CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342-G4 SYSTEMS INTEGRATION AND ARCHITECTURE 1

FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

Project Title: Mini App – User Registration & Authentication

Prepared By: Jon Vic T. Barcenas

Date of Submission: February 3, 2026

Version: 1.0

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this document is to define the functional and non-functional requirements of a simple user authentication system.

1.2. Scope

The system allows users to register an account, log in, view their profile or dashboard, and log out. Access to protected pages is restricted to authenticated users only. This system focuses only on basic authentication features and does not include advanced functionalities such as roles, permissions, or account management.

1.3. Definitions, Acronyms, and Abbreviations

Term	Definition
DB	Database
UI	User Interface
ERD	Entity Relationship Diagram
API	Application Programming Interface
JWT	JSON Web Token

2. Overall Description

2.1. System Perspective

The system is a standalone web-based application composed of a frontend user interface and a backend server connected to a database. It handles user authentication and session management.

2.2. User Classes and Characteristics

User Type	Description
Guest User	A user who has not logged in and can only register or log in
Authenticated User	A logged-in user who can view their profile/dashboard and log out

2.3. Operating Environment

- Frontend: Web browser (Chrome, Edge, Firefox)
- Backend: Spring Boot REST API
- Database: Relational database (MySQL / PostgreSQL)
- Tools: draw.io / diagrams.net, GitHub, IDE

2.4. Assumptions and Dependencies

- Users have internet access
- Users provide valid email addresses
- The system depends on a database for storing user data
- Authentication is handled using session or token-based logic

3. System Features and Functional Requirements

3.1. Feature 1: User Registration

Description: Allows a new user to create an account by providing personal and login details.
Functional Requirements:
- The system shall allow users to enter first name, last name, email, and password
- The system shall store user information securely in the database
- The system shall prevent duplicate email registration

3.2. Feature 2: User Login and Logout

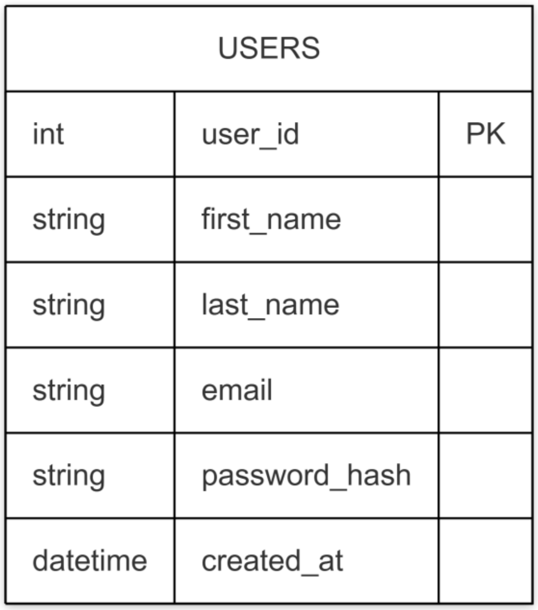
Description: Allows registered users to log in, access protected pages, and log out.
Functional Requirements:
- The system shall authenticate users using email and password
- The system shall allow logged-in users to view their profile/dashboard
- The system shall prevent access to protected pages when the user is logged out
- The system shall allow users to log out and terminate their session

4. Non-Functional Requirements

Security: Passwords shall be stored in encrypted/hashed form
Usability: The system shall be easy to use and understand
Performance: Login and registration responses shall occur within reasonable time
Reliability: The system shall handle invalid inputs gracefully

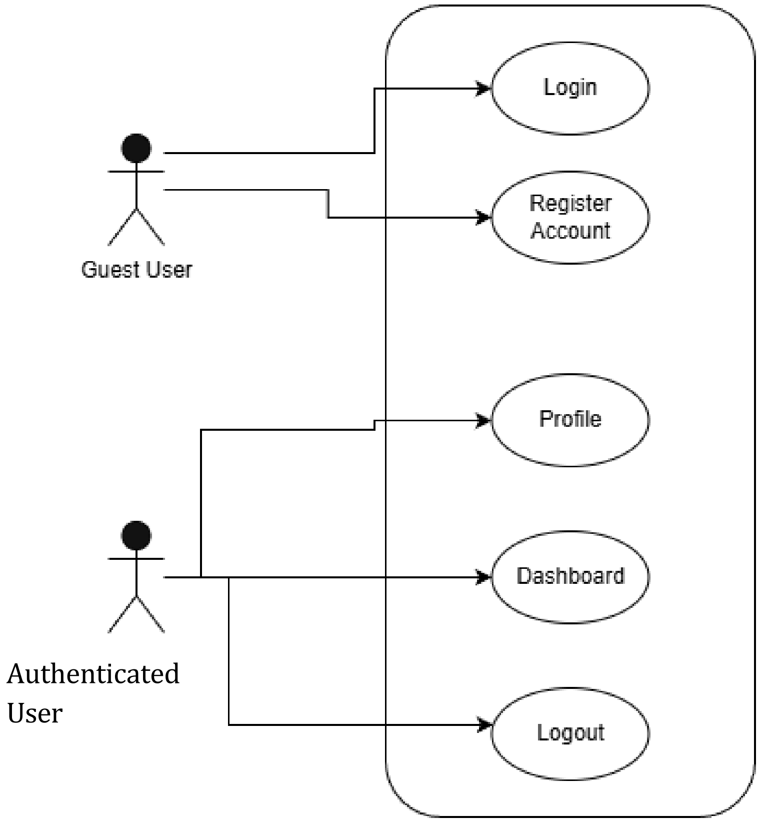
5. System Models (Diagrams)

5.1. ERD

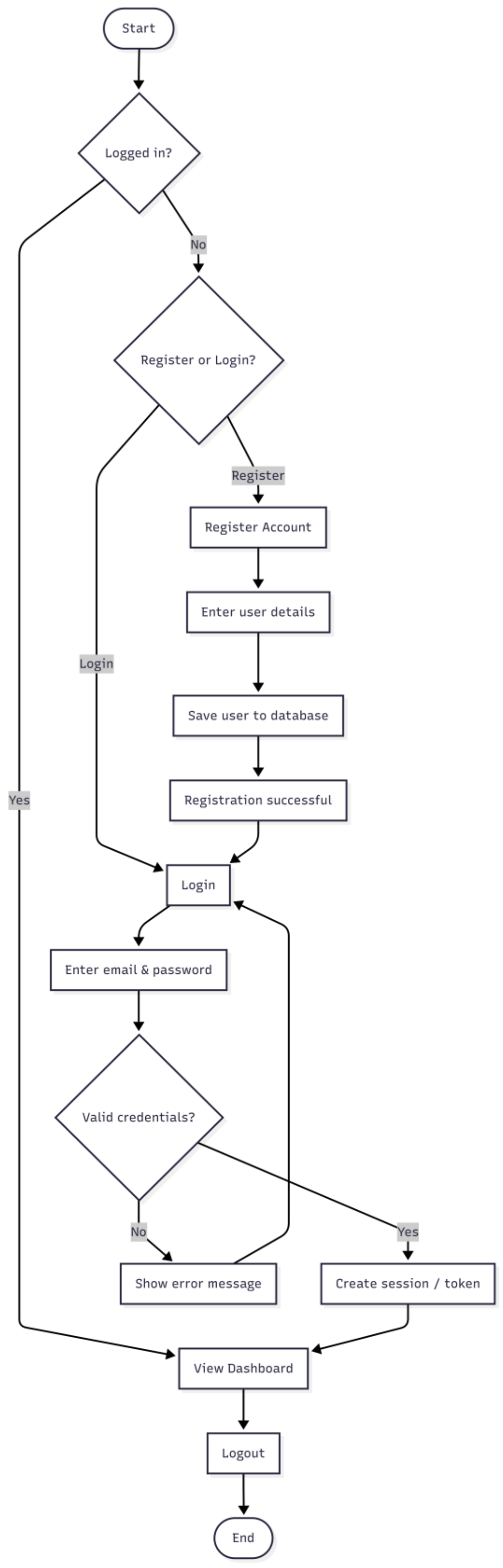


USERS		
int	user_id	PK
string	first_name	
string	last_name	
string	email	
string	password_hash	
datetime	created_at	

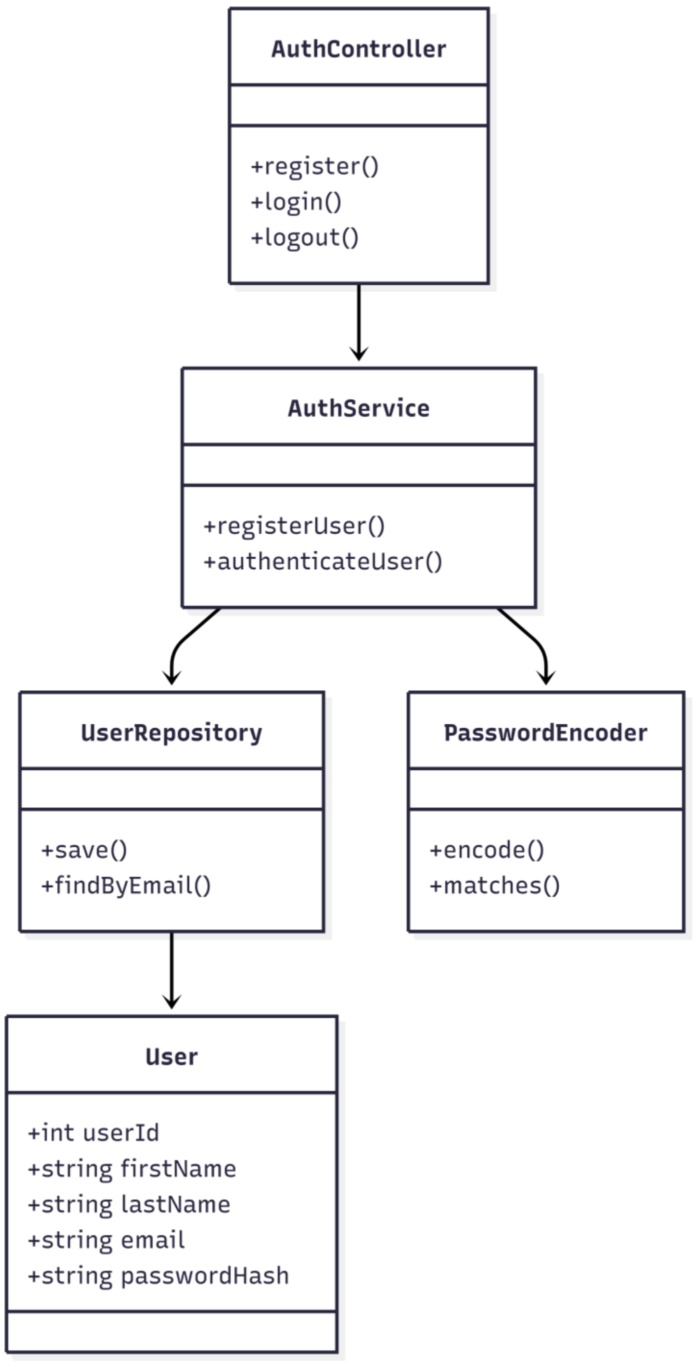
5.2. Use Case Diagram



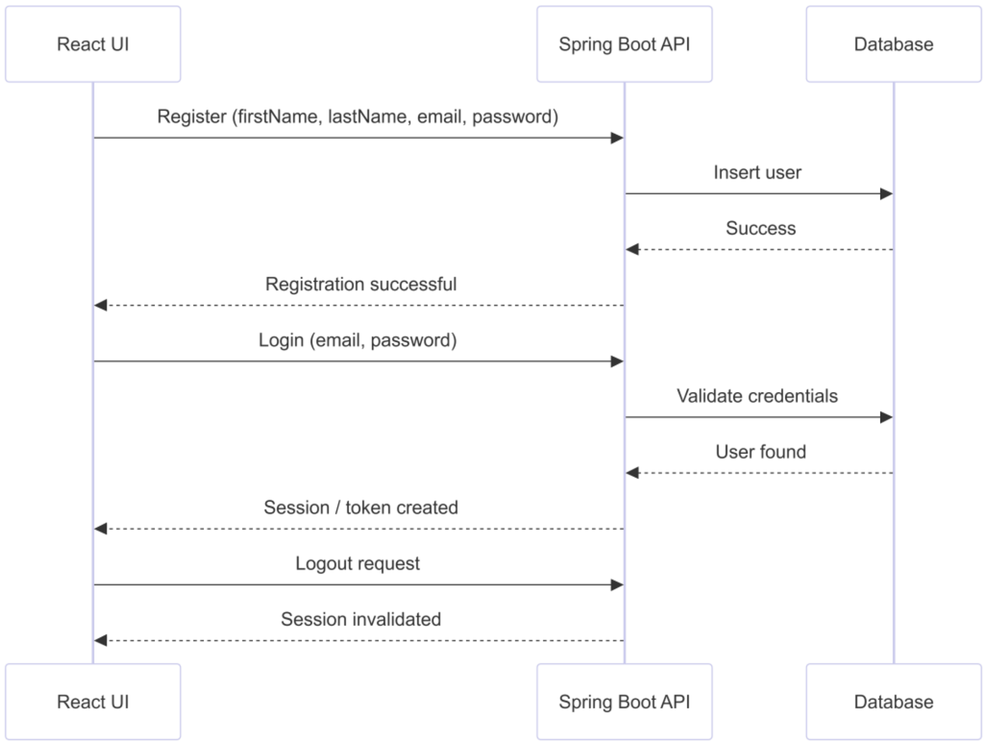
5.3. Activity Diagram



5.4. Class Diagram



5.5. Sequence Diagram

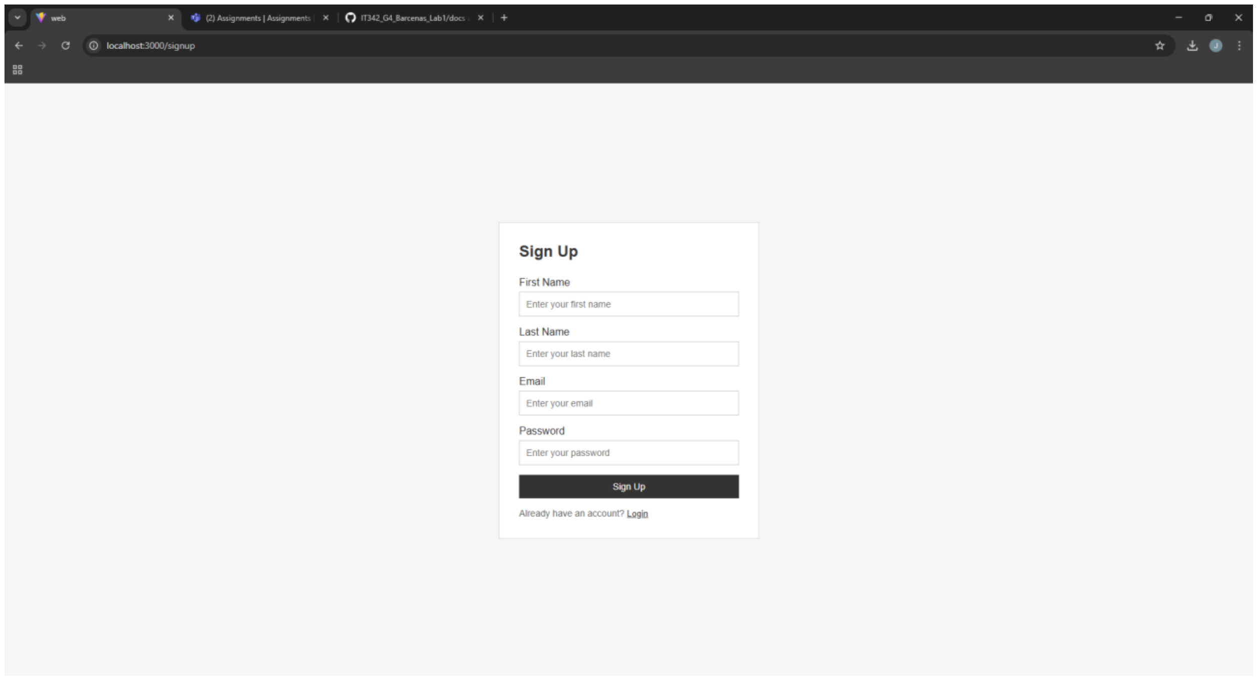


6. Appendices

All system diagrams were created using draw.io / diagrams.net and Mermaid syntax.

Web Screenshots

- **Register**



Sign Up

First Name

Enter your first name

Last Name

Enter your last name

Email

Enter your email

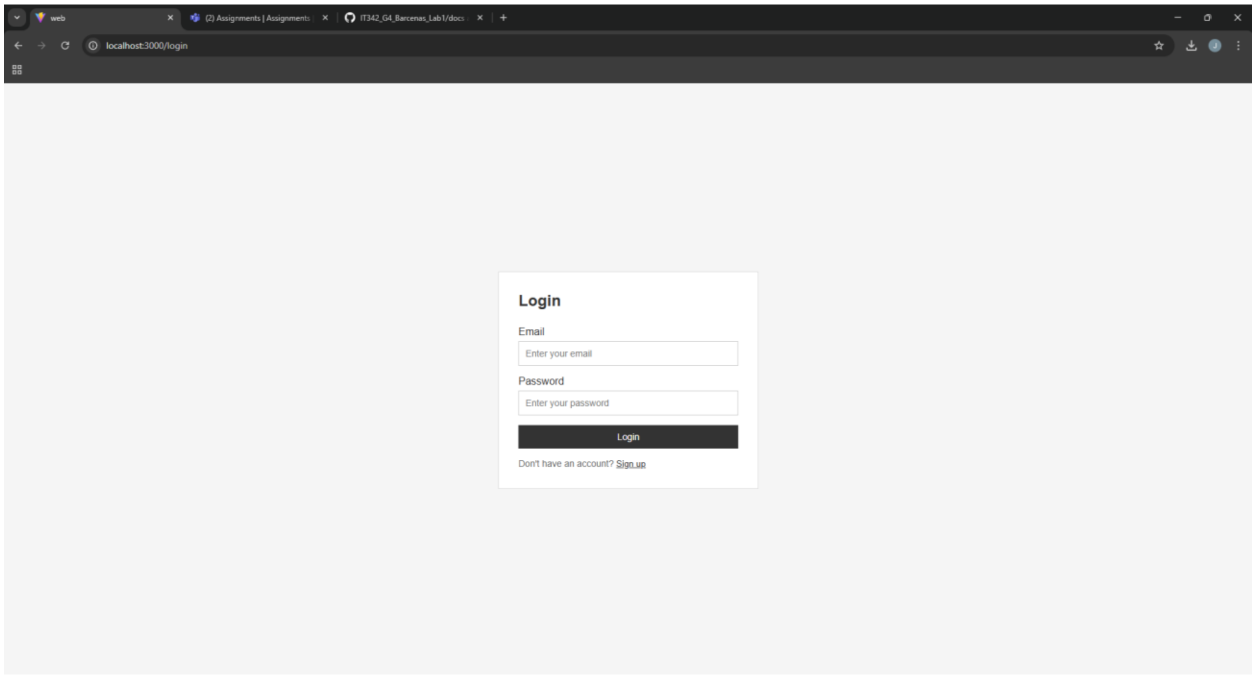
Password

Enter your password

Sign Up

Already have an account? Login

-Login



localhost:3000/login

Login

Email

Enter your email

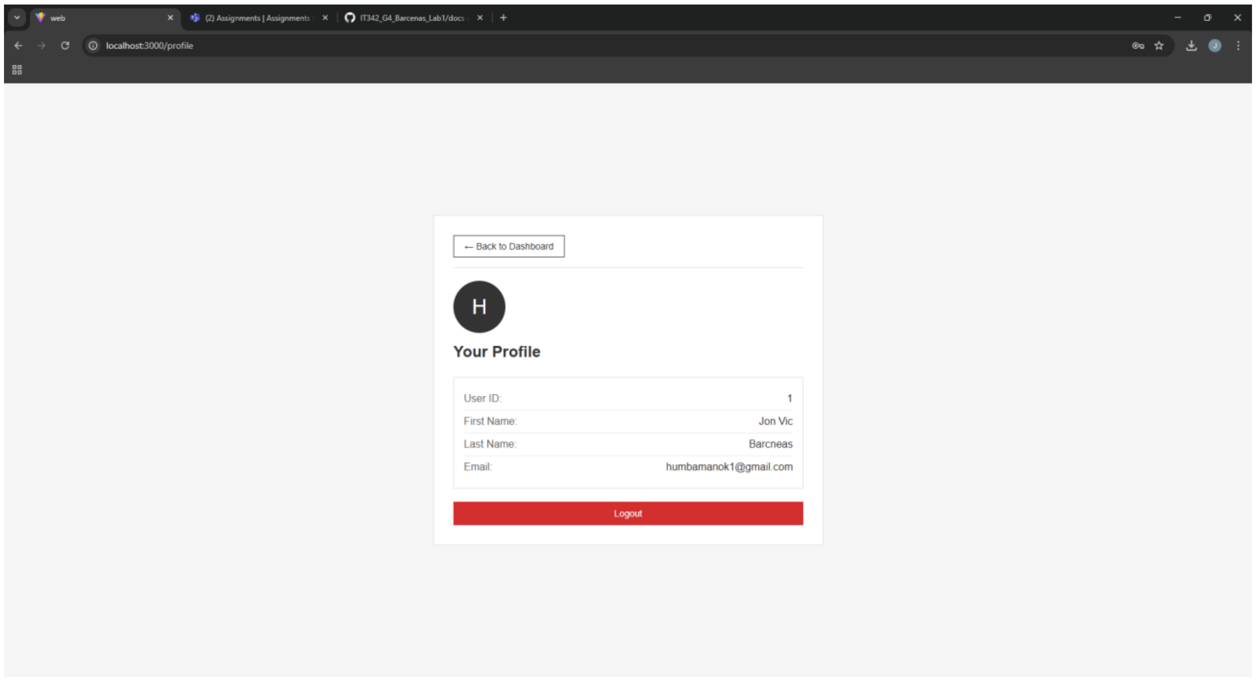
Password

Enter your password

Login

Don't have an account? Sign up

-Dashboard/Profile



localhost:3000/profile

← Back to Dashboard

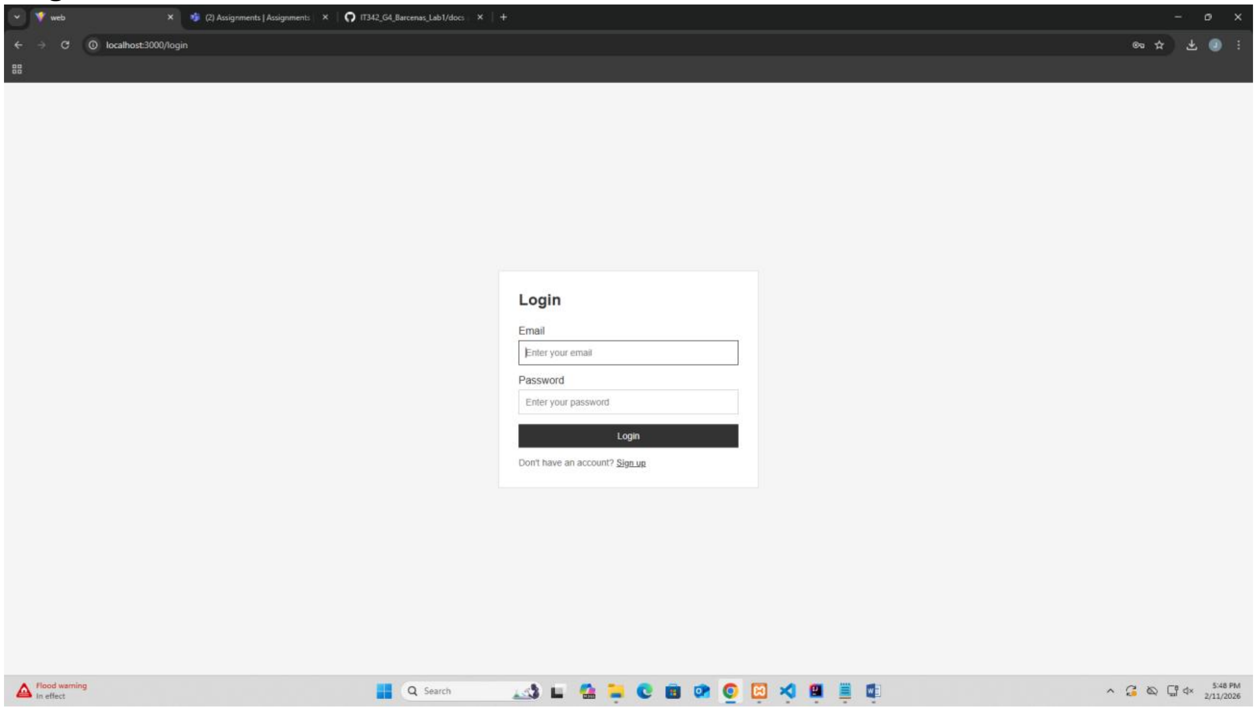
H

Your Profile

User ID:	1
First Name:	Jon Vic
Last Name:	Barcneas
Email:	humbamanok1@gmail.com

Logout

-Logout result



localhost:3000/login

Login

Email

Enter your email

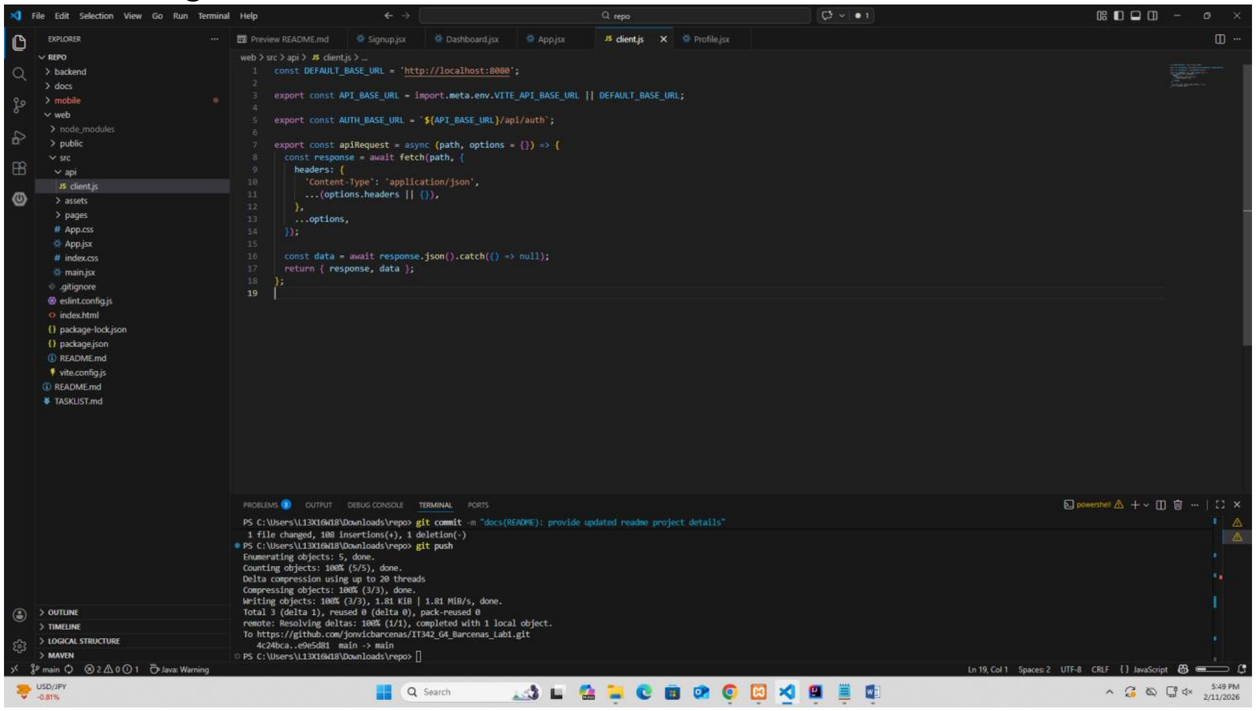
Password

Enter your password

Login

Don't have an account? Sign up

Proof of Integration



web > src > api > client.js > ...

```javascript
const DEFAULT_BASE_URL = 'http://localhost:8080';

export const API_BASE_URL = import.meta.env.VITE_API_BASE_URL || DEFAULT_BASE_URL;

export const AUTH_BASE_URL = `${API_BASE_URL}/api/auth`;

export const apiRequest = async (path, options = {}) => {
  const response = await fetch(path, {
    headers: {
      'Content-Type': 'application/json',
      ...(options.headers || {}),
    },
    ...options,
  });

  const data = await response.json().catch(() => null);
  return { response, data };
};

```

```
PS C:\Users\L13X16W18\Downloads\repo> git commit -m "docs(README): provide updated readme project details"
 1 file changed, 108 insertions(+), 1 deletion(-)
PS C:\Users\L13X16W18\Downloads\repo> git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 20 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 1.81 KiB | 1.81 MiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/jonvicbarcenas/IT342_G4_Barcenas_Lab1.git
   4c24bca..e9e5d81  main -> main
PS C:\Users\L13X16W18\Downloads\repo> 
```