# **OTP Verification System**

**Introduction**

This document describes the implementation of an OTP (One-Time Password) verification system. The system is designed using Python's Tkinter library for the GUI, and it utilizes random number generation and email services to send the OTP to a user’s email address for verification purposes. Such systems are commonly used in authentication processes to ensure the identity of a user before granting access to certain resources.

**Implementation Details**

**GUI Design**

The Graphical User Interface (GUI) is built using the Tkinter library. The main components of the GUI include a label for entering an email address, an entry widget for user input, and a button that triggers the OTP generation and email sending process.

**OTP Generation**

The OTP is generated using Python's random library. A random 6-digit number is created to serve as the OTP. This OTP is then displayed to the user or sent to their email for verification.

The following code snippet shows the OTP generation process:

def otp\_gen():  
 otp = random.randint(100000, 999999)  
 return otp

**User’s Input**

This function creates a simple form with a label, entry box, and button in a grid layout. The user enters their email in the entry box, and clicking the button will trigger the pro function with the entered email as input. The mail\_id function is a GUI component created using Python's Tkinter library. This function builds a simple user interface (UI) that prompts the user to enter their email address and then click a button to generate an OTP (One-Time Password).

def mail\_id():

    lbl = Label(root,text='Enter your E-mail')

    lbl.grid(row=0,column=1,padx=20,pady=30)

    user = Entry(root,width=30)

    user.grid(column=2,row=0,padx=10,pady=30)

    user\_btn = Button(root,text='click to get OTP',command=lambda:pro(user.get()))

    user\_btn.grid(column=3,row=0,padx=50,pady=30)

#Calling OTP function

def pro(mail):

        valid\_email\_address = user\_email(mail)

        if valid\_email\_address:

           send\_otp(valid\_email\_address)

**Email Validation**

The email validation process checks if the entered email address is valid based on known domains and extensions. The validation ensures that the email address follows the correct format before proceeding with the OTP sending.

The following code snippet shows the email validation process:

def user\_email(email\_add):  
 dom = ["gmail", "hotmail", "yahoo", "outlook"]  
 ext = ["com", "in", "org", "edu", "co.in"]  
 s = email\_add.split("@")  
 if len(s) != 2:  
 messagebox.showerror("Error", "Invalid Email. Please enter again.")  
 return None  
 else:  
 test = s[1].split(".", 1)  
 if (' ' not in email\_add) and ("@" in email\_add) and (test[0] in dom) and (test[1] in ext):  
 return email\_add  
 else:  
 messagebox.showerror("Error", "Invalid Email. Make sure you don't have spaces while entering.")  
 return None

**Sending OTP via Email**

Once the email address is validated, the OTP is sent to the user's email using the smtplib library. The SMTP protocol is used to connect to the email server and send the message.

The following code snippet shows how the OTP is sent via email:

def send\_otp(valid\_email\_address):  
 s = smtplib.SMTP("smtp.gmail.com", 587)  
 s.starttls()  
 s.login("your\_email@gmail.com", 'your\_app\_password')  
 subject = "OTP verification"  
 send = otp\_gen()  
 body = "Your OTP is: " + str(send)  
 message = f'subject:{subject}\n\n{body}'  
 s.sendmail('your\_email@gmail.com', valid\_email\_address, message)  
 messagebox.showinfo(title='Success', message='OTP sent successfully')

**Expected Output**

When the program is executed, a GUI window appears, asking the user to enter their email address. Upon entering a valid email and clicking the button, the OTP is generated and sent to the provided email address. If the email is invalid, an error message is displayed, prompting the user to enter a correct email address. The successful sending of the OTP is acknowledged with a message box informing the user that the OTP was sent.

**OTP VERIFICATION**

 This function handles the OTP validation process with input validation, error handling, and a retry mechanism.

 The user has a maximum of 4 attempts to enter the correct OTP. If they fail, they can choose to try again or exit the application.

Code Snippet :

count = 4

def otp\_val(user):

    global count

    try:

        user\_otp = int(user)

    except Exception as e:

        messagebox.showerror('Value Error','number must be integer')

        otp\_entry.delete(0,tk.END)

        return

    if user\_otp == otpnum:

        messagebox.showinfo('Success','Access Granted')

        root.quit()

    else:

                count -= 1

                if count>0:

                    messagebox.showwarning("warning",f'Invalid number.enter number Again.Last {count} chances')

                    otp\_entry.delete(0,tk.END)

                else:

                        messagebox.showerror("Error","Invalid number,your chances are over.")

                        user1=messagebox.askquestion('question','do you want to enter again.')

                        if user1 == 'yes':

                            count=4

                            mail\_id()

                            otp\_entry.delete(0,tk.END)

                        else :

                             root.quit()

**OTP User Input**

 The code creates a simple user interface for entering and verifying an OTP (One-Time Password).

 A label prompts the user to enter the OTP.

 An entry widget is used for input.

 A button is provided to trigger the verification process.

Code snippet :

lbl = Label(root,text='Enter OTP')

lbl.grid(row=1, column=1, padx=20, pady=30)

otp\_entry = Entry(root,width=10)

otp\_entry.grid(row=1, column=2, padx=10, pady=30)

otp\_btn = Button(root,text='Verify number',command=lambda:otp\_val(otp\_entry.get()))

otp\_btn.grid(row=1, column=3, padx=50, pady=30)

**OUTPUT** :







