Cahier des Charges

Projet PIC : ChatEILCO

Fait par :
BAKOUCH Youssef
SALAH Ibrahim
TAHEREDDINE Taha

Tuteurs :
M. BAUTISTA Esteban
M. LEMAIRE Adrien
M. ROUSSEL Gilles

Année Universitaire 2025/2026

Table des matières

Table des figures

I Introduction

1 Contexte et Problématique

Au sein de l'EILCO, l'accès aux informations administratives, pédagogiques et techniques demeure souvent complexe pour les étudiants, les enseignants et le personnel. Les données sont dispersées entre plusieurs sources et documents numériques (PDF, Word, etc.), ce qui rend la recherche d'informations parfois fastidieuse et chronophage. Cette situation conduit à un grand nombre de demandes répétitives adressées aux services administratifs et aux enseignants (procédures, formulaires, documents à télécharger, etc.), générant une charge de travail supplémentaire. Le personnel de l'école, tout comme les étudiants, doit pouvoir accéder rapidement et efficacement aux informations dont il a besoin dans le cadre de ses activités quotidiennes.

- Problématique :

  Comment concevoir une solution capable de centraliser l'accès à l'information au sein de l'EILCO, tout en réduisant les sollicitations répétitives et en améliorant l'efficacité du personnel et l'autonomie des étudiants ?

2 Solution Proposée

Afin de répondre à cette problématique, le projet vise à développer un assistant conversationnel intelligent ChatEILCO capable de comprendre les questions formulées en langage naturel et de fournir des réponses précises issues des différentes sources d'information de l'EILCO. Ce chatbot s'appuiera sur un modèle d'intelligence artificielle (IA) avancé, capable d'extraire et de synthétiser les informations pertinentes à partir de bases de données et de documents numériques. ChatEILCO sera conçu comme un outil polyvalent et accessible à l'ensemble de la communauté EILCO, incluant les étudiants, les enseignants et le personnel administratif. Il permettra de centraliser les informations, d'automatiser les réponses aux questions fréquentes.

3 Objectifs Généraux du Projet :

ChatEILCO a pour objectif principal de faciliter et d'unifier l'accès à l'information au sein de l'école en s'appuyant sur les technologies d'intelligence artificielle. Plus concrètement, les objectifs du projet sont les suivants :

1. Centraliser l'ensemble des sources d'information pertinentes (bases de données internes, documents numériques, ressources pédagogiques, etc.) dans un système unique et interrogeable en langage naturel.
2. Répondre aux questions fréquentes afin de réduire la charge de travail humaine liée à la gestion des demandes répétitives.
3. Améliorer la rapidité et la précision de la recherche d'informations grâce à l'intégration d'un modèle d'intelligence artificielle adapté aux besoins de l'école.

4 Organisation Générale

Le projet est conduit de manière collaborative entre les sites universitaires de Calais et de Saint-Omer. L'équipe projet est composée des membres suivants :

- **Équipe étudiante**
  * TAHEREDDINE Mohamed Taha – Étudiant en Génie Informatique, site de Calais
  * SALAH Ibrahim – Étudiant en Génie Informatique, site de Calais
  * BAKOUCH Youssef – Étudiant en Génie Industriel, site de Saint-Omer
- **Équipe d'encadrants pédagogiques**
  * M. BAUTISTA Esteban – CPJ, LISIC
  * M. LEMAIRE Adrien – Doctorant, LISIC
  * M. ROUSSEL Gilles – PR, LISIC

L'unité principale de traitement et de développement est localisée au LISIC de Saint-Omer, où seront disponibles les ressources matérielles et logicielles nécessaires au déploiement et à l'exécution du modèle d'intelligence artificielle. Comme mentionné précédemment, la collaboration inter-sites constitue un aspect central du projet. Afin d'assurer une communication fluide et un travail efficace entre les membres de l'équipe, plusieurs outils collaboratifs seront employés, tels que Mattermost pour la messagerie d'équipe et SSH pour l'accès distant sécurisé entre les environnements de développement de Saint-Omer et de Calais. Les outils de gestion de versions et de coordination technique seront détaillés dans la section relative à l'environnement technique et à la méthodologie. Concernant le suivi du projet, des réunions hebdomadaires seront organisées afin de faire le point sur l'avancement des travaux, de partager les difficultés rencontrées et de discuter des améliorations ou nouvelles idées à intégrer.

II Spécifications Fonctionnelles

1 Fonctions Principales du Chatbot

ChatEILCO doit permettre :

a) **Compréhension du langage naturel :** être capable d'interpréter les questions des utilisateurs (étudiants, enseignants, personnel administratif) formulées en langage courant.

b) **Recherche d'information intelligente :** extraire les réponses pertinentes à partir de multiples sources de données (documents PDF, Word, etc.).

c) **Synthèse et restitution :** présenter les informations de manière concise, claire et structurée afin de faciliter la compréhension et la prise de décision de l'utilisateur, tout en indiquant la source dont elles sont issues.

d) **Gestion des questions récurrentes :** automatiser les réponses aux demandes fréquentes afin de réduire la charge de travail du personnel.

e) **Contextualisation :** prendre en compte le contexte de la conversation et les questions posées précédemment durant la même session pour fournir une réponse plus pertinente.

2 Sources de Données Cibles

Le chatbot devra être en mesure d'exploiter principalement des documents textuels, tels que les fichiers aux formats Word (.docx), Markdown (.md) et PDF (.pdf). Chaque source devra être préparée et indexée pour assurer une récupération rapide et fiable des informations.

3 Exigences de Généricité et de Transférabilité

Le système doit rester réutilisable pour d'autres contextes ou institutions (académiques ou industrielles). La documentation complète du système, incluant la configuration, l'entraînement du modèle et les méthodes d'indexation, doit permettre à d'autres équipes de réentraîner ou d'adapter le chatbot sur de nouvelles données. Le système doit prévoir une modularité suffisante pour intégrer facilement de nouvelles sources de données ou de nouvelles fonctionnalités dans le futur.

III Spécifications Techniques Détaillées

1 Contraintes

Avant de déterminer les modèles d'intelligence artificielle à utiliser, il est essentiel de citer l'unité principale de traitement qui sera utilisée pour le développement et le déploiement du projet. Le laboratoire LISIC a mis à disposition un ordinateur avec les spécifications suivantes :

- CPU : 64 cœurs
- GPU : NVIDIA RTX A5000 24 Go
- RAM : 128 Go
- Stockage : suffisant pour les besoins du projet (à confirmer selon la taille finale des jeux de données et modèles)

Le système RAG à concevoir devra être développé en tenant compte de ces contraintes matérielles.

2 Modèle d'IA Principal

Le modèle d'intelligence artificielle à sélectionnés doit répondre aux critères suivants :

- Open source, afin de garantir la flexibilité et la réutilisabilité du système.
- Bonne compréhension du français, pour assurer des réponses précises et naturelles aux questions des utilisateurs de l'EILCO.
- Compatibilité avec les contraintes matérielles, en particulier ne pas dépasser 20 Go de VRAM GPU, pour s'adapter à l'ordinateur mis à disposition.

Après analyse des modèles de génération disponibles, les options suivantes ont été retenues pour étude :

- Mistral-Small-3.2-24B-Instruct-2506-GGUF (Q6_K) : taille de VRAM estimée à 19,3 Go, proche de la limite maximale.
- Mistral-Nemo-Instruct-2407-GGUF (Q8) : taille de VRAM estimée à 13 Go, offrant une marge confortable pour les opérations du système.

Pour l'embedding des documents, nous utiliserons un modèle différent :

- jina-embeddings-v4 : Il s'agit d'un modèle d'embeddings développé par Jina AI, conçu pour convertir du texte en vecteurs numériques (représentations vectorielles) exploitables dans des moteurs de recherche, des systèmes de récupération d'information ou des projets RAG (Retrieval-Augmented Generation). Ce modèle est optimisé pour capturer le sens sémantique des textes, permettant ainsi de retrouver des documents ou passages pertinents même lorsque les mots exacts ne correspondent pas. Il est rapide, efficace et adapté à de grandes collections de documents, offrant une bonne qualité d'embeddings pour des tâches de recherche et de similarité sémantique.

Ces modèles seront évalués selon leur performance en français, leur rapidité de réponse, ainsi que leur capacité à être intégrés dans un système RAG adapté aux documents et bases de données de l'EILCO.

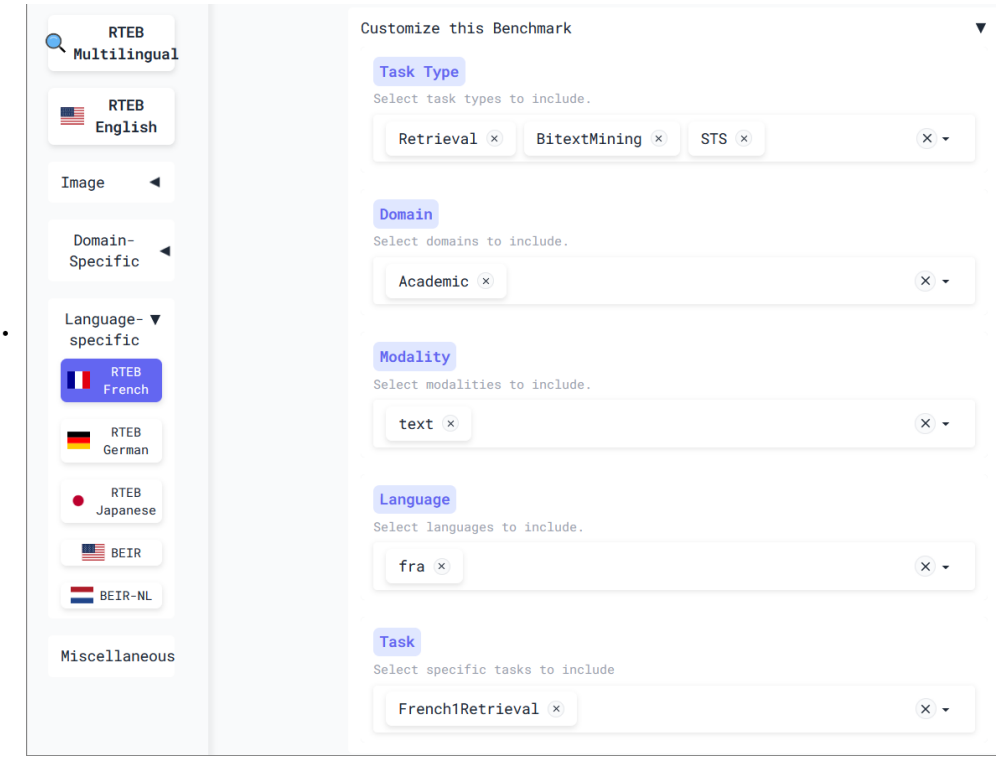

FIGURE 1 – Critères de sélection du modèle d'embdedding

Rank (Bor…	Model	Memory U…	Number of Pa…	Embedding Di…	Max Tokens	Mean (Publ…	Mean (Priva…	Mean (T…
1	Cohere-embed-v4.0	Unknown	Unknown	1536	128000		88.84	88.84
2	Cohere-embed-v4.0 (output_dtype=int8)	Unknown	Unknown	1536	128000		88.83	88.83
3	gemini-embedding-001	Unknown	Unknown	3072	2048		87.81	87.81
4	Qwen3-Embedding-8B	28866	7B	4096	32768		86.95	86.95
5	jina-embeddings-v4	7500	3B	2048	32768		86.87	86.87
6	Cohere-embed-v4.0 (output_dtype=binary)	Unknown	Unknown	1536	128000		86.47	86.47
7	voyage-3-large	Unknown	Unknown	1024	32000		85.44	85.44
8	voyage-3.5	Unknown	Unknown	1024	32000		84.68	84.68
9	voyage-3.5 (output_dtype=int8)	Unknown	Unknown	1024	32000		84.62	84.62
10	GritLM-7B	13813	7B	4096	32768		84.26	84.26

FIGURE 2 – Ranking des modèles d'embedding

3 Framework d'inférence

Nous prévoyons d'utiliser le framework llama.cpp, comme moteur d'inférence pour les modèles .GGUF.
Ceci va nous permettre d'utiliser des modèles quantifiés pour réduire la VRAM nécessaire et d'accélérer l'inférence.

Remarque :

Le binding ou package `llama-cpp-python` compile `llama.cpp` depuis la source et utilise probablement la dernière version disponible, qui est modifiée régulièrement et de manière constante. Cela peut engendrer des problèmes de compatibilité lorsqu'il s'agit de déployer le livrable final ou de reproduire exactement les résultats du développement.

Solution proposée :

- Au début du développement, noter précisément la version ou le commit de `llama.cpp` utilisé.
- Créer probablement un fork de `llama-cpp-python`, où le `Makefile` sera modifié pour forcer l'utilisation de la version spécifique initialement téléchargée, assurant ainsi la stabilité et la reproductibilité du projet.

4 Compatibilité entre le format du modèle et l'outil d'inférence

les modèles LLM séléctionnés seront convertis en format .gguf à cause de la compatibilité et l'efficacité, au contraire de .safetensors qui exige beaucoup de ressources.

5 Partie Gestion des Données et RAG

Concernant cette partie, on propose d'implémenter une architecture RAG pour satisfaire les besoins du projet.

5.1 Architecture globale

Cette architecture offre une vue d'ensemble macroscopique du traitement des requêtes des utilisateurs (étudiants, personnels, etc.) afin de fournir des réponses contextualisées et pertinentes.

Figure 3 – Architecture globale

5.2 Architecture RAG

La partie RAG sera chargée de récupérer les informations pertinentes à partir de la base de connaissances construite à partir des documents internes de l'EILCO, puis de les enrichir en fonction de la requête de l'utilisateur.

- pipe RAG : langChain pour orchestrer le retrieval, prompt, génération.
- Retriver : FAISS pour le stockage d'embeddings et effectuer la recherche.
- LLM de génération : llama.cpp pour charger le modèle .gguf et générer du texte.

6 Pipeline de création de la base de connaissance

Ce pipeline a pour objectif de traiter les données internes de l'EILCO (documents texte, fichiers PDF, etc.) afin de constituer la base de connaissance du projet.Il s'appuie sur une série de modules interdépendants qui assurent la transformation des données brutes en informations exploitables pour un système RAG (Retrieval-Augmented Generation).

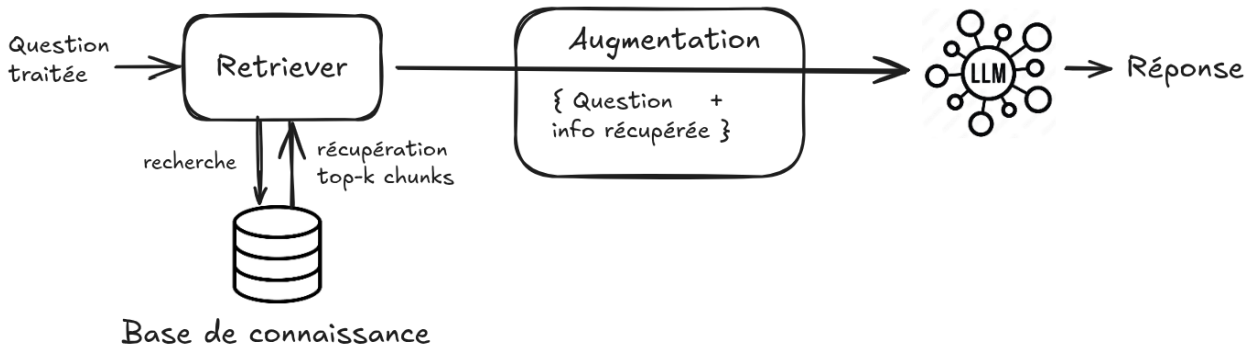

FIGURE 4 – Architecture RAG

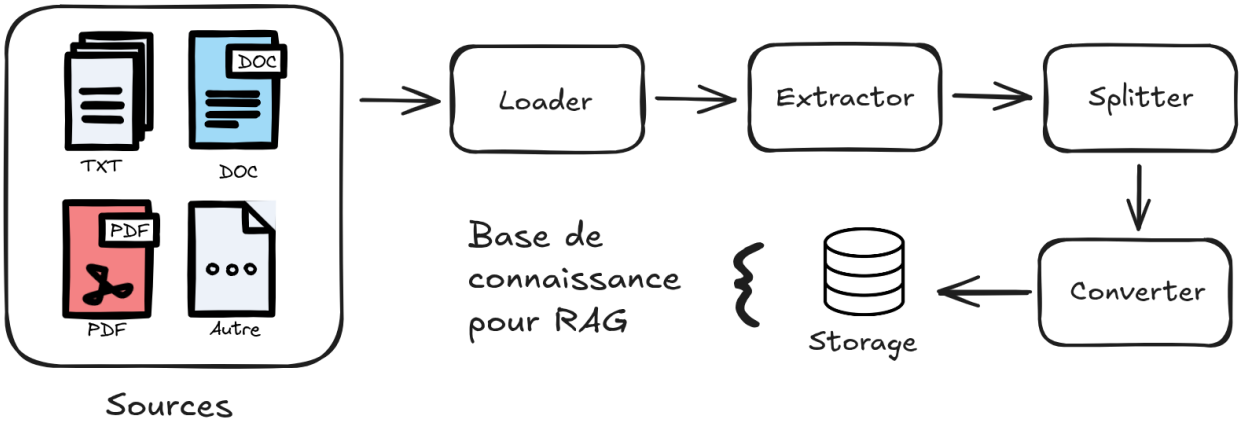

FIGURE 5 – Pipeline de création de la base de connaissances

a) **Module Loader** est chargé de la collecte et de l'importation des sources de données.Il prend en entrée différents formats (TXT, DOC, PDF, etc.) et les charge dans le système sous une forme uniforme.
   - Entrées : fichiers hétérogènes provenant des systèmes internes.
   - Sorties : flux de données normalisées prêtes à être traitées.
   - Objectif : garantir une ingestion fiable et homogène de toutes les sources d'information.
   - Outil : DocumentLoaders de langchain.

b) **Module Extractor :** a pour rôle d'extraire le contenu textuel pertinent des documents chargés.Il identifie et nettoie les données utiles en supprimant les éléments non significatifs (en-têtes, tableaux vides, métadonnées inutiles, etc.).
   - Entrées : données brutes issues du Loader.
   - Sorties : texte structuré et exploitable.
   - Objectif : isoler les informations pertinentes et préparer les données pour leur segmentation.
   - Outils : pandas, PyMuPDF, spire-doc.

c) **Module Splitter :** divise le contenu textuel extrait en unités logiques de connaissance (par exemple, des paragraphes ou des phrases).Cette segmentation facilite les traitements ultérieurs tels que l'indexation ou l'embedding.
   - Entrées : texte nettoyé et extrait.
   - Sorties : fragments textuels cohérents et de taille optimale.

- Objectif : assurer une granularité fine pour l'interrogation et la recherche d'information.
- Outil : RecursiveTextSplitter de langchain

d) **Module Converter :** Le module Converter transforme les fragments textuels en représentations vectorielles (embeddings) ou en formats adaptés au stockage.Il s'agit d'une étape de préparation à la recherche sémantique.

- Entrées : segments de texte produits par le Splitter.
- Sorties : données converties en vecteurs ou en formats indexables.
- Objectif : rendre la base de connaissance interrogeable de manière intelligente.
- Outil : llama.cpp-pyhton avec l'utilisation du model d'embedding jina-embeddings-v4 converti en .gguf avec la quantification si il y a un dépassement des ressources.

e) **Module Storage :** constitue le référentiel central de la base de connaissance.Il stocke les vecteurs ou documents traités dans une base de données optimisée pour la recherche et la récupération d'informations (par exemple, une base vectorielle).

- Entrées : données converties.
- Sorties : base de connaissance prête pour le système RAG.
- Objectif : permettre une recherche rapide, précise et contextuelle.
- Outil : FAISS(*Facebook AI similarity search*)

7 Stack technologique

La réalisation du projet ChatEILCO s'appuiera sur les technologies suivantes :

7.1 Langage et Frameworks de Développement

1. Python : Le langage de programmation principal pour le développement de la solution sera Python.
2. IA et LLMs : L'intelligence artificielle, spécifiquement les Large Language Models (LLMs), est au cœur du projet.
   - Langchain : Ce framework sera utilisé pour orchestrer les chaînes de traitement.
   - Huggingface Embedding : L'utilisation de modèles d'embedding de Huggingface est requise pour transformer les documents numériques (tels que PDF, Word, etc.) en vecteurs.

7.2 Gestion des Données et Préparation

- Interaction avec les bases de données : Le projet vise à développer un modèle d'IA capable d'interroger efficacement les bases de données propres à l'EILCO, ainsi que de rechercher des informations dans des documents numériques.
- Préparation de Données : Le travail inclura la constitution et la préparation de jeux de données pertinents pour la validation du prototype.
- Utilisation de FAISS :FAISS (*Facebook AI Similarity Search*) sera utilisé pour l'indexation et la recherche rapide de vecteurs d'embedding. Cet outil permet de stocker efficacement les représentations numériques des documents et d'effectuer des recherches de similarité à grande échelle. Il jouera un rôle central dans la phase de *Retrieval* du système RAG (Retrieval-Augmented Generation), en identifiant les passages les plus pertinents à partir des requêtes formulées par les utilisateurs.

7.3 Contrôle de Version et Déploiement

- Git : Le système de contrôle de version Git sera utilisé.
- Docker : L'outil de conteneurisation Docker sera employé pour garantir la reproductibilité de l'environnement de travail et l'intégration du système dans le contexte de l'EILCO. L'utilisation de Docker est essentielle pour conserver une généricité permettant de réutiliser les avancées du modèle initial

Remarque :
Les versions des bibliothèques et dépendances utilisées dans le projet (Python, etc.) peuvent également influencer le comportement et la reproductibilité. Il est donc important de les suivre avec précision.

7.4 L'application Web

L'application web représente le point d'accès principal pour les utilisateurs au chatbot. Grâce à son interface graphique, elle facilite l'interaction avec le système en offrant une expérience claire, fluide et ergonomique.

- Backend : pour le backend on va utiliser *FastAPI*, qui est un framework Web moderne et rapide (haute performance), pour la construction d'API avec Python basé sur des indices de type Python standard.
- Frontend : Nous allons utiliser le framework performant *ReactJS* afin d'assurer une expérience utilisateur fluide et une meilleure expérience développeur.

IV Gestion de projet

1 Analyse des Risques et Mesures Envisagées

L'analyse des risques est essentielle pour la réussite du projet ChatEILCO. Ce projet vise à simplifier et accélérer l'accès aux informations internes de l'EILCO en sélectionnant et en intégrant un modèle d'IA capable d'interroger à les documents numériques (PDF, Word...).

1.1 Risques Liés aux Ressources et aux Choix Technologiques

- **Risque de Coordination entre les Deux Sites**
  - *Problématique* : L'équipe est répartie sur deux sites distincts, avec Taha et Ibrahim à Calais et Youssef à Saint-Omer. Cette séparation géographique peut entraîner des difficultés de communication, un manque de synchronisation dans le développement (*frontend*, *backend, RAG, prompting*) et des problèmes de compatibilité entre les composants.
  - *Mesure de Mitigation* : Pour garantir la cohérence et l'intégration harmonieuse du projet, des réunions de coordination régulières seront organisées. L'équipe utilisera également des outils collaboratifs (**Git** pour le versionnement, documentation partagée et conteneurisation **Docker**) afin d'assurer la compatibilité et la continuité des développements entre les deux sites.
- **Limite Matérielle et Performance du Modèle IA**
  - *Problématique* : La quantité de mémoire GPU disponible pour l'inférence est limitée à 24 Go de VRAM. Si le grand modèle de langage (LLM) choisi est trop volumineux, cela compromettrait les performances optimales et réduirait l'espace nécessaire pour le contexte de la conversation (une fonction clé pour le système).
  - *Mesure de Mitigation* : Pour garantir des performances optimales, nous avons opté pour la sélection d'un modèle LLM de taille inférieure à 20 Go. Ce choix assure la fluidité du service et conserve la marge nécessaire pour l'intégration du contexte et du processus de *retrieval* de données.
- **Gestion des Dépendances Logicielles et Incompatibilités**
  - *Problématique* : L'intégration du système dans le contexte de l'EILCO nécessite la mobilisation de compétences variées (Python, IA, SQL, Git), et la gestion des dépendances logicielles complexes sur le serveur de déploiement peut générer des conflits et des difficultés d'intégration.
  - *Mesure de Mitigation* : La solution retenue est l'utilisation de **Docker**. La conteneurisation permet d'isoler l'environnement du ChatEILCO, garantissant ainsi une gestion fiable et reproductible des dépendances et facilitant l'intégration et le transfert du système.

1.2 Risques Liés à la Qualité et au Retrieval des Données

- **Pérennité et Actualisation des Données**
  - *Problématique* : Les documents numériques consultés peuvent ne pas être mis à jour régulièrement, ce qui mènerait le chatbot à fournir des informations obsolètes ou incorrectes.

  - *Mesure de Mitigation* : Le processus de préparation des données intègrera, dans la mesure du possible, des mécanismes de vérification des dates de dernière modification. De plus, les retours des utilisateurs sur la fraîcheur de l'information seront collectés lors du suivi hebdomadaire pour identifier les sources problématiques.
- **Efficacité du Retrieval de Données (*Top-K Chunks*)**
  - *Problématique* : Même avec un modèle LLM performant, la recherche d'informations dans les documents numériques (l'extension de fonctionnalité du chatbot) peut être sous-optimale si les fragments de texte pertinents (*top-k chunks*) ne sont pas correctement sélectionnés et présentés au modèle.
  - *Mesure de Mitigation / Solution alternative* : Pour optimiser la recherche, nous allons procéder par expérimentation pour déterminer la méthode de *chunking* la plus efficace pour les documents EILCO. Parallèlement, une amélioration sera d'implémenter une étape de reformulation des requêtes utilisateur pour s'assurer que les mots-clés transmis au système de *retrieval* sont clairs et précis. Ces essais itératifs permettront de trouver l'approche la plus performante.

2 Répartition des Tâches

La répartition des tâches au sein de l'équipe repose sur les compétences techniques de chacun (Python, IA, SQL, Web, Git, Docker) et les objectifs du projet ChatEILCO, un chatbot intelligent destiné à faciliter l'accès aux ressources de l'EILCO.

2.1 Pôle RAG et Infrastructure — Taha et Ibrahim

Taha et Ibrahim collaborent sur la conception et l'implémentation du système RAG, assurant la performance, la précision et l'efficacité de la recherche d'informations.

Tâches principales

- Préparer et ingérer les données pertinentes (documents EILCO, règlements, supports pédagogiques) dans un *Vector Store*.
- Expérimenter différentes méthodes de *chunking* et de similarité pour optimiser la recherche.
- Intégrer et tester les modèles d'embeddings et le LLM du pipeline RAG.
- Conteneuriser la solution (Docker) et assurer la cohérence du déploiement.

Répartition des tâches

- **Taha** : Travailler sur le RAG (l'implémentation du Retriever), développement du **frontend React** du chatbot, connecté à l'API backend et déploiement de la solution.
- **Ibrahim** : Travailler sur le RAG (l'implémentation du Retriever) et développement du **backend FastAPI**, intégration des modèles (embeddings, LLM) et optimisation des performances du système.

2.2 Pôle Prompting, Optimisation et Benchmarking LLM et Intégraion — Youssef

Youssef est responsable de l'intégration du système, de la conception des stratégies de *prompting* et de l'évaluation du RAG afin d'assurer la cohérence et la fiabilité des réponses du chatbot.

Tâches principales

- Assurer l'intégration des différentes composantes du système (front-end et back-end), ainsi que son déploiement.
- Concevoir les prompts d'analyse de requête (*pré-retrieval*) pour reformuler les questions utilisateurs et améliorer la recherche.
- Élaborer les prompts de sortie afin de générer des réponses claires, structurées et fidèles aux sources du RAG.
- Mettre en place des mécanismes pour réduire les hallucinations et valider la qualité des réponses.
- Identifier ou concevoir des métriques pertinentes pour mesurer les performances du RAG.
- Concevoir un jeu de données (Benchmark) adapté à l'évaluation du système.

3 Planning

Le tableau ci-dessus présente la planification prévisionnelle des différentes phases du projet, réparties sur les semaines académiques (S42 à S8). Il met en évidence les périodes d'activités principales, les vacances universitaires et les périodes de cours.

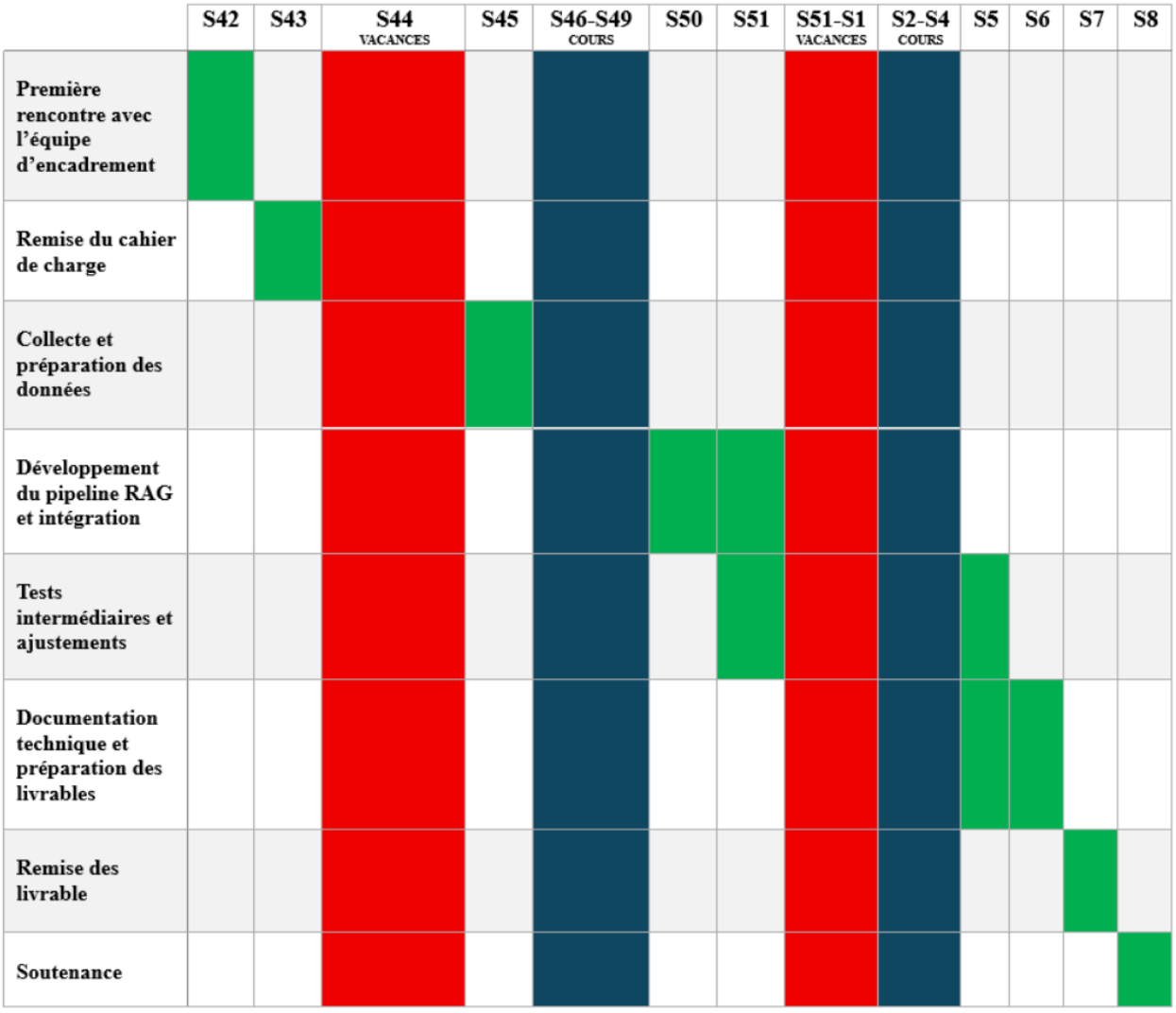

Figure 6 – Planning du projet

4 Milestones du projet

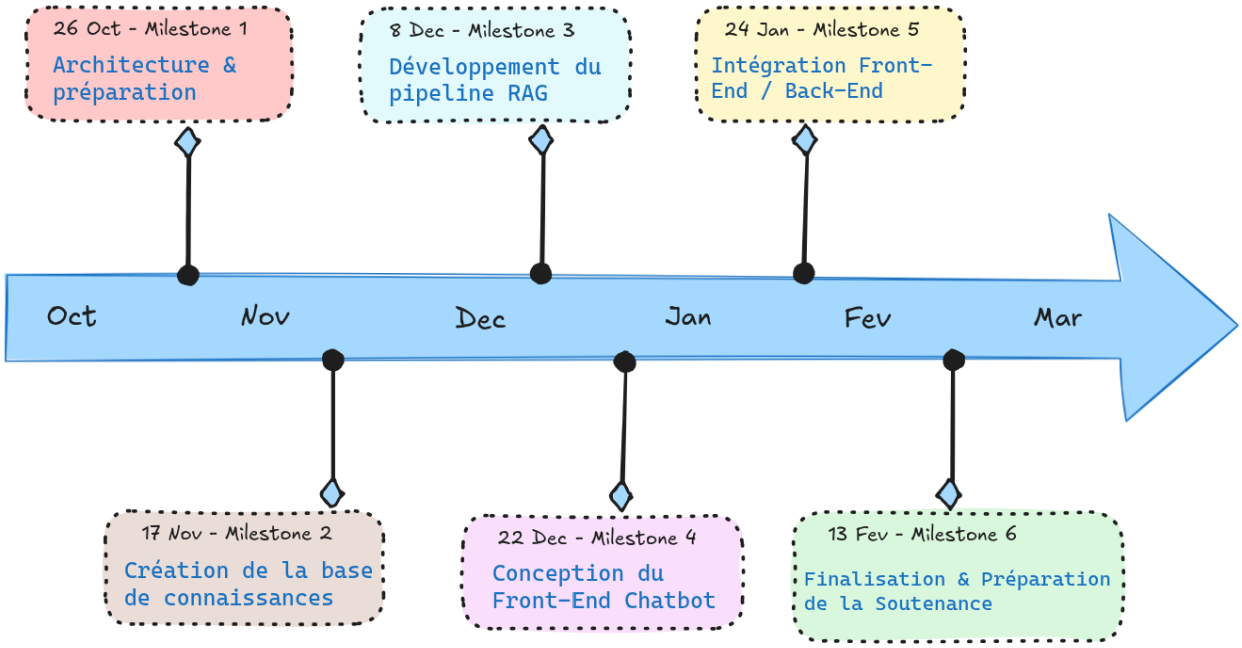

FIGURE 7 – Milestones du projet

Les principales étapes du projet ChatEILCO sont structurées en six jalons clés, garantissant une progression cohérente du cadrage à la soutenance finale :

- **Milestone 1 — Cadrage & Architecture du système (17–26 octobre)** Définition de l'architecture globale (Front-End, Back-End, Base RAG) et choix des technologies principales après la livraison du cahier des charges.
- **Milestone 2 — Création de la Base de Connaissances (4–17 novembre)** Collecte, nettoyage et préparation des données sources. Génération des embeddings et mise en place de la base vectorielle.
- **Milestone 3 — Développement du Pipeline RAG (18 novembre–8 décembre)** Conception et implémentation du retriever, du générateur (LLM) et de l'API `/ask`. Tests initiaux du pipeline complet.
- **Milestone 4 — Conception du Front-End Chatbot (9–22 décembre)** Développement de l'interface utilisateur interactive (React), intégration du design conversationnel.
- **Milestone 5 — Intégration Front-End / Back-End (23 décembre–24 janvier)** Connexion complète entre le front et le backend, vérification du flux d'échange et validation des performances globales.
- **Milestone 6 — Finalisation & Préparation de la Soutenance (25 janvier–13 février)** Stabilisation du prototype, rédaction de la documentation finale, préparation de la démonstration et de la soutenance.