Task: I was tasked to build a desktop app using Python that would recognize faces through a webcam and tell that person's name and home district from the database. Also, an admin panel will manually train a person's photo and name, which will be saved in the database.

So here it goes. The task has 2 python files.

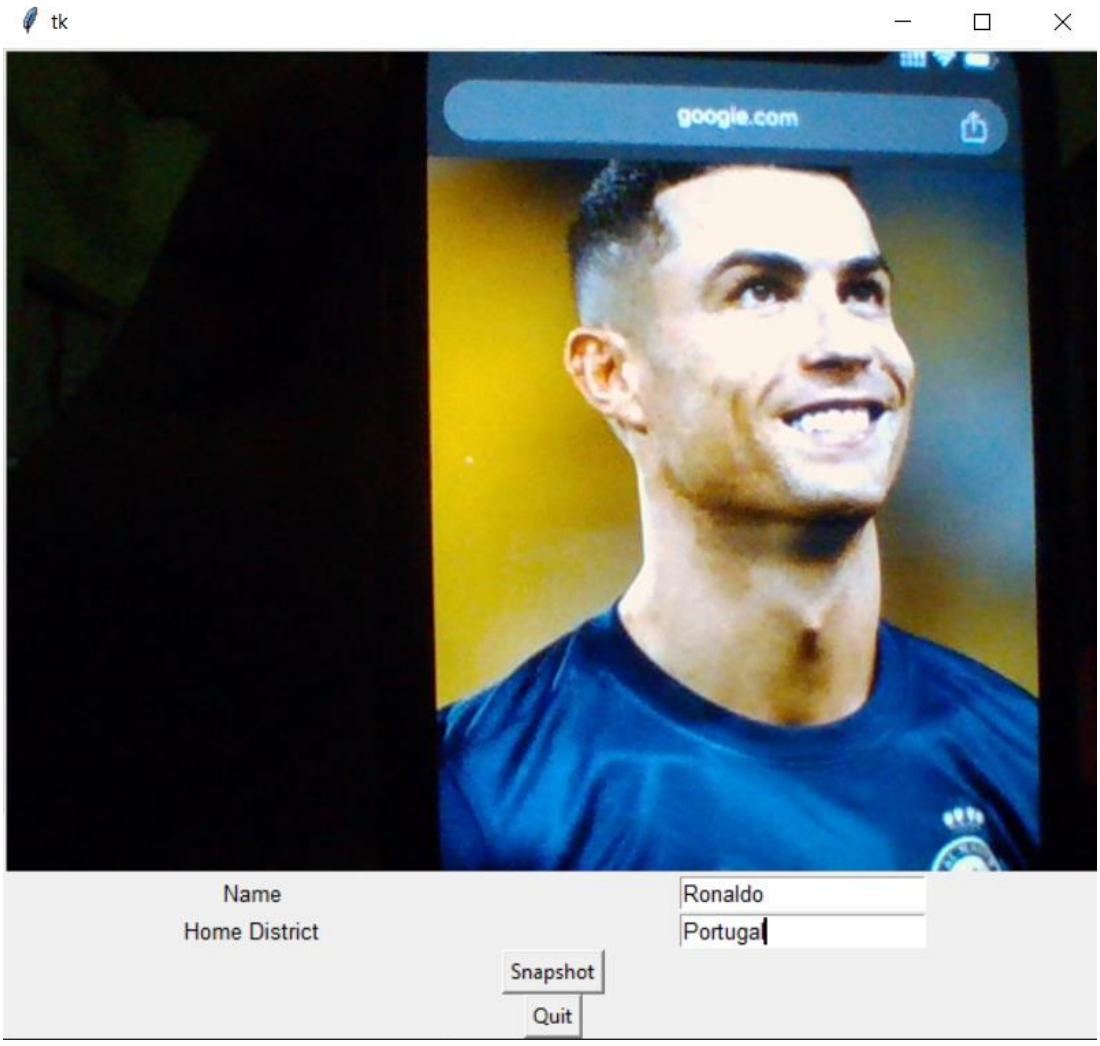
1. Dataload.py (for training the data)
2. Main.py (for facial recognition)

One additional python file is there just for checking the database tables.

myData is the folder to save the trained photos.

1. dataload.py

Interface:

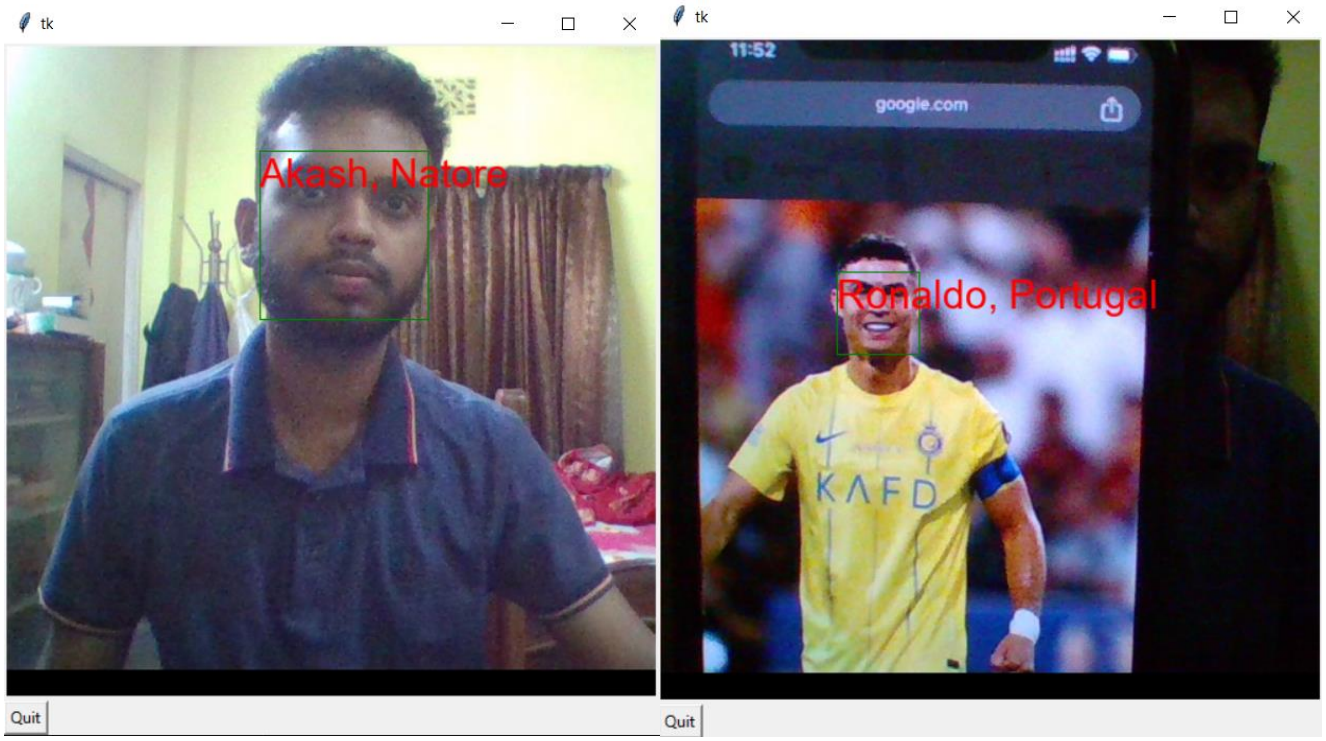


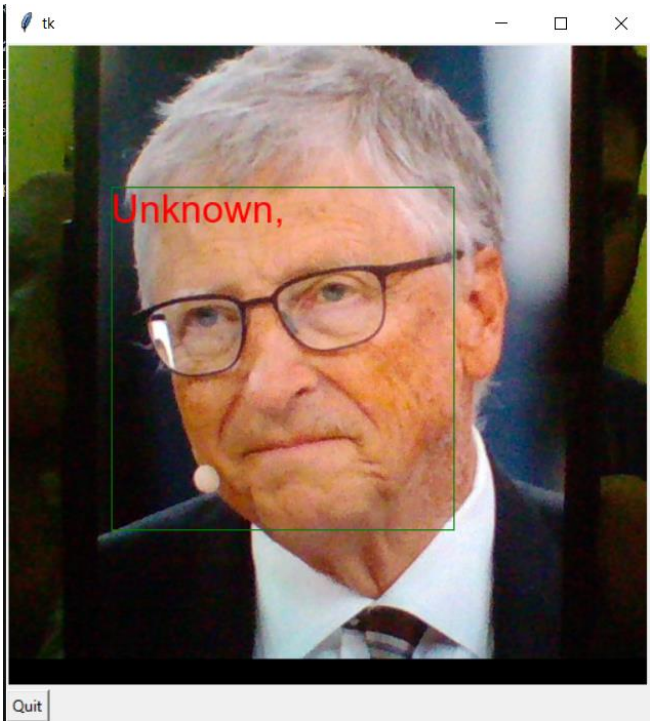
Description:

The script uses the Tkinter library to create a GUI application for face recognition. The application will continuously show video frames from the webcam. It has 2 input fields, one is name, another one is district. When the 'Snapshot' button is clicked, it will take a snapshot, save the cropped face image, and store the name, photo id, and district in the database. When the 'Quit' button is clicked, it will close the application.

2. main.py

interface:





Description:

The application will continuously show video frames from the webcam. When a face is recognized in the video frames, it will display the name and district of the person. If it does not recognize then it will show Unknown. The 'Quit' button can be used to close the application.

Checking Database Table information:

main.py sql.py

```python
import sqlite3

# Connect to SQLite database
conn = sqlite3.connect('face_recognition.db')
c = conn.cursor()

# Execute the command to see all tables
c.execute("SELECT * FROM faces")
tables = c.fetchall()

# Print all tables
for table in tables:
    print(table)

conn.close()
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Akash\Desktop\Taskv2> & C:/Users/Akash/miniconda3/envs/face/python.exe c:/Users/Akash/Desktop/Taskv2/dataload.py
PS C:\Users\Akash\Desktop\Taskv2> & C:/Users/Akash/miniconda3/envs/face/python.exe c:/Users/Akash/Desktop/Taskv2/main.py
PS C:\Users\Akash\Desktop\Taskv2> & C:/Users/Akash/miniconda3/envs/face/python.exe c:/Users/Akash/Desktop/Taskv2/sql.py
('Akash', 'myData/Akash.jpg', 'Natore')
('Jalal', 'myData/Jalal.jpg', 'Singra')
('Ronaldo', 'myData/Ronaldo.jpg', 'Portugal')
PS C:\Users\Akash\Desktop\Taskv2>
```

Folder Structure:

Taskv2

Name	Date modified	Type	Size
myData	5/21/2024 11:47 PM	File folder	
dataload.py	5/21/2024 11:16 PM	Python File	3 KB
face_recognition.db	5/21/2024 11:47 PM	Data Base File	8 KB
main.py	5/21/2024 11:27 PM	Python File	3 KB
sql.py	5/21/2024 11:24 PM	Python File	1 KB

myData folder:

Taskv2 > myData


Akash.jpg


Jalal.jpg


Ronaldo.jpg

Code:

Dataload.py

```
from tkinter import *
from PIL import Image, ImageTk, ImageDraw, ImageFont
import cv2
import face_recognition
import sqlite3

# Initialize the video capture object
cap = cv2.VideoCapture(0)

# Connect to SQLite database
conn = sqlite3.connect('face_recognition.db')
c = conn.cursor()

# Create table if it doesn't exist
c.execute('''CREATE TABLE IF NOT EXISTS faces
             (name TEXT, photo_id TEXT, district TEXT)''')

# Function to take a snapshot and save it
def snapshot():
    cv2image = cv2.cvtColor(cap.read()[1], cv2.COLOR_BGR2RGB)
    img = Image.fromarray(cv2image)
    face_locations = face_recognition.face_locations(cv2image)
    for face_location in face_locations:
        top, right, bottom, left = face_location
        # Crop the face from the image
        im1 = img.crop((left, top, right, bottom))
        name = name_var.get()
        district = district_var.get()  # Get home district
        photo_id = 'myData/' + name + '.jpg'
        # Save the cropped face with the given name
        im1.save(photo_id)
        # Insert data into database
        c.execute("INSERT INTO faces VALUES (?, ?, ?)", (name, photo_id, district))
        conn.commit()

# Function to continuously show frames from the camera
def show_frames():
    cv2image = cv2.cvtColor(cap.read()[1], cv2.COLOR_BGR2RGB)
    img = Image.fromarray(cv2image)
    imgtk = ImageTk.PhotoImage(image=img)
    label.imgtk = imgtk
    label.configure(image=imgtk)
    label.after(10, show_frames)  # Repeat after an interval

# Function to quit the application
def quitapp():
```

```
    cap.release()  # Release the camera
    win.destroy()  # Destroy the window
    conn.close()  # Close the database connection

# Create the main Tkinter window
win = Tk()
name_var = StringVar()
district_var = StringVar()  # Variable for home district

# Create a Label to display the video frames
label = Label(win)
label.grid(row=0, column=0, columnspan=2)  # Span across two columns

# Label for entering the name
name_label = Label(win, text="Name", font=('calibre', 10, 'normal'))
name_label.grid(row=1, column=0)

# Label for entering the home district
district_label = Label(win, text="Home District", font=('calibre', 10, 'normal'))
district_label.grid(row=2, column=0)

# Entry for entering the name
name_entry = Entry(win, textvariable=name_var, font=('calibre', 10, 'normal'))
name_entry.grid(row=1, column=1)

# Entry for entering the home district
district_entry = Entry(win, textvariable=district_var, font=('calibre', 10, 'normal'))
district_entry.grid(row=2, column=1)

# Button to take a snapshot
snap_btn = Button(win, text='Snapshot', command=snapshot)
snap_btn.grid(row=3, column=0, columnspan=2)  # Span across two columns

# Button to quit the application
quit_btn = Button(win, text='Quit', command=quitapp)
quit_btn.grid(row=4, column=0, columnspan=2)  # Span across two columns

# Show the video frames
show_frames()

# Start the Tkinter event loop
win.mainloop()
```

Main.py

```
import tkinter as tk
from tkinter import *
```

```
from PIL import Image, ImageTk, ImageDraw, ImageFont
import cv2
import face_recognition
import sqlite3
import numpy as np

# Connect to SQLite database
conn = sqlite3.connect('face_recognition.db')
c = conn.cursor()

# Fetch data from the database
c.execute("SELECT * FROM faces")
rows = c.fetchall()

names = []
images = []
districts = []  # Added for storing districts

for row in rows:
    name, photo_id, district = row
    names.append(name)
    images.append(cv2.imread(photo_id))
    districts.append(district)

def encoding1(images):
    encode = []

    for img in images:
        unk_encoding = face_recognition.face_encodings(img)[0]
        encode.append(unk_encoding)
    return encode

encodelist = encoding1(images)

# Initialize the video capture object
cap = cv2.VideoCapture(0)

def update_image():
    '''executed frequently, it updates frame/image on canvas'''

    # read one frame (and "return" status)
    ret, frame = cap.read()
    frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    frame1 = cv2.cvtColor(cap.read()[1], cv2.COLOR_BGR2RGB)

    if ret is None:
        print("Can't read from camera")
    else:
        image = Image.fromarray(frame)
```

```
        face_locations = face_recognition.face_locations(frame)
        curframe_encoding = face_recognition.face_encodings(frame1, face_locations)

        for encodeface, facelocation in zip(curframe_encoding, face_locations):
            results = face_recognition.compare_faces(encodelist, encodeface)
            distance = face_recognition.face_distance(encodelist, encodeface)
            match_index = np.argmin(distance)

            # Set a threshold for the face distance
            threshold = 0.6
            if distance[match_index] < threshold:
                name = names[match_index]
                district = districts[match_index]  # Retrieve district based on index
            else:
                name = "Unknown"
                district = ""

            top, right, bottom, left = facelocation

            draw = ImageDraw.Draw(image)
            draw.rectangle([left, top, right, bottom], outline="green")
            draw.text((left, top), f"{name}, {district}", font=myFont, fill=(255, 0, 0))  #
Display both name and district
        photo.paste(image)

    root.after(10, update_image)
def quitapp():
    cap.release()  # Release the camera
    root.destroy()  # Destroy the window
    conn.close()  # Close the database connection

root = Tk()
myFont = ImageFont.truetype('arial.ttf', 30)
image = Image.fromarray(np.zeros((500, 500, 3), dtype=np.uint8))
photo = ImageTk.PhotoImage(image)
canvas = tk.Canvas(root, width=photo.width(), height=photo.height())
canvas.pack(fill='both', expand=True)

canvas.create_image((0, 0), image=photo, anchor='nw')

button_quit = tk.Button(root, text="Quit", command=quitapp)
button_quit.pack(side='left')
update_image()
root.mainloop()
cap.release()
```