OTP VERIFICATION SYSTEM

- ***Arif Shaik***
- ***S10117***

Overview:

The OTP Verification System is a secure application that enhances user authentication using One-Time Passwords (OTPs). By generating a unique 6-digit OTP sent to the user's email, this system ensures secure identity verification. It is particularly useful for logging into accounts, confirming transactions, and account recovery, effectively reducing the risk of unauthorized access.

Features:

The OTP Verification System include several key features:

- **Secure OTP Generation:** Generates a random 6-digit OTP for each session.
- **Email Delivery:** Sends the OTP directly to the user's email via SMTP.
- **User-Friendly GUI:** Offers an intuitive interface built with Tkinter.
- **Multiple Verification Attempts:** Limits users to a set number of attempts to enhance Security.
- **Error Handling:** Provides informative messages for failed OTP sending or incorrect entries.
- **Customizable Email Credentials:** Allows easy configuration of email settings.
- **Cross-Platform Compatibility:** Runs on Windows, macOS, and Linux.

These features create a robust solution for secure user identity verification.

Dependencies:

To run this application, the following Python libraries are required:

- **random:** Used for generating random numbers for the OTP.
- **smtplib**: Facilitates sending emails using the **Simple Mail Transfer Protocol (SMTP).**
- **email.mime:** Used to construct email messages; included in the standard library.
- **tkinter**: The standard GUI toolkit for Python, used to create the graphical interface for user interaction.
- **tkinter.font**: For customizing font styles in the GUI.

Code Breakdown: Imports

```
import random
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
import tkinter as tk
from tkinter import messagebox
import tkinter.font as tkfont
```

- **Explanation:** This section imports the necessary libraries for the application. **random** is used to generate the OTP, **smtplib** handles email sending, and **tkinter** is used for creating the GUI components.

Generating OTP

```
# Generating 6 digit OTP
def generate_otp() :
    return f"{random.randint(100000, 999999)}"
```

- **Description:** This function generates a random 6-digit OTP. It uses **random.randint()** to create a number between 100000 and 999999, ensuring that the OTP is always 6 digits long.
- **Returns:** A string representing the generated OTP.

Sending OTP

- **Parameters:**
  - **receiver_email: The** email address to which the OTP will be sent.
  - **otp**: The OTP to be sent.

- **Description:** This function handles the process of sending the OTP via email. It sets up the SMTP server, logs in with the sender's credentials, and constructs the email message.

- **Returns:**
  - **True** if the OTP is sent successfully.
  - **False** if there is an error in the sending process.

- **Error Handling:** The function includes a try-except block to catch exceptions during the email sending process, printing an error message for debugging.

```python
# Send otp to receviers email
def send_otp(receiver_email, otp) :
    try :
        sender_email = "Enter from Email you need to send OTP"
        sender_password = "App Password"

        smtp_server = "smtp.gmail.com"
        smtp_port = 587

        subject = "Your OTP for verification"
        body = f"Your OTP is: {otp}"

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

        with smtplib.SMTP(smtp_server, smtp_port) as server :
            server.starttls()
            server.login(sender_email, sender_password)
            server.send_message(message)

        return True
    except Exception as e :
            print(f"Error sending otp: {e}")
            return False
```

GUI Class Definition

```python
# Define GUI
class OTPApp :
  def __init__(self, root) :
    self.root = root
    self.root.title("OTP Verification")
    self.root.geometry("300x250")
    self.otp = ""
    self.attempts = 3

    font_style = tkfont.Font(family = "Arial Bold", size = 10)
    font_style2 = tkfont.Font(family = "Arial Bold", size = 9)

    # Email input
    self.email_label = tk.Label(root, text = "Enter your email:", font = font_style)
    self.email_label.pack(pady = 4)

    self.email_entry = tk.Entry(root, width = 30) # White background
    self.email_entry.pack(pady = 4)

    #Send otp button
    self.send_otp_button = tk.Button(root, text = "Send OTP", command = self.send_otp, bg = "#4CAF50", fg = "#fff", font = font_style2) # Green background
    self.send_otp_button.pack(pady = 10)

    # OTP input
    self.otp_label = tk.Label(root, text = "Enter OTP: ", font = font_style)
    self.otp_label.pack(pady = 4)

    self.otp_entry = tk.Entry(root, width = 15, bg = "#fff", fg = "#333") # White background
    self.otp_entry.pack(pady = 4)

    # Verify OTP button
    self.verify_otp_button = tk.Button(root, text = "Verify OTP", command = self.verify_otp, bg = "#2196F3", fg = "#fff", font = font_style2) # Blue Background
    self.verify_otp_button.pack(pady = 10)
```

- **Description:** This class defines GUI for the OTP verification system. The **__init__** method initializes the GUI components, setting up the layout and design of the application.

- **Attributes:**
  - **self.otp**: A string to store the generated OTP.
  - **self.attempts**: An integer to track the number of attempts left for OTP verification (initialized to 3).
  - **self.root:** The main window of the application.

- **Components:**
  - Labels and entry fields for user input (email and OTP).
  - Buttons for sending the OTP and verifying the entered OTP.
  - Font styles for a better visual experience.

Sending OTP Method

```
75    def send_otp(self) :
76      email = self.email_entry.get()
77      if not email :
78        messagebox.showerror("Error", "Please enter valid email ID")
79        return
80      self.otp = generate_otp()
81      if send_otp(email, self.otp) :
82        messagebox.showinfo("Success", "OTP sent successfully")
83      else:
84        messagebox.showerror("Error", "Enter valid email. Try again")
```

- **Description:** This method retrieves the email address from the input field, generates an OTP, and attempts to send it.

- **User Interaction:**
  - If the email field is empty, an error message is shown.
  - On successful sending, a success message is displayed.
  - If sending fails, an error message prompts the user to enter a valid email.

Verifying OTP Method

```
86  def verify_otp(self) :
87    user_otp = self.otp_entry.get()
88    if user_otp == self.otp :
89      messagebox.showinfo("Success", "OTP verified successfully. Access Granted")
90      self.attempts = 3
91      self.root.quit()
92
93    else:
94        self.attempts -= 1
95        if self.attempts > 0:
96            messagebox.showerror("Error", f"Incorrect OTP. You have {self.attempts} attempts left.")
97        else:
98          messagebox.showerror("Error", "No attempts left. Access denied.")
99          self.root.quit()
```

- **Description:** This method checks the OTP entered by the user against the generated OTP.

- **Logic:**
  - If the OTP matches, a success message is shown **access is granted**, and application closes.
  - If the OTP doesn't match, the number of attempts is Decremented.
  - If attempts remain, an error message indicates how many attempts are left.
  - If no attempts are left, an error message is displayed, and the application closes.

Main Execution Block

```
100  if __name__ == "__main__" :
101      root = tk.Tk()
102      app = OTPApp(root)
103      root.mainloop()
```

- **Description:** This block is executed when the script is run directly. It initializes the main application window and creates an instance of the OTPApp class, which sets up GUI and starts the event loop for user interaction.

User Instructions

1. **Enter Email:** Input your email address in the designated field. Ensure that the email is valid to receive the OTP.
2. **Send OTP:** Click the **"Send OTP"** button. The system will generate an OTP and send it to the provided email address.
3. **Check Email:** Open your email inbox and look for the message containing your OTP.
4. **Enter OTP**: Input the OTP received in your email in the provided field within the application.
5. **Verify OTP:** Click the **"Verify OTP"** button to validate the entered OTP against the generated one.
6. **Access:** If the OTP is correct, it will popup **"OTP verified successfully. Access Granted"**. If incorrect, you will be informed of the remaining attempts.

Security Considerations

- **Email Credentials**: Ensure that the sender's email and password are securely stored and not hardcoded in production code. Consider using environment variables or secure vaults for managing sensitive information.

- **SMTP Configuration:** The application currently uses Gmail's SMTP server. Ensure that the sender's account allows access to less secure apps or has an app password set up if two-factor authentication is enabled.

Conclusion

This OTP Verification System provides a robust method for user verification through email. It combines security with user-friendly design, making it suitable for a variety of applications requiring user authentication.