Ticket 1:

1. Étude comparative des frameworks RAG (*Retrieval-Augmented Generation*) :

- **Expliquer le concept RAG** :
  - Qu’est-ce que le Retrieval-Augmented Generation ?

est une approche novatrice qui combine : la **recherche d’informations** (retrieval, qui ne génère pas de réponse originale) et la **génération** de contenu (qui ne s’appuie que sur les données de son entraînement).

Traditionnellement, les **LLM** génèrent du contenu en s’appuyant uniquement sur les informations apprises durant leur phase d’entraînement. Le **RAG,** en revanche, permet au modèle de consulter une base de données ou un corpus de documents externes en temps réel pour enrichir sa génération de texte.
=> améliore significativement la précision, la pertinence et la richesse du contenu généré.

Le processus du RAG peut être divisé en deux grandes étapes :

1. **La récupération** : Lorsque le modèle reçoit une requête, il effectue une recherche dans un ensemble prédéfini de documents ou de données pour trouver les informations les plus pertinentes par rapport à la requête. Cette recherche est souvent facilitée par des techniques d’indexation et de récupération d’informations sophistiquées.
2. **La génération** : Une fois les informations pertinentes récupérées, le modèle les utilise, en plus de sa propre connaissance interne, pour générer une réponse ou un contenu qui non seulement répond à la requête initiale mais le fait de manière plus informée et précise.

  - À quoi sert-il ? Dans quels cas est-il utilisé ?
- À quoi sert

Répondre à des questions sur des sujets **spécifiques ou à jour** (ex: documents d'entreprise, actualités, données internes).

Éviter les **hallucinations** (inventions) des modèles (**génère du contenu qui semble correct**, mais qui est **faux)** en s'appuyant sur des sources fiables.

Intégrer un **savoir privé ou propriétaire** dans un chatbot (comme des politiques RH, documentation technique, etc.).

Créer des assistants virtuels **domaines spécifiques** (médecine, droit, finance…).

- Applications du RAG

Génération de contenu

Dans le **marketing**, le RAG peut être utilisé pour créer des articles, des billets de blog, des descriptions de produits qui sont personnalisés pour le public cible, en s'appuyant sur des données de recherche pertinentes.

Ventes

Le RAG peut dynamiser les stratégies de vente en créant des propositions commerciales sur mesure qui résonnent avec les besoins et les préférences spécifiques des prospects.

Basé sur l'analyse des interactions précédentes avec un client, le LLM peut générer des scripts de vente optimisés et des points de discussion pertinents.

Support client

Le RAG permet de fournir des réponses personnalisées et précises aux requêtes des clients en accédant en temps réel à une base de données exhaustive, améliorant ainsi l'expérience client.

Les systèmes de support qui en sont équipés peuvent générer des FAQ dynamiques, répondant aux questions courantes avec des informations à jour, réduisant le volume de requêtes nécessitant une intervention humaine.

- Identifier les principaux frameworks RAG open source

1. **LangChain**

offre une architecture modulaire et extensible qui permet aux développeurs d'enchaîner divers composants, notamment des chargeurs de documents, des diviseurs de texte, des modèles d'intégration (embedding), des magasins vectoriels et des récupérateurs.

Une vaste bibliothèque d'intégrations avec plus de 700 outils, une abstraction flexible "Chain" pour construire des pipelines complexes, et un écosystème croissant d'API de niveau supérieur comme `LangGraph` pour créer des systèmes RAG agentiques et cycliques.

2. **LlamaIndex**

pour construire des applications RAG robustes et de qualité production. Sa force réside dans ses stratégies d'indexation et de récupération sophistiquées, conçues pour gérer facilement des données complexes et multimodales.

Techniques d'indexation avancées comme les index structurés en arbre et sensibles aux mots-clés, routeurs de requêtes puissants pour diriger les questions vers les sources de données les plus pertinentes, et un accent mis sur l'ingestion de données provenant d'un large éventail de sources.

3. **Haystack**

un framework mature et modulaire conçu pour construire des systèmes NLP prêts pour la production, avec un fort accent sur la RAG. Il offre une approche flexible basée sur des pipelines qui permet l'intégration transparente de divers composants, notamment des récupérateurs, des lecteurs et des générateurs.

Une architecture hautement modulaire, un support robuste pour un large éventail de bases de données vectorielles et de modèles d'intégration, et des outils d'évaluation puissants pour évaluer les performances des pipelines RAG.

4. **RAGFlow**

fournit une interface visuelle à faible code pour construire et gérer des pipelines RAG

Un éditeur visuel convivial basé sur DAG, des flux de travail RAG automatisés, et un accent sur la compréhension approfondie des documents avec des fonctionnalités comme le découpage basé sur des modèles et l'inspection visuelle des résultats d'analyse.

5. DSPy: le paradigme "Programmer, pas Propmter"

introduit un nouveau modèle de programmation pour la RAG qui déplace l'accent de l'ingénierie manuelle des prompts vers une approche plus structurée et programmatique. Il permet aux développeurs de définir les composants de leur pipeline RAG, puis utilise un optimiseur pour générer et affiner automatiquement les prompts.

6. Verba: le chatbot RAG alimenté par Weaviate

offre une interface de bout en bout, conviviale, pour interagir avec vos données via une IA conversationnelle.

7. RAGatouille: ColBERT facile à utiliser dans n'importe quel pipeline RAG

une bibliothèque spécialisée axée sur la simplification de l'utilisation de ColBERT, un puissant modèle de récupération à interaction tardive, pour les applications RAG. Elle simplifie le processus d'entraînement, d'indexation et d'utilisation des modèles ColBERT, qui peuvent souvent surpasser les méthodes de récupération denses standard.

8. Unstructured.io

Bien qu'il ne soit pas un framework RAG à part entière, Unstructured.io est un outil indispensable pour toute implémentation RAG sérieuse. Il fournit une suite de bibliothèques open source pour l'analyse et le pré-traitement de documents non structurés complexes comme les fichiers PDF, HTML et les images, les préparant pour l'ingestion dans une base de données vectorielle.

Analyse de haute qualité d'une grande variété de types de documents, extraction de métadonnées précieuses et intégration transparente avec les frameworks RAG populaires comme LangChain et LlamaIndex.

Les frameworks RAG prêts pour l'entreprise

1. Marten : La centrale de données .NET

Pour les développeurs ancrés dans l'écosystème .NET, Marten fournit une base robuste pour la construction d'applications gourmandes en données, y compris des systèmes RAG sophistiqués. Il transforme intelligemment PostgreSQL en une base de données de documents et un magasin d'événements à part entière

Son puissant support JSONB est idéal pour stocker et indexer le texte non structuré et les intégrations vectorielles qui sont au cœur de la RAG.

2. Cheshire Cat AI : Le framework d'agent personnalisable

conçu pour créer des agents d'IA conversationnels hautement personnalisables. Sa philosophie est centrée sur une architecture de plugins extensible, qui permet aux développeurs d'intégrer facilement divers LLM, magasins vectoriels et outils personnalisés pour façonner le comportement de l'agent.

3. RAGAs : Le spécialiste de l'évaluation RAG

Une fois qu'un pipeline RAG est construit, comment savoir s'il est réellement efficace ? RAGAs est un framework open source dédié spécifiquement conçu pour répondre à cette question.
Il fournit une suite de métriques pour évaluer les pipelines RAG en fonction de leur qualité de récupération et de génération, sans dépendre d'étiquettes de vérité terrain annotées par des humains.

- **Faire un benchmark comparatif** :
  Pour chaque framework, répondre aux critères suivants :
    - Langages supportés
    - Intégration avec les LLM (OpenAI, LLaMA, etc.)
    - Types de connecteurs (vecteurs, bases, API…)
    - Facilité de prise en main
    - Documentation / communauté
    - Cas d'usage typiques

Framework	Langages supportés	Intégration avec les LLM	Type de connecteurs	Facilité de prise en main	Documentation / Communauté	cas d'usage typiques
LangChain	Python JavaScript / TypeScript	OpenAI(gpt 3.5,4), Anthropic(claude), google(gemini, PaLM), Meta(LLaMA), cohere, Hugging Face, Mistral	Bases vectorielles: (Weaviate, FAISS, chroma) Bases de données: (Postgres, MongoDB, SQL) APIs externes: (http, json, agents personnalisés) outils cloud: aws s3, slack, google drive..	modulaire et flexible(débutant à expert) intégration rapide avec LLM une API chains simple pour enchaîner les étapes: prompt -> modèle -> réponse	sur github, stack overflow, discord	RAG: génération avec recherche dynamique Agent LLM Chaining complexe de prompts (workflow multi-étapes)
LlamaIndex	Python	OpenAI (GPT-3.5/4), Anthropic Hugging Face, Cohere LLaMA / Mistral / Ollama	**Bases vectorielles** : FAISS, Pinecone, Chroma, Weaviate.. **Sources documentaires** : PDF, Word, Notion, sites web, bases SQL, Markdown, JSON	Facile à utiliser en mode "plug & play" Les concepts principaux sont : Loading des documents Indexation Requêtage intelligent (RAG)	Discord + nombreux notebooks collaboratifs	Chatbot documentaire Recherche intelligente dans des fichiers Systèmes de RAG avancé (index hiérarchiques, mise à jour dynamique, etc.) Ingestion de data non structurée pour assistants internes

Haystack	Python APIs rest	OpenAI, Cohere HuggingFace Transformers Azure OpenAI, Mistral…	Très fort côté bases : Elasticsearch, OpenSearch, Weaviate, FAISS, Pinecone, Milvus Connecteurs pour : PDF, Word, CSV SQL, JSON, markdown..	Moins accessible que LlamaIndex pour les débutants Très bon pour construire des pipelines complexes RAG et de search d'entreprise Offre un UI de démo intégrable (Streamlit, FastAPI)	Slack, forum, GitHub actif	FAQ dynamique Assistant interne en entreprise Recherche sémantique dans des bases très volumineuses Intégration LLM dans des systèmes existants
RAGFlow	Python	**OpenAI** Peut s'intégrer avec d'autres modèles via LangChain ou API directe	Basique : fichiers texte, documents, vecteurs (via Chroma ou FAISS) Peut intégrer des connecteurs externes si couplé à LangChain	Très facile pour débuter Fait pour créer des pipelines RAG simples rapidement Utilise une approche déclarative claire : "data" → "prompt" → "output"	GitHub Peu de docs officielles mais code assez lisible	Mini-RAG local rapide Projets étudiants ou expérimentaux Chatbot simple avec base documentaire
DSPy	Python	OpenAI, Anthropic, HuggingFace.. Fonctionne avec API standard + intégration d'orchestrateurs	**Pas orienté data source** : ce n'est **pas un outil de RAG ou ingestion** C'est un **framework de programmation de prompts intelligents** (self-improving prompts)	Demande une **bonne maîtrise des prompts et des LLM** Convient aux chercheurs, ingénieurs IA	Bien documenté pour un projet académique (Stanford)	Optimisation automatique de prompts (type "prompt tuning") Construction de chaînes d'instructions apprenantes Evaluation et réécriture auto de réponses de LLM
Verba	**Python** API CLI	OpenAI, HuggingFace,	Markdown, PDF, HTML, Notion,	Très facile, très **haut niveau**	Projet open-sourc	Q&A sur documents

		Mistral…	Fichiers locaux… Similaire à LlamaIndex mais plus simple	API rapide : tu charges tes docs, tu fais des requêtes	Démo en ligne disponible	Assistant pour blogs, docs techniques Recherche interne d'équipe
RAGatouille	Python	OpenAI, Mistral.. Appelle les LLM via APIs, intégré à LangChain possible	Focus sur **ChromaDB** Fichiers locaux principalement (markdown, texte)	simple pour lancer un **RAG local** Idéal pour tests, POC, ou petits assistants	GitHub nombreux notebooks tests	Petit assistant documentaire personnel Tests de qualité de recherche sur un jeu de documents Assistant en local sans backend lourd
Unstructured.io	Python	Ce n'est **pas un orchestrateur de LLM** → mais utilisé **en amont** avec LangChain, LlamaIndex..	parser : PDF, Word,PowerPoint, Emails, Images (OCR intégré) Gère **la structuration de documents complexes** en éléments exploitables (titres, paragraphes, tableaux…)	Nécessite un petit script Python Peut être intégré en pipeline avec RAG	Très bien documenté	Nettoyage de documents pour ingestion LLM Prétraitement intelligent Extraction de texte avec structure sémantique (titres, métadonnées…)

- **Formuler un avis critique** :
    - Quels outils recommanderiez-vous pour un POC rapide ?

// Un **POC** est une **démonstration rapide et limitée** dans le but de vérifier **qu'une idée, une solution technique ou une technologie fonctionne** dans un contexte donné.

RAGatouille

- Ultra simple et rapide (moins de 20 lignes de code)
- Parfait pour tester un **RAG local** avec des documents
- Utilise ChromaDB (local) → pas besoin d'infra externe

Idéal pour : **chatbot sur fichiers en local, sans serveur compliqué**

Gradio (pour interface rapide)

Permet de créer **une interface web en 10 minutes**

LangChain / LlamaIndex ne sont pas idéaux pour certains POC :

Trop complexe : parfois "overkill" pour un simple test

Temps pour voir un résultat : long

Besoin pour POC	Outil recommandé
RAG sur fichiers PDF/Word	**LlamaIndex, RAGatouille**
Extraction de texte complexe	**Unstructured.io**
Orchestration plus complexe	**LangChain**

Interface web rapide	**Gradio**
Optimisation des prompts	**DSPy**

2. Étude des techniques de *prompting* en intelligence artificielle

- Expliquer ce qu'est le développement assisté par l'IA.

fait référence à l'intégration de techniques et d'outils d'intelligence artificielle dans le processus de développement de logiciels, afin d'**accroître la productivité**, d'**améliorer la qualité du code**, d**'automatiser les tâches répétitives** et d'aider les développeurs dans divers aspects de leur flux de travail.
=> tirer parti des algorithmes d'IA pour:

- analyser le code
- automatiser les tests
- générer des extraits de code
- fournir des suggestions d'amélioration
- comprendre les exigences en langage naturel
- optimiser les performances
- prendre en charge les tâches de gestion de projet.

- Identifier les outils actuels utilisés par les développeurs, par exemple :
  - GitHub Copilot
  - ChatGPT (pour le code)

1. **GitHub Copilot** : collaboration entre GitHub et OpenAI

   Intégré directement dans les environnements de développement populaires comme Visual Studio Code

   Copilot est capable de générer des fonctions entières, de compléter des lignes de code complexes et même de suggérer des tests unitaires. Son modèle de langage, basé sur GPT-3, lui permet de comprendre le contexte du projet et de fournir des suggestions pertinentes.

2. **Tabnine** : se distingue par sa polyvalence et sa capacité à s'adapter à de nombreux langages de programmation. Utilisant un modèle d'IA entraîné sur des millions de fichiers de code open source, Tabnine offre des suggestions de code contextuelles et

apprend des habitudes de codage spécifiques de chaque développeur. Sa particularité réside dans son modèle d'IA qui **peut être exécuté localement**, garantissant ainsi la confidentialité du code et réduisant la dépendance à une connexion internet.

Contrairement à GitHub Copilot, qui est basé sur le Codex d'OpenAI, Tabnine fonctionne sur des modèles d'IA propriétaires formés spécifiquement pour la complétion de code et la suggestion.

3. **Kite** : Bien que Kite propose désormais un support pour plusieurs langages, il s'est d'abord fait connaître comme un assistant IA spécialisé pour Python. Kite se démarque par sa capacité à fournir **des explications détaillées sur les fonctions et les bibliothèques,** agissant ainsi comme un tuteur virtuel pour les développeurs.

4. **Amazon Q Developer**: Contrairement aux outils traditionnels de complétion de code, Amazon Q Developer s'intègre profondément aux services AWS, fournissant des recommandations basées sur l'IA et adaptées au développement d'applications basées sur le cloud.

Il aide les développeurs à naviguer dans l'architecture AWS, à générer de l'infrastructure en tant que code (IaC) et à améliorer la sécurité et les performances des applications web.

5. **Devin AI** : Contrairement aux assistants de codage IA qui se concentrent sur l'achèvement du code, Devin AI vise à **automatiser les flux de travail de codage complexes**, ce qui en fait un outil puissant pour les développeurs travaillant sur des applications à grande échelle.

- Décrire les **cas d'usage concrets** : génération de code, complétion, documentation, tests, refactorisation…
    1. **Génération de code** : peut aider à générer des extraits de code, des fonctions ou même des modules entiers sur la base de spécifications, d'exigences ou de modèles de code existants.
    2. **Examen du code et assurance qualité** : analyser le code pour détecter les bogues potentiels, les vulnérabilités en matière de sécurité et le respect des normes de codage, et suggérer des améliorations.
    3. **Tests automatisés** : automatiser les processus de test, générer des cas de test, identifier les cas limites et enfin analyser les résultats des tests afin d'optimiser la couverture et l'efficacité des tests.
    4. **Optimisation du code** : analyser les performances du code et suggérer des optimisations pour améliorer l'efficacité, réduire la consommation de ressources ou améliorer l'évolutivité.

5. **Traitement du langage naturel (NLP)** : comprendre et interpréter les exigences en langage naturel, les commentaires des utilisateurs ou la documentation, ce qui facilite la communication entre les développeurs et les parties prenantes.
6. **Gestion de projet** : contribuer à la planification des projets, à l'affectation des ressources et à la hiérarchisation des tâches en analysant les données historiques, la dynamique de l'équipe et les dépendances du projet.
7. **Analyse prédictive** : analyser les données historiques d'un projet afin de prévoir les problèmes potentiels, d'estimer l'effort de développement et de prévoir les délais du projet avec plus de précision.

- Mettre en avant les **avantages** (gain de temps, assistance, amélioration de la productivité) et les **limites/risques** (code incorrect, dépendance, manque de compréhension...).

+

1. **Productivité accrue** : automatiser les tâches répétitives, telles que la génération de code, les tests et l'optimisation
   => libérant ainsi le temps des développeurs pour qu'ils se concentrent sur des activités plus stratégiques et à plus forte valeur ajoutée.
   =>des cycles de développement plus rapides
2. **Amélioration de la qualité du code** : analyser le code pour détecter les bogues potentiels, les vulnérabilités en matière de sécurité et le respect des normes de codage,
   => améliorer la qualité du code
   => réduire la probabilité de défauts et de problèmes de maintenance en production.
   des solutions logicielles plus fiables et plus stables.
3. **Réduction des coûts** : en automatisant les tâches manuelles et en optimisant les processus de développement, le développement de logiciels assisté par l'IA
   => aider les entreprises à réduire les coûts de main-d'œuvre
   => améliorer l'efficacité opérationnelle.
   En outre, l'identification et la résolution des problèmes plus tôt dans le cycle de développement => réduire le coût global de la maintenance et de l'assistance logicielle.
4. **Évolutivité et flexibilité** :
   gérer des charges de travail croissantes
   s'adapter à l'évolution des besoins plus facilement que les processus manuels traditionnels.
   => permettent aux entreprises de répondre rapidement aux demandes du marché, de s'adapter à la croissance...
5. **Amélioration de l'expérience client** : en fournissant plus rapidement et plus efficacement des produits logiciels de meilleure qualité
   => les entreprises peuvent améliorer l'expérience et la satisfaction globales de leurs clients.
   => conduire à des taux de rétention de la clientèle plus élevés, à une plus grande fidélité des clients

–

- **absence de Créativité et innovation** : si l'IA peut aider à accomplir des tâches répétitives et algorithmiques, elle ne dispose pas de la créativité humaine et de la pensée innovante. Le développement de logiciels nécessite souvent une résolution créative des problèmes, la conceptualisation de nouvelles idées et la compréhension des besoins des utilisateurs.
- **manque de compréhension du contexte** : le développement de logiciels implique la compréhension d'exigences commerciales complexes, des besoins des utilisateurs et des contraintes du monde réel, qui ne sont pas toujours explicitement définis ou facilement saisis par les algorithmes de l'IA.
- **Propriété intellectuelle et droits d'auteur :** Les modèles d'IA étant entraînés sur des bases de code existantes, dont certaines peuvent être soumises à des licences restrictives, la légalité et l'éthique de l'utilisation du code généré font l'objet de débats intenses.

4. Préparation d'une liste de *prompts* pour générer des offres de sécurité autour des modèles d'IA

- Identifier les **risques de sécurité liés aux modèles IA**

  (ex : prompt injection, data leakage, vol de modèle, hallucinations critiques, dérives éthiques…).

- **Injection de Prompt :** Permet aux attaquants de contourner le comportement du modèle, de faire fuiter des données confidentielles ou d'exécuter des instructions malveillantes en manipulant l'entrée.
- **Attaques par Inversion de Modèle (Model Inversion Attacks) :** Reconstruire des données d'entraînement sensibles à partir des sorties du modèle.
- **Empoisonnement de la Chaîne d'Approvisionnement (Modèle ou Jeu de Données) :** Introduire des données ou des composants malveillants dans le processus de développement ou de déploiement d'un modèle, altérant son comportement futur.
- **Abus d'API LLM & Attaques Basées sur le Débit :** Exploiter les API LLM pour des requêtes excessives, des attaques par déni de service, ou pour contourner les limites d'utilisation.
- **Jailbreaking via des Prompts Synthétiques :** Utiliser des prompts élaborés pour contourner les garde-fous du modèle et le faire générer du contenu inapproprié ou dangereux.

- **Data Leakage (Fuite de Données) :** Exposition involontaire de données sensibles à travers les sorties du modèle ou des mécanismes internes.
- **Vol de Modèle (Model Theft) :** Extraction illégale ou réplication d'un modèle propriétaire.
- **Hallucinations Critiques :** Génération par le modèle d'informations plausibles mais fausses, pouvant avoir des conséquences graves dans des contextes critiques.
- **Dérives Éthiques et Biais :** Reflet ou amplification des biais présents dans les données d'entraînement, conduisant à des résultats discriminatoires ou non éthiques.
- **Attaques par Contre-Exemples (Adversarial Attacks) :** Création d'entrées légèrement modifiées qui induisent en erreur le modèle, même si elles sont imperceptibles pour un humain.
- **Vulnérabilités dans les Bibliothèques et Frameworks IA :** Failles de sécurité au sein des logiciels et outils utilisés pour développer et déployer les modèles IA.
- **Manque de Transparence et d'Explicabilité (Explainability) :** Difficulté à comprendre pourquoi un modèle prend certaines décisions, rendant difficile l'identification et la correction des biais ou erreurs.

Formulation d'une Série de Prompts Ciblés :

Objectif : Générer une offre de cybersécurité pour entreprise/organisation.

- **Diagnostic :**
    - "Quels sont les principaux risques de sécurité liés à l'intégration de modèles d'IA générative (comme les LLM) dans une organisation du secteur [Nom du Secteur, ex : financier, santé] ? Fournissez une analyse SWOT des menaces potentielles."
    - "Élaborez un questionnaire d'évaluation des risques de cybersécurité spécifiques aux systèmes d'IA, couvrant l'ensemble du cycle de vie du modèle, de la conception au déploiement."
    - "Décrivez un scénario d'attaque par empoisonnement de données sur un modèle de détection de fraude basé sur l'IA et proposez des mesures d'atténuation techniques et organisationnelles."
- **Prévention :**
    - "Générez une proposition d'offre de services de cybersécurité pour la protection des infrastructures d'IA, incluant la sécurité des données d'entraînement, la résilience aux attaques par injection de prompt et la détection des fuites de modèles."
    - "Rédigez une politique de sécurité des LLM pour une entreprise, abordant les bonnes pratiques d'utilisation, la gestion des accès et la réponse aux incidents de sécurité liés aux modèles."
    - "Quelles solutions technologiques recommanderiez-vous pour protéger un système d'IA contre les attaques par inversion de modèle et les attaques par contre-exemples ?"
- **Conformité :**

    - "Comment une organisation peut-elle démontrer sa conformité aux réglementations comme le RGPD ou l'AI Act de l'UE en ce qui concerne la sécurité des données utilisées par ses modèles d'IA ?"
    - "Créez une liste de contrôles de sécurité (checklist) pour évaluer la conformité d'un système d'IA aux normes de sécurité cyber, en mettant l'accent sur la gouvernance et la traçabilité."

Objectif : Décrire des cas d'usage de sécurité IA.

- **Diagnostic :**
    - "Identifiez et décrivez au moins cinq cas d'usage où l'IA peut être utilisée pour renforcer la cybersécurité d'une entreprise (ex: détection d'anomalies, analyse de vulnérabilités)."
    - "Comment l'IA peut-elle aider à prévenir les attaques par phishing et l'ingénierie sociale en identifiant les schémas comportementaux suspects ?"
- **Prévention :**
    - "Expliquez comment les techniques d'apprentissage automatique peuvent être appliquées pour la détection précoce des menaces persistantes avancées (APT) dans un réseau d'entreprise."
    - "Décrivez un cas d'usage de l'IA pour l'automatisation de la réponse aux incidents de sécurité, en détaillant les étapes et les bénéfices attendus."
- **Sensibilisation :**
    - "Fournissez des exemples concrets de la manière dont les attaquants utilisent l'IA pour mener des cyberattaques sophistiquées, à des fins de sensibilisation des employés."

Objectif : Simuler un plan d'audit de modèle IA.

- **Audit :**
    - "Élaborez un plan d'audit détaillé pour évaluer la robustesse d'un modèle de classification d'images face aux attaques par contre-exemples."
    - "Quels sont les critères clés à inclure dans un audit de sécurité d'un LLM, en se concentrant sur la détection du jailbreaking et de l'injection de prompt ?"
    - "Décrivez les étapes d'un processus d'audit pour vérifier la non-divulgation de données sensibles par un modèle d'IA (fuite de données)."
    - "Proposez un cadre d'évaluation pour la résilience d'un modèle d'IA aux attaques par empoisonnement de la chaîne d'approvisionnement."
- **Diagnostic :**
    - "Quels outils et techniques d'évaluation sont essentiels pour un audit de sécurité complet des modèles d'IA, y compris l'analyse de la surface d'attaque et des vulnérabilités des API ?"

Objectif : Proposer des actions de durcissement (hardening) ou de gouvernance autour des LLM.

- **Prévention :**
    - "Proposez une liste de 10 actions de durcissement pour un LLM exposé via une API publique, incluant des mesures techniques et organisationnelles."
    - "Décrivez les meilleures pratiques pour la gestion des données sensibles utilisées pour l'entraînement des LLM, afin de minimiser le risque de fuite de données."
    - "Comment peut-on mettre en œuvre des mécanismes de filtrage d'entrée et de sortie robustes pour un LLM afin de contrer les tentatives d'injection de prompt et de jailbreaking ?"
    - "Quelles stratégies de sécurisation des API devraient être mises en place pour protéger un LLM contre les abus d'API et les attaques basées sur le débit ?"
- **Gouvernance :**
    - "Élaborez un cadre de gouvernance pour l'utilisation éthique et sécurisée des LLM au sein d'une organisation, couvrant les responsabilités, les processus de validation et les mécanismes de supervision."
    - "Comment établir un programme de Bug Bounty spécifique aux LLM pour identifier et corriger les vulnérabilités liées à l'injection de prompt et aux hallucinations critiques ?"
    - "Quelles sont les responsabilités clés d'une équipe de sécurité dans la gestion du cycle de vie sécurisé des modèles d'IA ?"
- **Sensibilisation :**
    - "Rédigez un guide de bonnes pratiques pour les développeurs et les utilisateurs de LLM, axé sur la sécurité et la prévention des dérives."

Classification des Prompts par Objectif :

Pour une meilleure organisation, les prompts sont classés par objectif principal, bien que certains puissent relever de plusieurs catégories.

1. Diagnostic :

- "Quels sont les principaux risques de sécurité liés à l'intégration de modèles d'IA générative (comme les LLM) dans une organisation du secteur [Nom du Secteur, ex : financier, santé] ? Fournissez une analyse SWOT des menaces potentielles."
- "Élaborez un questionnaire d'évaluation des risques de cybersécurité spécifiques aux systèmes d'IA, couvrant l'ensemble du cycle de vie du modèle, de la conception au déploiement."
- "Décrivez un scénario d'attaque par empoisonnement de données sur un modèle de détection de fraude basé sur l'IA et proposez des mesures d'atténuation techniques et organisationnelles."
- "Identifiez et décrivez au moins cinq cas d'usage où l'IA peut être utilisée pour renforcer la cybersécurité d'une entreprise (ex: détection d'anomalies, analyse de vulnérabilités)."
- "Comment l'IA peut-elle aider à prévenir les attaques par phishing et l'ingénierie sociale en identifiant les schémas comportementaux suspects ?"

- "Quels outils et techniques d'évaluation sont essentiels pour un audit de sécurité complet des modèles d'IA, y compris l'analyse de la surface d'attaque et des vulnérabilités des API ?"

2. Prévention :

- "Générez une proposition d'offre de services de cybersécurité pour la protection des infrastructures d'IA, incluant la sécurité des données d'entraînement, la résilience aux attaques par injection de prompt et la détection des fuites de modèles."
- "Rédigez une politique de sécurité des LLM pour une entreprise, abordant les bonnes pratiques d'utilisation, la gestion des accès et la réponse aux incidents de sécurité liés aux modèles."
- "Quelles solutions technologiques recommanderiez-vous pour protéger un système d'IA contre les attaques par inversion de modèle et les attaques par contre-exemples ?"
- "Expliquez comment les techniques d'apprentissage automatique peuvent être appliquées pour la détection précoce des menaces persistantes avancées (APT) dans un réseau d'entreprise."
- "Décrivez un cas d'usage de l'IA pour l'automatisation de la réponse aux incidents de sécurité, en détaillant les étapes et les bénéfices attendus."
- "Proposez une liste de 10 actions de durcissement pour un LLM exposé via une API publique, incluant des mesures techniques et organisationnelles."
- "Décrivez les meilleures pratiques pour la gestion des données sensibles utilisées pour l'entraînement des LLM, afin de minimiser le risque de fuite de données."
- "Comment peut-on mettre en œuvre des mécanismes de filtrage d'entrée et de sortie robustes pour un LLM afin de contrer les tentatives d'injection de prompt et de jailbreaking ?"
- "Quelles stratégies de sécurisation des API devraient être mises en place pour protéger un LLM contre les abus d'API et les attaques basées sur le débit ?"

3. Conformité :

- "Comment une organisation peut-elle démontrer sa conformité aux réglementations comme le RGPD ou l'AI Act de l'UE en ce qui concerne la sécurité des données utilisées par ses modèles d'IA ?"
- "Créez une liste de contrôles de sécurité (checklist) pour évaluer la conformité d'un système d'IA aux normes de sécurité cyber, en mettant l'accent sur la gouvernance et la traçabilité."

4. Audit :

- "Élaborez un plan d'audit détaillé pour évaluer la robustesse d'un modèle de classification d'images face aux attaques par contre-exemples."
- "Quels sont les critères clés à inclure dans un audit de sécurité d'un LLM, en se concentrant sur la détection du jailbreaking et de l'injection de prompt ?"

- "Décrivez les étapes d'un processus d'audit pour vérifier la non-divulgation de données sensibles par un modèle d'IA (fuite de données)."
- "Proposez un cadre d'évaluation pour la résilience d'un modèle d'IA aux attaques par empoisonnement de la chaîne d'approvisionnement."

5. Gouvernance :

- "Élaborez un cadre de gouvernance pour l'utilisation éthique et sécurisée des LLM au sein d'une organisation, couvrant les responsabilités, les processus de validation et les mécanismes de supervision."
- "Comment établir un programme de Bug Bounty spécifique aux LLM pour identifier et corriger les vulnérabilités liées à l'injection de prompt et aux hallucinations critiques ?"
- "Quelles sont les responsabilités clés d'une équipe de sécurité dans la gestion du cycle de vie sécurisé des modèles d'IA ?"

6. Sensibilisation :

- "Fournissez des exemples concrets de la manière dont les attaquants utilisent l'IA pour mener des cyberattaques sophistiquées, à des fins de sensibilisation des employés."
- "Rédigez un guide de bonnes pratiques pour les développeurs et les utilisateurs de LLM, axé sur la sécurité et la prévention des dérives."