**Project- OTP Verification System using python**

**Overview:**

This project implements an OTP (One-Time Password) Verification System using Python. The system performs the following:

* Generates a secure 6-digit OTP
* Sends it to a user's email via Gmail SMTP
* Prompts the user to enter the OTP
* Verifies the entered OTP with up to 3 retry attempts
* Validates user inputs and handles exceptions for a secure and user-friendly experience

**Dependency:**

The system uses the following Python standard libraries:

* smtplib – for sending email via SMTP
* email.message – for creating email content
* random – for OTP generation
* re – for validating email format
* getpass – for secure password input

**Functionality of the each function:**

|  |  |
| --- | --- |
| **Function** | **Description** |
| generate\_otp() | Generates a random 6-digit OTP |
| is\_valid\_email(email) | Validates if the email is correctly formatted using regex |
| is\_valid\_password(password) | Checks if the App Password is exactly 16 characters |
| send\_otp\_real\_email(otp, recipient\_email, sender\_email, sender\_password) | Sends OTP via Gmail SMTP; returns True if successful |
| get\_user\_otp() | Prompts user to input received OTP |
| verify\_otp(original\_otp, max\_attempts=3) | Verifies OTP with up to 3 allowed attempts |
| main() | Controls the overall logic of the OTP verification system |

**How to Run the Program:**

1. **Set up Gmail App Password:**

* Enable **2-step verification** on your google account
* Visit : <https://myaccount.google.com/apppasswords>
* Generate an App Password for **Mail**.
* Copy the 16-charater password

1. **Run the Python Script**
2. **Provide required details:**

* Recipient email (any valid email)
* Your Gmail address (sender)
* The 16- digit App Password (not your regular password)

1. **OTP Verification:**

* Check your mail inbox for the OTP
* Enter it when prompted in the terminal

**Test Cases:**

**Test case 1: Correct OTP entry**

* **Input:** Correct recipient email, valid email & App password, correct OTP
* **Expected Results:** OTP sent and verified 🡪 **Access Granted**

**Test case 2: Incorrect OTP, then correct OTP**

* **Input:** Enter incorrect OTP once, then correct OTP
* **Expected Results:** Incorrect shown 🡪 Prompt again 🡪 success on second try

**Test case 3: All incorrect OTP entries**

* **Input:** Enter wrong OTP 3 times
* **Expected Results:** Access denied after 3 attempts

**Test case 4: Invalid Email format**

* **Input:** Email like abc@xyz or name
* **Expected Results:** Shown “Invalid email format” and asked to re-enter

**Test case 5: Invalid App password**

* **Input:** Password not 16 characters, or contain symbols
* **Expected Results:** Shown “Invalid App Password” and asked again

**Test case 6: Gmail Login Fails**

* **Input:** Wrong sender email or app password
* **Expected Results:** Email sending fails 🡪 shown error🡪 System exits

**Python Script:**

import smtplib

from email.message import EmailMessage

import random

import re

import time

from getpass import getpass

# Function to generate a 6-digit OTP

def generate\_otp():

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

# Validate email format

def is\_valid\_email(email):

    pattern = r'^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$'

    return re.match(pattern, email)

# Validate App Password (should be 16 characters, alphanumeric only)

def is\_valid\_password(password):

    return len(password) == 16 and password.isalnum()

# Send OTP via Gmail SMTP

def send\_otp\_real\_email(otp, recipient\_email, sender\_email, sender\_password):

    try:

        msg = EmailMessage()

        msg.set\_content(f'Your OTP is: {otp}')

        msg['Subject'] = 'Your OTP Verification Code'

        msg['From'] = sender\_email

        msg['To'] = recipient\_email

        server = smtplib.SMTP('smtp.gmail.com', 587)

        server.starttls()

        server.login(sender\_email, sender\_password)

        server.send\_message(msg)

        server.quit()

        print("OTP sent successfully.")

        return True

    except Exception as e:

        print("Failed to send OTP. Please check your sender email or App Password.")

        print(f"Error: {e}")

        return False

# Prompt user to enter OTP

def get\_user\_otp():

    return input("Enter the OTP you received in your email: ").strip()

# Verify OTP with retry logic

def verify\_otp(original\_otp, max\_attempts=3):

    attempts = 0

    while attempts < max\_attempts:

        user\_otp = get\_user\_otp()

        if user\_otp == original\_otp:

            print("Access Granted. OTP Verified Successfully.")

            return True

        else:

            attempts += 1

            print(f"Incorrect OTP. Attempts left: {max\_attempts - attempts}")

    print("Access Denied. You have exceeded the number of attempts.")

    return False

# Main driver function

def main():

    print("Welcome to the OTP Verification System")

    # Get and validate recipient email

    while True:

        recipient\_email = input("Enter the recipient's email address: ").strip()

        if is\_valid\_email(recipient\_email):

            break

        print("Invalid email format. Please enter a valid email (e.g., yourname@example.com).")

    # Get sender email

    sender\_email = input("Enter your Gmail address (sender): ").strip()

    # Get and validate App Password securely

    while True:

        sender\_password = getpass("Enter your 16-character App Password: ").strip()

        if is\_valid\_password(sender\_password):

            break

        print("Invalid App Password. Must be exactly 16 alphanumeric characters.")

    # Generate OTP

    otp = generate\_otp()

    # Send OTP

    if not send\_otp\_real\_email(otp, recipient\_email, sender\_email, sender\_password):

        print("Unable to proceed without sending OTP. Exiting...")

        return

    # Verify OTP

    verify\_otp(otp)

# Run the program

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**Final Test Case Summary:**

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Description** | **Result Status** |
| Test Case 1 | Correct OTP entered | Access Granted. OTP Verified Successfully |
| Test Case 2 | Incorrect OTP, then Correct OTP | \* Incorrect OTP. Attempts left: 2  \*Access Granted. OTP Verified Successfully |
| Test Case 3 | All Incorrect OTP attempts | \* Incorrect OTP. Attempts left: 2  \* Incorrect OTP. Attempts left: 1  \* Incorrect OTP. Attempts left: 0  \* Access Denied. You have exceeded the number of attempts. |
| Test Case 4 | Invalid recipient email format | \*Invalid email format. Please enter a valid email (e.g., [yourname@example.com](mailto:yourname@example.com)).  \* Enter the recipient's email address: |
| Test Case 5 | Invalid App Password format | \*Invalid App Password. Must be exactly 16 alphanumeric characters.  \*Enter your 16-character App Password: |
| Test Case 6 | Gmail login failed | \* Failed to send OTP. Please check your sender email or App Password.  \* Unable to proceed without sending OTP. Exiting... |

**Conclusion:**

The OTP Verification System is

* Robust
* Secure
* Handles various edge cases effectively.

All functionalities work as expected, and proper error handling ensures a user-friendly experience.