import cv2

import numpy as np

import face\_recognition

import os

from datetime import datetime

path = 'IMAGES'

images = []

personNames = []

myList = os.listdir(path)

print(myList)

for cu\_img in myList:

current\_Img = cv2.imread(f'{path}/{cu\_img}')

images.append(current\_Img)

personNames.append(os.path.splitext(cu\_img)[0])

print(personNames)

def faceEncodings(images):

encodeList = []

for img in images:

img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

encode = face\_recognition.face\_encodings(img)[0]

encodeList.append(encode)

return encodeList

def attendance(name):

with open('Attendance.csv', 'r+') as f:

myDataList = f.readlines()

nameList = []

for line in myDataList:

entry = line.split(',')

nameList.append(entry[0])

if name not in nameList:

time\_now = datetime.now()

tStr = time\_now.strftime('%H:%M:%S')

dStr = time\_now.strftime('%d/%m/%Y')

f.writelines(f'\n{name},{tStr},{dStr},{"Present"}')

encodeListKnown = faceEncodings(images)

print('All Encodings Complete!!!')

cap = cv2.VideoCapture(0)

while True:

ret, frame = cap.read()

faces = cv2.resize(frame, (0, 0), None, 0.25, 0.25)

faces = cv2.cvtColor(faces, cv2.COLOR\_BGR2RGB)

facesCurrentFrame = face\_recognition.face\_locations(faces)

encodesCurrentFrame = face\_recognition.face\_encodings(faces, facesCurrentFrame)

for encodeFace, faceLoc in zip(encodesCurrentFrame, facesCurrentFrame):

matches = face\_recognition.compare\_faces(encodeListKnown, encodeFace)

faceDis = face\_recognition.face\_distance(encodeListKnown, encodeFace)

# print(faceDis)

matchIndex = np.argmin(faceDis)

if matches[matchIndex]:

name = personNames[matchIndex].upper()

# print(name)

y1, x2, y2, x1 = faceLoc

y1, x2, y2, x1 = y1 \* 4, x2 \* 4, y2 \* 4, x1 \* 4

cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)

cv2.rectangle(frame, (x1, y2 - 35), (x2, y2), (0, 255, 0), cv2.FILLED)

cv2.putText(frame, name, (x1 + 6, y2 - 6), cv2.FONT\_HERSHEY\_COMPLEX, 1, (255, 255, 255), 2)

attendance(name)

cv2.imshow('Webcam', frame)

if cv2.waitKey(1) == 13:

break

cap.release()

cv2.destroyAllWindows()