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: Niña Nicole S. Villadarez

Date of Submission: 02/11/2026

Version: 2

Table of Contents

1. Introduction

1.1. Purpose

This document defines the functional and non-functional requirements of a mini authentication application for user registration and login.

It is intended for:

- Students
- Developers
- Testers
- Project stakeholders
- Instructors

1.2. Scope

In scope:

- User registration with basic information
- User login with credentials
- Token-based authentication
- Access to a protected user profile
- Secure logout
- Password hashing
- Username and email uniqueness validation

Out of scope:

- Password reset / forgot password
- Email verification
- Social login
- Roles and permissions
- Multi-factor authentication
- Profile editing and account deletion

1.3. Definitions, Acronyms, and Abbreviations

Term	Definition
JWT	A token used for stateless authentication
API	Application Programming Interface
BCrypt	Secure password hashing algorithm
REST	Web service architecture using HTTP methods

CORS	Security mechanism for cross-origin requests
SPA	Single Page Application
JPA	Java Persistence API for database access

2. Overall Description

2.1. System Perspective

The system provides a basic authentication layer for a web application.

It uses a three-tier structure:

- Presentation Layer – User interface for login and registration
- Business Logic Layer – Handles authentication rules and token validation
- Data Layer – Stores user credentials and profile data

The system communicates using RESTful APIs and maintains authentication through JWT tokens.

2.2. User Classes and Characteristics

Guest Users

- Can register and login
- No access to protected pages

Authenticated Users

- Can access profile/dashboard
- Can logout
- Have a valid JWT token

2.3. Operating Environment

- Runs on modern web browsers
- Requires internet connection
- Uses a backend server and relational database
- Tokens are stored in browser local storage

2.4. Assumptions and Dependencies

Assumptions

- Users have valid emails
- Internet access is available
- Browsers support localStorage
- Database and server are reachable

Dependencies

- Authentication framework
- JWT library
- Password hashing library
- Database driver
- HTTP client

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 new users to create an account by providing their basic information. The system validates the input, securely stores the credentials, and automatically logs the user in after successful registration.
Functional Requirements:
- Accept username, email, password, first name, last name

- Validate unique username and email

- Validate email format

- Enforce password length (min. 8)

- Hash password before saving

- Store user in database

- Generate JWT token after success

- Return: 201 Created on success, 409 Conflict if username/email exists, 400 Bad Request for validation errors

3.2. Feature 2: User Login

Description: Authenticates existing users by verifying their credentials and granting them access to protected areas of the system.
Functional Requirements:
- Accept username and password

- Validate user existence

- Verify password

- Generate JWT token

- Return: 200 OK on success, 401 Unauthorized on failure

- Client stores token and redirects to dashboard

3.3 Feature 3: Profile Access (Protected)

Description: Allows logged-in users to view their profile information. Access is restricted to users with a valid authentication token.

Functional Requirements:

- Require Authorization: Bearer <token>

- Validate token signature and expiration

- Extract username from token

- Fetch user profile

- Exclude password from response

- Return 401 Unauthorized if invalid

3.4 Feature 4: User Logout

Description: Ends the user's session by invalidating the current token, preventing further access to protected resources.

Functional Requirements:

- Require valid token

- Invalidate token (blacklist)

- Return success message

- Client deletes token and redirects to login

4. Non-Functional Requirements

Performance

- API responses < 500 ms
- Token validation < 50 ms
- Security

BCrypt password hashing

- Encrypted JWT tokens
- HTTPS enforced
- Input validation
- SQL injection protection
- Token blacklist

Usability

- Clear error messages
- Responsive UI
- Real-time form validation

Reliability

- Graceful error handling
- Authentication error rate < 1%

Maintainability

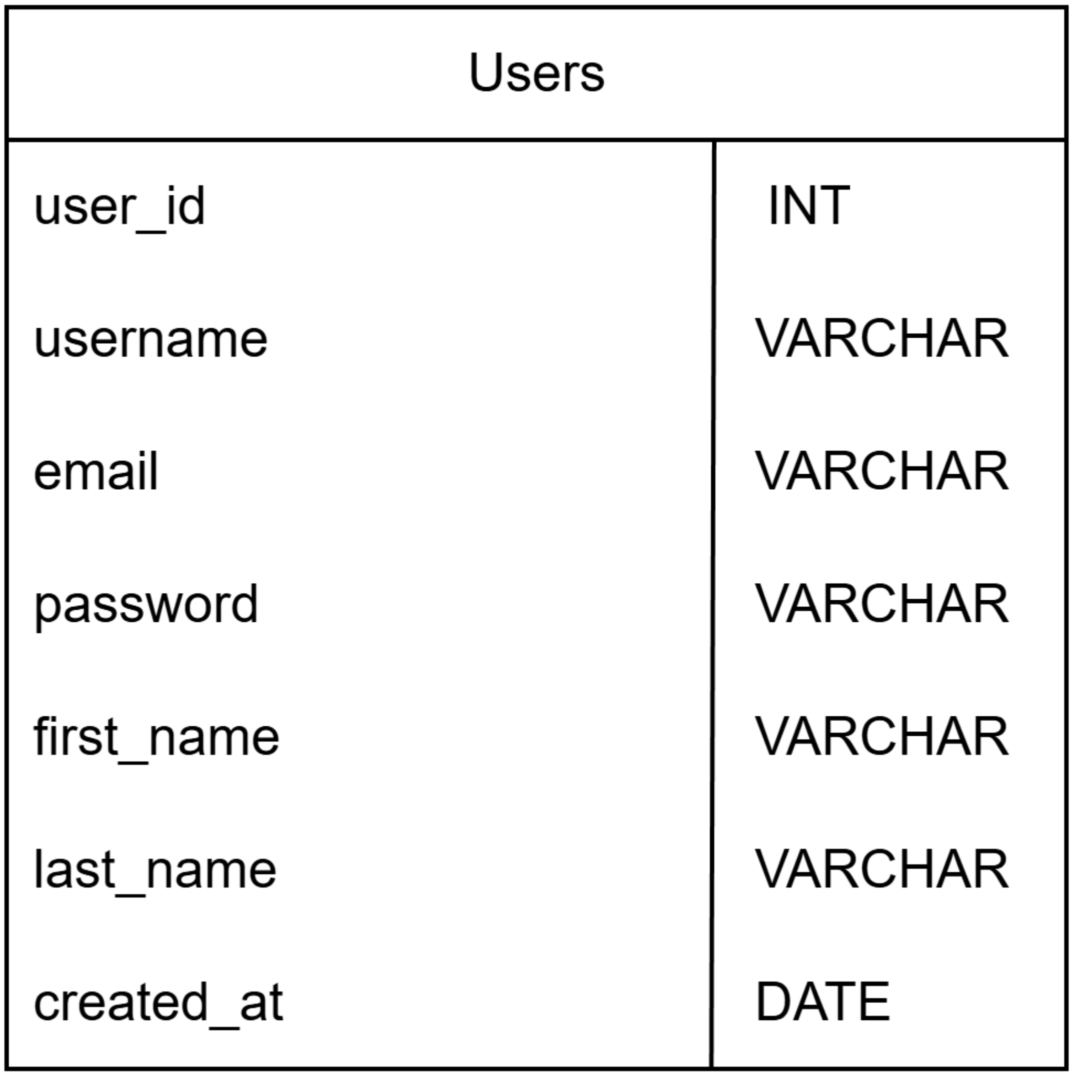
- Layered architecture
- Centralized business logic
- Externalized configuration

Compatibility

- Works on modern browsers
- Supports relational databases
- REST-compliant API

5. System Models (Diagrams)

5.1. ERD



Users	
user_id	INT
username	VARCHAR
email	VARCHAR
password	VARCHAR
first_name	VARCHAR
last_name	VARCHAR
created_at	DATE

Figure 1.1 ERD

5.2. Use Case Diagram

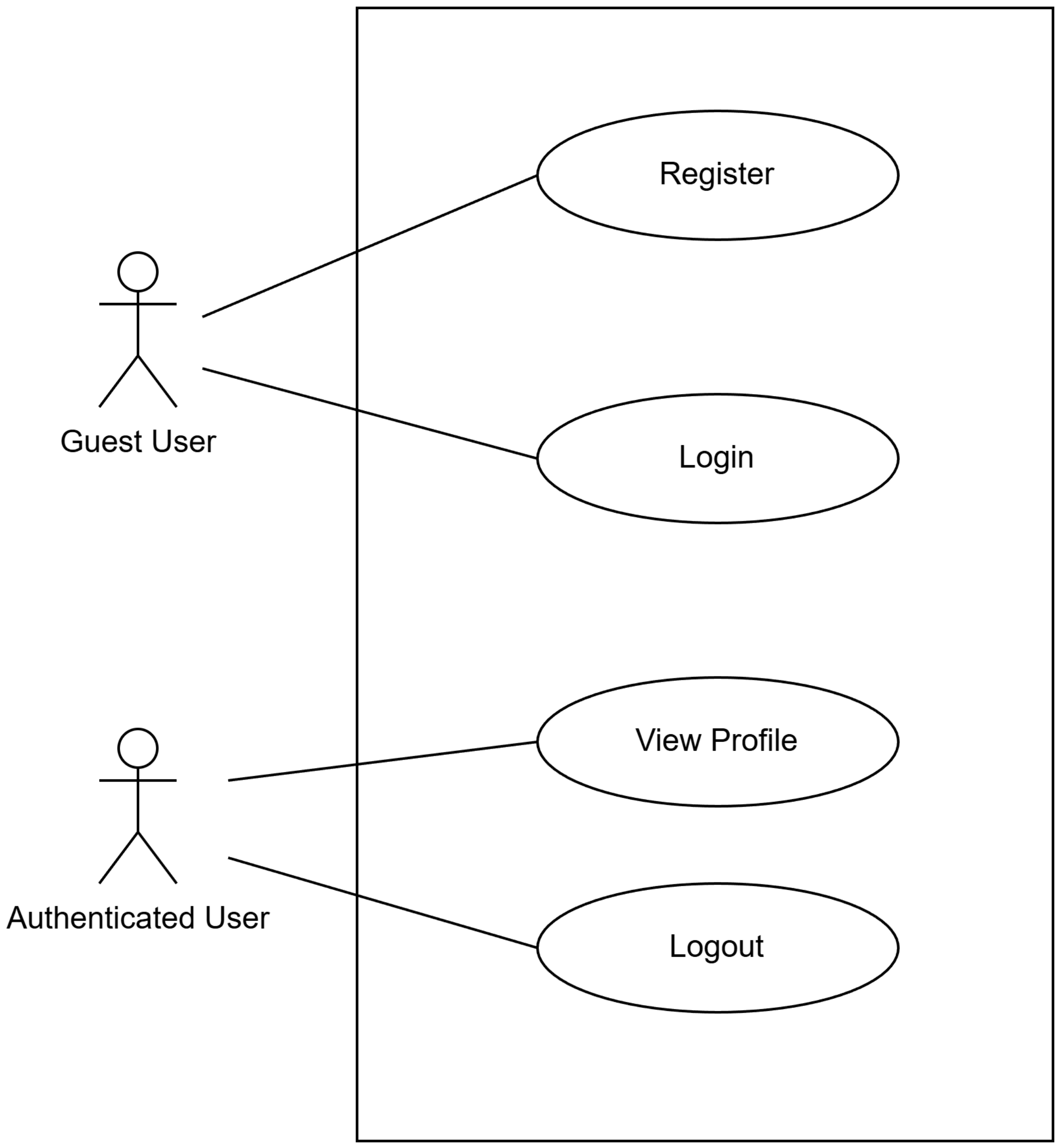


Figure 1.2 Use Case Diagram

5.3. Activity Diagram

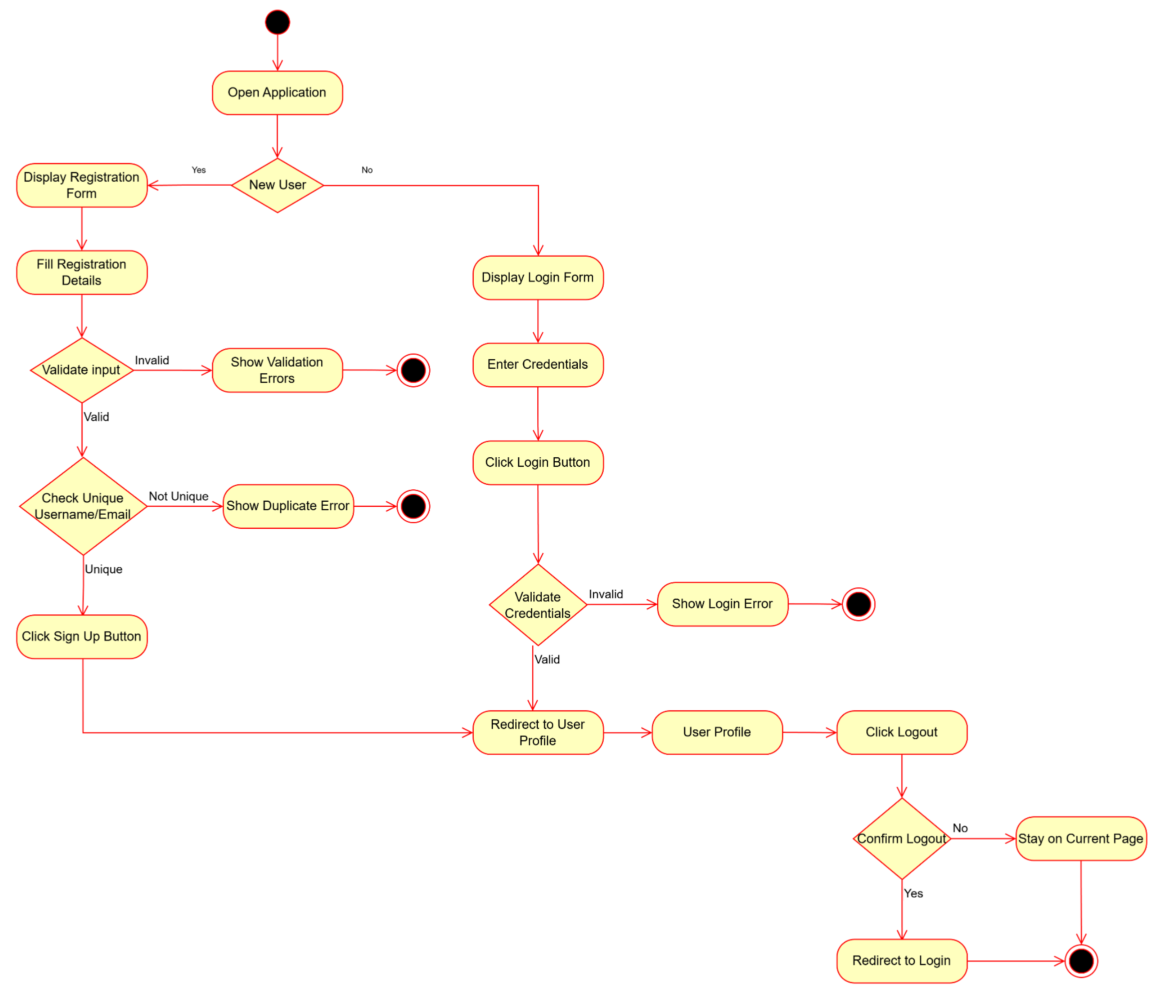


Figure 1.3 Activity Diagram

5.4. Class Diagram

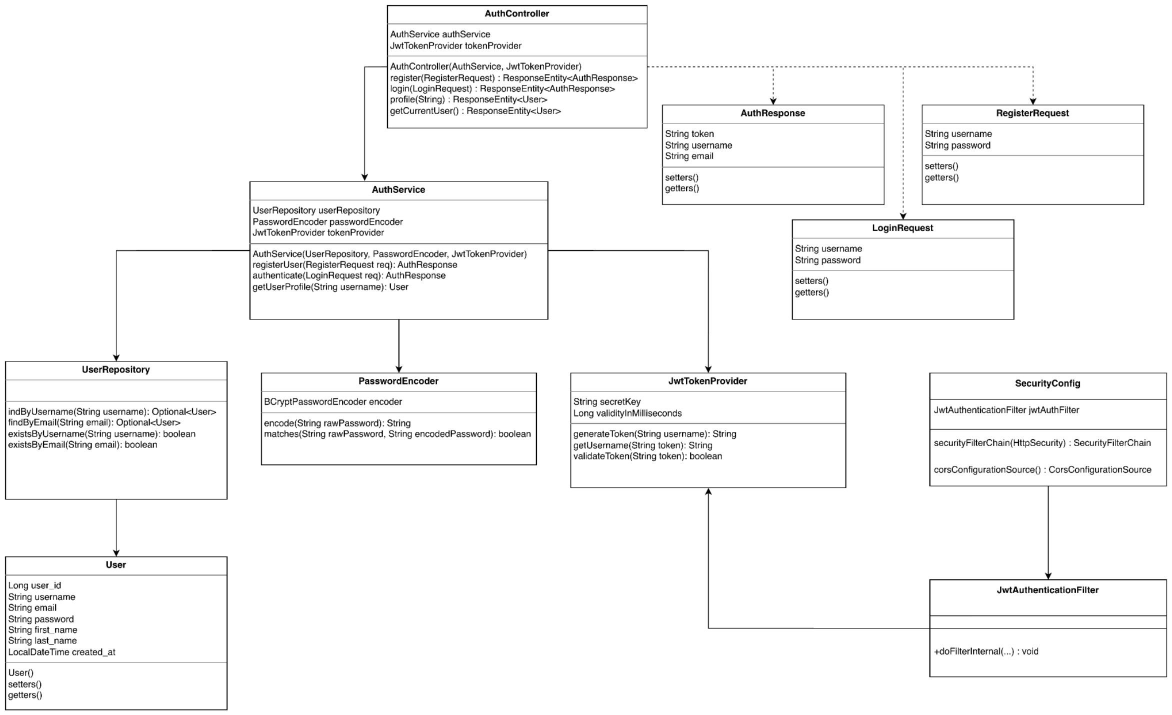

AuthController
AuthService authService
JwtTokenProvider tokenProvider
AuthController(AuthService, JwtTokenProvider)
register(RegisterRequest) : ResponseEntity<AuthResponse>
login(LoginRequest) : ResponseEntity<AuthResponse>
profile(String) : ResponseEntity<User>
getCurrentUser() : ResponseEntity<User>
AuthResponse
String token
String username
String email
setters()
getters()
RegisterRequest
String username
String password
setters()
getters()
LoginRequest
String username
String password
setters()
getters()
AuthService
UserRepository userRepository
PasswordEncoder passwordEncoder
JwtTokenProvider tokenProvider
AuthService(UserRepository, PasswordEncoder, JwtTokenProvider)
registerUser(RegisterRequest req): AuthResponse
authenticate(LoginRequest req): AuthResponse
getUserProfile(String username): User
UserRepository
indByUsername(String username): Optional<User>
findByEmail(String email): Optional<User>
existsByUsername(String username): boolean
existsByEmail(String email): boolean
PasswordEncoder
BCryptPasswordEncoder encoder
encode(String rawPassword): String
matches(String rawPassword, String encodedPassword): boolean
JwtTokenProvider
String secretKey
Long validityInMilliseconds
generateToken(String username): String
getUsername(String token): String
validateToken(String token): boolean
SecurityConfig
JwtAuthenticationFilter jwtAuthFilter
securityFilterChain(HttpSecurity) : SecurityFilterChain
corsConfigurationSource() : CorsConfigurationSource
User
Long user_id
String username
String email
String password
String first_name
String last_name
LocalDateTime created_at
User()
setters()
getters()
JwtAuthenticationFilter
+doFilterInternal(...) : void

Figure 1.4 Class Diagram

5.5. Sequence Diagram

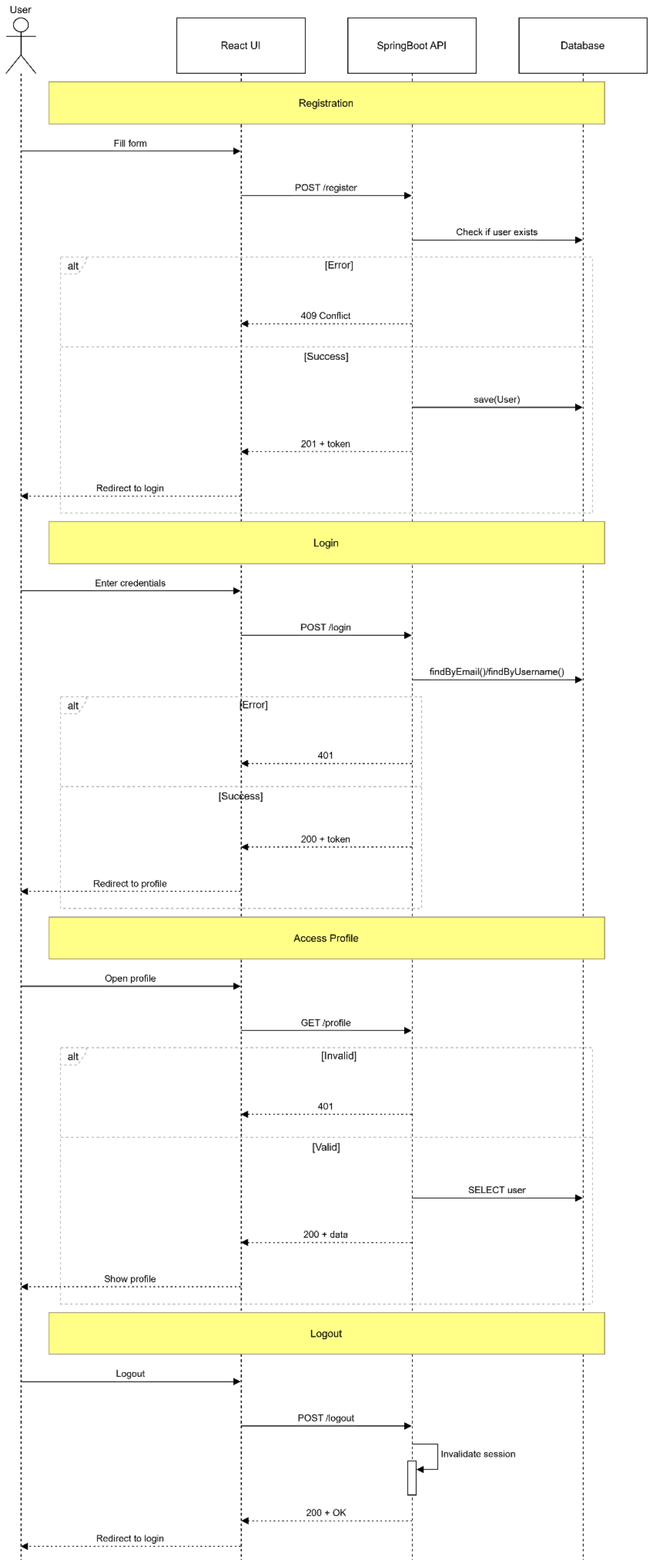
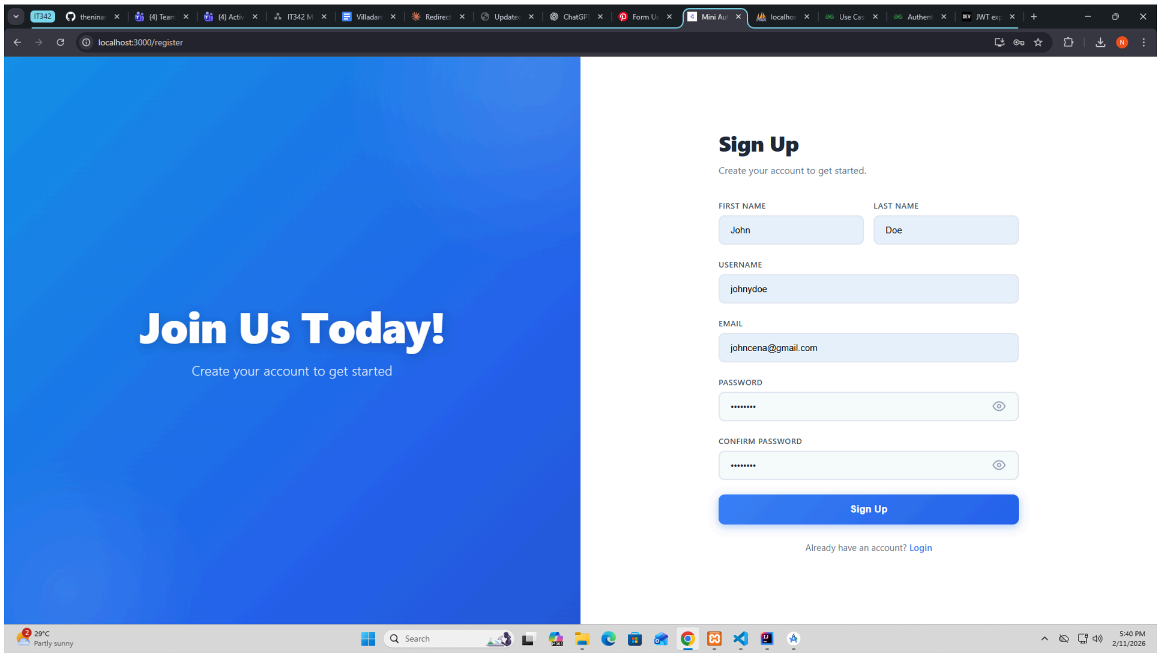


Figure 1.5 Sequence Diagram

6. Appendices

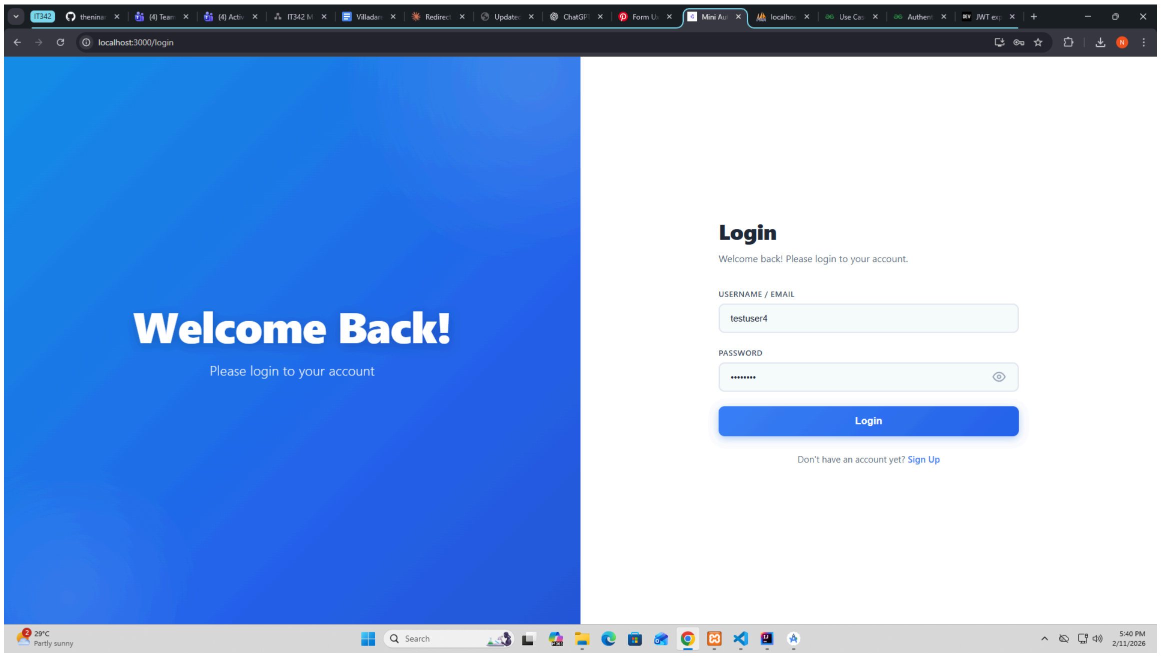
Appendix A: System Models

Appendix B: Screenshots (Web)


localhost:3000/register
Join Us Today!
Create your account to get started
Sign Up
Create your account to get started.
Sign Up
Already have an account? Login

FIRST NAME: John
LAST NAME: Doe
USERNAME: johnydoe
EMAIL: johncena@gmail.com
PASSWORD: ••••••••
CONFIRM PASSWORD: ••••••••

Registration Page


localhost:3000/login

Welcome Back!

Please login to your account

Login

Welcome back! Please login to your account.

USERNAME / EMAIL

testuser4

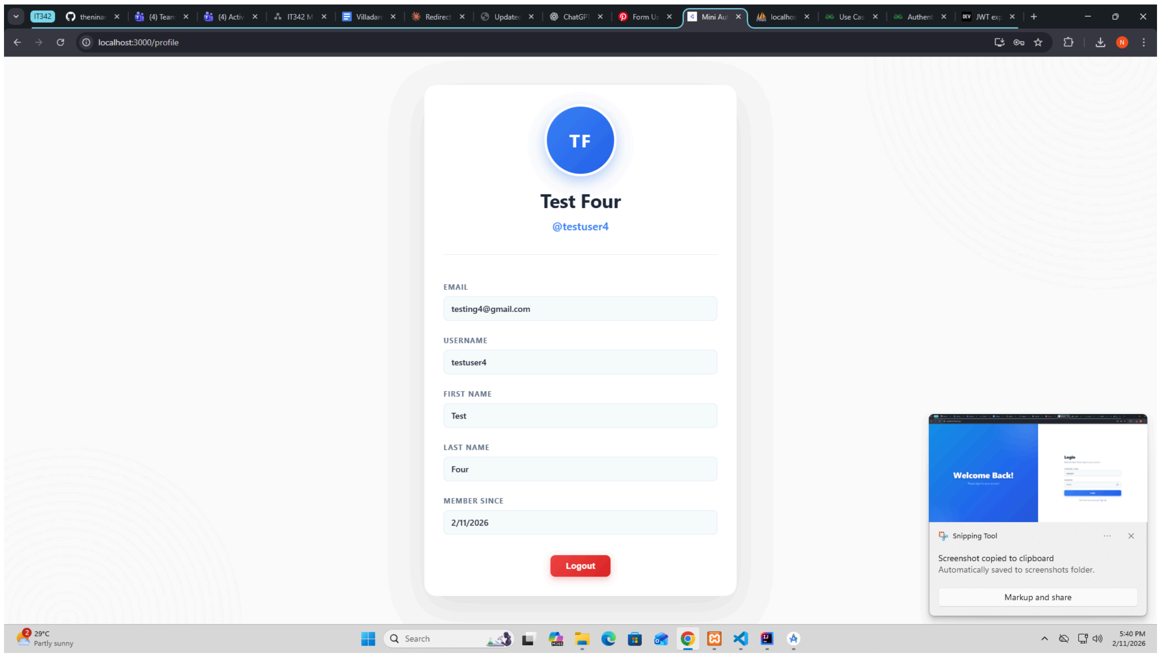
PASSWORD

••••••••

Login

Don't have an account yet? Sign Up

Login Page


localhost:3000/profile

TF

Test Four

@testuser4

EMAIL

testing4@gmail.com

USERNAME

testuser4

FIRST NAME

Test

LAST NAME

Four

MEMBER SINCE

2/11/2026

Logout

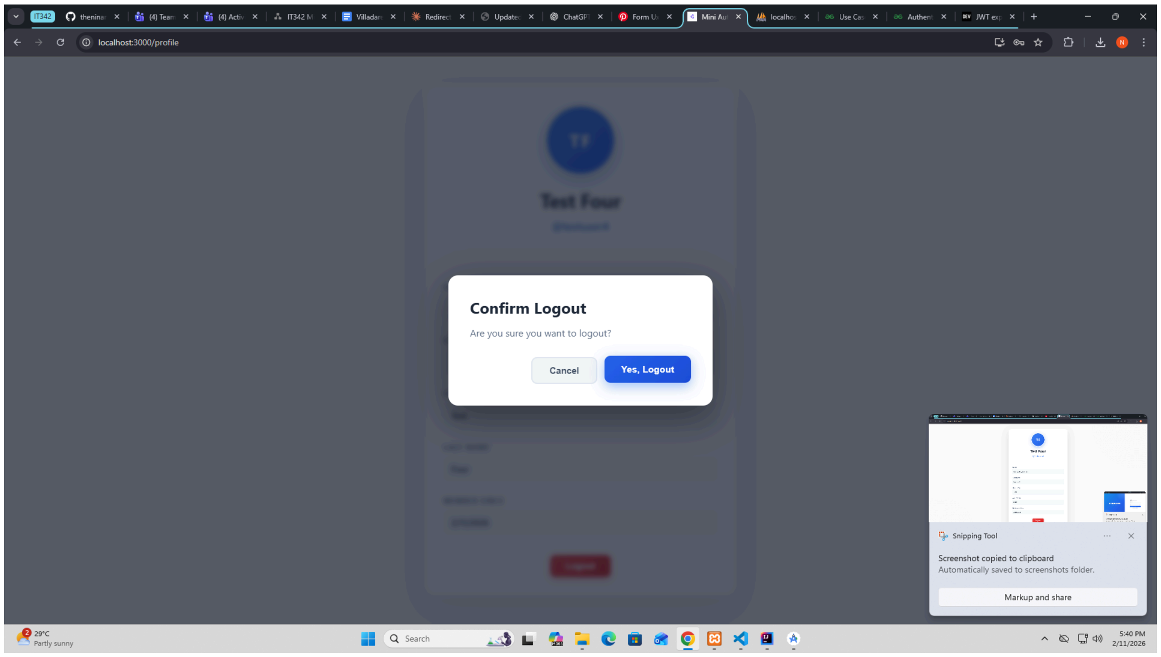
Snipping Tool

Screenshot copied to clipboard

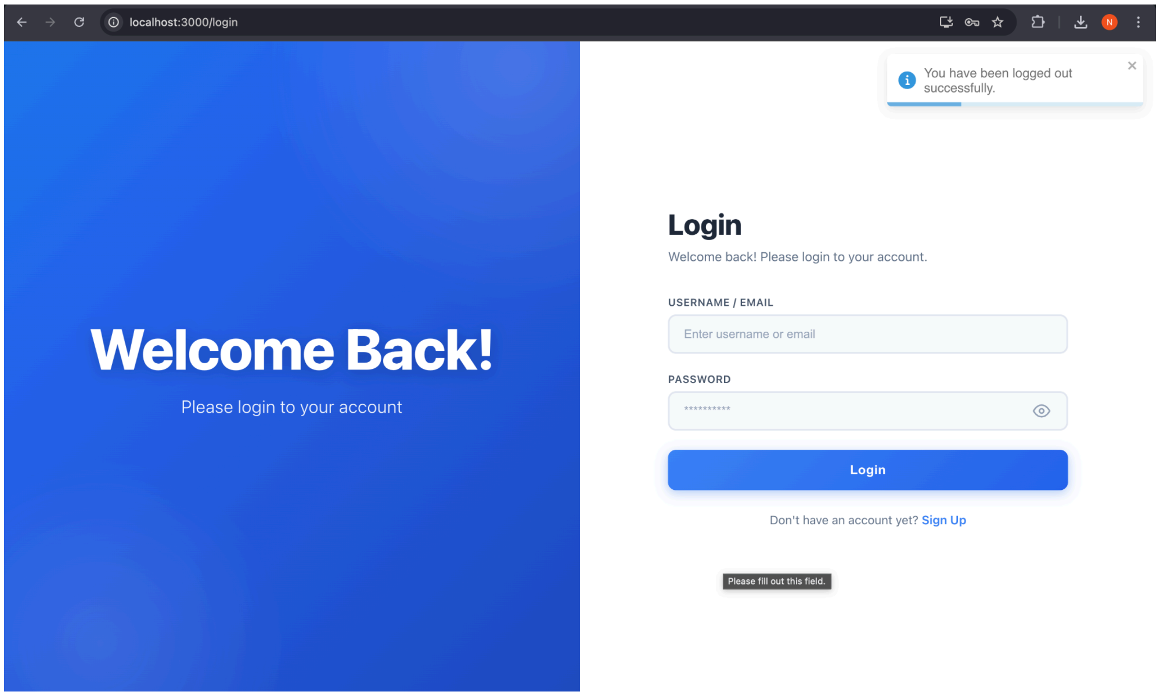
Automatically saved to screenshots folder.

Markup and share

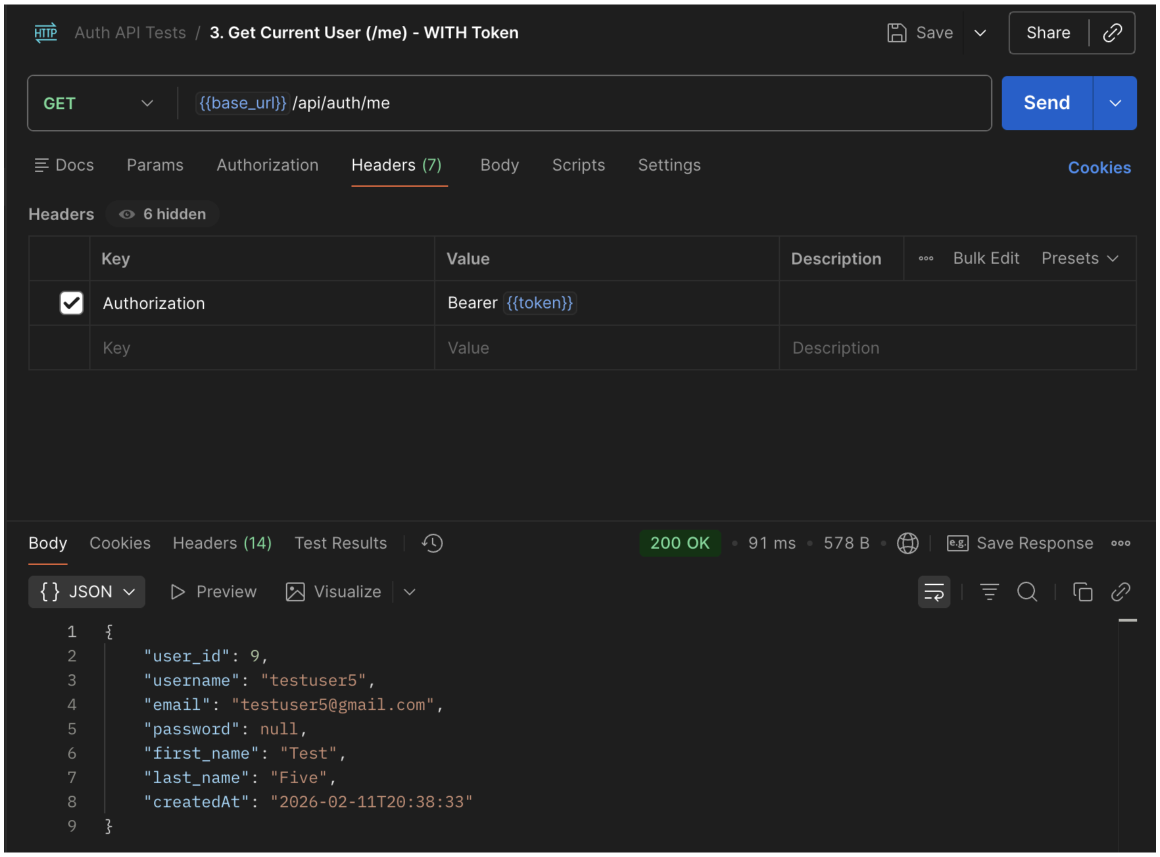
Profile Page


localhost:3000/profile
Confirm Logout
Are you sure you want to logout?
Cancel
Yes, Logout
Snipping Tool
Screenshot copied to clipboard
Automatically saved to screenshots folder.
Markup and share

Logout Modal


localhost:3000/login
You have been logged out successfully.
Welcome Back!
Please login to your account
Login
Welcome back! Please login to your account.
USERNAME / EMAIL
Enter username or email
PASSWORD
Login
Don't have an account yet? Sign Up
Please fill out this field.

Logout Result (After clicking "Yes, Logout" button)



GET {{base_url}} /api/auth/me

Headers (7)

Headers 6 hidden

Key	Value	Description
✓ Authorization	Bearer {{token}}	
Key	Value	Description

Body Cookies Headers (14) Test Results 200 OK • 91 ms • 578 B

JSON

```
{
    "user_id": 9,
    "username": "testuser5",
    "email": "testuser5@gmail.com",
    "password": null,
    "first_name": "Test",
    "last_name": "Five",
    "createdAt": "2026-02-11T20:38:33"
}
```

Proof of integration via Postman