PROFILEBOOK

CAPSTONE PROJECT Full-Stack Web Application (Angular + [ASP.NET](http://asp.net) Core)

**Submitted By : Richa Kumari**

Date: September 07,2025

BATCH : WIPRO-NGA-FSDA-FY26-C1

**INSTRUCTOR : PARTH SHUKLA**

* Frontend: Angular
* Backend: [ASP.NET](http://asp.net) Core MVC with Web API
* Database: SQL Server

**TABLE OF CONTENTS**

1. Problem Definition and Objectives

2. User Stories & Admin Stories

3. Frontend & Backend Architecture

4. Component Breakdown & API Design

5. Database Design & Storage Optimization

6. Entity-Relationships

7. Project Demonstration with Screenshots

8. Database Tables

9. Conclusion

10. Submission Checklist

**1.Problem Definition and Objectives**

**1.1 Problem Statement:**

ProfileBook is a full-stack web application developed using Angular (frontend) and ASP.NET Core Web API (backend), with SQL Server as the database. The system is designed to provide a centralized and secure way to manage user profiles, posts, and interactions in a social-media-like environment.

**1.2 Objectives:**

* Provide a centralized platform for managing user profiles.
* Enable secure login and registration using authentication mechanisms (JWT).
* Provide role-based access (User vs Admin) for better control.
* Integrate database optimization for fast profile queries.
* Allow administrators to perform user management tasks (view, update, delete).
* Ensure API-driven design for scalability and easy integration with other systems.

**2.User Stories & Admin Stories**

**2.1 User Stories:**

* As a new user, I want to register with my details so that I can create my profile.
* As a registered user, I want to log in securely so that I can access my personal profile.
* As a user, I want to edit/update my profile information so that it remains accurate.
* As a user, I want to view my profile so that I can see my stored data.
* As a user, I want to change my password for security.

**2.2 Admin Stories:**

* As an admin, I want to view all registered users so that I can manage the system.
* As an admin, I want to update or delete user profiles in case of incorrect or unauthorized entries.
* As an admin, I want to monitor login activities for security purposes.
* As an admin, I want to reset user credentials when required.

**3. Frontend & Backend Architecture**

**3.1 Technology Stack:**

* Frontend: Angular (UI pages like Login, Registration, Profile Page)
* Backend: ASP.NET Core Web API (controllers, authentication, business logic)
* Database: SQL Server (user profiles, login data, audit logs)
* Authentication: JWT-based token authentication
* UI Styling: Bootstrap
* API Documentation: Swagger for Web API

**3.2 System Flow:**

* User interacts with frontend pages (Login, Registration, Profile).
* The Frontend sends requests to the ASP.NET Core Web API.
* The Web API validates inputs, performs business logic, and communicates with the Database.
* JWT tokens are used for authentication and authorization.
* Admin uses the system with extended permissions to manage users.

**4. Component Breakdown & API Design**

**4.1.Frontend Components:**

* Login Component – Handles authentication using API.
* Registration Component – Allows new users to sign up.
* Profile Component – Displays and updates user information.
* Admin Dashboard Component – Lists all users with CRUD operations.

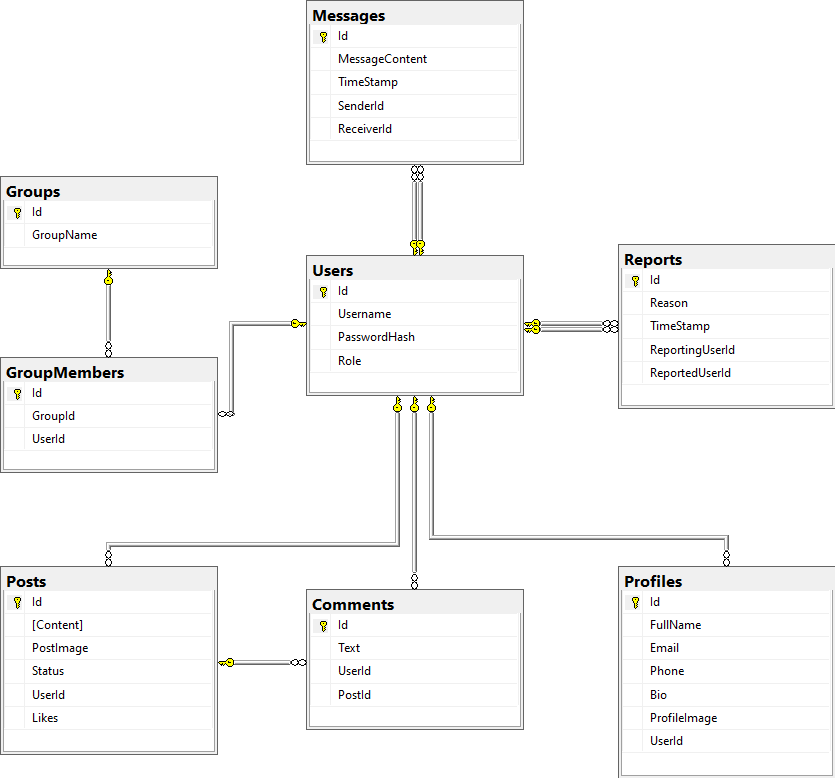
**4.2.Backend Components:**

* Controllers: AuthController, UserController, AdminController
* Services: Handle authentication, validation, and business logic
* Repositories: Communicate with the database using Entity Framework Core

**4.3.API Endpoints:**

* POST /api/auth/register – Register new user
* POST /api/auth/login – Authenticate user and return JWT
* GET /api/users/{id} – Get user profile by ID
* PUT /api/users/{id} – Update user profile
* DELETE /api/users/{id} – Delete user (Admin only)
* GET /api/admin/users – Get all users (Admin only)

**5. DATABASE DESIGN AND STORAGE OPTIMIZATION**



**5.1 ER-Diagram of ProfileBook**

Relationship Between User and Admin

* Users Table contains all system accounts.
* Role column determines if the account is a User or an Admin.
* Admins are also Users → they just have additional permissions.

**6. Entity Relationships**

**6.1. Users (Role = User/Admin)**

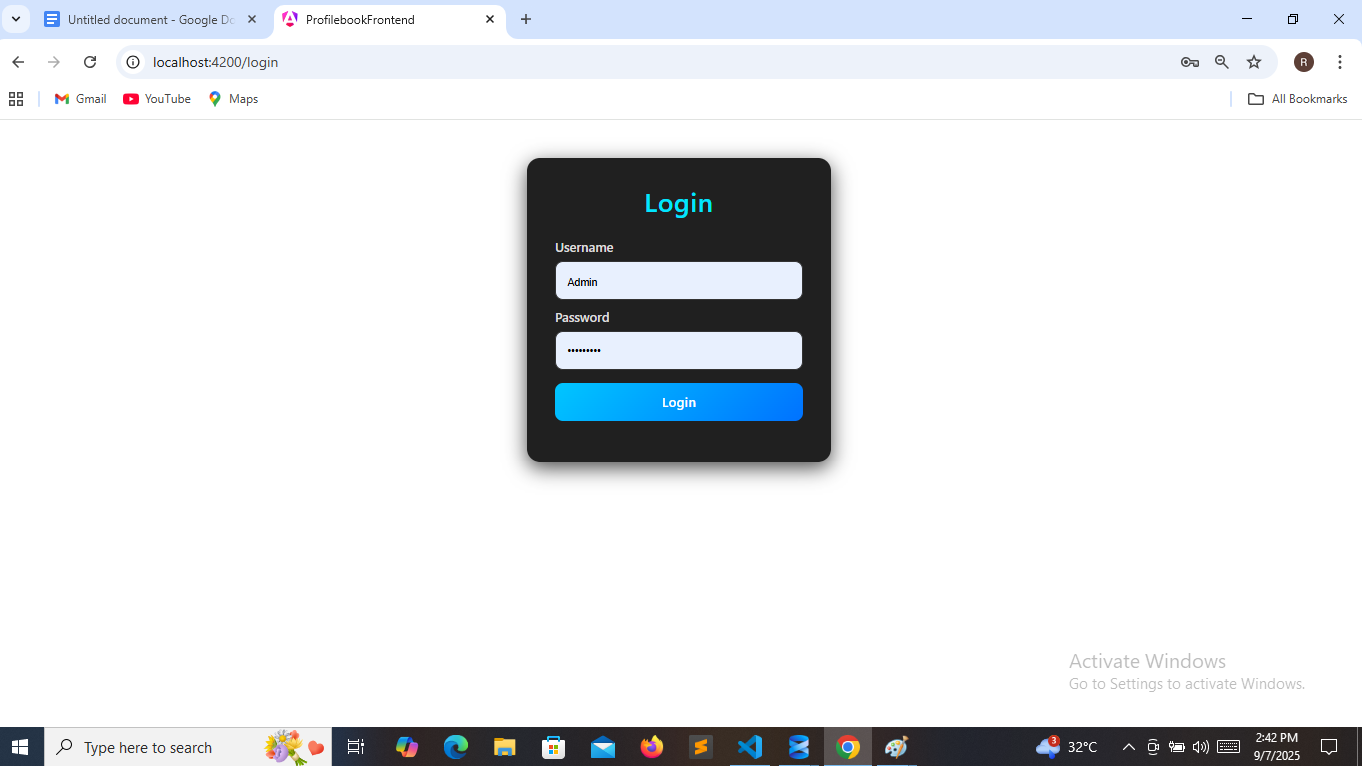
* Users → Posts → One-to-Many (a user can create multiple posts, admins typically don’t create posts but can approve them).
* Users → Messages → One-to-Many (a user can send/receive many messages).
* Users → Reports → One-to-Many (users can report others, admins can view/manage reports).

**6.2. Admin Special Relationship (via Role)**

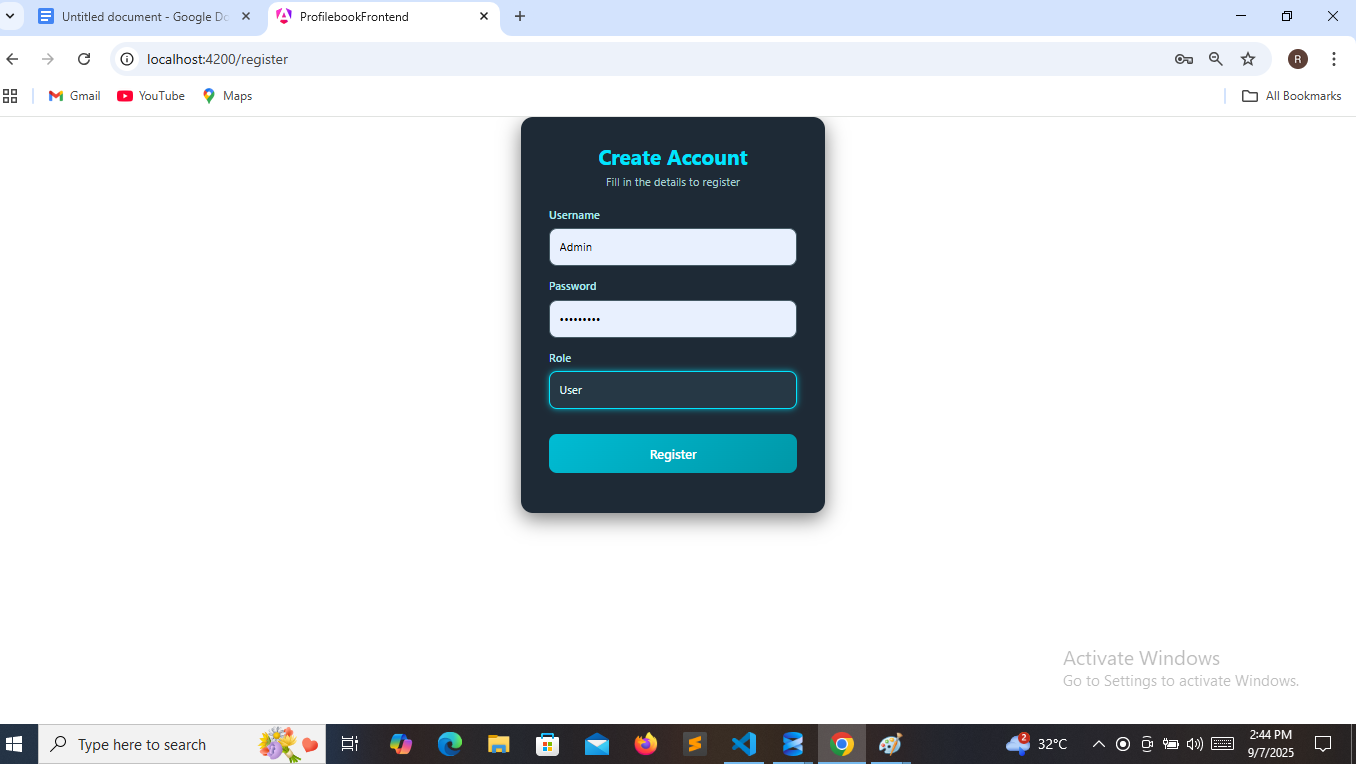
* Admins do not need a separate table.
* **Admins manage:**
* Posts → Approve/Reject user posts.
* Users → CRUD operations.
* Reports → View reported users and take actions.
* Groups → Create/manage groups.

**7. Project Demonstration via Screenshots:**

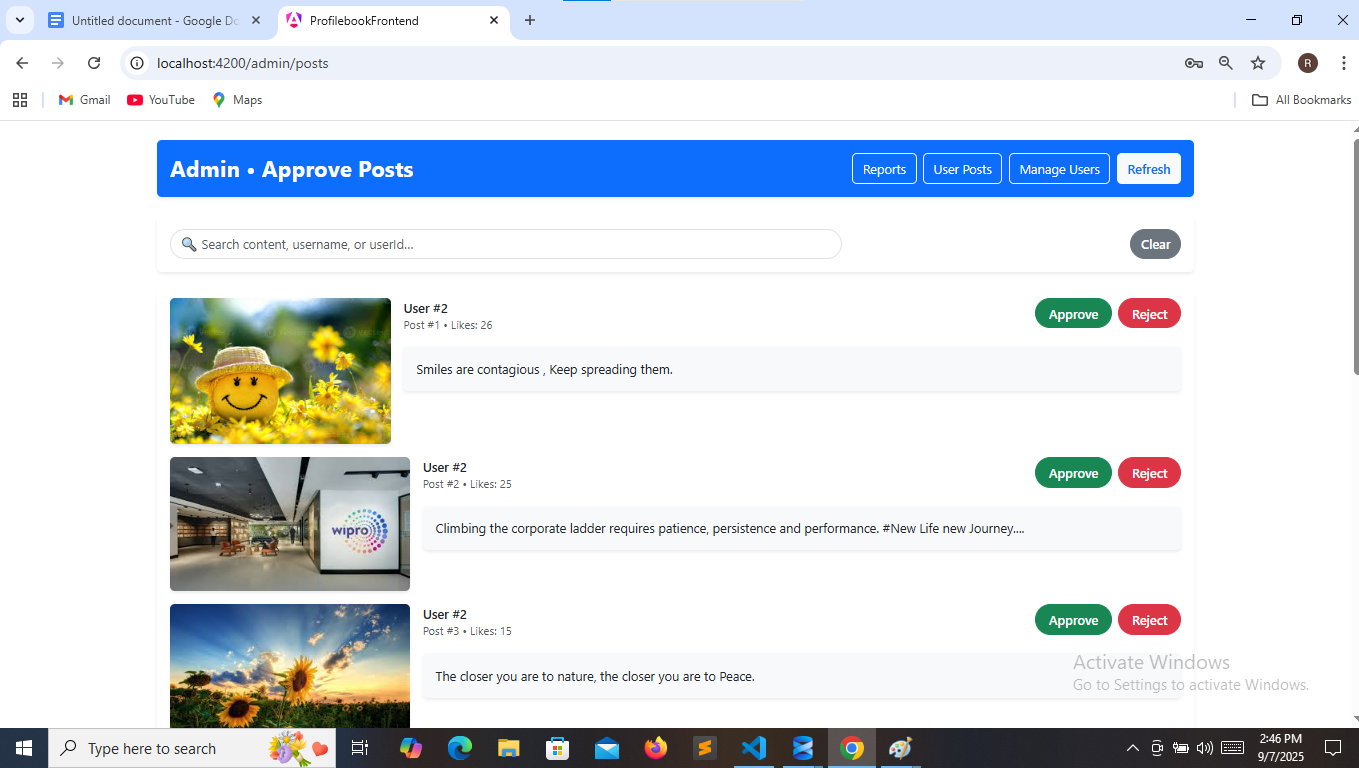
**7.1 LOGIN**

****

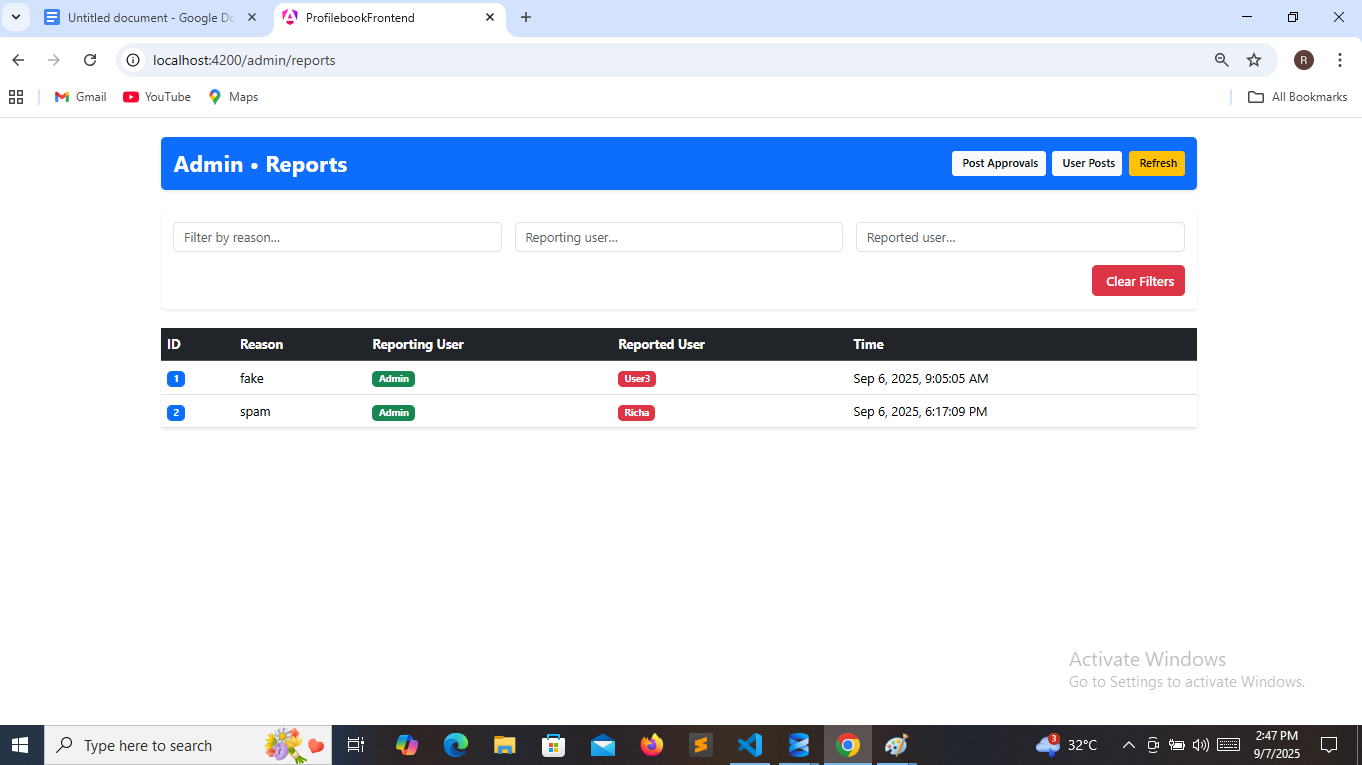
**7.2 REGISTER**

****

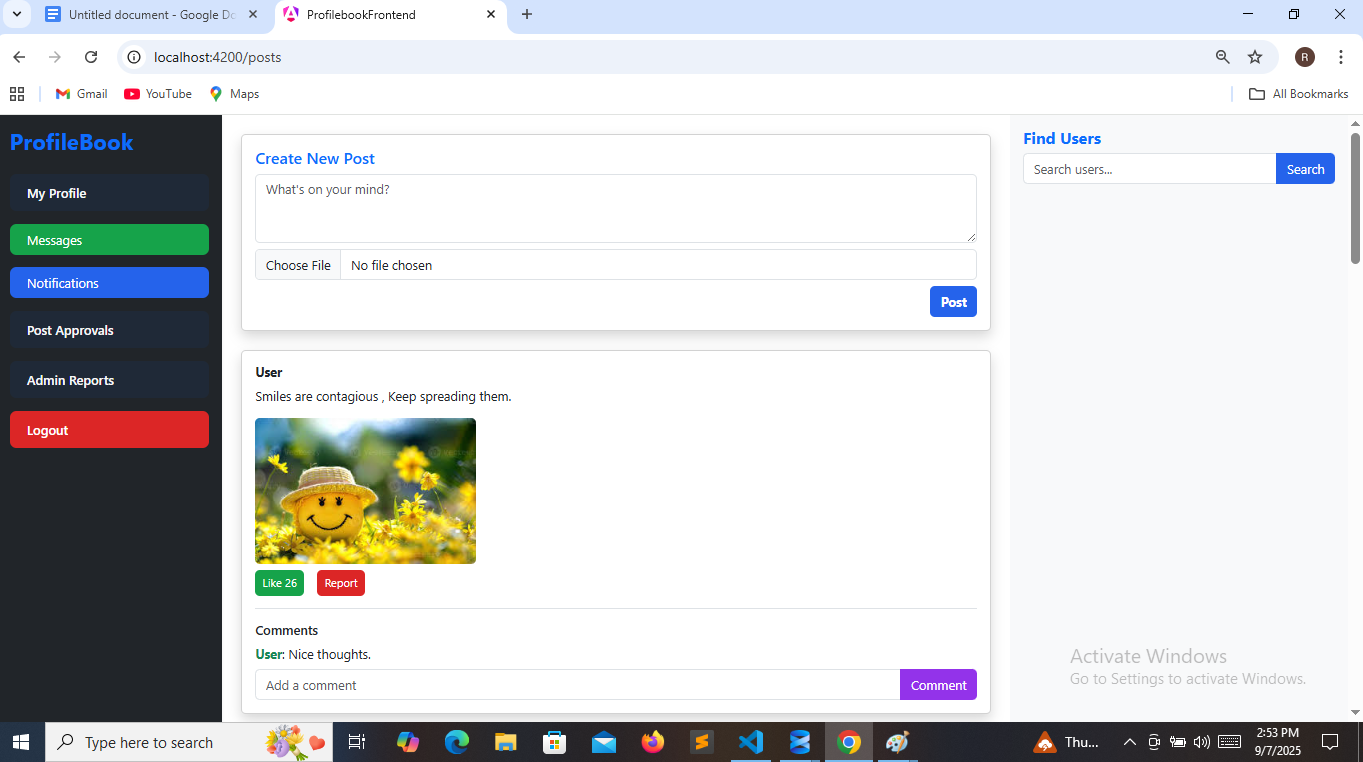
**7.3 ADMIN APPROVE / REJECT POSTS**

****

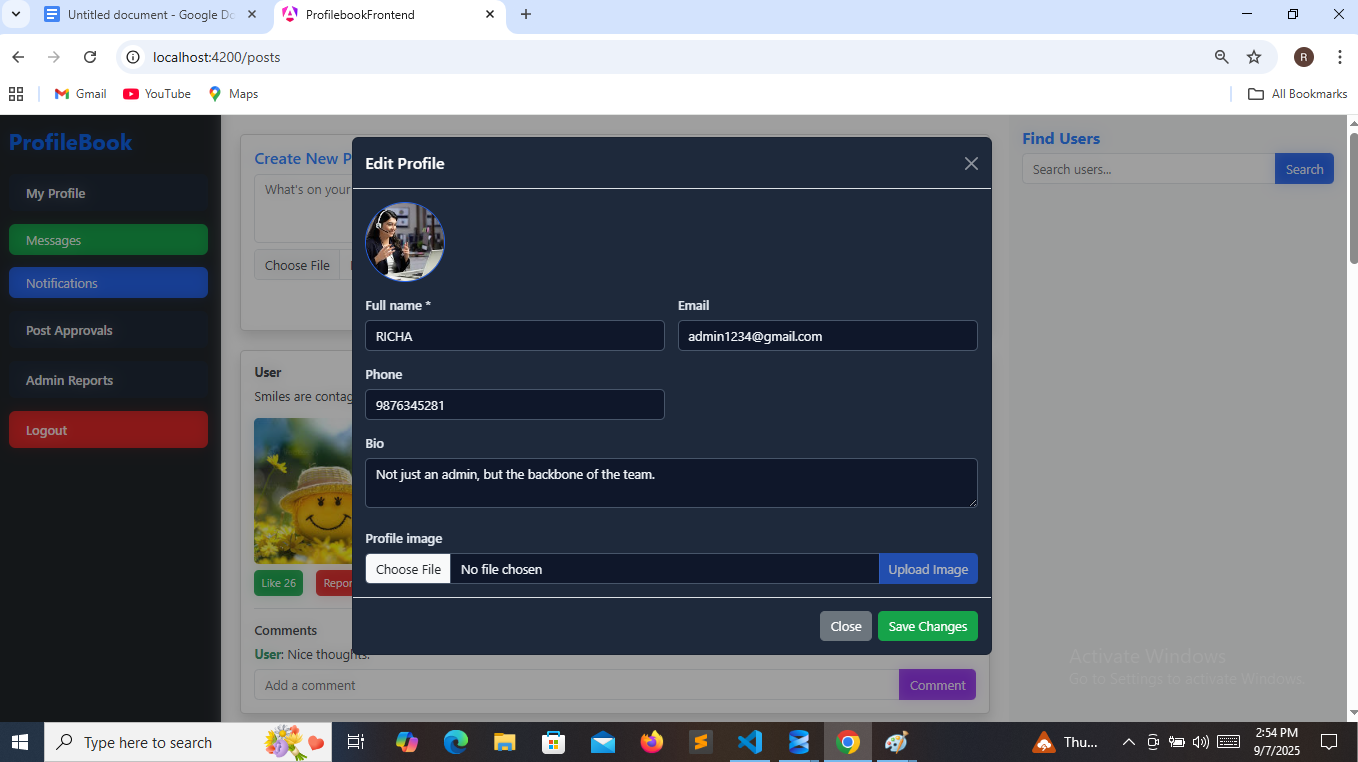
**7.4 ADMIN REPORT USERS**

****

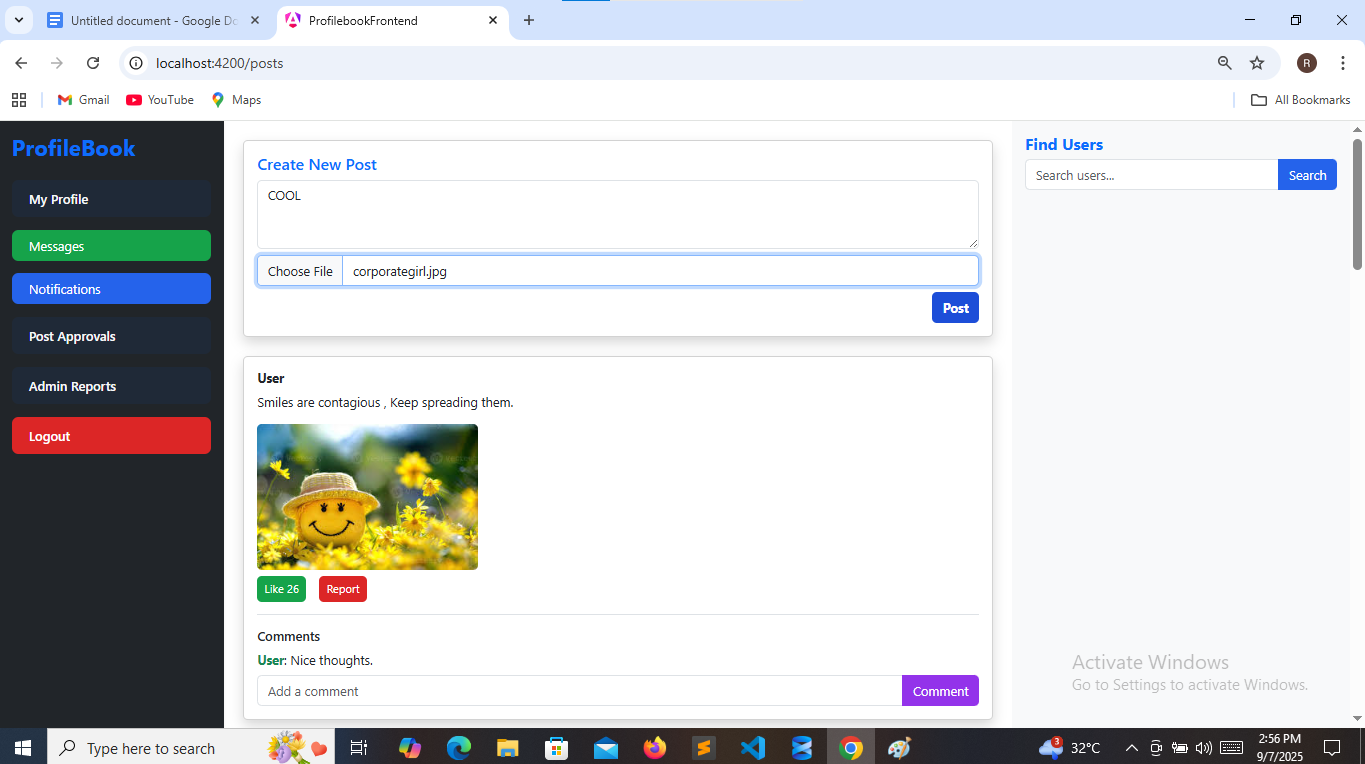
**7.5 PROFILEBOOK HOME PAGE**

****

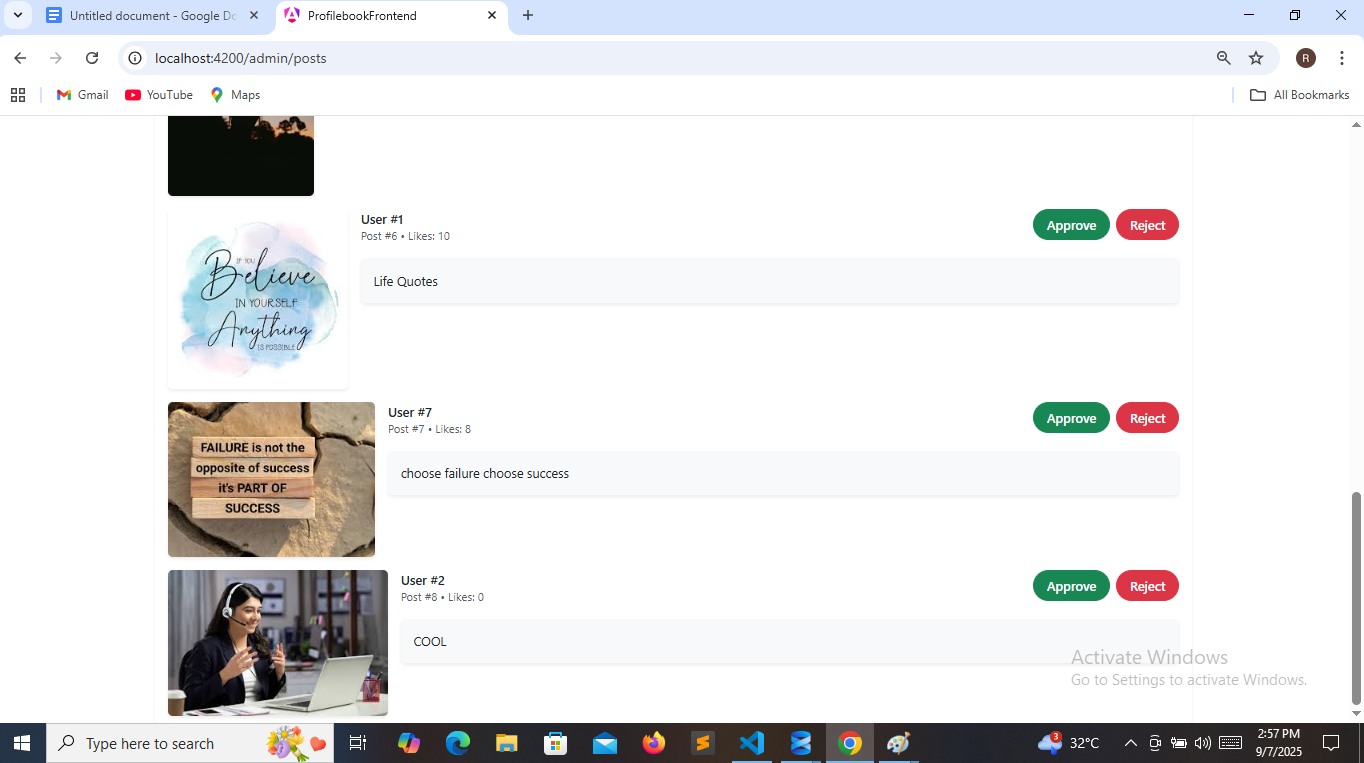
**7.6 MY PROFILE**

****

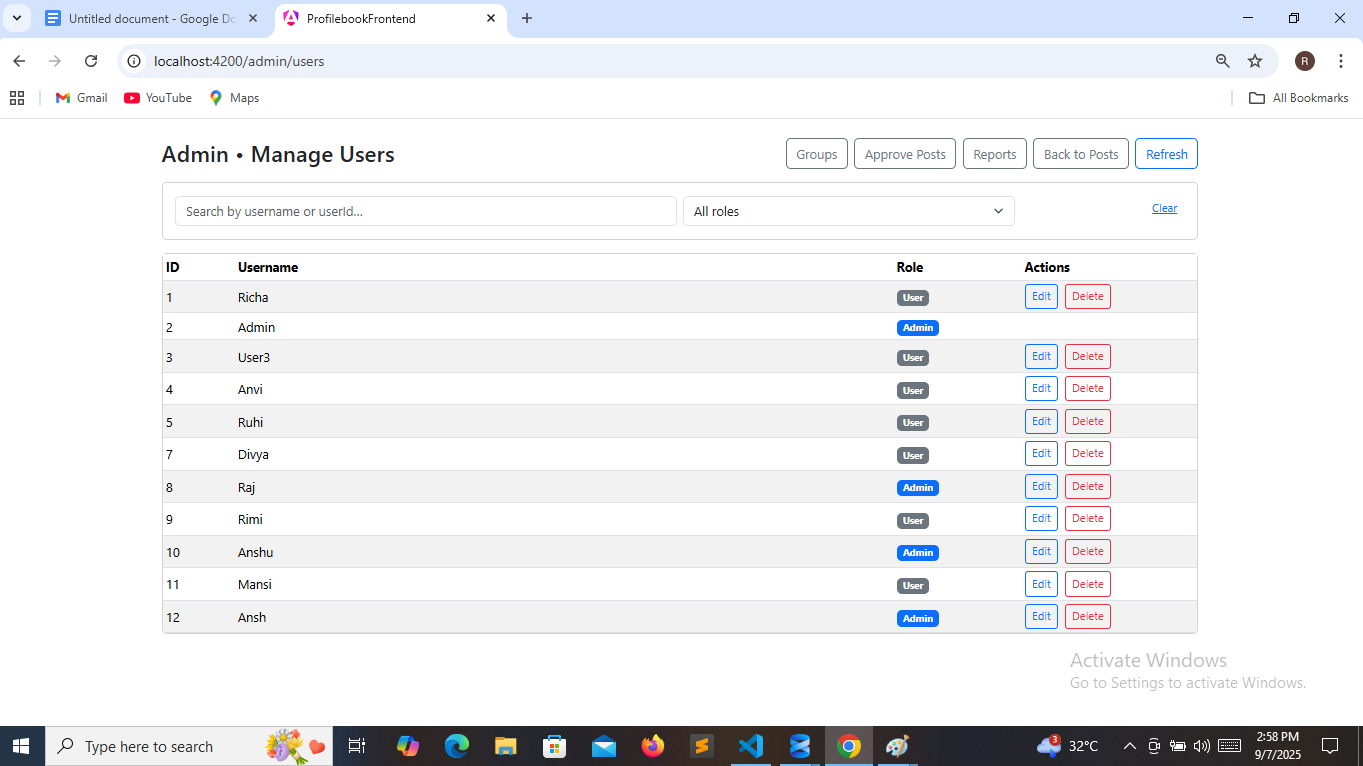
**7.7 CREATE NEW POST**

****

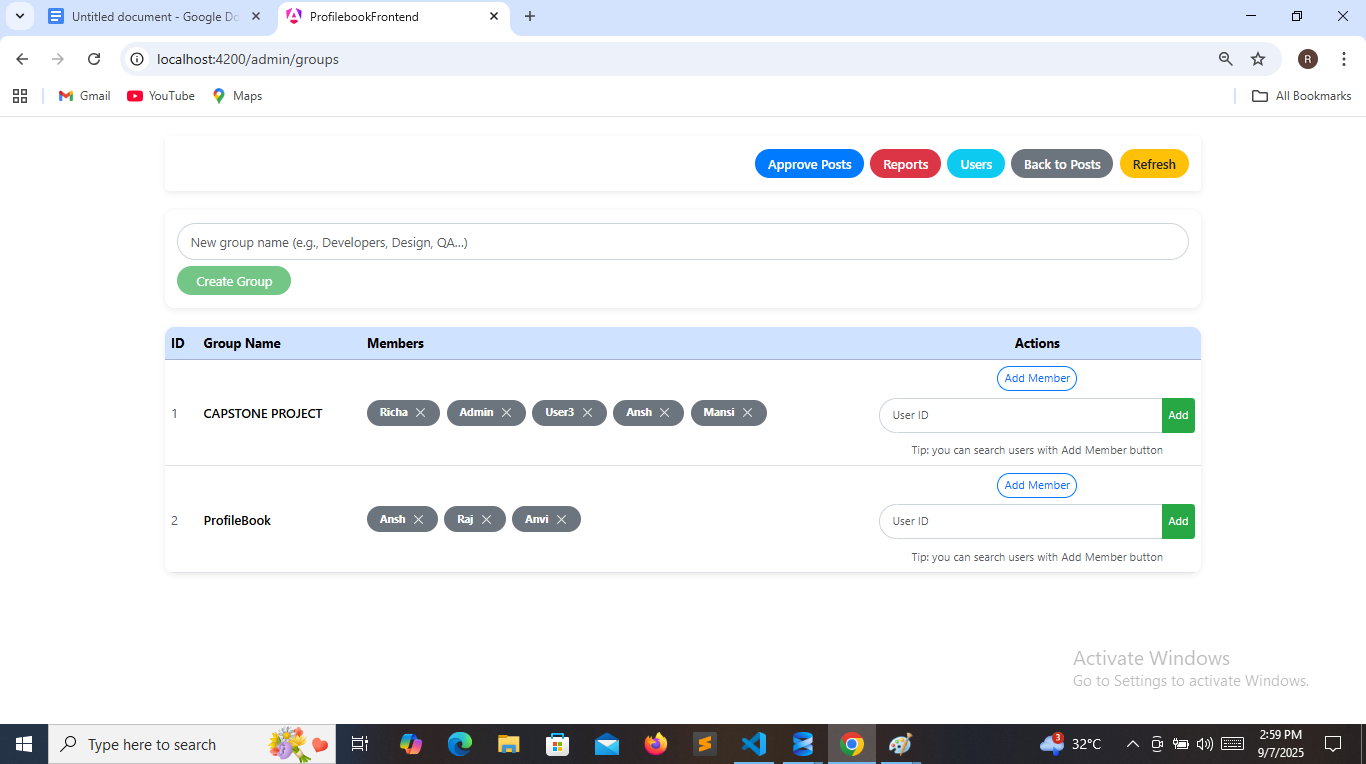
**7.8 POST APPROVAL DASHBOARD**

****

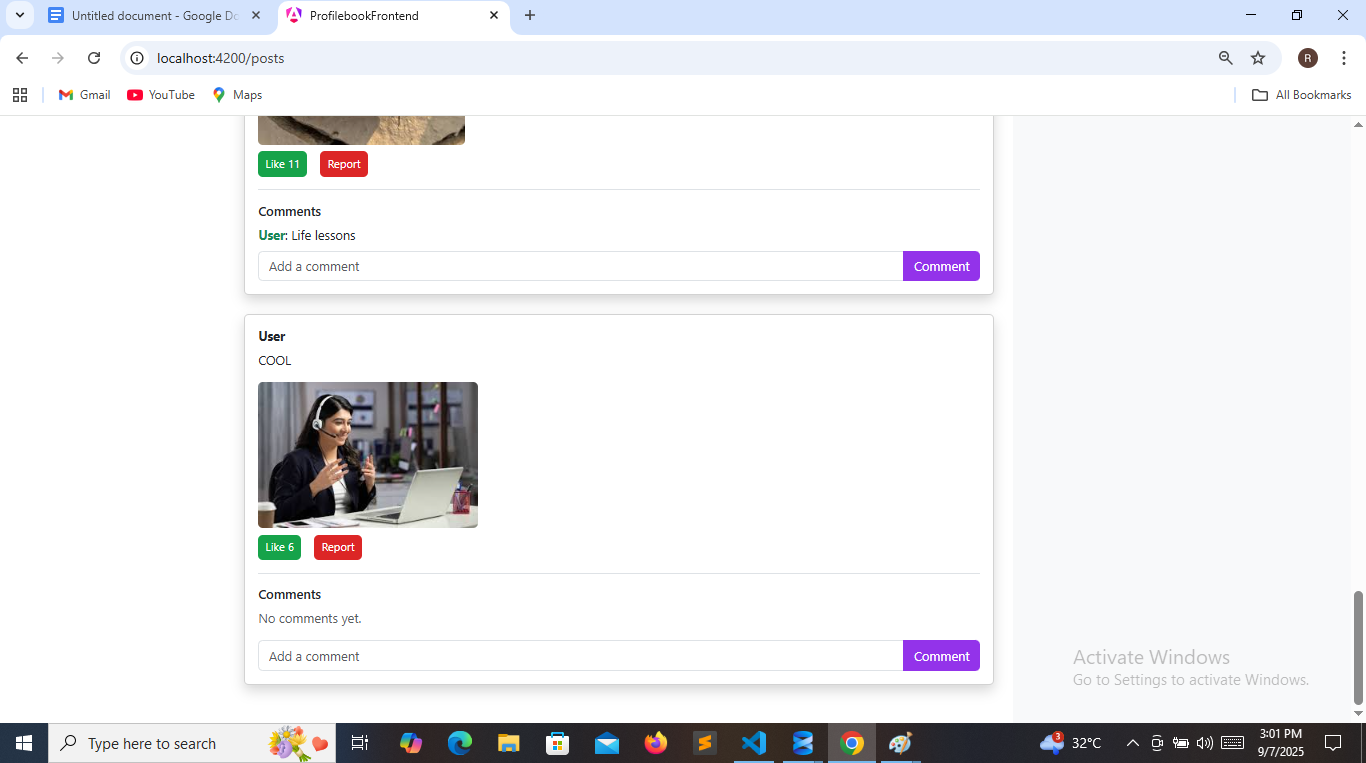
**7.9 ADMIN MANAGE USERS**

****

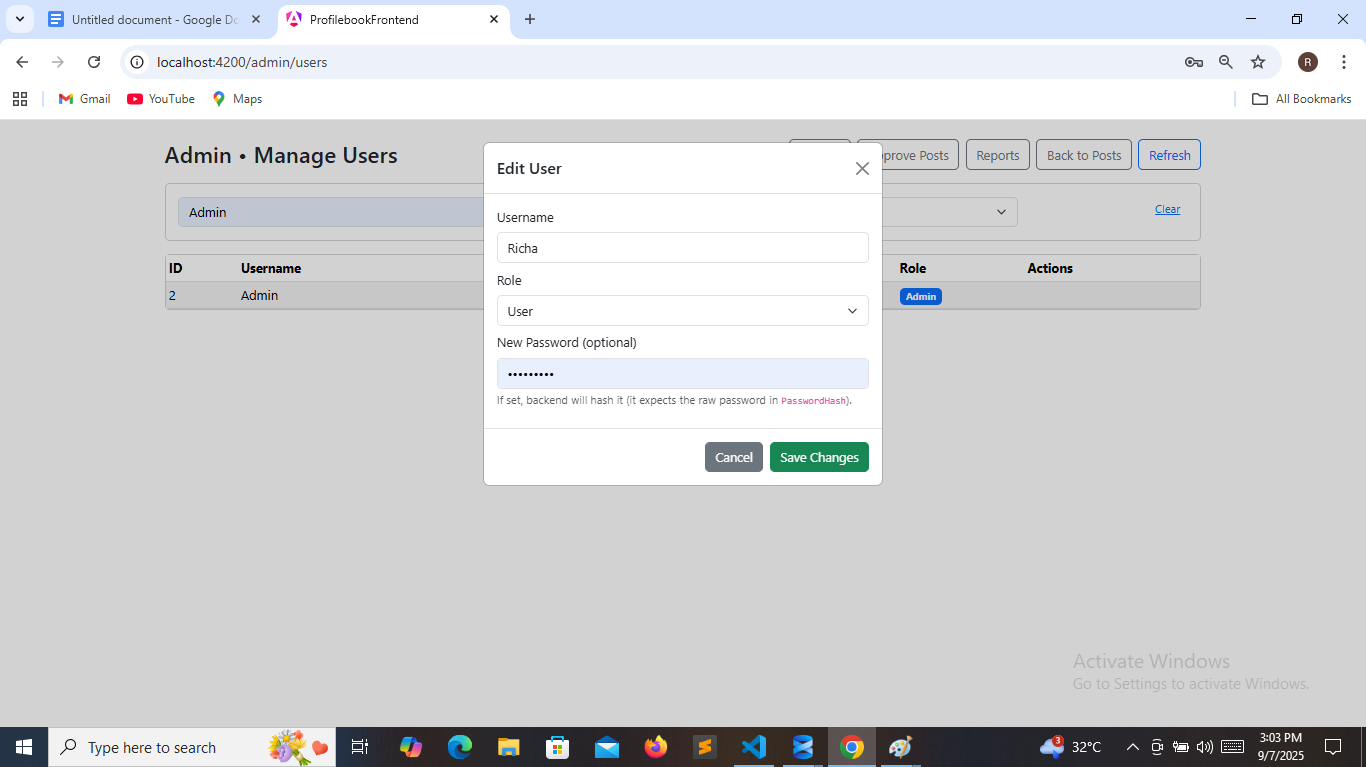
**7.10 CREATE GROUPS**

****

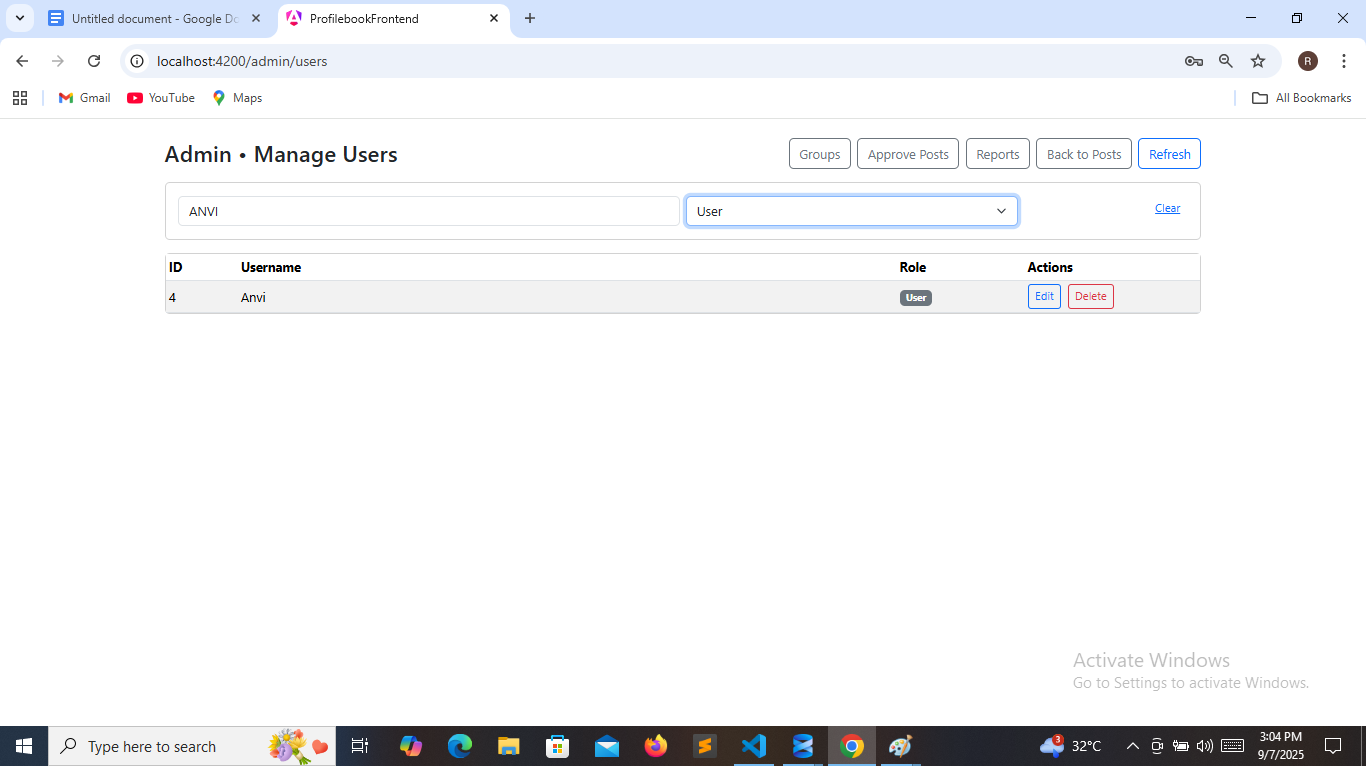
**7.11 LIKE , COMMENT AND REPORT USERS**

****

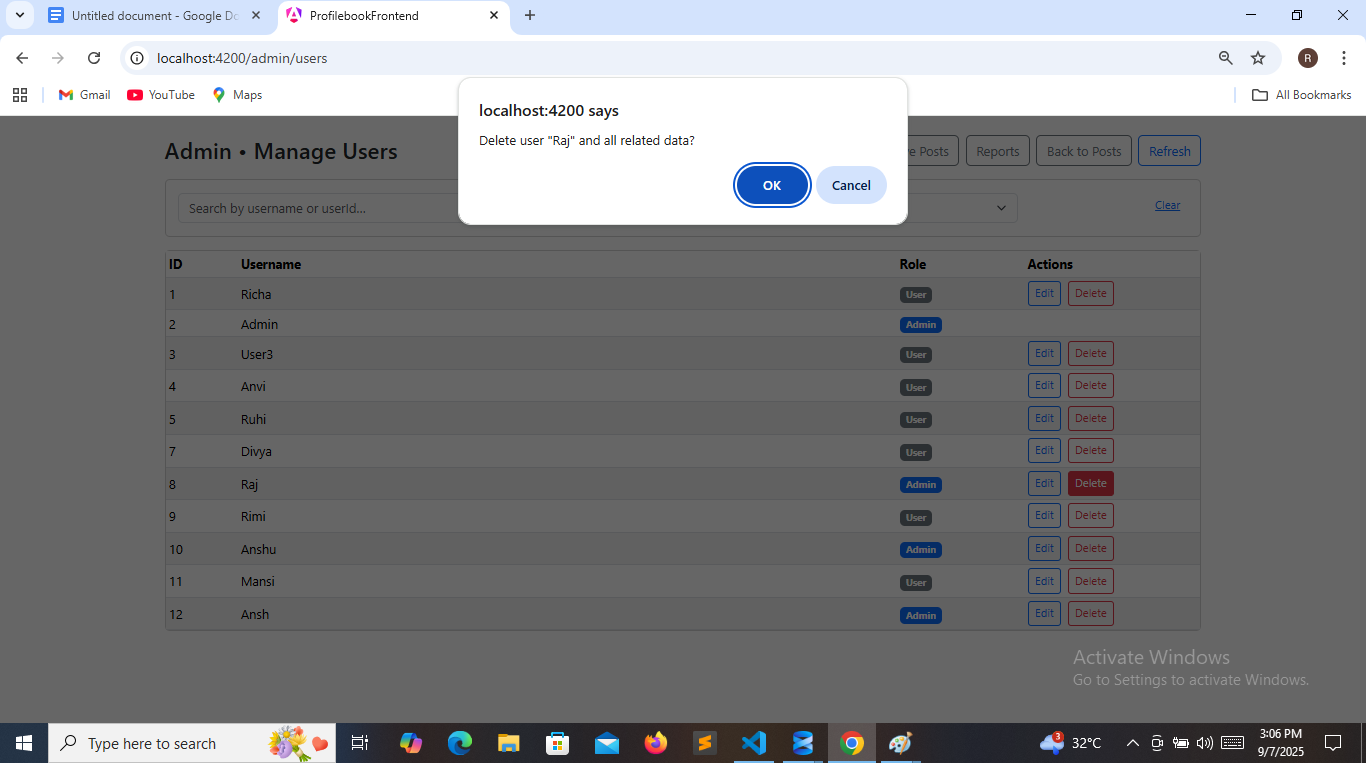
**7.12 USER MANAGEMENT INTERFACE**

****

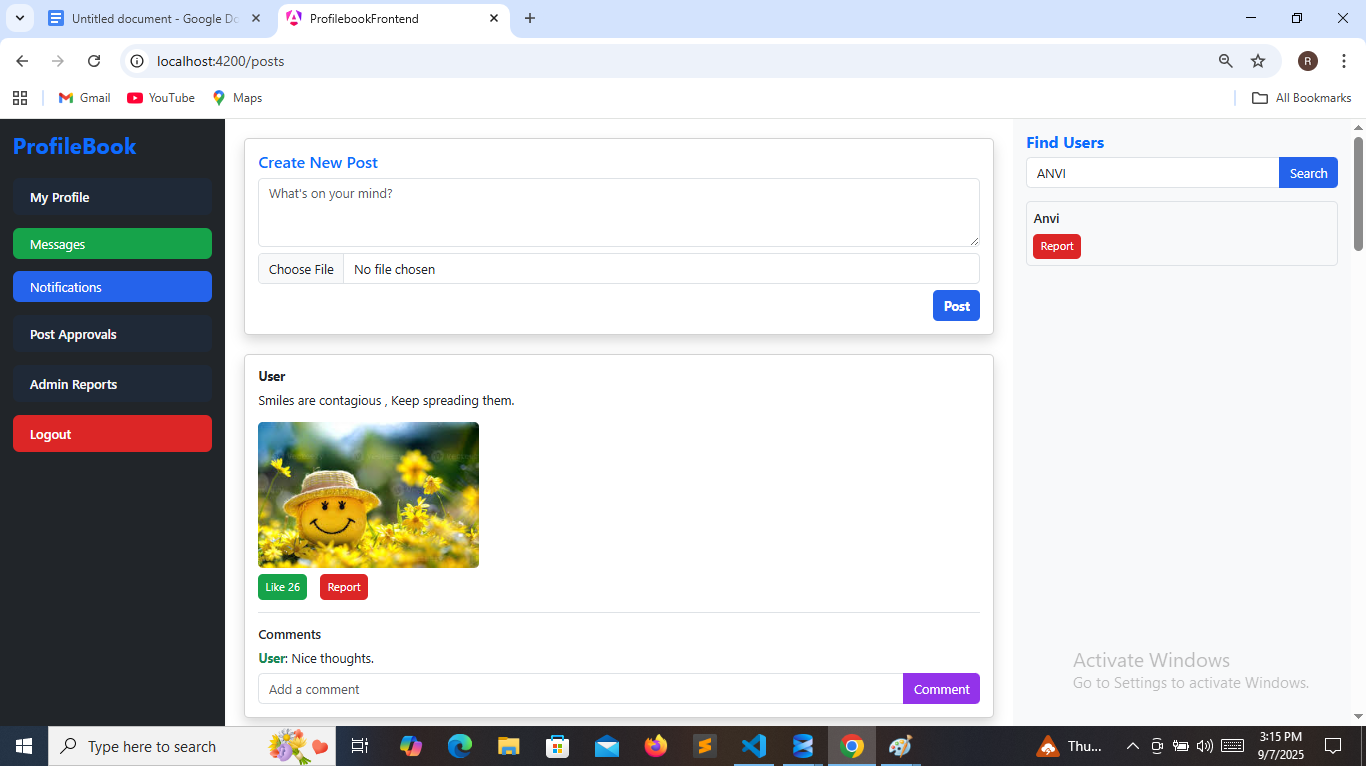
**7.13 SEARCH BY USERNAME AND ROLE**

****

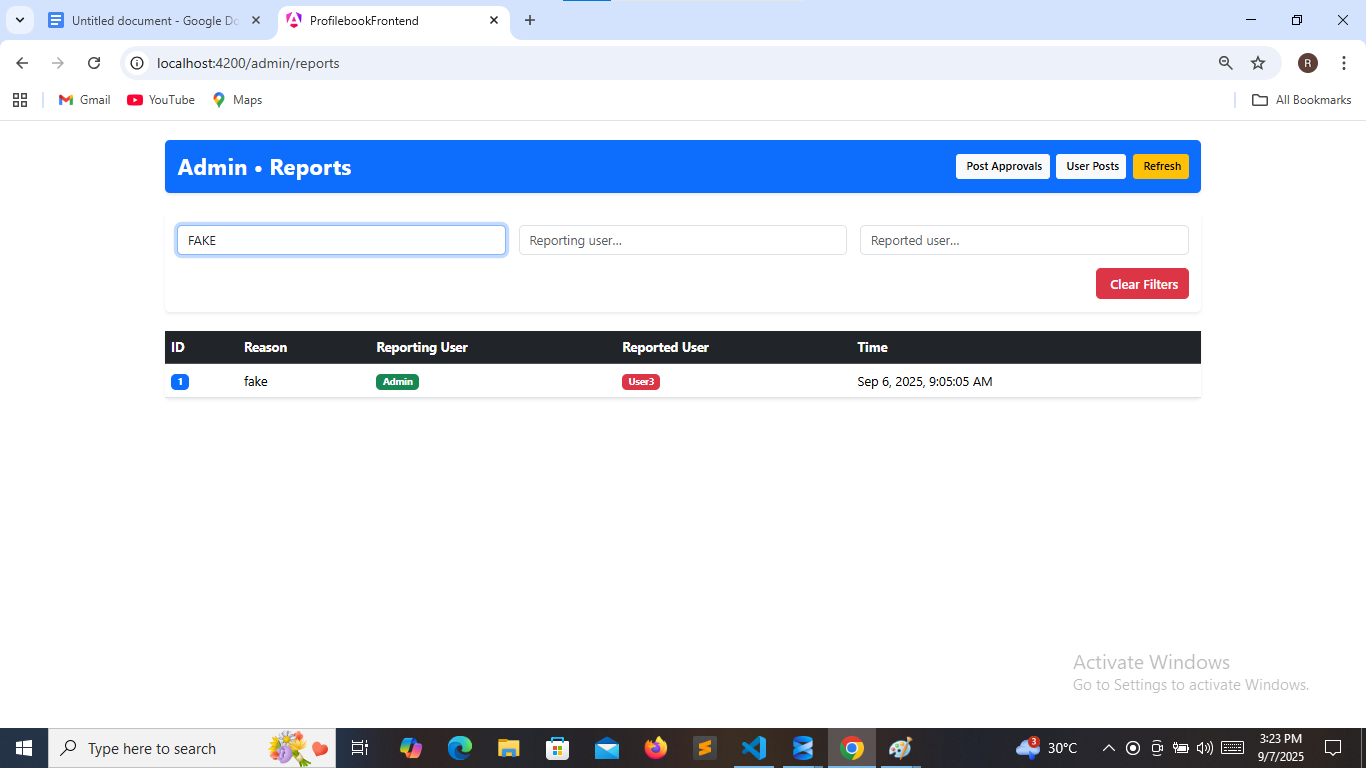
**7.14 ADMIN CAN DELETE USER**

****

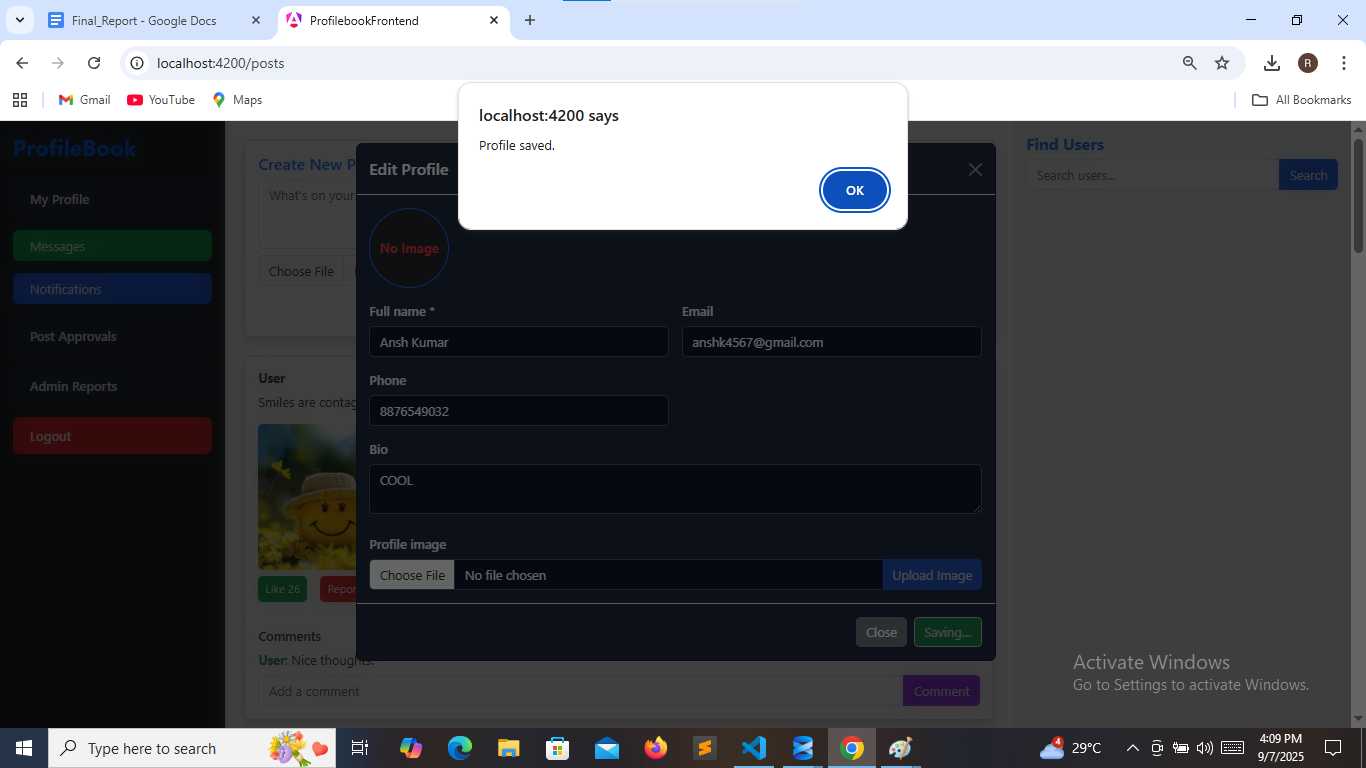
**7.14 SEARCH USERS**

****

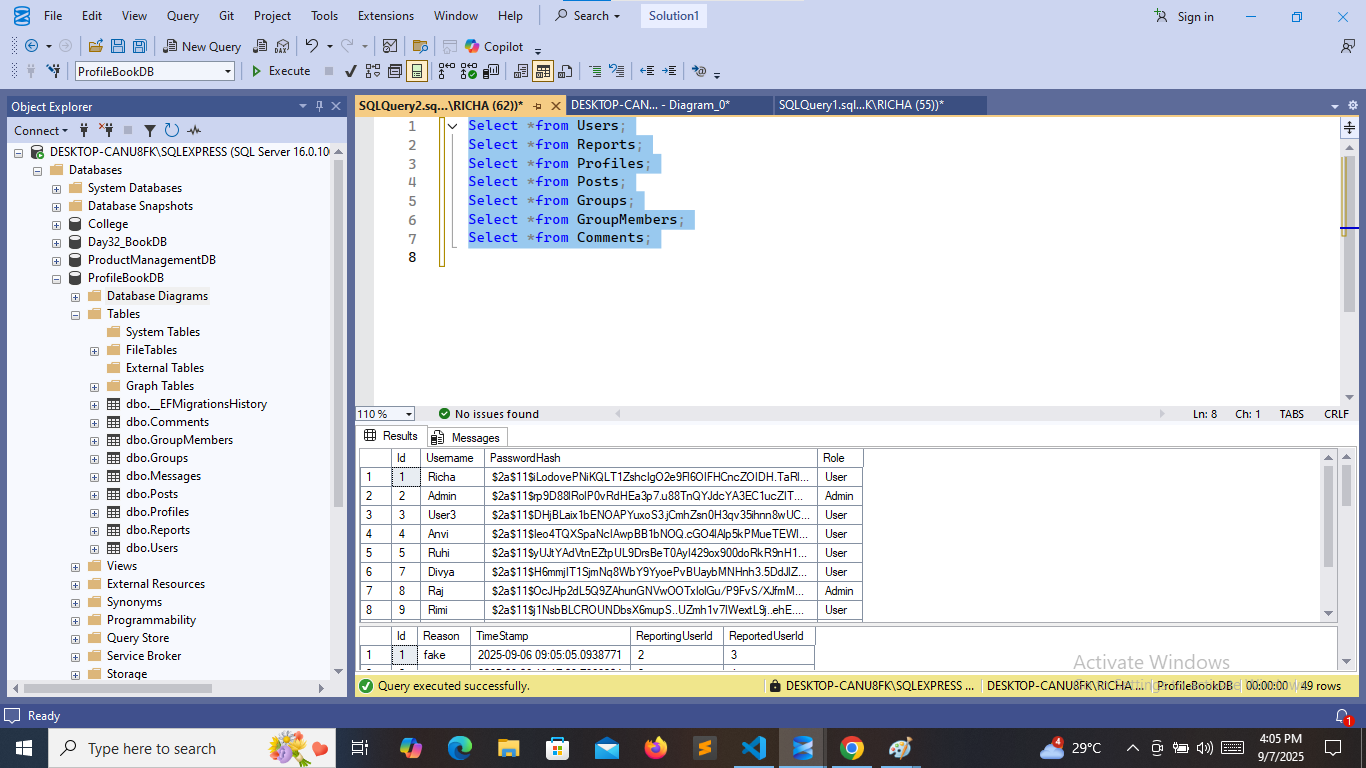
**7.15 SEARCH REPORTED USERS BY REASON**

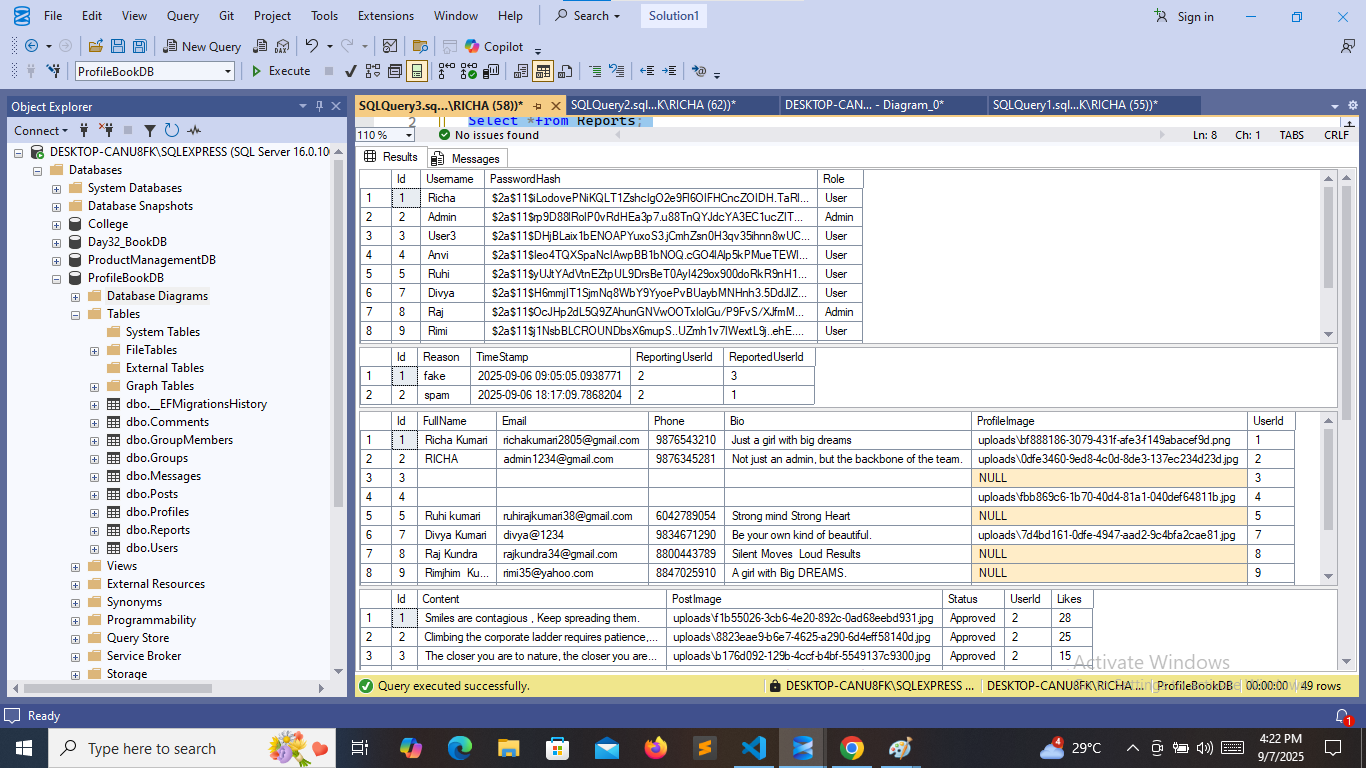
****

**7.16 PROFILE SAVED**

****

**8. DATABASE TABLES**

****

****

**9. CONCLUSION**

The ProfileBook application demonstrates the successful implementation of a full-stack web solution using Angular for the frontend, ASP.NET Core for the backend, and SQL Server for database management.

By integrating these technologies, the project achieves:

* A responsive and user-friendly interface through Angular.
* Secure and scalable backend services using ASP.NET Core with RESTful APIs.
* Efficient data storage and retrieval with SQL Server and Entity Framework Core.

The system provides both User and Admin functionalities, ensuring role-based access, profile management, and smooth data handling. This project highlights the power of combining modern frontend frameworks with robust backend architecture, resulting in a reliable, maintainable, and real-world applicable application.

Overall, the project not only meets its objectives of managing user profiles efficiently but also serves as a strong demonstration of practical full-stack development skills.

**10. Submission Checklist**

* **ProfileBook\_Final\_Report\_Richa\_Kumari.docx.**
* **Frontend Source ZIP File.**
* **Backend Source ZIP File.**
* **Implemented Screenshots of Project in Section 7.**