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: Dillan Marquin Ycoy

Date of Submission: February 5, 2026

Version: Version 2

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this activity is to create and showcase the requirements for a user registration and authentication system

1.2. Scope

The system will provide core identity management services, including:

- **User Registration:** Allow a user to create a unique account.
- **User Authentication:** Enable secure login using registered credentials.
- **Session Management:** Maintain a secure user session after login (using a token like JWT).
- **Authorization Enforcement:** Prevent unauthenticated users (Guest Users) from accessing protected resources (like the Profile/Dashboard).
- **Session Termination:** Provide secure logout functionality.

- **This documentation does not include:**

- Advanced features such as "Forgot Password," "Email Verification," or "Two-Factor Authentication (2FA)."
- User role-based access control (RBAC).
- Any application-specific business logic beyond the core authentication services.

1.3. Definitions, Acronyms, and Abbreviations

Term/Acronym	Definition
FRS	**F**unctional **R**equirements **S**pecification: This document.
UI	**U**ser **I**nterface: The front-end application (React UI) that users interact with.
API	**A**pplication **P**rogramming **I**nterface: The Spring Boot backend service that processes requests.
Guest User	An unauthenticated user of the system.
Authenticated User	A user who has successfully logged in and possesses a valid session/token.

JWT	**J**SON **W**eb **T**oken: The token used to securely transmit information and verify user identity after login.
Hashing	A one-way cryptographic process used to store passwords securely.

2. Overall Description

2.1. System Perspective

The User Registration & Authentication module functions as the core identity and access management layer for a larger "Mini App."It operates as a separate backend service (Spring Boot API) within a three-tier architecture, situated between the client (React UI) and the Database.Its primary role is to secure all other application services by handling user authentication, session token (JWT) management, and authorization enforcement.

2.2. User Classes and Characteristics

Guest User and Authenticated User

2.3. Operating Environment

- Frontend : This is the React UI which runs in a standard web browser (like Chrome or Firefox).
- Backend: This is the Spring Boot API, which is a program written in Java. It needs a server with a Java Virtual Machine (JVM) to run. It uses security tools like JWT for secure sessions and BCrypt to protect passwords.
- Database : This is where user accounts are stored, using a PostgreSQL or similar SQL database.

2.4. Assumptions and Dependencies

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

Assumptions

- Network/Environment: A reliable network connection exists between the client (React UI), the backend (Spring Boot API), and the Database (PostgreSQL).
- Client Implementation: The client application (React UI) correctly implements the logic for storing and transmitting the JWT (JSON Web Token) in subsequent requests.
- Backend Environment: A running Java Virtual Machine (JVM) environment is available for deploying the Spring Boot API.
- Database Availability: The PostgreSQL or similar SQL database is installed, running, and accessible to the Spring Boot API.

Dependencies (External Systems)

3. Database: A PostgreSQL-compatible database for persistent storage of user credentials and session data.
4. Client UI: The React front-end application is the consumer of the API services.

5. Security Library: The BCrypt hashing function library is available and properly integrated for password security.

6. System Features and Functional Requirements

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

6.1. Feature 1: User Identity Creation and Verification

Description:This feature encompasses the creation of new user accounts and the secure process for a user to prove their identity to the system.

Functional Requirements:

FR 1.1 - User Registration: The system must accept a unique identifier (e.g., email/username) and a password from a new user and create a corresponding account record.

FR 1.2 - Password Hashing: The system must use a strong, one-way cryptographic hash function (BCrypt) to store the user's password securely.

FR 1.3 - User Authentication: The system must validate a user's provided credentials against the stored records.

FR 1.4 - Authentication Response: Upon successful authentication, the system must issue a valid JSON Web Token (JWT) to the user. Upon failure, the system must return a generic failure message.

6.2. Feature 2: Session and Access Control

Description: This feature is responsible for maintaining a secure session after a user has logged in and preventing unauthenticated users from accessing protected parts of the application.

Functional Requirements:

FR 2.1 - Session Management: The system must accept and validate a JWT from an Authenticated User for all requests to protected resources.

FR 2.2 - Authorization Enforcement: The system must deny access to protected resources (e.g., Profile/Dashboard) if the user is a Guest User (unauthenticated) or presents an invalid/expired token.

FR 2.3 - Session Termination (Logout): The system must provide a secure endpoint for an Authenticated User to invalidate their current session/JWT.

7. Non-Functional Requirements

Security

- The system must store passwords using secure hashing (BCrypt).
- The system must use JWT for secure user sessions.
- Invalid or expired tokens must be rejected.
- Authentication error messages must not expose sensitive information.
- All authentication requests must use HTTPS.

Performance

- Login and registration requests should complete within a reasonable time.
- Token validation should be fast and not slow down the application.
- The system should handle multiple users logging in at the same time.

Reliability

- The system should remain available during normal operation.
- The system should handle errors without crashing.
- User data must remain consistent and safe in the database.

Usability

- API responses should be clear and easy for the frontend to process.
- Error messages shown to users should be simple and understandable.
- The login and registration process should be easy to complete.

Scalability

- The system should support multiple users at once.
- JWT-based authentication should allow the system to scale without storing sessions on the server.

Maintainability

- The system should be easy to update and maintain.
- Configuration values should be stored securely using environment variables.
- The code should follow standard development practices.

Portability

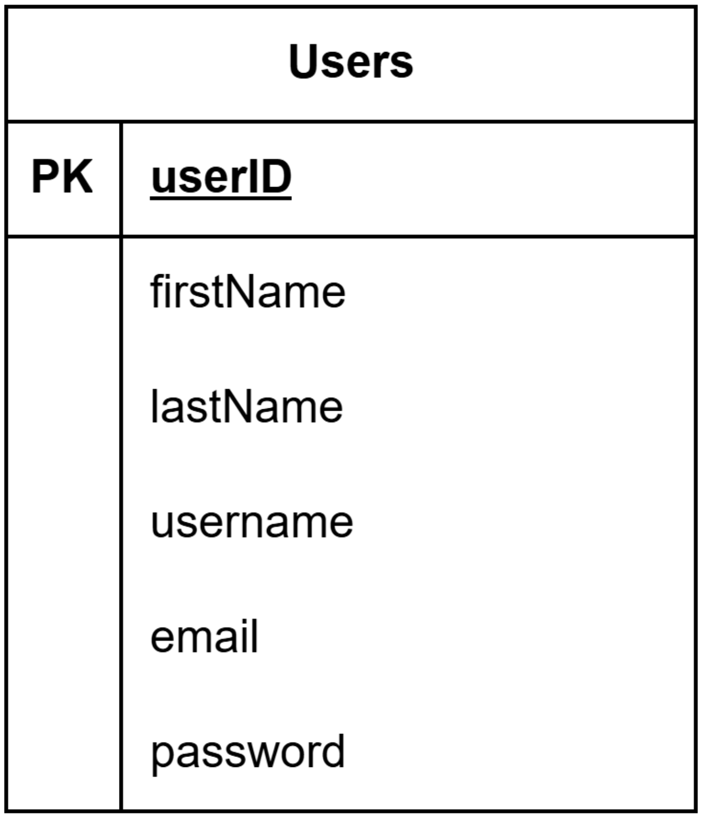
- The backend should run on any system with a Java Virtual Machine (JVM).
- The database should support PostgreSQL or similar SQL databases.

8. System Models (Diagrams)

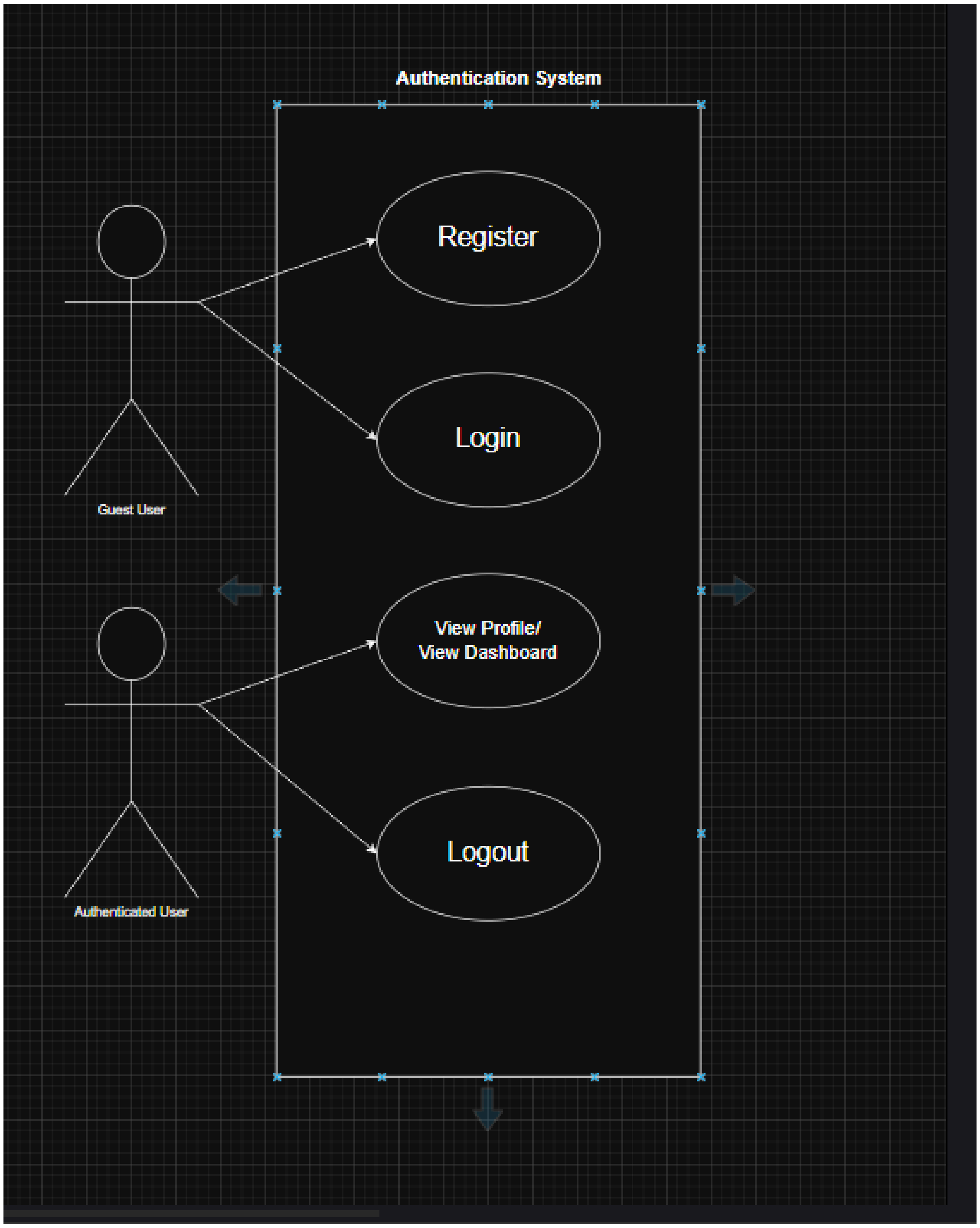
Insert the necessary diagrams for the system:

8.1. ERD

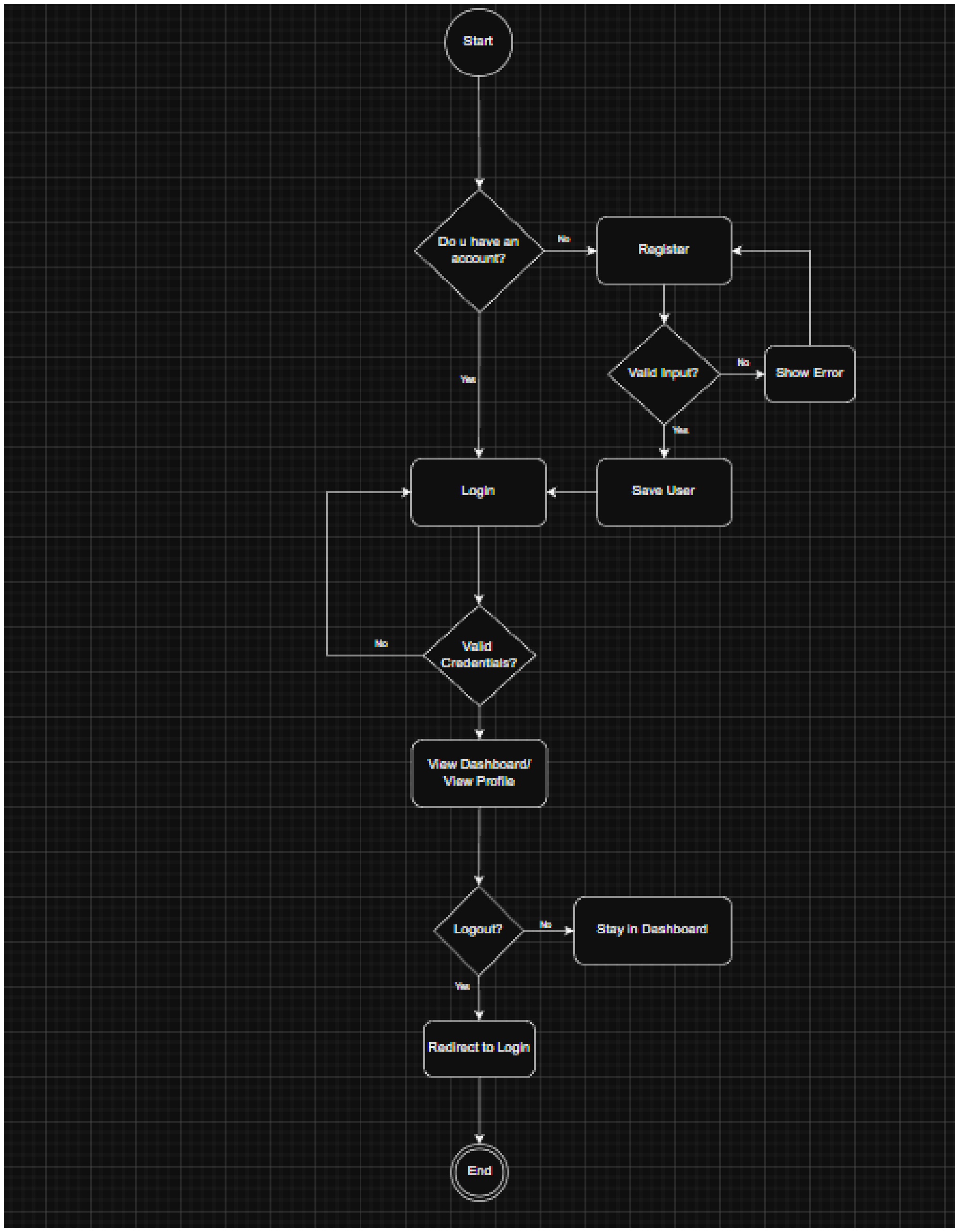
Insert ERD here



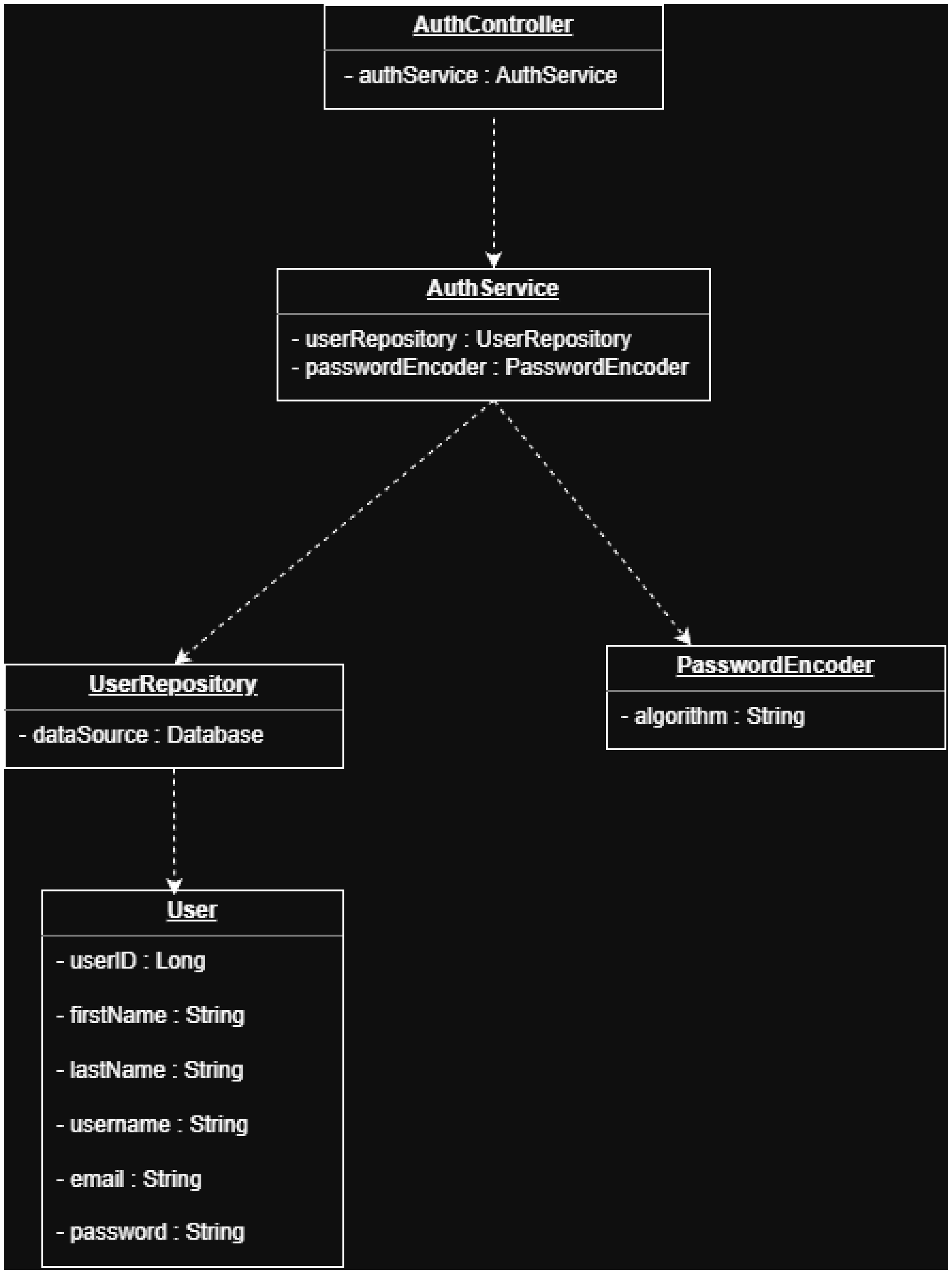
8.2. Use Case Diagram



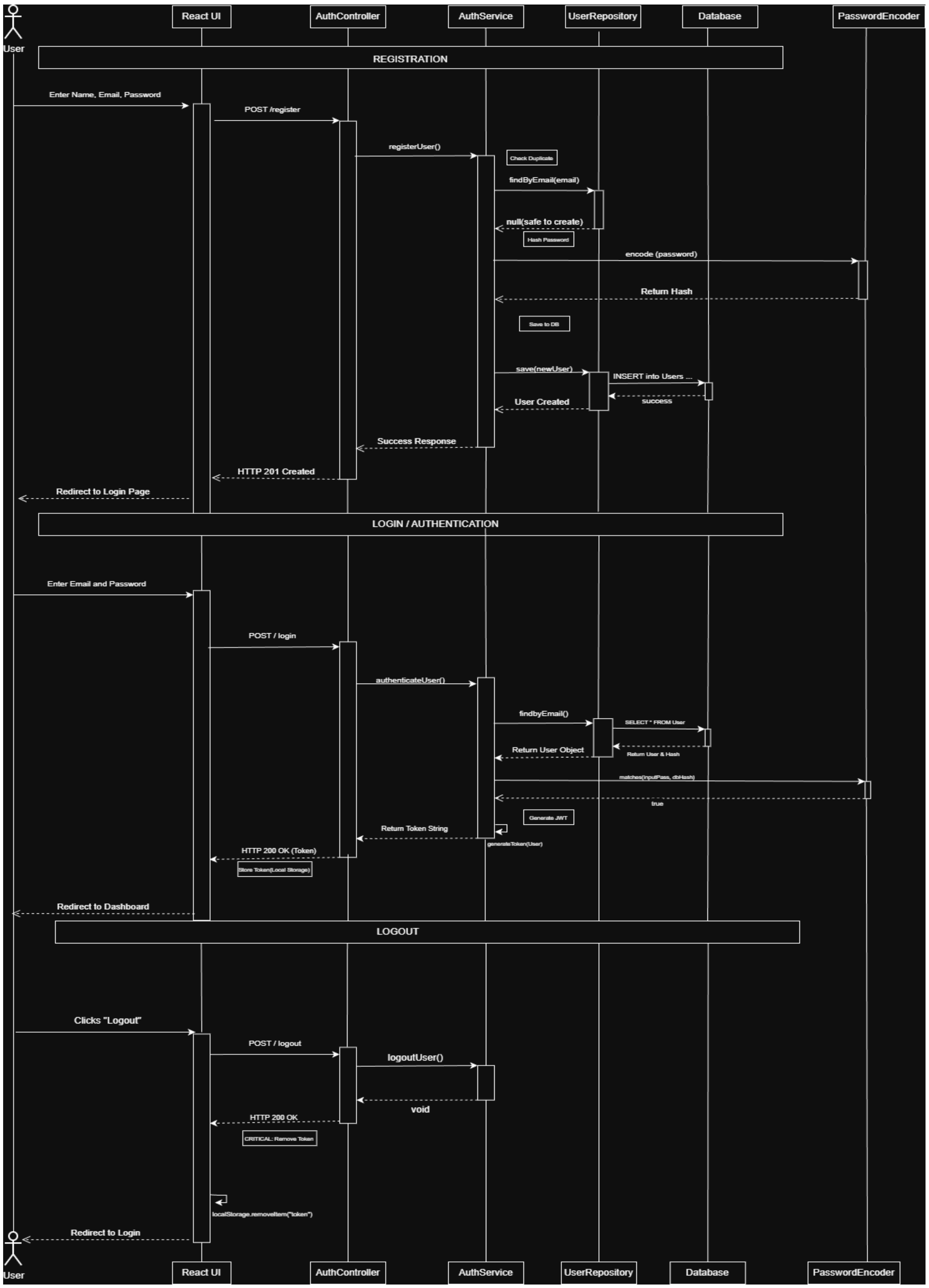
8.3. Activity Diagram



8.4. Class Diagram



8.5. Sequence Diagram


User
React UI
AuthController
AuthService
UserRepository
Database
PasswordEncoder
REGISTRATION
Enter Name, Email, Password
POST /register
registerUser()
Check Duplicate
findByEmail(email)
null(safe to create)
Hash Password
encode (password)
Return Hash
Save to DB
save(newUser)
INSERT into Users ...
success
User Created
Success Response
HTTP 201 Created
Redirect to Login Page
LOGIN / AUTHENTICATION
Enter Email and Password
POST / login
authenticateUser()
findbyEmail()
SELECT * FROM User
Return User & Hash
Return User Object
matches(inputPass, dbHash)
true
Generate JWT
generateToken(User)
Return Token String
HTTP 200 OK (Token)
Store Token(Local Storage)
Redirect to Dashboard
LOGOUT
Clicks "Logout"
POST / logout
logoutUser()
void
HTTP 200 OK
CRITICAL: Remove Token
localStorage.removeItem("token")
Redirect to Login
User
React UI
AuthController
AuthService
UserRepository
Database
PasswordEncoder

9. Appendices

Include any additional information, references, or support materials.

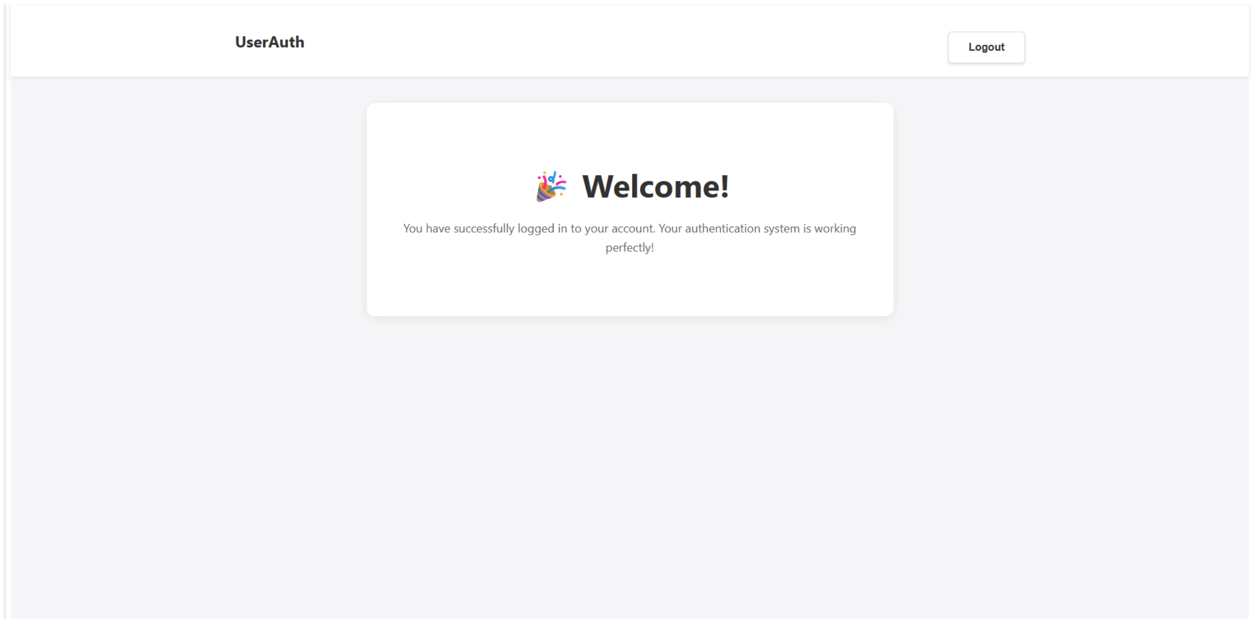
- BCrypt: Reference to the password-hashing function used for secure storage of user credentials.
- Java Virtual Machine (JVM): The environment required to run the Spring Boot API backend.
- React: The JavaScript library used for building the User Interface (UI).
- PostgreSQL: Example of the type of SQL database used for persistent user data storage.

B. Supporting Documentation

- Database Schema (D.1): Detailed Entity-Relationship Diagram (ERD) for the User and Session entities, which will be elaborated upon in Section 5.1.
- API Endpoints Documentation (D.2): A separate document detailing the specific HTTP methods, request/response bodies, and status codes for the Registration, Authentication, and Logout endpoints.
- Security Implementation Details (D.3): Documentation on the configuration of Spring Security, including the filter chain setup and custom JWT authentication provider.

C. References

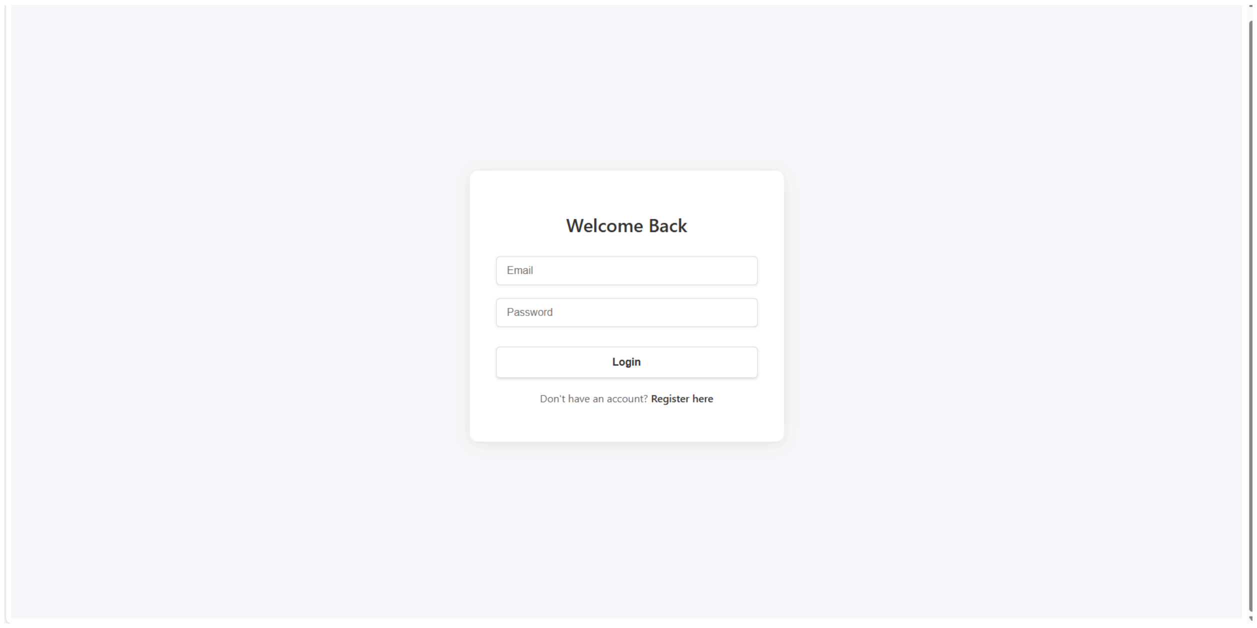
- JSON Web Token (JWT) Standard: RFC 7519
- Spring Boot Documentation: Official documentation for the framework used in the API.
- React Documentation: Official documentation for the frontend framework.



Welcome!

You have successfully logged in to your account. Your authentication system is working perfectly!

DASHBOARD


Welcome Back

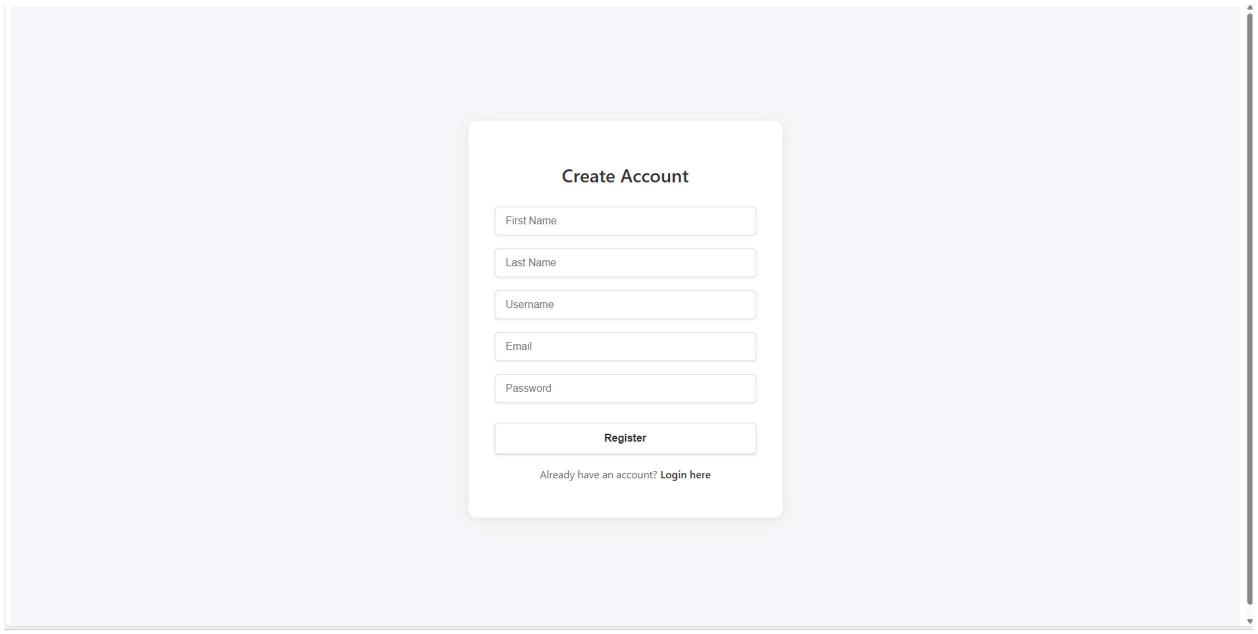
Email

Password

Login

Don't have an account? **Register here**

LOGIN


Create Account

First Name

Last Name

Username

Email

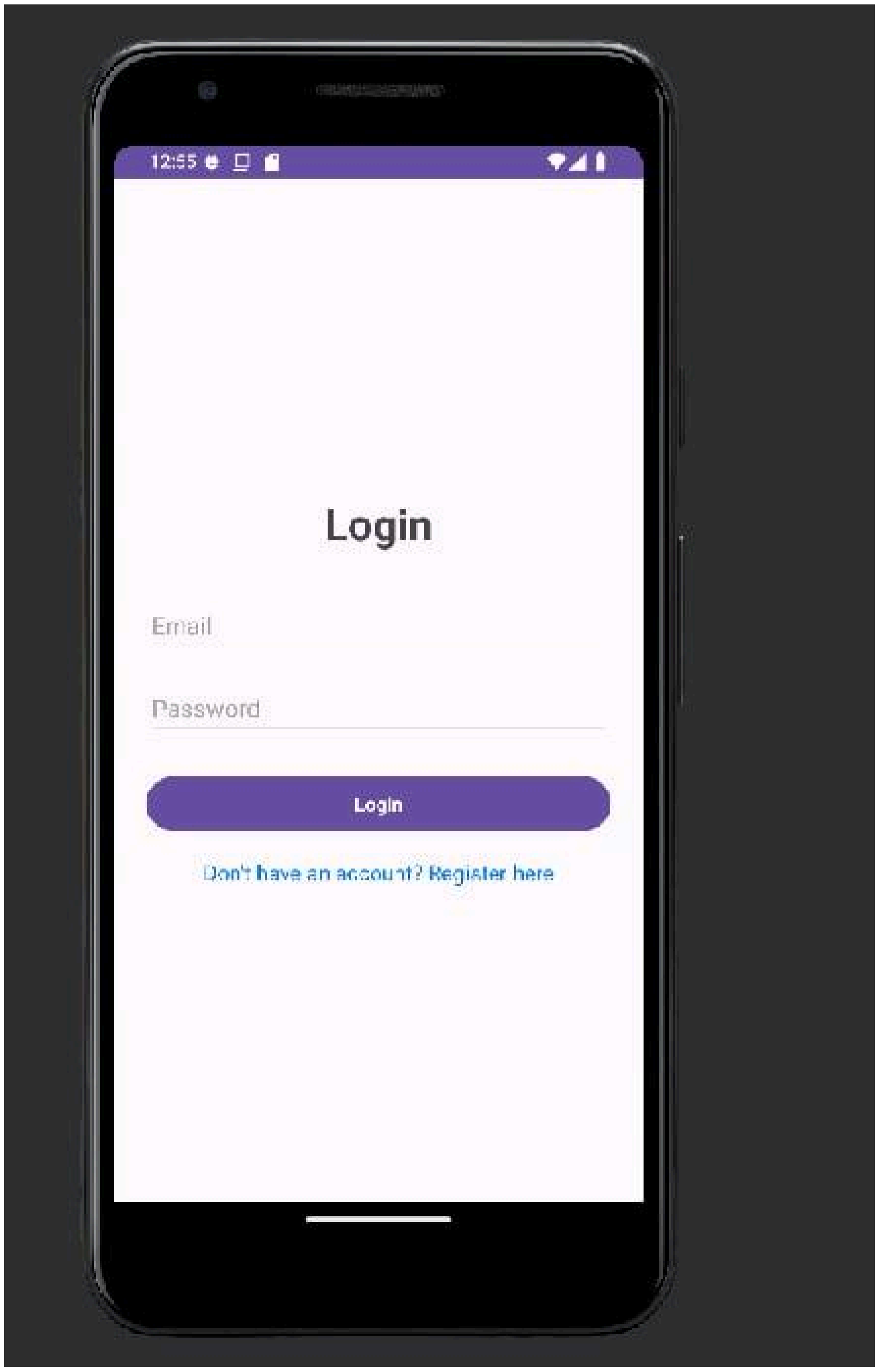
Password

Register

Already have an account? **Login here**

REGISTER

MOBILE


12:55

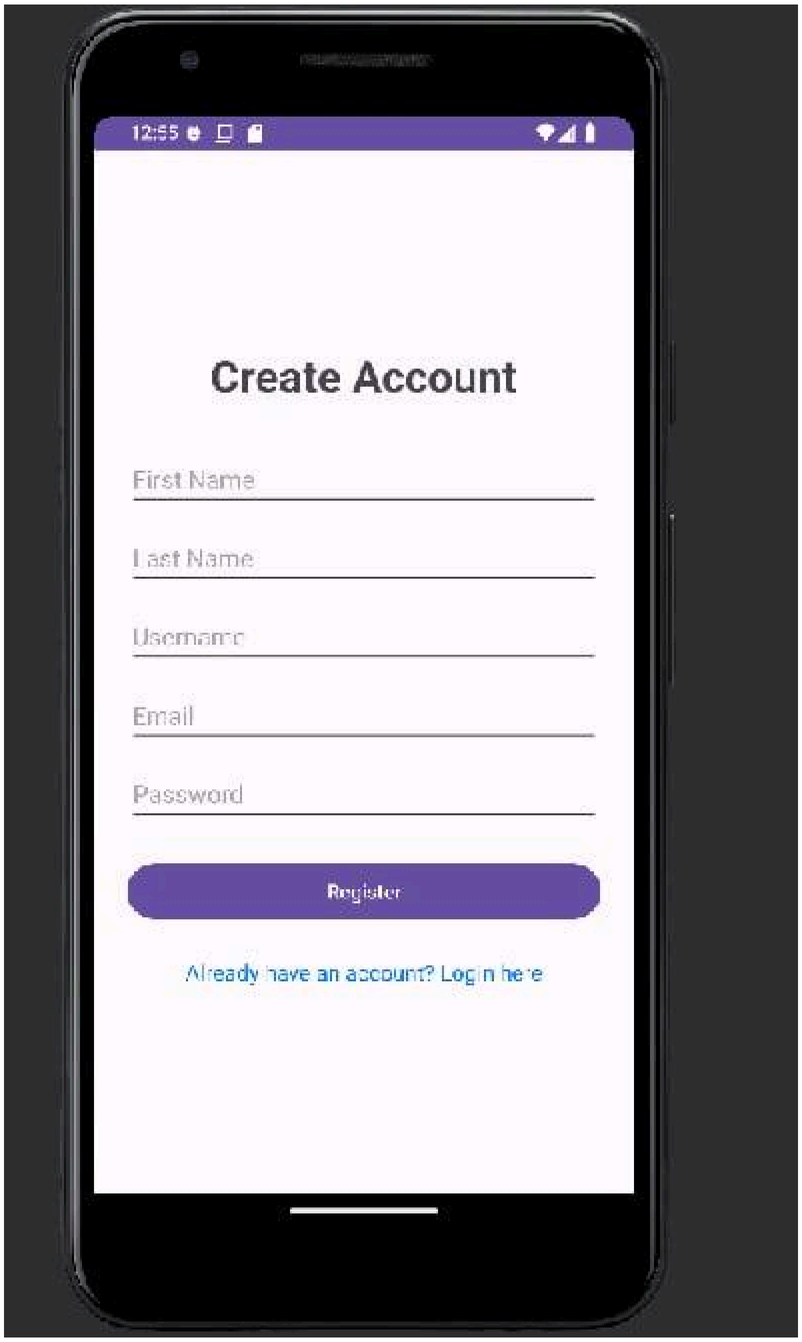
Login

Email

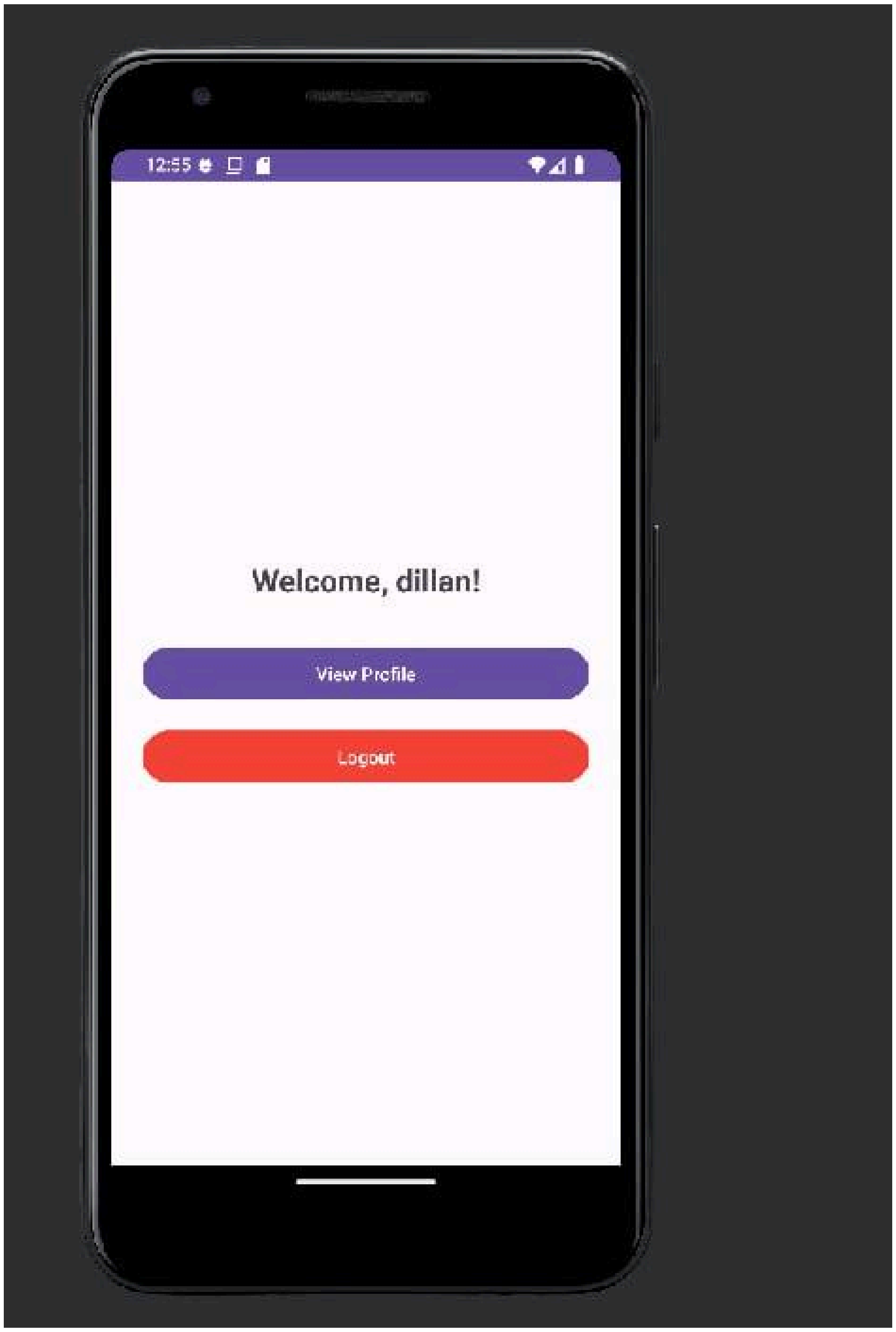
Password

Don't have an account? Register here

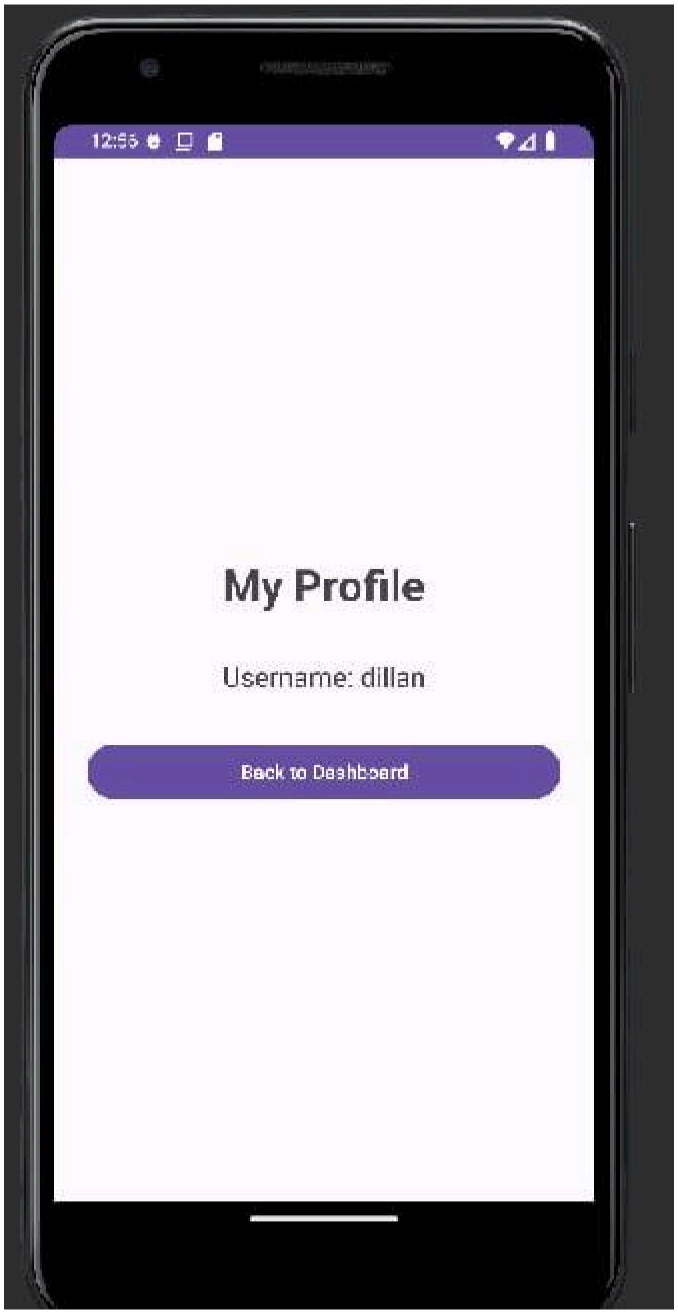
LOGIN


12:55
Create Account
First Name
Last Name
Username
Email
Password
Register
Already have an account? Login here

REGISTER



DASHBOARD


My Profile

Username: dillan

Profile