Name: Walid khan

Reg\_No: 22MBCS274

Department: AI

**Code Explanation**

This code is a **Multi-Factor Authentication (MFA) System** using email and SMS OTPs. It consists of multiple components implemented in Python:

**1. User Registration and Login Logic**

**a. register\_user(email, password, phone)**

* Stores user details in a dictionary (user\_db).
* Passwords are hashed using bcrypt for security before storing them.
* **Workflow**:
  1. Hash the password using bcrypt.gensalt().
  2. Save the email, hashed password, and phone number to the user\_db.

**b. validate\_user(email, password)**

* Verifies if the user's email exists and checks if the entered password matches the stored hashed password.
* Uses bcrypt.checkpw() to compare the hashed password with the entered password.

**2. Email OTP Sending**

**a. send\_email\_otp(email)**

* Generates a random 6-digit OTP using random.randint().
* Uses MailHog (local SMTP server) to send the OTP email.
* **SMTP Configuration**:
  + Host: localhost (MailHog running locally).
  + Port: 1025 (default MailHog SMTP port).
* The email is sent via smtplib, and the OTP is returned for verification.

**3. SMS OTP Sending**

**a. send\_sms\_otp(phone)**

* Generates a random 6-digit OTP.
* Simulates sending an SMS by printing the OTP to the console.

**4. CLI-Based MFA System**

**a. main()**

* **Interactive menu** for registration and login.
* **Workflow**:
  1. **Registration**:
     + Prompts the user to enter their email, password, and phone number.
     + Calls register\_user() to store user details.
  2. **Login**:
     + Prompts the user to enter email and password.
     + Validates credentials with validate\_user().
     + If valid, performs:
       - **Email OTP Verification**:
         * Sends an OTP to the email and verifies user input.
       - **SMS OTP Verification**:
         * Sends an OTP to the phone and verifies user input.

**5. GUI-Based MFA System (Tkinter)**

**a. MFAApp Class**

* **Purpose**: Creates a GUI for the MFA system.
* Uses the Tkinter library to create graphical components.

**b. Key Methods:**

* create\_widgets(): Displays initial options (Register, Login, Exit).
* show\_register(): Shows the registration form.
* register\_user(): Registers the user using register\_user() from the backend.
* show\_login(): Shows the login form.
* login\_user(): Validates user credentials using validate\_user().
* otp\_verification(): Handles OTP generation and verification.
  + Calls send\_email\_otp() and send\_sms\_otp() for OTP generation.
* verify\_email\_otp(): Validates the entered email OTP.
* verify\_sms\_otp(): Validates the entered SMS OTP.
* clear\_frame(): Clears the current screen for displaying new components.

**c. Workflow:**

1. The GUI opens with buttons to register, log in, or exit.
2. Upon selecting:
   * **Register**:
     + Opens a form to enter email, password, and phone.
     + Calls register\_user() to save the details.
   * **Login**:
     + Prompts for email and password.
     + Calls validate\_user() and then verifies email and SMS OTPs.

**6. Integration with MailHog**

* **MailHog**:
  + A local email testing tool.
  + Captures sent emails for testing purposes (accessible at http://localhost:8025).
* SMTP server and port configuration (localhost:1025) ensures the system works locally.

**Key Features**

1. **Secure Authentication**:
   * Passwords are hashed before storage.
   * Two-factor authentication adds another security layer.
2. **CLI and GUI Interfaces**:
   * Users can interact with the system via the terminal or GUI.
3. **Simulated Email and SMS Delivery**:
   * OTPs are sent via MailHog for email and printed to the console for SMS.

**Example Use Case**

1. **Registration**:
   * Input: email: user@example.com, password: 1234, phone: 1234567890.
   * Output: User registered successfully (saved in user\_db).
2. **Login**:
   * Enter credentials and receive an OTP for email and SMS.
   * Enter the OTPs to complete login.
3. **MailHog**:
   * View the email OTP in MailHog (http://localhost:8025).

This modular design makes the system extendable to real-world email and SMS services in production.