**GMAIL OTP VALIDATOR**

**-DHAIRYA MEHTA(1811023)**

**PROJECT DESCRIPTION AND OBJECTIVE:**The whole idea of the project is that with the help of a well known Python GUI ie Tkinter and another of Python's vast libraries smtplib,we can send an OTP to gmail which can be validated on the Tkinter GUI.

**BIBLIOGRAPHY:**

**The references for the project are:**

1. <https://www.geeksforgeeks.org/python-gui-tkinter/>
2. <https://www.geeksforgeeks.org/simple-mail-transfer-protocol-smtp/>
3. <https://www.tutorialspoint.com/python/python_gui_programming.htm>
4. <https://stackoverflow.com/questions/16512592/login-credentials-not-working-with-gmail-smtp>

**BACKGROUND AND INTRODUCTION:**

A **one-time password** (**OTP**), also known as **one-time pin** or **dynamic password**, is a password that is valid for only one login session or transaction, on a computer system or other digital device. OTPs avoid a number of shortcomings that are associated with traditional (static) password-based authentication; a number of implementations also incorporate [two-factor authentication](https://en.wikipedia.org/wiki/Two-factor_authentication" \o "Two-factor authentication) by ensuring that the one-time password requires access to *something a person has* (such as a small keyring fob device with the OTP calculator built into it, or a smartcard or specific cellphone) as well as *something a person knows* (such as a PIN).

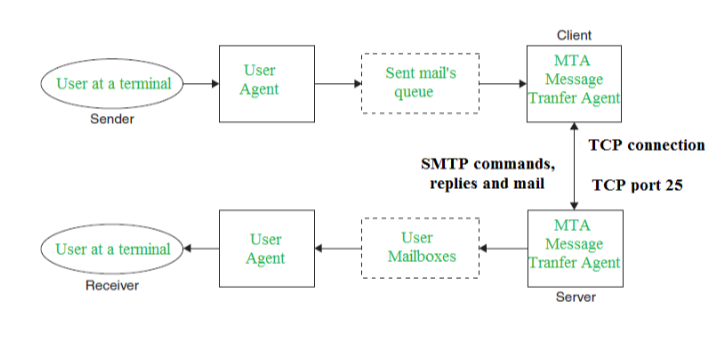
**SMTP:**

The SMTP model is of two type :

1. End-to- end method
2. Store-and- forward method

The end to end model is used to communicate between different organizations whereas the store and forward method are used within an organization. A SMTP client who wants to send the mail will contact the destination’s host SMTP directly in order to send the mail to the destination. The SMTP server will keep the mail to itself until it is successfully copied to the receiver’s SMTP.  
The client SMTP is the one which initiates the session let us call it as the client- SMTP and the server SMTP is the one which responds to the session request and let us call it as receiver-SMTP. The client- SMTP will start the session and the receiver-SMTP will respond to the request.  
  
**Model of SMTP system**

In the SMTP model user deals with the user agent (UA) for example Microsoft Outlook, Netscape, Mozilla, etc. In order to exchange the mail using TCP, MTA is used. The users sending the mail do not have to deal with the MTA it is the responsibility of the system admin to set up the local MTA. The MTA maintains a small queue of mails so that it can schedule repeat delivery of mail in case the receiver is not available. The MTA delivers the mail to the mailboxes and the information can later be downloaded by the user agents.



**Communication between sender and the receiver :**  
The senders, user agent prepare the message and send it to the MTA. The MTA functioning is to transfer the mail across the network to the receivers MTA. To send mail, a system must have the client MTA, and to receive mail, a system must have a server MTA.

**CODE:**

import random

import smtplib

import tkinter as tk

from tkinter import \*

from email.mime.text import MIMEText

from email.mime.multipart import MIMEMultipart

# Defining CreateWidgets() function to create necessary tkinter widgets

def CreateWidget():

emailLabel = Label(root, text="ENTER YOUR EMAIL-ID : ", bg="deepskyblue3")

emailLabel.grid(row=0, column=1, padx=5, pady=5)

emailEntry = Entry(root, textvariable=emailid, width=30)

emailEntry.grid(row=0, column=2, padx=5, pady=5)

sendOTPbutton = Button(root, text="Send OTP", command=sendOTP, width=20)

sendOTPbutton.grid(row=0, column=3, padx=5, pady=5)

root.msgLabel = Label(root, bg="deepskyblue3")

root.msgLabel.grid(row=1, column=1, padx=5, pady=5, columnspan=3)

otpLabel = Label(root, text="ENTER THE OTP : ", bg="deepskyblue3")

otpLabel.grid(row=2, column=1, padx=5, pady=5)

root.otpEntry = Entry(root, textvariable=otp, width=30, show="\*")

root.otpEntry.grid(row=2, column=2, padx=5, pady=5)

validOTPbutton = Button(root, text="Validate OTP", command=validOTP, width=20)

validOTPbutton.grid(row=2, column=3, padx=5, pady=5)

root.otpLabel = Label(root, bg="deepskyblue3")

root.otpLabel.grid(row=3, column=1, padx=5, pady=5, columnspan=3)

# Defining sendOTP() to generate and send OTP to user-input email-id

def sendOTP():

# Storing digits from 0 to 9 as string in numbers variable & declaring empty string

# variable named root.genOTP

numbers = "0123456789"

root.genOTP = ""

# Fetching and storing user-input mail id in receiverEmail variable

receiverEmail = emailid.get()

# Generating 6-digits OTP

for i in range(6):

root.genOTP += numbers[int(random.random() \* 10)]

# Concatenating and Storing generated OTP with Message to be sent in otpMSG

otpMSG = "YOUR OTP IS : " + root.genOTP

# Creating instance of class MIMEMultipart()

message = MIMEMultipart()

# Storing the email details in respective fields

message['FROM'] = "OTP VALIDATOR "

message['To'] = receiverEmail

message['Subject'] = "OTP VALIDATION"

# Attaching the otgMSG with MIME instance

message.attach(MIMEText(otpMSG))

# Creating a smtp session

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

# Starting TLS for security

smtp.starttls()

# Authenticating the sender using the login() method

smtp.login("xaekaindka@gmail.com", "zaykaindiaka")

# Sending the email with Mulitpart message converted into string

smtp.sendmail("xaekaindka@gmail.com", receiverEmail, message.as\_string())

# Terminating the session

smtp.quit()

# Formatting receiveEmail to replace(hide) mail id with \*

receiverEmail = '{}\*\*\*\*\*\*\*\*{}'.format(receiverEmail[0:2], receiverEmail[-10:])

# Displaying the success message

root.msgLabel.config(text = "OTP HAS BEEN SENT TO " + receiverEmail)

# Defining validOTP() to validate user-input OTP mail with script generated OTP

def validOTP():

# Storing user-input OTP

userInputOTP = otp.get()

# Storing system generated OTP

systemOTP = root.genOTP

# Validating OTP

if userInputOTP == systemOTP:

root.otpLabel.config(text="OTP VALIDATED SUCCESSFULLY")

else:

root.otpLabel.config(text="INVALID OTP")

# Creating object of tk class

root = tk.Tk()

# Setting the title, background color and disabling the resizing property

root.title("EmailOTP")

root.resizable(False, False)

root.config(background = "deepskyblue3")

# Creating tkinter variables

emailid = StringVar()

otp = StringVar()

# Calling the CreateWidgets() function with argument bgColor

CreateWidget()

# Defining infinite loop to run application

root.mainloop()

**CONCLUSION:**This project is a very small scale implementation of OTP validation and its validation through email is understood,implemented and learnt.There are better ways to implement OTP validation than this one but with the help of Tkinter, an easy but interesting GUI application is created.