocalDateTime
+getUserId() : : int
+getUserName() : : String
+getEmail() : : String
+getFirstName() : : String
+getLastName() : : String
+isActive() : : boolean
+getTokenVersion() : : int
+incrementTokenVersion() : : void
+deactivate() : : void
+activate() : : void

5.5. Sequence Diagram

Insert ERD here



Sequence Diagram
React UI
AuthController
AuthService
UserRepository
Database
PasswordEncoder
TokenProvider
==REGISTRATION ==
POST /api/auth/register (RegisterDTO)
registerUser(RegisterDTO)
existsByEmail(dto.email)
SELECT FROM Users WHERE email = ?
no results
false
existsByUserName(dto.userName)
SELECT FROM Users WHERE userName = ?
no results
false
encode(dto.password)
hashedPassword
create User object
save(user)
INSERT INTO Users (userName, email, password_hash, firstName, lastName, is_active, token_version, created_at)
userId generated
User saved
generateToken(user)
JWT token
AuthResponse with token
201 Created "Successfully registered"
Store token in localStorage
==LOGIN ==
POST /api/auth/login (LoginDTO)
authenticateUser(LoginDTO)
findByUserName(dto.userName)
SELECT * FROM Users WHERE userName = ? AND is_active = true
user record
Optional<User>
alt [User Found]
matches(dto.password, user.passwordHash)
alt [Password Matches]
true
generateToken(user)
JWT token
AuthResponse with token
200 OK "Successfully logged in"
Update token in localStorage
[Password Invalid]
false
AuthenticationException
401 Unauthorized "Incorrect username and password"
[User Not Found]
Optional.empty()
AuthenticationException
401 Unauthorized "Incorrect username and password"



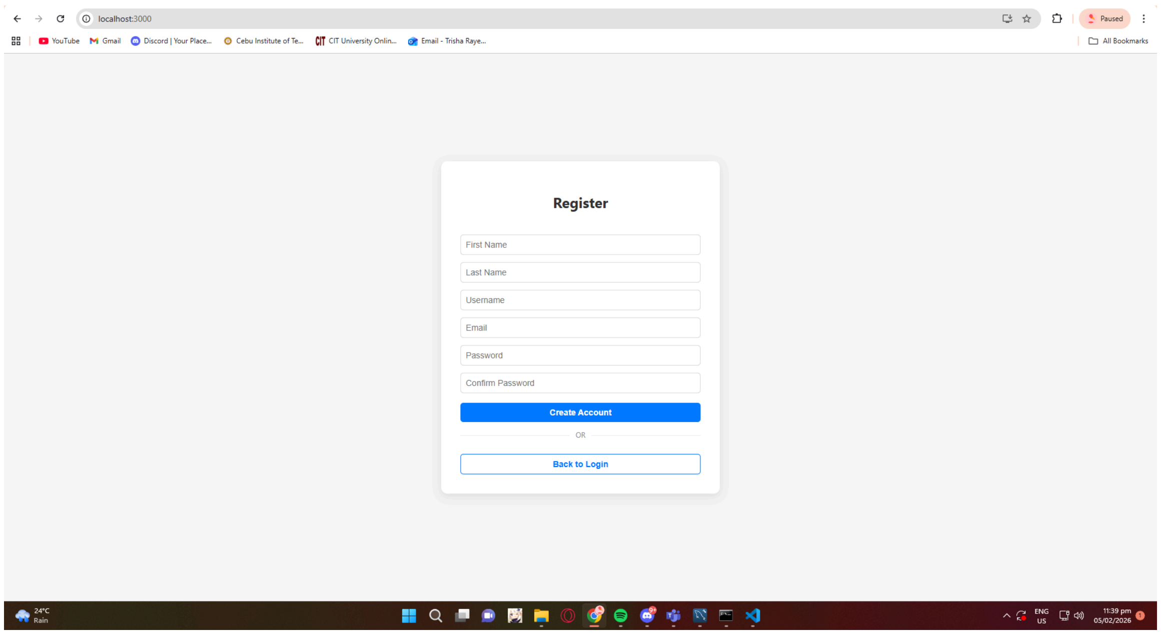
6. Appendices

Include any additional information, references, or support materials.

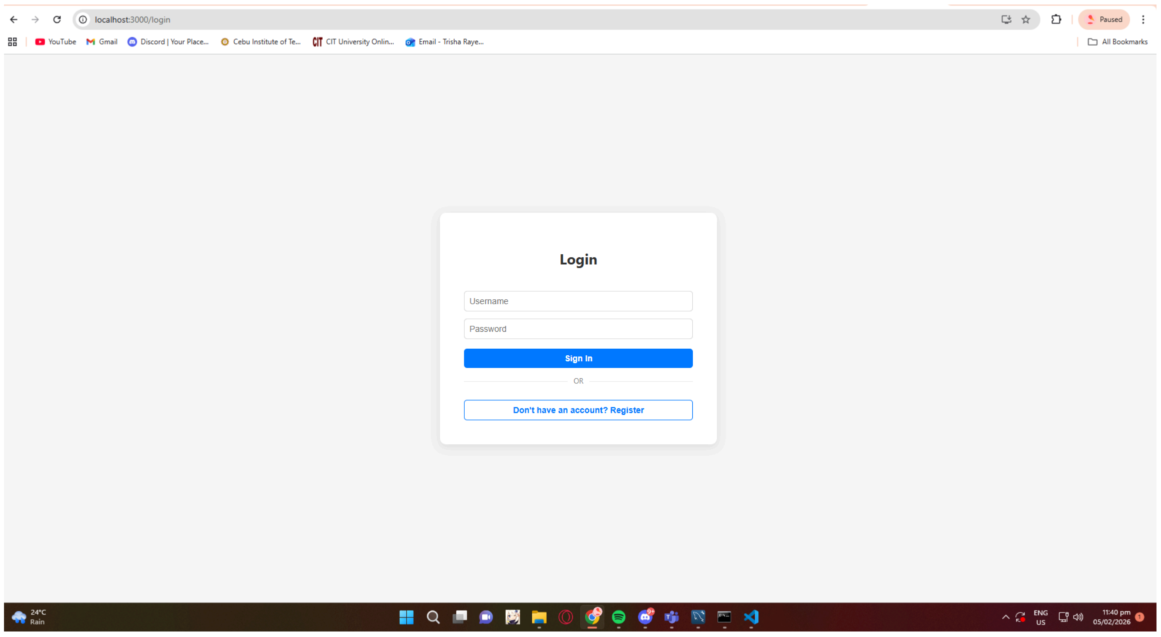
Screenshots:

FOR WEB UI:

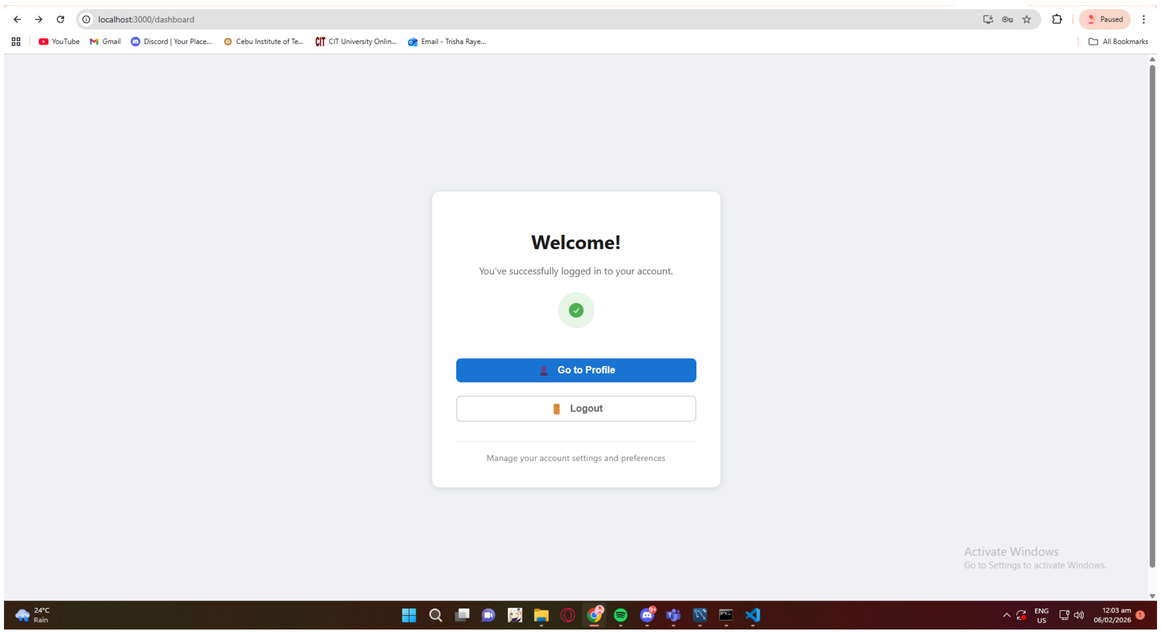
Register:



Login:



Dashboard:

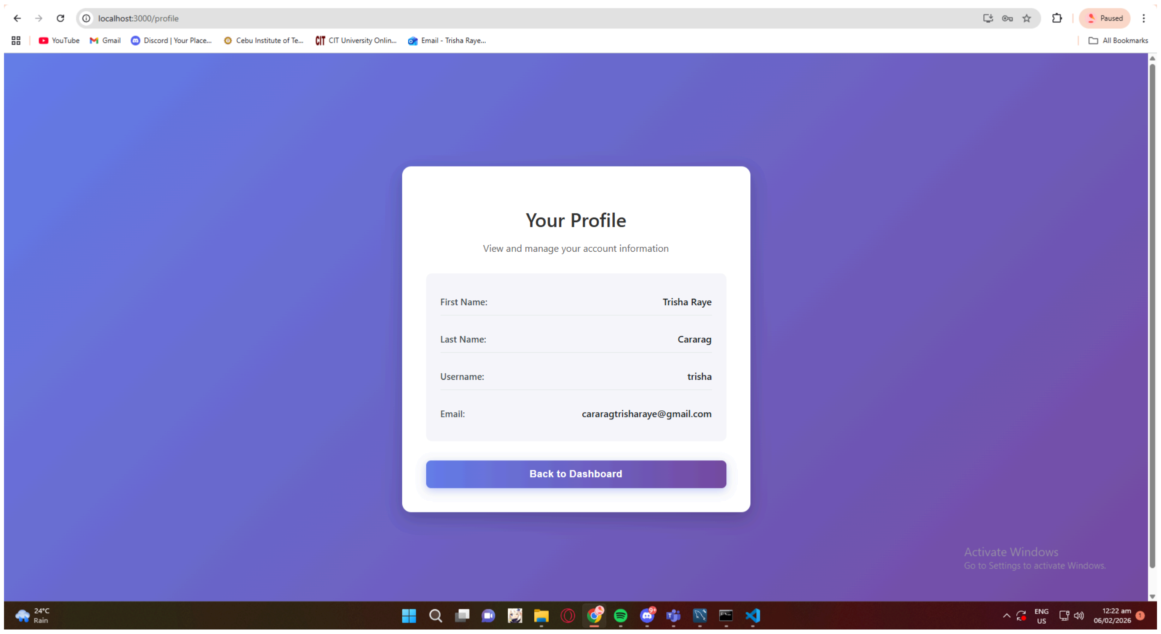

localhost:3000/dashboard

Welcome!

You've successfully logged in to your account.

Manage your account settings and preferences

Profile:


localhost:3000/profile

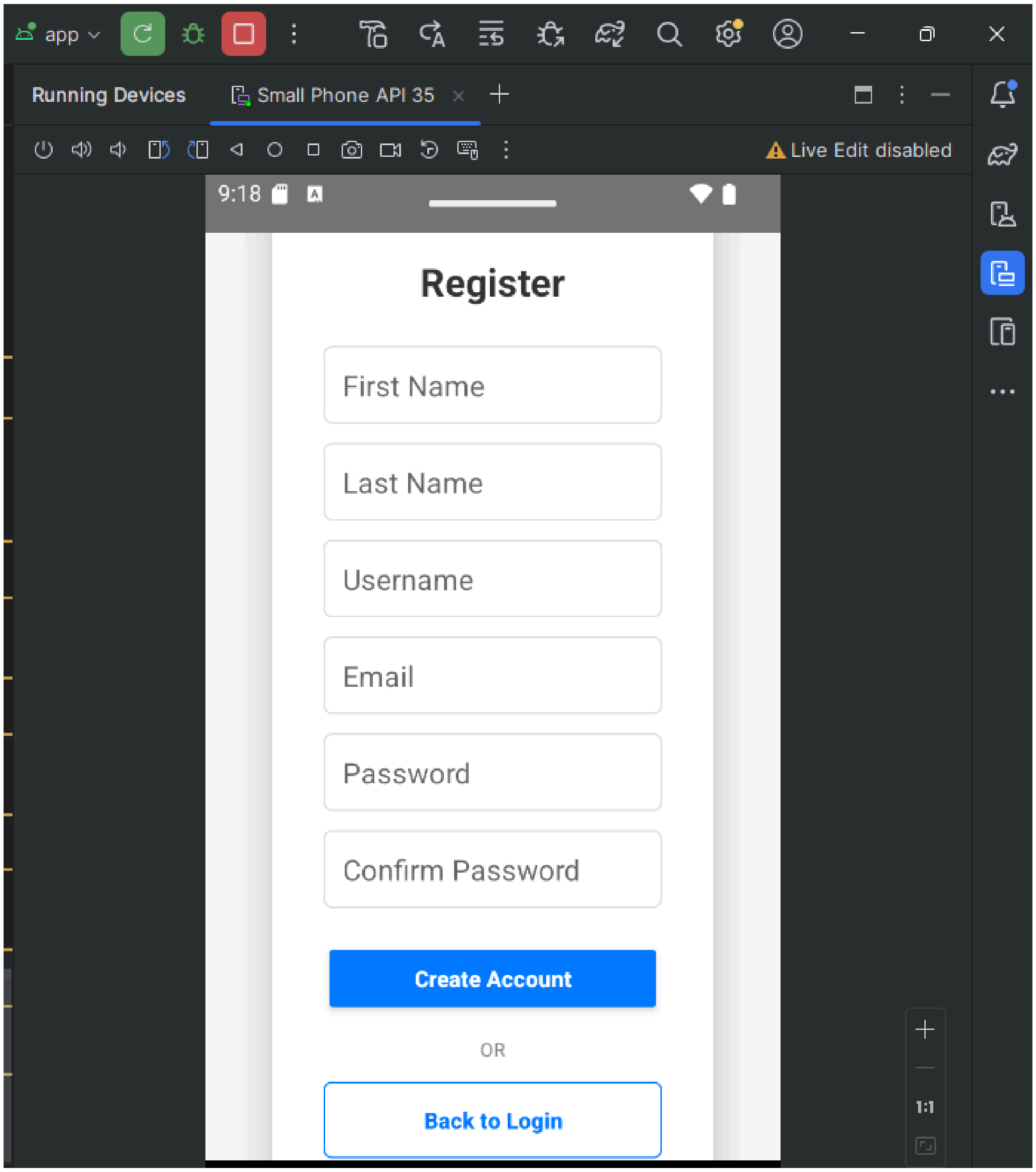
Your Profile

View and manage your account information

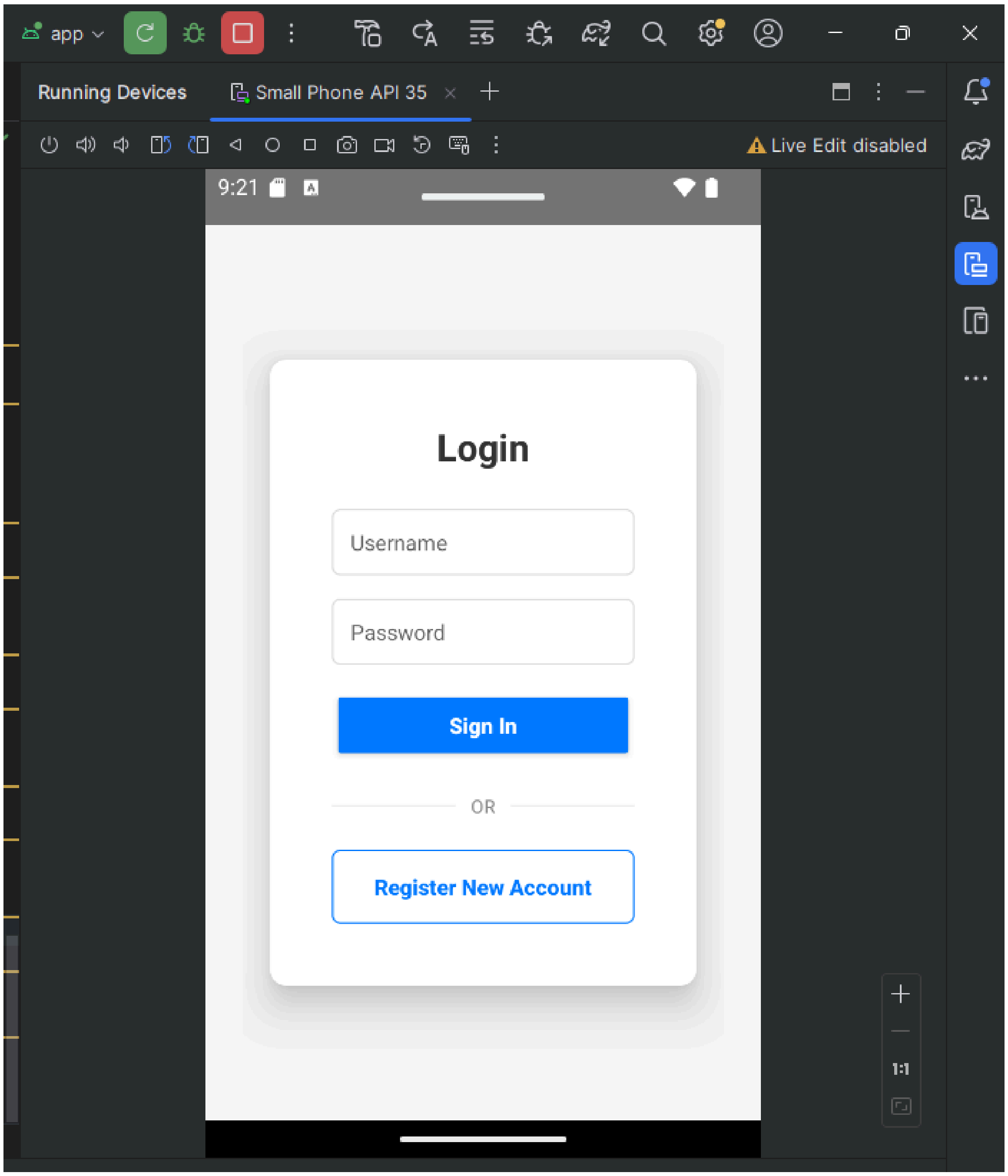
First Name:	Trisha Raye
Last Name:	Cararag
Username:	trisha
Email:	cararagtrisharaye@gmail.com

FOR MOBILE UI:

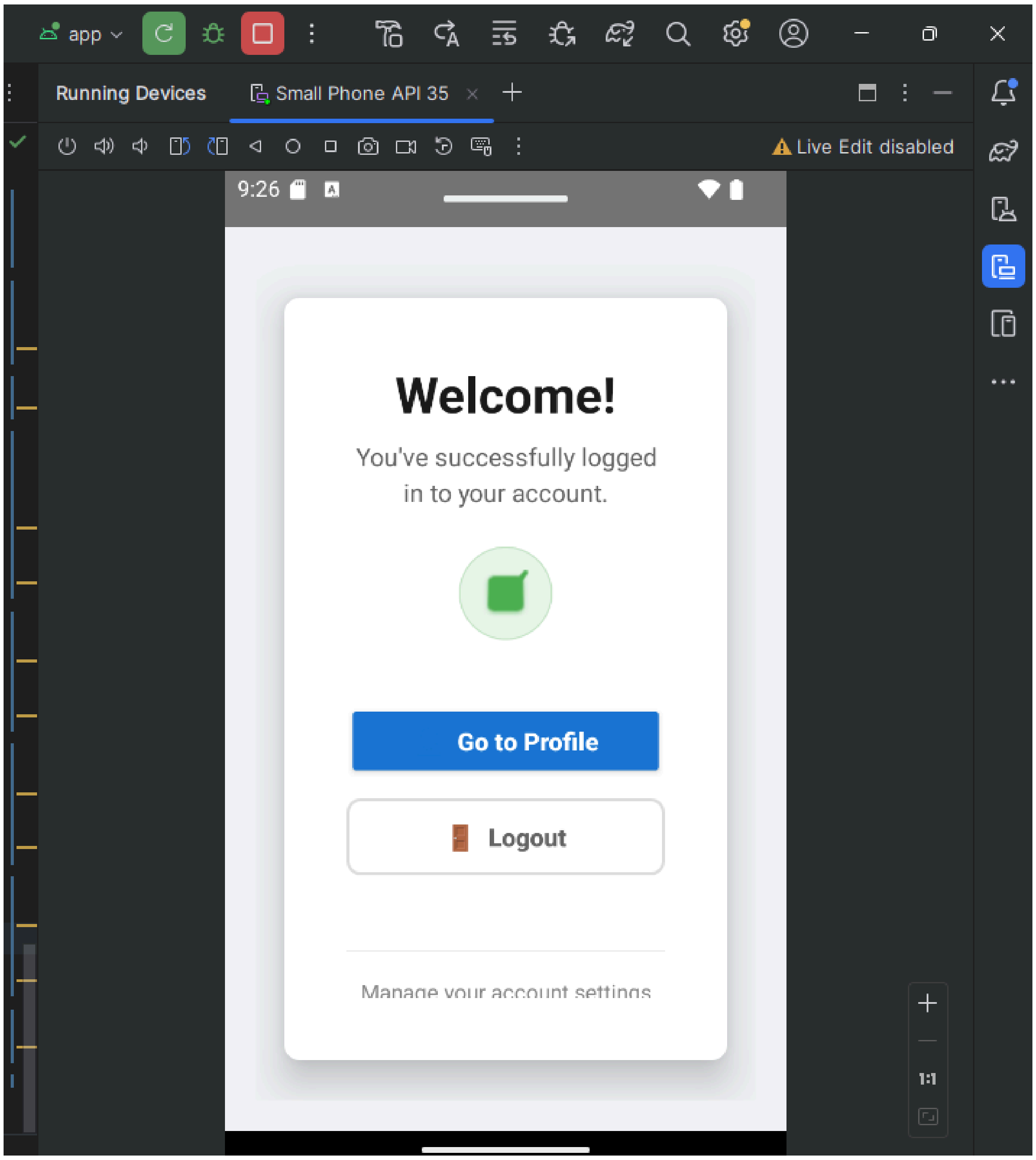
Register:



Login:


app
Running Devices
Small Phone API 35
Live Edit disabled
9:21
Login
Username
Password
Sign In
OR
Register New Account
1:1

Dashboard:

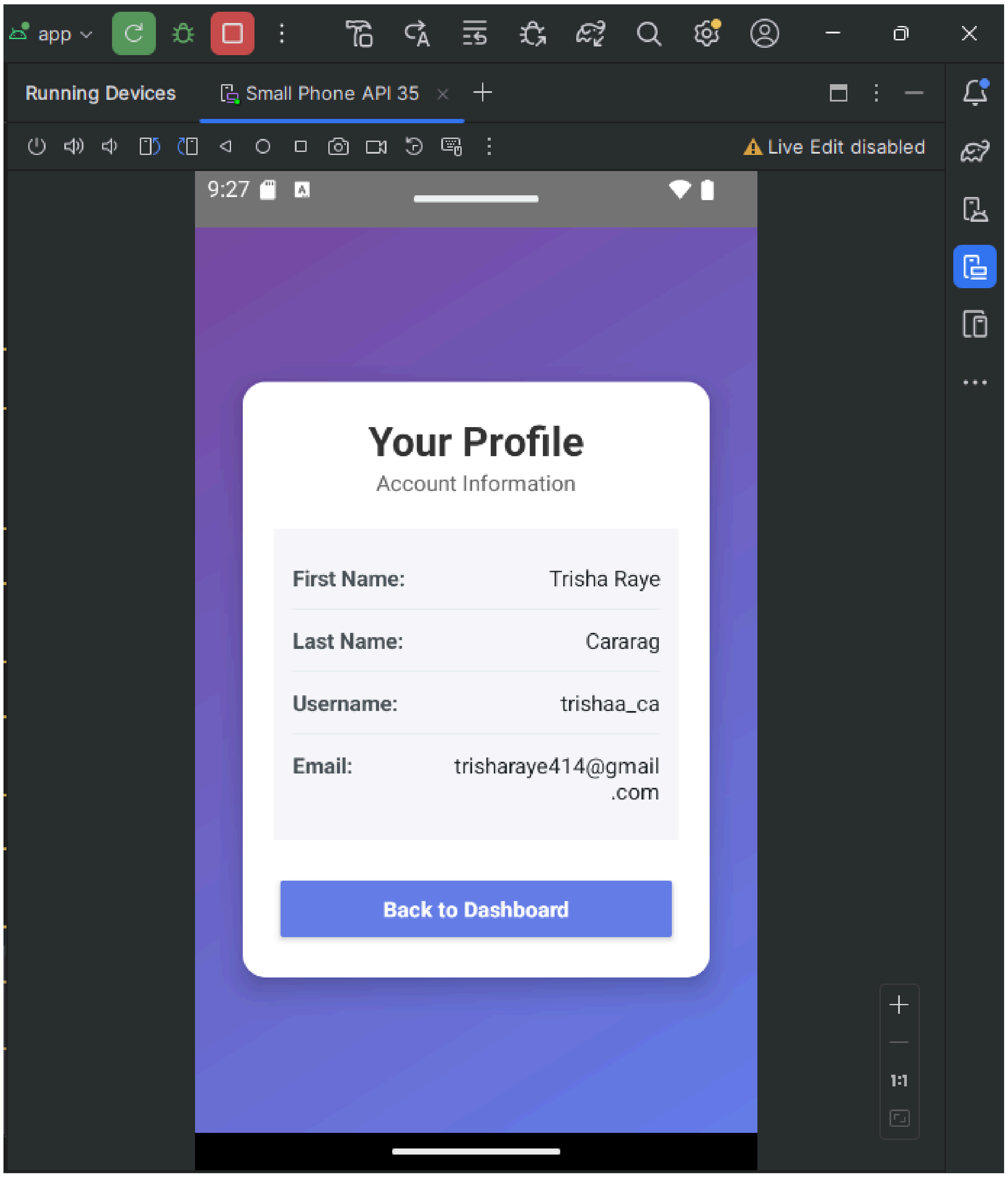


Welcome!

You've successfully logged in to your account.

Manage your account settings

Profile:



Your Profile

Account Information

First Name:	Trisha Raye
Last Name:	Cararag
Username:	trishaa_ca
Email:	trisharaye414@gmail.com