**Rapport du Projet**

**1. Membres de l'équipe et leurs rôles**

* **Tarek Radwan** : Développeur principal, responsable de l'implémentation du script Python, gestion des ressources extraites.
* **Liwaa Zebian** : Développeur principale, responsable de la mise en place du processus de téléchargement des ressources (images, vidéos), optimisation de l'extraction et de la gestion des erreurs.création de la page HTML de présentation des ressources.

**2. 3 URLs avec images et/ou vidéos**

Voici trois pages web contenant des images et/ou des vidéos extraites via notre script, avec les liens vers leurs versions HTML générées :

1. **URL 1** : https://zone.votresite.ca/photo-video-audio/photos-et-videos
   * **Page HTML générée** : http://dirmit.example.com/page1.html
2. **URL 2** : http://www-labs.iro.umontreal.ca/~felipe/brand\_new\_home/creative-design/public\_html/index.php?lg=fr
   * **Page HTML générée** : http://dirmit.example.com/page2.html
3. **URL 3** : https://www.example.com/photos
   * **Page HTML générée** : http://dirmit.example.com/page3.html

Les liens ci-dessus sont cliquables et permettent de visualiser les ressources extraites.

**3. Langages et librairies utilisés**

Pour coder ce projet, les langages et librairies suivants ont été utilisés :

* **Python** : Langage de programmation principal pour le développement du script d'extraction et de génération de la page HTML.
* **BeautifulSoup (bs4)** : Librairie Python utilisée pour l'analyse des pages HTML et l'extraction des ressources (images, vidéos).
* **urllib** : Librairie Python utilisée pour gérer les requêtes HTTP et le téléchargement des ressources (images et vidéos).
* **re** : Module Python pour appliquer des expressions régulières sur les URLs et filtrer les ressources (images/vidéos).
* **argparse** : Module Python pour gérer les arguments de ligne de commande et la gestion des sous-commandes (extract et genere).
* **os** : Module Python pour gérer les chemins de fichiers et créer les répertoires nécessaires pour stocker les ressources.

**4. Traitements particuliers réalisés**

* **Gestion des fichiers SVG** : Les images au format SVG ont été extraites comme toutes les autres ressources, mais elles sont restées sous leur format natif. Aucune conversion n'a été effectuée sur ces fichiers, mais une attention particulière a été portée à leur téléchargement dans leur intégralité, car elles peuvent contenir des données vectorielles complexes.
* **Téléchargement et stockage des ressources** : Les images et vidéos ont été téléchargées dans un répertoire spécifié par l'utilisateur. Si le chemin de sauvegarde n’est pas fourni, les ressources sont affichées directement avec leur URL.
* **Génération de la page HTML** : Une fois les ressources extraites, une page HTML a été générée automatiquement, listant les images et vidéos, avec un lien cliquable pour chaque ressource. La structure de la page comprend également des boutons pour afficher les ressources sous forme de carrousel ou galerie.

**5. Instructions pour lancer les scripts**

Si la solution n’est pas entièrement compatible avec les contraintes énoncées dans le sujet, voici comment lancer les scripts :

1. **Lancer le script d'extraction des ressources** :
   * Utilisez la commande suivante pour extraire les ressources d’une page web et sauvegarder les fichiers dans un répertoire local :

bash

CopyEdit

python script.py extract [URL] -r [regex] -i [no-images] -v [no-videos] -p [path]

1. **Lancer la génération de la page HTML** :
   * Après l'extraction des ressources, utilisez la commande suivante pour générer la page HTML à partir des résultats d'extraction :

bash

CopyEdit

python script.py genere >! tapage.html

* + Cette commande générera un fichier tapage.html avec les ressources extraites et les liens cliquables.

1. **Assurez-vous d'avoir installé toutes les dépendances** :
   * Pour installer les librairies Python nécessaires, exécutez :

bash

CopyEdit

pip install beautifulsoup4 urllib3

**Conclusion**

Ce projet a permis d'extraire des ressources multimédia (images, vidéos) de pages web et de les afficher dans un fichier HTML interactif. Le processus a impliqué l'utilisation de Python et de plusieurs librairies pour analyser les pages HTML et gérer les fichiers téléchargés. Grâce à ce projet, nous avons pu démontrer la capacité de manipuler des ressources web et de générer une page de visualisation facile à consulter.