User Management System Documentation

Problem Statement:

To develop a secure User Management System with Secure Authentication and Role-Based Access

1. System Overview

The User Management System is a secure web-based platform designed to manage users efficiently. It supports authentication, role-based access control, and complete user profile management. Built with **Flask (backend)**, **ReactJS (frontend)**, and **MySQL (database)**.

Key Features:

- JWT-based authentication for secure sessions
- Admin and Normal user roles with role-based permission
- Full CRUD operations for users
- Enable/Disable user accounts
- Responsive web interface with ReactJS
- RESTful API integration
- Azure AD SSO (Optional for enterprise environments)
- Audit logs for admin actions
- Multi-device support
- Forgot Password with OTP verification via email

---

2. Technology Stack

- **Frontend:** ReactJS, Bootstrap, HTML5, CSS3
- **Backend:** Flask, Python 3.x, Flask-JWT-Extended, Flask-CORS
- **Database:** MySQL
- **Authentication:** JWT, Azure AD SSO (Optional), OTP Email

Forgot Password and OTP Verification The system supports password recovery using OTP (One-Time Password) sent to the registered email. This ensures security and prevents unauthorized access.

Flow Steps:

1 User clicks on 'Forgot Password' in the login screen.

2 User enters registered email address.
3 System generates a random OTP and stores it temporarily in the backend.
4 OTP is sent to the user's registered email.
5 User enters OTP in the verification form.
6 If OTP is valid, the user is redirected to reset password form.
7 User sets a new password, which is hashed and updated in the database.

3. System Requirements

Functional Requirements:

- Authentication with JWT tokens
- Admin can manage all users

- Normal users can update their profile
- Role-based access control

Non-Functional Requirements:

- Secure password hashing
- Responsive design
- API error handling
- Easy maintenance and scalability

4. System Architecture

Architecture Layers:

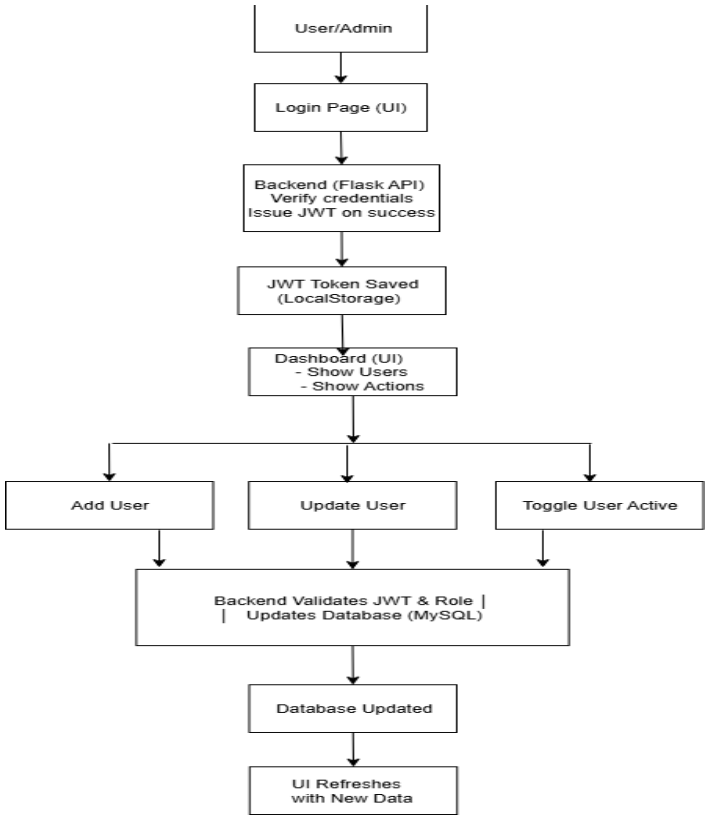
- **Frontend (ReactJS):** Navbar, Dashboard, Modals, Forms
- **Backend (Flask):** JWT auth, REST APIs, Role-based decorators
- **Database (MySQL):** User table with id, username, email, password_hash, role, is_active

5. Sequence Flows for Key Operations

- **Add User (Admin):**
  - Admin clicks "Add User" form.
  - Frontend sends POST `/admin/users` with JWT.
  - Backend validates admin role.
  - User is created in DB.
  - Response returned to frontend, table updated.
- **Update Profile (Normal User):**

    - User opens modal.
    - Frontend sends PUT `/users/me`.
    - Backend validates JWT, updates DB.
    - Response updates frontend view.
- **Enable/Disable User (Admin):**
    - Admin toggles switch.
    - PUT `/admin/users/<id>/toggle` request sent.
    - Backend toggles `is_active`.
    - Frontend updates table.

Flow Diagram:



6. API Endpoints

Endpoint	Method	Role	Description
/api/auth/login	POST	User/Admin	Authenticate user and return JWT
/api/auth/sso/login	GET	User/Admin	Redirect to Azure AD login
/api/auth/sso/callback	GET	User/Admin	Receive Azure AD token, create/update user, return JWT

Endpoint	Method	Role	Description
/api/auth/forgot-password	POST	User/Admin	Request OTP to email
/api/auth/verify-otp	POST	User/Admin	Verify OTP and allow reset
/api/auth/reset-password	POST	User/Admin	Reset password after OTP verification
/api/users	GET	User/Admin	List users (Admin) / Only self (User)
/api/users/me	PUT	User	Update own profile
/api/admin/users	POST	Admin	Add new user
/api/admin/users/	PUT	Admin	Update any user
/api/admin/users/	DELETE	Admin	Delete user
/api/admin/users/ /toggle	PUT	Admin	Enable/Disable user

---

7. Frontend Components

- **Navbar:** Shows logged-in user, logout button, admin menu
- **Dashboard:** User table with ID, Username, Email, Role, Active status
- **Modals:** Update modal for editing user info

-**User**: Add to new user

-**Login**: User authentication forms

-**Toggle** Switch For enabling/disabling user

---

8. Database Schema

Column	Type	Description
id	INT PK	Unique identifier

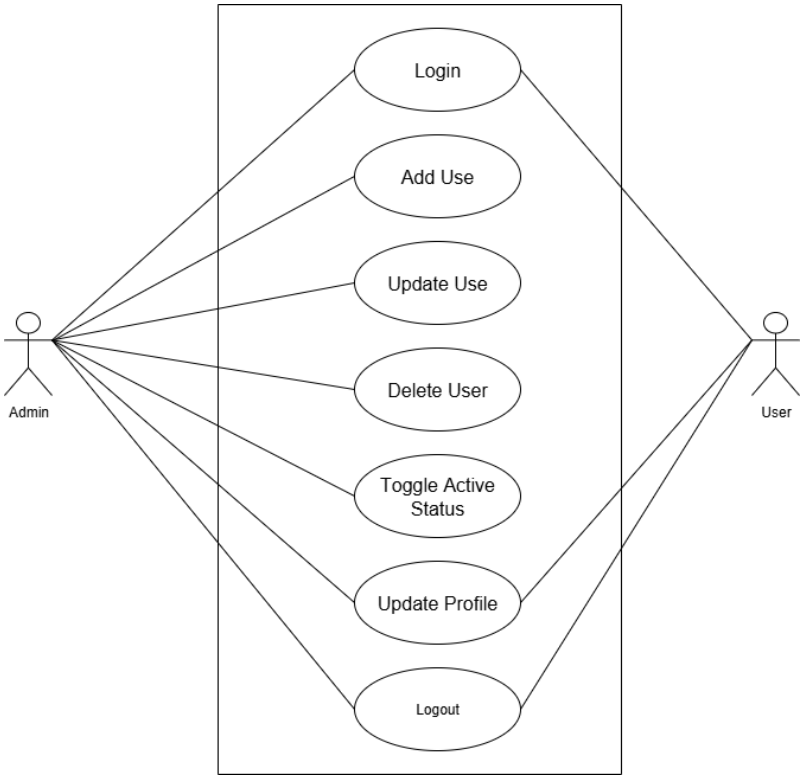
Column	Type	Description
username	VARCHAR(50)	Username
email	VARCHAR(100)	Email
password_hash	VARCHAR(255)	Hashed password
role	ENUM('user','admin')	Role of user
is_active	BOOLEAN	Active / Inactive
created_at	TIMESTAMP	Creation time
updated_at	TIMESTAMP	Last update

---

9. Security Considerations

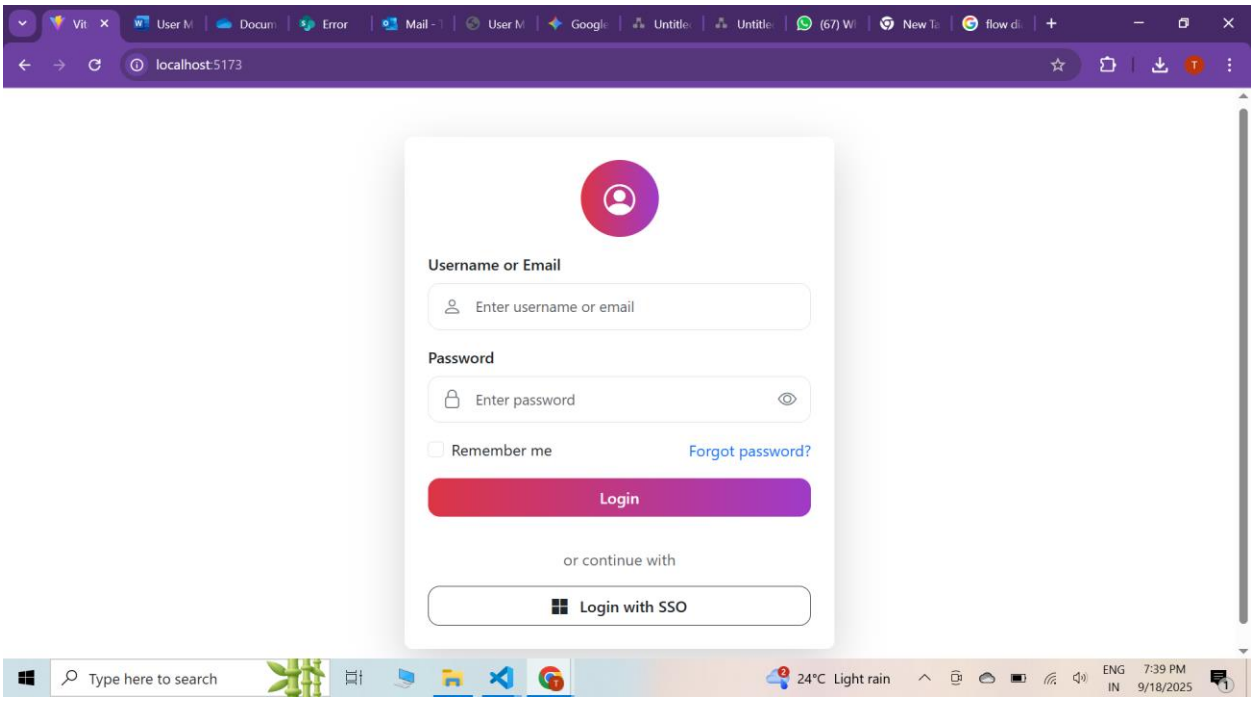
- **JWT Authentication:** Prevents unauthorized access.
- **Role-based Access Control:** Admin-only routes protected.
- **Password Hashing:** Secure hashing (e.g., bcrypt).
- **HTTPS Recommendation:** For production deployment.
- **SSO Integration:** Azure AD for company-wide login.
- **OTP Validation:** OTP is valid only for a limited time (e.g., 5 minutes).
- **OTP is stored** securely and invalidated after use.
- **Email sending** is configured with secure SMTP.
- **Strong password** hashing for new passwords.

10. Use Case Diagrams

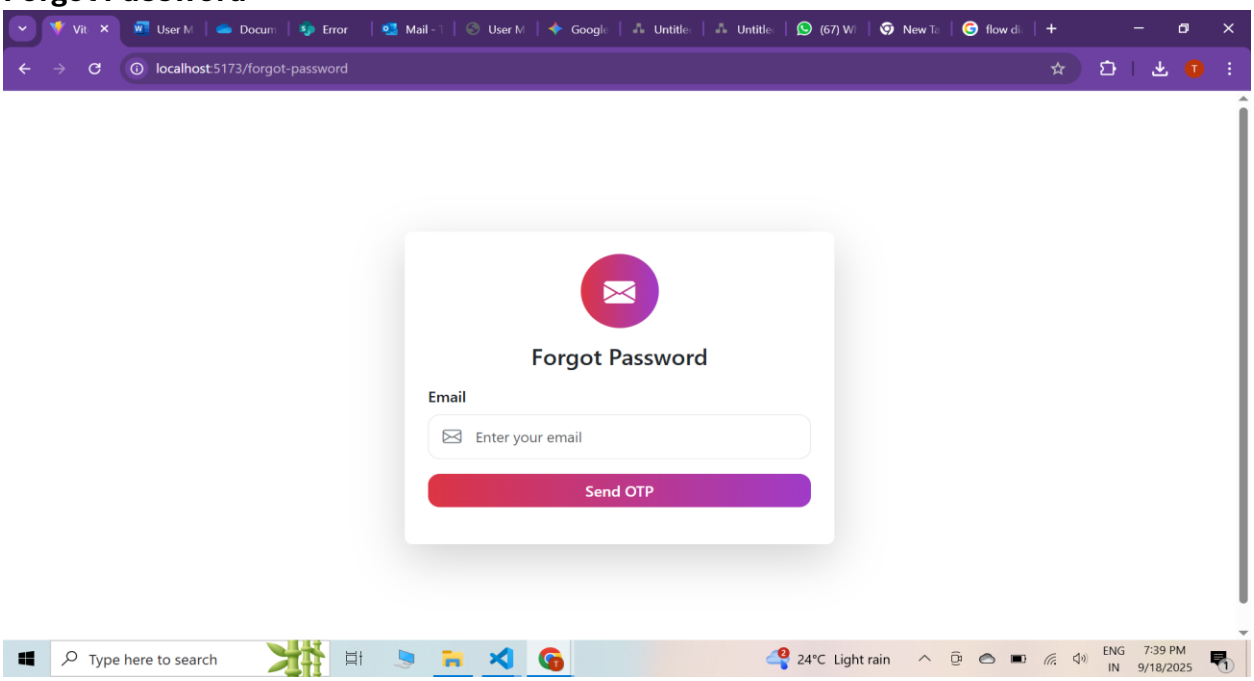


11. Screenshots

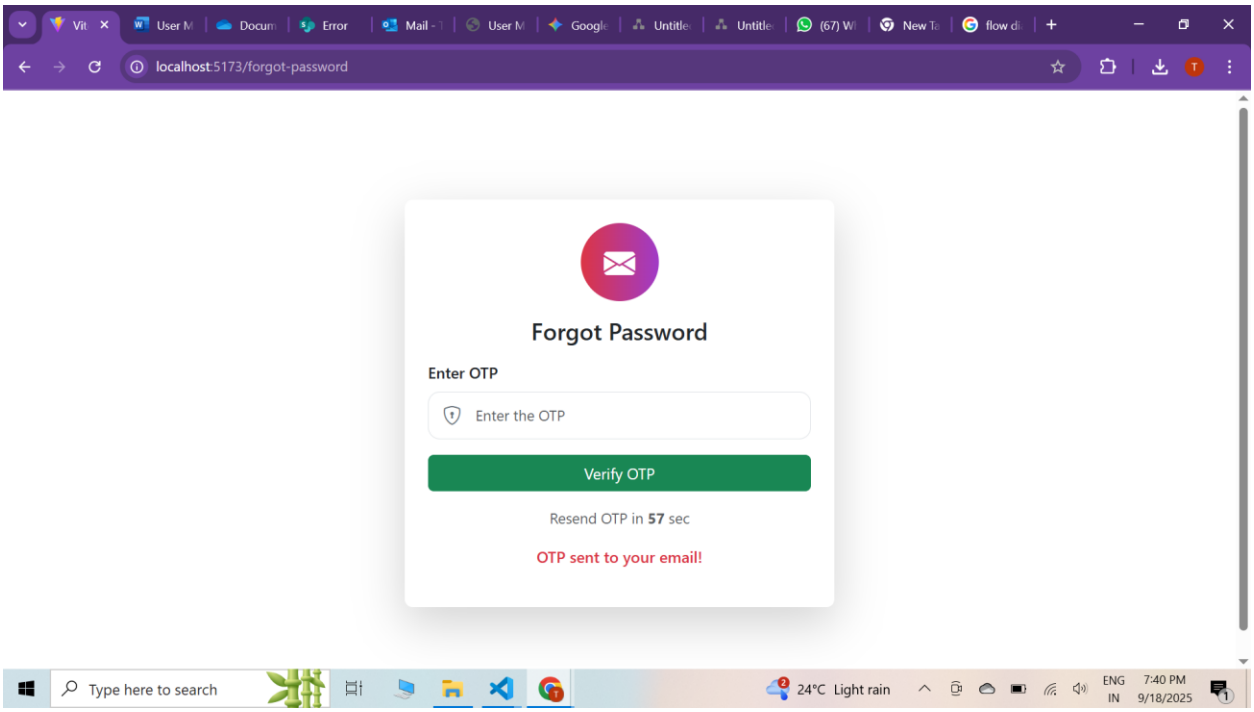
Login page



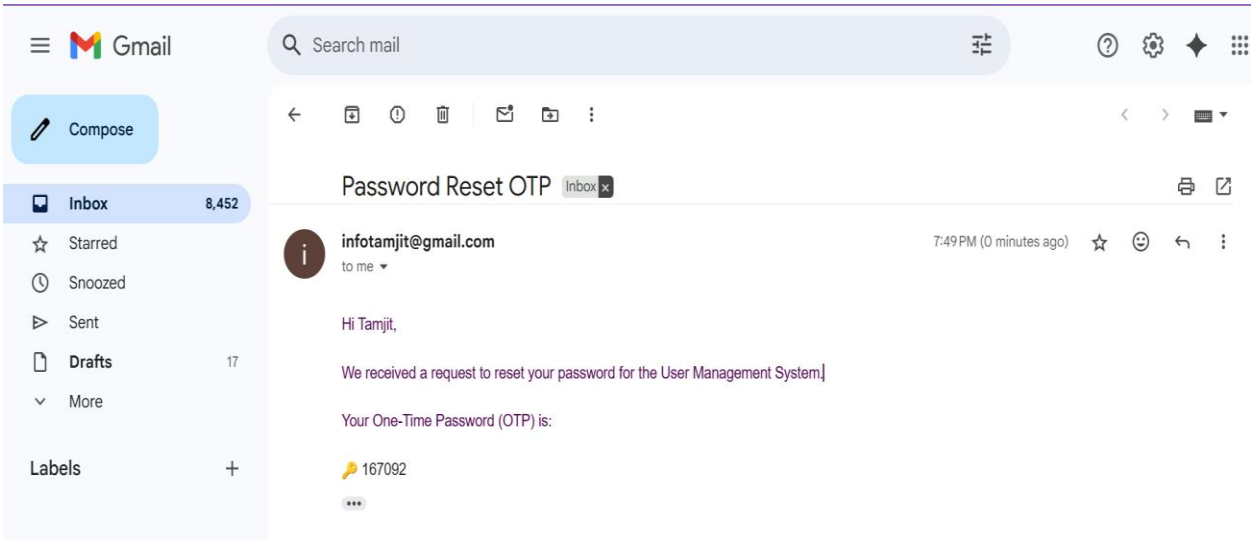
Forgot Password



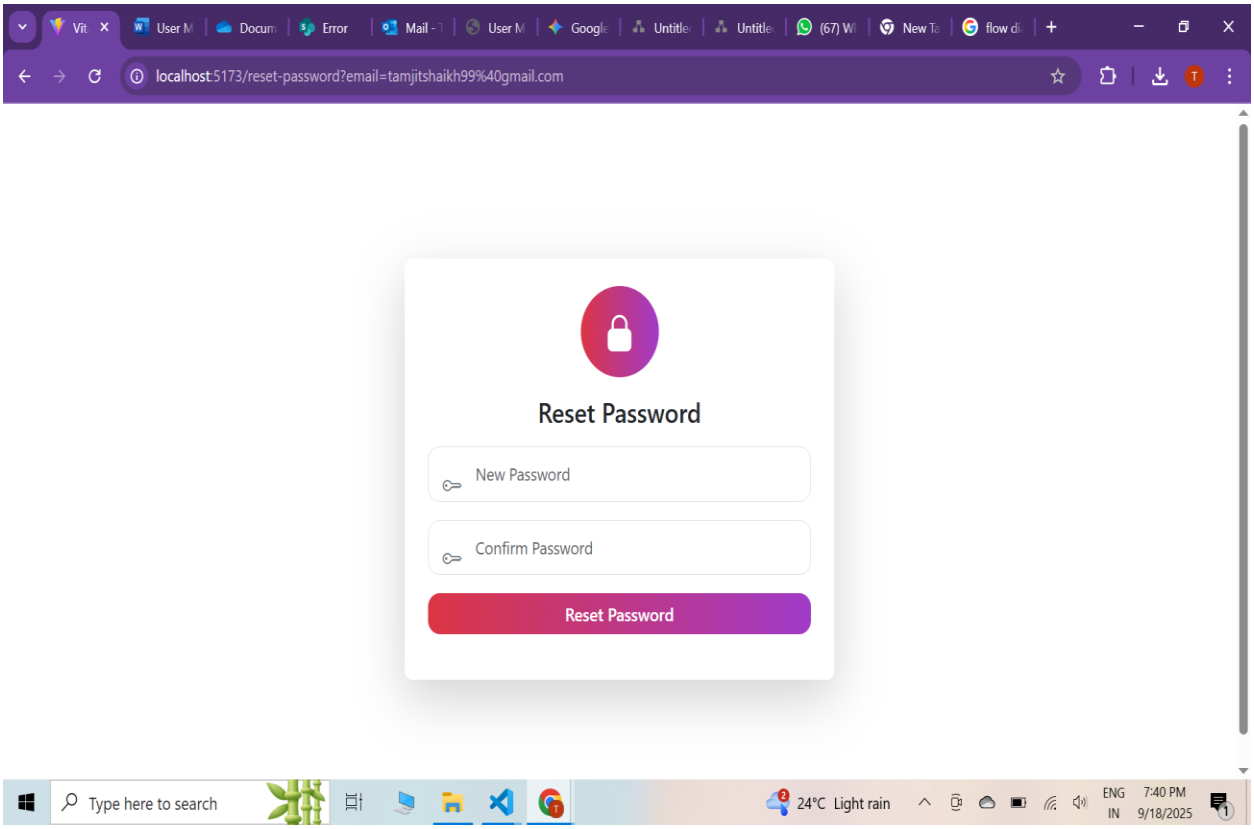
Enter OTP



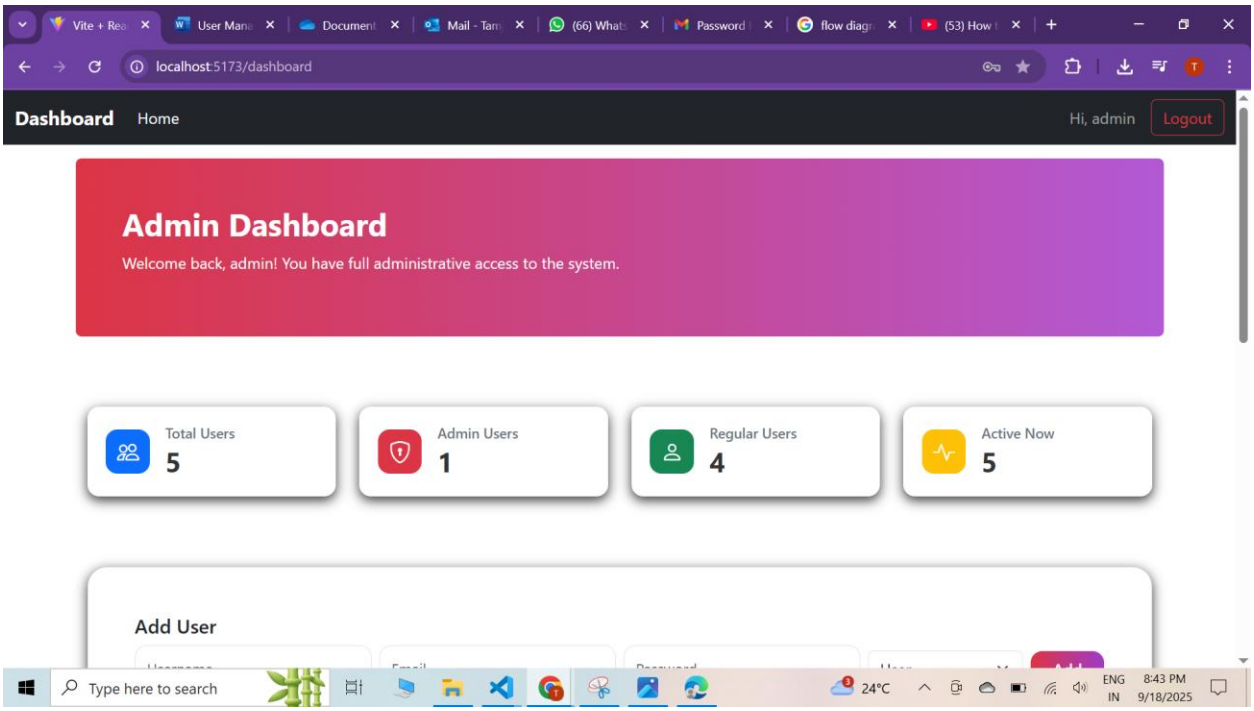
Password Reset OTP mail



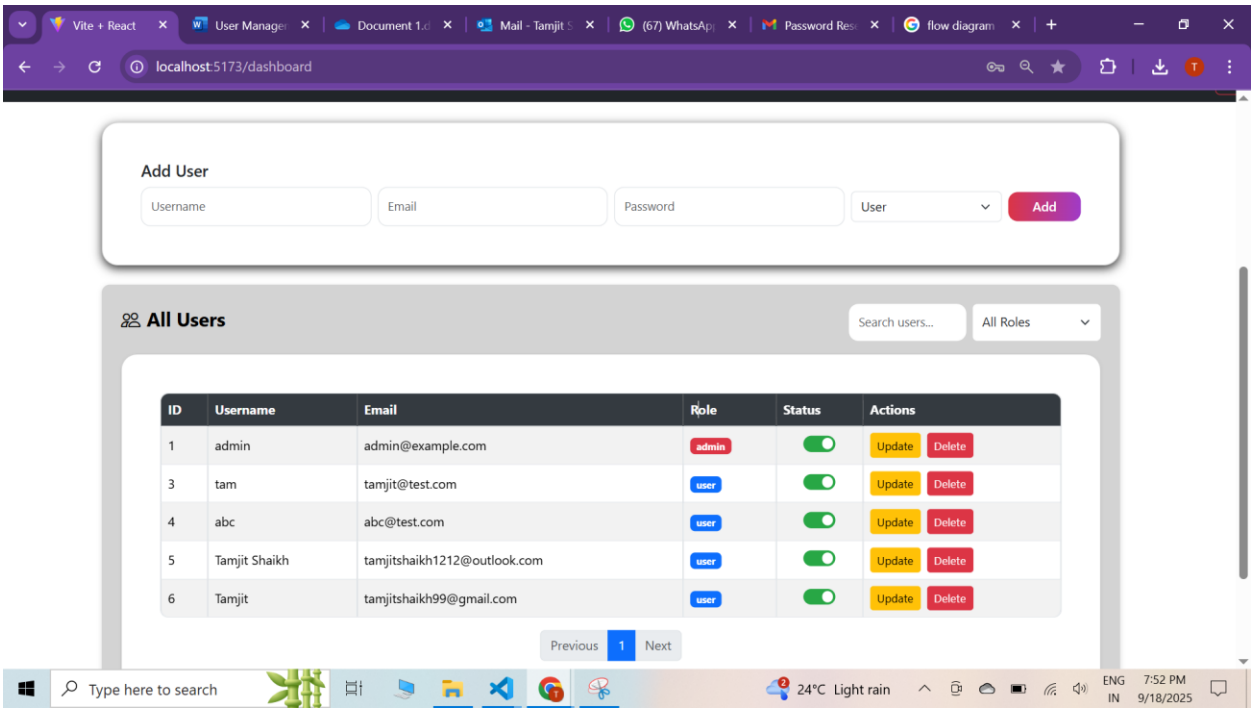
Reset Password



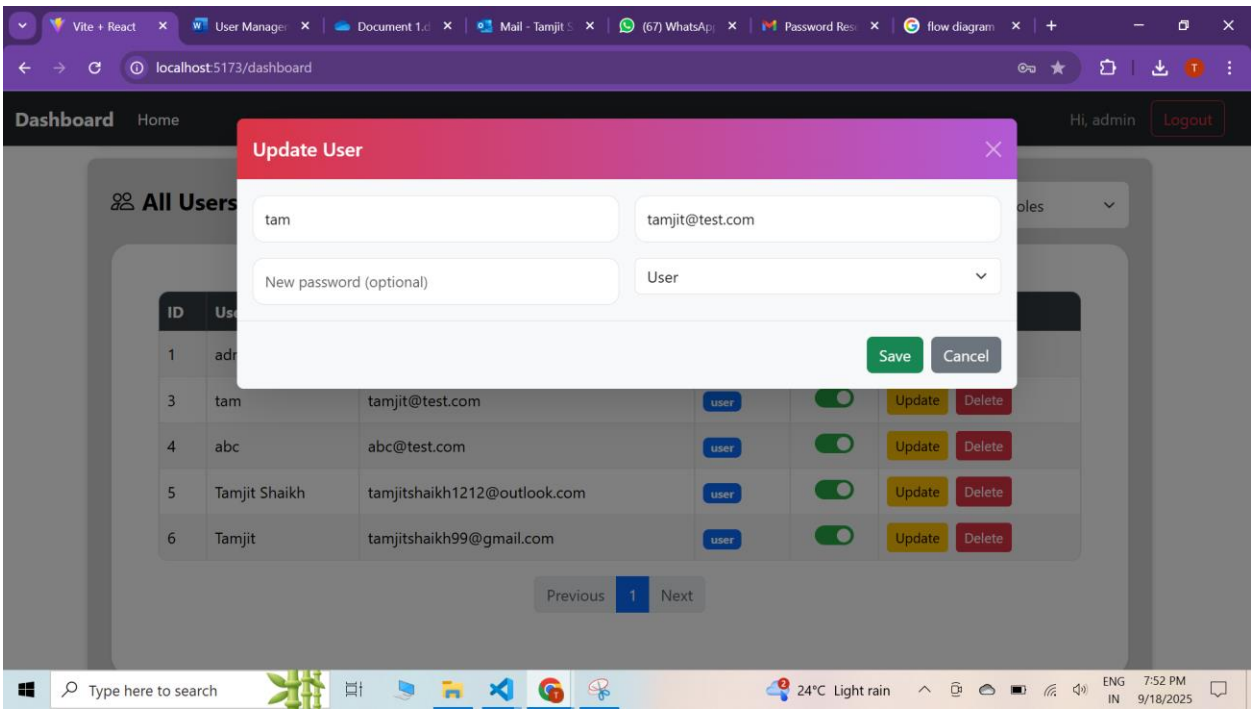
Admin Dashboard



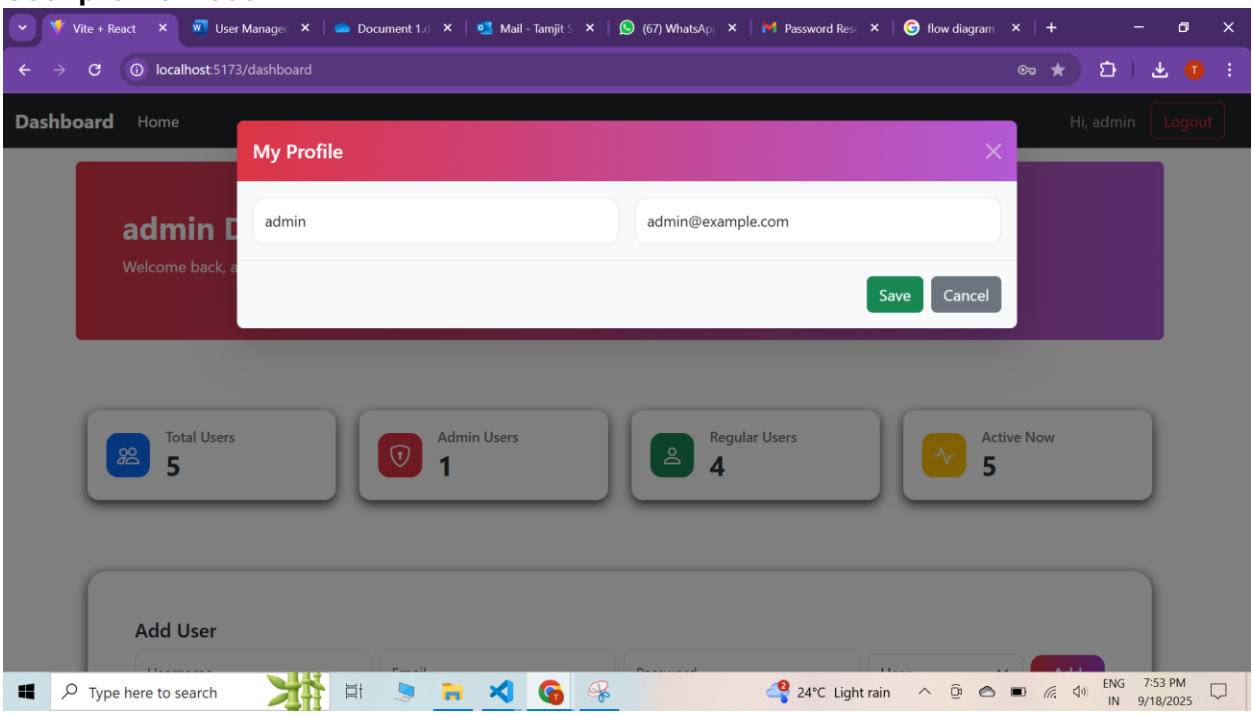
Add User and user Management



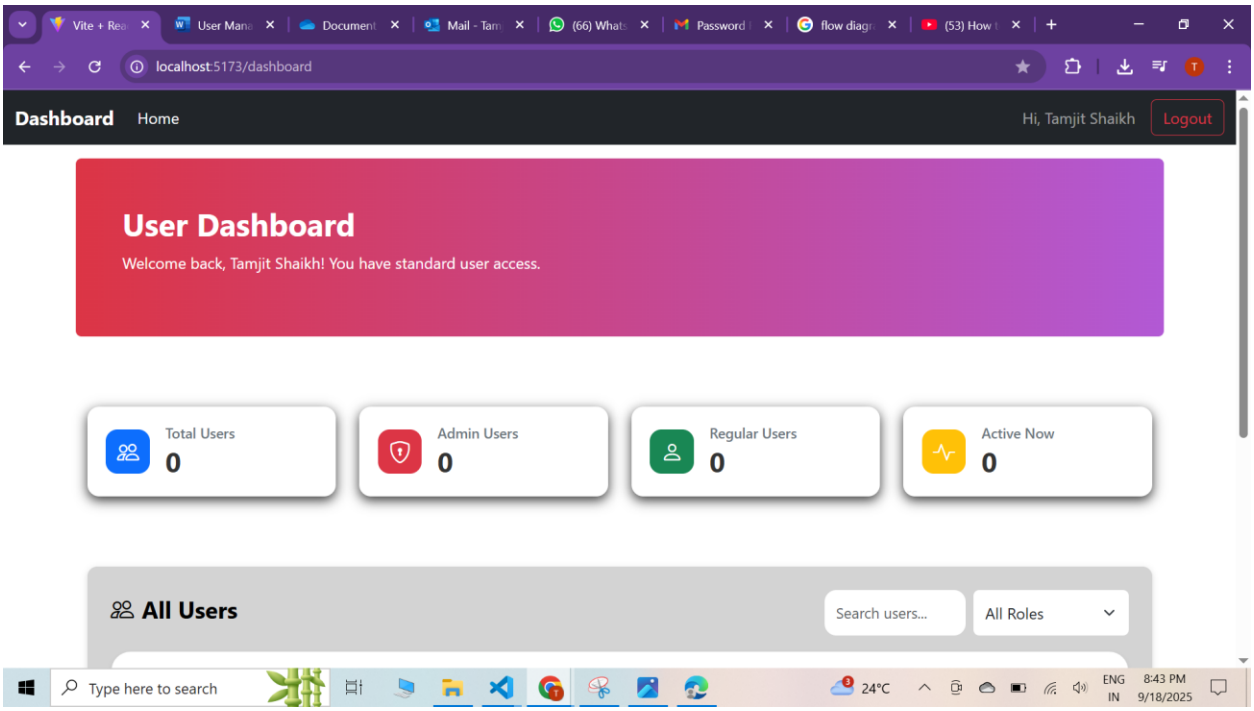
Update user Modal



User profile Modal



User Dashboard



User show only user list