CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342-Section SYSTEMS INTEGRATION AND ARCHITECTURE 1

FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

Project Title: User Registration and Authentication System

Prepared By: Frances Anne B. Riconalla

Date of Submission: February 2, 2026

Version: 1

Table of Contents

1. Introduction 3
   1.1. Purpose 3
   1.2. Scope 3
   1.3. Definitions, Acronyms, and Abbreviations 3
2. Overall Description 3
   2.1. System Perspective 3
   2.2. User Classes and Characteristics 3
   2.3. Operating Environment 3
   2.4. Assumptions and Dependencies 3
3. System Features and Functional Requirements 3
   3.1. Feature 1: 3
   3.2. Feature 2: 3
4. Non-Functional Requirements 3
5. System Models (Diagrams) 4
   5.1. ERD 4
   5.2. Use Case Diagram 4
   5.3. Activity Diagram 4
   5.4. Class Diagram 4
   5.5. Sequence Diagram 4
6. Appendices 4

1. Introduction

1.1. Purpose

The purpose of this document is to describe the functional requirements of the User Registration and Authentication System. It serves as a reference for developers, instructors, and stakeholders to understand the system's functionality, behavior, and design prior to implementation.

1.2. Scope

The system provides basic user authentication functionality, including:

- User registration
- User login
- Access to protected pages (profile/dashboard)
- User logout

The system does not include advanced features such as password recovery, email verification, or role-based administration. It focuses solely on authentication and access control.

1.3. Definitions, Acronyms, and Abbreviations

- Authentication – The process of verifying a user's identity.
- Guest User – A user who is not logged in.
- Authenticated User – A user who has successfully logged in.
- Token – A temporary credential used to maintain authenticated sessions.
- API – Application Programming Interface.

2. Overall Description

2.1. System Perspective

The User Registration and Authentication System operates as a backend service that supports a web-based frontend. It follows a client-server architecture where the frontend communicates with the backend through RESTful APIs, and the backend interacts with a relational database.

2.2. User Classes and Characteristics

- Guest User
    - Can register and log in
    - Cannot access protected pages
- Authenticated User
    - Has valid login credentials
    - Can access profile/dashboard
    - Can log out of the system

2.3. Operating Environment

- Frontend: Web browser (React-based UI)

- Backend: Spring Boot (Java)
- Database: MySQL
- Tools: IntelliJ IDEA, VS Code, Git, Postman

2.4. Assumptions and Dependencies

- Users have internet access.
- The database is available and reachable by the backend.
- Authentication tokens are securely generated and invalidated.
- The system assumes correct configuration of the backend environment.

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 users to submit registration details.
- The system shall validate registration data.
- The system shall store user information securely in the database.
- The system shall confirm successful registration.

3.2. Feature 2: User Login and Authentication

Description: Allows registered users to log in and access protected resources.
Functional Requirements:

- The system shall authenticate users using valid credentials.
- The system shall generate an authentication token upon successful login.
- The system shall allow authenticated users to access protected pages.
- The system shall deny access to protected pages for unauthenticated users.

3.3. Feature 3: User Logout

Description: Allows authenticated users to terminate their session.
Functional Requirements:

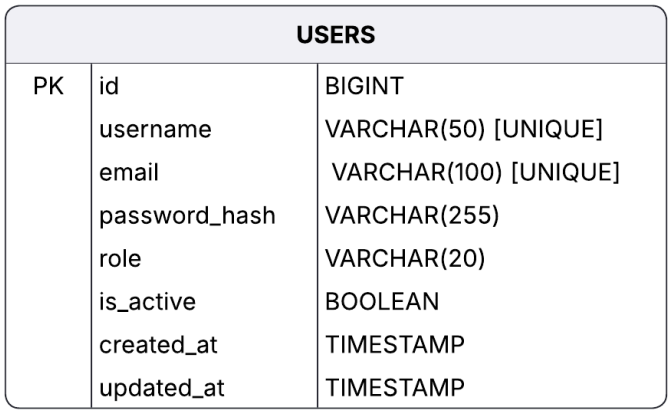
- The system shall allow authenticated users to log out.
- The system shall invalidate the user's authentication token.
- The system shall prevent further access to protected pages after logout

4. Non-Functional Requirements

- **Security:** Passwords shall be stored in hashed form.
- **Performance:** Authentication requests should be processed within acceptable response time.
- **Usability:** The system should provide clear feedback for authentication actions.
- **Reliability:** The system should handle concurrent user requests without failure.
- **Maintainability:** The system design should support future enhancements.

5. System Models (Diagrams)

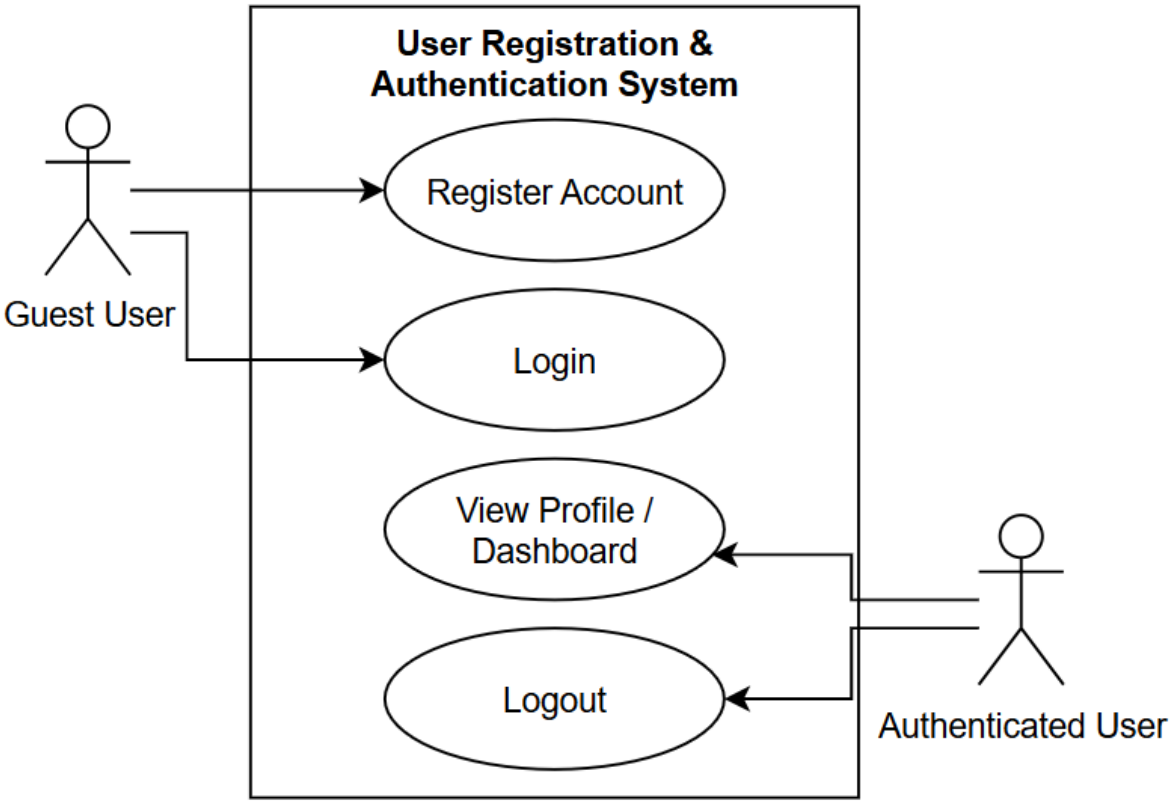
5.1. ERD



The Entity Relationship Diagram (ERD) illustrates the database structure of the User Registration and Authentication System. The system contains a single primary entity, USERS, which stores user credentials and account information. Each user is uniquely identified by a primary key (id), while username and email are defined as unique fields to maintain data integrity.

Passwords are stored securely using the password_hash field, which contains encrypted values generated through BCrypt. Additional attributes such as role, is_active, and timestamp fields support access control and account management. The ERD ensures a secure and normalized database design that supports user registration and authentication.

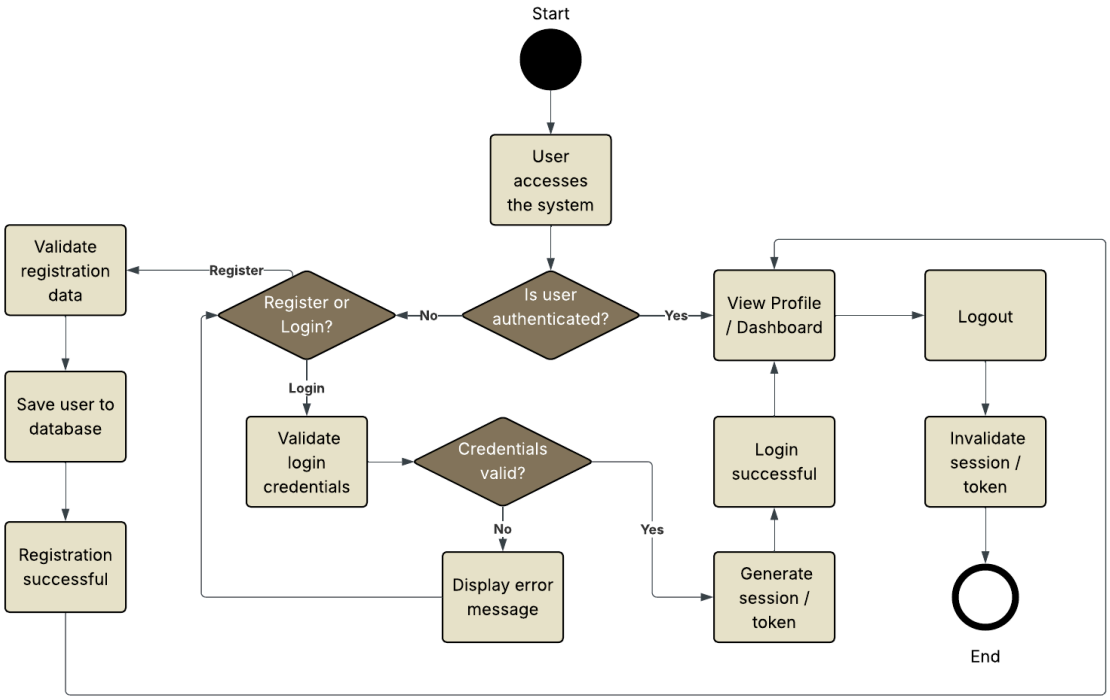
5.2. Use Case Diagram



The Use Case Diagram presents the functional interactions between system users and the application. It identifies two actors: the Guest User and the Authenticated User.

Guest Users can register and log in, while Authenticated Users can access protected features such as viewing the dashboard and logging out. The diagram clearly distinguishes between public and restricted functionalities, demonstrating the system's access control mechanism and security boundaries.

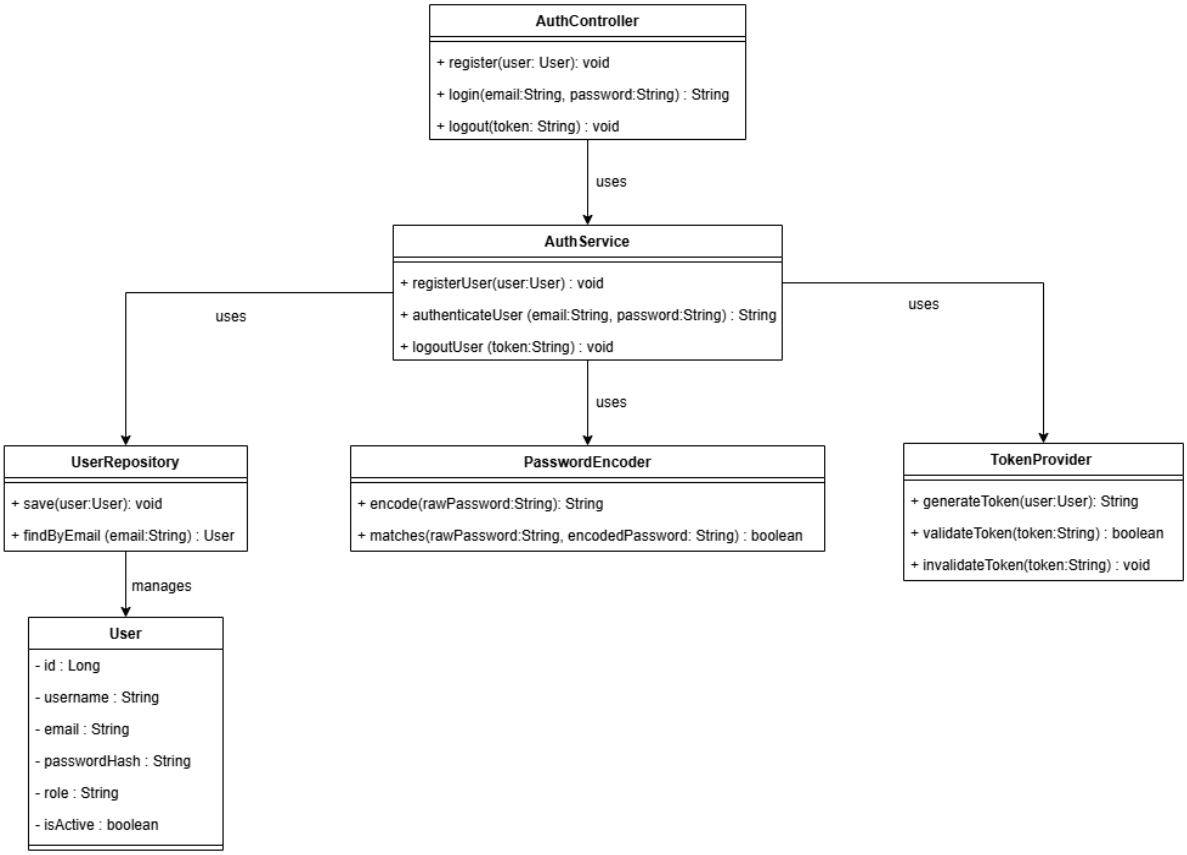
5.3. Activity Diagram



The Activity Diagram models the workflow of the registration, login, and logout processes. It begins with user access to the system, followed by a decision to register or log in. The system validates input data and checks credentials before granting access.

If authentication is successful, the user is redirected to the dashboard; otherwise, an error message is displayed. The logout process invalidates the session or token, ending authenticated access. This diagram highlights system validation, decision-making logic, and secure session management.

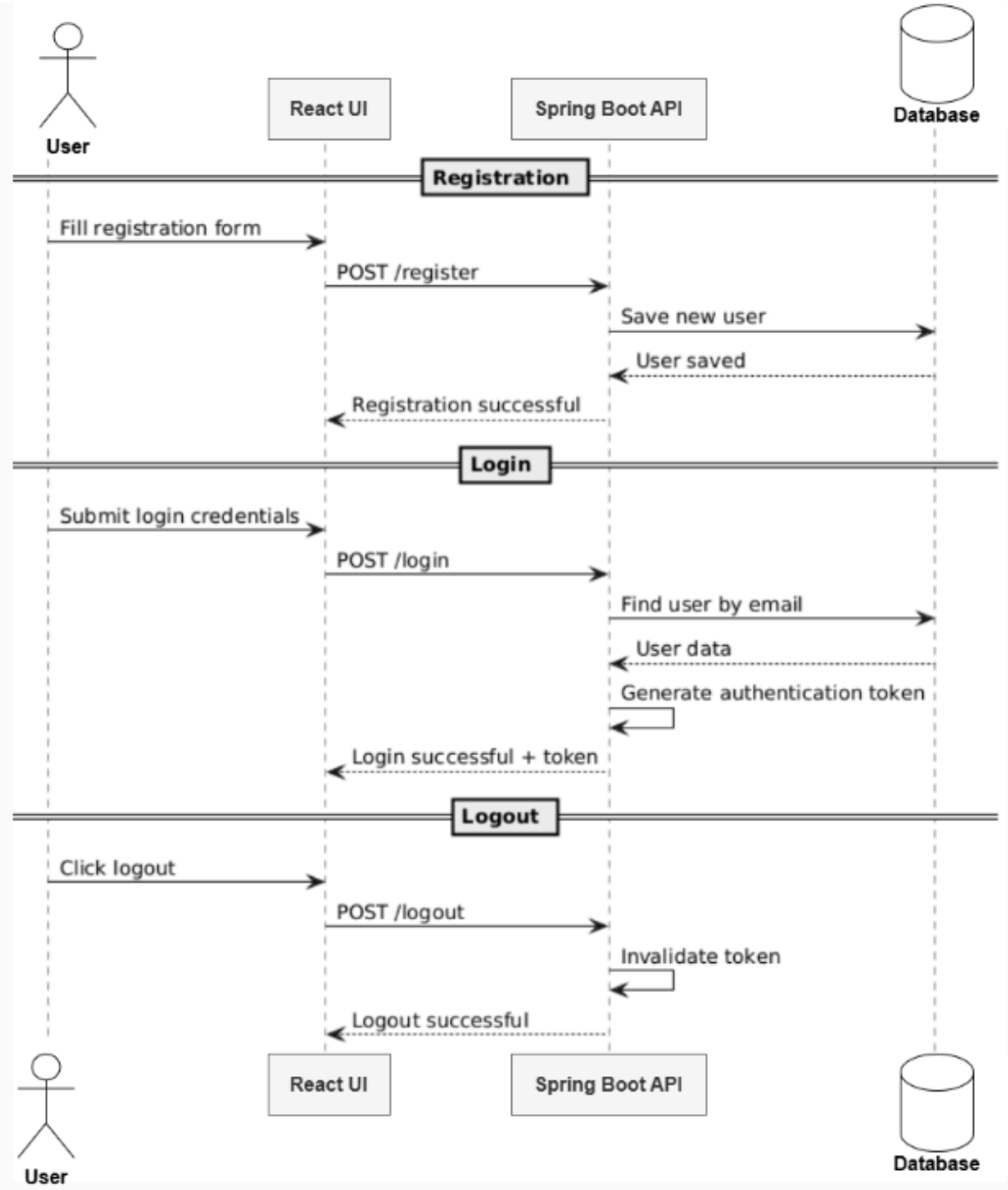
5.4. Class Diagram



The Class Diagram illustrates the structural design of the backend system. It follows a layered architecture composed of Controller, Service, Repository, and Security components.

The AuthController handles incoming requests, while the AuthService processes authentication logic. The UserRepository manages database interactions, and security-related operations are handled by the PasswordEncoder and TokenProvider. The User class represents the database entity. This structure promotes separation of concerns and maintainable system architecture.

5.5. Sequence Diagram



The Sequence Diagram describes how system components interact during registration, login, and logout. It shows communication between the User, React UI, Spring Boot API, and Database.

During registration and login, the frontend sends requests to the backend, which processes the data and interacts with the database. Upon successful authentication, access to protected resources is granted. The logout process invalidates the active session. This diagram demonstrates the request-response flow and client-server interaction within the system.

6. Appendices

Appendix A: API Endpoint Specifications

- POST /api/auth/register
    - ○ Request body schema (JSON)
    - ○ Response codes (200, 400, 500)
    - ○ Example request/response
- POST /api/auth/login
    - ○ Request/response format (JSON)
    - ○ Uses HTTP Basic Authentication
    - ○ Response codes (200, 401, 500)
- POST /api/auth/logout
    - ○ Authentication required
    - ○ Response format (success message)

Appendix B: Error Code Reference

Code	Message	Cause
400	Email already registered	Duplicate registration
401	Invalid email or password	Login failed
401	Unauthorized	Accessing protected endpoint without authentication
500	Server error	Unexpected system error

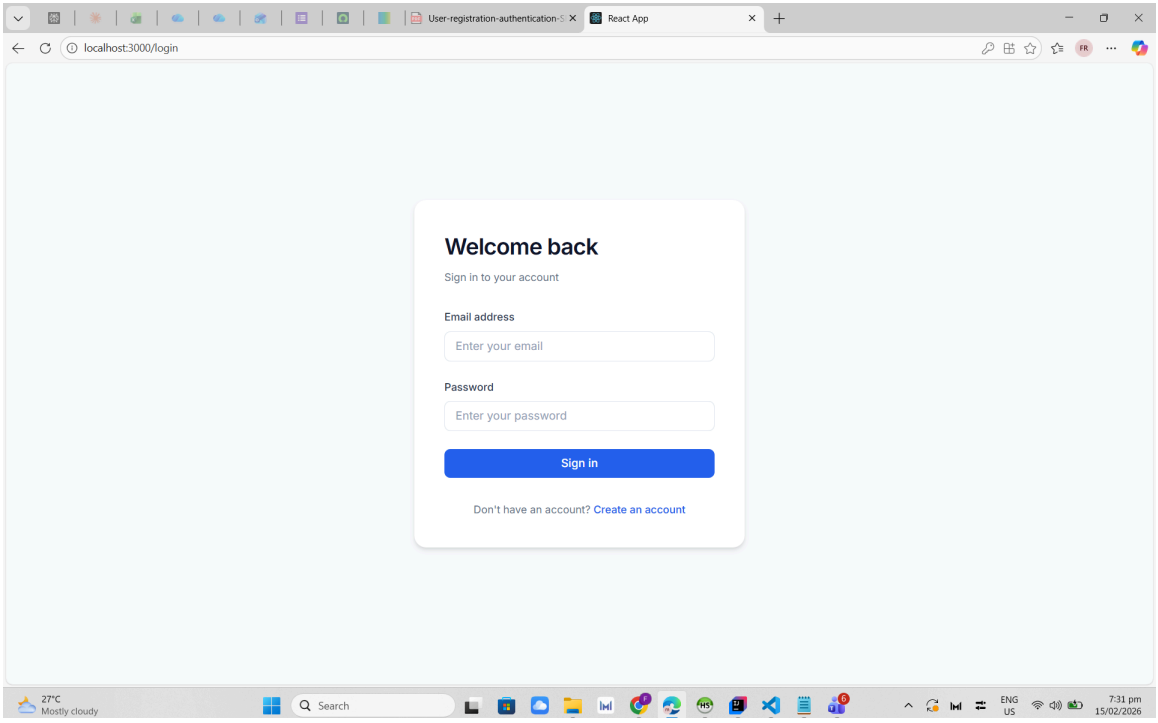
Appendix C: Technology Stack

- Frontend: React
- Backend: Spring Boot
- Security: Spring Security (HTTP Basic, BCrypt)
- Database: MySQL
- Database Tool: HeidiSQL
- Tools: Maven, IntelliJ IDEA, VS Code, Git & GitHub

7. Screenshots

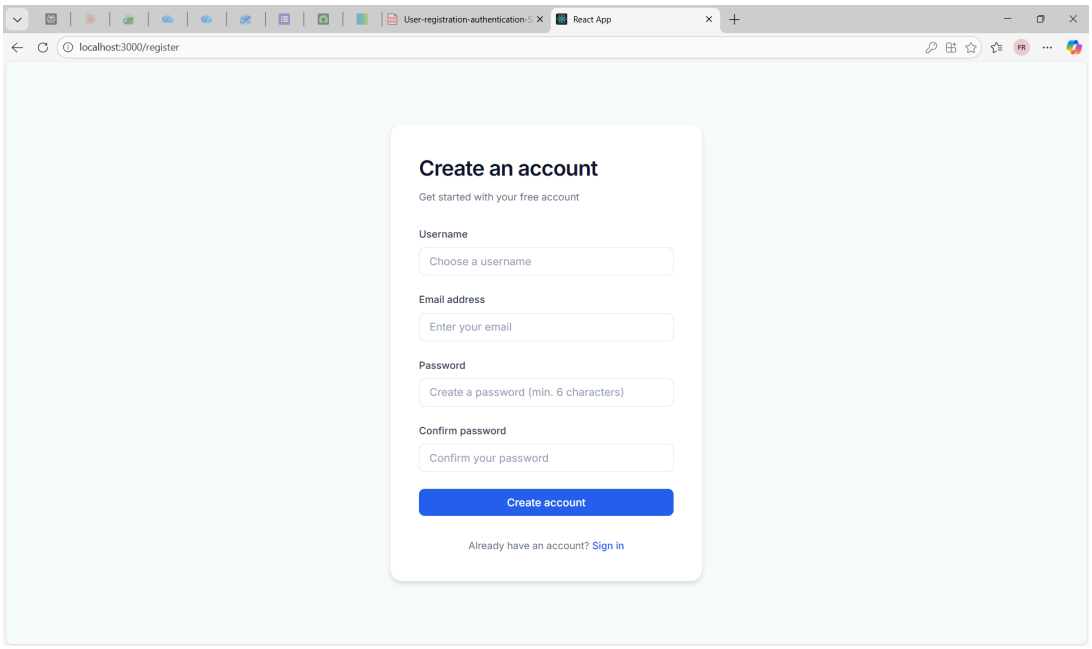
7.1. Login Page

Allows registered users to authenticate using email and password. Upon successful authentication, the system redirects the user to the dashboard.



7.2. Registration Page

Allows a guest user to create a new account by providing username, email, and password. The system validates inputs and stores encrypted credentials in the database.

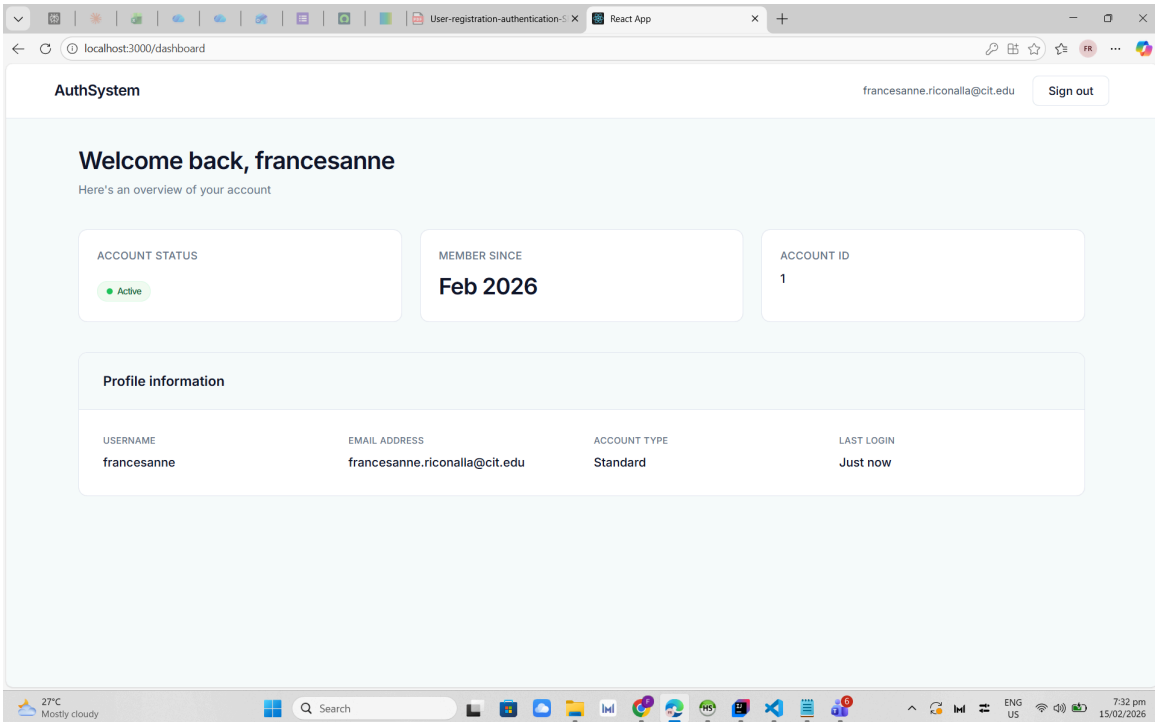


7.3. Dashboard Page

Displays authenticated user information including:

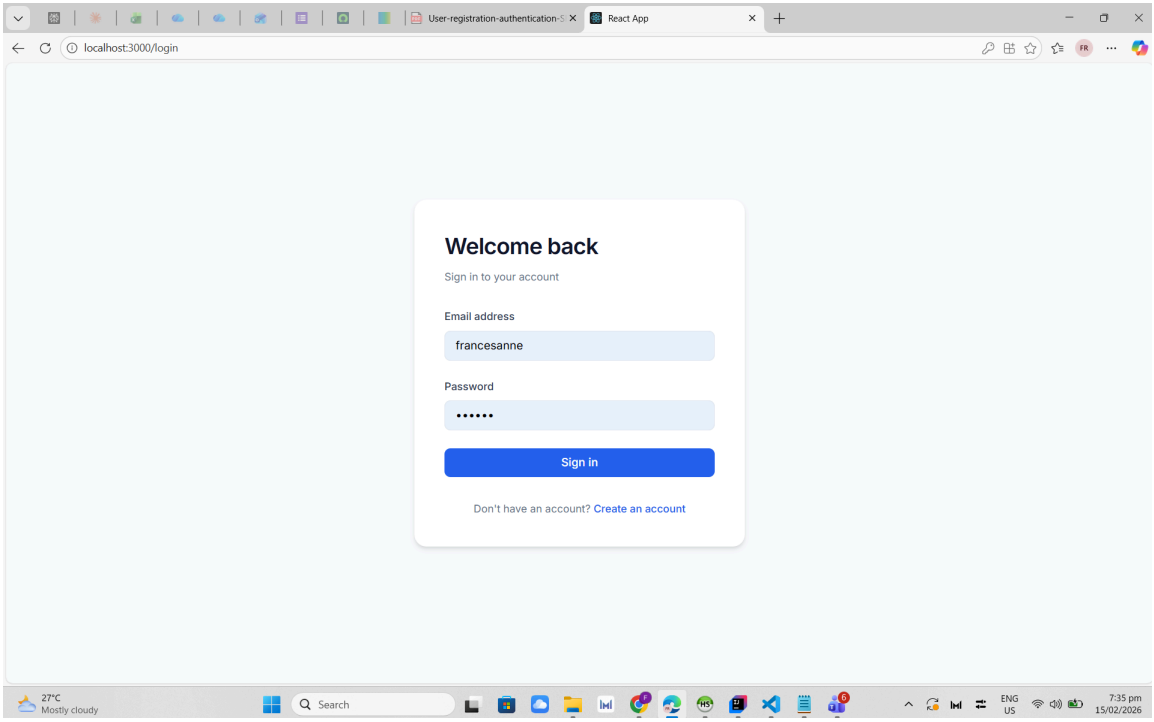
- Username
- Email address
- Account status
- Member since
- Account ID
- Last login

This page is protected and accessible only to authenticated users.



7.4. Logout Page

When the user clicks "Sign Out", the authentication token is cleared and the user is redirected back to the Login page. Protected routes become inaccessible.



7.5. Users Table (HeidiSQL View)

This screenshot shows the users table in the MySQL database after successful user registration. It confirms that user data is stored correctly, including encrypted passwords (BCrypt hash), account status, role, and timestamps.