Dimitra Savits
Rose Marly
Laure Demierre
Olivier Oppliger

Projet de développement logiciel
Davide Picca
24 Mai 2023

Projet Celsus

Introduction :

Dans le cadre du cours projet de développement de logiciel 2023 à l'Université de Lausanne, sous les directives de l'enseignant Davide Picca, nous avons dû réaliser un projet en lien avec une intelligence artificielle. En effet, cette dernière ayant un grand ressort en cette année, il était évident que des projets suivants cette thématique allaient survenir. Ce projet, venant de l'enseignant lui-même, est de réaliser un logiciel puisant dans les ressources de l'intelligence artificielle, passant par *LangChain*, en implémentant le modèle de chatbot d'*OpenAI* en son sein.

En soi, *chatGPT* peut par lui-même déjà effectuer un type d'action similaire (et techniquement parlant, par son évolution constante en parallèle à notre projet, peut maintenant effectuer lesdites actions de notre projet en son état actuel). Cependant, il ne peut importer de texte provenant directement de fichiers internes provenant de sa propre machine (et possédant une limite de caractères, copier l'intégralité de, par exemple, un livre ne serait définitivement pas une possibilité). Il existe aussi des applications de traitement textuel, donnant des analyses approfondies, mais nous cherchons avec notre projet un moyen simple d'obtenir des informations rapides sans détails encombrants.

But :

Une fois la liaison établie entre un module de lecture de fichier *PDF* et les possibilités prodiguées par *LangChain*, nous sommes en mesure de créer un logiciel de traitement textuel basique, afin de faciliter l'analyse et la compréhension de textes.
Les principaux outils que nous voulions mettre à disposition des utilisateurs furent :
- Créer des résumés
- L'extraction de citations
- Indiquer le ou les thèmes principaux du texte

Méthodologie :

Au commencement du projet, nous avions dû réaliser 4 éléments qui nous ont aidés à mieux structurer notre travail. Je vais vous les lister ainsi que les accompagner chacun de plusieurs captures d'écran. Le premier fut un diagramme de Gantt qui permit de déterminer plus simplement les périodes de travail de chacun ainsi que l'ordre de priorité des tâches que nous devions effectuer.

Figure 1 : Diagramme de Gantt

Ensuite, le work breakdown structure nous a aussi aidé à bien décomposer les tâches pour en faciliter leur gestion et leur suivi.

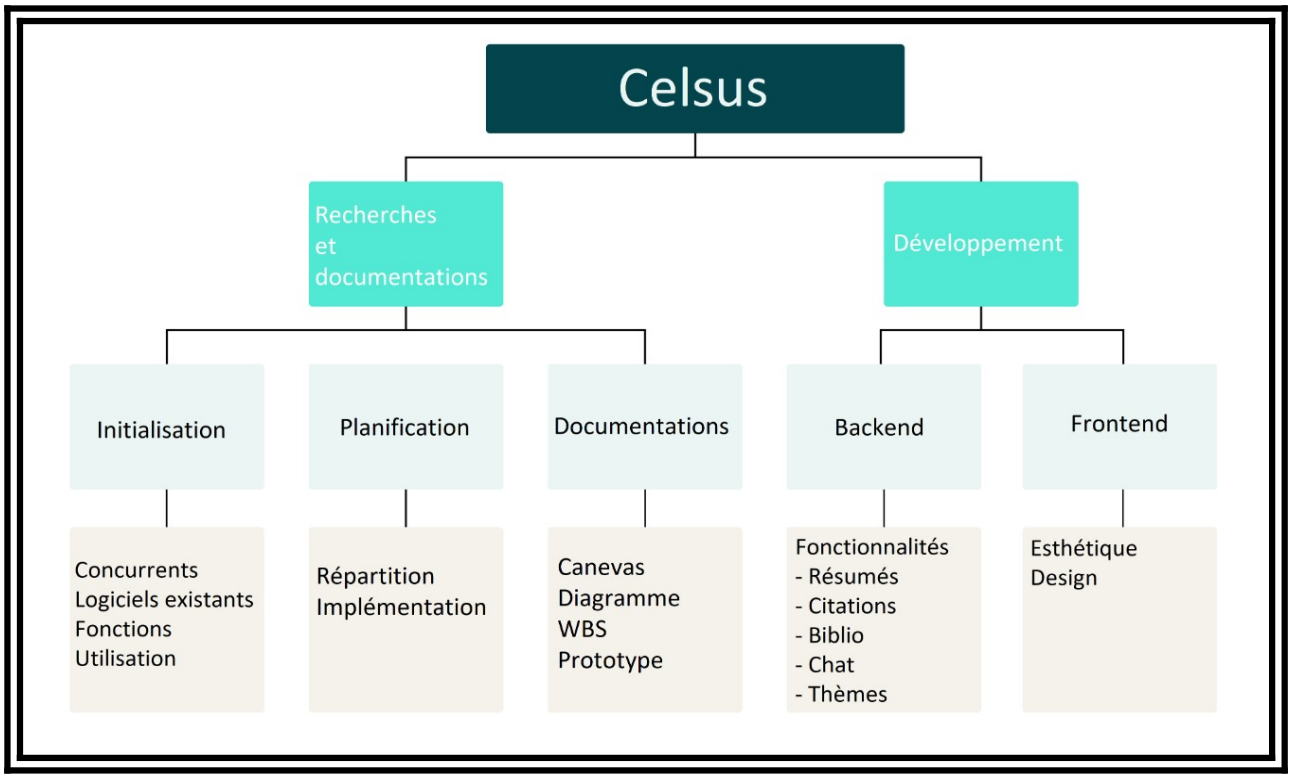

Figure 2 : Work Breakdown Structure

Troisièmement, un document qui dépeignait la gestion des risques avec les risques potentiels et les manières adéquates à utiliser pour les gérer.

Le projet doit faire face à de nombreux risques, pouvant surgir à tout moment. Le tableau ci-dessus changera d'apparence en fonction des problèmes survenus et traités par les différents responsables au fil de l'avancement du projet :

Risque	Responsable(s)	Procédure
Incompatibilité de système au sein du code	All	S'assurer de travailler sur le même espace virtuel. Signaler aux autres membres toute déviation
Incompatibilité de ressources externes	All	Lors d'une liaison, s'assurer de la compatibilité. Dans le cas d'une incompatibilité, vérifier s'il y a une procédure de changement de version, autrement chercher une autre ressource
Burnout	All	Tranquillisant dans la fesse gauche de la personne en détresse, puis rassurer la personne avec des peluches
Perte de code	All	S'assurer d'avoir des backups online et local
Vérification des ressources allouées	Sb	Une mauvaise ou ressource inutile pourrait être une nuisance. Chaque ressource proposée doit être vérifiée par au moins un sbire de la Team Rocket
Contact perdu avec un stakeholder	Ld	Initier la phase de diplomacie avec le stakeholder, afin de remotiver son intérêt avec preuve à l'appui de l'avancée du projet
Manque de temps	Sb	Vérifier constamment le calendrier des rendus, pour éviter les mauvaises surprises

Abréviations :

Ld : leadeuse

Sb : sbire

All : toute l'équipe

Figure 3 : Document sur la gestion des risques

Et le dernier élément, un prototype *Figma* (consultable ici[1]), trouva son utilité dans le fait que tous les membres aient une bonne idée de l'aspect visuel de notre application. Bien que beaucoup d'éléments aient changé par la suite (ex. : les options qui ne sont plus présentes).

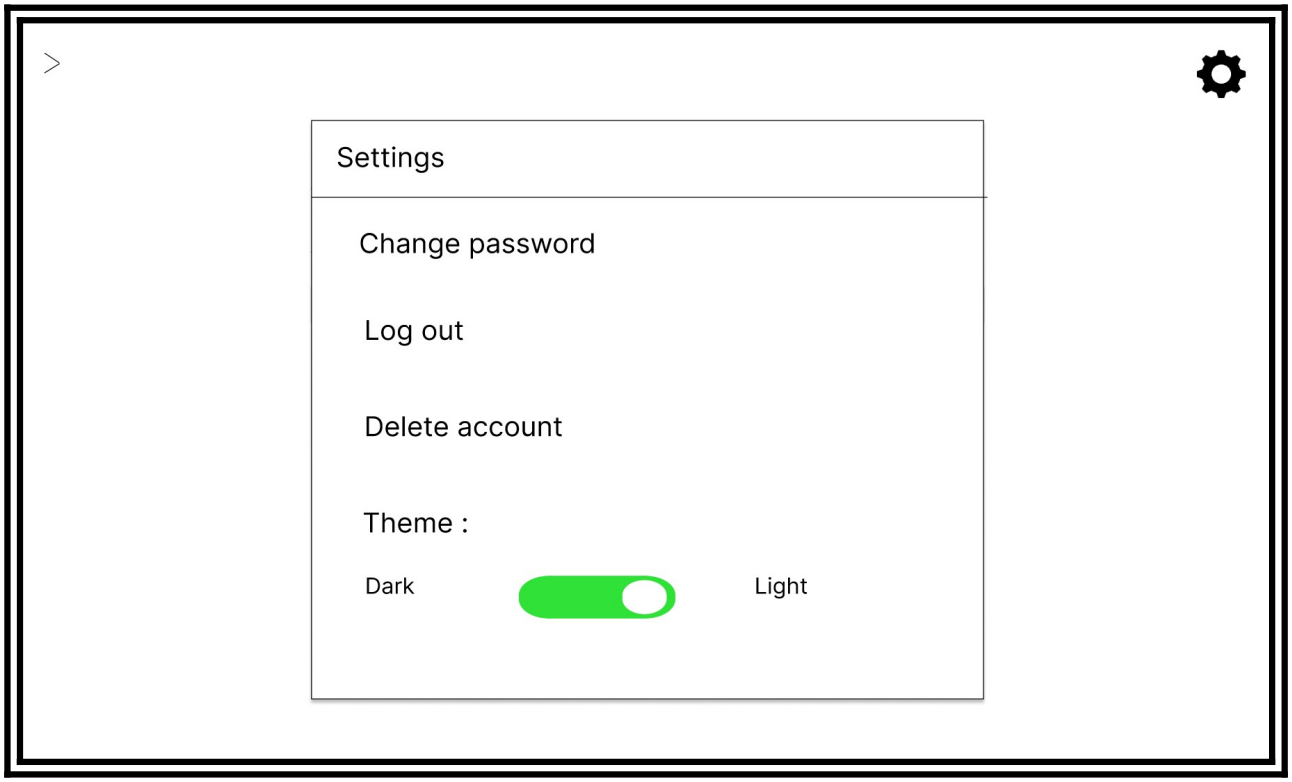

Figure 4 : Options de notre logiciel dans Figma

1 https://www.figma.com/proto/bISjrelBLAajSV1VlhlPNp/Celsus?type=design&node-id=26-104&scaling=min-zoom&page-id=0%3A1&starting-point-node-id=44%3A24 accédé le 17.05.2023 à 17h00

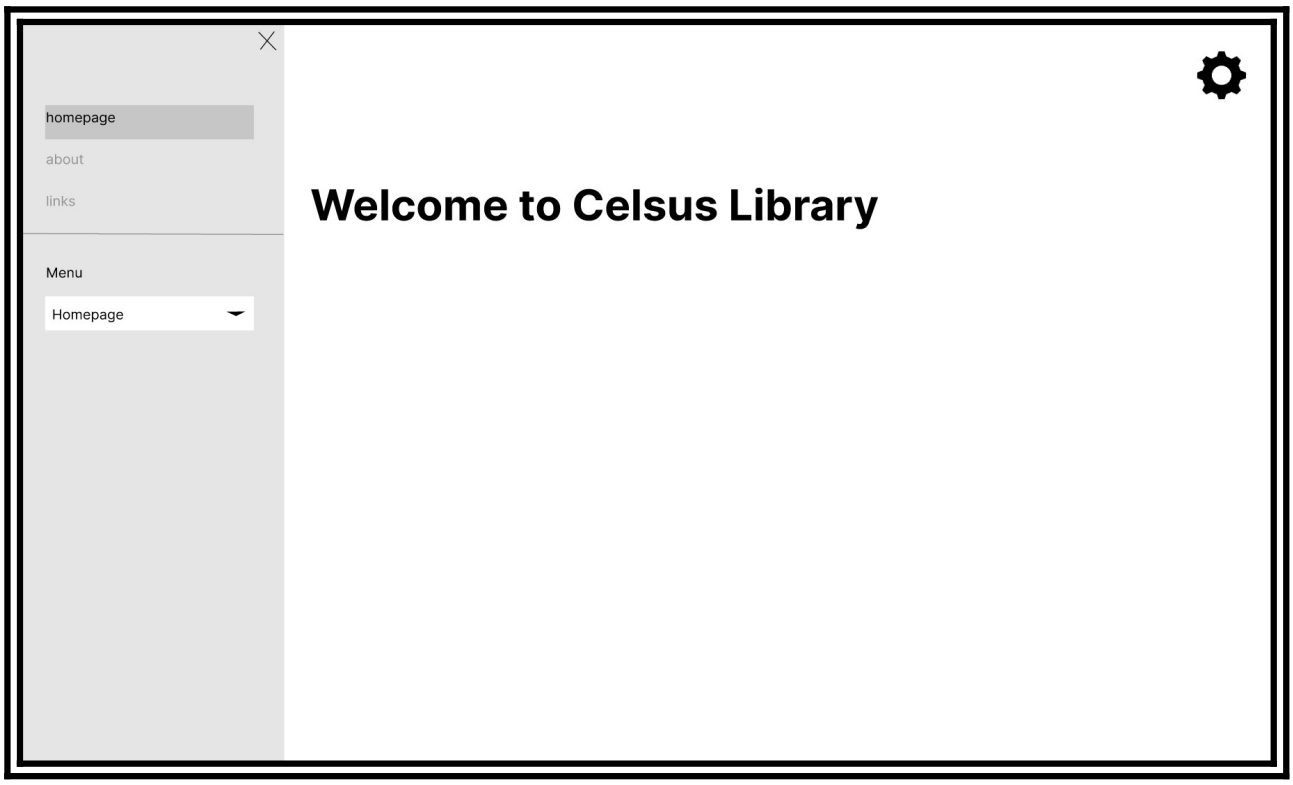

Figure 5 : Homepage de notre logiciel dans Figma

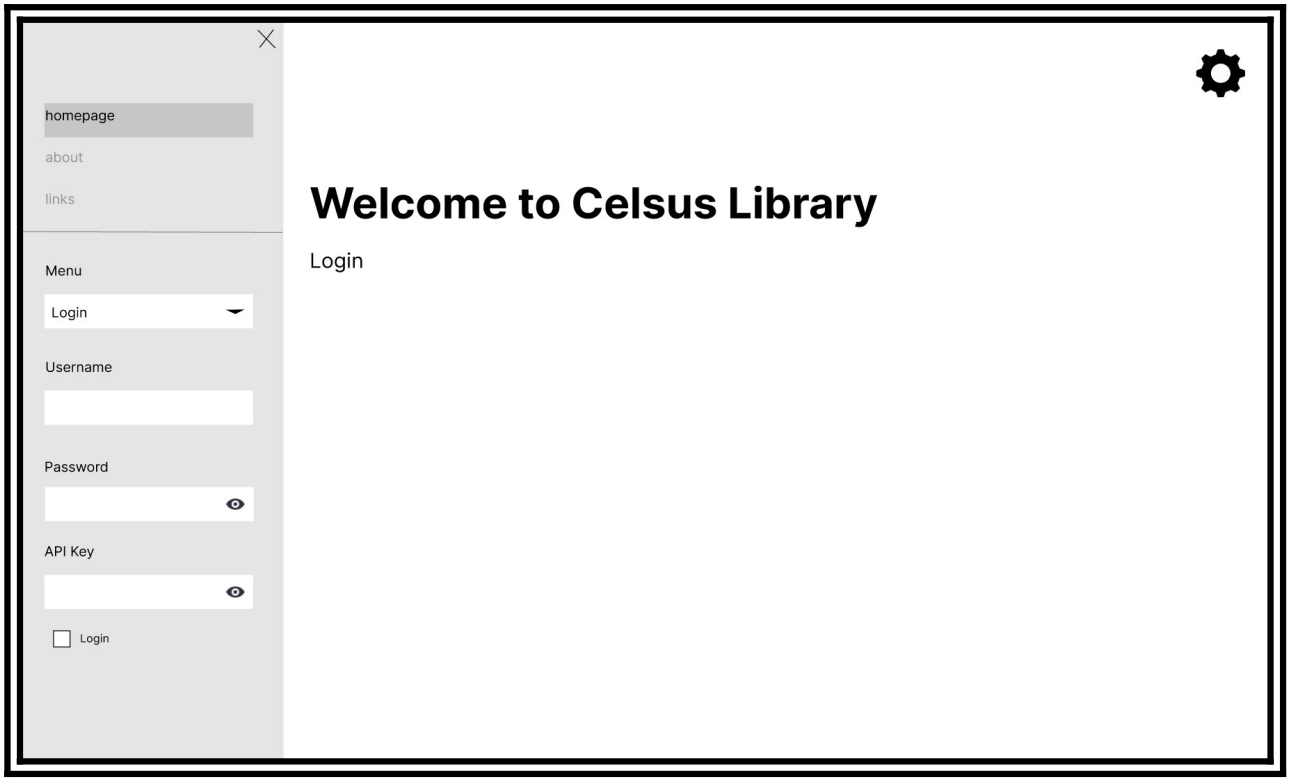

Figure 6 : Fonction Login de notre logiciel dans Figma

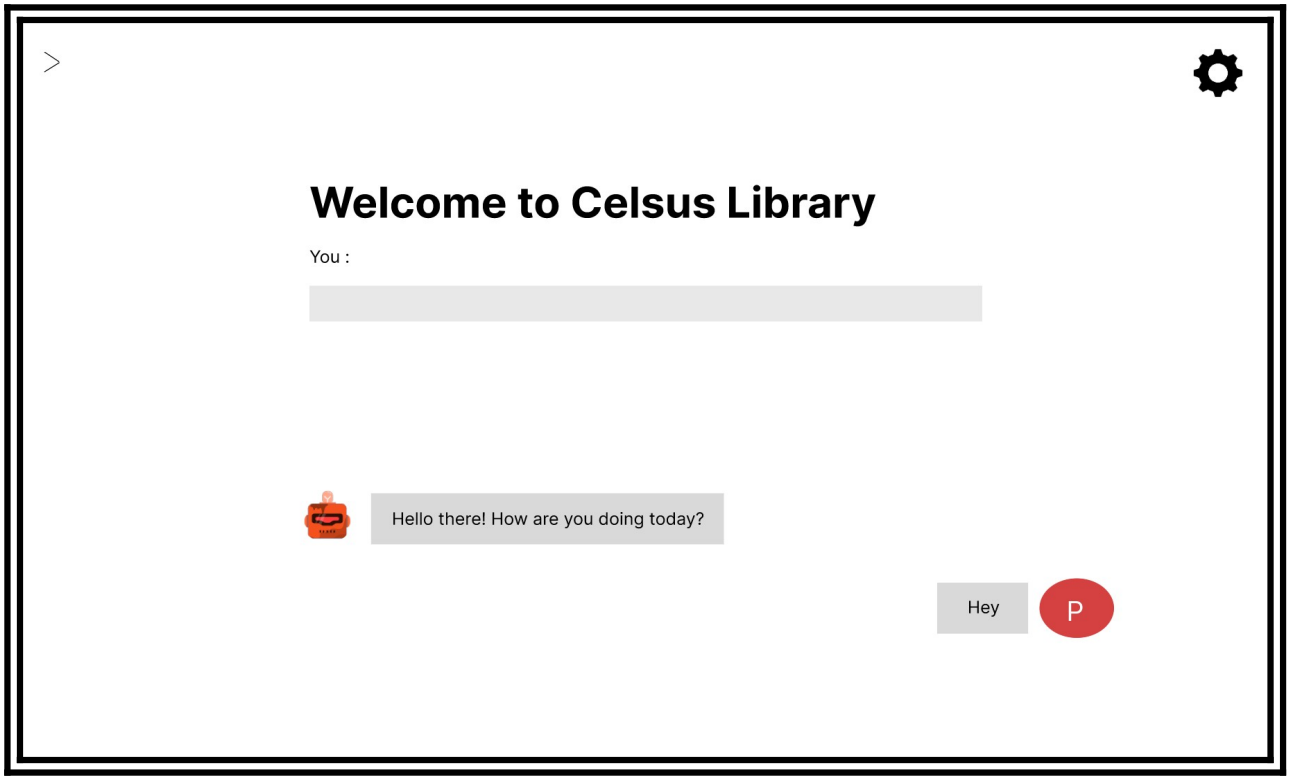

Figure 7 : Conversation avec l'AI de notre logiciel dans Figma

Par la suite, il nous a fallu déterminer le mode de travail, à savoir l'environnement. Comme ce cours fait suite aux cours de programmation orientée objet *Python*, nous sommes partis sur ce langage informatique. Ensuite, nous avons décidé de quel framework nous allions utiliser. Après l'élimination des modèles de *React* et *Pynecone*, nous nous sommes dirigés vers le modèle de *Streamlit*, le premier étant surtout orienté *JavaScript*, et le second possédant une certaine lacune dans sa documentation par rapport au modèle finalement utilisé.

Une fois ceci établi, nous avons commencé à nous documenter sur l'intérieur de notre logiciel. Un des premiers fut *LangChain*, permettant l'implémentation de mémoire, le traitement textuel et l'accès à un chatbot.
Cependant, *LangChain* fait appel à de nombreux modules divers et variés. Un des modules que nous avons cherché à implémenter a été le *projet Gutenberg*, permettant d'accéder à une grande quantité d'ouvrages. Mais nous avons délaissé par la suite cette bibliothèque en ligne, pour trois raisons majeures :
Premièrement, cela limitait les recherches aux documents présents sur la plateforme. Deuxièmement, l'implémentation de cette base de données au sein de notre code est beaucoup plus longue et fastidieuse par rapport à la qualité désirée, qui a pu finalement être augmentée en ne se limitant pas à l'espace textuel fourni par cette base de données en ligne.

C'est pour cette raison que nous avons dérivé sur l'utilisation de fichiers locaux que les utilisateurs posséderaient. Pour effectuer un traitement optimal, nous avons choisi que notre logiciel traite les fichiers sous format *PDF*. Nous avons trouvé le modèle de *PyPDF* afin de pouvoir rechercher et transformer les documents du type lié.

Par cette méthode, nous sommes arrivés au résultat actuel, à savoir un moteur de traitement textuel relié à un chatbot utilisant *LangChain*. Par ailleurs, nous avons enlevé la création de comptes sur

notre site, afin que les utilisateurs aient besoin de seulement leur clé *OpenAI*, au lieu d'avoir à créer plusieurs comptes pour avoir accès aux fonctionnalités proposées par notre logiciel. Cependant, pour que cela fonctionne, juste créer une simple connexion entre un fichier *PDF*, *chatGPT* et *streamlit* n'aurait pas été suffisant. Lié à *LangChain*, nous avons du faire appel à *Chroma db* avoir des documents-type permettant les réponses aux questions. Pour ce faire, nous avons utilisé la chaîne *RetrievalIQA* afin de relier lesdites réponses aux documents de la base de données, servant ainsi de passerelle entre l'utilisateur et la réponse à venir, recomposée par l'intelligence artificielle d'*OpenAI*. Avant cette chaîne, nous étions partis sur *VectorDB*, qui est devenue obsolète en comparaison de son rival plus récent et performant avec lequel nous avons échangé.

Afin d'avoir une méthode de travail optimal, le développement du logiciel a été divisé entre les membres de l'équipe de développement. Bien que la méthode principale était que «tout le monde travaille sur la même partie en même temps», le nombre grandissant d'actions à codifier poussa l'équipe à se répartir les tâches : pendant que deux personnes s'occupaient de spécificités du backend, deux autres s'occupaient du frontend ainsi que la documentation intra (présentation du logiciel et de l'équipe, règlement et liens vers les ressources externes primaires) et extra-logiciel (présentation, guide d'utilisation, règlement et commentaires).

Enfin, n'oublions pas de mentionner l'origine du nom du projet. Il s'agit d'une référence à la bibliothèque *Celsus*, se situant en Turquie. Elle date de l'époque romaine, et a été la fierté de la ville d'Éphèse durant le second siècle apr. J.-C. Il reste de nos jours que des ruines, mais celles-ci figurent comme ornement au passé glorieux de la région, fort détroit commercial en son temps.

Résultats :

Bien que nous ayons changé de voix plusieurs fois, changeant de méthode de traitement et d'outils à plusieurs reprises, impliquant le renouveau en entier du code, nous sommes arrivés aux résultats escomptés. Notre logiciel peut importer n'importe quel texte sous format *PDF*, puis de répondre à des questions concernant son contenu.
De la sorte, notre application sur *streamlit* peut résumer le texte, extraire les éléments clés, en ressortir les thématiques, rechercher des textes similaires et effectuer des citations, dont la réponse peut être sauvegardée dans un fichier externe sous format *.txt*, permettant ainsi aux utilisateurs de récupérer leurs réponses et de les copier, si nécessaire, dans leurs travaux de façon aisée.

Conclusion :

Le produit de notre travail nous a permis de développer nos capacités en matière de programmation, ainsi que notre autonomie, et bien sûr notre gestion des besoins. Le partage du travail était vital afin de pouvoir maximiser notre temps limité, et ainsi surmonter le défi.
Mais au-delà des buts personnels, nous avons marqué le territoire de l'informatique avec notre empreinte. Notre logiciel pourra être manipulé par des utilisateurs du monde entier cherchant à traiter des textes, et ce d'une façon relativement simple d'utilisation.

Références :

-https://github.com/mobarski/ask-my-pdf/blob/main/src/gui.py (accédé le 17.05.2023 à 13 :40)

-https://zero-shot-text-classifier.streamlit.app/ (accédé le 17.05.2023 à 13 :40)

-https://github.com/streamlit/example-app-zero-shot-text-classifier (accédé le 17.05.2023 à 13 :40)

-https://github.com/raduangelescu/gutenbergpy (accédé le 17.05.2023 à 13 :40)

-https://pypi.org/project/Gutenberg/ (accédé le 17.05.2023 à 13 :40)

-https://haystack.deepset.ai/ (accédé le 17.05.2023 à 13 :40)

-https://futurism.com/hack-deranged-alter-ego-chatgpt (accédé le 17.05.2023 à 13 :40)

-https://docs.streamlit.io/streamlit-community-cloud/get-started/deploy-an-app/connect-to-data-sources/secrets-management#how-to-use-secrets-management (accédé le 17.05.2023 à 13 :40)

-https://docs.streamlit.io/streamlit-community-cloud/get-started/deploy-an-app/connect-to-data-sources/secrets-management#how-to-use-secrets-management (accédé le 17.05.2023 à 13 :40)

-https://www.figma.com/team_invite/redeem/wSQL9ukpDuYhmSiOzvIbzL (accédé le 17.05.2023 à 13:40)

-https://code-with-me.global.jetbrains.com/zv5qfkRLHp4dWkyelgz_wQ#p=PY&fp=4EE54545B05F5425AF5F813313AB7711195C5465EB05408371FF0318D097224C (accédé le 17.05.2023 à 13 :40)

-https://components.streamlit.app/ (accédé le 17.05.2023 à 13 :40)

-https://pynecone.io/docs/library (accédé le 17.05.2023 à 13 :40)

-https://pynecone.io/docs/components/overview (accédé le 17.05.2023 à 13 :40)

-https://langchain.readthedocs.io/en/latest/modules/memory.html (accédé le 17.05.2023 à 13 :40)

-https://github.com/hwchase17/langchain (accédé le 17.05.2023 à 13 :40)

-https://fr.reactjs.org/ (accédé le 17.05.2023 à 13 :40)

-https://github.com/streamlit/example-app-zero-shot-text-classifier (accédé le 17.05.2023 à 13 :40)

-https://platform.openai.com/docs/api-reference/completions/create#completions/create-suffix (accédé le 17.05.2023 à 15:13)

-https://platform.openai.com/docs/tutorials/web-qa-embeddings (accédé le 17.05.2023 à 15:13)

-https://platform.openai.com/docs/models/content-filter (accédé le 17.05.2023 à 15:13)

-https://huggingface.co/spaces/sophiamyang/Panel_PDF_QA/blob/main/LangChain_QA_Panel_App.ipynb (accédé le 17.05.2023 à 15:13)