Automatic YouTube Navigator with Selenium

*Caragea Matei - Ioan*

Limbaje utilizate: Python si JavaScript

Sistem de operare si hardware: MacOs Sonoma 14.6.1 cu procesor M1 arm64

Librarii utilizate: librariile folosite au fost incarcate in fisierul ***requirements.txt***

**main.py:**

**- apeleaza functiile si thread-urile create in fisierele separate -**

- n-am intampinat nicio eroare majora, doar apelez funcțiile create separat si intiez căutarea si selectarea videoclipului (voi muta acest proces într-un fișier separat)

-main-ul ramane strict pentru initializarea valorilor si apelare functiilor; la finalul programului, driverul si scriptul se inchid automat

**logging\_config.py**

**- configureaza structura log-urilor si creeaza diferite tipuri (message, error, warning, debugg)**

- nu am intampinat probleme

**youtube\_interactions.py:**

**- definesc in acest fisier functiile de intializare a driverului chrome, deschidere a paginii de YouTube, cautare prin search a unui keyword setat si deschidere si rulare a videoclipului ales –**

- am intampinat probleme la selectarea videoclipului si am rezolvat prin adaugarea unui time.sleep(); videoclip-ul nu avea timp pentru a deveni accesibil de click

- am adaugat prin JavaScript o animatie de scroll mai smooth pana la videoclip

**driver\_monitoring.py**

**- verifica in permanenta starea driverului pentru a evita situatiile in care browser-ul se inchide si programul ruleaza in gol –**

- nu am intampinat probleme

**internet\_check.p**y:

**- monitorizeaza in permanenta conexiunea la internet, intr-un thread ce ruleaza in background si opreste programul in caz contrar –**

- programul trimite un ping o data la 5 secunde catre site-ul google pentru a verifica conexiunea

- nu am avut probleme majore in crearea si implementarea acesteia, poate doar ca se intersecta cu functiile de record, de aceea am trecut-o intr-un thread separat

**audio\_recording.py:**

**- inregistreaza output-ul Desktopului , se opreste dupa timpul setat (30s) si scrie rezultatul intr-un fisier .wav -**

- intai am facut inregistrare pentru inputul calculatorului in loc de output

- am intampinat probleme in captarea sunetului speakerelor ( spekerele corespundeau device-ului cu indexul 1, insa acesta tot aparea ca fiind nedetectabil)

- desi am acordat privilegiile necesare din setarile de privacy ale laptopului, am aflat ca este o problema legata de securitate a sist. de operare

- pentru a rezolva problema am folosit un program numit [BlackHole](https://existential.audio/blackhole/) ce actioneaza ca o punte intre aplicatie si audio; se comporta ca un “speaker virtual” si capteaza sunetul redat de speakere, pe urma eu redand audio-ul acestuia

**analyse\_audio.py:**

**- extrage date din fisierul audio si calculeaza nivelul mediu al sunetului, pe care-l scrie intr-un fisier separat –**

- programul foloseste ca referinta valoarea 0 ( sunet maxim posibil ) si – inf (sunet 0)

**screen\_recording.py**

**- inregistreaza ecranul Desktopului din momentul rularii scriptului, se opreste dupa timpul setat (30s) si salveaza rezultatele intr-un fisier .mp4 -**

- am urmarit tutorial primit in pdf, insa am in continuare probleme cu înregistrarea ecranului, video-ul salvat este corupt si apare ca având 0.5 bytes, iar debug-ul din ffpmeg sugerează ca are 0 secunde; videoclip-ul era salvat in format .avi cu codec-ul XVID

- am încercat schimbarea codecului in h.264/mp4v si salvarea fișierului in mp4, insa tot primesc aceeași eroare

- voi incerca rularea codului pe un calculator cu intel; in prezent, proiectul e făcut pe macOS cu arm64

- am rezolvat fara a mai folosi masina virtuala; am folosit codec-ul mp4v si salvez fisierul in .mp4; problema a fost la fps-ul pe care-l impuneam (30 fps); fiind prea mare, acesta crea un efect de speed-up al videoclipului

- am creat o functie ce calculeaza fps-ul optimal al sistemului meu si-l paseaza functiei de screen recording

**merge\_video\_audio.py:**

**- realizeaza merge intre fiserele video si audio si creeaza videoclipul final al inregistrarii de ecran-**

- nu funcționează datorită esuarii funcției de screen recording

- dupa rezolvarea fisierului video, merge-ul a mers, insa culorile erau distrorsionate

- am rezolvat acest lucru prin converirea din bgra in bgr, eliminand canalul alpha

**accept\_terms.py**

**- accepta automat pop-up-ul de termeni si conditii -**

- am intampinat probleme in găsirea butonului de Accept all/ Reject all

- am dat inspect pe pop up-ul de termeni si condiții si am găsit id-ul corect al butonului; primeam mesajul ca a fost apăsat butonul cu succes, insa programul imi da crash imediat după

- am adăugat un time.sleep(5) după apelarea funcției si a funcționat, problema a fost ca nu îmi găsea search bar-ul după acceptare intr-un timp asa scurt si se închidea programul

**skip\_premium.py**

**- da skip automat la pop-up-ul de YouTube premium -**

-nu am intampinat probleme

**youtube\_filters.py**

**- aplica filtre pentru lista de videoclipuri afisate –**

- am intampinat probleme in interactiunea cu butonul de filtre

- am rezolvat problema printr-un inspect mai detaliat al paginii

**set\_playback\_speed.py**

**- seteaza automat viteza de redare la valoarea scrisa in program –**

- am intampinat probleme in accesarea butonului de playback

- am rezolvat folosind JavaScript

**collect\_video\_data.py**

**- extrage date ale videoclipului intr-un fisier separat precum: titlu, autor, url, nr. de abonati, nr. de vizualizari, data publicarii si hashtag-uri**

- am intampinat probleme gasirea Xpath-ului potrivit pentru aceste date

- am rezolvat dupa mai mlte inspect-uri in pagina videoclipului