In [ ]:
```
!pip install --upgrade google-api-python-client google-auth-httplib2 google-auth-oauthlib
```

In [ ]:
```
!pip install pvporcupine==1.9.5
```

In [ ]:
```
!pip install SpeechRecognition
```

In [ ]:
```
!pip install pyttsx3
```

In [ ]:
```
!pip install requests
```

In [ ]:
```
!pip install google-cloud-dialogflow
```

In [ ]:
```
import pyttsx3
import os
import os.path
import speech_recognition as sr
import openai
from google.cloud import dialogflow_v2beta1 as dialogflow
import pvporcupine
import struct
import time
import pandas as pd
import pyaudio
import tkinter as tk
from tkinter import Tk, Label, Entry, StringVar, Button
from itertools import cycle
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

import os
import os.path
from datetime import date
from matplotlib import pyplot as plt
import smtplib
from  email.mime.text import MIMEText

from __future__ import print_function

import datetime
from tkinter import scrolledtext

from datetime import datetime
from google.auth.transport.requests import Request
from google.oauth2.credentials import Credentials
from google_auth_oauthlib.flow import InstalledAppFlow
from googleapiclient.discovery import build
from googleapiclient.errors import HttpError
class ModifiedChatbotGUI:

    SCOPES= ['https://www.googleapis.com/auth/calendar']

    #This is a lsit of keywords we could use as wakewords
    #we selected "blueberry"
    print(pvporcupine.KEYWORDS)

    #Project ID, session ID and dialog flow credntial are used to send prompts to Dialog
    DIALOGFLOW_PROJECT_ID = "alarm-uben"
    DIALOGFLOW_LANGUAGE_CODE = "en"
    SESSION_ID= 'me'
    os.environ["GOOGLE_APPLICATION_CREDENTIALS"]= "alarm_key.json"

    #THe Open ai api key is used to send prompts to Chatgpt3 and return responses
    openai.api_key = "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"
    #For text to speech we initialize the engine
    engine= pyttsx3.init('sapi5')
    #Next we set the rate of speech
    engine.setProperty('rate', 130)
    #and the colume of speech
    engine.setProperty('volume', 1.0)

    voices=engine.getProperty('voices')
    engine.setProperty('voice', voices[0].id)
   #REFERENCE FOR UI
#https://docs.python.org/3/library/tkinter.ttk.html#id4
    def __init__(self, root):
            self.root = root
            self.root.title("Voice Chatbot GUI")
            # GIF Animation
            self.gif_frames = [tk.PhotoImage(file='FITBOT.gif', format='gif -index %i' % i
            self.gif_frame_cycle = cycle(self.gif_frames)
            self.gif_label = tk.Label(self.root, image=next(self.gif_frame_cycle))
            self.gif_label.pack(pady=10)
            self.animate_gif()
            # Button to activate the chatbot
            self.activate_button = tk.Button(self.root, text="Activate Chatbot", command=s
            self.activate_button.pack(pady=20)

            # Label & Entry for chatbot's response
            self.bot_label = tk.Label(self.root, text="Chatbot Response:")
            self.bot_label.pack(pady=10)
            self.bot_entry_var = StringVar()
            self.bot_response = scrolledtext.ScrolledText(self.root, wrap=tk.WORD, width=5
            self.bot_response.pack(pady=10)
    #Here we create a function to have the bot speak
    def animate_gif(self):
        self.gif_label.configure(image=next(self.gif_frame_cycle))
        self.root.after(100, self.animate_gif)  # Adjust the delay if needed

    def speak(self,text):
        global engine
        self.engine.say(text)
        self.engine.runAndWait()
    """"REFERENCE2 USED TO MAKE A GRAPH ON INTERFACE
https://matplotlib.org/stable/api/backend_tk_api.html """
    def show_graph(self, df):
        # Create a new Toplevel window
        graph_window = tk.Toplevel(self.root)
        graph_window.title("Graph Display")

    # Create the graph
        fig, ax = plt.subplots(figsize=(5, 4))
        df.plot.line(x="Date", y="Weight", ax=ax)
        print(df)

    # Embed the graph into the Toplevel window
        canvas = FigureCanvasTkAgg(fig, master=graph_window)
        canvas_widget = canvas.get_tk_widget()
        canvas_widget.pack(side=tk.TOP, fill=tk.BOTH, expand=True)
        canvas.draw()
        graph_window.update()

    def prime(self):
        global SCOPES,DIALOGFLOW_PROJECT_ID,DIALOGFLOW_LANGUAGE_CODE,SESSION_ID,engine,voi
        self.update_bot_response("Profile:",name)

        Talk=True
        while Talk==True:
            if name != "nu":
                #User Profiles variables are taken from the csv
                df= pd.read_csv(name+".csv")
                today=date.today()
                years=df.iloc[0,1]
                goal=df.iloc[0,4]
                goal_date =df.iloc[0,5]
                weight=df["Weight"].tail()
                EMAIL=df.iloc[0,6]
                PASSWORD=df.iloc[0,7]
            #Next we set the microphone up to listen
            #IT will pass the audio to the Google API
            #for conversion to text
            r = sr.Recognizer()
            with sr.Microphone() as source:
                audio = r.listen(source)
                prompt=''
                try:
                    prompt =r.recognize_google(audio)
                    self.update_bot_response("USER:",prompt)
                except Exception as e:
                    self.update_bot_response("Exception"+str(e))

            #We Create the Dialogflow  session, pass it the prompt and return the intent a
            #In traing we also returned the confidence score and wrote that to a seperate
            session_client = dialogflow.SessionsClient( )
            session= session_client.session_path(self.DIALOGFLOW_PROJECT_ID, self.SESSION_
            text_input= dialogflow.types.TextInput(text=prompt, language_code=self.DIALOGF
            query_input= dialogflow.types.QueryInput(text=text_input)
            try:
                response = session_client.detect_intent(session=session, query_input=query
                self.update_bot_response("Query Text", response.query_result.query_text)
                self.update_bot_response("Intent is", response.query_result.intent.display
                answer=response.query_result.fulfillment_text
            except Exception as e:
                self.update_bot_response("Exception"+ str(e))
                answer="nu"
                pass
            if answer != "nu":
                self.speak(answer)
                Catch=True
                while Catch==True:
                    #If the intent is Exit the user is reaturned to the wake word
                    if response.query_result.intent.display_name == "Exit":
                        self.wake_word()
                        Talk=False
                        Catch=False
                    #This intent gives the user a list of features
                    if response.query_result.intent.display_name == "features":
                        self.speak("As a Fitness Chatbot I can help with a number of tasks
                        self.wake_word()
                    #If the intent is a question the prompt is sent to Chat GPT
                    elif response.query_result.intent.display_name == "Question about Nutr
                        #We create a list
                        messages=[]
                        #We define the system content. FOr this event we tell CHAT GPT
                        #IT is a chatbot
                        system_content='You are a fitness and nutrition assistant called F
                        #We append the role system and the system contest to the list
                        messages.append({"role":"system","content":system_content})
                        #We then append the role user and the prompt to the list
                    messages.append({"role":"user","content":prompt})
                    response=openai.ChatCompletion.create(
                                    #We specify the model
                                    model="gpt-3.5-turbo",
                                    messages=messages,
                                    max_tokens=1000,
                                    #Temperature controls creativity
                                    temperature=0.5)
                    output_text=response['choices'][0]['message']['content'].strip()
                    self.speak(output_text)
                    Catch=False
                    self.wake_word()

                elif response.query_result.intent.display_name == "Question about Work
                    messages=[]
                    system_content='You are a fitness and nutrition assistant called F
                    messages.append({"role":"system","content":system_content})
                    messages.append({"role":"user","content":prompt})
                    response=openai.ChatCompletion.create(
                                    model="gpt-3.5-turbo",
                                    messages=messages,
                                    max_tokens=1000,
                                    temperature=0.5)
                    output_text=response['choices'][0]['message']['content'].strip()
                    self.speak(output_text)
                    Catch=False
                    self.wake_word()
                #IF the Intent is Motivation the user will get a pep talk from chat gp
                elif response.query_result.intent.display_name == "Motivation":
                    messages=[]
                    system_content='You are a fitness and nutrition assistant called F
                    messages.append({"role":"system","content":system_content})
                    messages.append({"role":"user","content":prompt})
                    response=openai.ChatCompletion.create(
                                    model="gpt-3.5-turbo",
                                    messages=messages,
                                    max_tokens=1000,
                                    temperature=0.5)
                    output_text=response['choices'][0]['message']['content'].strip()
                    self.speak(output_text)
                    Catch=False
                    self.wake_word()
                #If the intent is a question about goals the users profile variables
                #are included in the system content using .format()
                elif response.query_result.intent.display_name == "Question about Goal
                    #THe first statment lets the user they can't use this feature with
                    if name == "nu":
                        self.speak("I'm sorry, you'll need to set up a profile to char
                        Catch=False
                        self.wake_word()
                    else:
                        messages=[]
                        system_content='You are a fitness and nutrition assistant call
                        messages.append({"role":"system","content":system_content})
                        messages.append({"role":"user","content":prompt})
                        response=openai.ChatCompletion.create(
                                        model="gpt-3.5-turbo",
                                        messages=messages,
                                        max_tokens=200,
                                        temperature=0.5)
                        output_text=response['choices'][0]['message']['content'].strip
                        self.speak(output_text)
                        Catch=False
                        self.wake_word()
                #THe Graph intent opens a seperate window with a graph of
                #the users weight
                elif response.query_result.intent.display_name == "Graph":
                    if name == "nu":
                        self.speak("I'm sorry, you'll need to set up a profile to char
                        Catch=False
                        self.wake_word()
                    else:
                        self.show_graph(df)
                        #df.plot.line(x="Date",y="Weight")
                    #plt.show()
                    Catch=False
                    self.wake_word()
                #THe Update weight intent will update the users profile
                #it will create a new row in the csv so the user can
                #Track their weight over time
                elif response.query_result.intent.display_name == "Update Weight":
                    if name == "nu":
                        self.speak("I'm sorry, you'll need to set up a profile to trac
                        Catch=False
                        self.wake_word()
                    else:
                        self.speak("How many pounds do you weigh?")
                        gab_w=True
                        while gab_w==True:
                            r = sr.Recognizer()
                            with sr.Microphone() as source:
                                audio = r.listen(source)
                                how_w=''
                                try:
                                    how_w=r.recognize_google(audio)
                                    self.update_bot_response("USER:",how_w)

                                except Exception as e:
                                    self.update_bot_response("Exception"+str(e))
                                    how_w="nu"
                            if "exit" in how_w:
                                self.speak("Exiting Now")
                                gab_w= False
                                Catch=False
                                self.wake_word()
                            elif how_w != "nu":
                                #Here we use Chat GPT to pull the number out of a sent
                                #THis allows a more conversational response
                                messages=[]
                                system_content='To extract the number from the sentenc
                                messages.append({"role":"system","content":system_cont
                                messages.append({"role":"user","content":how_w})
                                response=openai.ChatCompletion.create(
                                                model="gpt-3.5-turbo",
                                                messages=messages,
                                                max_tokens=1000,
                                                temperature=0.5)
                                weight=response['choices'][0]['message']['content'].st
                                self.update_bot_response(weight)
                                Dict={"User":[name], "Age":[years], "Weight":[weight],
                                update=pd.DataFrame.from_dict(Dict)
                                df=df.append(update,ignore_index=True)
                                df.to_csv(name+'.csv', index= False)
                                df= pd.read_csv(name+".csv")
                                self.speak("Your profile has been updated and you are
                                gab_w=False
                                Catch=False
                                self.wake_word()
                #Setting a goal allows the user to set a goal weight and a deadline
                elif response.query_result.intent.display_name == "Set a Goal":
                    if name == "nu":
                        self.speak("I'm sorry, you'll need a profile to set goals.")
```

```
                            Catch=False
                        else:
                            self.speak("How much would you like to weigh?")
                            gab_g=True
                            while gab_g==True:
                                r = sr.Recognizer()
                                with sr.Microphone() as source:
                                    audio = r.listen(source)
                                    how_g=''
                                    try:
                                        how_g=r.recognize_google(audio)
                                        self.update_bot_response("USER:",how_g)

                                    except Exception as e:
                                        self.update_bot_response("Exception"+str(e))
                                        how_g="nu"
                                if "exit" in how_g:
                                    self.speak("Exiting Now")
                                    gab_g= False
                                    Catch=False
                                    self.wake_word()
                                elif how_g != "nu":
                                    messages=[]
                                    system_content='To extract the number from the sentenc
                                    messages.append({"role":"system","content":system_cont
                                    messages.append({"role":"user","content":how_g})
                                    response=openai.ChatCompletion.create(
                                                    model="gpt-3.5-turbo",
                                                    messages=messages,
                                                    max_tokens=1000,
                                                    temperature=0.5)
                                    goal=response['choices'][0]['message']['content'].stri
                                    self.update_bot_response(goal)
                                    if "sorry" in goal:
                                        self.speak(goal)
                                        gab_d=False
                                        gab_g=False
                                        Catch=False
                                        self.wake_word()
                                    self.speak("And what date would you like as a deadline
                                    gab_d=True
                                    while gab_d==True:
                                        r = sr.Recognizer()
                                        with sr.Microphone() as source:
                                            audio = r.listen(source)
                                            how_d=''
                                            try:
                                                how_d=r.recognize_google(audio)
                                                self.update_bot_response("USER:",how_d)

                                            except Exception as e:
                                                self.update_bot_response("Exception"+str(
                                                how_d="nu"
                                        if "exit" in how_d:
                                            self.speak("Exiting Now")
                                            gab_d= False
                                            Catch=False
                                            self.wake_word()
                                        elif how_d != "nu":
                                            #here chat gpt formats the date
                                            messages=[]
                                            system_content='You fix the format of dates. 
                                            messages.append({"role":"system","content":sys
                                            messages.append({"role":"user","content":how_
                                            response=openai.ChatCompletion.create(
                                                            model="gpt-3.5-turbo",
                                                            messages=messages,
                                                            max_tokens=1000,
                                                            temperature=0.5)
                                            goal_date=response['choices'][0]['message']['c
                                            if "sorry" in goal_date:
                                                self.speak(text)
                                                gab_d=False
                                                gab_g=False
                                                Catch=False
                                                self.wake_word()
                                            else:
                                                df["Goal"]= goal
                                                df["Goal_Date"]=goal_date
                                                df.to_csv(name+'.csv', index= False)
                                                df= pd.read_csv(name+".csv")
                                                self.speak("Your profile has been updated
                                                gab_g=False
                                                gab_d=False
                                                Catch=False
                                                self.wake_word()
                    #After the email intent the user can send an email if they have gmail
                    elif response.query_result.intent.display_name == "email":
                        if name == "nu":
                            self.speak("I'm sorry, you'll need to set up a profile to set
                            Catch=False
                            self.wake_word()
                        else:
                            self.speak("could I have your email address?")
                            gab_e=True
                            while gab_e==True:
                                r = sr.Recognizer()
                                with sr.Microphone() as source:
                                    audio = r.listen(source)
                                    how_e=''
                                    try:
                                        how_e=r.recognize_google(audio)
                                        self.update_bot_response("USER:",how_e)

                                    except Exception as e:
                                        self.update_bot_response("Exception"+str(e))
                                        how_e="nu"
                                if "exit" in how_e:
                                    self.speak("Exiting Now")
                                    gab_e= False
                                    Catch=False
                                    self.wake_word()
                                elif how_e != "nu":
                                    messages=[]
                                    #here chat gpt formats the email address
                                    system_content='To extract and format the email addres
                                    messages.append({"role":"system","content":system_cont
                                    messages.append({"role":"user","content":how_e})
                                    response=openai.ChatCompletion.create(
                                                    model="gpt-3.5-turbo",
                                                    messages=messages,
                                                    max_tokens=1000,
                                                    temperature=0.5)
                                    email=response['choices'][0]['message']['content'].str
                                    self.update_bot_response(email)
                                    self.speak("Thank you. And could I have the 16 letters
                                    self.update_bot_response("Thank you. And could I have
                                    gab_p=True
                                    while gab_p==True:
                                        r = sr.Recognizer()
                                        with sr.Microphone() as source:
                                            audio = r.listen(source)
                                            how_p=''
                                            try:
                                                how_p=r.recognize_google(audio)
                                                self.update_bot_response("USER:",how_p)

                                            except Exception as e:
                                                self.update_bot_response("Exception"+str(
                                                how_p="nu"
                                        if "exit" in how_p:
                                            self.speak("Exiting Now")
                                            gab_p= False
                                            Catch=False
                                            self.wake_word()
                                        elif how_p != "nu":
                                            pfix=how_p
                                            password=pfix.replace(" ","")
                                            self.update_bot_response(password)
                                            self.speak("Thank you. I am updating your ema
                                            self.update_bot_response("Thank you. I am upda
                                            df["Email"]= email
                                            df["Password"]=password
                                            df.to_csv(name+'.csv', index= False)
                                            df= pd.read_csv(name+".csv")
                                            self.speak("Your profile has been updated and
                                            gab_e=False
                                            Catch=False
                                            self.wake_word()
                    #THe Invite intent will allow gmail user to invite a friend to work o
                    #We learned how to send email from Google Documentation and an articl
                    #by Ashutosh Krishna at
                    #https://www.freecodecamp.org/news/python-project-how-to-build-your-o
                    elif response.query_result.intent.display_name == "Invite":
                        #THe following if statements let the user know if there profile i
                        if name == "nu":
                            self.speak("I'm sorry, you'll need to set up a profile to sen
                            Catch=False
                            self.wake_word()
                        elif EMAIL =="nu":
                            self.speak("I'm sorry, you'll need to update your profile ema
                            Catch=False
                            self.wake_word()
                        elif PASSWORD =="nu":
                            self.speak("I'm sorry, you'll need to update your profile ema
                            Catch=False
                            self.wake_word()

                        else:
                            self.speak("could I have the email address of the person you 
                            gab_f=True
                            while gab_f==True:
                                r = sr.Recognizer()
                                with sr.Microphone() as source:
                                    audio = r.listen(source)
                                    how_f=''
                                    try:
                                        how_f=r.recognize_google(audio)
                                        self.update_bot_response("USER:",how_f)

                                    except Exception as e:
                                        self.update_bot_response("Exception"+str(e))
                                        how_f="nu"
                                if "exit" in how_f:
                                    self.speak("Exiting Now")
                                    gab_f= False
                                    Catch=False
                                    self.wake_word()
```

```
        elif how_f != "nu":
            #Here Chatgpt will format the email address
            messages=[]
            system_content='To extract and format the email addres
            messages.append({"role":"system","content":system_cont
            messages.append({"role":"user","content":how_f})
            response=openai.ChatCompletion.create(
                            model="gpt-3.5-turbo",
                            messages=messages,
                            max_tokens=1000,
                            temperature=0.5)
            recipient=response['choices'][0]['message']['content']
            self.update_bot_response(recipient)
            self.speak("Thank you. What would you like the subject
            self.update_bot_response("Thank you. What would you li
            gab_s=True
            while gab_s==True:
                r = sr.Recognizer()
                with sr.Microphone() as source:
                    audio = r.listen(source)
                    how_s=''
                    try:
                        how_s=r.recognize_google(audio)
                        self.update_bot_response("USER:",how_s)

                    except Exception as e:
                        self.update_bot_response("Exception"+str(e
                        how_s="nu"
                if "exit" in how_s:
                    self.speak("Exiting Now")
                    gab_s= False
                    Catch=False
                    self.wake_word()
                elif how_s != "nu":
                    subject= how_s
                    self.speak("Thank you. And what would you like
                    self.update_bot_response("Thank you. And what
                    gab_m=True
                    while gab_m==True:
                        r = sr.Recognizer()
                        with sr.Microphone() as source:
                            audio = r.listen(source)
                            how_m=''
                            try:
                                how_m=r.recognize_google(audio)
                                self.update_bot_response("USER:",h

                            except Exception as e:
                                self.update_bot_response("Exceptio
                                how_m="nu"
                        if "exit" in how_m:
                            self.speak("Exiting Now")
                            gab_m= False
                            Catch=False
                            self.wake_word()
                        elif how_m != "nu":
                            body=how_m
                            sender=EMAIL
                            password=PASSWORD
                            recipients=[recipient]
                            try:
                                msg= MIMEText(body)
                                msg["Subject"] =subject
                                msg["From"]= sender
                                msg['To']=','.join(recipients)
                                with smtplib.SMTP_SSL('smtp.gmail.
                                        smtp_server.login(sender, pass
                                        smtp_server.sendmail(sender, r
                                        self.speak("Invitation sent")
                                        self.update_bot_response("Invi
                                    self.wake_word()
                                except Exception as e:
                                    self.update_bot_response("Email co
                                    self.wake_word()
#THe Schedule intent will allow users to book a workout on their goog
elif response.query_result.intent.display_name == "Schedule":
    self.speak("because this app is in the test phase you will need to
    self.speak("What would you like the event location to be?")
    self.update_bot_response("What would you like the event location t
    gab_l=True
    while gab_l==True:
        r = sr.Recognizer()
        with sr.Microphone() as source:
            audio = r.listen(source)
            how_l=''
            try:
                how_l=r.recognize_google(audio)
                self.update_bot_response("USER:",how_l)

            except Exception as e:
                self.update_bot_response("Exception"+str(e))
                how_l="nu"
        if "exit" in how_l:
            self.speak("Exiting Now")
            gab_l= False
            Catch=False
            self.wake_word()
        elif how_l != "nu":
            LOCATION= how_l
            self.speak("Thank you. And what would you like the event d
            self.update_bot_response("Thank you. And what would you li
            gab_d=True
            while gab_l==True:
                r = sr.Recognizer()
                with sr.Microphone() as source:
                    audio = r.listen(source)
                    how_d=''
                    try:
                        how_d=r.recognize_google(audio)
                        self.update_bot_response("USER:",how_d)

                    except Exception as e:
                        self.update_bot_response("Exception"+str(e))
                        how_d="nu"
                if "exit" in how_d:
                    self.speak("Exiting Now")
                    gab_d= False
                    Catch=False
                    self.wake_word()
                elif how_d != "nu":
                    DESCRIPTION= how_d
                    self.speak("Thank you. Including the year, what wo
                    self.update_bot_response("Thank you. Including the
                    gab_st=True
                    while gab_st==True:
                        r = sr.Recognizer()
                        with sr.Microphone() as source:
                            audio = r.listen(source)
                            how_st=''
                            try:
                                how_st=r.recognize_google(audio)
                                self.update_bot_response("USER:",how_s
                except Exception as e:
                    self.update_bot_response("Exception"+s
                    how_st="nu"
            if "exit" in how_st:
                self.speak("Exiting Now")
                gab_st= False
                Catch=False
                self.wake_word()
            elif how_st != "nu":
                messages=[]
                system_content='"To format the date and ti
                messages.append({"role":"system","content'
                messages.append({"role":"user","content":h
                response=openai.ChatCompletion.create(
                                model="gpt-3.5-turbo",
                                messages=messages,
                                max_tokens=1000,
                                temperature=0.5)
                STARTfix=response['choices'][0]['message']
                STARTfix=STARTfix.replace("00","0")

                self.update_bot_response(STARTfix)
                #Following code formats the date
                stlist=STARTfix.split()
                #self.update_bot_response("stlist:", stlis
                stnum=[int(i) for i in stlist]
                #self.update_bot_response("stnum:", stnum,
                stfix=datetime(stnum[0],stnum[1],stnum[2],
                #self.update_bot_response("stfix:", stfix,
                STARTTIME=stfix.isoformat()
                self.update_bot_response("STARTTIME:",STAR
                self.speak("Thank you. Including the year,
                self.update_bot_response("Thank you. Inclu
                gab_et=True
                while gab_et==True:
                    r = sr.Recognizer()
                    with sr.Microphone() as source:
                        audio = r.listen(source)
                        how_et=''
                        try:
                            how_et=r.recognize_google(audi
                            self.update_bot_response("USER

                        except Exception as e:
                            self.update_bot_response("Exce
                            how_et="nu"
                    if "exit" in how_et:
                        self.speak("Exiting Now")
                        gab_et= False
                        Catch=False
                        self.wake_word()
                    elif how_et != "nu":
                        messages=[]
                        system_content='"To format the dat
                        messages.append({"role":"system",'
                        messages.append({"role":"user","co
                        response=openai.ChatCompletion.cre
                                        model="gpt-3.5-tur
                                        messages=messages,
                                        max_tokens=1000,
                                        temperature=0.5)
                        ENDfix=response['choices'][0]['mes
                        #The fllow functions format the da
                        ENDfix=ENDfix.replace("00","0")
                        self.update_bot_response(ENDfix)
```

```
etlist=ENDfix.split()
#self.update_bot_response("etlist.
etnum=[int(i) for i in etlist]
#self.update_bot_response("etnum:'
etfix=datetime(etnum[0],etnum[1],e
#self.update_bot_response("etfix:'
ENDTIME=etfix.isoformat()
#self.update_bot_response("ENDTIME
#Now we check if the user has a to
#in the case of first time tester
#For those with a credntials.json
#to sign in through google
creds=None
if os.path.exists('token.json'):
    creds= Credentials.from_author

if not creds or not creds.valid:
    if creds and creds.expired and
        creds.refresh(Request())
    else:
        flow = InstalledAppFlow.fr
        creds = flow.run_local_ser

    with open('token.json','w') as
        token.write(creds.to_json
try:
    service= build('calendar', 'v3
    now = datetime.utcnow().isofor
    #The event formats all the eve
    #The time zone is always US Ea
    #It would be better to let the
    #So it is a limitation
    #We learned how to work with G
    #From Google Documentation.
    event = {
        'summary': 'Work Out',
        'location': LOCATION,
        'description': DESCRIPTION
        'start': {
            'dateTime': STARTTIME,
            'timeZone': 'America/N
        },
        'end': {
            'dateTime': ENDTIME,
            'timeZone': 'America/N
        },
        'attendees': [
            {'email': EMAIL}
        ],
        'reminders': {
            'useDefault': False,
            'overrides': [
                {'method': 'email'
                {'method': 'popup'
            ],
        },
    }

    event = service.events().inser
    self.speak("Your calendar has
    self.update_bot_response('Even
    self.wake_word()
except Exception as e:
    self.speak("I'm sorry I can no
    self.update_bot_response("I'm
    self.update_bot_response("Exce
```

```python
                                                        self.wake_word()


                else:
                    self.speak("did you have a question about nutrition?")

                    Catch=False

#this is a simple function to have the bot ask what it can help with
#This talk was longer before we decided to always return to the wake word
def the_talk(self):
    self.speak("What can I help you with today?")
    self.prime()

#This function will walk the user through setting up a simple profile
#THe user can exit at any stage by using the word exit in a sentence
#We use Chat gpt to extract numbers from sentences so that the user
#can speak to the bot in a conversational manner.
def build_profile(self):
    self.speak( "what is your age?")
    gab_age=True
    while gab_age==True:
        r = sr.Recognizer()
        with sr.Microphone() as source:
            audio = r.listen(source)
            how_old=''
            try:
                how_old=r.recognize_google(audio)
                self.update_bot_response("USER:",how_old)

            except Exception as e:
                self.update_bot_response("Exception"+str(e))
                how_old="nu"
        if "exit" in how_old:
            self.speak("Exiting Now")
            gab_age= False
            self.wake_word()
        elif how_old != "nu":
            messages=[]
            system_content='To extract the number from the sentence, the system will
            messages.append({"role":"system","content":system_content})
            messages.append({"role":"user","content":how_old})
            response=openai.ChatCompletion.create(
                            model="gpt-3.5-turbo",
                            messages=messages,
                            max_tokens=1000,
                            temperature=0.5)
            years=response['choices'][0]['message']['content'].strip()
            self.update_bot_response(years)
            self.speak("Great. How much do you weigh?")
            gab_w=True
            while gab_w==True:
                r = sr.Recognizer()
                with sr.Microphone() as source:
                    audio = r.listen(source)
                    how_w=''
                    try:
                        how_w=r.recognize_google(audio)
                        self.update_bot_response("USER:",how_w)

                    except Exception as e:
                        self.update_bot_response("Exception"+str(e))
                        how_w="nu"
                if "exit" in how_w:
                    self.speak("Exiting Now")
                    gab_w= False
```

```
                        self.wake_word()
                    elif how_w != "nu":
                        messages=[]
                        system_content='To extract the number from the sentence, the syste
                        messages.append({"role":"system","content":system_content})
                        messages.append({"role":"user","content":how_w})
                        response=openai.ChatCompletion.create(
                                        model="gpt-3.5-turbo",
                                        messages=messages,
                                        max_tokens=1000,
                                        temperature=0.5)
                        weight=response['choices'][0]['message']['content'].strip()
                        self.update_bot_response(weight)
                        today= date.today()
                        self.speak( "Thank you. I will build your profile now.")
                        self.update_bot_response("Thank you. I will build your profile now
                        Dict={"User":[name], "Age":[years], "Weight":[weight], "Date":[to
                        df=pd.DataFrame.from_dict(Dict)
                        df.to_csv(name+'.csv', index= False)
                        self.speak("Okay you are all good to go!")
                        self.update_bot_response("Okay you are all good to go!")
                        self.the_talk()

    #This function allows the user to decide whether or not they want
    #to build a profile
    def decide(self):
        self.speak("I don't think we've met before, would you like to create a profile?")
        gab=True
        while gab==True:
            r = sr.Recognizer()
            with sr.Microphone() as source:
                audio = r.listen(source)
                pq=''
                try:
                    pq=r.recognize_google(audio)
                    self.update_bot_response("USER:",pq)

                except Exception as e:
                    self.update_bot_response("Exception"+str(e))
                    pq="nu"
            if "exit" in pq:
                self.speak("Exiting Now")
                pq="nu"
                gab=False
                self.wake_word()
            if "no" in str(pq):
                global name
                name="nu"
                self.speak("Okay. But not all features will be available until you set up
                self.update_bot_response("Okay. But not all features will be available unt
                self.the_talk()
            if "yes" in str(pq):
                self.build_profile()
                time.sleep(1)

    #This function checks if the user has a profile. If they do the bot
    #moves onto the primary function, if thy do not, the bot will move
    #to the decide function and let them choose whether or not to make
    #a profile.
    def intro(self):
        global exit_flag
        self.speak("Can I have your name")
        chat=True
        while chat==True:
            r = sr.Recognizer()
            self.update_bot_response("Listening...")
            with sr.Microphone() as source:
                audio = r.listen(source)
                global name
                name=''
                try:
                    name=r.recognize_google(audio)
                    self.update_bot_response("USER:",name)

                except Exception as e:
                    self.update_bot_response("Exception"+str(e))
                    name="nu"
            if "exit" in name.lower():
                self.speak("Exiting Now")
                #exit_flag = True
                name="nu"
                self.wake_word()
            elif name != "nu":
                messages=[]
                system_content='To extract the name from the sentence, the system will use
                messages.append({"role":"system","content":system_content})
                messages.append({"role":"user","content":name})
                response=openai.ChatCompletion.create(
                                model="gpt-3.5-turbo",
                                messages=messages,
                                max_tokens=1000,
                                temperature=0.5)
                name=response['choices'][0]['message']['content'].strip()
                self.update_bot_response(name)
                self.update_bot_response("checking for profile...")
                path=(name+".csv")
                check_file = os.path.isfile(path)
                if str(check_file)=="True":
                    self.update_bot_response("It's great to talk to you again")
                    self.speak("It's great to talk to you again")
                    self.the_talk()
                if check_file == False:
                    self.decide()



    def wake_word(self):
        #global exit_flag
        #we used version 1.9.5 because you do not need an api key and it is free. we lear
        #how to use it from tutorials on youtube, particularly,
        #https://www.youtube.com/watch?v=i7kF6EjrYW0&list=PLI5RX9MkxrmIv-q7AFTb1tvwLX3gzG
        porcupine= None
        pa= None
        audio_stream= None

        self.update_bot_response("Just say blueberry when you need me!")

        try:
            porcupine = pvporcupine.create(keywords=["blueberry"])
            pa = pyaudio.PyAudio()
            audio_stream = pa.open(
                            rate=porcupine.sample_rate,
                            channels=1,
                            format=pyaudio.paInt16,
                            input=True,
                            frames_per_buffer=porcupine.frame_length)
            #print("helooooooo")
            while True:
                #if exit_flag:
                    #break
                pcm = audio_stream.read(porcupine.frame_length)
                pcm = struct.unpack_from("h" * porcupine.frame_length, pcm)
                keyword_index = porcupine.process(pcm)
                #print("hhhhhhhhhhhhhhhhhhhhhhhhhhhhhh")
                if keyword_index >= 0:
                    self.update_bot_response("Hello")
                    self.speak("Hello")
                    self.intro()
                    time.sleep(1)

        finally:
            if porcupine is not None:
                porcupine.delete()
    def activate_chatbot(self):
            # Call the wake_word function to start the chatbot
            self.wake_word()  # Replace with your self.wake_word() function
           # self.user_entry_var.set(user_input)
            #self.bot_entry_var.set(chatbot_response)
    def update_bot_response(self, *args):
        text = ' '.join(map(str, args))  # Convert all arguments to strings and join them
        self.bot_response.insert(tk.END, text + "\n")
        self.bot_response.yview(tk.END)
        self.root.update_idletasks()

  # Force an update of the GUI
```

In [ ]:

```
# Code to run the GUI
if __name__ == "__main__":
    root = tk.Tk()
    gui = ModifiedChatbotGUI(root)
    root.mainloop()
```

In [ ]: