import speech\_recognition as sr

import pyttsx3

import webbrowser

import datetime

import googleapiclient.discovery

import cv2

import face\_recognition

import numpy as np

# Set up speech recognition

r = sr.Recognizer()

# Set up variables

muted = False

wake = True

api\_key = "AIzaSyCmz1hqVoMDqCy6UDKQf6G7UtV4srIwwoE"

youtube = googleapiclient.discovery.build("youtube", "v3", developerKey=api\_key)

# Set up the pyttsx3 engine

engine = pyttsx3.init()

engine.setProperty('volume', 1)

engine.setProperty('rate', 150)

# Wakeword setup

def wakeword(x):

global wake

if x == True:

wake = True

elif x == False:

wake = False

# Camera setup

def cam():

video\_capture = cv2.VideoCapture(0)

known\_face\_encodings = []

with open('names.txt', 'r') as file:

lines = file.readlines()

lines = [line.strip() for line in lines]

known\_face\_names = lines

face\_locations = []

face\_encodings = []

face\_names = []

process\_this\_frame = True

# Create an empty dictionary

images = {}

# Open the text file in read mode

with open("names.txt", "r") as file:

# Iterate over the lines of the text file

for line in file:

# Split the line on the comma character

names = line.strip().split(",")

# Iterate over the names in the line

for name in names:

# Load the image for the element

image = face\_recognition.load\_image\_file(f"faces/{name}.jpg")

# Add the image and name to the dictionary

images[name] = image

# Set up variables

known\_face\_encodings = []

known\_face\_names = []

# Iterate over the keys and values in the images dictionary

for name, image in images.items():

# Get the face encoding for the element

encoding = face\_recognition.face\_encodings(image)[0]

# Append the encoding to the known\_face\_encodings list

known\_face\_encodings.append(encoding)

# Append the name to the known\_face\_names list

known\_face\_names.append(name)

while True:

# Grab a single frame of video

ret, frame = video\_capture.read()

# Only process every other frame of video to save time

if process\_this\_frame:

# Resize frame of video to 1/4 size for faster face recognition processing

small\_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)

# Convert the image from BGR color (which OpenCV uses) to RGB color (which face\_recognition uses)

rgb\_small\_frame = small\_frame[:, :, ::-1]

# Find all the faces and face encodings in the current frame of video

face\_locations = face\_recognition.face\_locations(rgb\_small\_frame)

face\_encodings = face\_recognition.face\_encodings(rgb\_small\_frame, face\_locations)

face\_names = []

for face\_encoding in face\_encodings:

# See if the face is a match for the known face(s)

matches = face\_recognition.compare\_faces(known\_face\_encodings, face\_encoding)

name = "Unknown"

# # If a match was found in known\_face\_encodings, just use the first one.

# if True in matches:

# first\_match\_index = matches.index(True)

# name = known\_face\_names[first\_match\_index]

# Or instead, use the known face with the smallest distance to the new face

face\_distances = face\_recognition.face\_distance(known\_face\_encodings, face\_encoding)

best\_match\_index = np.argmin(face\_distances)

if matches[best\_match\_index]:

name = known\_face\_names[best\_match\_index]

face\_names.append(name)

process\_this\_frame = not process\_this\_frame

# Display the results

for (top, right, bottom, left), name in zip(face\_locations, face\_names):

# Scale back up face locations since the frame we detected in was scaled to 1/4 size

top \*= 4

right \*= 4

bottom \*= 4

left \*= 4

# Draw a box around the face

cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

# Draw a label with a name below the face

cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255), cv2.FILLED)

font = cv2.FONT\_HERSHEY\_DUPLEX

cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)

# Display the resulting image

cv2.imshow('Video', frame)

# Hit 'q' on the keyboard to quit!

if cv2.waitKey(1) & 0xFF == ord('q'):

break

# Release handle to the webcam

video\_capture.release()

cv2.destroyAllWindows()

def saveFace():

camera = cv2.VideoCapture(0)

\_, frame = camera.read()

image = frame

print("Person Recognized, what is your name?")

engine.say("Person Recognized, what is your name?")

engine.runAndWait()

sname = input("name:")

engine.say("You said " + sname + " correct?")

engine.runAndWait()

response = input("y/n")

response = response.lower()

print(response)

if 'y' in response:

cv2.imwrite("faces/{}.jpg".format(sname), image)

with open("names.txt", "a") as file:

# Write the new line to the text file

file.write(f",{sname}")

engine.say("Person save success")

engine.runAndWait()

elif 'n' in response:

engine.say("Alright, lets try that again. What is your name?")

engine.runAndWait()

sname = input("name:")

engine.say("You said " + sname + " correct?")

engine.runAndWait()

response = input("y/n")

response = response.lower()

if 'y' in response:

cv2.imwrite("faces/{}.jpg".format(sname), image)

# Open the text file in append mode

with open("names.txt", "a") as file:

file.write(f",{sname}")

engine.say("Person save success")

engine.runAndWait()

else:

engine.say("Person save failed, please try saving them again")

engine.runAndWait()

else:

engine.say("Unable to save user due to invalid response, please reattempt save.")

engine.runAndWait()

# ai options

def options(text):

if text.startswith("zulu "):

text = text[5:]

if text == "quit": # Quit if user says "no"

engine.say("Goodbye, See you next time")

engine.runAndWait()

exit()

elif 'current time' in text:

now = datetime.datetime.now()

time\_str = now.strftime("%H:%M:%S")

print(f"Current time: {time\_str}")

engine.say(now)

engine.runAndWait()

elif "today's date" in text:

now = datetime.datetime.now()

print(f"Current time: {now}")

engine.say("Today's date and time is " + now)

engine.runAndWait()

elif 'wake word on' in text:

wakeword(True)

elif 'wake word off' in text:

wakeword(False)

elif 'open eclass' in text:

webbrowser.open(f"https://publish.gwinnett.k12.ga.us/gcps/home/gcpslogin?error=&username=")

print("Sure. Opening EClass")

engine.say("Opening E Class")

engine.runAndWait()

elif 'search youtube for' in text:

query = text[18:]

print(query)

request = youtube.search().list(part="id,snippet", type='video', q=query, maxResults=1)

response = request.execute()

video\_id = response["items"][0]["id"]["videoId"]

video\_url = f"https://www.youtube.com/watch?v={video\_id}"

webbrowser.open(video\_url)

engine.say("Searching youtube for" + text[18:])

engine.runAndWait()

elif 'open' in text:

webbrowser.open(f"www.{text[10:]}.com")

elif 'search for' in text:

# Open a search query in the default web browser

webbrowser.open(f"https://www.google.com/search?q={text[10:]}")

print("Searching for" + text[10:])

engine.say("Searching for" + text[10:])

engine.runAndWait()

elif 'looking at' in text:

print("Activating Camera, press Q to quit")

engine.say("Activating Camera, press Q to quit")

engine.runAndWait()

cam()

elif 'save face' in text:

saveFace()

else:

print("unknown command")

# Wait for user to speak

def zulu():

with sr.Microphone() as source:

print("Say something:")

sound = r.listen(source) # Define the audio variable here

return sound # Return the audio variable

while True:

if muted == False :

audio = zulu() # Assign the return value of zulu to the global audio variable

try:

text1 = r.recognize\_google(audio)

text3 = text1.lower()

print(f"You said: {text3}")

if wake == True:

if text3.startswith(("zulu")):

options(text3)

elif wake == False:

options(text3)

except sr.UnknownValueError:

print("Sorry, I didn't understand that.")

except sr.RequestError as e:

print("Error making request to Google Speech Recognition service: {0}".format(e))

engine.say("Error making request to Google Speech Recognition service: {0}".format(e)) # Convert the text to speech and play it

engine.runAndWait()