UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Ajutor vocal de navigare online

propusă de

Rareș Arvinte

Sesiunea: Iulie, 2018

Coordonator științific

Profesor Doctorand Dan Cristea

UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI

FACULTATEA DE INFORMATICĂ

Ajutor vocal de navigare online

Rareș Arvinte

Sesiunea: Iulie, 2018

Coordonator științific

Profesor Doctorand Dan Cristea

Avizat,

Îndrumător Lucrare de Licenţă

Titlul, Numele şi prenumele _____

Data _____ Semnătura _____

DECLARAȚIE privind originalitatea conţinutului lucrării de licenţă

Subsemntatul(a) ……

domiciliul în ……..

născut(ă) la data de ……., identificat prin CNP ……., absolvent(a) al(a) Universităţii „Alexandru Ioan Cuza" din Iaşi, Facultatea de …….. specializarea ……., promoţia ……., declar pe propria răspundere, cunoscând consecinţele falsului în declaraţii în sensul art. 326 din Noul Cod Penal şi dispoziţiile Legii Educaţiei Naţionale nr. 1/2011 art.143 al. 4 si 5 referitoare la plagiat, că lucrarea de licenţă cu titlul: _____

_____

_____elaborată sub îndrumarea dl. / d-na _____, pe care urmează să o susţină în faţa comisiei este originală, îmi aparţine şi îmi asum conţinutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licenţă să fie verificată prin orice modalitate legală pentru confirmarea originalităţii, consimţind inclusiv la introducerea conţinutului său într-o bază de date în acest scop.

Am luat la cunoştinţă despre faptul că este interzisă comercializarea de lucrări ştiinţifice in vederea facilitării fasificării de către cumpărător a calităţii de autor al unei lucrări de licenţă, de diploma sau de disertaţie şi în acest sens, declar pe proprie răspundere că lucrarea de faţă nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Dată azi, …… Semnătură student ……

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul *Ajutor vocal de navigare online*, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza" din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iaşi, *data*

Absolvent *Rareş Arvinte*

_____

(semnătura în original)

Cuprins

Introducere

1.1 Motivația alegerii temei

În lume există persoane cu deficiențe de vedere, acestea fiind cauzate de diferiți factori neurologici sau fiziologici. În secolul 21, odată cu dezvoltarea rapidă a internetului, a apărut nevoia creării condițiilor necesare pentru ca aceste persoane să poată folosi noile tehnologii care chiar pot să îi ajute să se integreze mai ușor în societate și de asemenea să le faciliteze viața.

Această aplicație dorește să ofere persoanelor cu probleme de vedere o șansă de a avea acces la noile tehnologii, mai exact cele din mediul online fără a avea nevoie de o tastatură specială în alfabetul braille. Utilizatorii vor avea prin intermediul vocii acces atât la anumite facilități oferite de *Google* cât și la fișiere din calculatorul personal. Utilizatorul va putea accesa *e-mailul*, asculta muzică, va putea primi definiții pentru cuvinte din dicționar, va avea acces la *Google Drive* și multe altele. Astfel un utilizator va transmite o comandă vocală către aplicație iar aplicația va accesa motorul de căutare *Google* în vederea obținerii informațiilor dorite.

Ceea ce m-a motivat să aleg această temă este în primul rând dragostea mea pentru inteligența artificială, mereu mi-au plăcut roboții și ideea de a-ți putea dezvolta propria mașină care să te ajute sau să iți fie ca un prieten. În esență omul este o ființă socială care are nevoie în permanență de cineva cu care să comunice sau să practice anumite activități în prezența ori folosindu-se de cineva sau ceva.

1.2 Obiectivele generale ale lucrării

În această lucrare se urmărește prezentarea aplicației dezvoltate de mine prin descrierea acesteia, explicarea aplicabilității acesteia în societate și în viitoare proiecte. De asemenea se urmărește nuanțarea tehnologiilor folosite în crearea acesteia și posibile idei de dezvoltare în viitor.

Prin descrierea aplicației se va urmării detalierea funcționalităților acesteia, tehnologiile folosite din mediul *open-source* și limbajul folosit, acesta fiind *Python*. Se dorește de asemenea să se aducă la cunoștință cititorilor acestei lucrări existența unor tehnologii folosite pentru prelucrare vocala care vor putea fi folosite în viitor în dezvoltarea unor aplicații cu aceași tematică.

1.3 Propunere de metodologie

Pentru a crea această aplicație va fi folosit limbajul de programare *Python* și câteva dintre bibliotecile aferente acestuia precum: *Pyttsx3, SpeechRecognition, WebBrowser, PyDictionary, UrlLib, ImapLib* etc. De asemenea, în urma *research*-ului făcut pentru alegerea librăriilor necesare, le-am ales pe acelea compatibile cu ultima versiune de *Python* la momentul respectiv (*Python 3.6.4).* Funcționalitatea de bază a proiectului este *Speech-to-Speech* aceasta fiind oferită de cele două biblioteci: *SpeechRecognition* si *Pyttsx3,* ce fac conversia voce-text respectiv text-voce. Alte funcționalități precum *web-surfing,* dicționar și acces la *e-mail* vor fi implementate cu ajutorul bibliotecilor *WebBrowser, PyDictionary, UrlLib, ImapLib* și *email* care împreună oferă și accesul la conținutul de pe *Youtube.*

1.4 Structura lucrării

Lucrarea este structurată pe cinci părți. Prima parte este descrierea aplicației din punct de vedere tehnic. Aici se regăsesc descrierile metodelor implementate. În următoarea parte sunt enumerate tehnologiile folosite și scopul lor în proiect. După care se continuă cu o concluzie personală cu privire la contribuția mea la acest proiect. Lucrarea se sfârșește cu bibliografia și anexele proiectului.

Descrierea aplicației

2.1 Descriere funcțională

Aplicația dezvoltată de mine reprezintă o modalitate prin care perosanele cu dezabilități de vedere pot interacționa în mediul online. Aplicația rulează în background neavând o interfață grafică, acesta funcționând ca un majordom. La prima utilizare, *user*-ul este rugat să urmeze anumiți pași de configurare ai aplicației. După aceste configurări aplicația este pregătită pentru a fi utilizată, *user*-ul având posibilitatea de a efectua următoarele acțiuni: acces la pagini web, dicționar, *e-mail*-uri sau *Youtube*.

Fiecare dintre aceste servicii poate fi apelat printr-un cuvânt cheie rostit de utilizator. În funcție de serviciul ales aplicația îl va îndruma pe utilizator către următorul pas.

În cazul în care utilizatorul dorește oprirea aplicației acesta va folosi un cuvânt cheie (spre exemplu *quit*) în momentul în care aplicația este în *stand-by*.

2.2 Configurare Aplicație

În urma primei execuții a aplicației utilizatorului i se solicită configurarea aplicației cu anumite date cu caracter personal. În primă instanță i se va solicita introducerea unui nume de utilizator pentru ca aplicația să îl poată recunoaște. Ulterior i se vor cere informații referitoare la accesarea contului *Google* spre exemplu o addresă de *e-mail* și parolă, doar în cazul în care utilizatorul va dori să acceseze prin intermediul aplicației date cu caracter personal.

2.3 Caracteristica de rulare în *background*

Aplicația rulează în background independent de alte aplicații ea îndeplinind acțiuni prin intermediul unor comenzi date de utilizator. La întâlnirea unei comenzi de tipul *quit* aplicația se va închide, iar la întâlnirea unei comenzi de tip *pause* aceasta va continua rularea fără a îndeplinii comenzile viitoare. Aplicația va putea fi repornită prin utilizarea comenzii *start*. Am ales această abordare pentru a nu permite aplicației să interfereze cu alte conversații ale utilizatorului.

2.4 Căutarea în online

Aplicația are optiunea de interogare a unui motor de căutare, opțiune ce poate fi selectată doar în cazul în care aplicația este într-o stare activă. Această opțiune poate fi selectată cu ajutorul cuvântului cheie *search*, utilizatorul fiind nevoit apoi să introducă datele căutării. Aplicația va accesa primul rezultat oferit de motorul de căutare și va oferi de pe respectiva pagină un rezumat al acesteia dacă este posibil. Dacă rezumatul nu a putut fi efectuat vor fi accesate pagini până la realizarea acestuia sau până la o limită predefinită de căutare. (Pentru un exemplu de cod a se consulta Anexa 1)

2.5 Căutarea cuvintelor în dicționar

Aplicația oferă posibilitatea de a căuta definiții intr-un dicționar online, citind utilizatorului definiția oferită. Această opțiune poate fi selectată cu ajutorul cuvântului cheie *dex,* utilizatorul fiind nevoit apoi să introducă datele căutării. În cazul în care utilizatorul rostește un singur cuvânt, aplicația se va folosi de *PyDictionary* pentru a oferi utilizatorului definiția pentru cuvântul cerut. În cazul în care utilizatorul introduce mai mult de un cuvânt, aplicația va apela funcția de căutare în online pentru o definire cât mai precisă. (Pentru un exemplu de cod a se consulta Anexa 2)

2.6 Acces Youtube

Dat fiind că oamenii cu dezabilități vizuale folosesc ca principal organ de simț auzul vor fi încântați să utilizeze *YouTube*-ul direct din aplicație. În vederea soluționării acestei nevoi aplicația caută și redă videoclipuri de pe *YouTube* în concordanță cu comanda primită. În cazul în care videoclipul nu este cel căutat utilizatorul are posibilitatea de a selecta următorul videoclip din rezultatul căutării.

Aplicația oferă posibilitatea de a reda un *video* de pe *Youtube* în *browser*-ul prestabilit. Această opțiune poate fi selectată cu ajutorul cuvântului cheie *Youtube,* utilizatorul fiind nevoit apoi să introducă datele căutării. În urma introducerii parametrilor de căutare, aplicația reține rezultatele căutării într-un vector, redând primul rezultat utilizatorului. Utilizatorul se poate folosi de cuvinte cheie precum *next* sau *back* pentru a se întoarce în lista de rezultate din vector și a selecta pentru redare următorul video. (Pentru un exemplu de cod a se consulta Anexa 3)

2.7 Acces E-mail

În urma configurărilor efectuate de utilizator, dacă acesta a optat pentru accesul la *e-mail,* aplicația îi poate oferi această facilitate. În urma conectării la *gmail* aplicația îi va putea citi utilizatorului orice mesaj dorește sau îi va oferi posibilitatea de a comunica prin intermediul poștei electronice.

Aplicația oferă posibilitatea de a accesa *e-mail*-uri prin intermediul contului de *gmail*. Această opțiune poate fi selectată cu ajutorul cuvântului cheie *e-mail,* aplicația oferind o listă de *e-mail*-uri necitite ce va fi citită utilizatorului. (Pentru un exemplu de cod a se consulta Anexa 4)

Metode Utilizate

În creearea acestui proiect s-a folosit *Python*. Am optat pentru acest limbaj deoarece este nou, codul este simplificat și se pot construi cu el în special aplicații de dimensiuni reduse.

De asemenea *Python* suportă programarea orientată obiect, imperativă, funcțională sau procedurală. Sistemul de tipizare este dinamic iar administrarea memoriei decurge automat prin intermediul unui serviciu „gunoier" (*garbage collector*). Un alt avantaj al limbajului este existența unei ample biblioteci standard de metode (*Downey, Allen B., 2012*).

3.1 Biblioteca Pyttsx3

Pentru vocea care răspunde cererilor utilizatorului s-a folosit biblioteca *Pyttsx3*. Această bibliotecă din *python* preia cuvintele ca *stringuri* și le pasează la *engine*-ul *pyttsx* care face conversie *text-to-speech*. Prin comanda *say* se trimite la *output stringul*, acesta fiind comunicat vocal. Când *Pyttsx3* primește o cerere se invocă funcția *init*() care primește o referință la o instanță a *engine*-ului. *Engine*-ul creeaza un obiect *DriverProxy* responsabil pentru conversia *text-to-speech*. Se poate specifica ce *driver* audio să se folosească în funcție de sistemul de operare. Dacă nu este specificat se va încerca prin *default* să se aleagă cel mai potrivit *driver*. Momentan există trei *drivere*. Unul dintre acestea este *SAPI5* pentru *Windows*, altul este *NSSSpeechSynthesizer* pentru *Mac OS* și unul compatibil pe orice platformă numit *eSpeak*. Biblioteca vine și cu posibilitatea de a customiza vocea de răspuns sau rata de răspuns (cuvinte pe minut). Se pot creea de asemenea proprietăți în cazul în care se dorește creearea unui dialog între personaje. (*Natesh M Bhat. , 2017*)

3.2 Biblioteca de recunoaștere vocală

Biblioteca *Speech Recognition* a fost folosită pentru recunoaștere vocală. În momentul interacțiunii utilizatorului cu programul acesta preia conținutul vocal și îl convertește într-un *string*. Pentru a folosi biblioteca este recomandat să avem acces la un microfon, *bind*-ul microfonului facându-se cu ajutorul bibliotecii *PyAudio (Hubert Pham, 2006).* Astfel, se instanțiază un nou microfon cu comanda *Microphone(),* iar în cazul în care nu se știe ce fel de microfon se folosește sau avem mai multe microfoane, se invocă funcția *list_microphone_names()* (ce returnează o listă cu toate microfoanele conectate la dispozitiv) și programul va alege următorul microfon din listă până va găsi unul compatibil. Pentru a functiona, aplicația are nevoie de un *recognizer,* ce se instanțiază prin funcția *Recognizer(),* acesta având rolul de a recunoaște sunetele. La această fucție se pot face o multitudine de setări pentru a captura cât mai bine și mai clar sunetul. În momentul rulării aplicației se instanțiază microfonul. Cât timp acesta capteaza *input* audio, *recognizer*-ul capturează sunetele folosind funcția *listen()*. Datele audio sunt pasate unei funcții *recognize_google()* care convertește aceste date într-un format potrivit pentru a face request-uri la *google*. Biblioteca poate fi folosită și în scopul de a crea fișiere audio, care pot fi prelucrate mai târziu, sau pentru înregistrare vocală. De asemenea, biblioteca, pe lângă faptul că face *request*-uri la *google*, oferă posibilitatea de a lucra cu *Microsoft Bing Speech API, Wit.ai API, Google cloud Speech API, Houndify API* sau *IBM Speech to Text API*. (Anthony.Zhang, Dec 5, 2017*)*

3.3 Biblioteca WebBrowser

WebBrowser este o bibliotecă *python* folosită pentru deschiderea de pagini web. Funcția *open()* primește un *URL* ca date de intrare și pe baza acestuia se accesează pagina respectivă. Această bibliotecă este utilă în momentul în care dorim să accesăm rapid o pagină web. Înainte de a folosi biblioteca este necesar să specificăm ce fel de *webbrowser* folosim, acest lucru realizându-se prin funcția *get()* căreia i se atribuie un *string* cu *PATH*-ul unde se află instalat *webbrowserul*. (sursa???)

3.4 Biblioteca PyDictionary

PyDictionary este o bibliotecă ce oferă acces la un dicționar. Aceasta conține funcții pentru a căuta și oferi definiții, sinonime, antonime sau traduce cuvinte în alte limbi. Avem funcția *meaning()* care returnează definiția unui cuvânt , funcția *synonim()* care returnează sinonimele unui cuvânt dat ca și *input*, funcția *antonym()* care returnează antonimele unui cuvânt și funcția *translate()* care primește doi parametri, primul fiind cuvâmtul de tradus iar al doilea este limba în care să se traducă. Rezultatele sunt returnate în format *JSON* din bazele de date ce rulează pe servere de la *Red Hat*. Datele de *input* sunt transmise vocal de către utilizator și sunt apoi convertite în text cu ajutorul bibliotecii *SpeechRecognition*. Datele de *output* sunt transmise vocal utilizatorului prin intermediul bibliotecii *Pyttsx3*. (Pradipta Bora, 2014)

3.5 Biblioteca UrlLib

UrlLib este un pachet ce conține mai multe module ce lucrează cu *URL*-uri. (Python Software Foundation, ultimul update Jun 17, 2018)

1.5.1 UrlLib request

Modulul de *request* pune la dispoziție o colecție de funcții și clase în scopul de a deschide *URL*-uri. Funcția *urlopen()* primește ca parametru un *string* și deschide resursa de la adresa specificată. De asemenea se pun la dispoziție funcții pentru gestionarea resursei în cauză. (Python Software Foundation, ultimul update Jun 17, 2018)

1.5.2 UrlLib error

Acest pachet este folosit pentru a detecta eventualele erori apărute în urma execuției unei funcții din *UrlLib request*. Conține funcția *URLError()* care returnează motivul erorii sub forma unei instanțe din clasa *URLError*. (Python Software Foundation, ultimul update Jun 17, 2018)

1.5.3 UrlLib parse

Această bibliotecă se ocupă de manipularea *string*-urilor date ca *input* sau a conținutului propriu-zis al unui *URL*. De asemenea ajută la identificarea conținutului unui *URL*. Funcția *urlparse()* sparge *URL*-ul in fragmente și comunică detalii despre acestea cum ar fi schema addresei, *username* și parolă, dacă acestea au fost specificate în *URL* și multe altele. Această bibliotecă este utilă deoarece putem codifica și decodifica *URL*-uri ceea ce este necesar când se fac spre exemplu *request*-uri la *YouTube* sau când accesăm un *URL* și vrem să vedem ce semnifică codificarea sa. Acestea se realizează cu funcțiile *urlencode()* și *urldecode()*. (Python Software Foundation, ultimul update Jun 17, 2018)

3.6 Biblioteca ImapLib

Scopul acestei biblioteci este de a realiza comunicarea cu adresa de *e-mail* a utilizatorului. Biblioteca *imaplib* vine cu trei clase: *IMAP4*, *IMAP4_SSL* și *IMAP4_stream*. Clasa *IMAP4* implementează protocolul *imap4* care creează o conexiune având ca date de intrare *host*-ul și *port*-ul (*port*-ul este opțional).

IMAP4_SSL este asemănătoare cu *IMAP4* doar că acesta realizează conexiunea peste un *socket* criptat *SSL*. La fel se furnizează numele *host*-ului și *port*-ul (dacă *port*-ul este omis se va conecta la *port*-ul standard *IMAP-over-SSL 993*), iar opțional se poate specifica o cheie privată sau un certificat pentru conexiune *SSL*, accestea asigurând securitatea transmiterii datelor.

IMAP4_stream, la fel ca *IMAP4_SSL*, este o subclasă a clasei *IMAP4* al cărui scop este de a crea o conexiune *stdin/stdout*.

După realizarea unei conexiuni funcția *login()* primește ca *input* un nume de utilizator și o parolă și încearcă să ofere accesul clientului la *e-mail*-urile sale. Se recomandă folosirea funcției *login_cram_md5()* pentru a asigura securitatea datelor cu caracter personal. Biblioteca conține de asemenea o multitudine de funcții pentru selectarea, citirea, scrierea și trimiterea mesajelor. Mesajele pot fi selectate prin mai multe moduri, de la selectarea întregului *inbox* până la căutarea unui anumit cuvânt cheie. Dacă utilizatorul dorește să citească un mesaj, acesta va fi selectat cu un procedeu de selecție, apoi va fi redat vocal de către program. (Python Software Foundation, ultimul update Apr 29, 2018)

Concluzii

4.1. Contribuții originale

În urma cercetărilor efectuate în domeniul *speech recognition* am ajuns la concluzia că acestui tip de proiect nu i s-a acordat destulă atenție datorită lipsei resurselor din mediul *open-source* dar și a lipsei de cerere pe piață. Acest lucru m-a determinat să dezvolt un punct de plecare pentru viitoare proiecte *open-source*.

În mediul *online* se găsesc diferite biblioteci ce se ocupă spre exemplu de recunoașterea *imput*-ului vocal, convertirea acestuia în *string*-uri și căutarea *web*. Contribuția mea constă în îmbinarea acestor elemente avînd ca scop efectuare de *task*-uri în mediul *online* făcând redundantă folosirea tastaturii.

Ca prim pas, am început prin a implementa o metodă de căutare pe *google*, având ca *input* un *string* primit vocal. Ulterior am adăugat o funcție care accesează primul *link* și returnează rezumatul conținutului de pe acea pagină. Am propus o abordare noua, diferită și simplistă demnă de luat în vedere în momentul realizării unui proiect cu aceași tematică.

4.2 Dezvoltări ulterioare

Pe viitor îmi propun să dezvolt noi facilități ce vor reduce interacțiunea fizică a utilizatorului cu mașina.

O idee ar fi implementarea unei opțiuni pentru adăugarea de evenimente sau notificarea utilizatorului despre viitoarele evenimente din calendarul *google*.

O altă funcționalitate ce se dorește a fi adăugată este interacțiunea cu mediul *cloud*. La recunoșterea anumitor cuvinte cheie, aplicația va încărca sau descărca fișiere din *google drive*.

Cât despre partea de *message sharing* îmi propun să implementez o funcție ce va permite scrierea și trimiterea de mesaje.

Ca să trag o concluzie la ceea ce am explicat mai sus, aș spune că în viitor de pe urma acestei aplicații vor beneficia spre exemplu persoanele cu dizabilități de ordin fizic. Totodată, aceasta eficientizează procesul de căutare prin faptul că înlocuiește timpul necesar scrierii unei căutări cu echivalentul său oral.

Pe viitor ar fi interesant ca o mare majoritate a populației să folosească dispozitive cu interacțiune vocală astfel reducând folosirea textului și permițând utilizatorului să acorde o atenție sporită mediului înconjurător. (Voxtab, Octombrie 15, 2015)

Pentru cercetări viitoare aș recomanda studiere de librării în *python* mai ales cele din mediul *open-source* dar și studierea posibilități de a oferi suport pentru limba română.

Bibliografie

- Downey, Allen B. (May 2012). Think Python: How to Think Like a Computer Scientist (Version 1.6.6 ed.). ISBN 978-0-521-72596-5. Disponibil la http://www.greenteapress.com/thinkpython/html/
- Natesh M Bhat. (2017). Pyttsx3 - Text-to-speech x-platform. Disponibil la https://pyttsx3.readthedocs.io/en/latest/
- Anthony Zhang, Arvind Chembarpu, Kevin Smith, Kamus Hadenes, Sarah Braden, Bohdan Turkynewych, Steve Dougherty, Broderick Carlin (Dec 5, 2017). Speech recognition module for Python, supporting several engines and APIs, online and offline. Disponibil la https://github.com/Uberi/speech_recognition#readme
- Hubert Pham (2006). PyAudio Documentation. Disponibil la http://people.csail.mit.edu/hubert/pyaudio/docs/
- Python Software Foundation (Ultimul update Apr 29, 2018). Convenient Web-browser controller https://docs.python.org/2/library/webbrowser.html
- Pradipta Bora (2014), (ultima versiune Mar 10, 2015). PyDictionary. Disponibil la https://pypi.org/project/PyDictionary/
- Python Software Foundation (Ultimul update Jun 17, 2018). URL handling modules. Disponibil la https://docs.python.org/3/library/urllib.html
- Python Software Foundation (Ultimul update Jun 17, 2018). Extensible library for opening URLs. Disponibil la https://docs.python.org/3/library/urllib.request.html#module-urllib.request
- Python Software Foundation (Ultimul update Jun 17, 2018). Exception classes raised by urllib.request. Disponibil la https://docs.python.org/3/library/urllib.error.html#module-urllib.error
- Python Software Foundation (Ultimul update Jun 17, 2018). Parse URLs into components. Disponibil la https://docs.python.org/3/library/urllib.parse.html#module-urllib.parse
- Python Software Foundation (Ultimul update Apr 29, 2018). IMAP4 protocol client. Disponibil la https://docs.python.org/2/library/imaplib.html
- Voxtab (Octombrie 15, 2015). Voice Recognition Software: Pros and Cons. Disponibil la https://www.voxtab.com/transcription-blog/voice-recognition-software-pros-and-cons/

Anexe

Anexa 1

```
def search_google():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        engine = pyttsx3.init()
        mesaj_question = "What do you wish to search?"
        engine.say(mesaj_question)
        print(mesaj_question)
        engine.runAndWait()
        r.adjust_for_ambient_noise(source, duration=1)
        audio = r.listen(source, phrase_time_limit=5)
    try:
        webbrowser.get(google_api).open_new(url + r.recognize_google(audio))
    except:
        pass
```

Anexa 2

```
def cauta_dex():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        engine = pyttsx3.init()
        mesaj_question = "What do you wish to search?"
        engine.say(mesaj_question)
        print(mesaj_question)
        engine.runAndWait()
        r.adjust_for_ambient_noise(source, duration=1)
        audio = r.listen(source, phrase_time_limit=5)
        message = r.recognize_google(audio)
    if lungime_text(message) == 1:
        try:
            engine = pyttsx3.init()
            bot_response = dictionary.meaning(message)
            print(bot_response)
            if not bot_response:
                webbrowser.get(google_api).open_new(url + message)
            else:
                engine.say(bot_response)
                print(bot_response)
                engine.runAndWait()
        except (IndexError, ValueError):
            webbrowser.get(google_api).open_new(url + message)
    elif lungime_text(message) > 1:
        try:
            webbrowser.get(google_api).open_new(url + r.recognize_google(audio))
        except:
            pass
```

Anexa 3

```
def youtube_helper():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("Ce doriti sa cautati?")
        r.adjust_for_ambient_noise(source, duration=1)
        audio = r.listen(source, phrase_time_limit=5)

        query_string = urllib.parse.urlencode({"search_query": audio})
        html_content = urllib.request.urlopen(url2 + r.recognize_google(audio))
        search_results = re.findall(r'href=\"\/watch\?v=(.{11})', html_content.read().decode())
        print(search_results)
    try:
        webbrowser.get(google_api).open_new(url3 + search_results[0])
    except:
        pass
```

Anexa 4

```
result, data = mail.uid('search', None, "(UNSEEN)")
i = len(data[0].split()) # data[0] is a space separate string
for x in range(i):
    latest_email_uid = data[0].split()[x]

    result, email_data = mail.uid('fetch', latest_email_uid, '(RFC822)')

    raw_email = email_data[0][1]

    raw_email_string = raw_email.decode('utf-8')

    email_message = email.message_from_string(raw_email_string)

    for part in email_message.walk():
        if part.get_content_type() == "text/plain":  # ignore attachments/html
            body = part.get_payload(decode=True)
            print(body)
mail.logout()
```