Mediul de lucru

* Am instalat Python versiunea 3.13 de pe site-ul <https://www.python.org/downloads/release/python-3130/>.
* Ca si browser voi folosi Chrome. Am intrat in setari pentru a vedea ce versiune de chrome am. Dupa verificarea versiunii am intrat pe site-ul <https://sites.google.com/chromium.org/driver/> petru a cauta driver-ul pentru versiunea corecta si l-am descarcat.
* Am adaugat in path variables path-ul ‘C:\\Users\\marac\\AppData\\Local\\Programs\\Common’ unde am adaugat chrome driver-ul pentru a fii accesibil programelor.
* Am descarcat Anaconda si mi-am facut un environment unde am instalat selenium si librariile necesare folosind urmatoarele comenzi:
  + pip install selenium
  + pip install numpy
  + pip install opencv-python
  + py -m pip install pyautogui
  + pip install PyAudio
* Am descarcat Pycharm ca mediu de lucru si am creat un proiect folosind environment-ul creat mai sus.
* Am creat un repository pe GitHub cu numele “AROBS\_Internship” unde voi incarca progresul proiectului.
* In timpul dezvoltarii programului am instalat libraria moviepy.editor pentru a da merge la cele doua inregistrari, audio si video, dar si audioop pentru calcularea rms-ului unui videoclip. Instalarea s-a realizat din IDE-ul PyCharm.

Analiza situatiilor/problemelor neprevazute

* Lipsa conexiunii la internet: Programul trebuie să verifice dacă există o conexiune la internet și să încerce din nou in cazul in care nu s-a reusit conexiunea din prima. Programul trebuie să înregistreze eroarea în fișierul de log.
* Pop-up-uri neprevazute pe YouTube: YouTube afișează uneori pop-up-uri precum acorduri de cookie-uri, conectare la cont sau sugestii de abonare la YouTube Premium. Programul trebuie sa gestioneze aceste pop-up-uri inchizandu-le pentru a putea contiunua cu rularea videoclipului. Programul va înregistra pop-up-ul si inchiderea acestuia în fișierul de log.
* Gestionarea reclamelor din videoclipuri: Youtube poate reda reclame la inceputul unui videoclip. In acest caz programul trebuie sa astepte terminarea timpului de asteptare pana la posibilitatea de a da skip reclamei si sa dea skip sau sa astepte terminarea automata a reclamei.

Documentatia programului

Proiectul consta in 5 fisiere, main-ul de unde gestionam programul si 4 clase.

Clasa Logger

Clasa Logger are un constructor si doua funtii.

Constructorul creaza fisierul in care se vor scrie log-urile, fisierul este de tip ‘write’, acesta va rescrie continutul, nu il va adauga la cel existent.

Functia log\_message primeste ca argument un mesaj apoi ia timpul curent si scrie in fisier timpul si mesajul primit.

Functia close\_file inchide fisierul.

Clasas ScreenRecorder

Aceasta clasa este dedicata inregistrarii ecranului. Clasa contine un constructor si doua functii.

Constructorul este ccel care creaza fisierul, videoclipul propriu zis cu specificatiile setate de mine.

Functia record\_for\_duration primeste ca parametru lungimea Dorita a videoclipului. Ne vom utiliza de 3 parametrii,

* finish\_time – timpul la care trebuie sa se termine inregistrarea
* current\_time – timpul current
* last\_screenshot\_time – timpul la care s-a efectuat ultimul screenshot

Se calculeaza finis\_time si se initializeaza current\_time si last\_screenshot\_time cu timpul current si ce incepe procesul de inregistrare.

Cat timp current\_time este mai mic decat finish\_time se calculeaza cat timp a trecut de cand s-a facut ultimul screenshot si se verifica daca este mai mare decat frecventa pe care am calculate-o. Frecventa v-a repezenta 1/ numarul de fps-uri, in cazul nostrum 20. Daca diferenta este mai mare decat frecventa atunci se va face un nou screenshot, se va scrie screenshot-ul in fisierul videoclipului si se vor actualiza timpii.

Functia start\_recording primeste si ea ca parametru lungimea dorita a videoclipului. Functia are rolul de a crea thread-ul ce va gestiona inregistrarea ecranului.

Clasa AudioRecorder

Aceasta clasa contine metodele ce se vor ocupa de inegistrarea audio, dar si de calcularea nivelului de sunet in dB. Clasa contine un constructor si doua functii si este similara cu clasa ScreenRecorder.

Constructorul este cel care creaza fisierul audio dupa specificatiile alese de mine.

Functia record\_for\_duration primeste ca parametru durata pentru care vrem sa inregistram. Ea verifica cat timp timpul current este mai scurt decat timpul la care trebuie sa se termine inregistrarea si citeste datele de pe cate un chunk scriindu-le apoi in fisierul audi creat. De asemenea citeste nivelul rms de pe fiecare chunk si incrementeaza un contor de chunk-uri pentru a putea face media ulterior. Dupa ce media e calculate se stocheaza in proprietatea obiectului. La final se scriu datele si se inchide fisierul.

Functia start\_recording este cea care creaza thread-ul ce va gestiona inregistrarea audio.

Clasa AudioVideoMerger

Aceasta clasa se foloseste de libraria moviepy.editor pentru a da merge celor doua inregistrari. Clasa are un constructor si doua functii.

Constructorul initializeaza path-urile si fisierele necesare. Functia merge\_video\_with\_audio este cea care da merge efectiv la cele doua fisiere, iar functia remove\_original\_source\_files este cea care sterge cele doua fisiere originale.

Fisierul main

Am initializat variabila RECORDING\_LENGTH = 120. Am creat o instanta a clasei Logger pentru a loga mesajele importante.

Se specifica path-ul driver-ului pentru chrome si se initializeaza.

Pentru a handle-ui mai bine erorile ce ar putea aparea am utilizat blocuri de tip try catch.

Prima veriricare facuta este daca exista conexiune la internet, daca nu exista, o exceptie de tipul WebDriverException va fii aruncata iar programul se va inchide. Dupa aceasta, se cauta butonul de Accept All pentru cookies si se accepta daca se gaseste sau se arunca o exceptie daca a expirat timpul alocat pentru cautare. Am utilizat un bloc de try catch in blocul cel mare de trycatch deoarece pop-up-ul de cookies poate sa nu apara mereu iar codul ce urmeaza trebuie executat.

Pentru a vedea ca pagina de youtube s-a incarcat bine cautam bara de sus ce contine logo-ul, bara de search, butonul de profil si alte inca cateva butoane.

Apoi se cauta bara de search, se introduce textul si se da search. Dam click pe videoclipul ce contine textul dorit.

Imediat dupa accesarea videoclipul cream instante ale claselor de record, cream thread-urile si le pornim. Vom da join thread-urilor pentru a astepta terminarea executiei amandurora inainte de continuarea urmatorului cod.

Ca sa gestionam reclamele am cautat daca apare butonul de skip si l-am apasat daca da, daca nu am scris un log precum ca nu exista un buton de skip.

Dupa expirarea celor doua minute se va inchide fereastra de youtube. Se vor afisa decibelii si se va da merge la inregistrari. Dupa ce merge-ul e gata si am logat toate mesajele necesare se va inchide fisierul iar programul v-a lua sfarsit.

Probleme

1. Libraria pyautogui duce un anumit numar de fps-uri, iar daca eu nu ii dadeam cate fps-uri avea nevoie, videoclipul se derula la o viteza mai mare. De exemplu daca eu ii dadeam doar 15 fps-uri, el asteptand 20, imi completa prima secunda luand 5 fps-uri din secunda urmatoare si tot asa, facand videoclipul sa ruleze mai repede. Am rezolvat problema masurand cate fps-uri pe secunda duce libraria pyautogui.
2. Atunci cand dadeam start recording la cele doua inregistrari programul nu le facea concomitent, astfel audio-ul se intrerupea cand se realiza screenshot-ul, problema s-a rezolvat utilizand thread-uri.