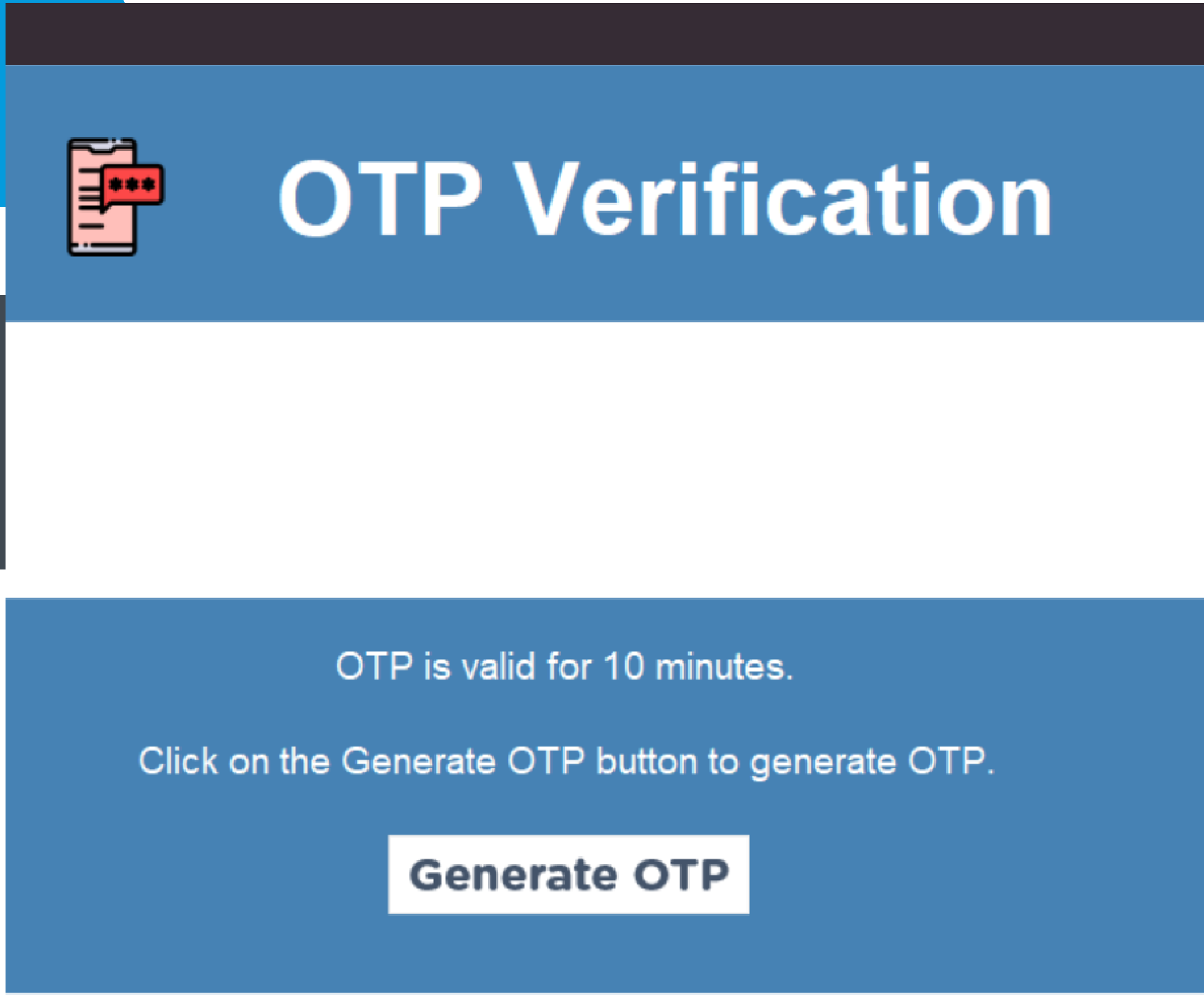
Borcelle School

PROJECT

OTP VERIFICATION GUI

Presentation by

Shashwat Srivastava


OTP Verification

OTP is valid for 10 minutes.

Click on the Generate OTP button to generate OTP.

Table Of Content

About the Project

Project Workflow

Modules Imported

Widgets Used

Functions Used

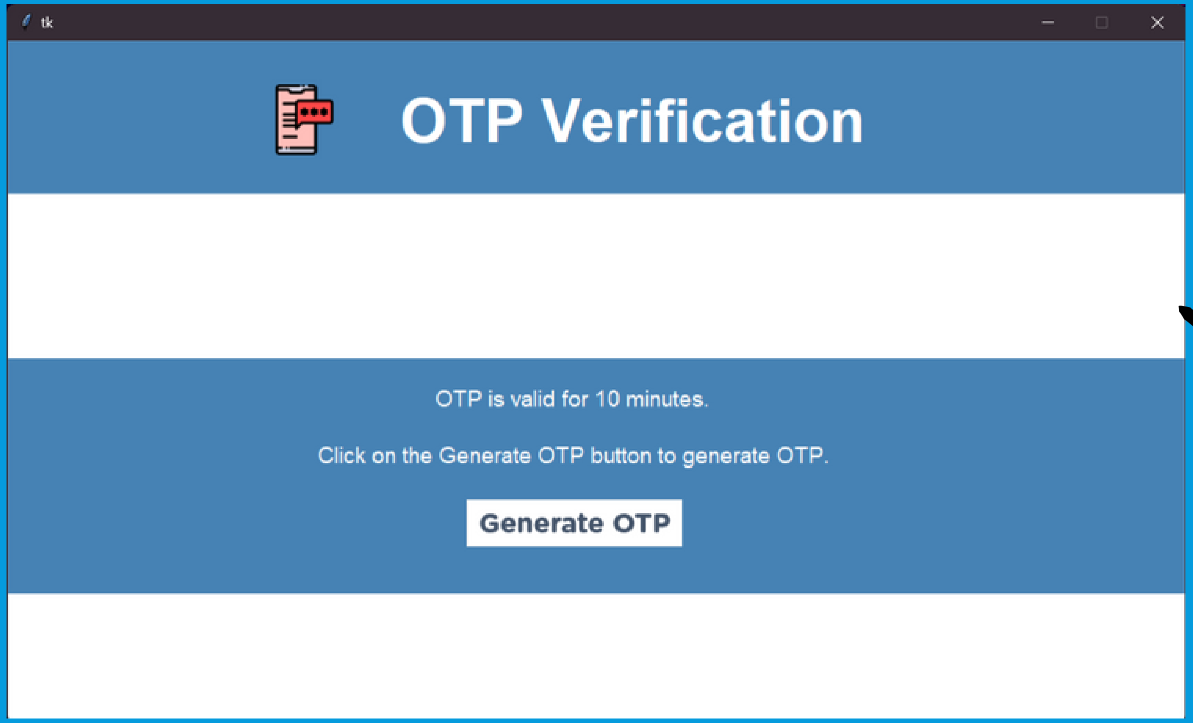
Application Benefits

Source Code

About The Project

OTP Verification is the process of verifying a user by sending a unique password so that the user can be verified before completing a registration or payment process. OTP Verification GUI verifies the Mobile Number of users by sending a unique OTP verification code during registration and login. It removes the possibility of a user registering with a fake Email Address/Mobile Number.

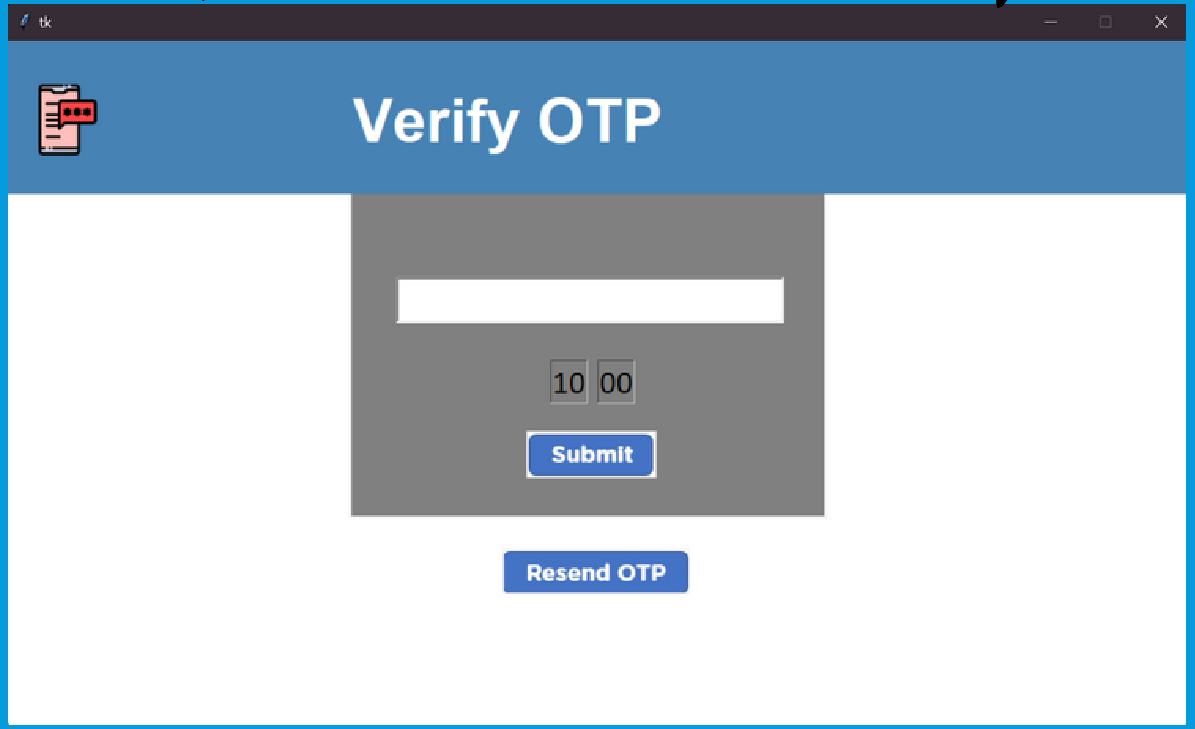
Project Workflow


tk

OTP Verification

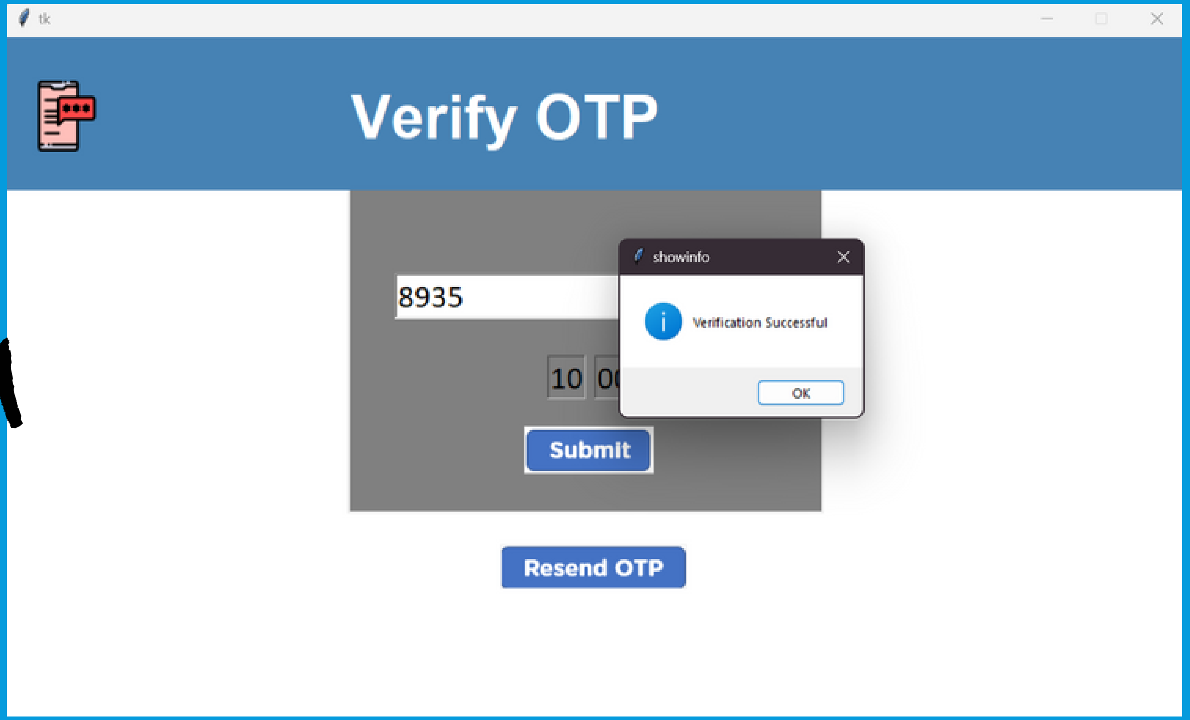
OTP is valid for 10 minutes.

Click on the Generate OTP button to generate OTP.


tk

Verify OTP

10 00


tk

Verify OTP

8935

10 00

showinfo

Verification Successful

Modules Imported

Tkinter

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's de facto standard GUI. Tkinter is included with standard Linux, Microsoft Windows and macOS installs of Python.

Twilio

The Twilio Python Helper Library makes it easy to interact with the Twilio API from your Python application. It is used to send OTP to users Phone Number using an virtual Twilio number.

Random

Python has a built-in module that you can use to make random numbers. Random is being used in the project to generate a random OTP within a certain range using randrange function.

Widgets Used

Frame

A frame is a rectangular region on the screen. A frame can also be used as a foundation class to implement complex widgets. It is used to organize a group of widgets.

Label

The Label is used to specify the container box where we can place the text or images. This widget is used to provide the message to the user about other widgets used in the python application.

Canvas

The Canvas widget lets us display various graphics on the application. It can be used to draw simple shapes to complicated graphs. We can also display various kinds of custom widgets according to our needs.

Functions Used

Geometry

Tkinter provides many methods; one of them is the geometry() method. This method is used to set the dimensions of the Tkinter window and is used to set the position of the main window on the user's desktop.

PhotoImage

PhotoImage() function returns the image object. To display image in Python is as simple as that. PhotoImage class only supports GIF and PGM/PPM formats. The more generalized formats are JPEG/JPG and PNG.

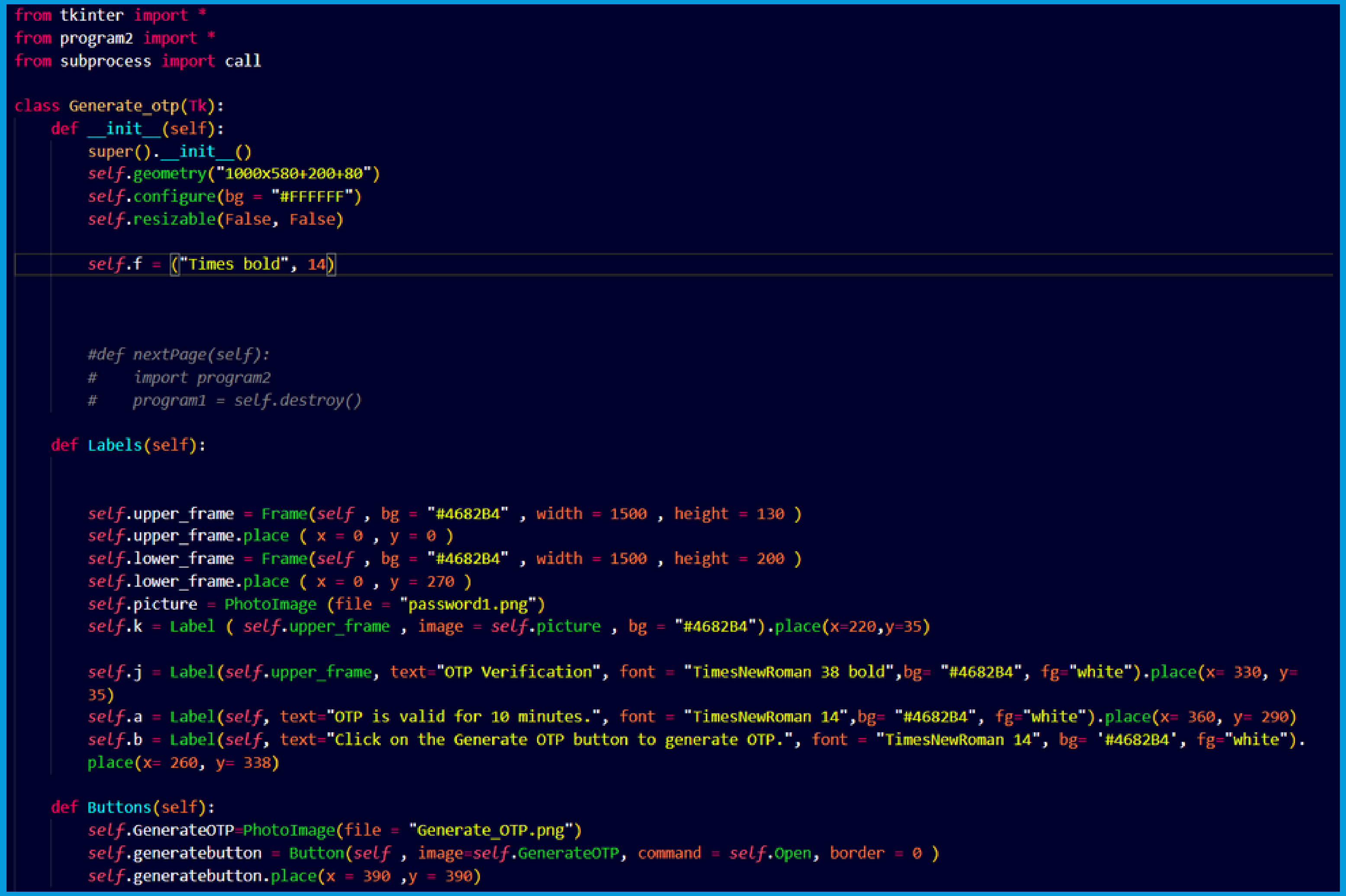
MessageBox

MessageBox Widget is used to display the message boxes in the python applications. This module is used to display a message using provides a number of functions.

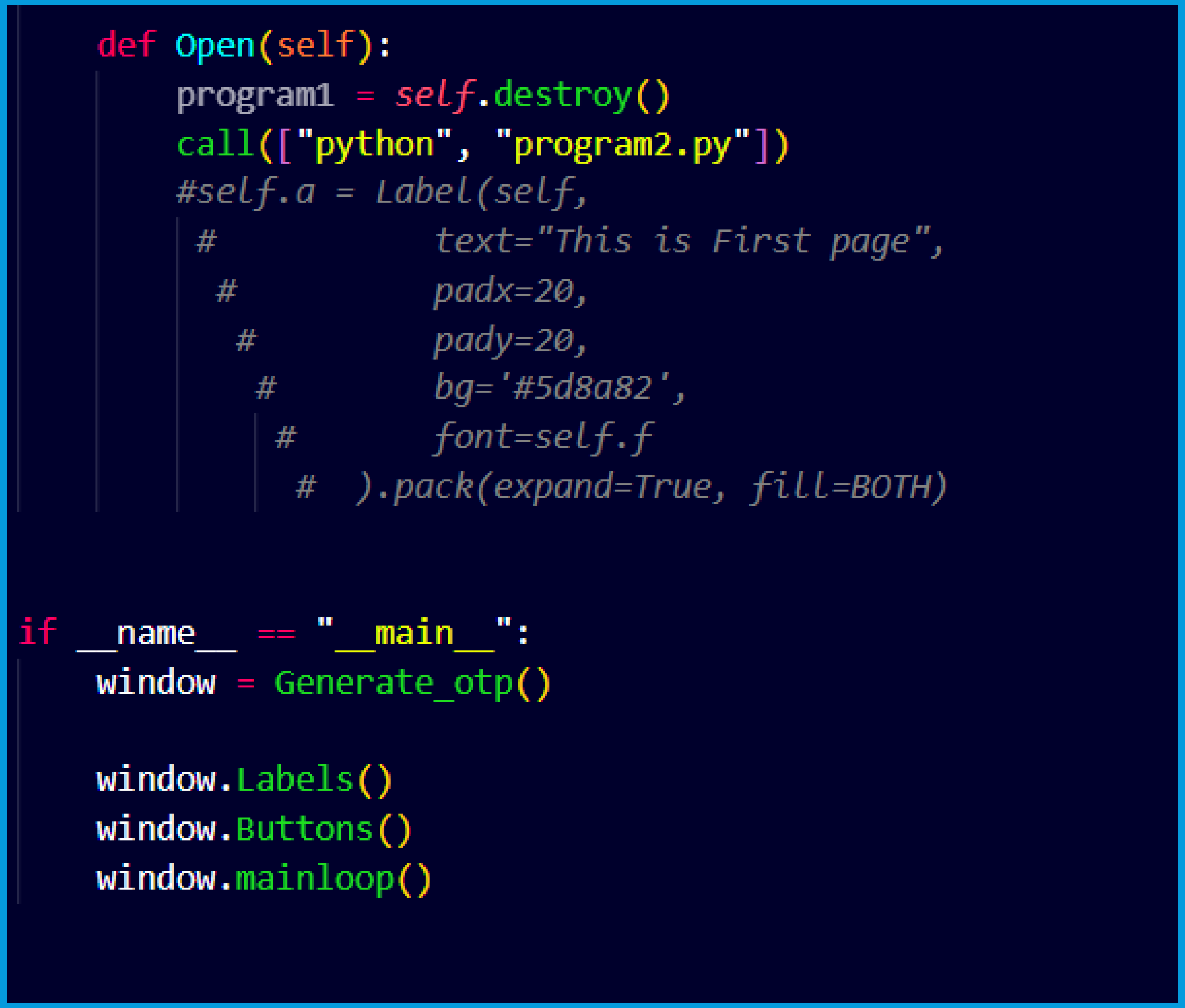
Application Benefits

- OTPs can be used as **single-factor authentication** to replace static passwords, where:
- Instead of a customer creating their own username and password, they're issued a unique PIN for each session
- Or, they can be used *in addition* to user-generated credentials for **two-factor authentication** (2FA) during sign-up, login, or transaction approvals, where:
- A customer attempts to use their username and password from an unrecognized device
- The customer then receives and uses their OTP to verify their identity and device
- Timer used in the GUI helps restrict the user to use fake IDs for registrations or any transaction process.

Source Code

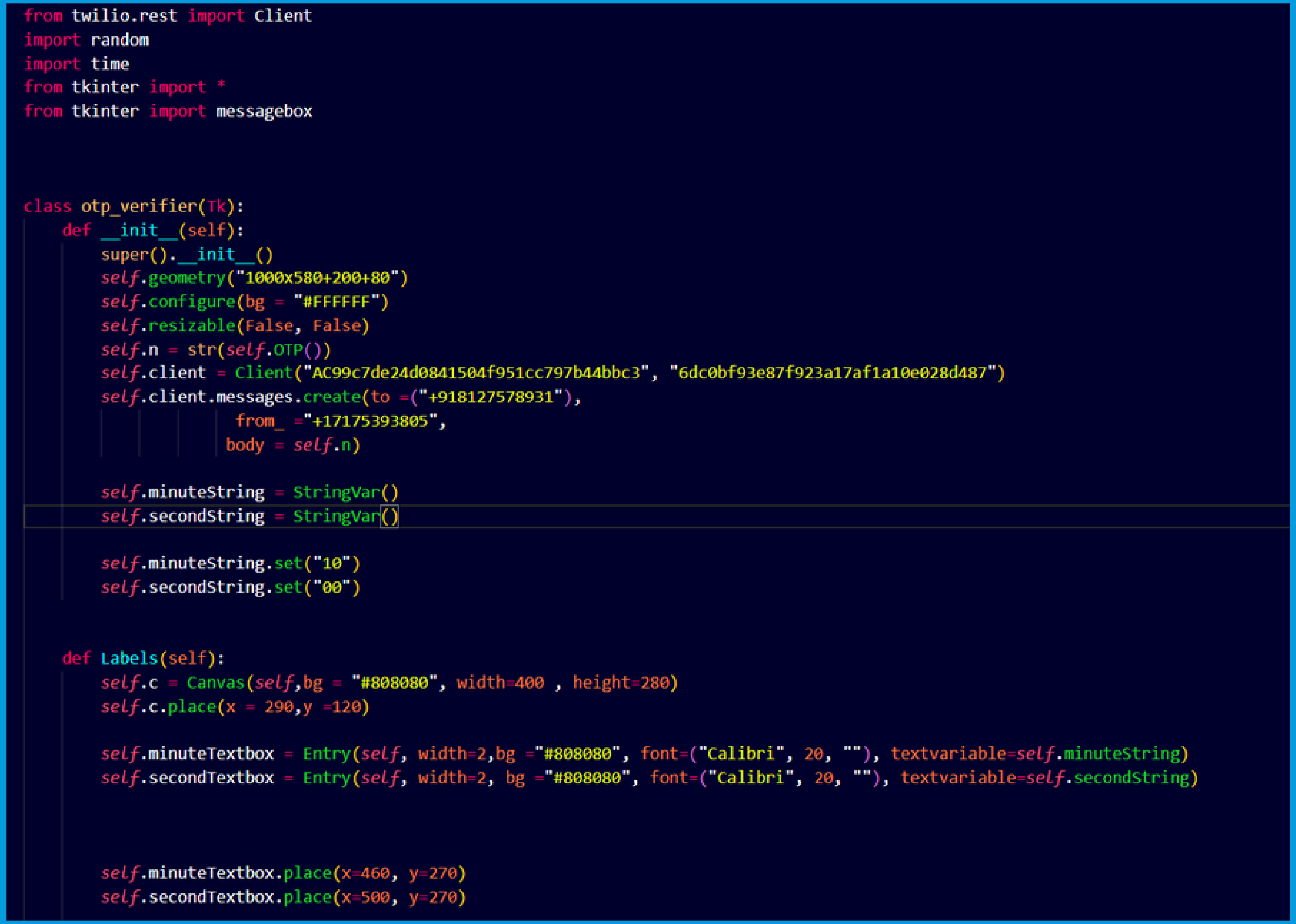


```python
from tkinter import *
from program2 import *
from subprocess import call

class Generate_otp(Tk):
    def __init__(self):
        super().__init__()
        self.geometry("1000x580+200+80")
        self.configure(bg = "#FFFFFF")
        self.resizable(False, False)

        self.f = ("Times bold", 14)



        #def nextPage(self):
        #    import program2
        #    program1 = self.destroy()

    def Labels(self):


        self.upper_frame = Frame(self , bg = "#4682B4" , width = 1500 , height = 130 )
        self.upper_frame.place ( x = 0 , y = 0 )
        self.lower_frame = Frame(self , bg = "#4682B4" , width = 1500 , height = 200 )
        self.lower_frame.place ( x = 0 , y = 270 )
        self.picture = PhotoImage (file = "password1.png")
        self.k = Label ( self.upper_frame , image = self.picture , bg = "#4682B4").place(x=220,y=35)

        self.j = Label(self.upper_frame, text="OTP Verification", font = "TimesNewRoman 38 bold",bg= "#4682B4", fg="white").place(x= 330, y=
        35)
        self.a = Label(self, text="OTP is valid for 10 minutes.", font = "TimesNewRoman 14",bg= "#4682B4", fg="white").place(x= 360, y= 290)
        self.b = Label(self, text="Click on the Generate OTP button to generate OTP.", font = "TimesNewRoman 14", bg= '#4682B4', fg="white").
        place(x= 260, y= 338)

    def Buttons(self):
        self.GenerateOTP=PhotoImage(file = "Generate_OTP.png")
        self.generatebutton = Button(self , image=self.GenerateOTP, command = self.Open, border = 0 )
        self.generatebutton.place(x = 390 ,y = 390)
```

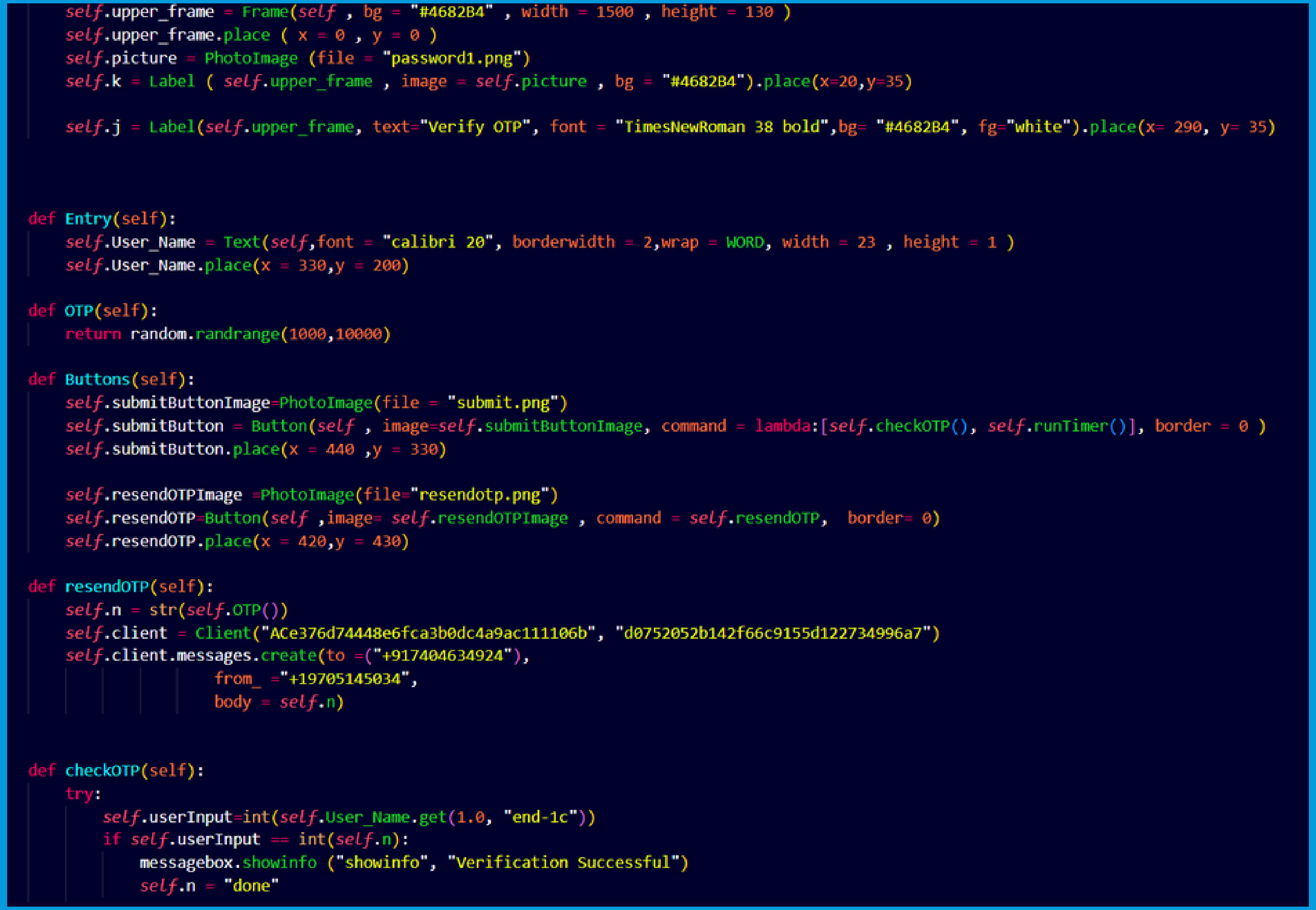


```python
    def Open(self):
        program1 = self.destroy()
        call(["python", "program2.py"])
        #self.a = Label(self,
        #           text="This is First page",
         #          padx=20,
          #         pady=20,
           #        bg='#5d8a82',
            #       font=self.f
             #  ).pack(expand=True, fill=BOTH)


if __name__ == "__main__":
    window = Generate_otp()

    window.Labels()
    window.Buttons()
    window.mainloop()
```

Source Code

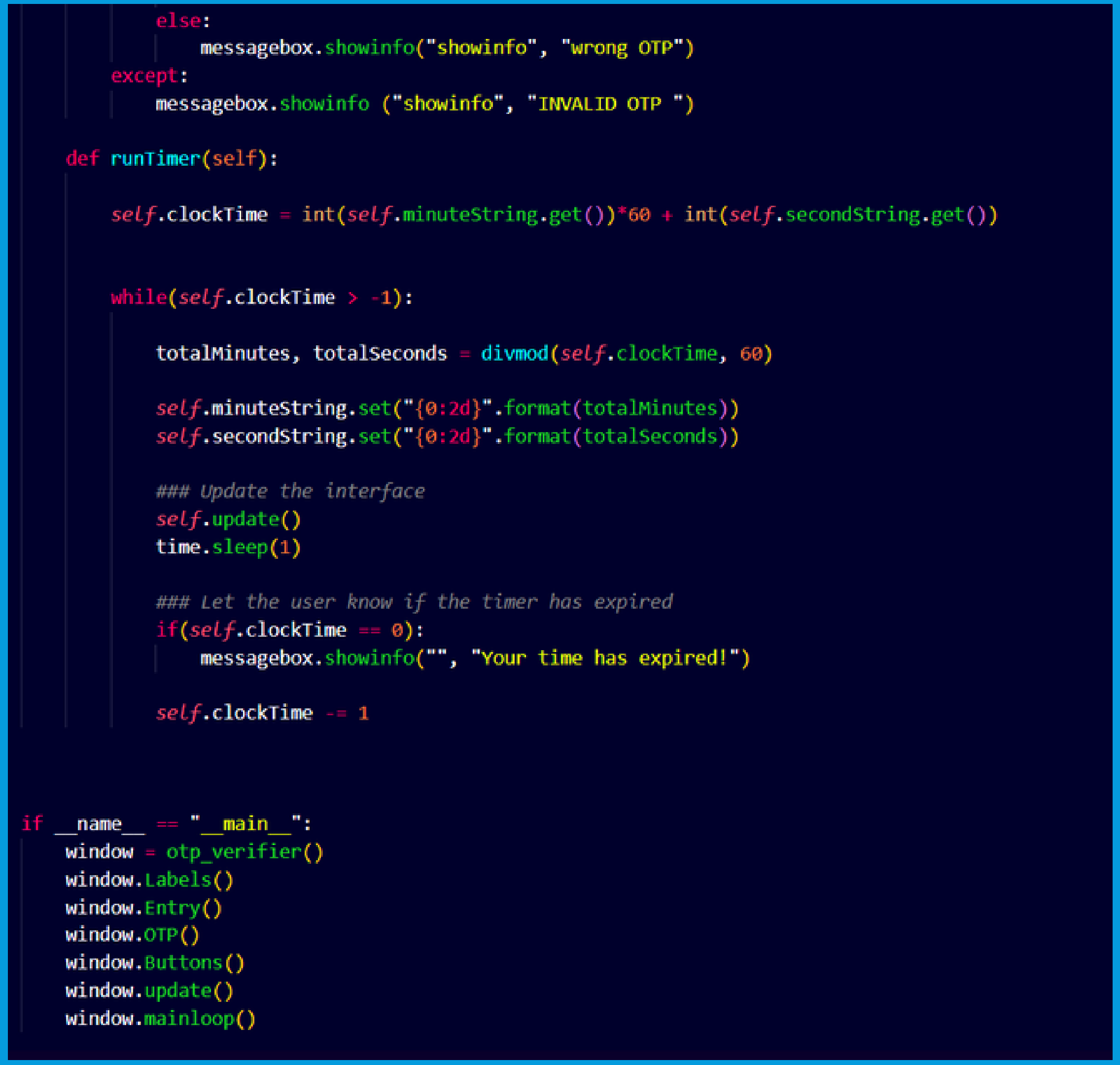


```python
from twilio.rest import Client
import random
import time
from tkinter import *
from tkinter import messagebox



class otp_verifier(Tk):
    def __init__(self):
        super().__init__()
        self.geometry("1000x580+200+80")
        self.configure(bg = "#FFFFFF")
        self.resizable(False, False)
        self.n = str(self.OTP())
        self.client = Client("AC99c7de24d0841504f951cc797b44bbc3", "6dc0bf93e87f923a17af1a10e028d487")
        self.client.messages.create(to =("+918127578931"),
                                     from_ ="+17175393805",
                                    body = self.n)

        self.minuteString = StringVar()
        self.secondString = StringVar()

        self.minuteString.set("10")
        self.secondString.set("00")


    def Labels(self):
        self.c = Canvas(self,bg = "#808080", width=400 , height=280)
        self.c.place(x = 290,y =120)

        self.minuteTextbox = Entry(self, width=2,bg ="#808080", font=("Calibri", 20, ""), textvariable=self.minuteString)
        self.secondTextbox = Entry(self, width=2, bg ="#808080", font=("Calibri", 20, ""), textvariable=self.secondString)



        self.minuteTextbox.place(x=460, y=270)
        self.secondTextbox.place(x=500, y=270)
```



```python
        self.upper_frame = Frame(self , bg = "#4682B4" , width = 1500 , height = 130 )
        self.upper_frame.place ( x = 0 , y = 0 )
        self.picture = PhotoImage (file = "password1.png")
        self.k = Label ( self.upper_frame , image = self.picture , bg = "#4682B4").place(x=20,y=35)

        self.j = Label(self.upper_frame, text="Verify OTP", font = "TimesNewRoman 38 bold",bg= "#4682B4", fg="white").place(x= 290, y= 35)



    def Entry(self):
        self.User_Name = Text(self,font = "calibri 20", borderwidth = 2,wrap = WORD, width = 23 , height = 1 )
        self.User_Name.place(x = 330,y = 200)

    def OTP(self):
        return random.randrange(1000,10000)

    def Buttons(self):
        self.submitButtonImage=PhotoImage(file = "submit.png")
        self.submitButton = Button(self , image=self.submitButtonImage, command = lambda:[self.checkOTP(), self.runTimer()], border = 0 )
        self.submitButton.place(x = 440 ,y = 330)

        self.resendOTPImage =PhotoImage(file="resendotp.png")
        self.resendOTP=Button(self ,image= self.resendOTPImage , command = self.resendOTP,  border= 0)
        self.resendOTP.place(x = 420,y = 430)

    def resendOTP(self):
        self.n = str(self.OTP())
        self.client = Client("ACe376d74448e6fca3b0dc4a9ac111106b", "d0752052b142f66c9155d122734996a7")
        self.client.messages.create(to =("+917404634924"),
                                    from_ ="+19705145034",
                                    body = self.n)


    def checkOTP(self):
        try:
            self.userInput=int(self.User_Name.get(1.0, "end-1c"))
            if self.userInput == int(self.n):
                messagebox.showinfo ("showinfo", "Verification Successful")
                self.n = "done"
```

Source Code



```
            else:
                messagebox.showinfo("showinfo", "wrong OTP")
        except:
            messagebox.showinfo ("showinfo", "INVALID OTP ")

    def runTimer(self):

        self.clockTime = int(self.minuteString.get())*60 + int(self.secondString.get())


        while(self.clockTime > -1):

            totalMinutes, totalSeconds = divmod(self.clockTime, 60)

            self.minuteString.set("{0:2d}".format(totalMinutes))
            self.secondString.set("{0:2d}".format(totalSeconds))

            ### Update the interface
            self.update()
            time.sleep(1)

            ### Let the user know if the timer has expired
            if(self.clockTime == 0):
                messagebox.showinfo("", "Your time has expired!")

            self.clockTime -= 1



if __name__ == "__main__":
    window = otp_verifier()
    window.Labels()
    window.Entry()
    window.OTP()
    window.Buttons()
    window.update()
    window.mainloop()
```

THANK YOU

PROJECT SUBMITTED TO : **DR. HARSH SOHAL**

PROJECT SUBMITTED BY : **SHASHWAT SRIVASTAVA**

ROLL NO : **211560**