CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

# IT342-G4
# SYSTEMS INTEGRATION AND ARCHITECTURE 1

# FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

Project Title: Mini Authentication Application

Prepared By: Campilanan, Reenah Mae R.

Date of Submission: 02/03/2026

Version: 1.0

# Table of Contents

# 1. Introduction

## 1.1. Purpose

The purpose of this document is to describe the functional and non-functional requirements of the Mini Authentication System. This document serves as a reference for developers, instructors, and stakeholders to understand the system's behavior, features, and design.

## 1.2. Scope

Mini Authentication System allows users to:

- Register an account
- Log in using valid credentials
- View a protected dashboard/profile
- Log out securely

The system ensures that protected pages cannot be accessed without proper authentication. This system focuses only on basic authentication features and does not include advanced features such as password reset or multi-factor authentication.

## 1.3. Definitions, Acronyms, and Abbreviations

API – Application Programming Interface

JWT – JSON Web Token

UI – User Interface

ERD – Entity Relationship Diagram

UML – Unified Modeling Language

Authentication – Process of verifying user identity

Authorization – Process of granting access to resources

# 2. Overall Description

## 2.1. System Perspective

Mini Authentication System is a web-based system consisting of:

- React UI (Frontend)
- Spring Boot API (Backend)
- Relational Database (User data storage)

The system follows a three-layer architecture:

- Presentation Layer (React UI)
- Business Logic Layer (Spring Boot API)
- Data Layer (Database)

## 2.2. User Classes and Characteristics

### 1. Guest User

- Can access registration and login pages
- Cannot access protected pages
- No authentication required

### 2. Authenticated User

- Successfully logged-in user
- Can access dashboard/profile
- Can log out
- Requires valid authentication token

## 2.3. Operating Environment

Frontend: React (Web Browser)

Backend: Spring Boot

Database: MySQL or PostgreSQL

Tools: diagrams.net (for modeling)

Operating System: Windows

Browser: Google Chrome or any modern browser

## 2.4. Assumptions and Dependencies

The system assumes users have internet access.

The database server is running and accessible.

Users provide valid email and password during registration.

Authentication is handled using token-based mechanism (e.g., JWT).

No external third-party authentication provider is used.

# 3. System Features and Functional Requirements

## 3.1. Feature 1: User Registration

**Description:** Allows a guest user to create a new account in the system.

**Functional Requirements:**

- The system shall allow a guest user to enter username, email, and password.
- The system shall validate user input.
- The system shall check if the username or email already exists.
- The system shall hash the password before saving.
- The system shall store the new user in the database.
- The system shall return a success message if registration is successful.
- The system shall return an error message if the username or email already exists.

## 3.2 Feature 2: User Login

**Description:** Allows a registered user to log into the system using valid credentials.

**Functional Requirements:**

- The system shall allow a user to enter identifier (email/username) and password.
- The system shall retrieve the user record from the database.
- The system shall validate the password using password hashing.
- The system shall generate an authentication token upon successful login.
- The system shall return a 401 Unauthorized response if credentials are invalid.
- The system shall allow access to protected pages only after successful login.

## 3.3 Feature 3: View Dashboard/ Profile

**Description:** Allows an authenticated user to access protected resources.

**Functional Requirements:**

- The system shall require a valid authentication token to access protected endpoints.
- The system shall validate the token before processing the request.
- The system shall retrieve user information from the database.
- The system shall return 401 Unauthorized if the token is invalid or missing.
- The system shall display profile/dashboard data if the token is valid.

## 3.4 Feature 4: User Logout

**Description:** Allows an authenticated user to terminate their session.

**Functional Requirements:**

- The system shall accept a logout request.
- The system shall invalidate the authentication token.
- The system shall terminate the user session.
- The system shall redirect the user to the login page.
- The system shall prevent access to protected pages after logout.

## 4. Non-Functional Requirements

**1. Performance**

- The system shall respond to authentication requests within 2 seconds.

**2. Security**

- Passwords shall be stored in hashed format.
- Authentication shall use token-based mechanism.
- Protected endpoints shall require valid authentication tokens.

**3. Usability**

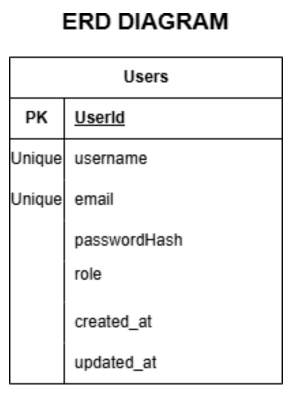
- The system shall provide clear error messages.
- The interface shall be simple and easy to use.

**4. Reliability**

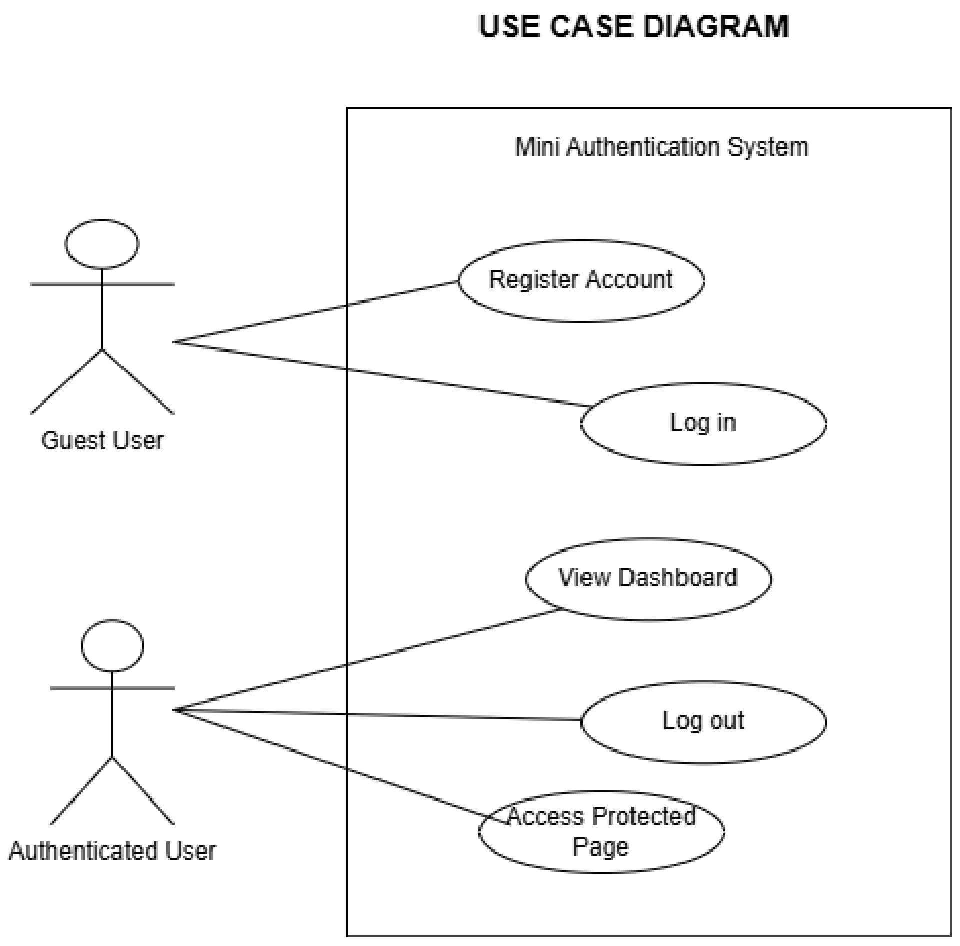
- The system shall maintain consistent database integrity.
- User data shall not be lost during normal operation.

## 5. System Models (Diagrams)

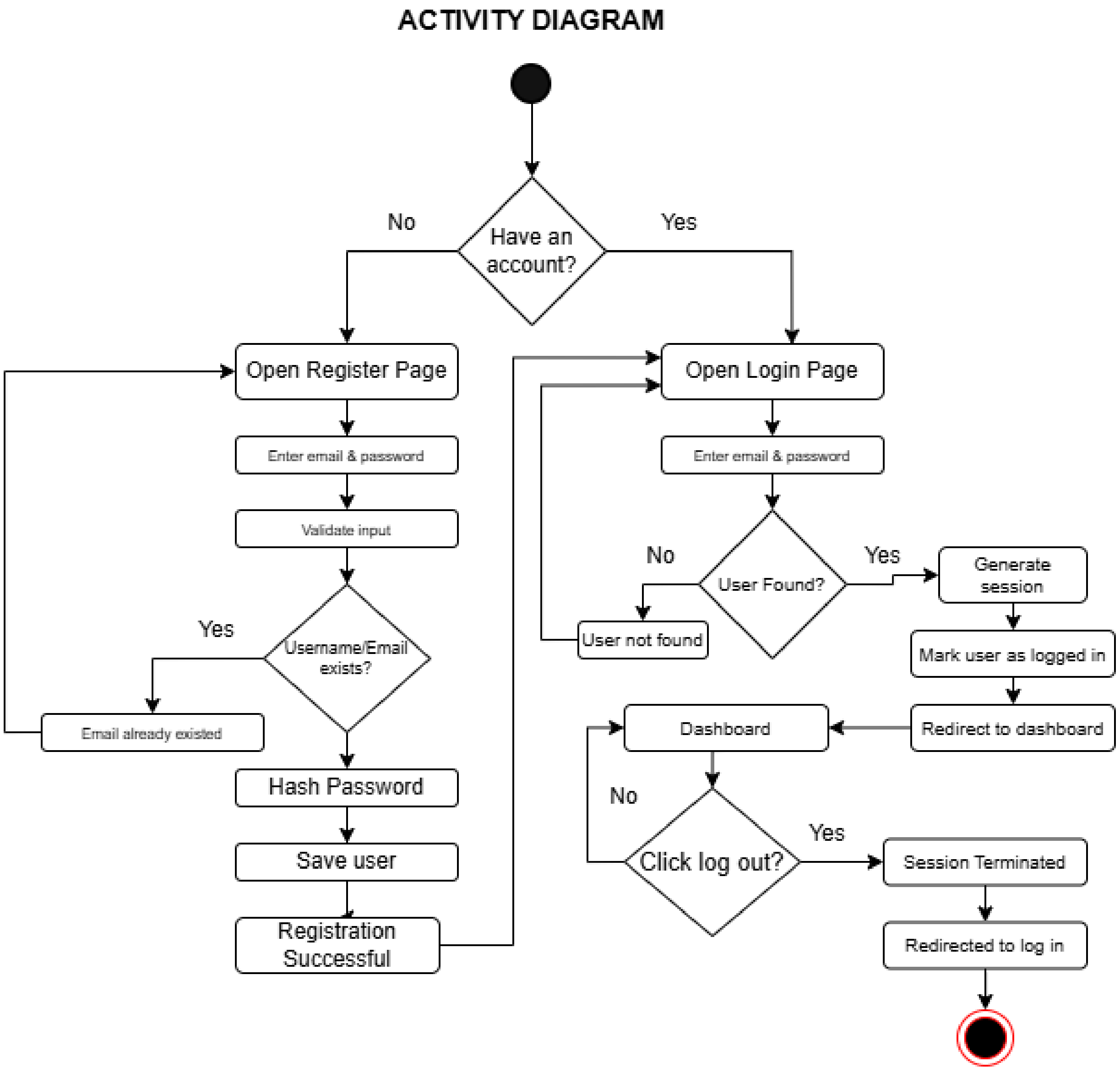
### 5.1. ERD



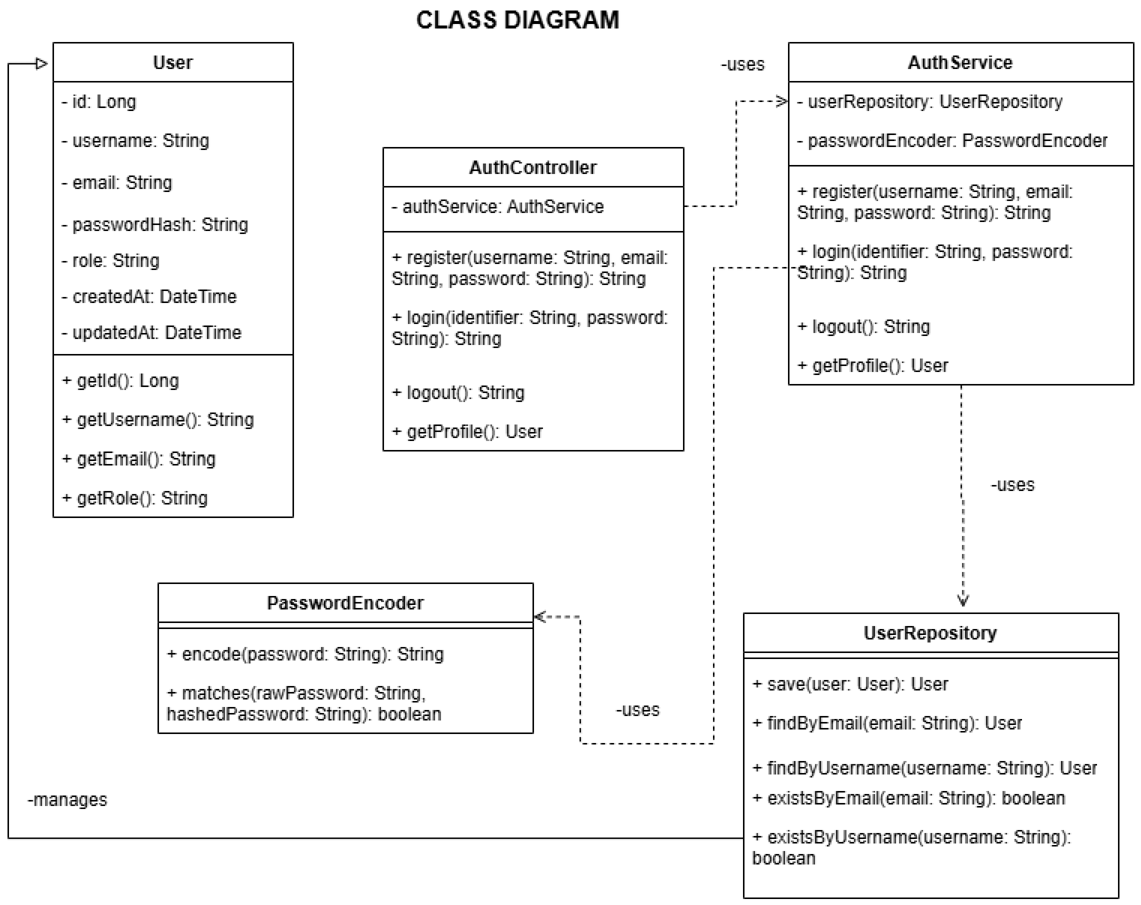
### 5.2. Use Case Diagram



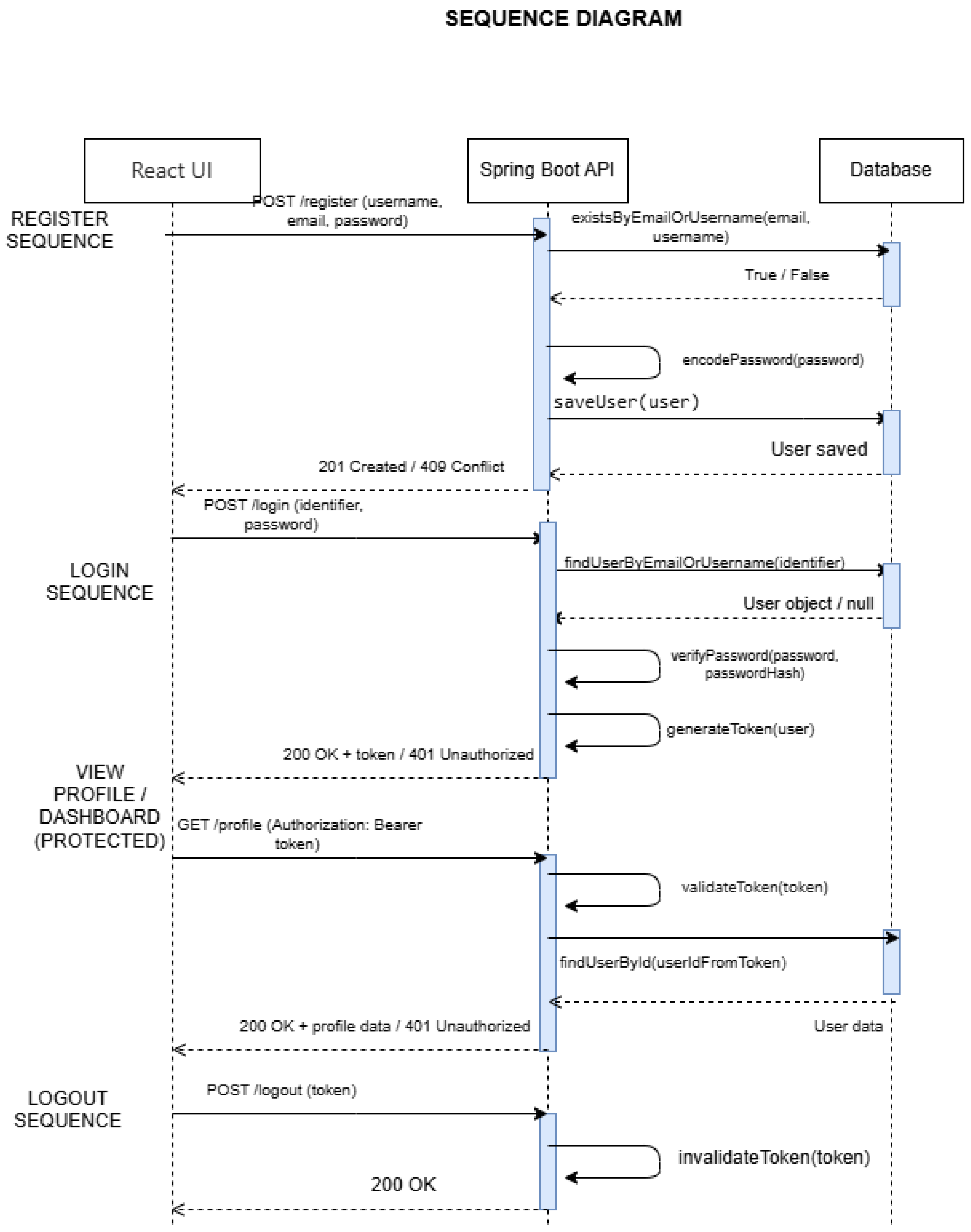
## 5.3. Activity Diagram



## 5.4. Class Diagram


CLASS DIAGRAM
User
- id: Long
- username: String
- email: String
- passwordHash: String
- role: String
- createdAt: DateTime
- updatedAt: DateTime
+ getId(): Long
+ getUsername(): String
+ getEmail(): String
+ getRole(): String
AuthController
- authService: AuthService
+ register(username: String, email: String, password: String): String
+ login(identifier: String, password: String): String
+ logout(): String
+ getProfile(): User
-uses
AuthService
- userRepository: UserRepository
- passwordEncoder: PasswordEncoder
+ register(username: String, email: String, password: String): String
+ login(identifier: String, password: String): String
+ logout(): String
+ getProfile(): User
-uses
PasswordEncoder
+ encode(password: String): String
+ matches(rawPassword: String, hashedPassword: String): boolean
-uses
UserRepository
+ save(user: User): User
+ findByEmail(email: String): User
+ findByUsername(username: String): User
+ existsByEmail(email: String): boolean
+ existsByUsername(username: String): boolean
-manages

## 5.5. Sequence Diagram


SEQUENCE DIAGRAM
React UI
Spring Boot API
Database
REGISTER SEQUENCE
POST /register (username, email, password)
existsByEmailOrUsername(email, username)
True / False
encodePassword(password)
saveUser(user)
User saved
201 Created / 409 Conflict
LOGIN SEQUENCE
POST /login (identifier, password)
findUserByEmailOrUsername(identifier)
User object / null
verifyPassword(password, passwordHash)
generateToken(user)
200 OK + token / 401 Unauthorized
VIEW PROFILE / DASHBOARD (PROTECTED)
GET /profile (Authorization: Bearer token)
validateToken(token)
findUserById(userIdFromToken)
User data
200 OK + profile data / 401 Unauthorized
LOGOUT SEQUENCE
POST /logout (token)
invalidateToken(token)
200 OK

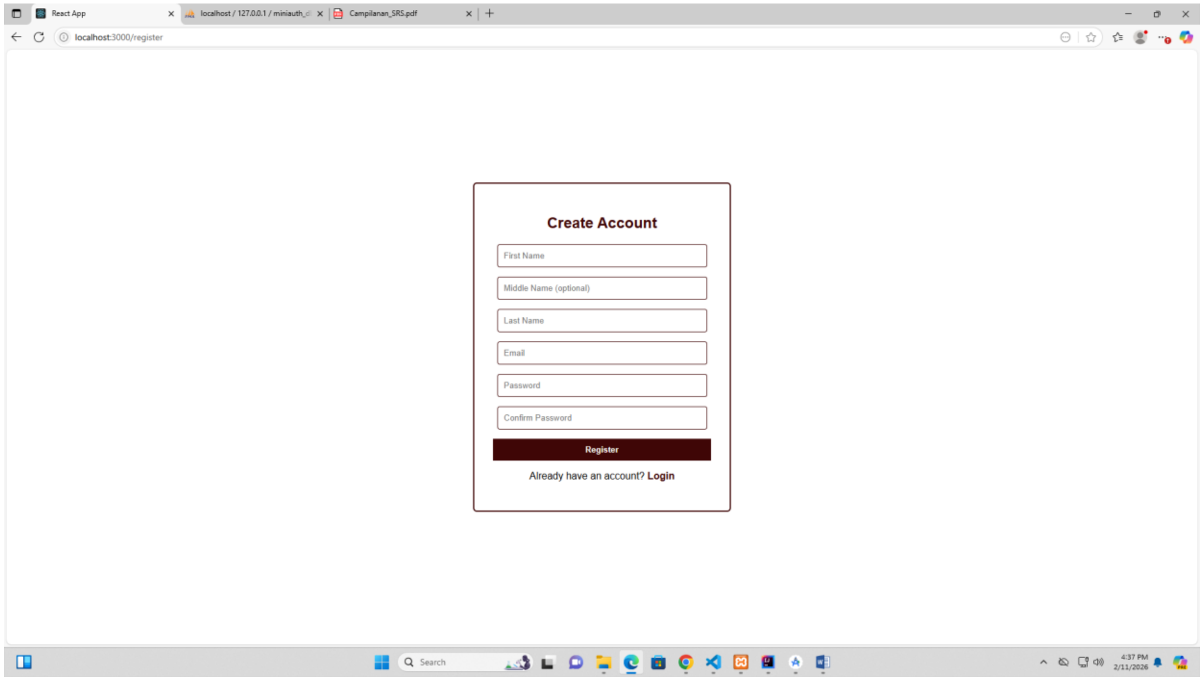
## 6.

## Appendices

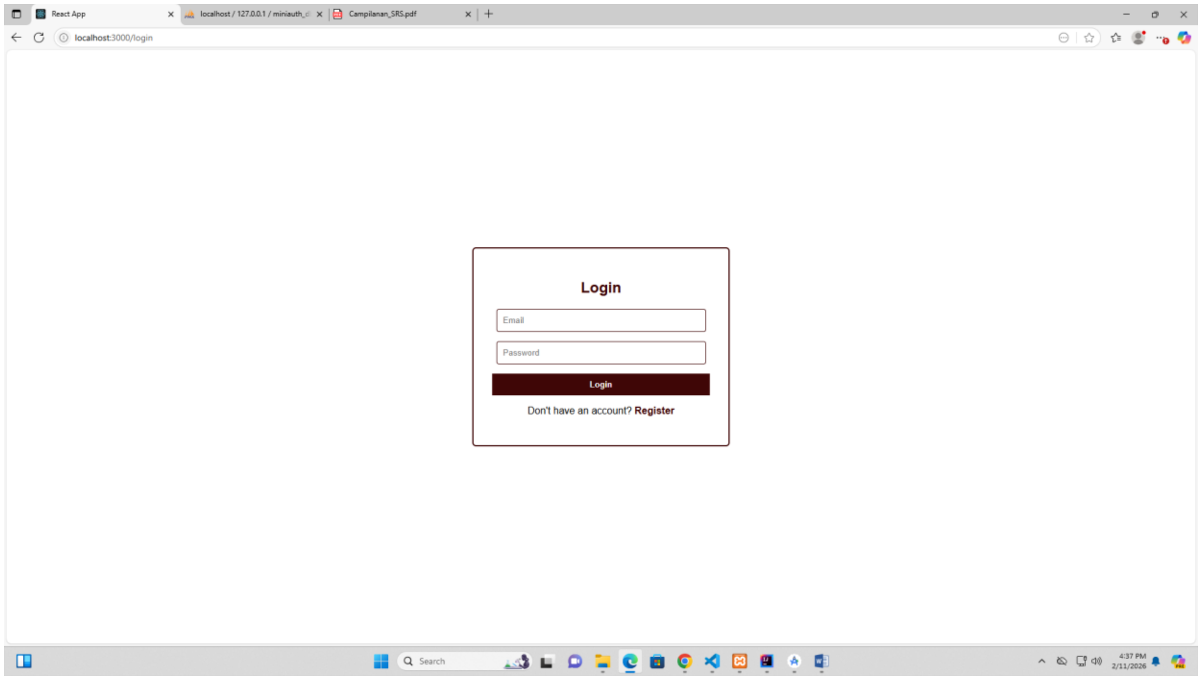
Include any additional information, references, or support materials.

**Updated Github Repository**: https://github.com/ReenahMae/IT342_G4_Campilanan_Lab1-.git

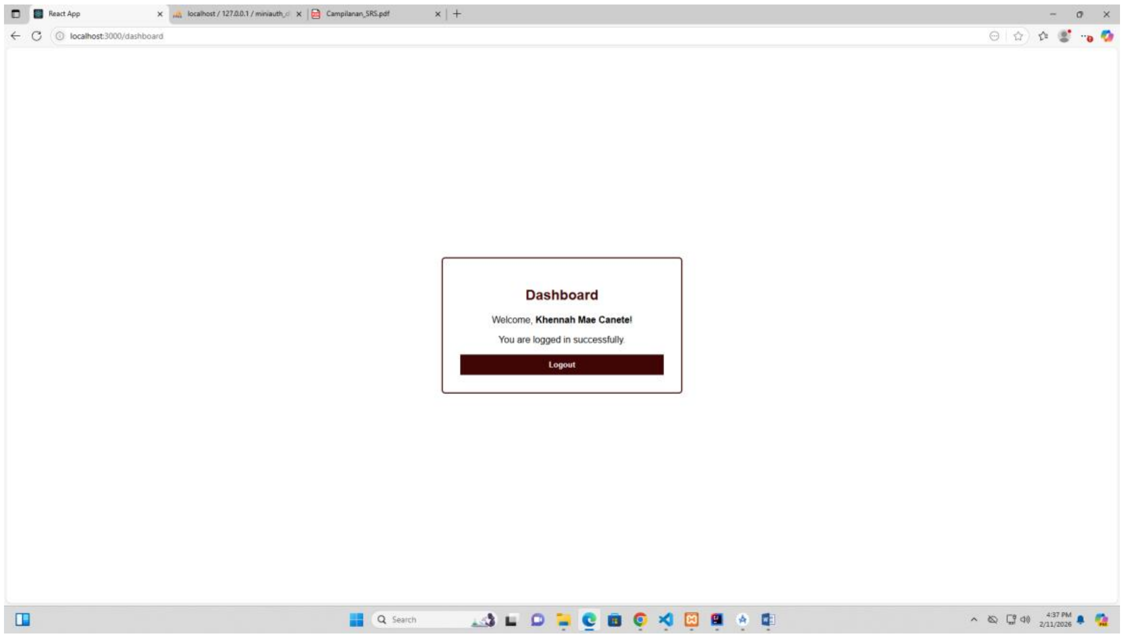
**Register:**



**Login:**



## Dashboard:

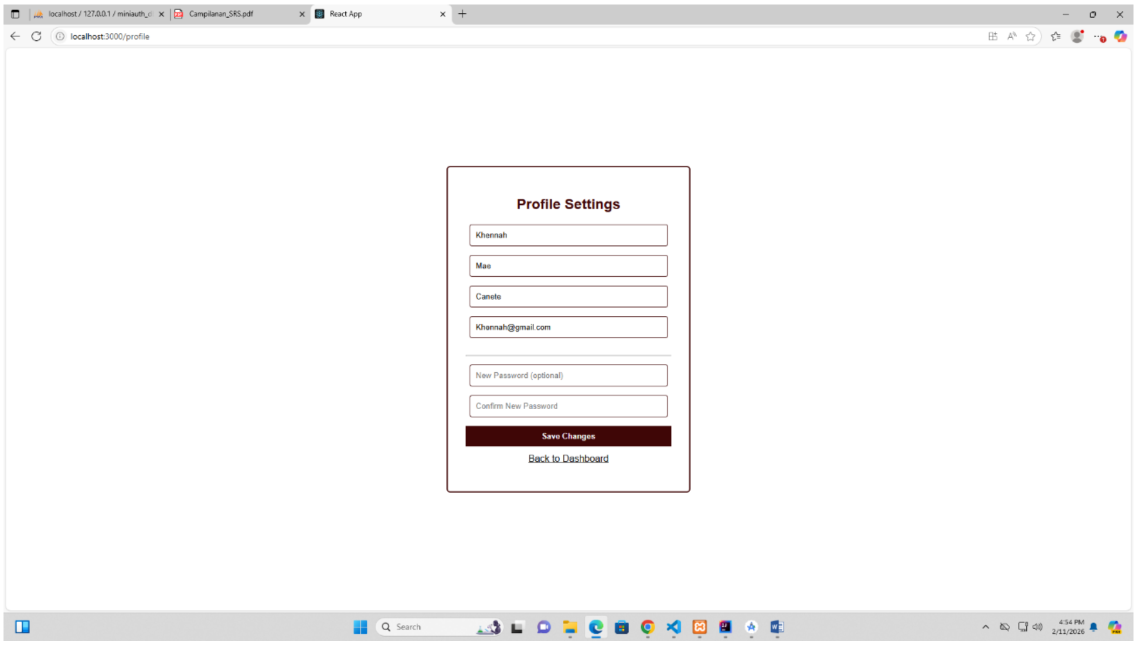


Dashboard

Welcome, **Khennah Mae Canete!**

You are logged in successfully.

## Profile:



Profile Settings

Khennah

Mae

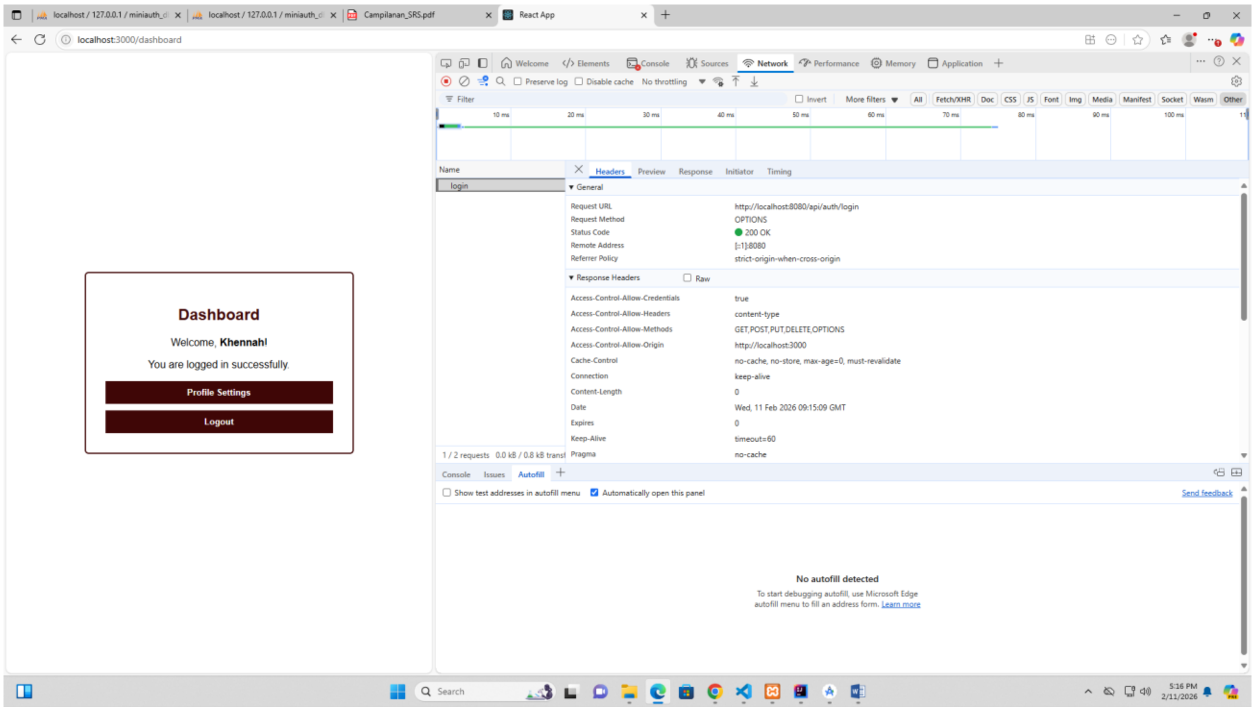
Canete

Khennah@gmail.com

New Password (optional)

Confirm New Password

## Proof of Integration for Web:



▾ General

| | |
|---|---|
| Request URL | http://localhost:8080/api/auth/login |
| Request Method | OPTIONS |
| Status Code | ● 200 OK |
| Remote Address | [::1]:8080 |
| Referrer Policy | strict-origin-when-cross-origin |

▾ Response Headers ☐ Raw

| | |
|---|---|
| Access-Control-Allow-Credentials | true |
| Access-Control-Allow-Headers | content-type |
| Access-Control-Allow-Methods | GET,POST,PUT,DELETE,OPTIONS |
| Access-Control-Allow-Origin | http://localhost:3000 |
| Cache-Control | no-cache, no-store, max-age=0, must-revalidate |
| Connection | keep-alive |
| Content-Length | 0 |
| Date | Wed, 11 Feb 2026 09:15:09 GMT |
| Expires | 0 |
| Keep-Alive | timeout=60 |
| Pragma | no-cache |

1 / 2 requests 0.0 kB / 0.8 kB transf

Console
Issues
Autofill
Show test addresses in autofill menu
Automatically open this panel
Send feedback

No autofill detected

To start debugging autofill, use Microsoft Edge autofill menu to fill an address form. Learn more