Année scolaire 2024-2025
dans le cadre du projet SAE du Semestre 5

# Dossier de test

## Création du site de recherche et de recommandation de séries

Ce rapport de test a été réalisé par Cédric Longuet et par Valentane Vacquié, non-alternant.

iut TOULOUSE
INSTITUT UNIVERSITAIRE DE TECHNOLOGIE

# Sommaire

# Introduction

Ce dossier regroupe une série de tests destinés à valider les fonctionnalités clés de l'application Python/Flask, qui analyse et recommande des séries à partir de sous-titres. Chaque test suit une méthodologie en trois étapes : préparation, exécution et validation des résultats. L'objectif est d'assurer le bon fonctionnement de l'application, qu'il s'agisse de la gestion des fichiers bruts, des traitements algorithmiques ou des interactions via les API et l'interface utilisateur.
Pour chaque fonction ou route, les tests configurent l'environnement, exécutent les appels nécessaires avec les paramètres appropriés, et valident que les résultats répondent aux attentes. Ce dossier garantit la robustesse et la fiabilité de l'application, assurant ainsi une expérience utilisateur fluide et des performances optimales.

# I - Dossier python_nettoyage

## A) Test de la Fonction main()

1. Préparation
   - Créer un dossier "sous- titres “ structuré comme ci après :
     sous_titres »
     » B@ňeṡ .zip » episode 1.srt (00:16 ^ Je sui6 mËd◦cin légiste.)
     » episode2.srt (Cet os est cassé.)
     » episode1.srt (I'm medical examiner.)
     » episode2.srt (This bone is broken.)

2. Exécution : Appel de la fonction main().
3. Test validé si
   Le test est validé si après l'appel de la fonction main() mon dossier “sous-titres” est comme ci-dessous :
   sous_titres » bones » vf.txt (Je suis médecin légiste. Cet os est cassé.)
   » vo.txt (I'm medical examiner. This bone is broken.)

# II - Dossier flask - Sous Dossier api - fichier api.py

## A) Test de la Fonction get_info_serie(title)

1. Exécution : Appel de la fonction get_info_series(bones).
2. Test validé si
   Le test est validé si le résultat du "print” est :
   Title: Bones, Year: 2005-2017, Rated: TV-14, Released: 13 Sep 2005, …,
   total Seasons : 12, Response: True

## B) Test de la Fonction get_top5_json(mots)

1. Préparation
   - Mettre le dossier sous_titres avec les 125 sous dossiers de série nettoyer dans la variable st_dir.
   - Créer la liste "mots" : mots = ("os", "légiste", "victime", "docteur", "agent")

2. Exécution : Appel de la fonction get_top5_json(mots).
3. Test validé si
   Le test est validé si quand je "print" l'appelle de cette fonction j'obtiens le résultat ci dessous :

   [ { ' Title': 'Doctor Who', …}, { 'Title': 'Bornes' , …},{ 'Title': 'Eleventh Hour'...} , { 'Title': 'Criminal Minds',...},{ 'Title': 'Travelers',...} ]

## C) Fonction get_all_serie_json_by5(st_dir, page)

1. Préparation
   - Mettre le dossier sous_titres avec les 125 sous dossiers de série nettoyer dans la variable st_dir.
   - Créer la variable "page" : page =1.

2. Exécution : Appel de la fonction get_all_serie_json_by5(stdir, page).
3. Test validé si
   Le test est validé si le "print" de la fonction correspond à cela :

   [ { ' Title': '24', …}, { 'Title': '90210' , …},{ 'Title': 'Alias'...} , { 'Title': 'Angel',...},{ 'Title': 'Battlestar Galactica',...} ]

# III - Dossier flask - Sous Dossier api - fichier tf_idf.py

## A) Fonction main (st_dir , mot_input)

1. Préparation
   - Mettre le dossier sous_titres avec les 125 sous dossiers de série nettoyer dans la variable st_dir.
   - Définissez une liste de mots-clés pour la recherche, par exemple :
     mot_input = ["os", "docteur", "victime"]
   - Supprimer les fichiers pickle du projet si il existe déjà.

2. Exécution Test 1 : Appel de la fonction main(st_dir, mot_input).
3. Test 1 validé si
   Le test est validé si deux fichiers pikle (.pkl) se sont créés dans le dossier du projet et si le résultat du "print" de la fonction correspond à ceci :
   Chargement des sous-titres …
   Chargement des documents …
   Fichier non trouvé.
   Calcul de la matrice tfidf et du vectorizer …
   Enregistrement de la matrice tfidf et du vectorizer …
   Recherche de la série la plus similaire …
   Résultat : Doctor Who, Bones, Criminals Minds , Whitechapel, Eleventh Hour

4. Exécution Test 2 : Refaire Appel à la fonction main(st_dir, mot_input) car maintenant les fichiers pickle existent déjà et la fonction va les utiliser et ne pas les refaire.
5. Test 2 validé si
   Le test est validé si le résultat du "print" de la fonction correspond à ceci :
   Chargement des sous-titres …
   Chargement des documents …
   Fichier trouvé.
   Recherche de la série la plus similaire …
   Résultat : Doctor Who, Bones, Criminals Minds , Whitechapel, Eleventh Hour

### B) Fonction get_recommandation(st_dir)

1. Préparation
   - Mettre le dossier sous_titres avec les 125 sous dossiers de série nettoyer dans la variable st_dir.
   - Ajouter des séries aimées dans la session (session['liked']) : “Bones” et “NCIS”.
   - Ajouter des séries non aimées dans la session (session['disliked']) : “Criminal Minds”

2. Exécution : Appel de la fonction get_recommandation(st_dir).
3. Test validé si
   - Les séries aimées sont bien prises en compte pour le calcul des similarités.
   - Les séries aimées ne figurent pas dans les recommandations.
   - Les recommandations retournées sont limitées à un maximum de 5 séries.
   - La liste des recommandations obtenus est celle ci : “Criminal Minds” (Emoticone “Je n’aime pas” en rouge) , “Community”, “Raines”, “The Shield”, “Gary Unmarried”.

## IV - Dossier Flask - Sous Dossier static - fichier manage.py

### A) Route pour la page d'accueil

1. Préparation
   - S'assurer que le fichier index.html existe dans le dossier templates.

2. Exécution : Appeler la route / via une requête GET.
3. Test validé si
   - La réponse a un code HTTP 200.
   - Le contenu retourné contient le HTML correspondant à index.html.

### B) Route /recommandations

1. Préparation
   - Ajouter des séries aimées dans la session (session['liked']).
   - Ajouter des séries non aimées dans la session (session['disliked']).
   - S'assurer que le fichier recommandation.html existe dans le dossier templates.

2. Exécution : Appeler la route /recommandations via une requête GET.
3. Test validé si
   - La réponse a un code HTTP 200.
   - Les séries retournées dans le template incluent les recommandations avec leurs détails.
   - Les séries aimées et non aimées sont correctement affichées.

## C) Route /all_series

1. Préparation
   - Ajouter des séries aimées dans la session (session['liked']).
   - Ajouter des séries non aimées dans la session (session['disliked']).
   - S'assurer que le fichier allSeries.html existe dans le dossier templates.

2. Exécution : Appeler la route /all_series via une requête GET.
3. Test validé si
   - La réponse a un code HTTP 200.
   - Le contenu de la page inclut les 125 séries connues ainsi que leurs données de session.
   - Les cinq premières séries ont pour titres “24”, “90210”, “Alias”, “Angel”, “Battlestar Galactica”.

## 4. Route /search

1. Préparation
   - Ajouter la série “Bones” dans la session (session['liked']).
   - Ajouter la série “Doctor Who” dans la session (session['disliked']).

2. Exécution : Appeler la route /search?motSerie=os docteur victime.
3. Test validé si
   - La réponse a un code HTTP 200.
   - Ce qui suit est visible sur la page web : Affiches, informations des cinqs séries ci dessous avec la série “Bones” déjà aimée (bouton vert) et la série “Doctor Who” non apprécié (bouton rouge) : Doctor Who, Bones, Criminals Minds , Whitechapel, Eleventh Hour

## 5. Route /getLikes

1. Préparation
   - Ajouter la série “Bones” dans la session (session['liked']) et vérifier qu’elle soit la seule.

2. Exécution : Appeler la route /getLikes via une requête GET.
3. Test validé si
   - La réponse a un code HTTP 200.
   - Le JSON retourné contient la ou les séries likées : sur le site, la série “Bones” à le bouton “J’aime” activé (vert).

## 6. Route /getDislikes

1. Préparation
   - Ajouter la série “Doctor Who” dans la session (session['disliked']) et vérifier qu’elle soit la seule.

2. Exécution : Appeler la route /getDislikes via une requête GET.
3. Test validé si
   - La réponse a un code HTTP 200.
   - Le JSON retourné contient la ou les séries non aimées : sur le site, la série “Doctor Who” à le bouton “Je n’aime pas” activé (rouge).

## 7. Route /feedback

1. Préparation
   - Vérifier que la session (session['liked']) existe et qu’elle soit vide.
   - Ajouter uniquement la serie “Breaking Bad” dans la session (session['disliked']).

2. Exécution Test 1 : Appeler la route /feedback via une requête POST avec des données JSON { "id": "Breaking Bad", "action": "like" }.
3. Test 1 validé si
   - La réponse a un code HTTP 200.
   - L'action like ajoute correctement la série dans la liste liked et la supprime de disliked si elle s'y trouve.
   - Aucune série n'est ajoutée deux fois dans une liste.
   - La réponse est correctement formatée : { "success": true }.

4. Exécution Test 2 : Appeler la route /feedback via une requête POST avec des données JSON { "id": "Breaking Bad", "action": "dislike" }.
5. Test 2 validé si
   - La réponse a un code HTTP 200.
   - L'action dislike ajoute correctement la série dans la liste disliked et la supprime de liked si elle s'y trouve.
   - Aucune série n'est ajoutée deux fois dans une liste.
   - La réponse est correctement formatée : { "success": true }.