FACE RECOGNITION SYSTEM

Computer Vision Project

Submit BY: Ghazanfar Ali (FA19-MSCS-0016)

Submit To: Dr. Ghazanfar Monir

Muhammad Ali Jinnah University

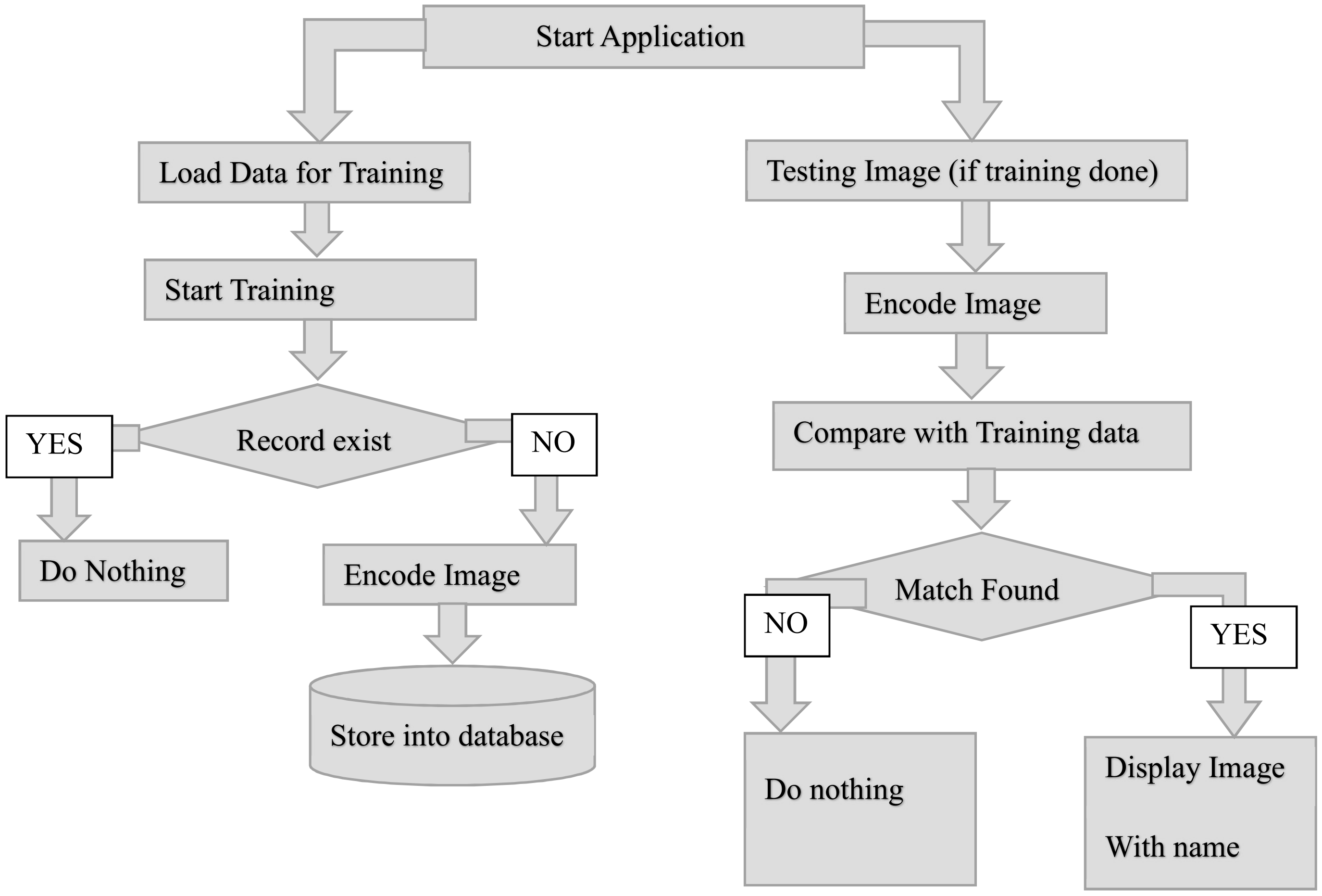
27-Jan-2021

PROJECT REPORT

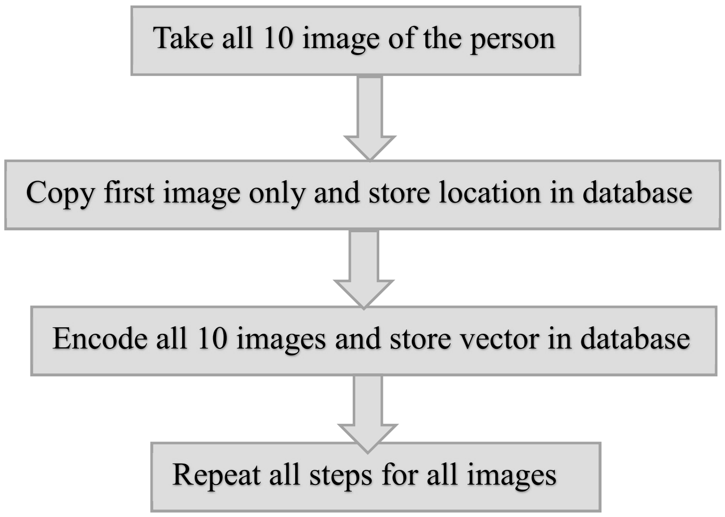
Problem Statement:

Use face recognition library (https://pypi.org/project/face-recognition/) to implement a face recognition system that detects multiple faces from an image and identify those persons. Test it for 500 persons.

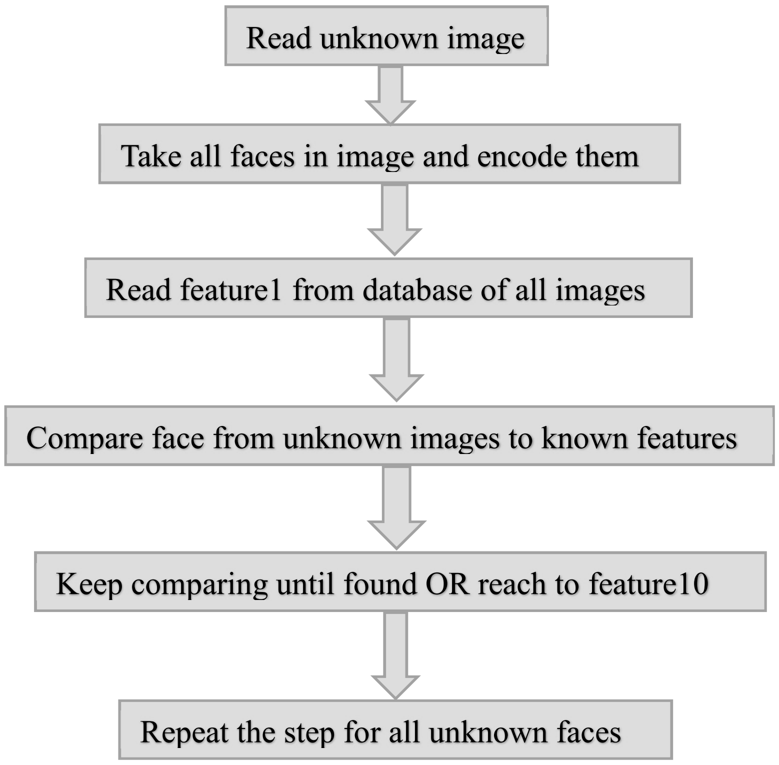
Flow Diagram of Application:



Flow Diagram of inside working of Training Images:


Take all 10 image of the person
Copy first image only and store location in database
Encode all 10 images and store vector in database
Repeat all steps for all images

Flow Diagram of inside working of Testing Image:

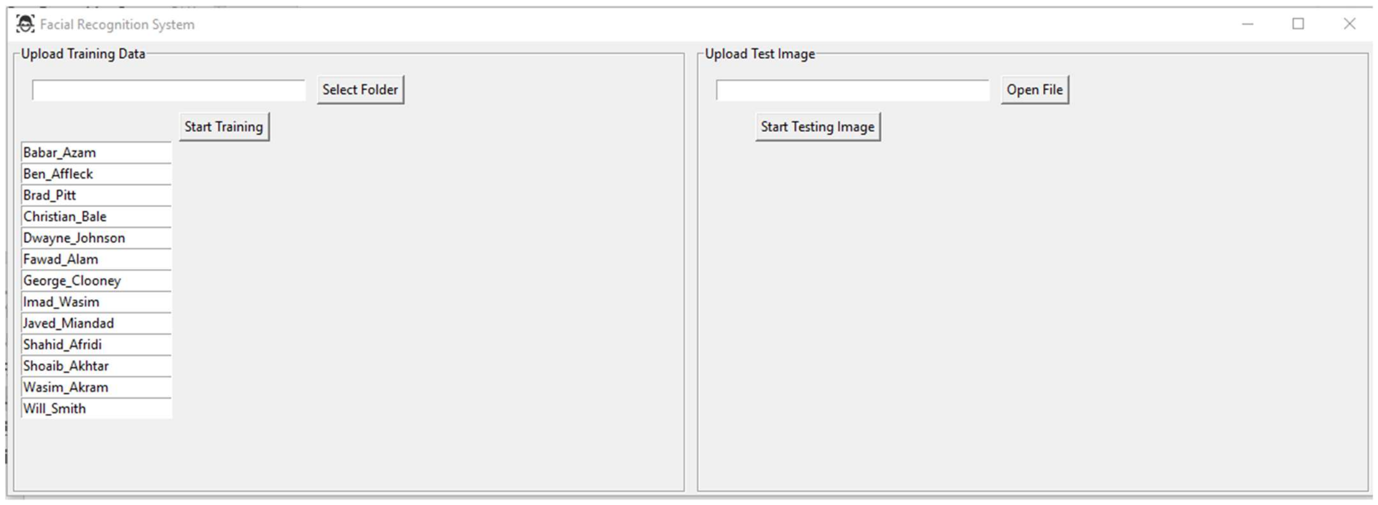

Read unknown image
Take all faces in image and encode them
Read feature1 from database of all images
Compare face from unknown images to known features
Keep comparing until found OR reach to feature10
Repeat the step for all unknown faces

First of all importing necessaries libraries:

```python
import ast
import re
import tkinter as tk
from PIL import ImageTk, Image
import face_recognition
import os
import cv2
import mysql.connector
from PIL import ImageTk, Image
from tkinter import filedialog
import shutil
import numpy as np
```

For GUI Main Window:

```python
root = tk.Tk()
root.title("Facial Recognition System")
root.iconbitmap('./frs.ico')
# root.geometry("1250x800")

root.attributes("-fullscreen", False)
root.bind("<F11>", lambda event: root.attributes("-fullscreen",
                                    not root.attributes("-fullscreen")))
root.bind("<Escape>", lambda event: root.attributes("-fullscreen", False))
```



```
Babar_Azam
Ben_Affleck
Brad_Pitt
Christian_Bale
Dwayne_Johnson
Fawad_Alam
George_Clooney
Imad_Wasim
Javed_Miandad
Shahid_Afridi
Shoaib_Akhtar
Wasim_Akram
Will_Smith
```

Training GUI Area:

Following code is only for creating Graphical User Interface of Training Area. As also shown in previous diagram.

```
      my_train_frame = tk.LabelFrame(root, text="Upload Training Data",
padx=5, pady=5, width=600, height=400)
my_train_frame.grid(row=0, column=0, padx=5, pady=2)
my_train_frame.grid_propagate(False)


def load_train_images():
    root.dirname = filedialog.askdirectory(initialdir="./", title="Select a
folder")
    train_img_entry_box.insert(0, root.dirname)

    my_train_img_label = tk.Label(my_train_frame, text=root.dirname)
    my_train_img_label.grid(row=1, column=0, columnspan=5)


train_img_entry_box = tk.Entry(my_train_frame, width=40)
train_img_entry_box.grid(row=0, column=0, columnspan=5, padx=10, pady=10)
train_img_load_btn = tk.Button(my_train_frame, text="Select Folder",
command=load_train_images)
train_img_load_btn.grid(row=0, column=6)

def load_data():
    mycursor.execute("SELECT name FROM frs")
    uname = mycursor.fetchall()
    uname_lst = [uname[i][0] for i in range(len(uname))]

    rows = []
    for i in range(len(uname_lst)):
        cols = []
        for j in range(1):
            e = tk.Entry(my_train_frame, relief=tk.GROOVE)
            e.grid(row=i+7, column=j, sticky=tk.NSEW)
            e.insert(tk.END,  uname_lst[i])
            cols.append(e)
        rows.append(cols)

load_data()
```

Testing GUI:

Following code is to display Graphical User Interface of testing area as also shown in previous diagram.

```
      my_test_frame = tk.LabelFrame(root, text="Upload Test Image", padx=5,
pady=5, width=600, height=400)
my_test_frame.grid(row=0, column=1, padx=5, pady=2)
my_test_frame.grid_propagate(False)

def load_test_image():
    global my_test_img
    global my_test_img_label
    root.filename = filedialog.askopenfilename(initialdir="./",
title="Select a file",
                                               filetypes=(("all files",
"*.*"), ("jpg files", "*.jpg")))

    # my_test_img_label = tk.Label(my_test_frame, text=root.filename)
    # my_test_img_label.grid(row=2, column=0)
    test_img_entry_box.insert(0, root.filename)

    WIDTH, HEIGHT = 300, 300
    resize_img = Image.open(root.filename).resize((WIDTH, HEIGHT),
Image.ANTIALIAS)

    my_test_img = ImageTk.PhotoImage(resize_img)
    # my_test_img = ImageTk.PhotoImage(Image.open(root.filename))
    my_test_img_label = tk.Label(my_test_frame, image=my_test_img)
    my_test_img_label.grid(row=1, column=1)


test_img_entry_box = tk.Entry(my_test_frame, width=40)
test_img_entry_box.grid(row=0, column=0, columnspan=5, padx=10, pady=10)
test_img_load_btn = tk.Button(my_test_frame, text="Open File",
command=load_test_image)
test_img_load_btn.grid(row=0, column=6)
```

Database Connection:

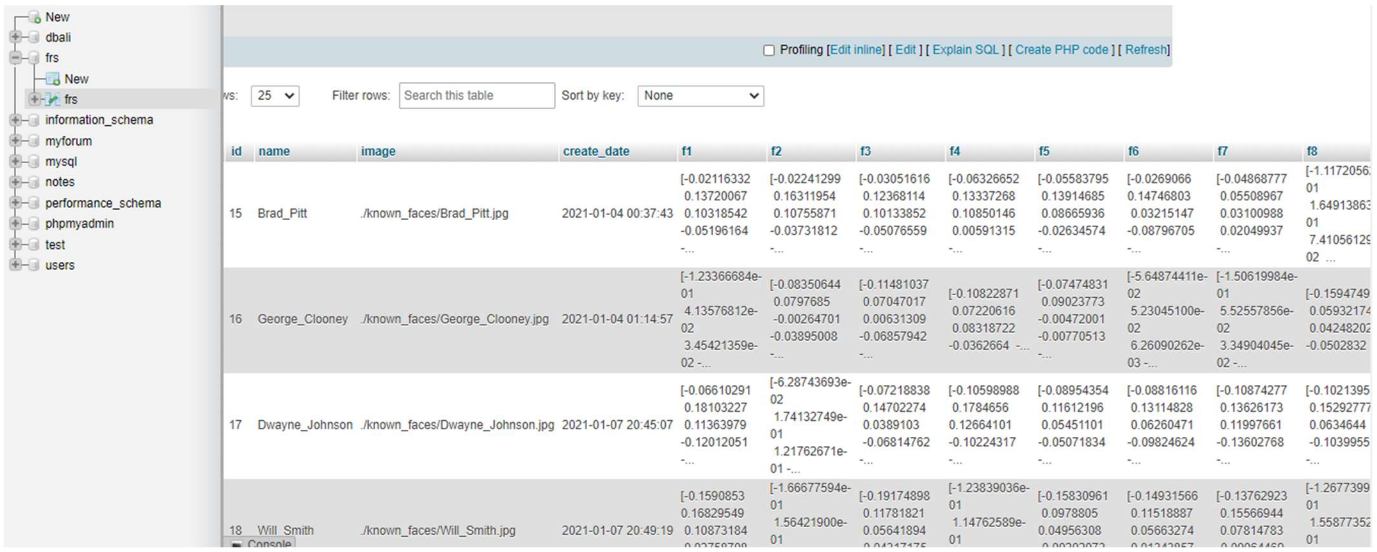
Following code for creating database connection.

```
conn = mysql.connector.connect(
  host="localhost",
  user="root",
  password="",
  database = "frs"
)
```

To insert a record in the database:

Following code is for inserting record in the database.

```
def insert_record(uname, img_path, known_images):
    NAME = uname
    IMAGE = img_path

    print("Length of known images: ", len(known_images))
    print("Type of known images: ", type(known_images))
    print(len(known_images[0]))
    print(type(known_images[0]))

    f = []
    for img in known_images:
        f.append(np.array_str(img))

    print(f[0])
    print(type(f[0]))
    print(len(f))

    # INSERT INTO TABLES
    sql = "INSERT INTO frs (name, image, f1, f2, f3, f4, f5, f6, f7, f8, f9,
f10) " \
          "VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
    val = (NAME, IMAGE, f[0], f[1], f[2], f[3], f[4], f[5], f[6], f[7],
f[8], f[9])
    mycursor.execute(sql, val)

    conn.commit()
    print(mycursor.rowcount, "record inserted.")
    # conn.close()
```



id	name	image	create_date	f1	f2	f3	f4	f5	f6	f7	f8
15	Brad_Pitt	./known_faces/Brad_Pitt.jpg	2021-01-04 00:37:43	[-0.02116332 0.13720067 0.10318542 -0.05196164 -...	[-0.02241299 0.16311954 0.10755871 -0.03731812 -...	[-0.03051616 0.12368114 0.10133852 -0.05076559 -...	[-0.06326652 0.13337268 0.10850146 0.00591315 -...	[-0.05583795 0.13914685 0.08665936 -0.02634574 -...	[-0.0269066 0.14746803 0.03215147 -0.08796705 -...	[-0.04868777 0.05508967 0.03100988 0.02049937 -...	[-1.1172056 01 1.6491386 01 7.4105612 02 ...
16	George_Clooney	./known_faces/George_Clooney.jpg	2021-01-04 01:14:57	[-1.23366684e-01 4.13576812e-02 3.45421359e-02 -...	[-0.08350644 0.0797685 -0.00264701 -0.03895008 -...	[-0.11481037 0.07047017 0.00631309 -0.06857942 -...	[-0.10822871 0.07220616 0.08318722 -0.0362664 -...	[-0.07474831 0.09023773 -0.00472001 -0.00770513 -...	[-5.64874411e-02 5.23045100e-02 6.26090262e-03 -...	[-1.50619984e-01 5.52557856e-02 3.34904045e-02 -...	[-0.1594749 0.0593217 0.0424820 -0.0502832
17	Dwayne_Johnson	./known_faces/Dwayne_Johnson.jpg	2021-01-07 20:45:07	[-0.06610291 0.18103227 0.11363979 -0.12012051 -...	[-6.28743693e-02 1.74132749e-01 1.21762671e-01 -...	[-0.07218838 0.14702274 0.0389103 -0.06814762 -...	[-0.10598988 0.1784656 0.12664101 -0.10224317 -...	[-0.08954354 0.11612196 0.05451101 -0.05071834 -...	[-0.08816116 0.13114828 0.06260471 -0.09824624 -...	[-0.10874277 0.13626173 0.11997661 -0.13602768 -...	[-0.1021395 0.1529277 0.0634644 -0.1039955 -...
18	Will_Smith	./known_faces/Will_Smith.jpg	2021-01-07 20:49:19	[-0.1590853 0.16829549 0.10873184	[-1.66677594e-01 1.56421900e-01	[-0.19174898 0.11781821 0.05641894	[-1.23839036e-01 1.14762589e-01	[-0.15830961 0.0978805 0.04956308	[-0.14931566 0.11518887 0.05663274	[-0.13762923 0.15566944 0.07814783	[-1.2677399 01 1.5587735 01

This is the database on mysql and has following fields as show in above diagram:

- **Id** (auto generated) – uniquely indentifier
- **Name** – name of a person
- **Image** – it's store a single image of the person (store location where image reside)
- **Create_date** (auto generated) – It's hold the creation time of the record
- **f1 – f2** – Hold the encoded vector of each 10 image from f1 to f10

Train Images:

For Start training click on training data button but before load the image by browsing the root folder where all images have been stored. Following code is for do the training, which means it will encode each image and call the insert record function to store it into the database:

```
def training_data():
    print("Loading known faces...")
    print(root.dirname)
    TRAINING_DIR = root.dirname

    mycursor.execute("SELECT name FROM frs")
    name_exist = mycursor.fetchall()
    print(name_exist[0][0])
    print(type(name_exist[0][0]))
    ne = [name_exist[i][0] for i in range(len(name_exist))]
    print(ne)
    for name in os.listdir(TRAINING_DIR):
        print("\nName of Person: ",name)
        known_faces = []
        if name not in ne:
            for i, filename in enumerate(os.listdir(f"{TRAINING_DIR}/{name}")):
                if i == 0:
                    dest_dir = KNOWN_FACES_DIR
                    src_dir = f"{TRAINING_DIR}/{name}"
                    print(src_dir, " -- ", dest_dir)
                    src_file = src_dir + "/" + filename   #os.path.join(src_dir,
filename)
                    print("source file: ", src_file)
                    shutil.copy(src_file, dest_dir)  # copy file to destination dir
                    os.chdir(dest_dir)               # change directory to
destination folder
                    dest_file = filename
                    oldfname, ext = filename.split(".")
                    print(oldfname, ext)
                    newfname = name + "." + ext
                    os.rename(dest_file, newfname)   # rename
                    print("destination file:", newfname)
                    img_path = KNOWN_FACES_DIR + "/" + newfname

                print("Images: ",filename)
                image =
face_recognition.load_image_file(f"{TRAINING_DIR}/{name}/{filename}")
                print(type(image), len(image))
                try:
                    encoding = face_recognition.face_encodings(image)[0]
                    print("encoding: \n", encoding)
                except:
                    print("Error in processing image", i, filename)
                known_faces.append(encoding)
                known_names.append(name)
            insert_record(name, img_path, known_faces)
            os.chdir(TRAINING_DIR)

    load_data()     # Refresh list of name
start_training_btn = tk.Button(my_train_frame, text="Start Training",
command=training_data)
start_training_btn.grid(row=3, column=1)
```

This code perform following task:

- Check if record exist or not
- If record already exist then skip it to load the next training image
- If record does not found then Copy only first image and store it in local folder and keep the location in the database
- Encode all 10 images and store it into database in vector form
- Keep doing this process for all the person that is given by user

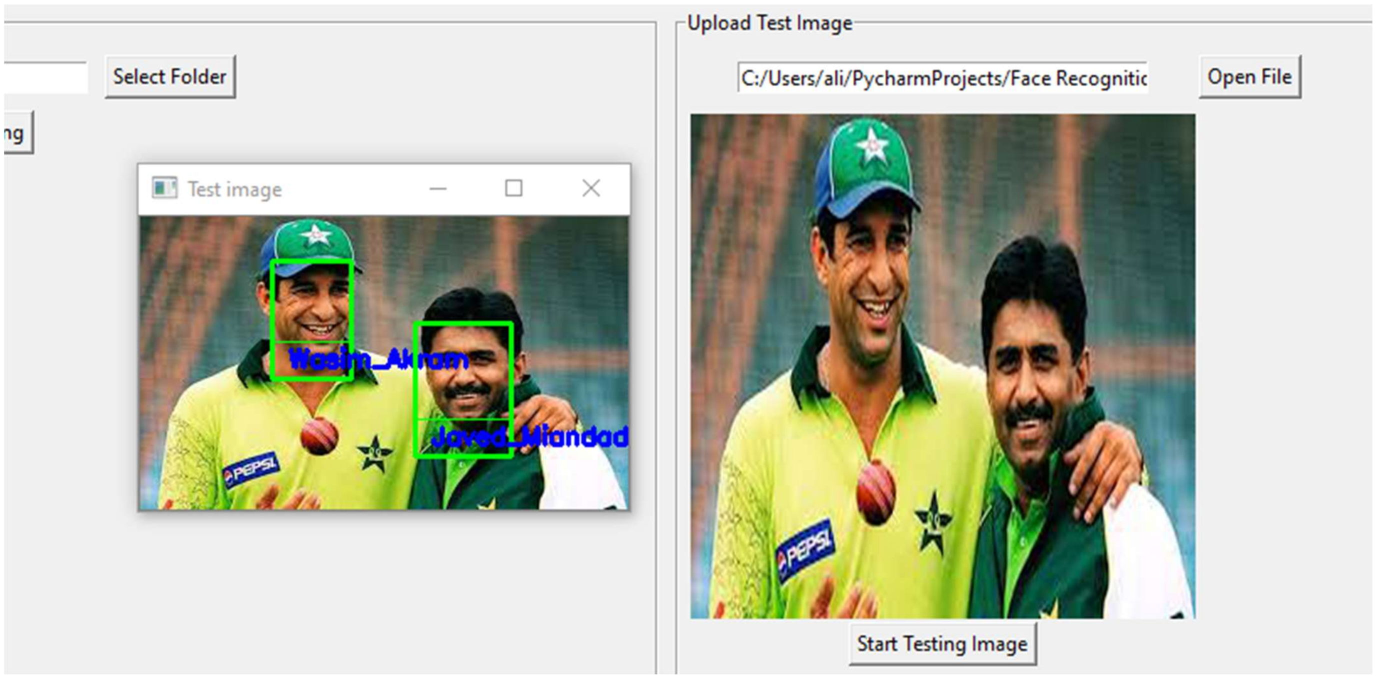
Testing the image:

Following code is for testing image to recognize faces on it.

- First of all it read unknown image
- Encode all the faces in the image, and store the location of each face
- Compare the faces to all image known faces available in the database. For that it’s actually use “**CNN**” model as defined, and it compare with all at once.
- Inside, it is using “**Euclidean Distance**” to compare the face encoding of unknown faces with known faces.
- While loop is here to check until found all faces. As soon as all faces found in the image, it will out from the loop and does not need to check further.
- To compare, fetches feature “f1” of all image and compare it and so on if needed.
- For loop only works if unknown image has multiple faces to recognize.

```
def testing_data():
    global my_test_img
    global my_test_img_label
    print("Processing test image...", root.filename)
    # for filename in os.listdir(UNKNOWN_FACES_DIR):
    image = face_recognition.load_image_file(root.filename)
    # print("before encoding: \n", image)
    locations = face_recognition.face_locations(image, model=MODEL)
    # print("Locations: ", locations)
    print("Length of locations(Number of person in testing image): ",
len(locations))
    encodings = face_recognition.face_encodings(image, locations)
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
    # print("after encoding: \n", image)

    mycursor.execute("SELECT id,name FROM frs")
    name_exist = sorted(mycursor.fetchall())
    print("\n name_exist:------>>>>>", type(name_exist), name_exist)
    ne = [name_exist[i][1] for i in range(len(name_exist))]

    print("Name Exist: ", ne)
    num_unknown = len(locations)
    to_break = []
    print("To break list and its length: ", to_break, len(to_break))
    # Actual Code to Compare the unkown face from training data (known_faces)
    img_feature = 1
    while(len(to_break) <= num_unknown and img_feature<=10):
        print("img_feature no: ----->", img_feature)
        print("To Break List: ", to_break)
        mycursor.execute("SELECT f" + str(img_feature) + " FROM frs")
        myresult = mycursor.fetchall()
        print(type(myresult), len(myresult))
        print(myresult[0])

        kfaces = []
        for i in range(len(myresult)):
            f = myresult[i]
            flst = re.sub('\s+', ',', f[0])
            farr = np.array(ast.literal_eval(flst))
            kfaces.append(farr)

        for face_encoding, face_location in zip(encodings, locations):
            results = face_recognition.compare_faces(kfaces, face_encoding,
TOLERANCE)
            match = None
            print(type(results), "--> ", results)

            if True in results:
                match = ne[results.index(True)]
                print(f"match found: {match}")
                # to_break[results.index(True)] = True
                to_break.append(True)

                top_left = (face_location[3], face_location[0])
                bottom_right = (face_location[1], face_location[2]+22)
                color = [0, 255, 0]
                cv2.rectangle(image, top_left, bottom_right, color,
FRAME_THICKNESS)

                top_left = (face_location[3], face_location[2])
                bottom_right = (face_location[1], face_location[2])
                cv2.rectangle(image, top_left, bottom_right, color, cv2.FILLED)
                cv2.putText(image, match, (face_location[3]+10,
face_location[2]+15),
                                            cv2.FONT_HERSHEY_SIMPLEX, 0.5,
(255,0,0), FONT_THICKNESS)

        img_feature = int(img_feature) + 1

    cv2.imshow("Test image", image)
    cv2.waitKey(0)



start_testing_btn = tk.Button(my_test_frame, text="Start Testing Image",
command=testing_data)
start_testing_btn.grid(row=3, column=1)
```

- After select an image, click on testing image button to recognize face in image.
- It will open image in another window with draw a rectangle and display the name, as shown below:


Select Folder
ng
Test image
Wasim_Akram
Javed_Miandad
Upload Test Image
C:/Users/ali/PycharmProjects/Face Recognitic
Open File
Start Testing Image