Dayananda Sagar College of Engineering
Department of Electronics and Communication Engineering

Assignment

Program	: B.E.	**Semester:**	6
Course	: Python Programming	**Section**	: D
Course Code	: 19CE6IEPYP	**Date**	: 01-07-2022

A Report on:

Secure Virtual Key Entry
Develop a mechanism to secure the virtual key entry.

Submitted by:

USN : 1DS19EC711	**NAME: CHIRANTHGOWDA Y.T**
USN : 1DS19EC717	**NAME: HRUSHIKESH H .D**
USN : 1DS19EC719	**NAME: KISHAN R. B**
USN : 1DS19EC724	**NAME: N K VINAY GOWDA**

Faculty In-charge:

DR. S. THENMOZHI

Signature of Faculty In-charge

- **Use Case–Heading:**

Secure Virtual Key Entry

Develop a mechanism to secure the virtual key entry.

- **Use case- Description** :

To come up with a more efficient authentication algorithm for secure virtual key entry.

- **Section : D**

- **Team Members :**

Chiranth Gowda Y. T

Hrushikesh H.D

Kishan R.B

N. K Vinay Gowda

- **Algorithm :**

USER SETUP PROCESS:

1) Taking sample images of the user and training the model.
2) Labelling the user using a unique ID
3) Storing the Name, and email ID for two-step authentication.
4) And the mail will be sent to all previously registered users and new users will get QR for the next authentication process.
5) Set-up is done.

RECOGNIZING USERS AND VERIFYING USING 2-FACTOR AUTHENTICATION:

1) Turning on the camera and taking a real-time face identification.
2) If a user is identified, the email ID will be retrieved using the person's label.

3) If no users are found, access is denied, and the photo of an unknown person will be captured and sent to all registered users as a security alert.

4) A previous QR CODE image that is sent to the email ID must be used for 2-factor authentication.

5) The user must then show the QR code picture to the camera.

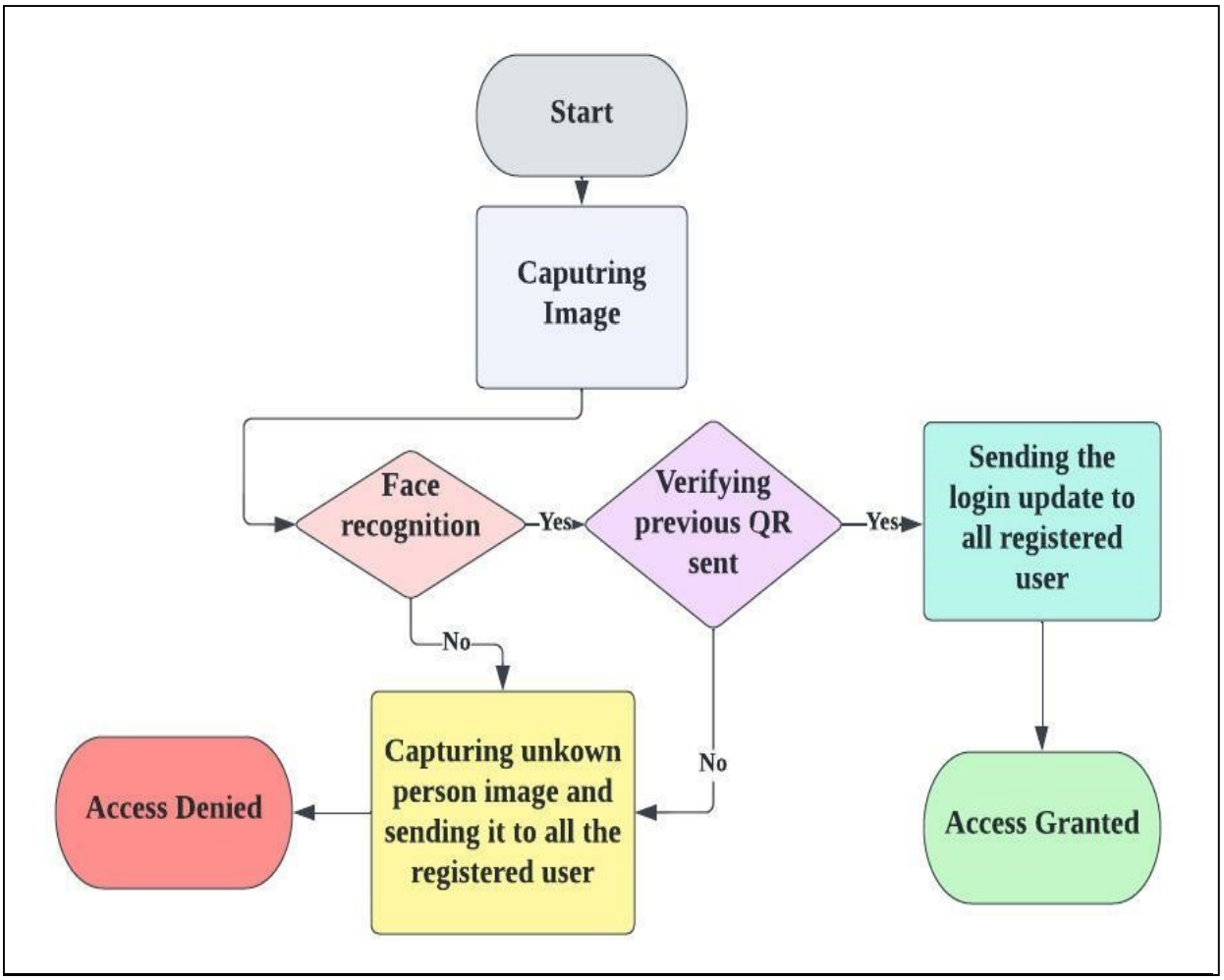
6) The previous QR sent to all registered users will be stored that will be used for authentication.

7) Access is allowed if the QR CODE matches.

8) If not, access is prohibited.

9) End

- **Flowchart :**



- **Requirement:**
  opencv-python=4.6.0.66
  face-recognition=1.3.0
  pickleshare=0.75
  tk=8.6.11
  numpy=1.23.0
  qrcode=7.3.1
- **Program with comments :**
  EXECUTABLE PROGRAMS

#----------

Register_exe.py

```
#THE FIRST PROGRAM TO REGISTER THE USER
import Register_backend as lud
import time
from E_Mail_backend import *

win = Tk()

win.geometry('600x600')
win.title('Registeration Window')
win.config(bg='Lightskyblue')

emails = []

def store_data():
    global emails
    ref_id = textin_id.get()
    name = textin_name.get()
    email = textin_email.get()
    ref_dictt, embed_dict , Data= lud.reg(ref_id , name , email)
    if Data:
        for key , value in ref_dictt.items():
            try:
                email = ref_dictt[key]['email']
                emails.append(email)
            except:
                pass
        user_id = f'ID:- {ref_id}'
        user_name = f'Name:- {name}'
        user_email = f'Email:- {email}'
        list.insert(0,user_id)
        list.insert(1,user_name)
        list.insert(2,user_email)
```

```
        emails.remove(email)
        send_mail_to_all(emails ,email  , name , 'Registeration Update', 'Successfully Registered')
    else:
        win.destroy()
        time.sleep(2)
        win2 = Tk()
        win2.geometry('400x300')
        win2.title('ERROR....!!!!')
        win2.config(bg='black')
        l1 = Label(win2, text='NO DATA STORED' , width=30 , padx=3 , font = ('calibre',15), bg = 'black' , fg =
'red')
        l1.place(relx = 0.5, rely = 0.5 , anchor = 'center')
        win2.mainloop()
        send_mail_to_all(emails,email ,name ,"Regsiteration Update" ,  'Failed To Register')

l = Label(win, text='USER REGISTRATION' , width=20 , padx=3 , font = ('calibre',15) , bg = 'lightgreen')
l.place(x=210,y=30)

l1 = Label(win, text='Enter ID' , font = ('calibre',10), bg = 'red', padx=3)
l1.place(x=285,y=100)
textin_id = StringVar()
e1 = Entry(win, width=30 , textvariable = textin_id,font=('Ubuntu', 15))
e1.place(x = 150 , y = 130)

l2 = Label(win, text='Enter Name' , font = ('calibre',10), bg = 'red', padx=3)
l2.place(x=280,y=170)
textin_name = StringVar()
e2 = Entry(win, width=30 , textvariable = textin_name,font=('Ubuntu', 15))
e2.place(x = 150 , y = 200)

l3 = Label(win, text='Enter Email' , font = ('calibre',10), bg = 'red', padx=3)
l3.place(x=280,y=240)
textin_email = StringVar()
e3 = Entry(win, width=30 , textvariable = textin_email,font=('Ubuntu', 15))
e3.place(x = 150 , y = 270)

b1 = Button(win, text='Register Myself' , font=('Ubuntu', 10) , command = store_data)
b1.place(x = 270 , y = 320)

l4 = Label(win, text='After Registering Wait For The Camera To Open \n Press "S" 5 Times To Capture Your
Photo' , font = ('bold',10), bg = 'lightpink', padx=3)
l4.place(x=170,y=370)

l4 = Label(win, text='Registered User Info' , font = ('calibre',10), bg = 'red', padx=3)
l4.place(x=255,y=460)
```

```
list = Listbox(win, width=60 , height=5)
list.place(x=140,y=500)

win.mainloop()
```

OUTPUT:

#----------

Two_Step_Verify.py

#THIS PROGRAM WILL OUTPUT THE DETECTION WINDOW FOR DETECTING USER

```
from tkinter import *
import time
import Access_Denied_Handler
import Access_Granted_Handler
from E_Mail_backend import *
from Access_Granted_Handler import *
import cv2

def close():
    time.sleep(1)
```

```
    win.destroy()

user_email = ''
name = ''
emails = []

def scan():
    global id, user_email , name , emails
    win.destroy()
    granted = Access_Granted_Handler.scan_code(id)
    if granted:
        user_email = user_mail(id)
        name = user_name(id)
        emails = all_emails()
        send_mail_to_all(emails ,user_email  , name ,'User Login Found' , 'Logged In')

id , closed = Access_Denied_Handler.main()

if ((id == '404') | closed):

    win2 = Tk()
    win2.geometry('400x300')
    win2.title('ERROR....!!!!')
    win2.config(bg='black')
    l1 = Label(win2, text='Face Not Detected' , width=30 , padx=3 , font = ('calibre',15), bg = 'black' , fg = 'red')
    l1.place(relx = 0.5, rely = 0.5 , anchor = 'center')
    win2.mainloop()

else:
    win = Tk()
    win.geometry('600x200')
    win.title('Verification Window')
    win.config(bg='Lightskyblue')

    l = Label(win, text='Show The QR Code You Have' , width=40 , padx=3 , font = ('calibre',15) , bg =
'lightgreen')
    l.place(x=80,y=50)
    b = Button(win, text='Open Camera' , font=('Ubuntu', 10) ,bg = 'red', command = scan)
    b.place(x = 230, y = 150)
    win.mainloop()
```

OUTPUT

#----------

delete.py

#THIS PROGRAM WILL DELETE THE USER DATA

```
import pickle

f=open("ref_name.pkl","rb")
ref_dictt=pickle.load(f)        #ref_dict=ref vs name
f.close()

f=open("ref_embed.pkl","rb")
embed_dictt=pickle.load(f)      #embed_dict- ref  vs embedding
f.close()

print(ref_dictt)
print(embed_dictt.keys())

print("----------------------------------------")

ref_dictt.clear()
embed_dictt.clear()

f=open("ref_name.pkl","wb")
pickle.dump(ref_dictt,f)
f.close()

f=open("ref_embed.pkl","wb")
pickle.dump(embed_dictt,f)
f.close()

f=open("ref_name.pkl","rb")
ref_dictt=pickle.load(f)        #ref_dict=ref vs name
f.close()

f=open("ref_embed.pkl","rb")
embed_dictt=pickle.load(f)      #embed_dict- ref  vs embedding
```

```
f.close()

print(ref_dictt)
print(embed_dictt.keys())
#-----------------------------------------------------------------------------------------------------------------------------------
```

HELPER PROGRAMS

#----------

Register_backend.py

#THIS PROGRAM HELPS IN REGISTER PROCESS

```
import sys
import cv2
import face_recognition
import pickle

def reg(ref_id, name , email):
    Exit = False
    Data = True
    try:
        f=open("ref_name.pkl","rb")
        ref_dictt=pickle.load(f)
        f.close()
    except:
        ref_dictt={}
    ref_dictt[ref_id] = dict()
    ref_dictt[ref_id]['name'] = name
    ref_dictt[ref_id]['email'] = email

    f=open("ref_name.pkl","wb")
    pickle.dump(ref_dictt,f)
    f.close()
    try:
        f=open("ref_embed.pkl","rb")
        embed_dictt=pickle.load(f)
        f.close()
    except:
        embed_dictt={}

    for i in range(5):
        key = cv2. waitKey(1)
        webcam = cv2.VideoCapture(0)
        while True:
            check, frame = webcam.read()
            cv2.imshow("Capturing", frame)
            small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
            rgb_small_frame = small_frame[:, :, ::-1]
```

```
        key = cv2.waitKey(1)

        if key == ord('s') :
            face_locations = face_recognition.face_locations(rgb_small_frame)
            if face_locations != []:
                face_encoding = face_recognition.face_encodings(frame)[0]
                if ref_id in embed_dictt:
                    embed_dictt[ref_id]+=[face_encoding]
                else:
                    embed_dictt[ref_id]=[face_encoding]
                webcam.release()
                cv2.waitKey(1)
                cv2.destroyAllWindows()
                break
        elif key == ord('q'):
            webcam.release()
            cv2.destroyAllWindows()
            Exit = True
            break

    if Exit:
        break

f=open("ref_embed.pkl","wb")
pickle.dump(embed_dictt,f)
f.close()

f=open("ref_name.pkl","rb")
ref_dictt=pickle.load(f)        #ref_dict=ref vs name
f.close()

f=open("ref_embed.pkl","rb")
embed_dictt=pickle.load(f)      #embed_dict- ref  vs embedding
f.close()

return ref_dictt , embed_dictt , Data
#-----------------------------------------------------------------------------------------------------------------------------------
```

Recognition_Backend.py

```
#THIS PROGRAM HELPS IN THE RECOGNITION PROCESS
import face_recognition
import cv2
import numpy as np
import glob
import time
import csv
import pickle
```

```
import statistics as st

def reco():
    f=open("ref_name.pkl","rb")
    ref_dictt=pickle.load(f)        #ref_dict=ref vs name
    f.close()

    f=open("ref_embed.pkl","rb")
    embed_dictt=pickle.load(f)      #embed_dict- ref  vs embedding
    f.close()

    known_face_encodings = []  #encodingd of faces
    known_face_names = []      #ref_id of faces

    for ref_id , embed_list in embed_dictt.items():
        for embed in embed_list:
            known_face_encodings +=[embed]
            known_face_names += [ref_id]

    video_capture = cv2.VideoCapture(0)
    # Initialize some variables
    face_locations = []
    face_encodings = []
    face_names = []
    process_this_frame = True
    Exit = False
    # global face_locations, face_names , face_encodings , process_this_frame , Exit
    ref_dictt['404'] = dict({'name':'Unknown'})
    f=open("ref_name.pkl","wb")
    pickle.dump(ref_dictt,f)
    f.close()
    # print(ref_dictt)
    while True  :
        ret, frame = video_capture.read()
        small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
        rgb_small_frame = small_frame[:, :, ::-1]
        if process_this_frame:
            face_locations = face_recognition.face_locations(rgb_small_frame)
            # print('locations: ' , face_locations)
            # if (face_locations):
            face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)
            face_names = []
            for face_encoding in face_encodings:
                # print("face_encoding" , face_encoding)
                # print(np.array(face_encoding).shape)
                matches = face_recognition.compare_faces(known_face_encodings, face_encoding , tolerance = 0.60)
```

```
        # print("matches: " , matches)
        if (matches):
        # name = "Unknown"
            face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
            # print("dist: " , face_distances)
            for i, face_distance in enumerate(face_distances):
                if any(face_distances < 0.5):
                    # print("<0.5")
                    best_match_index = np.argmin(face_distances)
                    if matches[best_match_index]:
                        name = known_face_names[best_match_index]
                        # print(name)
                    if ref_dictt[name] in list(ref_dictt.values()):
                        # print("Matched")
                        Exit = True
                    face_names.append(name)
                else:
                    # print(">0.5")
                    face_names.append('404')
                    Exit = True
        else:
            cv2.waitKey(delay = 5000)
            face_names.append('404')
            # print("NO DATA FOUND")
            Exit = True
    # else:
    #     cv2.waitKey(delay = 5000)
    #     # name = '404'
    #     face_names.append('404')
    #     # print("NO DATA FOUND")
    #     Exit = True
process_this_frame = not process_this_frame
# print(face_names)
for (top_s, right, bottom, left), name in zip(face_locations, face_names):
    try:
        top_s *= 4
        right *= 4
        bottom *= 4
        left *= 4
        cv2.rectangle(frame, (left, top_s), (right, bottom), (0, 0, 255), 2)
        cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255), cv2.FILLED)
        font = cv2.FONT_HERSHEY_DUPLEX
        cv2.putText(frame, ref_dictt[name]['name'], (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)
    except:
        # print("Not Matched")
        pass
```

```
        font = cv2.FONT_HERSHEY_DUPLEX
        cv2.imshow('Video', frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
        if Exit:
            cv2.waitKey(delay = 3000)
            break
    video_capture.release()
    cv2.destroyAllWindows()
    if face_names:
        return st.mode(face_names)
#-----------------------------------------------------------------------------------------------------------------------------------------
```

E_Mail_backend.py

```
#THIS PROGRAM HELPS IN THE E-MAIL PROCESS
import qrcode
import imghdr
import smtplib
from email.message import EmailMessage
import random as  rd

sent = True

def send_mail_to_all(emails , user_email , name ,subject, body, filename = None):
    Reciever_Email = emails
    Sender_Email = "testfor2factor@gmail.com"
    Password = "cdnhgnzxbtoftchy"
    newMessage = EmailMessage()
    newMessage['From'] = Sender_Email
    newMessage['To'] = Reciever_Email
    newMessage['Subject'] = subject
    newMessage.set_content(f'{name}  {body}')
    if filename:
        try:
            with open(filename, 'rb') as f:
                image_data = f.read()
                image_type = imghdr.what(f.name)
                image_name = f.name

            newMessage.add_attachment(image_data, maintype='image', subtype=image_type, filename='Unknown
Person')

            with smtplib.SMTP_SSL('smtp.gmail.com', 465) as smtp:
                smtp.login(Sender_Email, Password)
                smtp.send_message(newMessage)
                print("MAIL SENT TO USER")
```

```
        except:
            print("MAIL NOT SENT")

    else:
        with smtplib.SMTP_SSL('smtp.gmail.com', 465) as smtp:
            smtp.login(Sender_Email, Password)
            smtp.send_message(newMessage)
            print("Mail Sent To All Others")
        if (body == 'Successfully Registered'):
            subject = 'Registeration Successful'
            send_mail_to_one(name , subject , user_email)
        else:
            if user_email:
                subject = subject
                send_mail_to_one(name ,subject ,user_email)
            else:
                pass

def send_mail_to_one(name,subject, email):
    global sent
    Sender_Email = "testfor2factor@gmail.com"
    Reciever_Email = email
    Password = "cdnhgnzxbtoftchy"
    newMessage = EmailMessage()
    newMessage['Subject'] = subject

    newMessage['From'] = Sender_Email
    newMessage['To'] = Reciever_Email
    newMessage.set_content('Please Use This QR Code During Verification')
    data_orig = str(rd.randint(10000000 , 100000000))
    filename = f'user_{name}.png'
    img = qrcode.make(data_orig)
    img.save(filename)
    try:
        with open(filename, 'rb') as f:
            image_data = f.read()
            image_type = imghdr.what(f.name)
            image_name = f.name

        newMessage.add_attachment(image_data, maintype='image', subtype=image_type, filename='Verificaton
Code')

        with smtplib.SMTP_SSL('smtp.gmail.com', 465) as smtp:
            smtp.login(Sender_Email, Password)
            smtp.send_message(newMessage)
```

```
        print("MAIL SENT TO USER")
        sent =True

    except:
        sent = False
```

#----------

Access_Denied_Handler.py

#THIS PROGRAM HELPS IN THE ACCESS DENIED PROCESS

```
from tkinter import *
from tkinter import messagebox
import time
from Access_Granted_Handler import *
from E_Mail_backend import *

win1 = Tk()
id = ''

def win_2():
    win2 = Tk()
    win2.geometry('400x300')
    win2.title('ERROR....!!!!')
    win2.config(bg='black')
    l1 = Label(win2, text='ACCESS DENIED....!!!' , width=30 , padx=3 , font = ('calibre',15), bg = 'black' , fg =
'red')
    l1.place(relx = 0.5, rely = 0.5 , anchor = 'center')
    win2.mainloop()

closed = False

def on_closing():
    global closed
    if (messagebox.askokcancel("Quit", "Do you want to quit?")):
        closed = True
        win1.destroy()
        return closed

def capture_and_send_mail(emails):
    cam = cv2.VideoCapture(0)
    result, image = cam.read()
    cv2.imwrite('unknown_user.png' , image)
    if result:
        filename = f'unknown_user.png'
        send_mail_to_all(emails , None , 'Unknown Person',  'Action Required!!!',  'Tried To Login' , filename =
filename)
    else:
```

```
        send_mail_to_all(emails , None , 'Unknown Person',  'Action Required!!!',  'Tried To Login' ,  filename =
None)

def open_cam():
    global id
    time.sleep(3)
    id = recog()
    name = user_name(id)
    if (id == '404'):
        emails = all_emails()
        capture_and_send_mail(emails)
        time.sleep(1)
        win1.destroy()
        # win_2()
    else:
        time.sleep(1)
        win1.destroy()

def main():
    global id , closed
    win1.geometry('350x200')
    win1.title('Detection Window')
    win1.config(bg='Lightskyblue')

    l = Label(win1, text='Click Open Camera \n Wait For The Camera To Open \n Look Into The Camera' ,
width=30 , padx=3 , font = ('calibre',15) , bg = 'lightgreen')
    l.place(x=5,y=30)
    win1.protocol("WM_DELETE_WINDOW", on_closing)

    b1 = Button(win1, text='Open Camera' , font=('Ubuntu', 10) ,bg = 'red',  command = open_cam)



    # b1 = Button(win1, text='Open Camera' , font=('Ubuntu', 10) , bg = 'red')
    b1.place(x = 120, y = 140)

    win1.mainloop()
    return id , closed

#-----------------------------------------------------------------------------------------------------------------------------------------
```

Access_Granted_Handler.py

```
#THIS PROGRAM HELPS IN THE ACCESS GRANTED PROCESS
import Recognition_Backend as rt
import numpy as np
import pickle
import qrcode
```

```
import smtplib
import imghdr
from email.message import EmailMessage
import cv2
import random as  rd
import sys
from tkinter import *

sent =False
data_orig = 0

def all_emails():
    emails = []
    f=open("ref_name.pkl","rb")
    ref_dictt=pickle.load(f)        #ref_dict=ref vs name
    f.close()
    for key , value in ref_dictt.items():
        try:
            email = ref_dictt[key]['email']
            emails.append(email)
        except:
            pass

    return emails

def recog():
    name = rt.reco()
    return name

def user_name(name):
    f=open("ref_name.pkl","rb")
    ref_dictt=pickle.load(f)        #ref_dict=ref vs name
    f.close()
    return ref_dictt[name]['name']

def user_mail(name):
    f=open("ref_name.pkl","rb")
    ref_dictt=pickle.load(f)        #ref_dict=ref vs name
    f.close()
    return ref_dictt[name]['email']

def send_mail(name):
    global sent,  data_orig
    f=open("ref_name.pkl","rb")
    ref_dictt=pickle.load(f)        #ref_dict=ref vs name
    f.close()
```

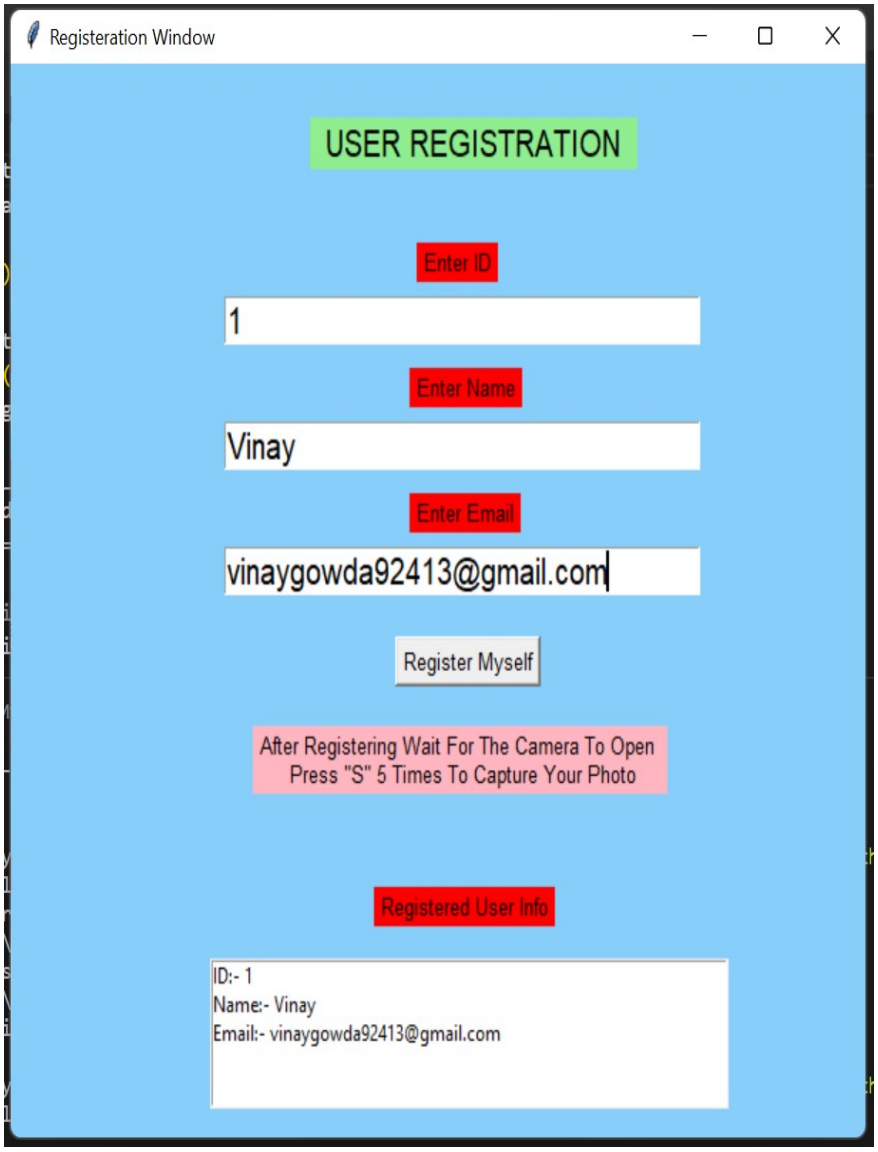
```
    f=open("ref_embed.pkl","rb")
    embed_dictt=pickle.load(f)      #embed_dict- ref  vs embedding
    f.close()

    data_orig = str(rd.randint(10000000 , 100000000))

    # data_orig = ref_dictt[name]['name']
    # output file name
    filename = "site.png"
    # generate qr code
    img = qrcode.make(data_orig)
    # save img to a file
    img.save(filename)

    # if(name == '404'):
    #     sys.exit("Access denied!!!")

    Sender_Email = "testfor2factor@gmail.com"
    Reciever_Email = ref_dictt[str(name)]['email']
    Password = "cdnhgnzxbtoftchy"
    print("Hello ",ref_dictt[str(name)]['name'], " check your email and verify QR code.")

    newMessage = EmailMessage()
    newMessage['Subject'] = "QR code for two-step verification"
    newMessage['From'] = Sender_Email
    newMessage['To'] = Reciever_Email
    newMessage.set_content('Please verify by showing this QR code to camera.')

    try:
        with open(filename, 'rb') as f:
            image_data = f.read()
            image_type = imghdr.what(f.name)
            image_name = f.name

        newMessage.add_attachment(image_data, maintype='image', subtype=image_type, filename='verificaton
code')

        with smtplib.SMTP_SSL('smtp.gmail.com', 465) as smtp:

            smtp.login(Sender_Email, Password)
            smtp.send_message(newMessage)

        print("MAIL SENT")
        sent =True
```

```
    except:
        sent = False

    return sent

Granted = False

def scan_code(id):
    global Granted
    f=open("ref_name.pkl","rb")
    ref_dictt=pickle.load(f)        #ref_dict=ref vs name
    f.close()
    name = ref_dictt[id]['name']
    filename = f'user_{name}.png'

    img = cv2.imread(filename)

    detector = cv2.QRCodeDetector()

    data_orig , bbox , strainght_qrcode = detector.detectAndDecode(img)

    cap = cv2.VideoCapture(0)
    # initialize the cv2 QRCode detector
    detector = cv2.QRCodeDetector()
    count = 0
    while True:
        _, img = cap.read()
        # detect and decode
        data, bbox, _ = detector.detectAndDecode(img)
        # check if there is a QRCode in the image
        if bbox is not None:
            # display the image with lines
            for i in range(len(bbox)):
                cv2.line(img, np.array(bbox[i][0]).astype(int), np.array(bbox[(i+1) % len(bbox)][0]).astype(int),
color=(255, 0, 0), thickness=2)

            if data:
                print("[+] QR Code detected, data:", data)
                count += 1

        # display the result
        cv2.imshow("img", img)
        if (data == data_orig):
            cap.release()
            cv2.destroyAllWindows()
            win2 = Tk()
```

```
            win2.geometry('400x300')
            win2.title('SUCCESSFUL')
            win2.config(bg='black')
            l1 = Label(win2, text='ACCESS GRANTED...!!!' , width=30 , padx=3 , font = ('calibre',15), bg = 'black' ,
fg = 'red')
            l1.place(relx = 0.5, rely = 0.5 , anchor = 'center')
            win2.mainloop()
            print("ACCESS GRANTED")
            cv2.waitKey(delay = 5000)
            Granted = True
            break
        if (count == 5):
            cap.release()
            cv2.destroyAllWindows()
            win2 = Tk()
            win2.geometry('400x300')
            win2.title('ERROR....!!!!')
            win2.config(bg='black')
            l1 = Label(win2, text='ACCESS DENIED...!!!\nWRONG CODE SHOWN' , width=40 , padx=3 , font =
('calibre',15), bg = 'black' , fg = 'red')
            l1.place(relx = 0.5, rely = 0.5 , anchor = 'center')
            win2.mainloop()
            print("ACCESS DENIED")
            cv2.waitKey(delay = 5000)
            break
        if cv2.waitKey(1) == ord("q"):
            break
    cap.release()
    cv2.destroyAllWindows()
    return Granted
#-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

- **RESULTS** :

1. **THE USER LOADING THE DATA**

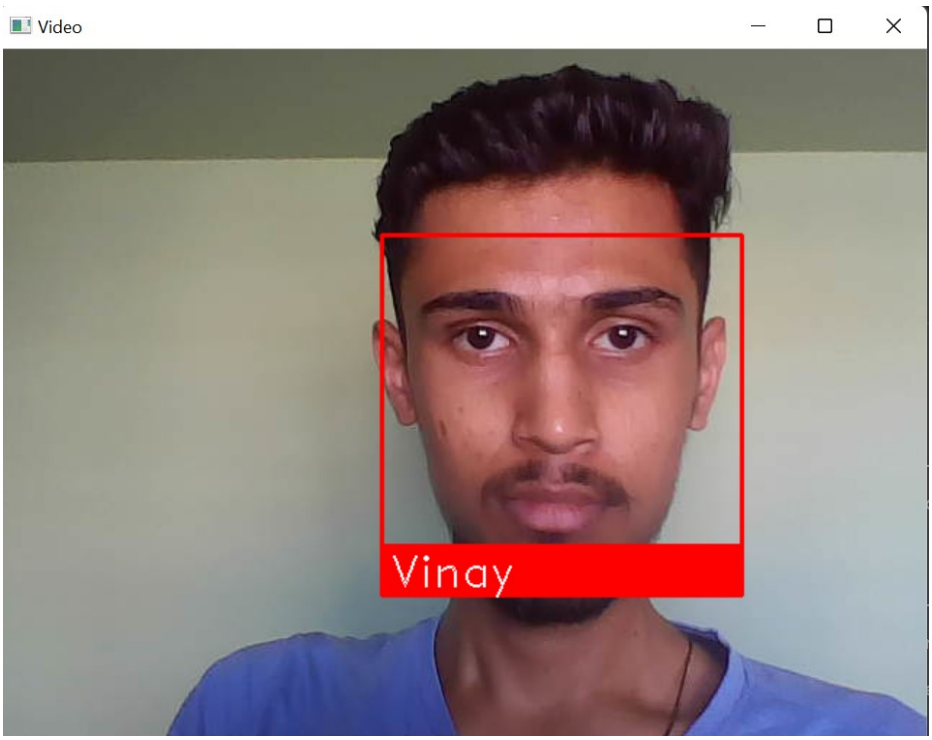


IT TAKES 5 IMAGES TO TRAIN THE FACES.

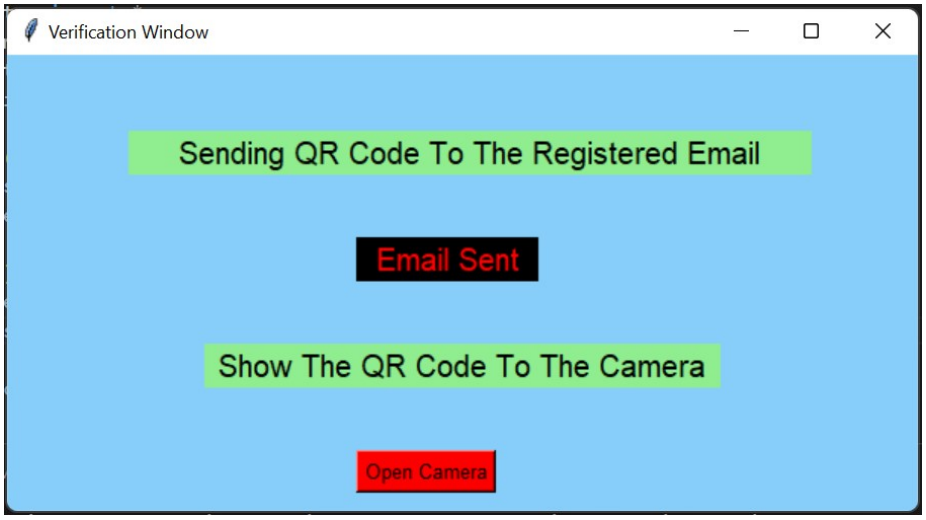
WHEN THE REGISTERED USER COMES IN FRONT OF THE CAMERA

WHEN THE USER CLICKS ON THE OPEN CAMERA THE CAMERA WINDOW POP-UPS AND TAKES IMAGES AND IT STARTS THE RECOGNITION PROCESS.

IF THE PERSON IS A REGISTERED USER



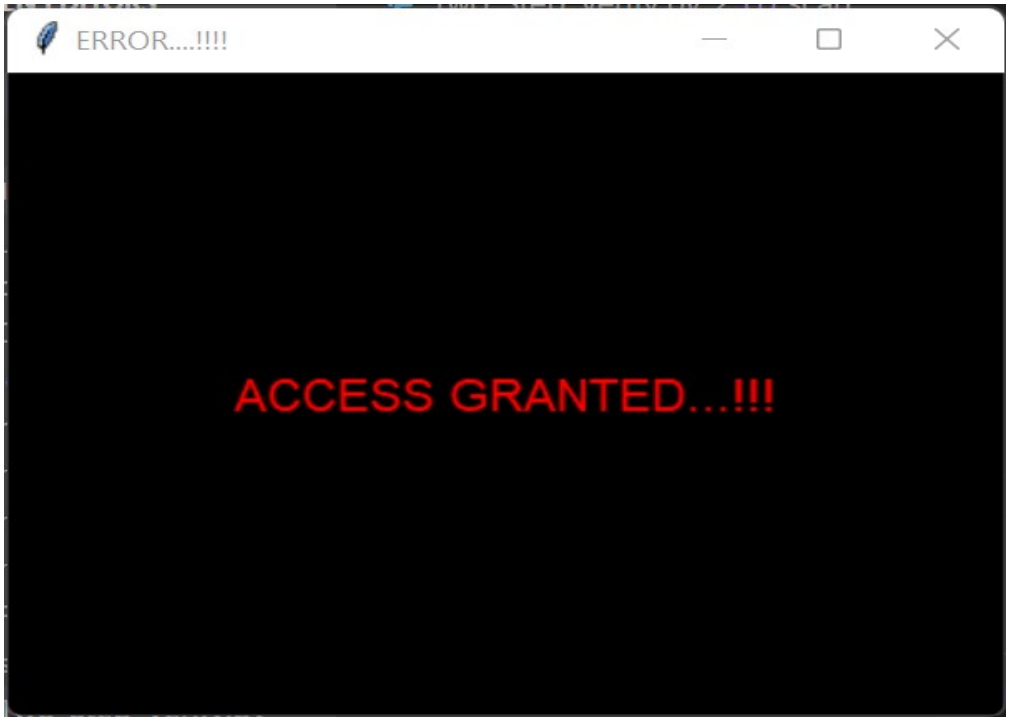
THE USER HAS TO SHOW THE PREVIOUSLY SENT QR CODE FOR TWO-STEP VERIFICATION



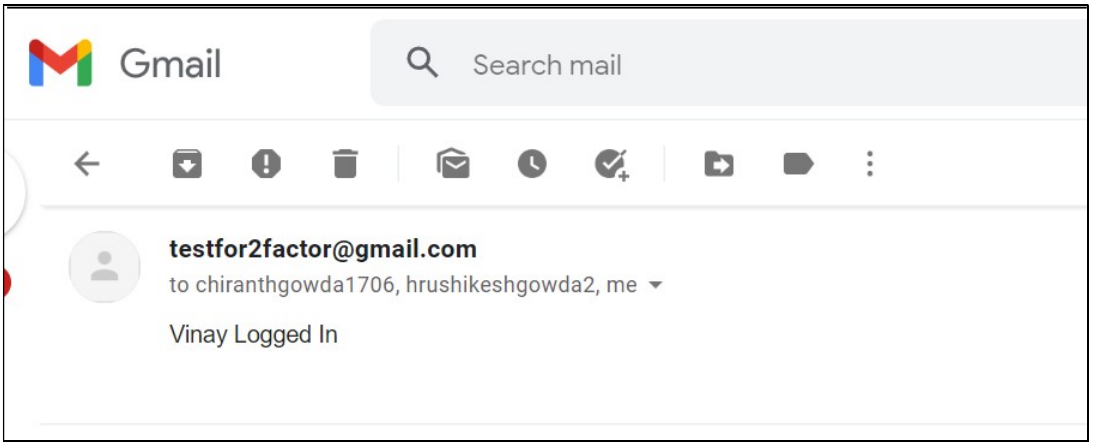
THEN USER NEED TO CLICK ON OPEN CAMERA AND SHOULD SHOW THE QR CODE

TO THE CAMERA

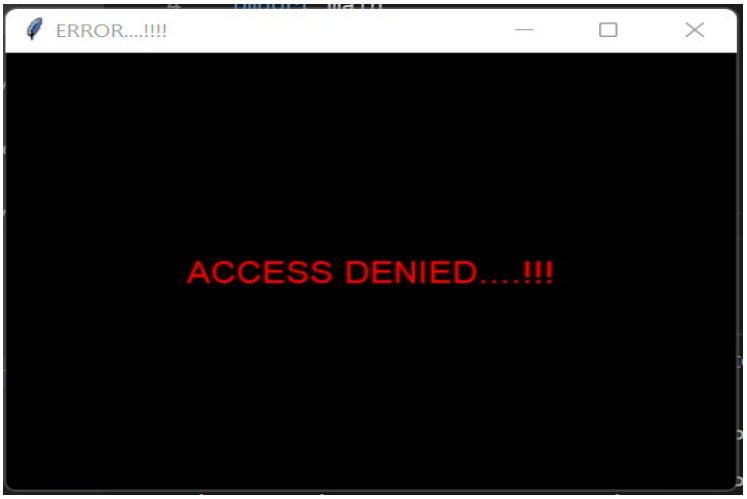
IF OR CODE IS VERIFIED THE ACCESS IS GRANTED



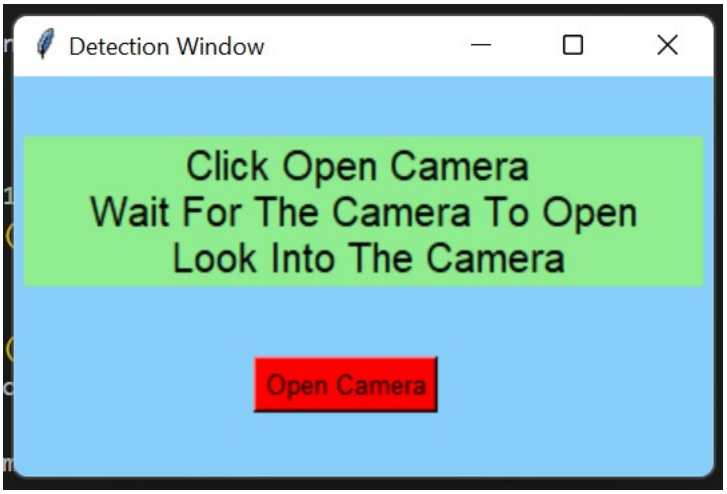
AFTER THE ACCESS IS GRANTED ALL THE REGISTERED USERS WILL GET A UPDATE AS TO WHO LOGGED IN.

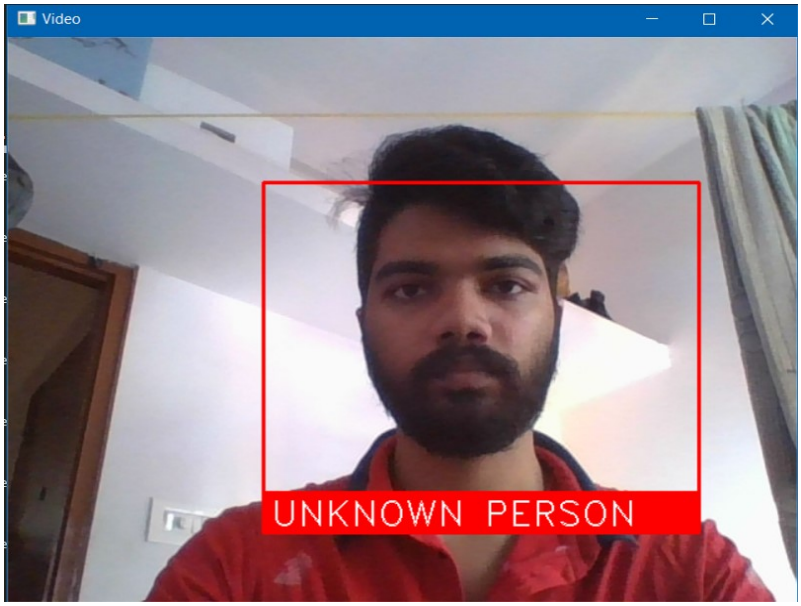


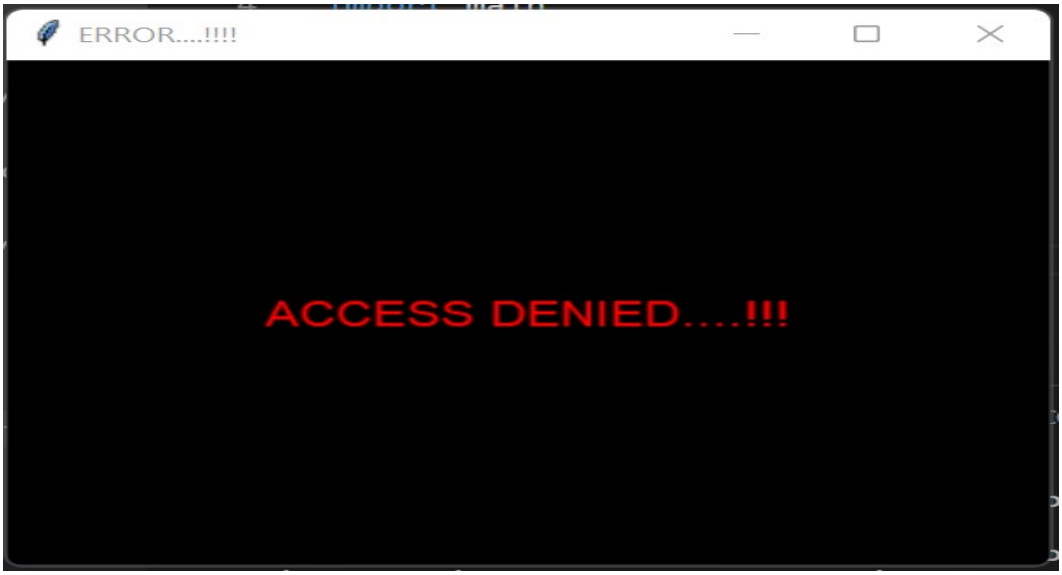
IF THE QR CODE IS WRONG THE ACCESS WILL BE DENIED

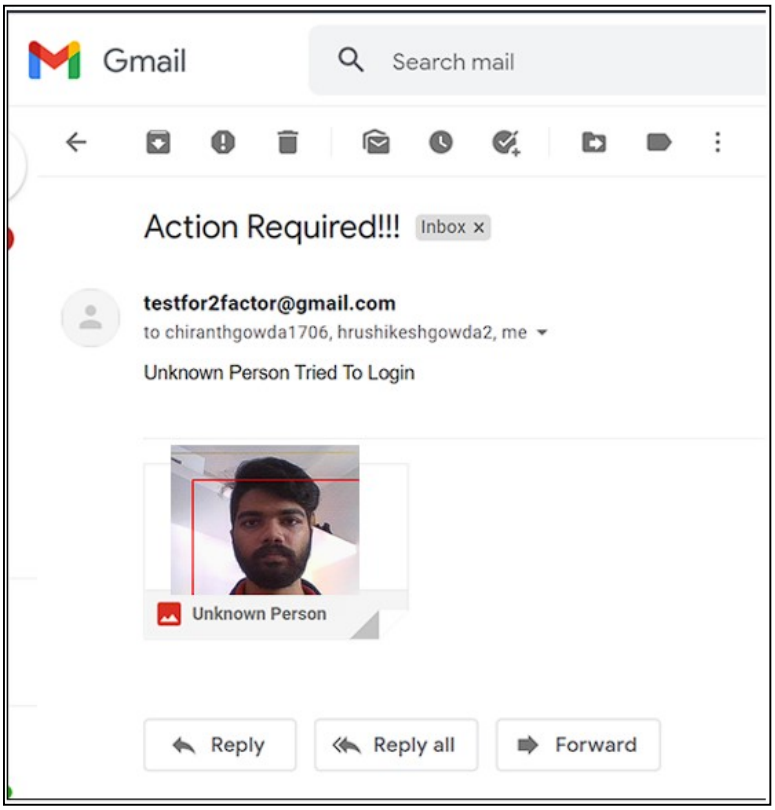


[2] IF THE USER IS NOT REGISTERED AND THEN IF HE TRIES TO ACCESS WILL BE DENIED AND A MAIL WILL BE SENT TO ALL REGISTERED USER AS A ALERT MESSAGE WITH THE UNKOWN PERSON IMAGE WHO TRIED TO LOGIN.


Detection Window
Click Open Camera
Wait For The Camera To Open
Look Into The Camera
Open Camera


Video
UNKNOWN PERSON


ERROR....!!!!
ACCESS DENIED....!!!



- **References :**

[1]. https://www.thepythoncode.com/article/generate-read-qr-code-python

[2]. https://docs.opencv.org/3.4/da/d60/tutorial_face_main.html

[3]. https://www.tutorialspoint.com/python/python_sending_email.htm#:~:text=Python%20provides%20smtplib%20module%2C%20which,SMTP%20or%20ESMTP%20listener%20daemon.&text=host%20%E2%88%92%20This%20is%20the%20host,domain%20name%20like%20tutorialspoint.com.

[4]. https://realpython.com/face-recognition-with-python/

[5]. https://pypi.org/project/face-recognition/

[6]. https://towardsdatascience.com/building-a-face-recognizer-in-python-7fd6630c6340

[7]. https://www.javatpoint.com/generate-a-qr-code-using-python

[8]. https://www.geeksforgeeks.org/send-mail-gmail-account-using-python/