SECURITY USING FACE DETECTION

Submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

SUBMITTED BY:

HITESH KUMAR :19BCS1377

RISHABH SHARMA :19BCS1394

AMIT KUMAR :19BCS1370

PUSHP RAJ :19BCS1372

SUBMITTED TO :

PROJECT TEACHER: DR. VIKAS JINDAL

PROJECT MENTOR: SUGANDHI MIDHA
(ECODE:8219)

(Mentor Signature)

Implementation (Python Code) :

```
import cv2
import numpy as np
import face_recognition
import tkinter as tk
from datetime import datetime
import pickle
from PIL import Image, ImageTk
import tkinter.messagebox as tmsg


class MainWindow:
    def __init__(self):
        self.vs = cv2.VideoCapture(0)
        self.frame = None
        self.current_frame = None
        self.all_face_encodings = {}
        self.facenames = []
        self.faceIds = []
        self.name_for_face = None
        self.facess = []

        '''Root Window'''
        self.root = tk.Tk()
        self.root.geometry("700x500")
        self.root.title("Security using face detection")
        self.root.configure(background="#d9d9d9")
        self.root.configure(highlightbackground="#d9d9d9")
        self.root.configure(highlightcolor="black")
        self.root.wm_protocol("WM_DELETE_WINDOW", self.destructor)
```

```
        '''frame_1, frame_2'''

        self.root.columnconfigure(0, weight=2)

        self.root.columnconfigure(1, weight=8)

        self.frame_1 = tk.Frame(self.root)

        self.frame_1.grid(column=0, sticky="nsew")

        self.frame_1.place(relheight=1, relwidth=1, relx=0, rely=0)

        self.frame_2 = tk.Frame(self.root)

        self.frame_2.grid(column=1, sticky="nsew")

        self.frame_2.place(relheight=1, relwidth=1, relx=0.2, rely=0)


        self.panel = tk.Label(self.frame_2)

        self.panel.pack(padx=10, pady=10, fill=tk.BOTH)


        self.load_encoding()


        ''' new_face register, authenticate and remove buttons'''

        new_face_btn = tk.Button(self.frame_1, text="Register New Face",
command=lambda: self.registering())

        new_face_btn.grid(sticky='nsew', pady=20)

        authenticate_btn = tk.Button(self.frame_1, text="Authenticate Face",
command=self.authentication)

        authenticate_btn.grid(sticky='nsew', pady=20)

        remove_btn = tk.Button(self.frame_1, text="Remove Face",
command=lambda: self.remove_face())

        remove_btn.grid(sticky='nsew', pady=20)


        ''' start a self.video_loop that constantly read video frame'''

        self.video_loop()
```

```
    def findEncodings(self, imag):
        try:
            imag = cv2.cvtColor(imag, cv2.COLOR_BGR2RGB)
            faces = face_recognition.face_locations(imag)
            encode = face_recognition.face_encodings(imag, faces)[0]
            return encode
        except:
            return []


    def load_encoding(self):
        try:  # reading face encode(dictionary) key:name of person ,value:
face encoding file
            with open('dataset_faces.dat', 'rb') as f:
                self.all_face_encodings = pickle.load(f)
                f.close()
        except:
            self.all_face_encodings = {}
        if len(self.all_face_encodings) == 0:
            # all_face_encodings = {}
            self.facenames = []
            self.faceIds = []
        else:
            self.facenames = list(self.all_face_encodings.keys())
            self.faceIds = list(self.all_face_encodings.values())


    def video_loop(self):
        """ Get frame from the video stream and show it in Tkinter """
        ok, self.frame = self.vs.read()  # read frame from video stream
        if ok:  # frame captured without any errors
            self.facess = face_recognition.face_locations(self.frame)
```

```
            for i in range(len(self.facess)):
                self.img = cv2.rectangle(self.frame, (self.facess[i][3],
self.facess[i][0]),
                                         (self.facess[i][1],
self.facess[i][2]), (0, 255, 0), 2)

            cv2image = cv2.cvtColor(self.frame, cv2.COLOR_BGR2RGB)
            self.current_frame = Image.fromarray(cv2image)
            imgtk = ImageTk.PhotoImage(image=self.current_frame)
            self.panel.imgtk = imgtk
            self.panel.config(image=imgtk)
        self.root.after(30, self.video_loop)


    def registering(self):
        def validate_name():
            if self.name_for_face.get() in self.facenames:
                tmsg.showinfo("Error", self.name_for_face.get() + " already
in use try different name!!!")
            else:
                name_window.destroy()
                enc = self.findEncodings(self.frame)
                if len(enc) == 0:
                    tmsg.showinfo("Message", "No face found....failed...")
                    return
                self.all_face_encodings[self.name_for_face.get()] = enc
                with open('dataset_faces.dat', 'wb') as f:
                    pickle.dump(self.all_face_encodings, f)
                    tmsg.showinfo("Message", self.name_for_face.get() + "
Registered")
                self.load_encoding()

        def cancel_face_name_input():
```

```
            name_window.destroy()


        faces = self.facess

        if len(faces) != 1:

            tmsg.showinfo("message", "Face not Detected Try again!!!")

            return False

        else:

            # name input window

            name_window = tk.Toplevel()

            name_window.geometry('300x150')

            name_window.title('Register Face')


            face_label = tk.Label(name_window, text="Name",
relief=tk.GROOVE).grid(row=0, column=0, padx=15, pady=15)

            self.name_for_face = tk.StringVar()

            face_label_entry = tk.Entry(name_window,
textvariable=self.name_for_face).grid(row=0, column=1, pady=15)


            ok_btn = tk.Button(name_window, text="Continue",
command=validate_name).grid(row=1, column=0)

            cancel_btn = tk.Button(name_window, text="Cancel",
command=cancel_face_name_input).grid(row=1, column=1)


    def marktime(self, name):

        with open('logindetails.csv', 'a+') as f:

            myDataList = f.readlines()

            nameList = []

            for line in myDataList:

                entry = line.split(',')

                nameList.append(entry[0])

            if name not in nameList:

                now = datetime.now()
```

```
                dtString = now.strftime('%H:%M:%S')
                f.writelines(f'\n{name},{dtString}')

    def authentication(self):
        if len(self.facess) != 1:
            tmsg.showinfo("message", " Face Not Detected Try Again!!!")
        self.current_frame  = cv2.cvtColor(self.frame, cv2.COLOR_BGR2RGB)
        facesCurFrame = face_recognition.face_locations(self.current_frame)
        if len(facesCurFrame) != 0:
            encodesCurFrame =
face_recognition.face_encodings(self.current_frame, facesCurFrame)
            for encodeFace, faceLoc in zip(encodesCurFrame, facesCurFrame):
                matches = face_recognition.compare_faces(self.faceIds,
encodeFace)
                self.faceDis = face_recognition.face_distance(self.faceIds,
encodeFace)
                if len(self.faceDis) != 0:
                    matchIndex = np.argmin(self.faceDis)
                    if matches[matchIndex]:
                        name = self.facenames[matchIndex]
                        self.marktime(name)
                        tmsg.showinfo("message", name + " Authenticated")
                        return
                else:
                    tmsg.showinfo("message", "Unregistered Face")
                    name = "unauthorized"
                    self.marktime(name)

    def remove_face(self):

        def remove():
```

```
            if self.name_for_face.get() not in self.facenames:
                tmsg.showinfo("Error", "Not In database Try different
name!!!")

            else:
                remove_window.destroy()
                try:
                    self.all_face_encodings.pop(self.name_for_face.get())
                    with open('dataset_faces.dat', 'wb') as f:
                        pickle.dump(self.all_face_encodings, f)
                        tmsg.showinfo("message", "Face deleted")
                    self.load_encoding()

                except FileExistsError:
                    tmsg.showinfo("Error", "some error occured Try
again!!!")

        def cancel_face_name_input():
            remove_window.destroy()

        def set_face_name(event):
            widget = event.widget
            selection = widget.curselection()
            value = widget.get(selection[0])
            self.name_for_face.set(value)

        remove_window = tk.Toplevel()
        remove_window.geometry('300x400')
        remove_window.title('Delete Face')

        remove_window.rowconfigure(0, weight=3)
```

```
        remove_window.rowconfigure(1, weight=7)

        remove_window_frame1 = tk.Frame(remove_window)

        remove_window_frame2 = tk.Frame(remove_window)


        face_label = tk.Label(remove_window_frame1, text="Name",
relief=tk.GROOVE)

        face_label.grid(sticky='ew', row=0, column=0, padx=30, pady=20)

        self.name_for_face = tk.StringVar()

        face_label_entry = tk.Entry(remove_window_frame1,
textvariable=self.name_for_face, width=15)

        face_label_entry.grid(sticky='ew', row=0, column=2)


        ok_btn = tk.Button(remove_window_frame1, text="Continue",
command=remove)

        ok_btn.grid(sticky='ew', padx=30, row=1, column=0)

        cancel_btn = tk.Button(remove_window_frame1, text="Cancel",
command=cancel_face_name_input)

        cancel_btn.grid(sticky='ew', row=1, column=2, columnspan=2)

        remove_window_frame1.grid(row=0, sticky="nsew")

        remove_window_frame1.place(relheight=0.30, relwidth=1, relx=0,
rely=0)


        tk.Label(remove_window_frame2, text="Registered Faces").pack()

        sb = tk.Scrollbar(remove_window_frame2)

        lb = tk.Listbox(remove_window_frame2, font=('calibre', 10),
yscrollcommand=sb.set, height=15, width=10)

        for i in self.facenames:

            lb.insert(tk.END, i)

        lb.bind("<<ListboxSelect>>", set_face_name)

        sb.configure(command=lb.yview, orient=tk.VERTICAL)

        sb.pack(side=tk.RIGHT, fill=tk.Y)

        lb.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

        remove_window_frame2.grid(row=1, sticky="nsew")
```

```
        remove_window_frame2.place(relheight=0.70, relwidth=1, relx=0,
rely=0.30)


    def destructor(self):

        """ Destroy the root object and release all resources """

        print("[INFO] closing...")

        self.root.destroy()

        self.vs.release()  # release web camera

        cv2.destroyAllWindows()  # it is not mandatory in this application



# start the app

print("[INFO] starting...")

pba = MainWindow()

pba.root.mainloop()

##########################################################################
########################################
```

Modules used:

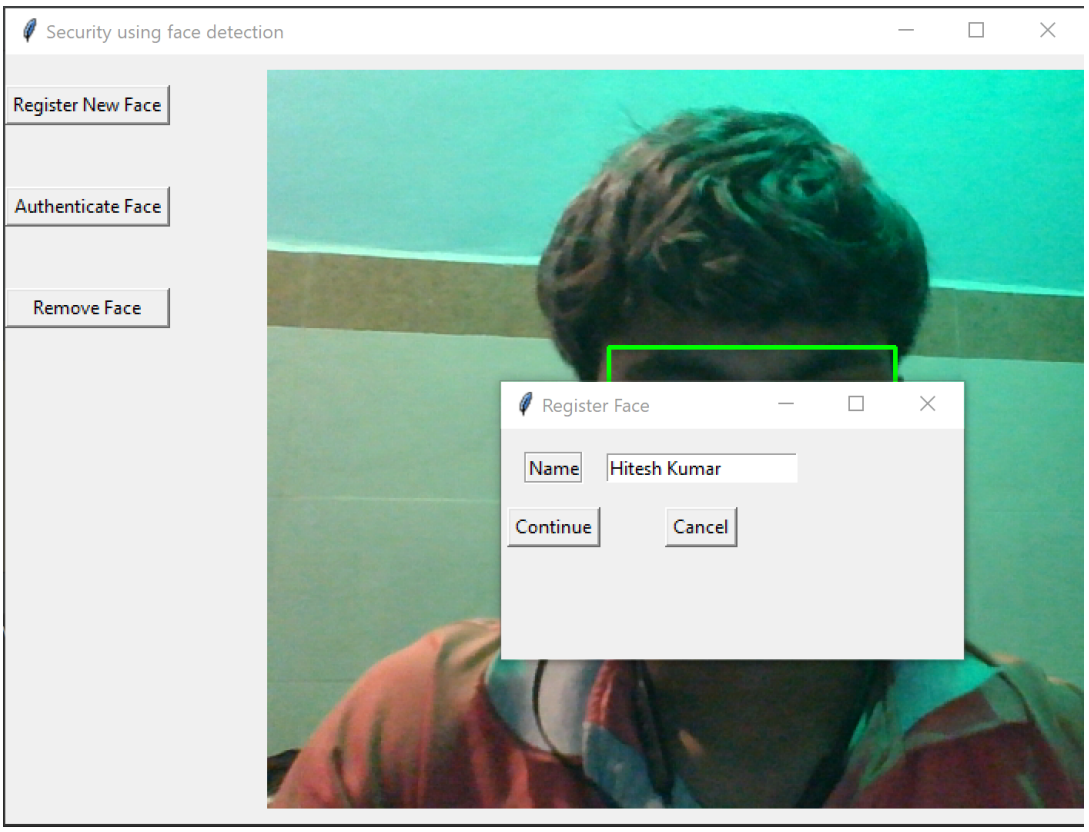
1. **OpenCV:** OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such as Numpy which is a highly optimized library for numerical operations, then the number of weapons increases in your Arsenal i.e whatever operations one can do in Numpy can be combined with OpenCV.
2. **Face_Recognition:**Built using dlib‘s state-of-the-art face recognition built with deep learning. The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark.This also provides a simple face_recognition

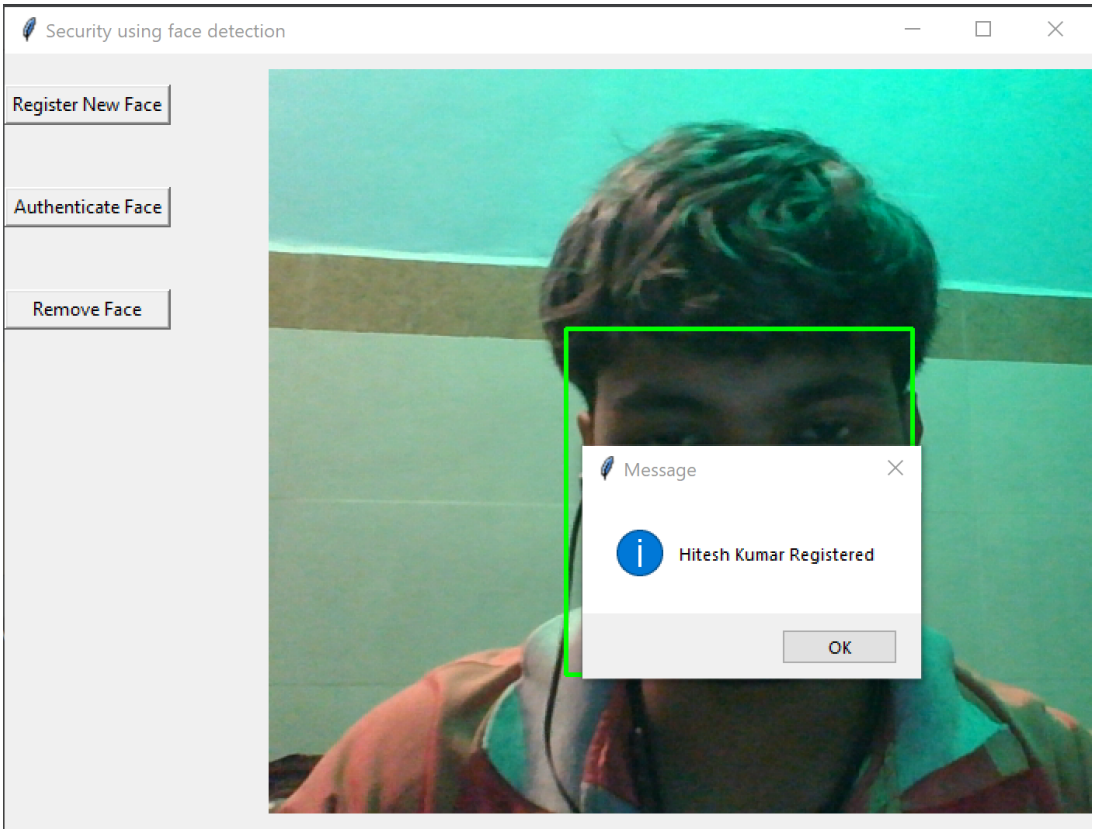
command line tool that lets you do face recognition on a folder of images from the command line!

3. **Tkinter:**Tk/Tcl has long been an integral part of Python. It provides a robust and platform independent windowing toolkit, that is available to Python programmers using the tkinter packageThe tkinter package is a thin object-oriented layer on top of Tcl/Tk. To use tkinter, you don't need to write Tcl code, but you will need to consult the Tk documentation, and occasionally the Tcl documentation. tkinter is a set of wrappers that implement the Tk widgets as Python classes.
4. **PIL:**The Python Imaging Library adds image processing capabilities to your Python interpreter.This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities.The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool.
5. **Others:** We have also used some very basic predefined modules like numpy, pickle and datetime.

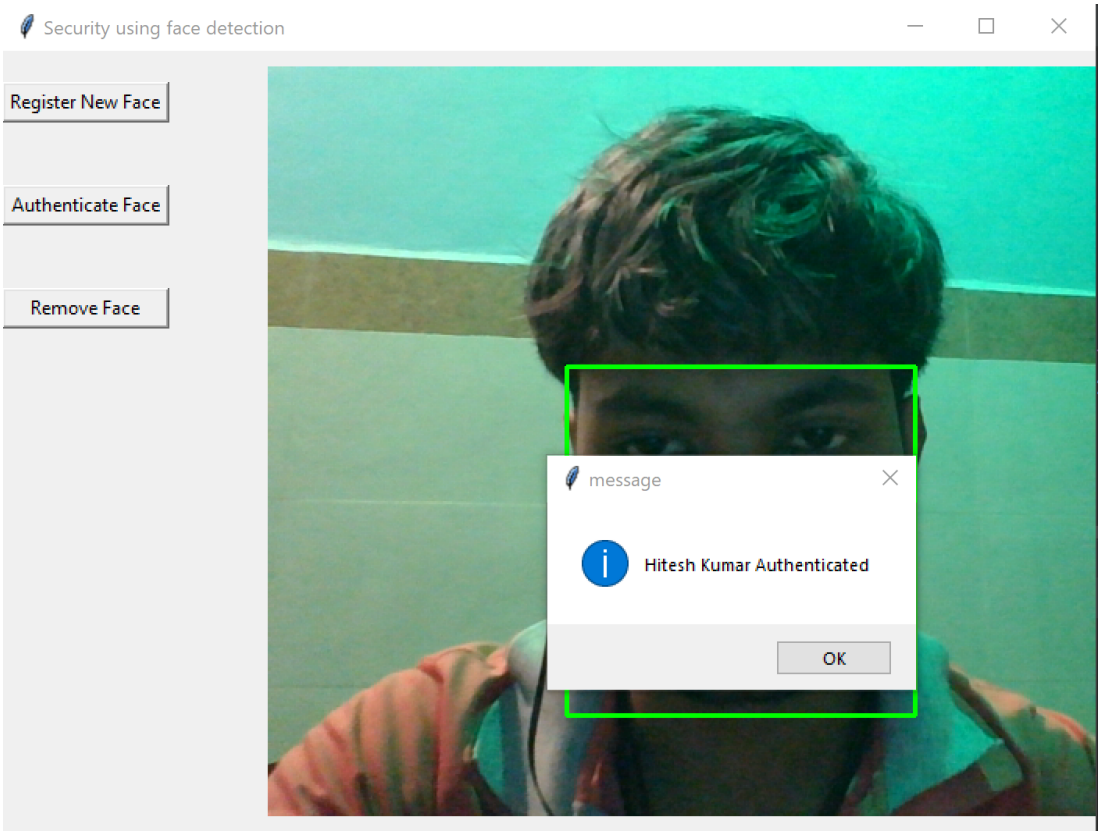
Output Screenshots and Applications of Program:

Register New Face :

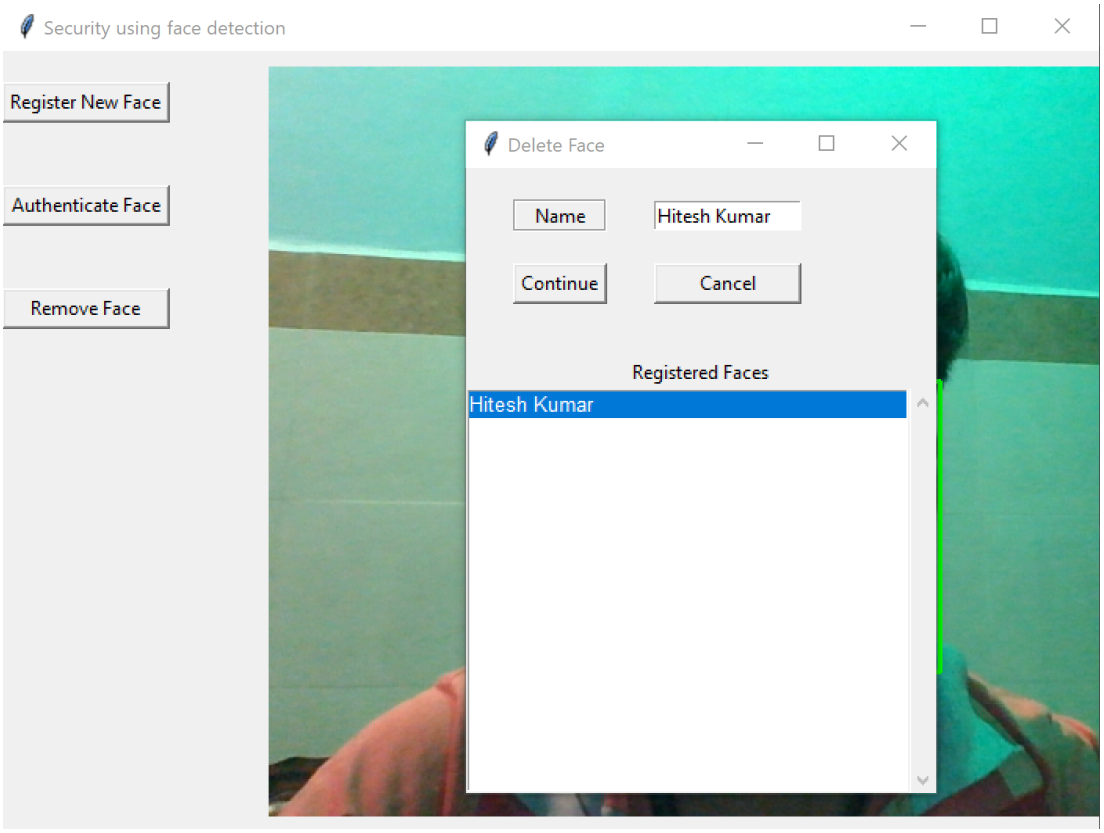


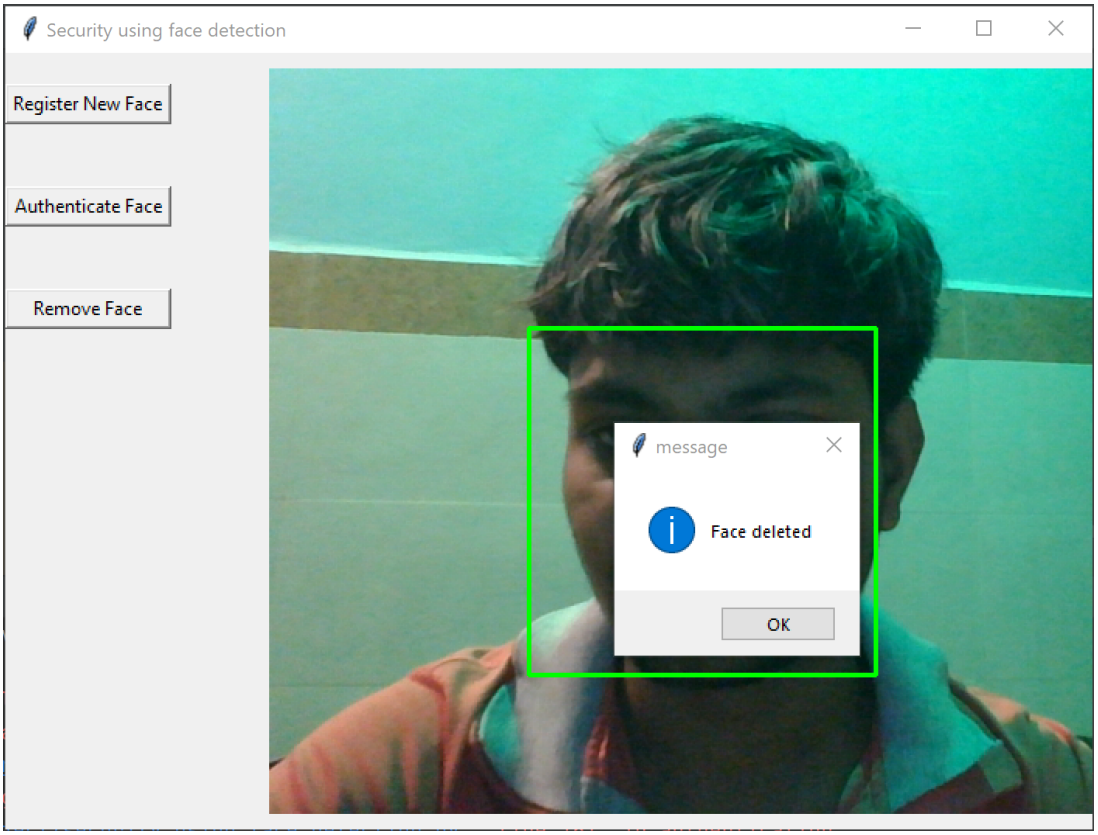


Authenticate Face :

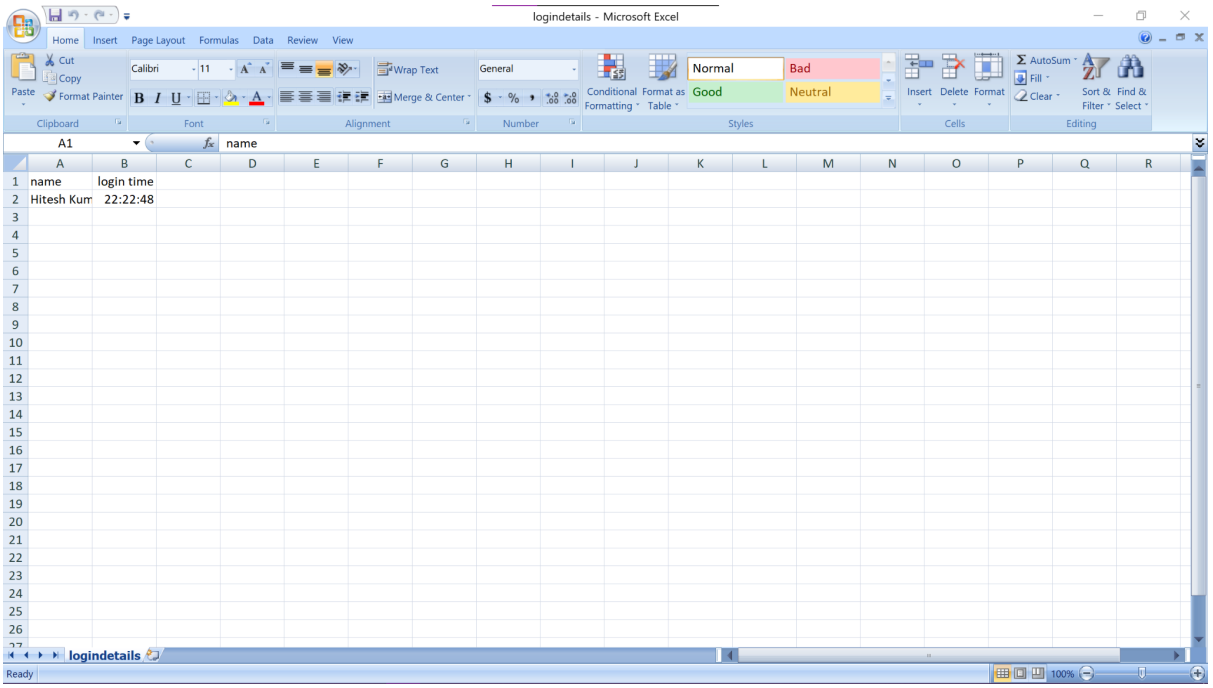


Delete Face :





Login_details (csv file):



Applications:

1. In terms of security, it can be used in public cameras by instantly detecting and matching faces to watchlists containing persons of concern and missing persons in real time.
2. Can be used in companies as a security tool or as an attendance manager which records employee details along with time of authentication.
3. By bringing facial recognition to cameras, airports are able to identify security threats much earlier than before, as well as opportunities for improving operations that previously required much greater effort and cost.
4. The door access control is implemented by using face recognition technology, which grants access to only authorized people to enter that area.
5. Face recognition makes it easier to track down burglars, thieves and trespassers. The technology is capable of analyzing the feed private and public CCTV camera networks.
6. Applications are not limited to physical security but encompass cybersecurity as well. Companies can use face recognition technology as a substitute for passwords to access computers.
7. While Banks have become sophisticated at using one time passwords to access accounts or authorize transfers, there is still room for improvement. One possible solution is biometric identification via facial recognition technology.
8. Instead of those pesky one-time passwords, you could authorize transactions by looking at your smartphone or computer. Biometric online banking is another of the benefits of face recognition.
9. With face recognition, there are no passwords that hackers could compromise. Even if hackers stole your photo database, it would be of little use, as “liveness detection,” prevents using them for impersonation purposes.
10. Forget long checkout lines with slow payments, new “face pay” technologies can shorten them.

Team Work:

We have divided the work into two sections : Coding and Testing.
Amit and Hitesh worked upon functions and GUI which includes defining all the functions and creating a suitable GUI for it (using tkinter).
Rishabh and Pushp worked upon code formatting and testing/debugging.

How this project enhanced our knowledge?

This project has greatly boosted our concepts of python programming and logic building.

Now, we have enough experience of problem solving and implementing things practically. We've also learnt about some very famous python modules like Tkinter for GUI, OpenCv and PIL for image processing, Pickle for file handling and numpy for operating on large data sets(multi-dimensional arrays).

We've gained a good amount of knowledge about ML basic algorithms like regression and clustering.

**********END OF PHASE-II**********