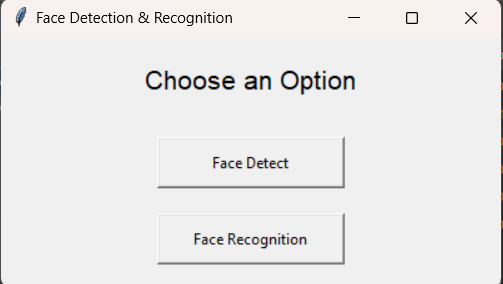
Face Recognition and Detection

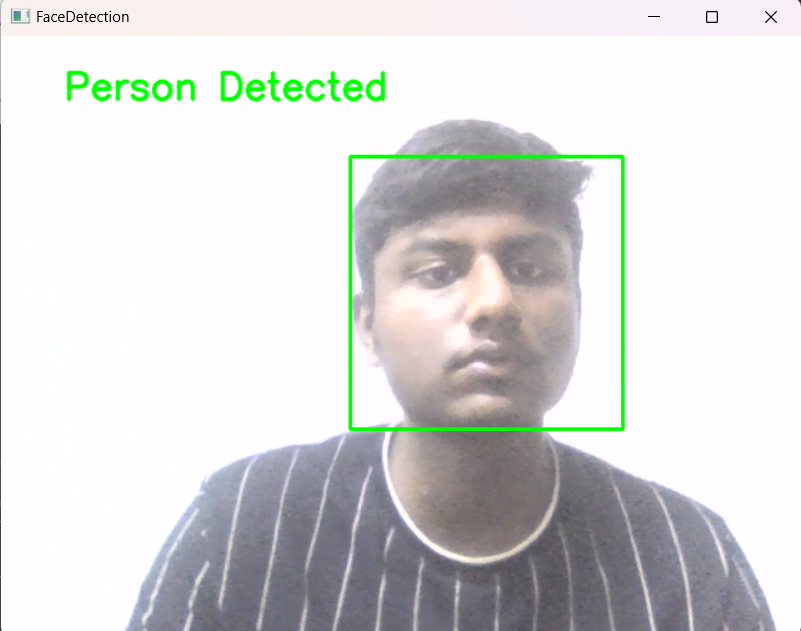
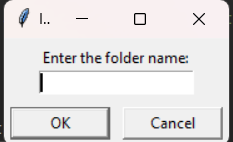
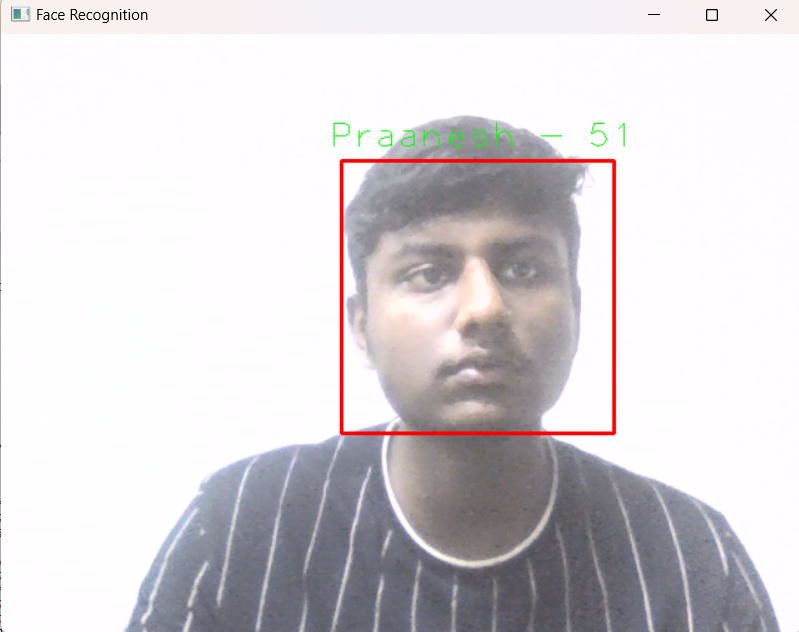
# Coding:

import tkinter as tk  
from tkinter import simpledialog, messagebox  
import sys  
import os  
import cv2  
import numpy as np  
  
sys.path.append('/mnt/data')  
haar\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade\_frontalface\_default.xml")  
  
  
def face\_detect():  
 try:  
 folder\_name = simpledialog.askstring("Input", "Enter the folder name:")  
 if folder\_name:  
 dataset = "dataset1"  
 name = folder\_name  
 path = os.path.join(dataset, name)  
 print(os.path.isdir(path))  
 if not os.path.exists(path):  
 os.makedirs(path)  
 (width, height) = (130, 100)  
 cam = cv2.VideoCapture(0)  
 count = 1  
 while count < 301:  
 \_, img = cam.read()  
 gray\_img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
 face = haar\_cascade.detectMultiScale(gray\_img, 1.3, 4)  
 if len(face) > 0:  
 for (x, y, w, h) in face:  
 cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)  
 print(count)  
 print("Person Detected")  
 cv2.putText(img, 'Person Detected', (50, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 255, 0), 2,  
 cv2.LINE\_AA)  
 else:  
 cv2.putText(img, 'No Person Detected', (50, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 0, 255), 2,  
 cv2.LINE\_AA)  
 print("NO Person Detected")  
 cv2.imshow("FaceDetection", img)  
 key = cv2.waitKey(10)  
 if key == 27:  
 break  
 if len(face) > 0:  
 for (x, y, w, h) in face:  
 face\_only = img[y:y + h, x:x + w]  
 resize\_img = cv2.resize(face\_only, (width, height))  
 cv2.imwrite("%s/%s.jpg" % (path, count), resize\_img)  
 count += 1  
 print("Completed Face Detection")  
 cam.release()  
 cv2.destroyAllWindows()  
 else:  
 messagebox.showwarning("Input Error", "Folder name cannot be empty!")  
 except Exception as e:  
 print(f"Error occurred[Detect]: {str(e)}")  
  
  
def face\_recognize():  
 try:  
 datasets = 'dataset1'  
 print('Training...')  
  
 (images, labels, names, id) = ([], [], {}, 0)  
  
 for (subdirs, dirs, files) in os.walk(datasets):  
 for subdir in dirs:  
 names[id] = subdir  
 subjectpath = os.path.join(datasets, subdir)  
 print("Subject Path:", subjectpath)  
 for filename in os.listdir(subjectpath):  
 path = os.path.join(subjectpath, filename)  
 print("Image Path:", path)  
 label = id  
 image = cv2.imread(path, cv2.IMREAD\_GRAYSCALE)  
 if image is None:  
 print("Error loading image:", path)  
 continue  
 print("Image shape:", image.shape)  
 images.append(image)  
 labels.append(int(label))  
 id += 1  
 (images, labels) = [np.array(lis) for lis in [images, labels]]  
 print("Number of images loaded:", len(images))  
 print("Labels:", labels)  
 (width, height) = (130, 100)  
 model = cv2.face.LBPHFaceRecognizer\_create()  
 model.train(images, labels)  
 webcam = cv2.VideoCapture(0)  
 cnt = 0  
 while True:  
 ret, img = webcam.read()  
 if not ret:  
 print("Failed to capture image from webcam")  
 break  
 gray\_img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
 faces = haar\_cascade.detectMultiScale(gray\_img, 1.3, 5)  
 for (x, y, w, h) in faces:  
 cv2.rectangle(img, (x, y), (x + w, y + h), (0, 0, 255), 2)  
 face\_crop = gray\_img[y:y + h, x:x + w]  
 resized\_face = cv2.resize(face\_crop, (width, height))  
 prediction = model.predict(resized\_face)  
 if prediction[1] < 800:  
 cv2.putText(img, "%s - %.0f" % (names[prediction[0]], prediction[1]), (x - 10, y - 10),  
 cv2.FONT\_HERSHEY\_PLAIN, 2, (0, 255, 0))  
 print(names[prediction[0]])  
 cnt = 0  
 else:  
 cnt += 1  
 cv2.putText(img, 'Unknown', (x - 10, y - 10), cv2.FONT\_HERSHEY\_PLAIN, 2, (0, 0, 255))  
 if cnt > 100:  
 print("Unknown Person")  
 cv2.imwrite("unknown.jpg", img)  
 cnt = 0  
 cv2.imshow("Face Recognition", img)  
 key = cv2.waitKey(10)  
 if key == 27:  
 break  
 webcam.release()  
 cv2.destroyAllWindows()  
 except Exception as e:  
 print(f"Error occurred[Recognize]: {str(e)}")  
  
  
root = tk.Tk()  
root.title("Face Detection & Recognition")  
root.geometry("400x200")  
label = tk.Label(root, text="Choose an Option", font=("Arial", 16))  
label.pack(pady=20)  
detect\_button = tk.Button(root, text="Face Detect", command=face\_detect, width=20, height=2)  
detect\_button.pack(pady=10)  
recognize\_button = tk.Button(root, text="Face Recognition", command=face\_recognize, width=20, height=2)  
recognize\_button.pack(pady=10)  
root.mainloop()

if you want to download the code:  
<https://codeshare.io/3y03jz>

# Output:



**Face Detect:  
  
  
Face Recognition:  
**