A black background with red text

AI-generated content may be incorrect.

IT342-G2

SYSTEMS INTEGRATION AND ARCHITECTURE 1

**FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)**

Project Title: Mini App

Prepared By: Elvin Opalla Lagamo Jr.

Date of Submission: 02/17/2026

Version: 3

Table of Contents

[1. Introduction 3](#_Toc220513504)

[1.1. Purpose 3](#_Toc220513505)

[1.2. Scope 3](#_Toc220513506)

[1.3. Definitions, Acronyms, and Abbreviations 3](#_Toc220513507)

[2. Overall Description 3](#_Toc220513508)

[2.1. System Perspective 3](#_Toc220513509)

[2.2. User Classes and Characteristics 4](#_Toc220513510)

[2.3. Operating Environment 4](#_Toc220513511)

[2.4. Assumptions and Dependencies 4](#_Toc220513512)

[3. System Features and Functional Requirements 4](#_Toc220513513)

[3.1. Feature 1: 4](#_Toc220513514)

[3.2. Feature 2: 4](#_Toc220513515)

[4. Non-Functional Requirements 5](#_Toc220513516)

[5. System Models (Diagrams) 7](#_Toc220513517)

[5.1. ERD 7](#_Toc220513518)

[5.2. Use Case Diagram 7](#_Toc220513519)

[5.3. Activity Diagram 8](#_Toc220513520)

[5.4. Class Diagram 10](#_Toc220513521)

[5.5. Sequence Diagram 11](#_Toc220513522)

[6. Appendices 12](#_Toc220513523)

[7. Documentations 13](#_Toc220513523)

## Introduction

### Purpose

The purpose of this document is to define the functional and non-functional requirements for the **"Mini App"** Authentication System. This document serves as a guideline for the design, implementation, and verification of the system's security and user management modules. It is intended for the development team, system architects, and project stakeholders (instructors) to ensure a clear understanding of the system's deliverables.

### Scope

The **Mini App** is a web-based application focused on secure user identity management. The system encompasses the following core functionalities:

* **User Registration:** Capturing detailed demographic data and credentials.
* **Authentication:** Verifying user credentials securely to grant access.
* **Session Management:** utilizing token-based providers to manage user states.
* **Dashboard Access:** Providing a secured view for authenticated users.
* **Logout:** Securely terminating user sessions.

The system boundary is limited to the authentication and user profile management modules, it does not currently include external third-party integrations (e.g., Google/Facebook login) or complex business logic beyond user management.

### Definitions, Acronyms, and Abbreviations

* **FRS:** Functional Requirements Specification.
* **ERD:** Entity Relationship Diagram.
* **DTO:** Data Transfer Object (used to transfer data between the Controller and Service layers).
* **JWT:** JSON Web Token (implied mechanism for the TokenProvider).
* **Service Layer:** The architectural layer responsible for business logic.
* **Repository Layer:** The architectural layer responsible for database interactions.
* **UI:** User Interface (Frontend).

## Overall Description

### System Perspective

The Mini App operates within a **Client-Server Layered Architecture**.

* **Frontend (Client):** Handles user interactions and input validation.
* **Backend (Server):** structured into Controller, Service, and Repository layers to ensure separation of concerns.
* **Database:** A relational database storing persistent user records. The system relies on HTTP/RESTful communication between the client and the server.

### User Classes and Characteristics

 **Guest User:** An unauthenticated user who can only access the "Landing Page," "Login Page," and "Registration Page." Their goal is to create an account or gain access to the system.

 **Authenticated User:** A user who has successfully logged in. They have access to the "Dashboard" and can view their profile details. They also have the ability to "Log out."

### Operating Environment

 **Client Side:** Modern Web Browser (Chrome, Firefox, Edge) with JavaScript enabled.

 **Server Side:** Java-based application server (Spring Boot environment).

 **Database:** MySQL or compatible relational database management system.

 **Network:** Stable internet or intranet connection.

### Assumptions and Dependencies

 **Assumption:** Users will provide valid email addresses during registration.

 **Assumption:** The browser supports LocalStorage for saving authentication tokens.

 **Dependency:** The backend API must be running and accessible for the frontend to function.

 **Dependency:** The Database service must be active to persist user data.

## System Features and Functional Requirements

### Feature 1: User Registration

**Description:** This feature allows a Guest User to create a new account by providing personal details and credentials. The system must validate the input, ensure the email is unique, and securely store the password.

**Functional Requirements:**

* **REQ-REG-01:** The system shall provide a registration form requiring the following fields: Firstname, Middlename, Lastname, Street, Barangay, Municipality, Province, Country, Email, Password, and Contact Number.
* **REQ-REG-02:** The system shall validate that the email address format is correct and does not already exist in the database (via existsByEmail check).
* **REQ-REG-03:** The AuthService shall use a PasswordEncoder to hash the raw password before saving it to the database.
* **REQ-REG-04:** Upon successful registration, the system shall redirect the user to the Login Page.
* **REQ-REG-05:** If validation fails (e.g., duplicate email), the system shall display an appropriate error message to the user.

### Feature 2: User Login (Authentication)

**Description:** This feature allows users with existing accounts to access the secured Dashboard. It verifies the provided email and password against the stored records.

**Functional Requirements:**

* **REQ-LOG-01:** The system shall accept an Email and Password via the Login Modal/Page.
* **REQ-LOG-02:** The AuthService shall retrieve the user record using the email provided.
* **REQ-LOG-03:** The system shall verify the entered password against the stored hashed password using the PasswordEncoder.
* **REQ-LOG-04:** If credentials are valid, the TokenProvider shall generate an authentication token.
* **REQ-LOG-05:** The system shall return the token to the client, store it in LocalStorage, and redirect the user to the Dashboard.
* **REQ-LOG-06:** If authentication fails, the system shall display an "Invalid Credentials" error message.

### Feature 3: Logout

**Description:** This feature allows an Authenticated User to securely terminate their session and return to the public area of the application.

**Functional Requirements:**

* **REQ-OUT-01:** The system shall display a confirmation modal when the user clicks the "Logout" button.
* **REQ-OUT-02:** Upon confirmation, the client shall send a logout request to the AuthController.
* **REQ-OUT-03:** The AuthService shall invalidate the user's current session/token.
* **REQ-OUT-04:** The client application shall clear the stored authentication token from LocalStorage.
* **REQ-OUT-05:** The system shall redirect the user back to the Login Page immediately after successful logout.

## Non-Functional Requirements

 **Security:**

* Passwords must never be stored in plain text; they must be hashed using industry-standard algorithms (e.g., BCrypt).
* API endpoints for the Dashboard must require a valid authentication token.

 **Performance:**

* Login and Registration requests should be processed within 2 seconds under normal network conditions.

 **Reliability:**

* The system should prevent data inconsistency by using database transactions during User creation.

 **Usability:**

* The user interface should be responsive and provide clear feedback (success/error messages) for all user actions.

 **Maintainability:**

* The code must follow the Layered Architecture (Controller-Service-Repository) to facilitate future updates and debugging.

## System Models (Diagrams)

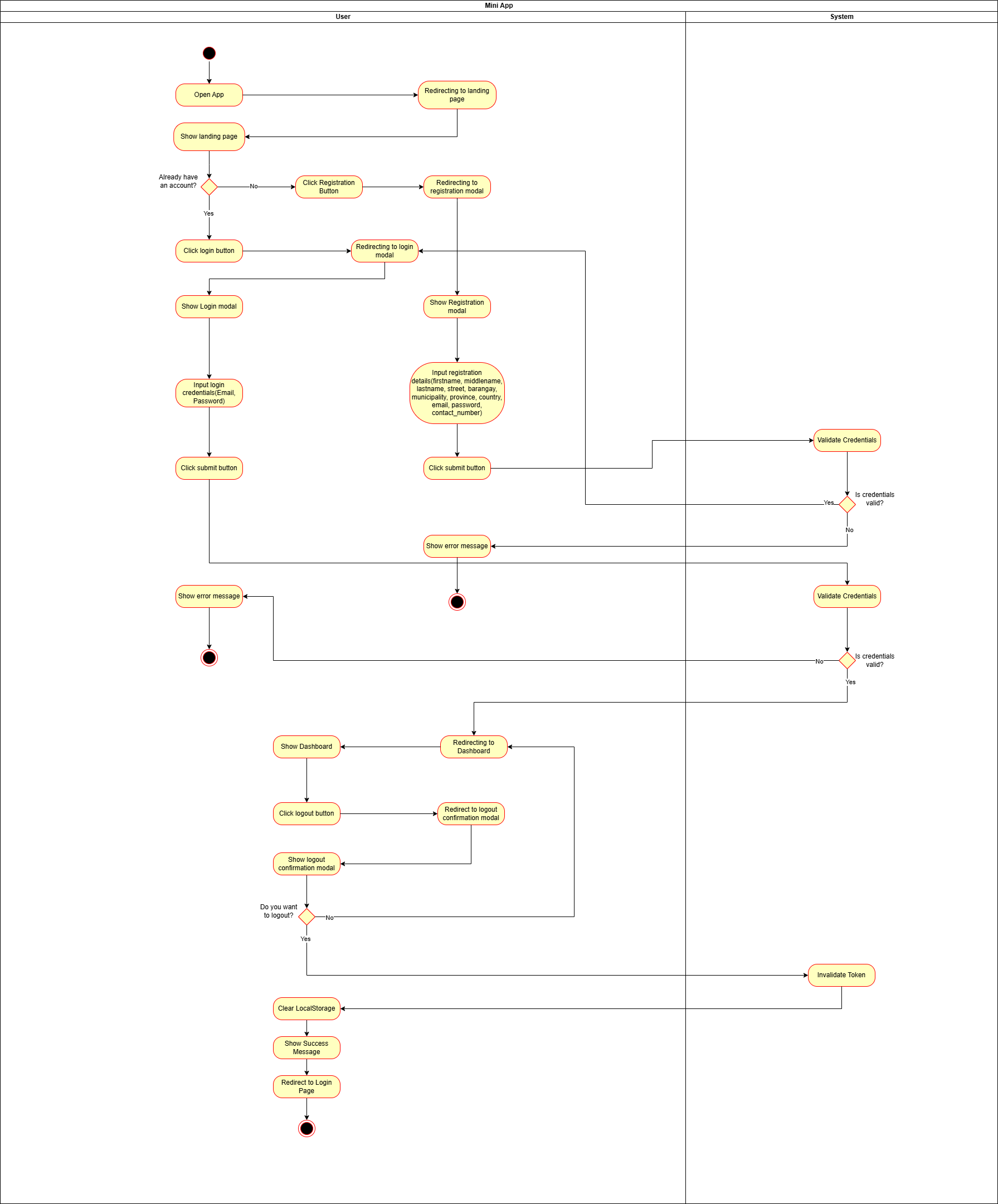
### ERD

### 

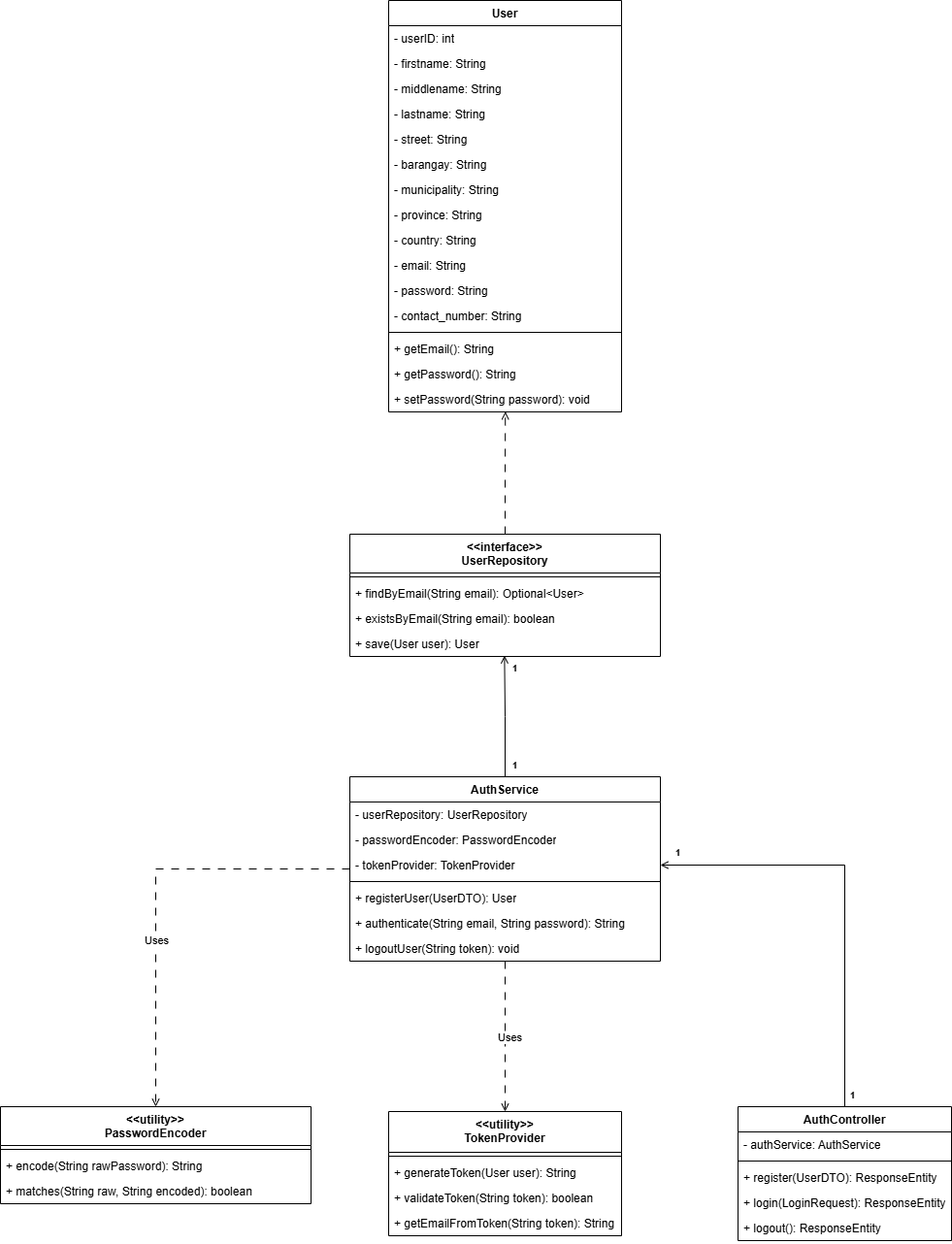
### Use Case Diagram

### 

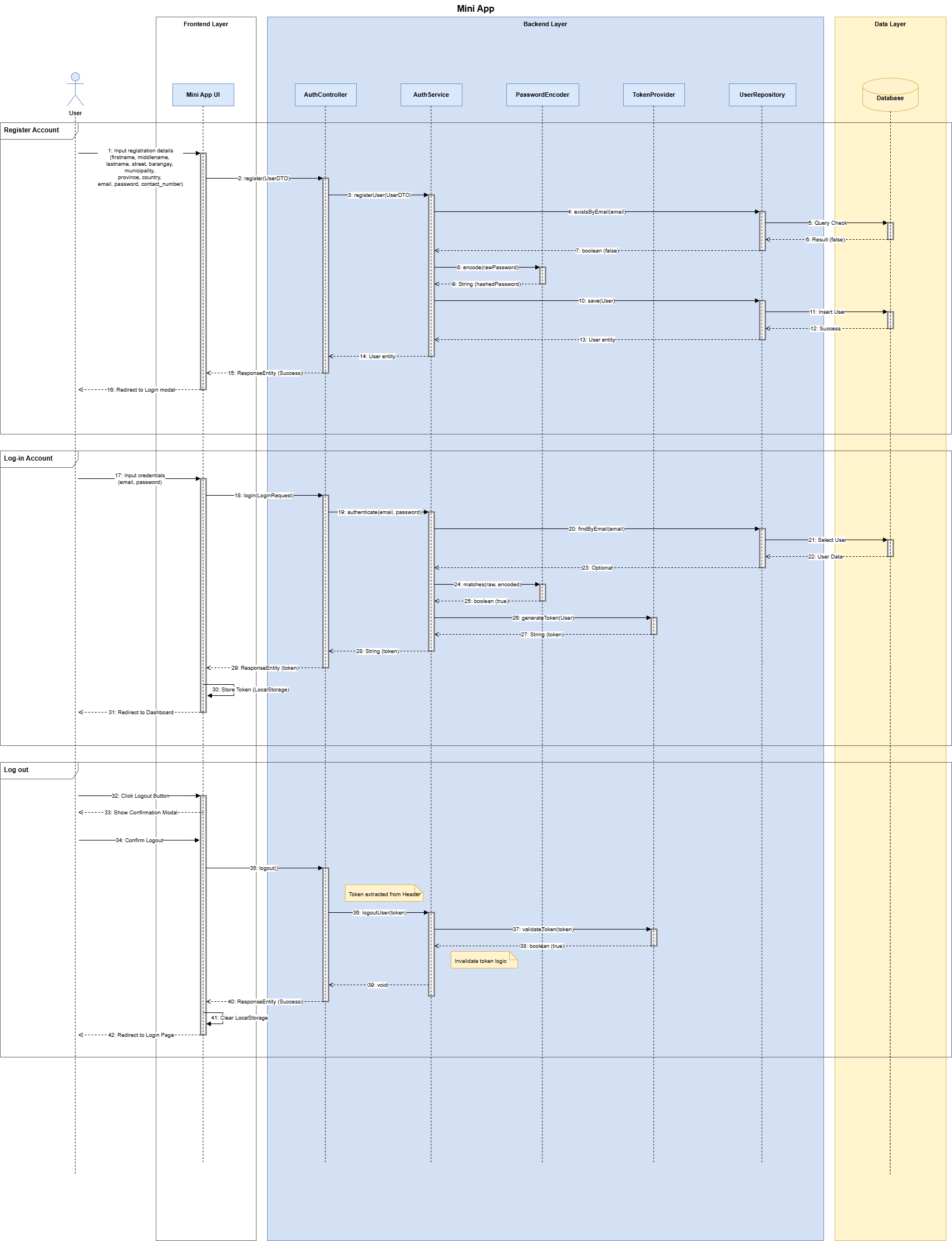
### Activity Diagram



### Class Diagram



### Sequence Diagram



## Appendices

**A. Tech Stack Reference:**

* Backend Framework: Spring Boot (Java)
* Frontend Library: ReactJS
* Database: MySQL

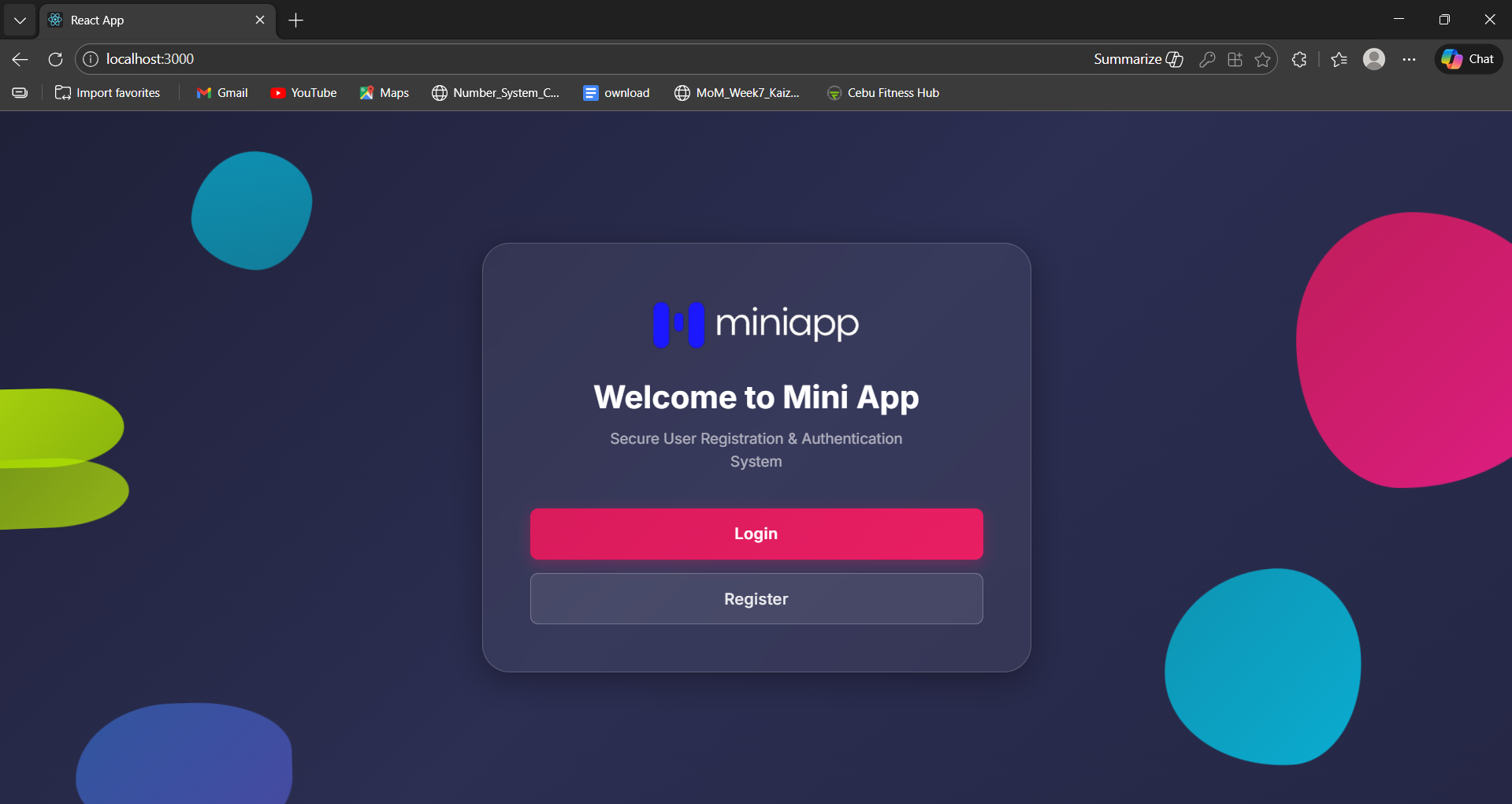
**B. Security Standards:**

* Password Encoding: BCrypt
* Token Standard: JWT (JSON Web Token) implementation guidelines.

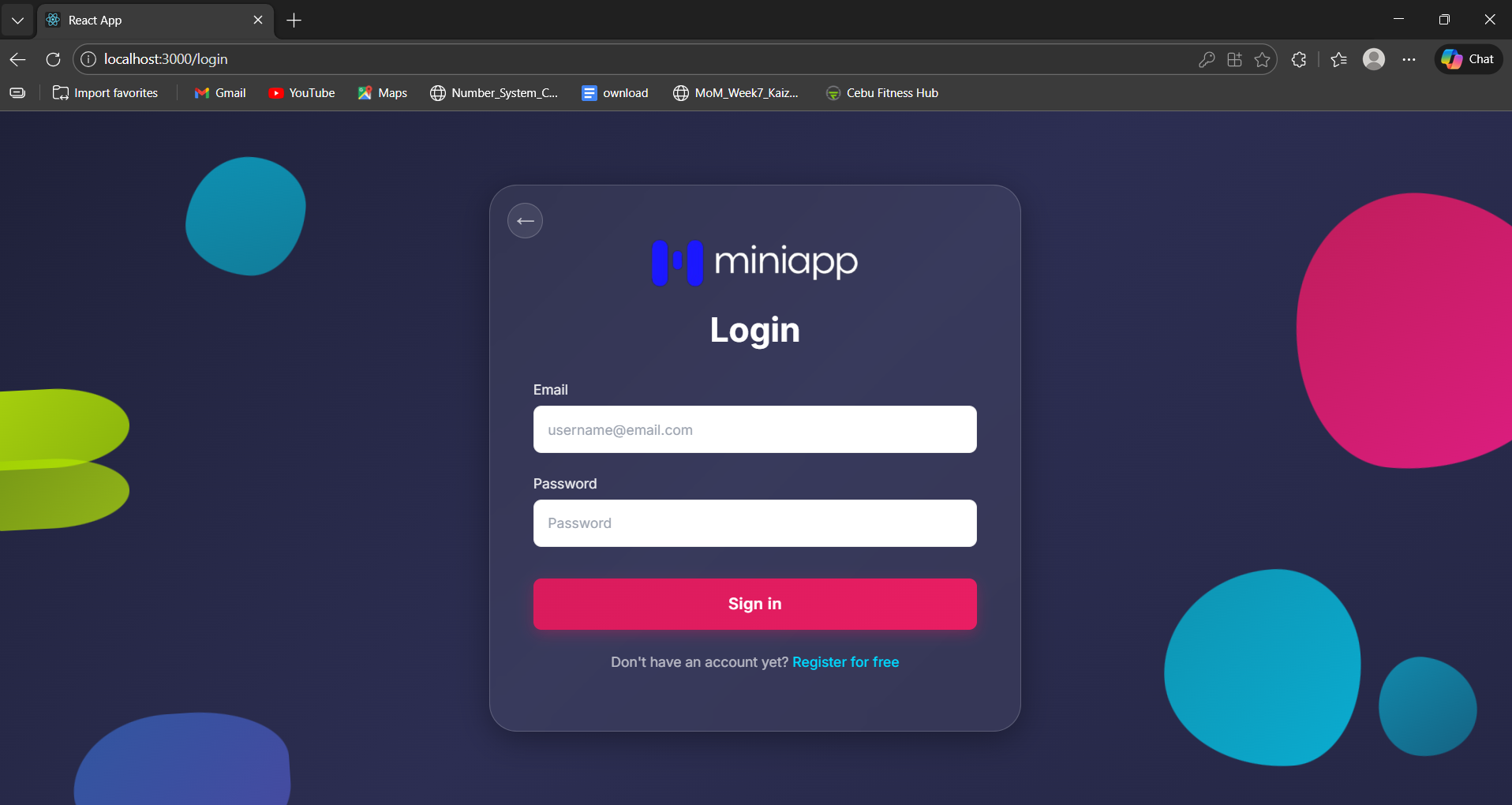
## Documentations

Website Screenshot:

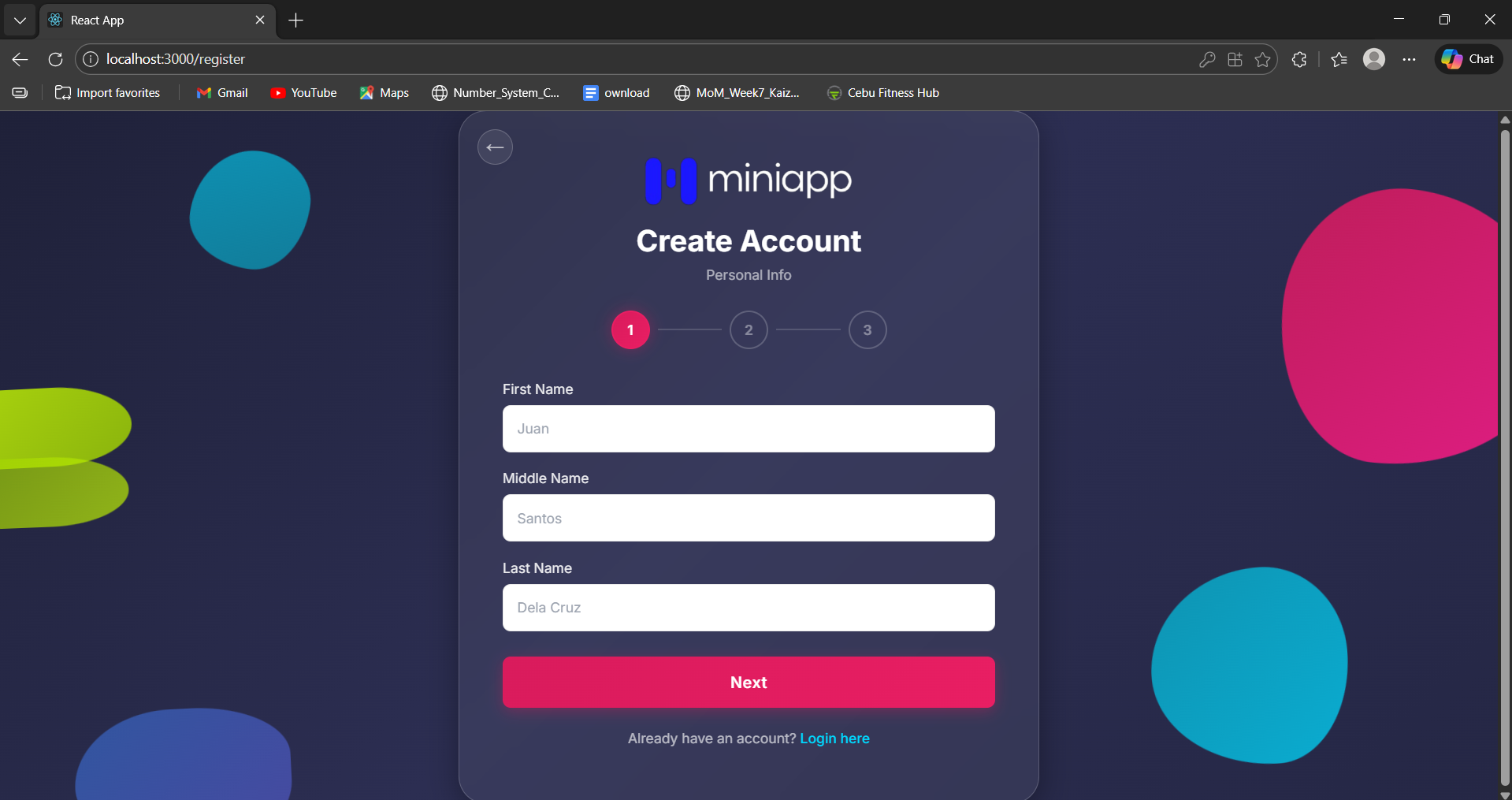
Landing page

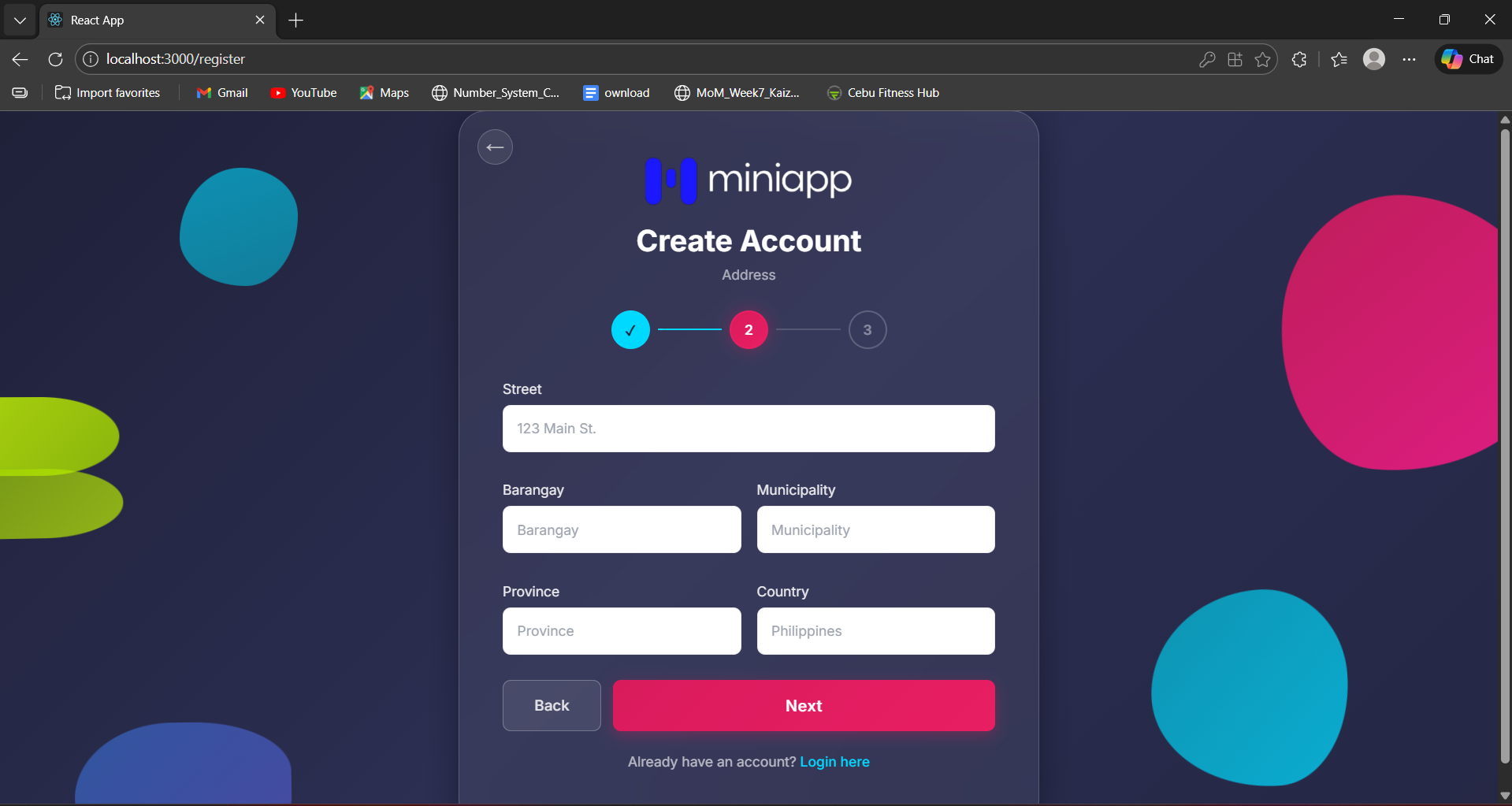


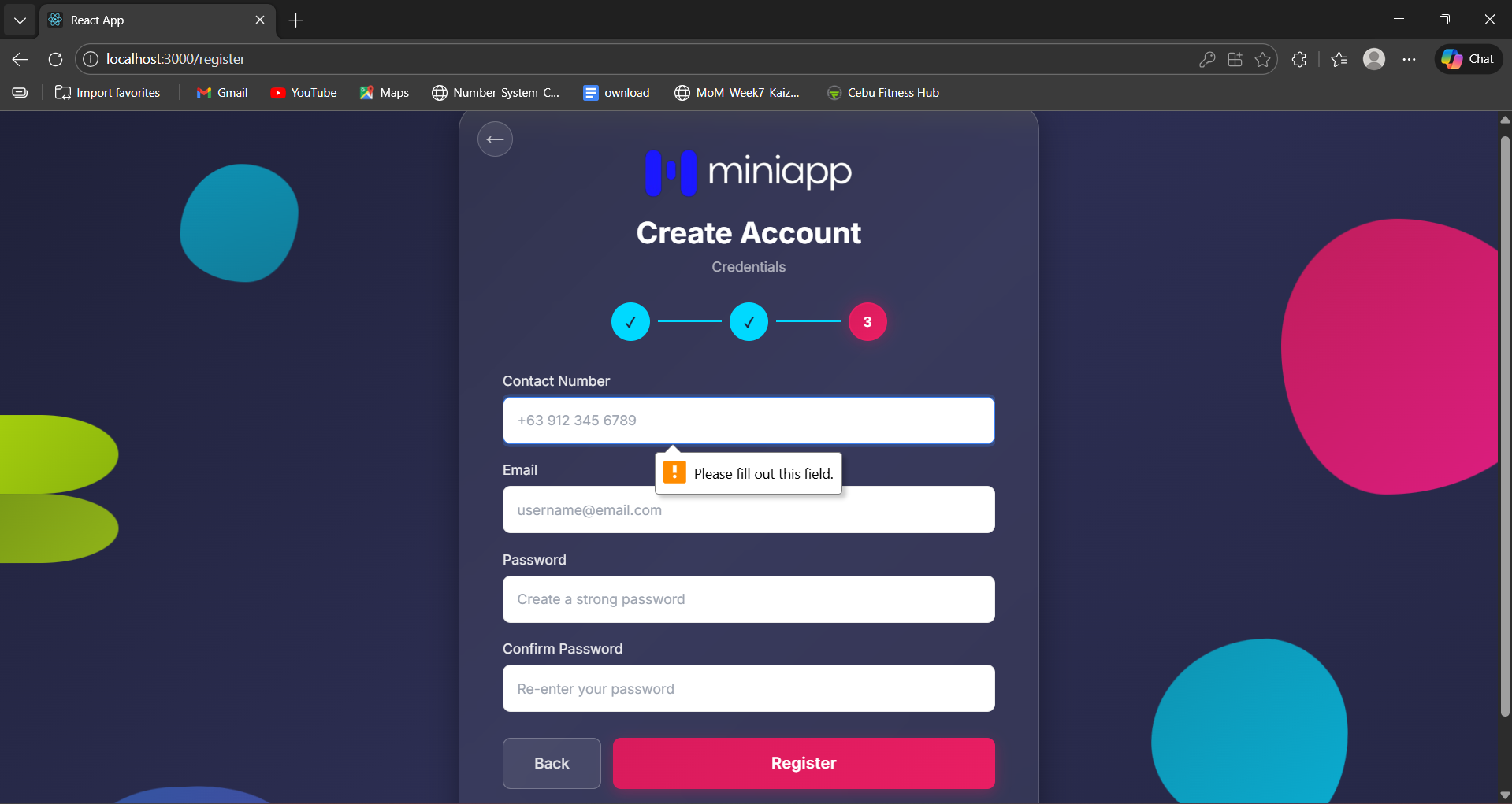
Login Page



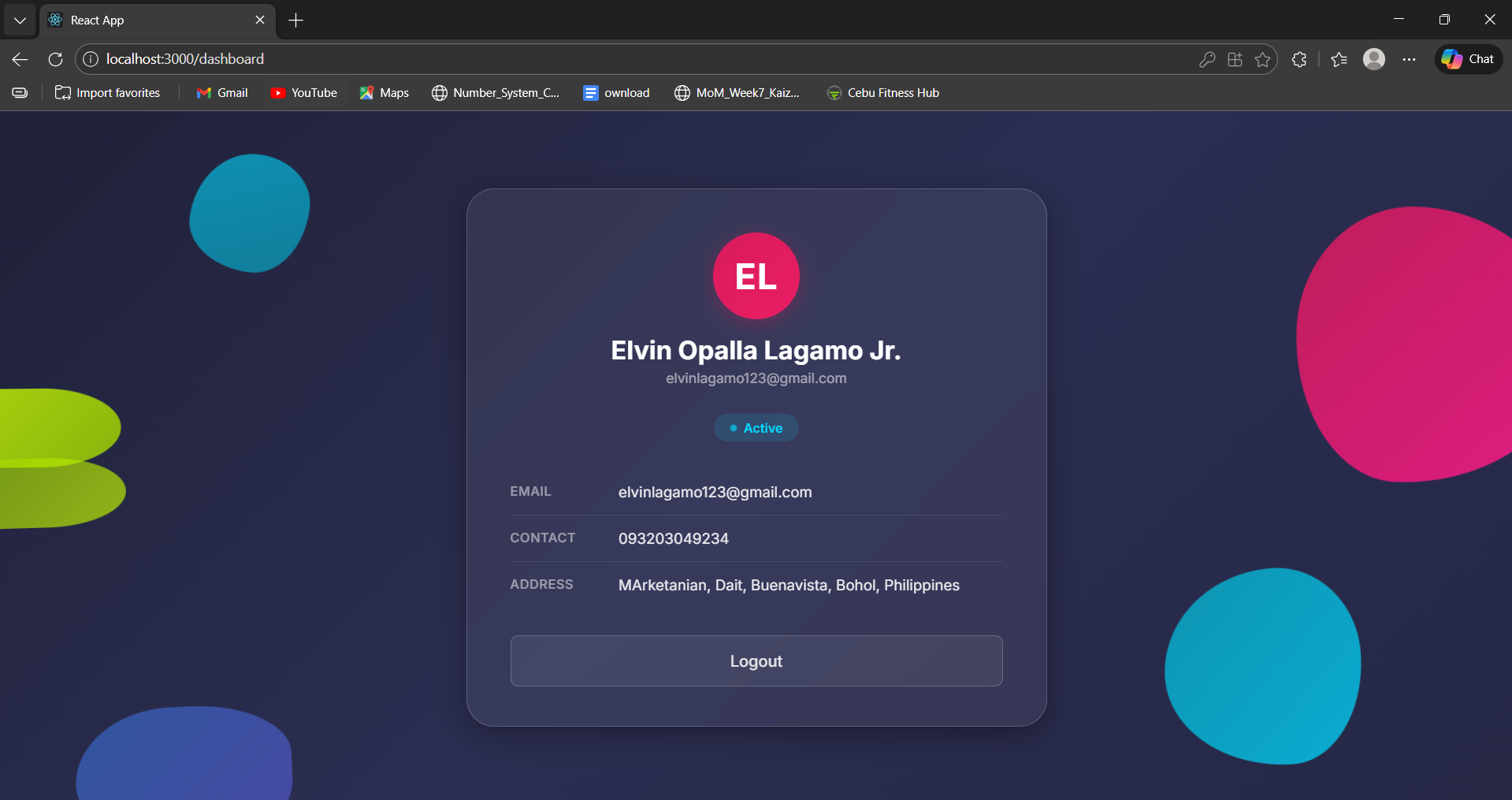
Registration Page



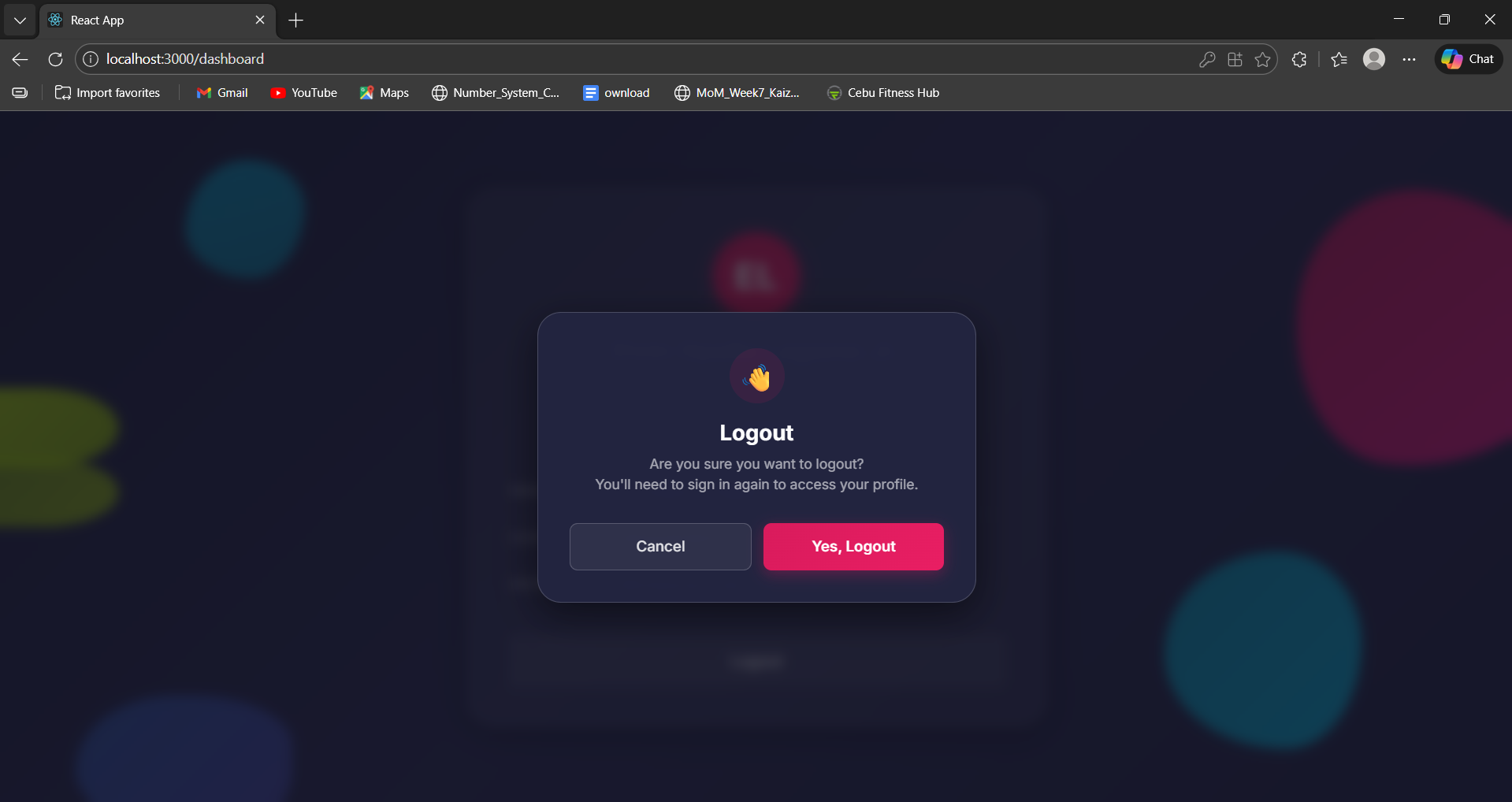


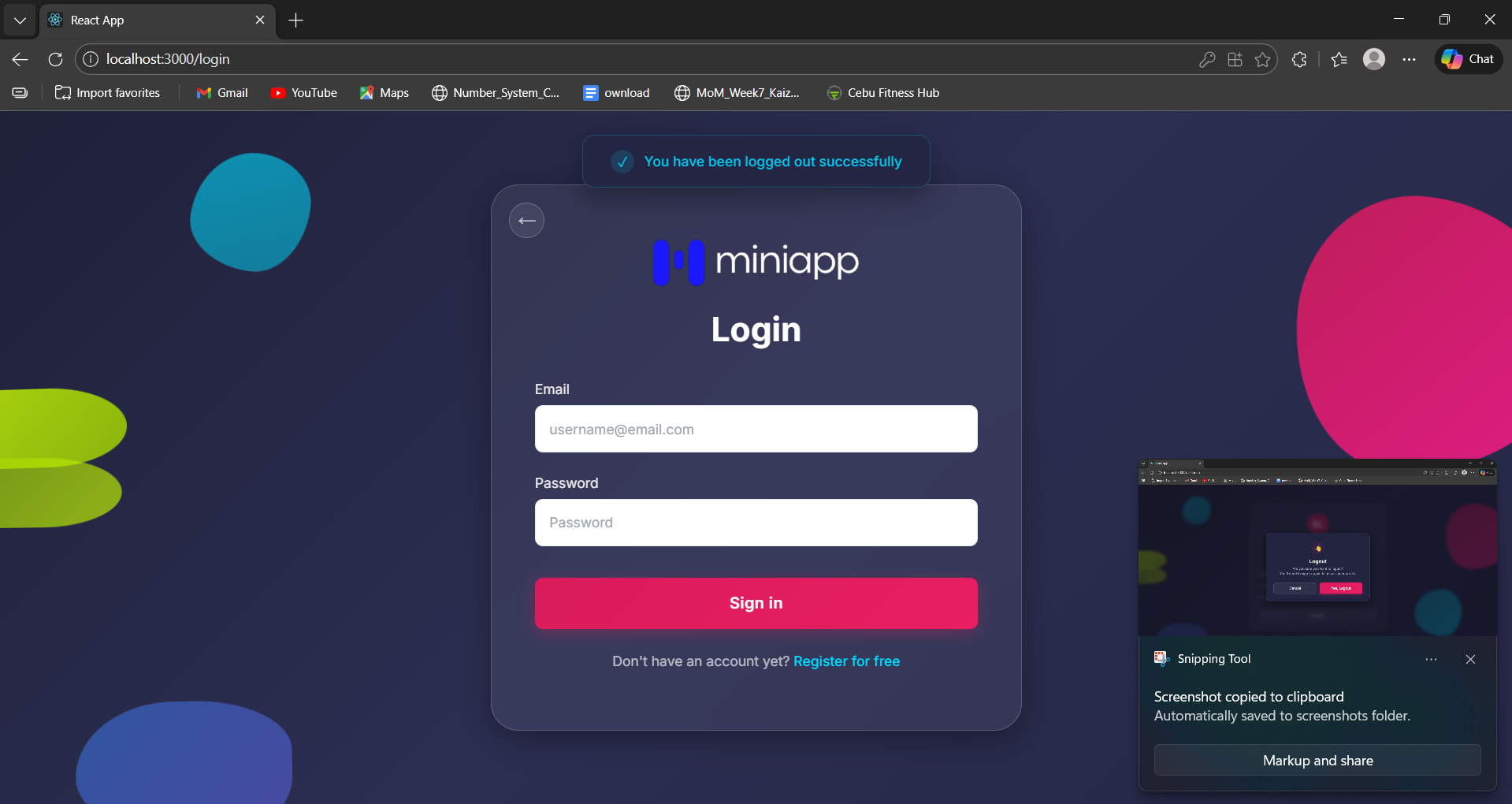


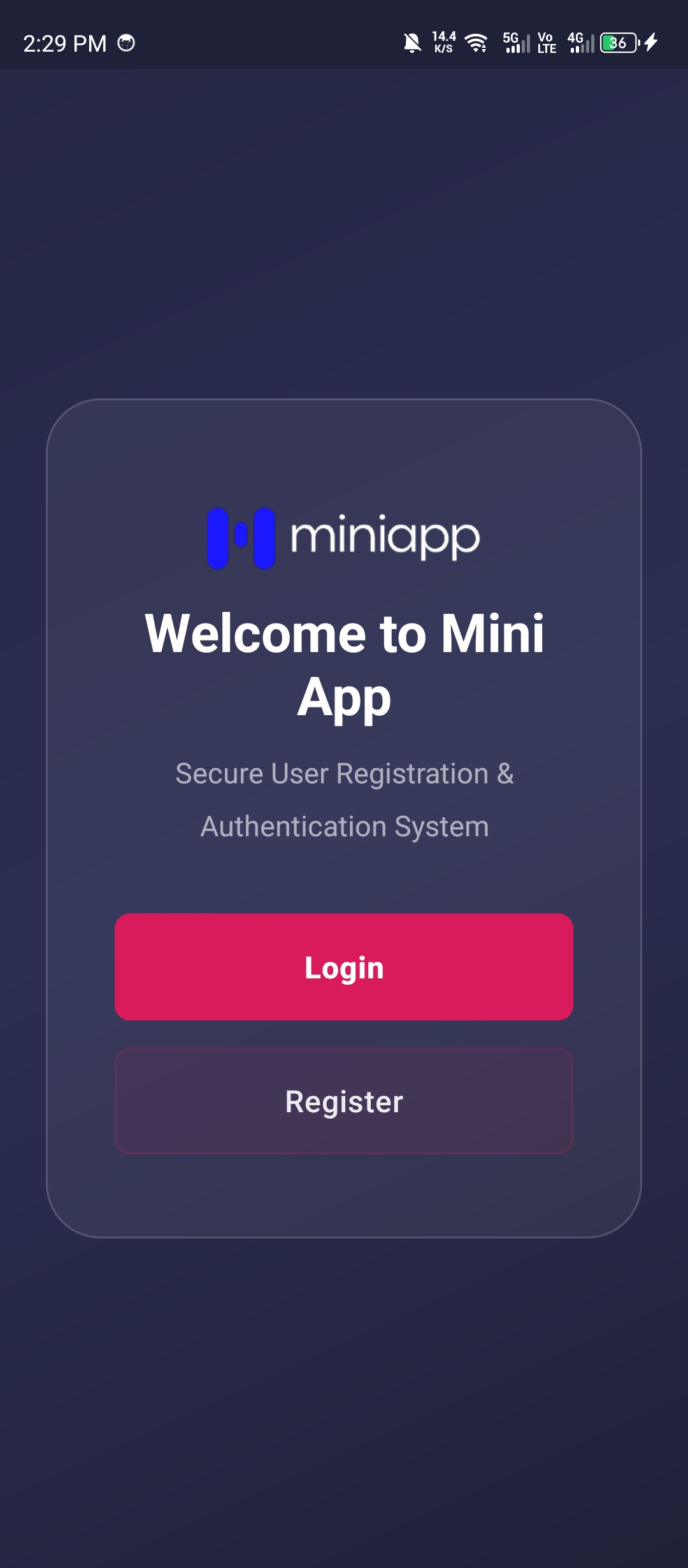
Dashboard/Profile page with Logout Button

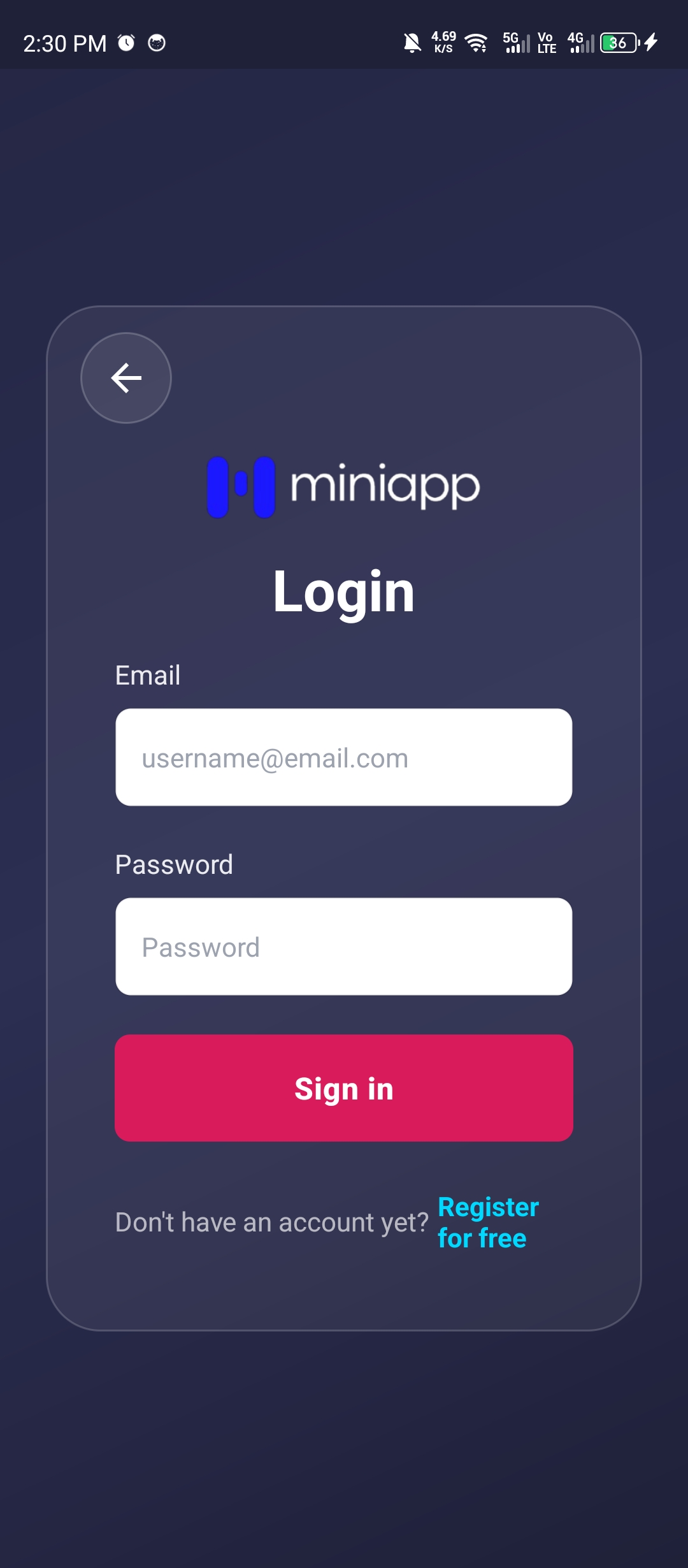
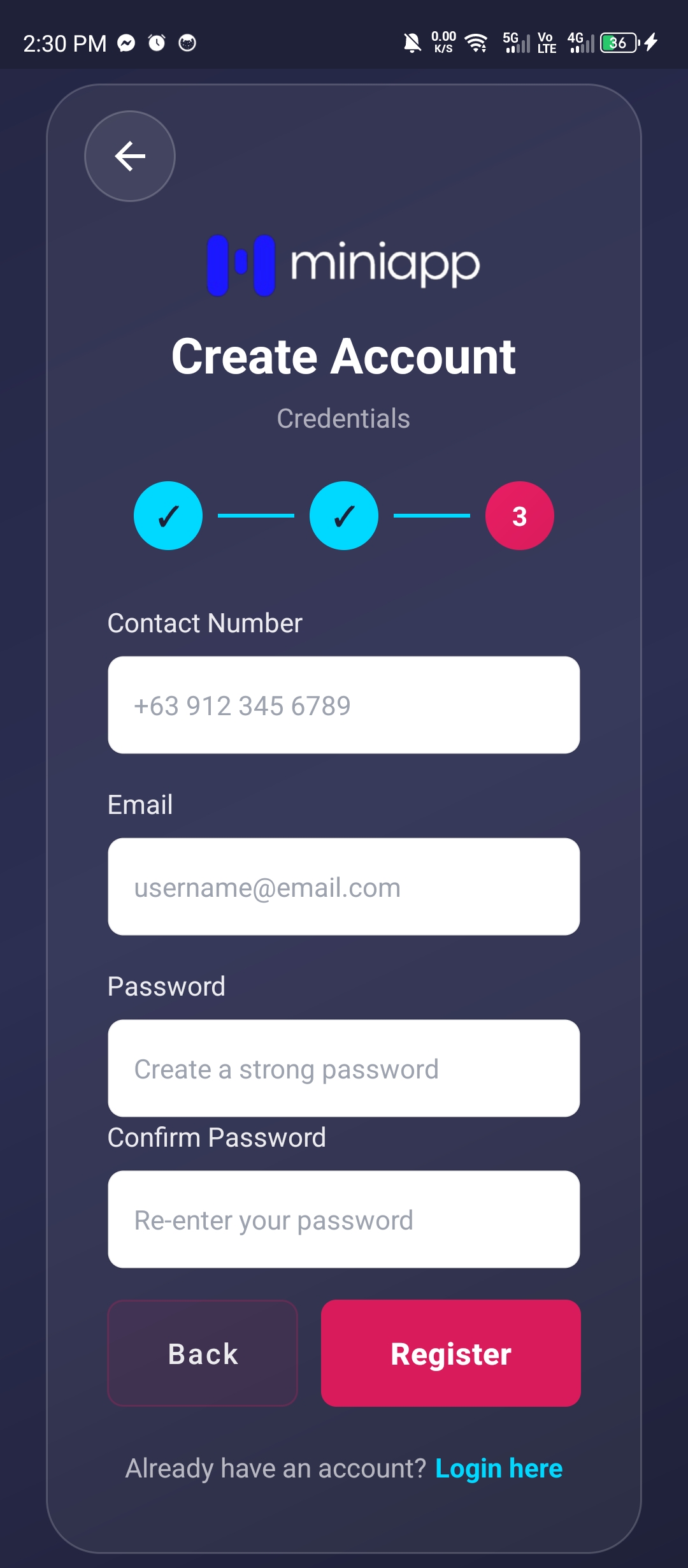
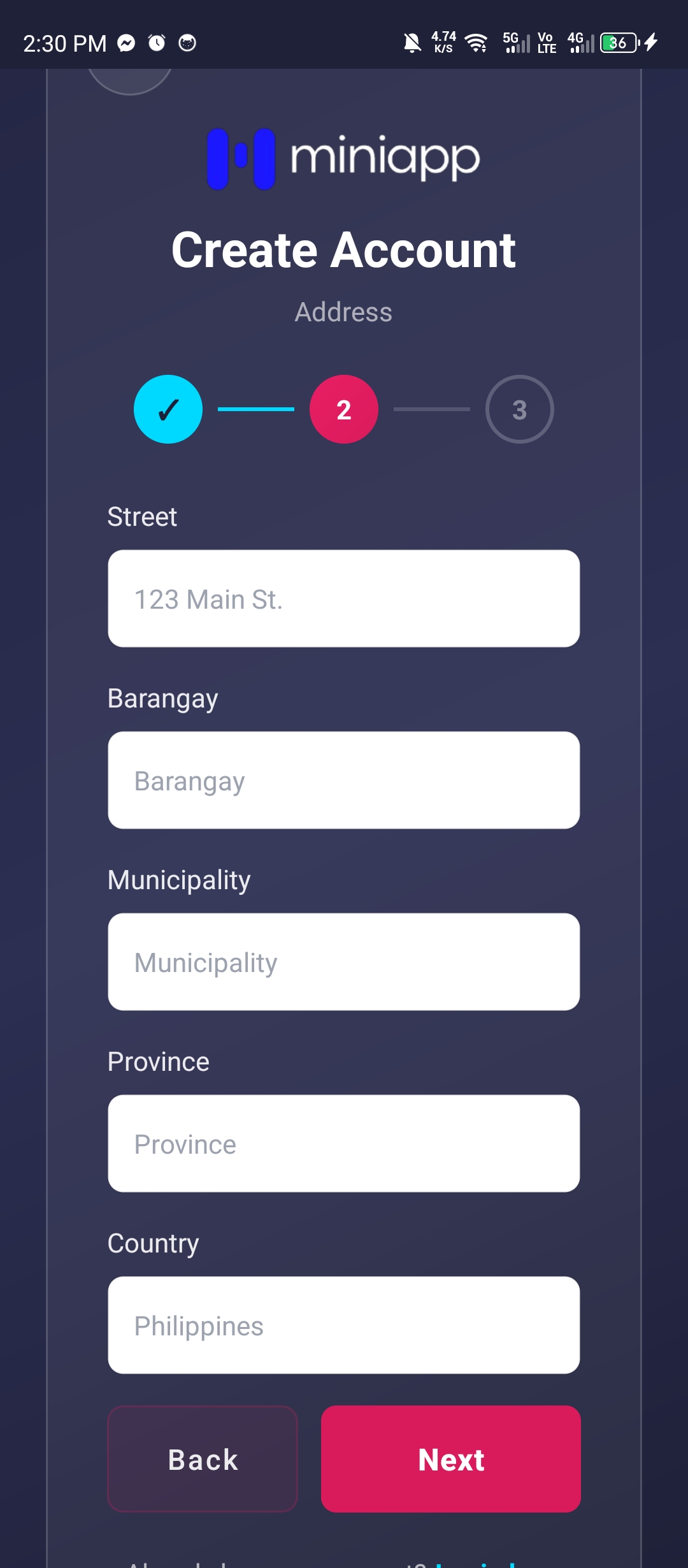
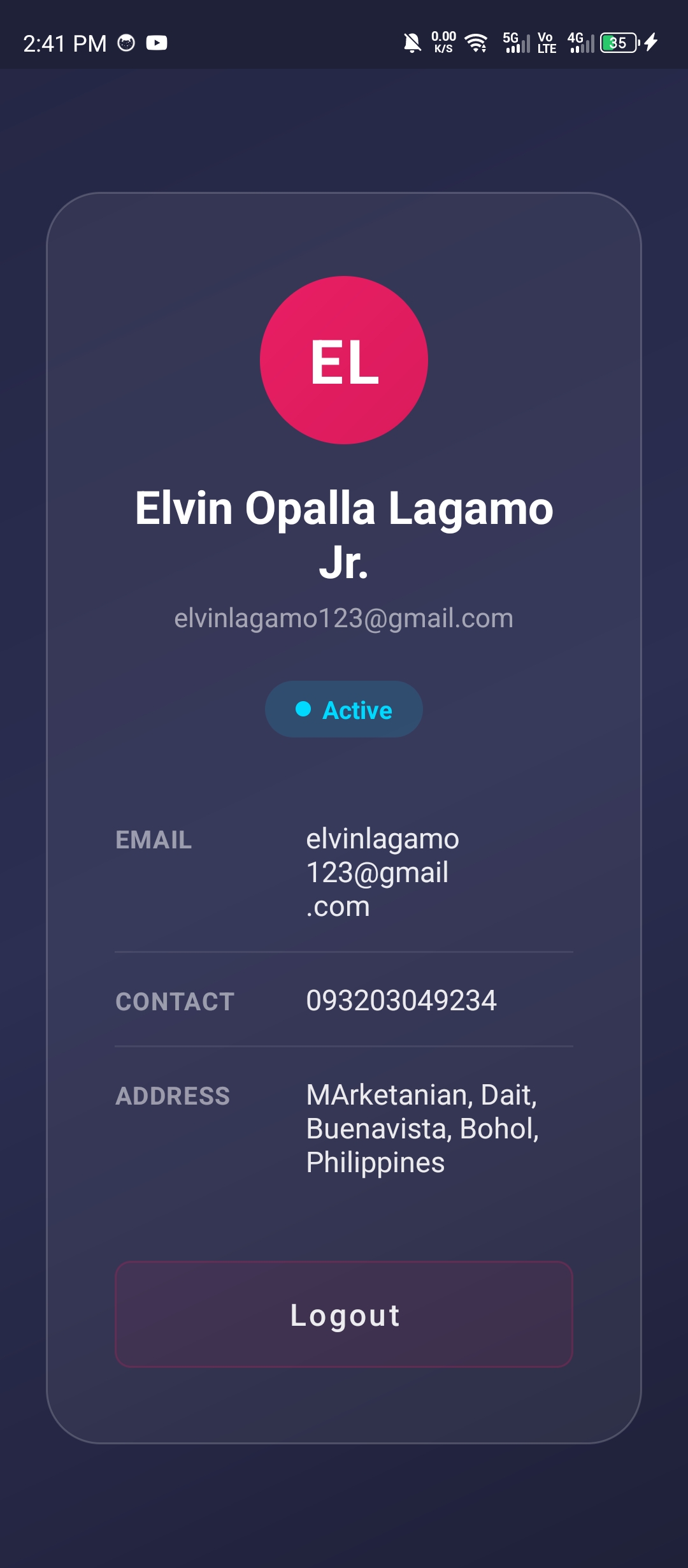


Logout Confirmation Modal w/ Success notification





Mobile Screenshots  
Landing Page  


Login Page  
  
  
Registration Page  
  
  
  
Dashboard/Profile Page w/ Logout button  
  
Logout Confirmation Modal w/ Success notification  
