SECURE LOGIN SYSTEM WITH USER ROLE MANAGEMENT

Project Report

On

“Secure login System with User Role Management”

DEVELOPED BY:

NAME: SALONI PAREKH

SUBMISSION DATE: 31/08/2025

SECURE LOGIN SYSTEM WITH USER ROLE MANAGEMENT

INDEX

CHAPTER-1

❖ PROBLEM IDENTIFICATION DEFINITION & MODIFICATION:

➢ Definition:

- A **Secure Login System with User Role Management** is a web-based application that provides controlled access to system resources by verifying user identities (authentication) and granting permissions based on predefined roles (authorization). The system ensures that sensitive user data, such as passwords, is securely stored using cryptographic hashing, while also protecting against common security threats like brute force attacks and SQL injection.
- This system implements **Role-Based Access Control (RBAC)**, allowing administrators to manage users and granting different levels of access to Admins and regular Users.

➢ Abstract:

- This project presents the design and development of a **Secure Login System with User Role Management**, aimed at enhancing web application security by integrating user authentication, authorization, and role-based access control. The system is built using **HTML, CSS, Flask (Python) and MongoDB**, ensuring a structured client-server architecture.

- The project allows users to register**,** log in**,** and access dashboards based on their assigned roles. Admins can view and manage all users, while regular users have limited access to their own dashboard. To strengthen security, features such as password hashing (bcrypt), input validation, CAPTCHA, and account lockout mechanisms are implemented.

- The project follows a structured development cycle — from frontend and backend setup to testing and debugging — ensuring a practical understanding of secure coding practices. This system demonstrates key cybersecurity principles in a development environment and serves as a learning model for building secure web applications.

SECURE LOGIN SYSTEM WITH USER ROLE MANAGEMENT

➢ INTRODUCTION:

→ In the digital era, secure user authentication and authorization are essential components of any web-based application. Unauthorized access, weak passwords, and insecure session handling are some of the most common vulnerabilities exploited by attackers. To address these concerns, this project — **Secure Login System with User Role Management** — has been developed as a practical implementation of cybersecurity concepts.

→ The system provides a structured way for users to register, log in, and access different functionalities based on their assigned roles. By implementing Role-Based Access Control (RBAC**)**, the project ensures that administrators have complete control over user management, while regular users are restricted to accessing only their personal dashboards.

→ Security is a top priority in the design, achieved through password hashing (bcrypt), CAPTCHA validation, account lockout mechanisms, and protection against SQL injection attacks. The system is built with HTML, CSS, Js (frontend) and Flask/Python with MongoDB (backend) to create a scalable and secure application environment.

➢ BENEFITS SECURE LOGIN SYSTEM WITH USER ROLE MANAGEMENT:

→ Enhanced Security
→ Role-Based Access Control (RBAC)
→ Scalability
→ User-Friendly Interface
→ Educational Value
→ Foundation for Future Enhancements

SECURE LOGIN SYSTEM WITH USER ROLE MANAGEMENT

❖ PROJECT PROFILE:

Project Name :	**Secure Login System with User Role Management**
Name:	**Saloni Parekh**
Frontend :	**HTML, CSS**
Backend:	**Flask(Python)**
Database:	**MongoDB**
Security	**JWT (JSON Web Token) for authentication**
Other:	**CAPTCHA for bot prevention**
Project Type:	**Web Application**

❖ PROJECT SCOPE:

The scope of this project is to design and implement a secure login system that provides user authentication, password encryption, and role-based access control for Admin and User roles. The system ensures that only authorized users can access specific functionalities by validating credentials and enforcing role restrictions. It also integrates essential security features such as CAPTCHA, account lockout after multiple failed attempts, and input validation to prevent common web vulnerabilities.

CHAPTER-2

❖ SYSTEM REQUIREMENT SPECIFICATION:

The Secure Login System with User Role Management requires both **functional** and **non-functional** requirements to ensure proper operation and security.

❖ **Requirement Gathering and Analysis: -**

➢ **Requirement Analysis: -**

➢ Functional Requirements:

1) User registration with username, email, password, and role (Admin/User).
2) Password hashing and secure storage in the database.
3) User login with credential verification.
4) Role-based access to dashboards (Admin/User).
5) Admin functionalities: view, update, delete users.
6) Account lockout after multiple failed login attempts.
7) CAPTCHA validation during login to prevent brute-force attacks.
8) Functionality for all users.

➢ Non-Functional Requirements:

1) **Security:** Strong password policies, hashing (bcrypt), input validation.
2) **Performance:** Fast response time for login/registration requests.
3) **Scalability:** System should support multiple users simultaneously.
4) **Usability:** Simple and intuitive interface for all users.
5) **Reliability:** Database should ensure data consistency and recovery.

➢ **Performance Requirements:**

- The system should be able to handle **multiple concurrent login and registration requests** without failure.

SECURE LOGIN SYSTEM WITH USER ROLE MANAGEMENT

❖ SOFTWARE AND HARDWARE REQUIREMENT:

➢ Software Interface:

- **Frontend:** HTML, CSS
- **Backend:** Flask (Python)
- **Database:** MongoDB
- **Browser:** Chrome/Firefox/Edge (latest version)
- **OS:** Windows

➢ Hardware Interface:

Hardware requirements for insurance on internet will be same for both parties which are as follows:

- **Processor:** Intel i5 or higher.
- **RAM:** Minimum 4 GB.
- **Storage:** Minimum 500 MB free space.
- **Device:** Laptop with internet access.

➢ Communication Interfaces:

- The two parties should be connected by LAN or WAN for the communication purpose.

❖ Feasibility Study:

The objective of feasibility study is to determine whether or not the proposed system is feasible.

- **Technical Feasibility:**
  The project is technically feasible as it uses widely available technologies such as HTML, CSS, Flask(Python), and MongoDB. These technologies are open-source, well-documented, and supported across different platforms, making implementation practical.
- **Operational Feasibility:**
  The system is user-friendly, with simple interfaces for login, registration, and dashboards. Admins can easily manage users, while regular users have restricted access, ensuring smooth operation. It meets the security and usability needs of both developers and end-users.
- **Economic Feasibility:**
  The project requires minimal financial investment since all tools and frameworks used (Flask, MongoDB, JWT) are free and open-source. Development can be done on standard hardware and free IDEs (VS Code), making it cost-effective.

CHAPTER-3

❖ PROBLEM SOLUTION OUTLINE:

❖ MODULES DESCRIPTION:

1. Home Page

- Acts as the entry point of the system.
- Provides navigation links to **Login** and **Registration** pages.
- Displays basic project information and a user-friendly welcome interface.

2. Registration Page

- Allows new users to create an account with username, email, password, and role (Admin/User).
- Performs client-side and server-side validation.
- Stores user details securely in the database with password hashing.

3. Login Page

- Authenticates users with email and password.
- Includes CAPTCHA verification and account lockout after failed attempts.
- Grants access based on user role.

4. Authentication & Authorization

- Maintains secure sessions using JWT/session cookies.
- Implements Role-Based Access Control (RBAC) to restrict unauthorized access.

5. User Dashboard Page

- Provides personalized access for normal users.
- Displays account-related information and logout option.

6. Admin Dashboard Page

- Admins to view, edit, and delete user accounts.
- Provides full control over the system with restricted access.

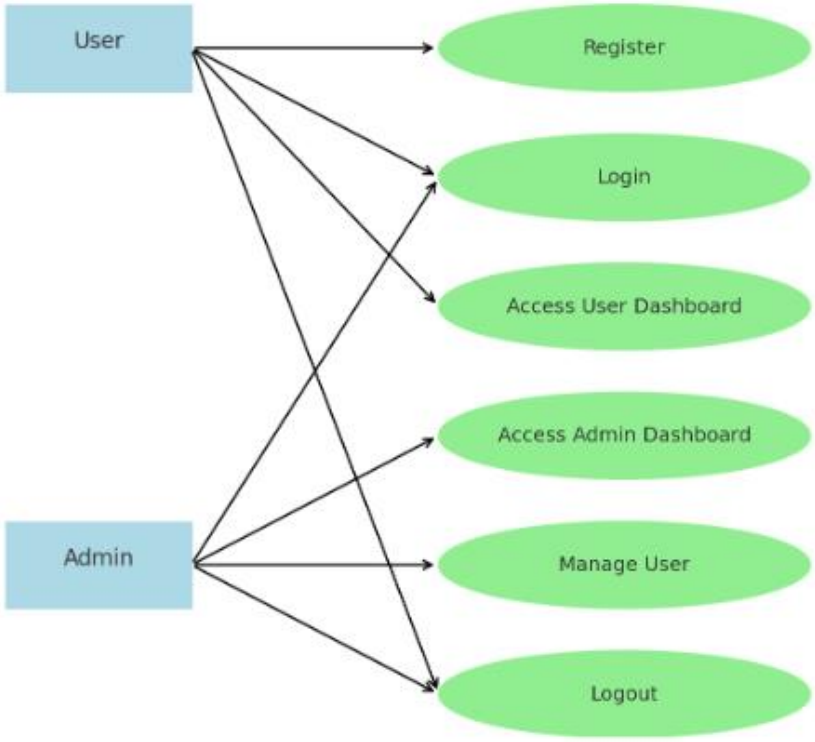
7. Security

- Handles password hashing (bcrypt) and input validation.
- Prevents SQL Injection, XSS, and brute-force attacks.
- Ensures secure data storage and communication.

SECURE LOGIN SYSTEM WITH USER ROLE MANAGEMENT

❖ DIAGRAM:

❖ **Use case Diagram For Admin:**



SECURE LOGIN SYSTEM WITH USER ROLE MANAGEMENT

Use Case Description:

1. Register

- **Actors:** User, Admin
- **Description:** Allows new users or admins to create an account by providing username, email, password, and role.
- **Pre-condition:** User is not already registered.
- **Post-condition:** User account is stored securely in the database with hashed password.
- **Exceptions:** Duplicate email or invalid inputs show error message.

2. Login

- **Actors:** User, Admin
- **Description:** Authenticates user/admin using email and password. CAPTCHA prevents bot attacks.
- **Pre-condition:** User/Admin must already be registered.
- **Post-condition:** User/Admin is redirected to their respective dashboard.
- **Exceptions:** Invalid credentials, failed CAPTCHA, or locked account after multiple attempts.

3. Access User Dashboard

- **Actor:** User
- **Description:** After login, the User can view their personal dashboard with limited features.
- **Pre-condition:** User must be logged in with valid credentials.
- **Post-condition:** User dashboard is displayed with user-specific content.
- **Exceptions:** Unauthorized access redirects to login.

4. Access Admin Dashboard

- **Actor:** Admin
- **Description:** Admin can view the admin dashboard with advanced functionalities.
- **Pre-condition:** Admin must be logged in with valid credentials.
- **Post-condition:** Admin dashboard is displayed with management tools.
- **Exceptions:** Unauthorized access redirects to login.

5. Manage Users

- **Actor:** Admin
- **Description:** Admin can view, update, or delete user accounts.
- **Pre-condition:** Admin must be logged in.
- **Post-condition:** User records updated in database.
- **Exceptions:** Invalid user ID or database failure.

6. Logout

- **Actors:** User, Admin
- **Description:** Ends the current session and returns to home/login page.
- **Pre-condition:** User/Admin must be logged in.
- **Post-condition:** Session/JWT is cleared and user is logged out securely.
- **Exceptions:** Session timeout auto-logs out the user.

SECURE LOGIN SYSTEM WITH USER ROLE MANAGEMENT

➢ All features Implemented Overview

This project implements a **secure authentication system** with user roles, CAPTCHA, and account lockout features. It ensures **data security, user management, and protection against brute force attacks**.

Features Implemented

1. User Registration with password hashing
2. Secure Login with JWT/session
3. Role-based access (Admin/User)
4. Admin Dashboard for managing users
5. CAPTCHA validation to prevent bots
6. Account lockout after multiple failed logins
7. Input validation to prevent SQL/NoSQL injection
8. Full frontend (HTML, CSS, JS) + backend (Flask + MongoDB)

Technologies Used

- **Flask** (Python Backend)
- **MongoDB** (Database)
- **Flask-Bcrypt** (Password Hashing)
- **Flask-Session** (Session Management)
- **PyJWT** (Authentication)
- **HTML, CSS, JavaScript** (Frontend)

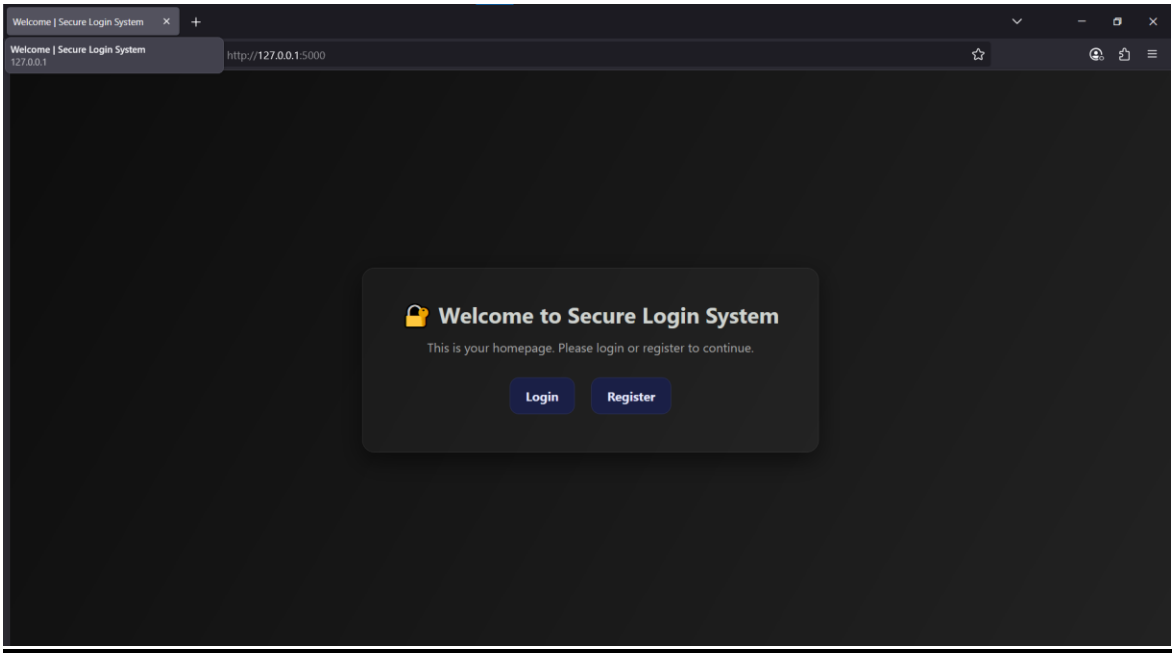
Challenges Faced & Solutions

- Duplicate registration → Email uniqueness check
- Session handling issue → Used flask-session
- Naive vs aware datetime error → Used timezone-aware datetime.now()
- Brute-force login attempts → CAPTCHA + account lockout

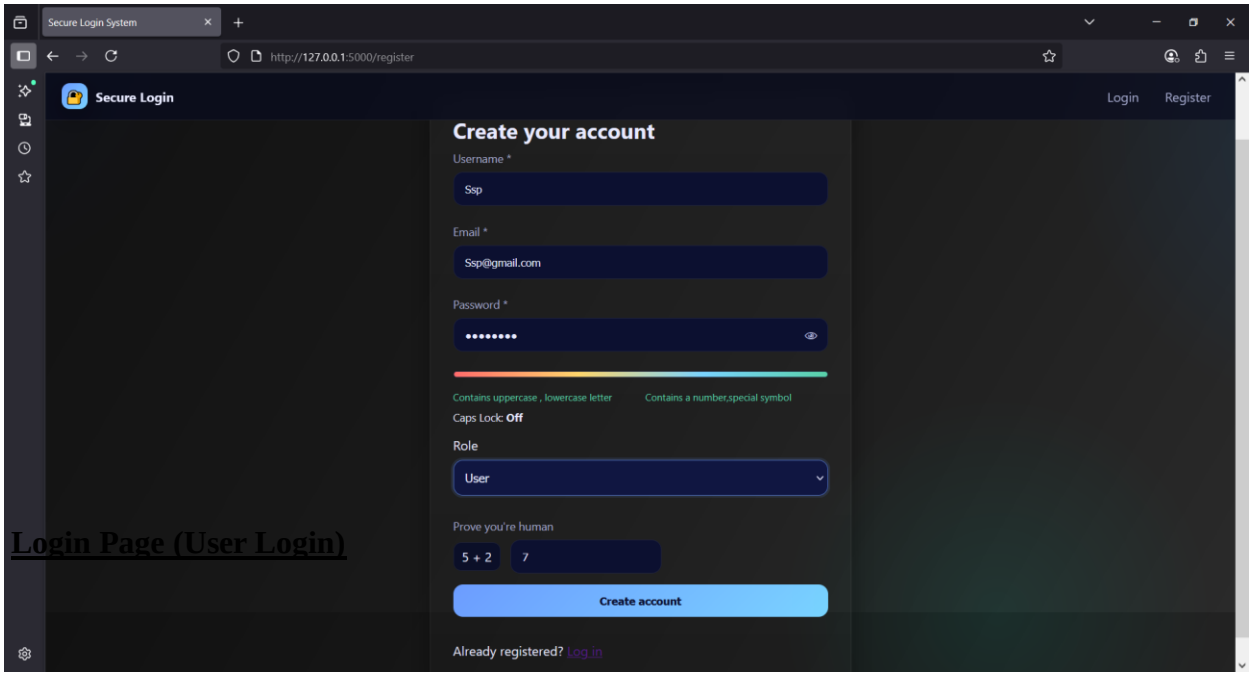
SECURE LOGIN SYSTEM WITH USER ROLE MANAGEMENT

- **Screenshots**

Home Page



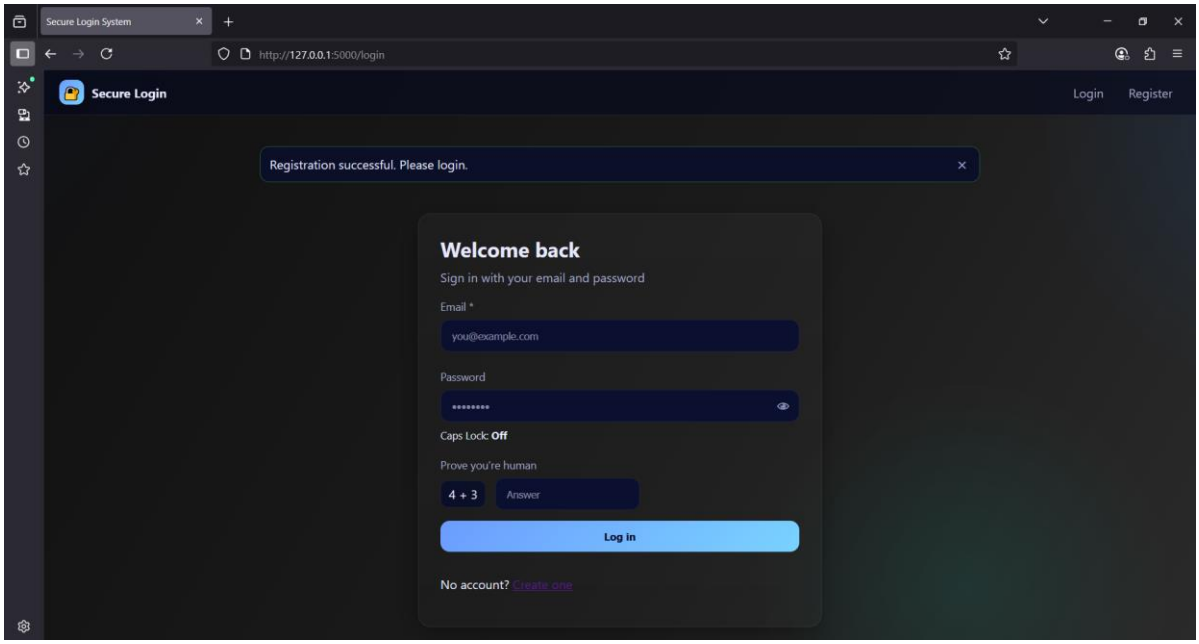
Register Page



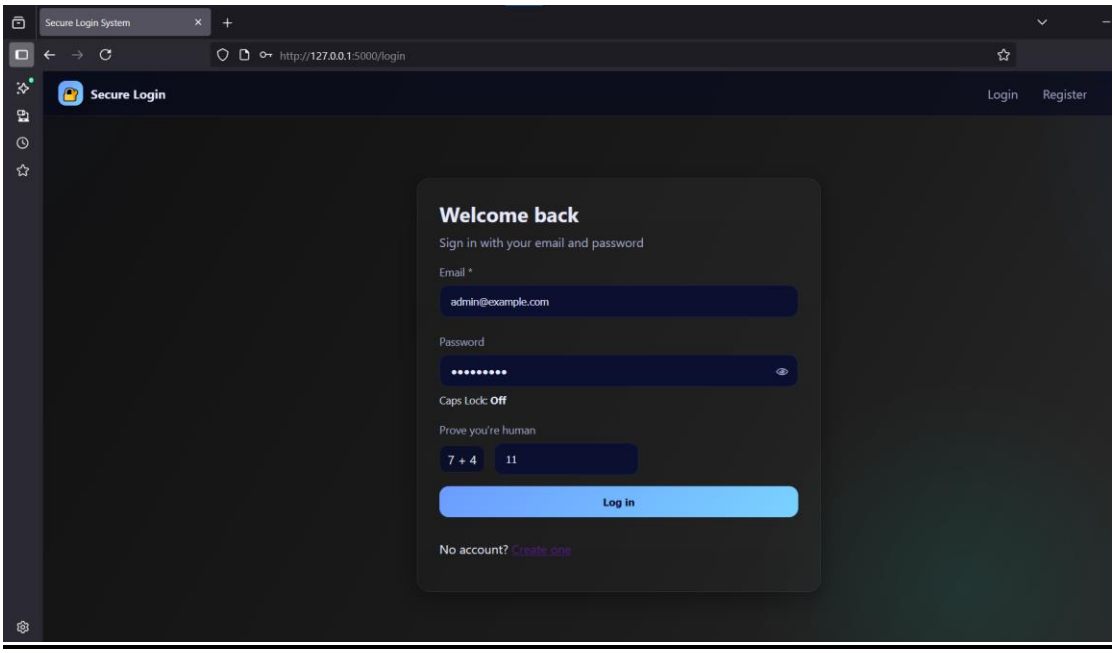
Login Page (User Login)

SECURE LOGIN SYSTEM WITH USER ROLE MANAGEMENT

Registration successfully

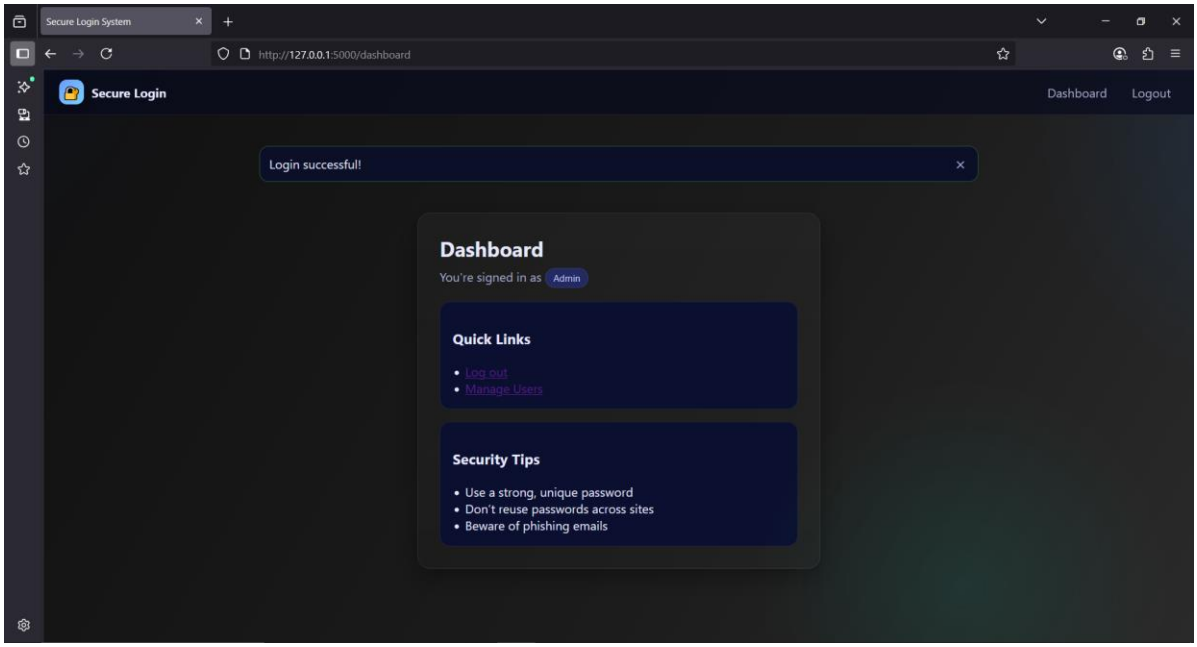


Login Page (Admin Login)

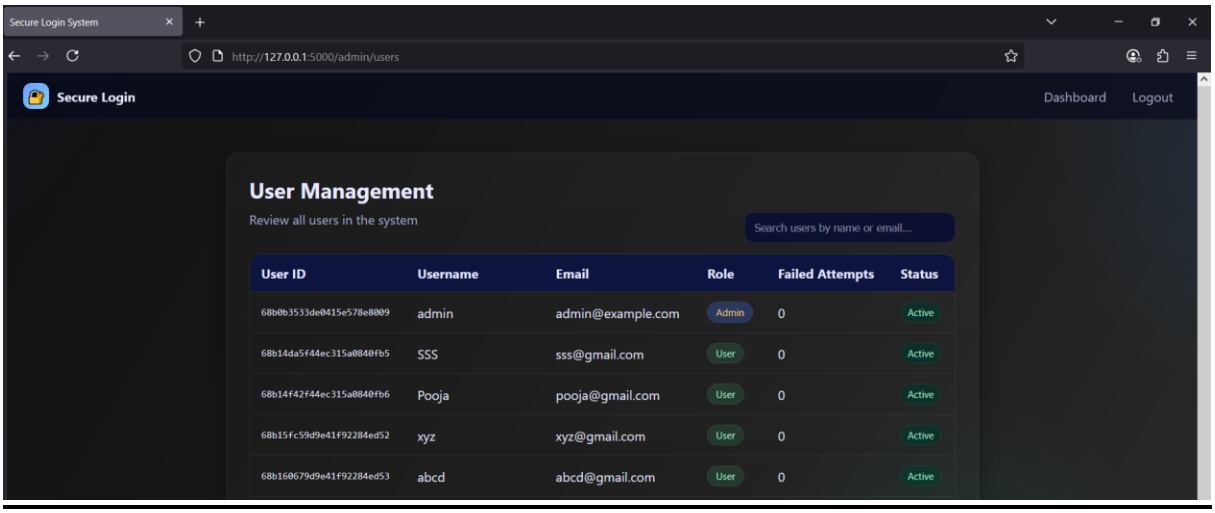


SECURE LOGIN SYSTEM WITH USER ROLE MANAGEMENT

Admin Dashboard

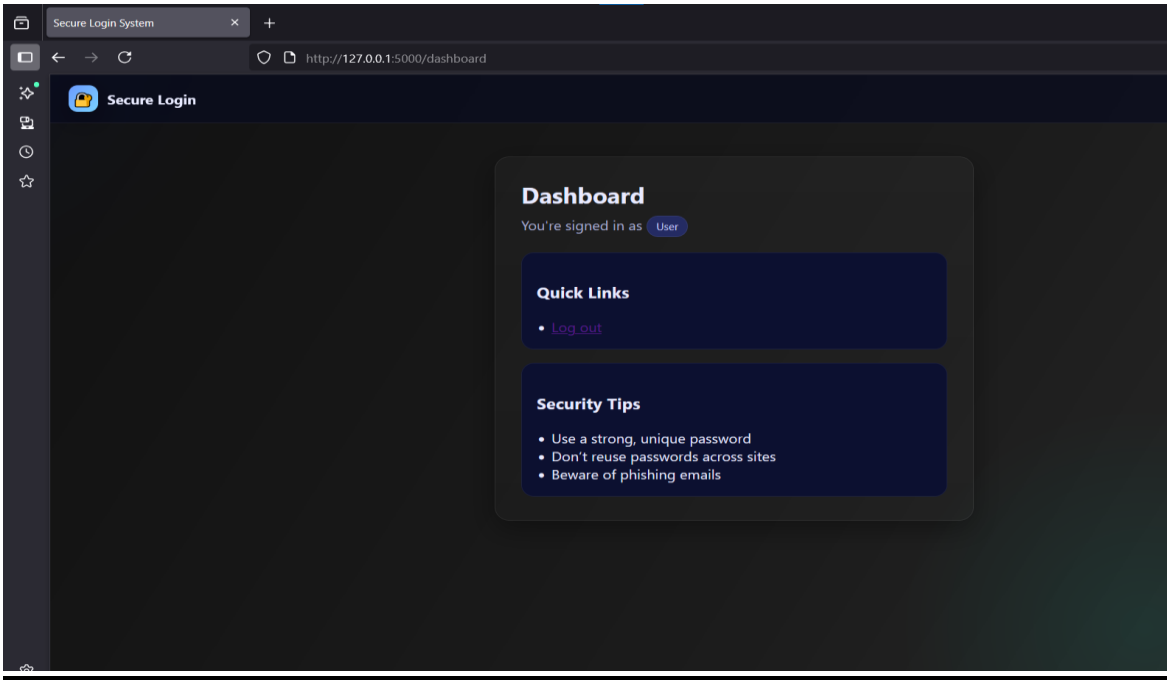


Admin dashboad in User management

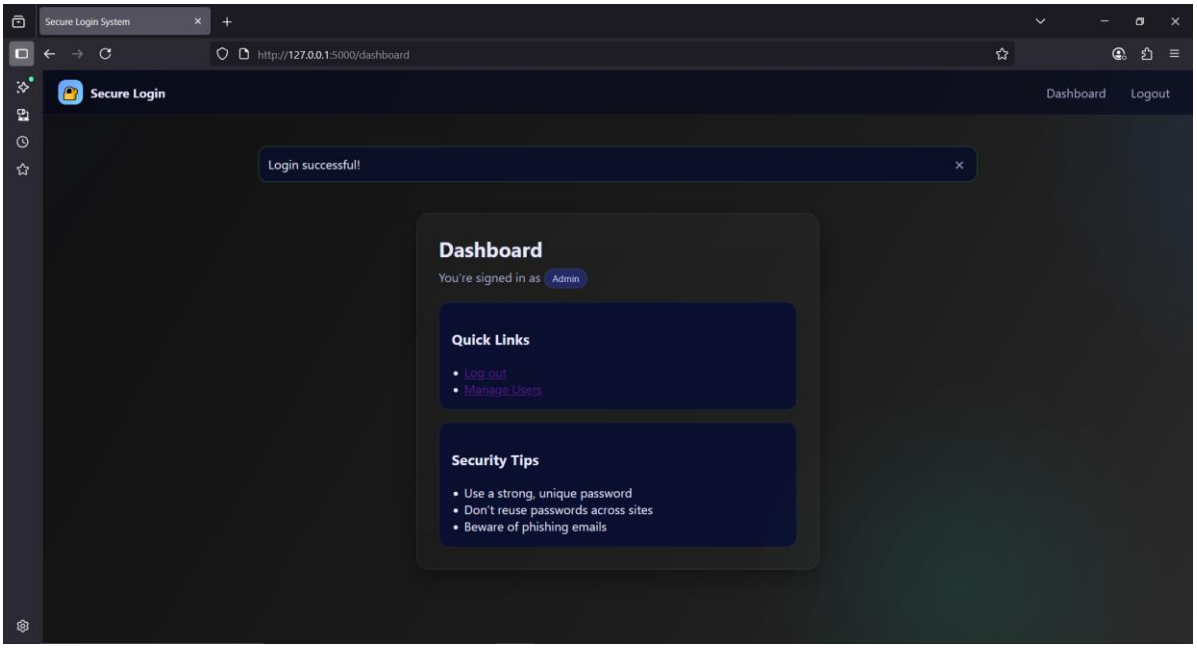


SECURE LOGIN SYSTEM WITH USER ROLE MANAGEMENT

Dashboard Page (User)



Dashboard Page (Admin)



SECURE LOGIN SYSTEM WITH USER ROLE MANAGEMENT

Logout