Python - Capstone Project

Project Title: OTP Verification System

Problem Statement:

You are tasked with developing an OTP (One-Time Password) verification system in Python. The system should generate a 6-digit OTP and send it to the user's email address for verification. Upon receiving the OTP, the user should enter it into the system for validation. If the entered OTP matches the generated OTP, access should be granted; otherwise, access should be denied.

Project Requirements:

- Implement a function to generate a 6-digit OTP randomly.
- Develop a function to simulate sending the OTP to the user's email address.
- Create a function to prompt the user to enter the OTP received in their email.
- Implement a function to verify if the entered OTP matches the generated OTP.
- Ensure proper error handling and user-friendly prompts throughout the system.
- Allow the user to retry OTP entry in case of incorrect input.

Project Deliverables:

- Python script containing the implementation of the OTP verification system.
- Documentation explaining the functionality of each function, how to run the program, and any dependencies required.
- Test cases to ensure the system functions correctly under various scenarios, including correct and incorrect OTP entries.
- Optionally, you can create a simple GUI interface for the OTP verification system to enhance user experience.

Project Breakdown

1.Generate OTP:

A function to generate a 6-digit OTP using random digits.

2.Send OTP to Email:

A function to simulate sending the OTP to a user's email (In a real system, this would use an SMTP service).

3.User Input:

A function to prompt the user to input the OTP received in their email.

4.Verify OTP:

A function to verify if the entered OTP matches the generated OTP.

5.Error Handling:

Ensure proper error handling if an invalid OTP is entered, and allow retries.

Implementation :

(With Python Code)

```
import smtplib
import random
import time
import re
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

# Function to generate a 6-digit OTP

def generate_otp():

  return str(random.randint(100000, 999999))

# Function to send the OTP to the user's email

def send_email(recipient_email, otp):
```

```
try:
        # Email Configurations

        sender_email = "monakumari08@gmail.com"
        sender_password = "niueguwoshoruejc"  # It's
recommended to use an app password or environment variables


# Setting up the email message

        subject = "Your OTP Verification Code"
        body = f"Your One-Time-Password (OTP) is:
{otp}\nThis OTP is valid for 5 minutes."

        message = MIMEMultipart()
        message['From'] = sender_email
        message['To'] = recipient_email
        message['Subject'] = subject
        message.attach(MIMEText(body, 'plain'))

        # Sending email using SMTP server

        with smtplib.SMTP('smtp.gmail.com', 587) as server:
            server.starttls()  # Enable encryption
            server.login(sender_email, sender_password)
            server.send_message(message)
        print("OTP sent successfully to email.")

    except smtplib.SMTPAuthenticationError:
        print("Error: Authentication failed. Check your
email or app password.")
        exit(1)
    except Exception as e:
        print(f"Error: Failed to send email: {e}")
        exit(1)
```

```
#Function to validation OTP input and enforce exactly 6 digits

def validate_otp_input(new_value):

# Allow only digits and restrict to a maximum of 6 characters

  return new_value.isdigit() and len(new_value) <=6


# Function to validate Email

def is_valid_email(email):

  email_regex = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
  return re.match(email_regex, email) is not None



# Function to Verify OTP with retries and timeout
def otp_verification():

    while True:

        # Request the user's email address

        recipient_email = input("Enter your email address: ").strip()

        # Validate if email is provided
        if not recipient_email:
            print("Error: Email address cannot be empty.")
            continue
```

```
# Check if the email format is valid
        if not is_valid_email(recipient_email):
            print("Warning: Invalid email format. Please
enter a valid email address.")
            continue

        # Once valid, break the loop and proceed
        break

    # Generate and send OTP
    otp = generate_otp()
    send_email(recipient_email, otp)

    # Record the time the OTP was generated
    otp_time = time.time()
    retries = 3  # Number of retries allowed

    # OTP verification loop
    while retries > 0:
        # Ask the user to enter the OTP
        entered_otp = input("Enter the OTP sent to your
email: ").strip()

        # Validate OTP input (should be 6 digits)
        if not validate_otp_input(entered_otp):
            print("Error: Invalid OTP format. Please enter
a 6-digit OTP.")
            continue

        # Check if OTP is expired (5 minutes timeout)
        if time.time() - otp_time > 300:  # 5 minutes
timeout
            print("Error: OTP has expired. Please request
a new OTP.")
            return
```

```
# Verify OTP entered by the user
        if entered_otp == otp:
            print("Success: OTP verified successfully! Access
granted.")
            return
        else:
            retries -= 1
            if retries > 0:
                print(f"Error: Incorrect OTP. You have
{retries} retries left.")
            else:
                print("Error: Verification failed. Access
denied!")
                return

# Run the OTP Verification Process
otp_verification()
```

Output :

* If we entered Correct OTP(one time password)

```
Enter your email address: gunnalaranjithkumar31@gmail.com
OTP sent successfully to email.
Enter the OTP sent to your email: 516823
Success: OTP verified successfully! Access granted.
```

Output :

- If we entered wrong OTP (one time password)

```
Enter your email address: gunnalaranjithkumar31@gmail.com
OTP sent successfully to email.
Enter the OTP sent to your email: 879765
Error: Incorrect OTP. You have 2 retries left.
Enter the OTP sent to your email: 909878
Error: Incorrect OTP. You have 1 retries left.
Enter the OTP sent to your email: 778867
Error: Verification failed. Access denied!
```

Python Code Explanation :

Key Components of the Code:

1.Imports and Libraries:

- ✓ smtplib: Used for sending the email over SMTP.*(Simple Mail Transfer Protocol)*
- ✓ random: Used for generating the 6-digit OTP.
- ✓ time: Used to track the OTP generation time and implement timeout logic.
- ✓ re: Used for validating the format of the email input.
- ✓ MIMEText and MIMEMultipart: Used to create and send a well-structured email with a plain text body.

2.Function: generate_otp()

- ✓ Purpose: Generates a random 6-digit OTP.
- ✓ Implementation: Uses random.randint(100000, 999999) to generate the OTP and converts it to a string.

3.Function: send_email(recipient_email, otp)

- ✓ Purpose: Sends an email with the generated OTP to the provided email address.
- ✓ Implementation:
- ✓ Configures the email subject and body.
- ✓ Uses the Gmail SMTP server (smtp.gmail.com) to send the email securely using starttls().
- ✓ Logs in with the sender's email and password and sends the email.

4.Function: validate_otp_input(new_value)

- ✓ Purpose: Validates the OTP entered by the user.
- ✓ Implementation: Ensures that the OTP is a 6-digit number. This prevents invalid or incorrect inputs like non-numeric or overly long OTPs.

5.Function: is_valid_email(email)

- ✓ Purpose: Validates if the entered email address is in the correct format.
- ✓ Implementation: Uses a regular expression (regex) to ensure the email is in the format username@domain.com.

6.Function: otp_verification()

- ✓ Purpose: Coordinates the entire OTP process (email validation, OTP generation, and OTP verification).

- ✓ **Implementation:**

- ✓ Prompts the user to input their email and validates it using is_valid_email().
- ✓ If valid, it generates the OTP and sends it using send_email().
- ✓ The time of OTP generation is recorded to track the expiration time (5 minutes).
- ✓ The user is asked to input the OTP, which is validated for format (6 digits) and expiration (5-minute limit).
- ✓ Allows up to 3 retries if the OTP is incorrect, after which access is denied.

7.Error Handling and Validation:

- ✓ If the email address is invalid or empty, the script prompts the user to enter a valid email.
- ✓ If the OTP is incorrect or expired, it provides feedback and allows retries (up to 3 times).
- ✓ The script ensures the OTP input is strictly 6 digits.

8.Run the Verification:

- ✓ The otp_verification() function is executed at the end, which runs the OTP generation, email sending, and verification process.

Security Considerations:

- ✓ App Passwords: In real-world scenarios, using an app password (for Gmail or similar services) instead of the regular email password is recommended to enhance security.
- ✓ Timeout: The OTP expires after 5 minutes (300 seconds) to ensure it can't be used after a long delay.

Summary :

This project implements an OTP verification system with the following functionalities:

- ✓ It generates a random 6-digit OTP and sends it to the user's email.
- ✓ It validates the email format and OTP input.
- ✓ It has retry logic for incorrect OTP entries and handles OTP expiration.
- ✓ It incorporates basic error handling to ensure a smooth user experience.

- ➢ *By using Python's smtplib, random, time, and re modules, this project demonstrates practical implementation of email sending, input validation, and timed processes, commonly used in authentication systems.*

Thank You Everyone

Gunnala Ranjith Kumar
DS47A - S9709