Spécification des besoins :

* Moteur de recherche (classique)
* Ajouter des filtres de recherches (en fonction de la date des documents, par type de contenu, etc.)
* Ajouter des tris sur les résultats
* Prise en charge de l’anglais et du français
* Pagination : Diviser les résultats de recherche par pages ???
* Permettre à l’utilisateur d’ajouter des textes ???

# Problématique du sujet et spécifications :

(Description de ce que doit faire le programme, en langage naturel)

Dans le cadre de l’enseignement « Programmation de spécialité : Python », il nous a été demandé de réaliser un moteur de recherche, c’est-à-dire un outil informatique conçu pour trouver des informations stockées dans une base de données en réponse à une requête formulée par un utilisateur.

Le moteur de recherche que nous avons choisi de concevoir correspond à un moteur de type spécialisé, qui est un moteur spécialisé dans un seul domaine d’intérêt spécifique, en l’occurrence dans les domaines…

OU

Le moteur de recherche que nous avons choisi de concevoir correspond à un moteur de type général, qui est un moteur conçu pour répondre à différents types de requêtes et de sujets.

Ce moteur de recherche doit permettre à ses utilisateurs d’accéder à différents textes, tous enregistrés localement, à partir de requêtes contenant les mots-clés que l’utilisateur recherche dans les différents textes. L’accès à ses textes doit ainsi se baser sur la pertinence des mots présents dans les textes en fonctions des mots-clés de l’utilisateur : par exemple, si un texte présente plusieurs fois le mot clé recherché par l’utilisateur, et que ce même mot n’est pas ou est peu présent dans les autres textes présents dans le corpus de texte du moteur de recherche, alors celui-ci sera considéré comme pertinent et sera affiché à l’utilisateur.

Le moteur de recherche contient uniquement des documents reddit et arxiv, et compte près de … articles.

Tous les articles ont comme particularité de contenir plus de … mots.

Au-delà de sa fonctionnalité de base qui consiste à rechercher des articles en fonction de mots-clés, le moteur de recherche doit permettre au client de de filtrer ses résultats de recherche, notamment en lui permettant de choisir une date de publication précise pour les articles auxquels il souhaite accéder, ou bien uniquement les articles de certains auteurs lorsque ceux-ci sont précisés.

Le moteur doit également permettre à l’utilisateur de trier les résultats de sa requête en fonction de la date de publication des articles, ou bien tout simplement par ordre alphabétiques des titres des articles.

Enfin, il doit également permettre à l’utilisateur d’accéder à des articles anglophones en plus d’articles francophones si l’utilisateur le désire.

# Analyse effectuée :

*(= précise comment les spécifications vont être réalisées, cad quel est l'environnement de travail ? Quelles sont les données identifiées dans la spécification ? Un diagramme de classe (avec présentation des noms de classe et méthodes, relations d'héritages, et relation d'utilisations entre les classes (appel ?)))*

# Commentaire sur l'implantation :

*(Dont répartition des tâches)*

*PAS DE LIGNES DE CODE DANS LE RAPPORT*

Pour réaliser ce travail, nous avons, décidé en premier lieu de diviser l’implémentation selon les trois versions de ‘implémentations demandées dans les consignes, à savoir la constitution d’une base pour l’application, la création du moteur de recherche, et la réalisation de l’interface de l’application. Pour la première étape, à savoir la constituons d’une base pour l’application, Tiavina s’est occupé de charger les textes du corpus, les sauvegarder localement sur un fichier csv, ainsi que de créer les principales classes de l’applications, à savoir les classes « Document », « Auteurs », « Corpus », « Reddit » et « ArxivDocument », et quelques-unes de leurs méthodes, tels que leurs constructeurs, d’après ce que nous avons réalisé lors des TD 3 à 5, tandis que Quentin s’est chargé de rédiger la problématique du sujet et la spécification des besoins, de réaliser le diagramme de classe, et d’écrire la première partie du commentaire sur l’implémentation.

# Présentation des tests :

Les test unitaires ont été réalisés avec la librairies « unittest ». A l’aide de ChatGPT nous avons pu établir les test principaux à faire sur chaque fonction des classes. Ainsi, nous avons réalisé les tests unitaires sur les classes Corpus, Author, et Document.

Dans la classe corpus, le principal enjeu est de vérifier si les documents s’ajoutent convenablement dans le corpus et qu’ils restent dans le même état que lors de l’interrogation de l’api. La sauvegarde doit aussi créer le fichier avec le bon chemin d’accès. Il doit aussi pouvoir confirmer que les présent dans le corpus sont ajouté uniquement si ils ne sont pas déjà présent.

Concernant les test sur la classe Author, nous vérifions l’affichage des informations des auteurs ainsi que le traitement lors de l’ajout de ses productions.

Enfin, les tests dur les documents se concentrent sur l’affichage des informations des documents ce qui montre que l’interrogation de l’api à bien fonctionné.

Il est à noter que les fichiers tests doivent être dans le même répertoire que les fichiers contenant les classes. En effet, lors des essais, les classes étaient inatteignables si les fichiers tests étaient dans un dossier spécifique.

# Conclusions et perspectives :