import csv

from datetime import datetime import cv2

import face\_recognition import numpy as np

#sets up the webcam for capturing video.

video\_capture = cv2.VideoCapture(0) #signifies the code is using the default camera for capturing.

##himanshu\_image = face\_recognition.load\_image\_file(r"C:\Users\DELL\Desktop\ethics\photos\himanshu.j peg") #imgs are loaded

##himanshu\_encoding = face\_recognition.face\_encodings(himanshu\_image)[0] # generates face encodings(vectors representing faical features) are loaded

aastha\_image = face\_recognition.load\_image\_file(r"C:\Users\DELL\Desktop\backend\photos\aastha.jp eg")

aastha\_encoding = face\_recognition.face\_encodings(aastha\_image)[0]

tesla\_image = face\_recognition.load\_image\_file(r"C:\Users\DELL\Desktop\backend\photos\tesla.jpe g")

tesla\_encoding = face\_recognition.face\_encodings(tesla\_image)[0]

##ramraj\_image = face\_recognition.load\_image\_file(r"C:\Users\DELL\Desktop\ethics\photos\ramraj.jpe g")

##ramraj\_encoding = face\_recognition.face\_encodings(ramraj\_image)[0]

#A list containing - face encodings for known individuals. known\_face\_encoding = [

##himanshu\_encoding, aastha\_encoding, tesla\_encoding, ##ramraj\_encoding

]

#A list containing - corresponding names for known individuals. known\_faces\_names = [

"aastha",

"tesla", ##"ramraj"

]

# A copy of the list of known faces' names, used for tracking attendance. students = known\_faces\_names.copy()

#Lists to store face detection results. face\_locations = []

face\_encodings = [] face\_names = []

# A boolean variable used to control certain actions (e.g., printing students and writing to CSV).

s = True

now = datetime.now() # Gets the current date and time.

current\_date = now.strftime("%Y-%m-%d") #strftime - used to format time in YYYY- MM-DD

f = open(current\_date + '.csv', 'w+', newline='') #opens or create a csv file with the current date as file name

lnwriter = csv.writer(f) #creating a writer object and writing rows of data into f file

while True:

\_, frame = video\_capture.read() #captures video frames from the webcam. small\_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25) #resizes the

captured frame in smaller size making it more #efficient for face recognition procesing.

rgb\_small\_frame = cv2.cvtColor(small\_frame, cv2.COLOR\_BGR2RGB) #converts the resized frame from default BGR to RGB

#which is the format expected by face recognition library. if s:

#detected faces location and encodings are stored. face\_locations = face\_recognition.face\_locations(rgb\_small\_frame) face\_encodings = face\_recognition.face\_encodings(rgb\_small\_frame,

face\_locations) #images are encoded. face\_names = []

#processes each detected face, compares it with known faces, and updates the attendance status.

for face\_encoding in face\_encodings:

matches = face\_recognition.compare\_faces(known\_face\_encoding, face\_encoding) #Compares the current face encoding with the known face encodings stored in known\_face\_encoding

name = ""

face\_distance = face\_recognition.face\_distance(known\_face\_encoding, face\_encoding) #Calculates the face distance (similarity) between the current face encoding and known face encodings.

best\_match\_index = np.argmin(face\_distance) # Finds the index of the smallest face distance, indicating the best match.

if matches[best\_match\_index]:

name = known\_faces\_names[best\_match\_index] #name is assigned from known\_faces\_names

face\_names.append(name)

if name in known\_faces\_names:

font = cv2.FONT\_HERSHEY\_SIMPLEX

bottomLeftCornerOfText = (10, 100)

fontScale = 1.5

fontColor = (255, 0, 0)

thickness = 3

lineType = 2

#shows text on the video feed which includes name of the student(jpg) and present.

#helps us to show real time results.

cv2.putText(frame, name + ' Present', #putText helps to mark the

attendence

bottomLeftCornerOfText, font,

fontScale, fontColor, thickness, lineType)

if name in students: students.remove(name) print(students)

current\_time = now.strftime("%H-%M-%S") lnwriter.writerow([name, current\_time]) #written to csv file.

cv2.imshow("attendence system", frame) # display the video feed.

if cv2.waitKey(1) & 0xFF == ord('q'): #if key is q then its terminated. break

video\_capture.release() #Release the webcam resources. cv2.destroyAllWindows() #closes all opencv windows. f.close() # close the csv file.