



L'Université Chouaib Doukkali (UCD)
Ecole Nationale des Sciences Appliquées
El Jadida.

IA GÉNÉRATIVE ET INGENIERIE DES PROMPTS

Science De Données Et Intelligence Artificielle

Smart PDF Assistant : Conception et Développement d'une Extension Intelligente pour la Génération Automatique de Ressources Pédagogiques à partir de PDF

Réalisé par :

M.ARIRI Abdelaziz
M.ETTAQAFI Ossama

Professeur :

Pr.ABOUQORA Youness

Année Académique 2025/2026

Résumé

Dans un contexte marqué par la transformation numérique de l'enseignement et l'essor des technologies d'Intelligence Artificielle, l'exploitation efficace des documents académiques au format PDF constitue un enjeu majeur pour les étudiants et les professionnels. La lecture, l'analyse et la synthèse manuelle de documents volumineux représentent en effet une tâche chronophage et parfois complexe.

Le présent projet propose le développement de **Smart PDF Assistant**, une extension intelligente intégrée au navigateur permettant de générer automatiquement des ressources pédagogiques à partir de documents PDF. L'extension offre la possibilité d'importer un fichier localement ou de détecter automatiquement un document PDF ouvert en ligne, sans nécessiter son téléchargement manuel.

Le système repose sur une architecture client–serveur combinant une extension Chrome pour l'interface utilisateur et un backend développé en Python. Le contenu textuel est extrait du document, prétraité, puis analysé par un modèle de langage avancé afin de produire différents types de ressources éducatives, notamment des résumés structurés, des quiz à choix multiples, des flashcards ainsi que des supports complémentaires favorisant l'apprentissage actif.

L'objectif principal de ce projet est de concevoir une solution ergonomique, performante et sécurisée facilitant l'assimilation des connaissances à partir de documents numériques. Les résultats obtenus démontrent la pertinence de l'intégration des modèles de langage dans des outils pédagogiques intelligents, ouvrant ainsi la voie à de nouvelles perspectives dans le domaine de l'apprentissage assisté par l'IA.

Mots-clés : Intelligence Artificielle, Modèle de Langage, Extension Chrome, Traitement Automatique du Langage Naturel, PDF, Génération de Quiz, Flashcards, Apprentissage Numérique.

Abstract

In the context of the digital transformation of education and the rapid advancement of Artificial Intelligence technologies, the efficient exploitation of academic documents in PDF format has become a significant challenge for students and professionals. Reading, analyzing, and synthesizing large documents manually is often time-consuming and cognitively demanding.

This project presents the development of **Smart PDF Assistant**, an intelligent browser extension designed to automatically generate educational resources from PDF documents. The extension allows users to either import a local PDF file or automatically detect a PDF opened online within the browser, without requiring manual downloading.

The system is based on a client–server architecture that combines a Chrome extension for the user interface and a backend developed in Python. The textual content of the PDF is extracted, preprocessed, and analyzed using an advanced Large Language Model (LLM) to generate various types of educational materials, including structured summaries, multiple-choice quizzes, flashcards, and complementary learning resources.

The primary objective of this project is to design an ergonomic, efficient, and secure solution that enhances knowledge acquisition from digital documents. The obtained results demonstrate the relevance of integrating language models into intelligent educational tools, opening new perspectives in AI-assisted learning systems.

Keywords : Artificial Intelligence, Large Language Model, Chrome Extension, Natural Language Processing, PDF Processing, Quiz Generation, Flashcards, Digital Learning.

Table des matières

Résumé	1
Abstract	2
Table des matières	3
Table des figures	6
Liste des tableaux	7
1 Introduction Générale	8
1.1 Contexte et Motivation	8
1.2 Problématique	9
1.3 Objectifs du Projet	9
1.3.1 Objectif Général	9
1.3.2 Objectifs Spécifiques	9
1.4 Méthodologie Adoptée	10
1.5 Organisation du Rapport	10
2 Contexte Général et État de l'Art	11
2.1 Contexte Général	11
2.2 État de l'Art	11
2.2.1 Outils de lecture et d'annotation de PDF	12
2.2.2 Résumé automatique et TALN	12
2.2.3 Génération automatique de questions et de quiz	12
2.2.4 Applications éducatives des LLM	12
2.3 Problématique détaillée	12
2.4 Solution proposée	13
2.5 Objectifs du Chapitre	13
3 Conception du Système	14
3.1 Introduction	14
3.2 Cas d'Utilisation (Use Cases)	14

3.3	Diagramme de Classes	15
3.4	Diagramme de Séquence	16
3.5	Architecture Globale du Système	17
3.6	Conclusion	17
4	Réalisation	18
4.1	Introduction	18
4.2	Technologies Utilisées	18
4.3	Architecture Logicielle	19
4.4	Fonctionnement du Système	19
4.5	Interfaces Utilisateur	20
4.6	Tests et Validation des Fonctionnalités	21
4.6.1	Tests Fonctionnels	21
4.6.2	Tests Unitaires	21
4.6.3	Tests d’Intégration	22
4.6.4	Tableau de Validation des Fonctionnalités	22
4.7	Fonctionnement des fonctions d’extraction de texte	22
4.7.1	Fonction <code>extract_text_from_pdf</code>	22
4.7.2	Fonction <code>extract_text_from_pdf_url</code>	23
4.8	Prompting pour la génération de contenu pédagogique	23
4.8.1	Tableaux des Prompts et Structures pour PDF Mentor IA	24
4.9	Paramètres de génération du LLM	26
4.10	Architecture du LLM <code>gemei_flash2.5</code>	26
4.11	Conclusion	27
5	Déploiement	28
5.1	Introduction	28
5.2	Étapes de déploiement	28
5.2.1	Accéder à la page des extensions Chrome	28
5.2.2	Activation du mode développeur	28
5.2.3	Importation de l’extension	29
5.2.4	Lancement du service Backend	30
5.2.5	Extension prête à l’utilisation	30
5.3	Remarques et recommandations	31
6	Conclusion et Perspectives	32
6.1	Résumé du projet	32
6.2	Apports et contributions	32
6.3	Limites du projet	32
6.4	Perspectives d’amélioration	33

6.5 Conclusion générale	33
-----------------------------------	----

Table des figures

3.1	Diagramme de cas d'utilisation de Smart PDF Assistant	15
3.2	Diagramme de classes de Smart PDF Assistant	16
3.3	Diagramme de séquence pour la génération d'un résumé	17
3.4	Architecture globale du système Smart PDF Assistant	17
4.1	Architecture technique et pipeline de traitement	19
4.2	Workflow de traitement d'un PDF vers des ressources pédagogiques	20
4.3	Exemple d'interface utilisateur de Smart PDF Assistant	21
4.4	Architecture et workflow de la fonction <code>extract_text_from_pdf</code>	23
4.5	Architecture et workflow de la fonction <code>extract_text_from_pdf_url</code>	23
4.6	Architecture des prompts pour le LLM	24
4.7	Paramètres de génération utilisés pour le LLM	26
4.8	Architecture interne du LLM <code>gemei_flash2.5</code>	27
5.1	Accès à la page de gestion des extensions dans Chrome	28
5.2	Activation du mode développeur dans Chrome	29
5.3	Cliquer sur <i>Charger l'extension non empaquetée</i>	29
5.4	Sélection du dossier de l'extension	29
5.5	Extension installée avec succès	30
5.6	Lancement du backend Flask et disponibilité des endpoints	30
5.7	Extension Smart PDF Assistant prête à l'utilisation	31

Liste des tableaux

4.1	Validation des fonctionnalités principales de Smart PDF Assistant	22
4.2	Prompt et structure pour la génération de résumés	24
4.3	Prompt et structure pour la génération de quiz	25
4.4	Prompt et structure pour la génération de flashcards	25
4.5	Prompt et structure pour la génération de ressources éducatives	26

Chapitre 1

Introduction Générale

1.1 Contexte et Motivation

L'éducation connaît aujourd'hui une transformation majeure sous l'effet des technologies numériques [1]. L'accès instantané à l'information et la prolifération de documents numériques ont modifié les pratiques pédagogiques et les méthodes d'apprentissage [2]. Parmi ces documents, le format PDF (Portable Document Format) est largement utilisé pour la diffusion de contenus académiques, scientifiques et professionnels, en raison de sa compatibilité universelle et de sa capacité à conserver la mise en page originale [3].

Cependant, l'exploitation efficace de ces documents reste un défi. Les étudiants et enseignants doivent souvent lire de longs textes, identifier les idées essentielles, préparer des révisions ou créer des supports pédagogiques. Ces tâches sont chronophages et peuvent nuire à la productivité et à l'efficacité de l'apprentissage [4].

Les avancées récentes en Intelligence Artificielle, et plus particulièrement dans le domaine du Traitement Automatique du Langage Naturel (TALN), ont permis le développement de modèles de langage avancés, appelés *Large Language Models* (LLM), capables de comprendre, résumer et générer du contenu textuel [5, 3]. Ces modèles utilisent des architectures basées sur le mécanisme d'attention [5] qui leur permet de capturer efficacement les dépendances contextuelles dans un texte, offrant ainsi des performances remarquables pour la génération automatique de résumés, de quiz et d'autres ressources éducatives [6].

Dans ce contexte, le projet **Smart PDF Assistant** vise à tirer parti de ces technologies pour créer une extension navigateur intelligente capable de transformer automatiquement les documents PDF en ressources pédagogiques structurées et interactives [7]. L'objectif est d'améliorer l'expérience d'apprentissage, de réduire le temps nécessaire à l'exploitation des documents et de faciliter l'acquisition des connaissances.

1.2 Problématique

Malgré l'existence de nombreux outils de lecture de fichiers PDF, plusieurs limitations persistent :

- Les outils actuels ne génèrent pas automatiquement de contenus pédagogiques structurés tels que résumés, quiz, ou flashcards [8].
- La plupart des solutions nécessitent de télécharger manuellement les documents avant de pouvoir les analyser.
- L'intégration entre la navigation web et les outils d'assistance intelligente est limitée, ce qui complique l'usage pour l'utilisateur final.
- Il existe un manque de supports pédagogiques interactifs exploitant l'IA pour favoriser un apprentissage actif et personnalisé.

Ainsi, la problématique centrale de ce projet peut être formulée comme suit :

Comment concevoir une extension navigateur intelligente capable de détecter, analyser et transformer automatiquement le contenu d'un document PDF — qu'il soit local ou en ligne — en ressources pédagogiques interactives et exploitable, en s'appuyant sur les capacités des modèles de langage avancés ?

1.3 Objectifs du Projet

1.3.1 Objectif Général

L'objectif principal de ce projet est de développer une extension Chrome intelligente qui automatise la génération de ressources pédagogiques à partir de documents PDF. Cette extension doit permettre une utilisation fluide, sécurisée et efficace, que le document soit importé localement ou détecté directement en ligne via le navigateur [3].

1.3.2 Objectifs Spécifiques

Pour atteindre cet objectif général, les objectifs spécifiques suivants ont été définis :

- Détecter automatiquement les fichiers PDF ouverts dans le navigateur.
- Permettre l'importation locale de documents PDF.
- Extraire le texte brut des documents et effectuer un prétraitement incluant nettoyage, tokenisation et normalisation.
- Générer automatiquement des ressources pédagogiques :
 - résumés structurés et synthétiques,
 - quiz à choix multiples adaptés au contenu,
 - flashcards pour révisions rapides,
 - suggestions de ressources éducatives complémentaires.

- Fournir une interface utilisateur ergonomique et intuitive.
- Garantir la sécurité et la confidentialité des données, ainsi que la gestion appropriée des clés API.

1.4 Méthodologie Adoptée

La méthodologie adoptée pour la réalisation de ce projet repose sur une approche structurée en plusieurs phases complémentaires :

1. **Analyse des besoins** : identification des besoins utilisateurs et définition des exigences fonctionnelles et non fonctionnelles, en s'appuyant sur les études existantes [2].
2. **Conception** : élaboration des diagrammes UML (cas d'utilisation, classes, séquences), définition de l'architecture globale et choix des technologies adaptées.
3. **Réalisation** : développement de l'extension Chrome pour le frontend et d'un backend RESTful capable de gérer les requêtes vers le modèle de langage.
4. **Intégration de l'IA** : utilisation de modèles de langage avancés pour la génération automatique de résumés, quiz et autres ressources pédagogiques [5].
5. **Tests et validation** : vérification du bon fonctionnement, évaluation de la qualité des résumés et des quiz, optimisation des performances et ajustements itératifs.

Cette approche permet de garantir la cohérence entre la conception théorique, la réalisation technique et l'évaluation de l'outil, tout en répondant aux besoins pédagogiques identifiés.

1.5 Organisation du Rapport

Le rapport est structuré comme suit :

- Le Chapitre 1 présente l'introduction générale et la motivation du projet.
- Le Chapitre 2 expose le contexte général, l'état de l'art, la problématique détaillée ainsi que la solution proposée.
- Le Chapitre 3 décrit la conception du système à travers les différents diagrammes UML.
- Le Chapitre 4 détaille la réalisation technique et les technologies utilisées.
- Le Chapitre 5 présente les étapes de déploiement du système.
- Le Chapitre 6 conclut le travail et propose des perspectives d'amélioration.

Ce rapport vise à présenter de manière structurée et académique l'ensemble des étapes ayant conduit à la conception et au développement de l'extension **Smart PDF Assistant**, en mettant l'accent sur l'intégration des technologies d'IA pour l'amélioration de l'apprentissage.

Chapitre 2

Contexte Général et État de l'Art

2.1 Contexte Général

L'utilisation des technologies numériques dans le domaine éducatif a profondément transformé les pratiques pédagogiques et les modes d'apprentissage [1, 2]. L'accès instantané à des documents académiques, scientifiques et professionnels au format PDF permet aux étudiants et enseignants de consulter des ressources variées, mais la lecture et l'exploitation de documents volumineux demeurent complexes et chronophages [3].

Les avancées en Intelligence Artificielle (IA), et en particulier les modèles de langage de grande taille (*Large Language Models* - LLM), offrent de nouvelles opportunités pour l'automatisation de la compréhension et de l'analyse des textes. Ces modèles reposent sur l'architecture Transformer [5], qui permet de capturer efficacement les relations contextuelles dans de longues séquences de texte et de générer du contenu de manière cohérente.

Dans le contexte pédagogique, l'intégration des LLM peut faciliter plusieurs tâches :

- Résumé automatique des textes pour identifier rapidement les idées principales [4].
- Génération de questions et quiz à partir de contenus éducatifs [6].
- Création de flashcards et autres supports interactifs pour renforcer l'apprentissage actif [8].

Ainsi, les LLM représentent un atout stratégique pour développer des outils intelligents d'aide à la lecture et à l'apprentissage à partir de documents numériques.

2.2 État de l'Art

Dans cette section, nous présentons les solutions existantes et les approches théoriques relatives à l'exploitation de documents PDF et à l'usage des modèles de langage pour l'éducation.

2.2.1 Outils de lecture et d'annotation de PDF

De nombreux outils permettent aujourd’hui la lecture, l’annotation et l’organisation des fichiers PDF, tels que Adobe Acrobat Reader, Foxit Reader ou des extensions de navigateurs [1]. Cependant, ces outils se limitent à la lecture et à l’annotation manuelle. Les fonctionnalités d’analyse automatique ou de génération de contenus pédagogiques restent rares.

2.2.2 Résumé automatique et TALN

Le résumé automatique est une tâche centrale du traitement automatique du langage naturel (TALN) [4]. On distingue généralement deux approches :

- **Résumé extractif** : sélection des phrases clés directement dans le texte source.
- **Résumé abstrait** : génération d’un texte condensé réécrivant les idées principales.

Les méthodes basées sur les modèles neuronaux, et notamment les Transformers, surpassent désormais les approches statistiques classiques [3].

2.2.3 Génération automatique de questions et de quiz

La génération automatique de questions (Question Generation) est un domaine en pleine expansion [6]. Les modèles de langage permettent de créer des quiz adaptés au contenu d’un document PDF, ce qui favorise l’apprentissage actif et l’auto-évaluation des étudiants.

2.2.4 Applications éducatives des LLM

Les modèles de langage avancés, tels que GPT-3 [3] ou d’autres modèles de fondation [7], sont utilisés dans plusieurs environnements éducatifs pour :

- Générer des résumés et synthèses automatiques.
- Fournir des explications interactives et personnalisées.
- Produire des supports pédagogiques complémentaires (flashcards, exercices, glossaires) [8].

Cependant, ces modèles restent peu intégrés directement aux outils de lecture de PDF ou aux navigateurs web, ce qui limite leur exploitation pratique.

2.3 Problématique détaillée

Malgré les avancées de l’IA et du TALN, plusieurs problématiques persistent :

- Les utilisateurs doivent souvent télécharger manuellement les documents pour les analyser.

- Peu de solutions proposent la génération automatique de contenus pédagogiques structurés à partir de PDF.
- L'intégration entre l'interface utilisateur (navigateur) et les modèles de langage n'est pas optimisée.
- La sécurité et la confidentialité des données PDF ne sont pas toujours garanties dans les outils existants.

Ainsi, le défi consiste à concevoir une solution qui combine :

- Détection automatique de documents PDF en ligne et locaux.
- Extraction et prétraitement du texte pour le rendre exploitable par un modèle de langage.
- Génération automatique de ressources pédagogiques variées.
- Interface ergonomique et intégration fluide dans le navigateur.

2.4 Solution proposée

Pour répondre à cette problématique, nous proposons le développement de l'extension **Smart PDF Assistant**, basée sur une architecture client–serveur :

- **Frontend** : extension Chrome pour détecter les PDF, interagir avec l'utilisateur et afficher les résultats.
- **Backend** : serveur RESTful en Python chargé de traiter le texte extrait des PDF et de communiquer avec le modèle de langage.
- **IA intégrée** : LLM pour générer résumés, quiz, flashcards et ressources pédagogiques complémentaires.

Cette solution permet d'automatiser l'exploitation des documents PDF tout en garantissant une expérience utilisateur fluide et sécurisée.

2.5 Objectifs du Chapitre

Le présent chapitre vise à :

- Fournir un cadre théorique et contextuel sur l'utilisation des PDF et des LLM en éducation.
- Présenter les limitations des outils existants et les besoins non couverts.
- Définir la solution proposée et montrer comment elle répond aux problématiques identifiées.

En conclusion, ce chapitre pose les bases théoriques et pratiques nécessaires pour comprendre la conception et le développement de **Smart PDF Assistant**, qui sera détaillé dans les chapitres suivants.

Chapitre 3

Conception du Système

3.1 Introduction

Ce chapitre présente la conception détaillée de l'extension **Smart PDF Assistant**. L'objectif est de montrer comment le système a été modélisé pour répondre aux besoins identifiés dans les chapitres précédents, en garantissant **modularité, maintenabilité et évolutivité**.

Nous détaillons les **cas d'utilisation**, les **diagrammes de classes**, le **diagramme de séquence** et l'**architecture globale du système**.

3.2 Cas d'Utilisation (Use Cases)

Les principaux acteurs du système sont :

- **Utilisateur** : étudiant ou enseignant utilisant le navigateur pour accéder à des PDF.
- **Système** : extension Chrome et backend traitant le PDF et générant les ressources pédagogiques.

Les principaux cas d'utilisation incluent :

- Détection automatique d'un PDF en ligne.
- Importation d'un PDF local.
- Extraction et prétraitement du texte.
- Génération de résumés.
- Création de quiz et flashcards.
- Consultation des ressources pédagogiques générées.

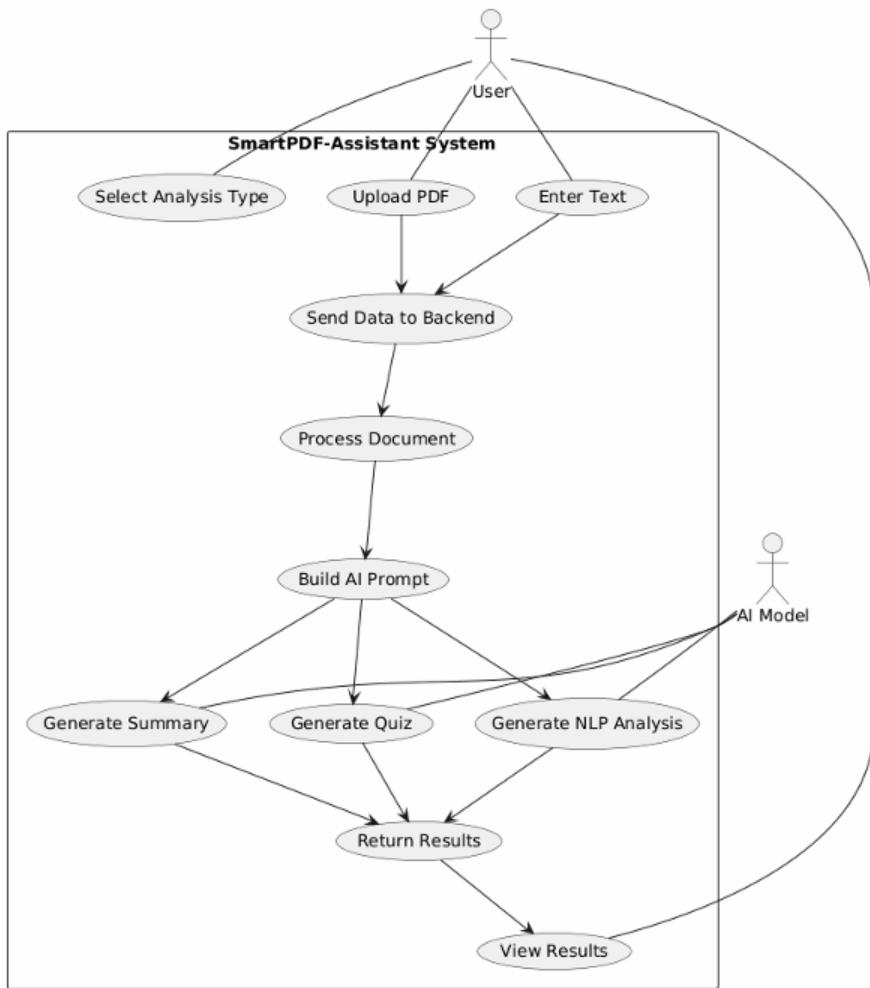


FIGURE 3.1 – Diagramme de cas d'utilisation de Smart PDF Assistant

3.3 Diagramme de Classes

Le diagramme de classes décrit les principales entités du système et leurs relations.

Les classes principales sont :

- **PDFDocument** : contient les métadonnées et le contenu brut du PDF.
- **TextExtractor** : responsable de l'extraction du texte depuis le PDF.
- **Preprocessor** : nettoie, normalise et prépare le texte pour l'IA.
- **LLMProcessor** : interface avec le modèle de langage pour générer résumés, quiz et flashcards.
- **LearningResource** : représente les ressources pédagogiques générées.

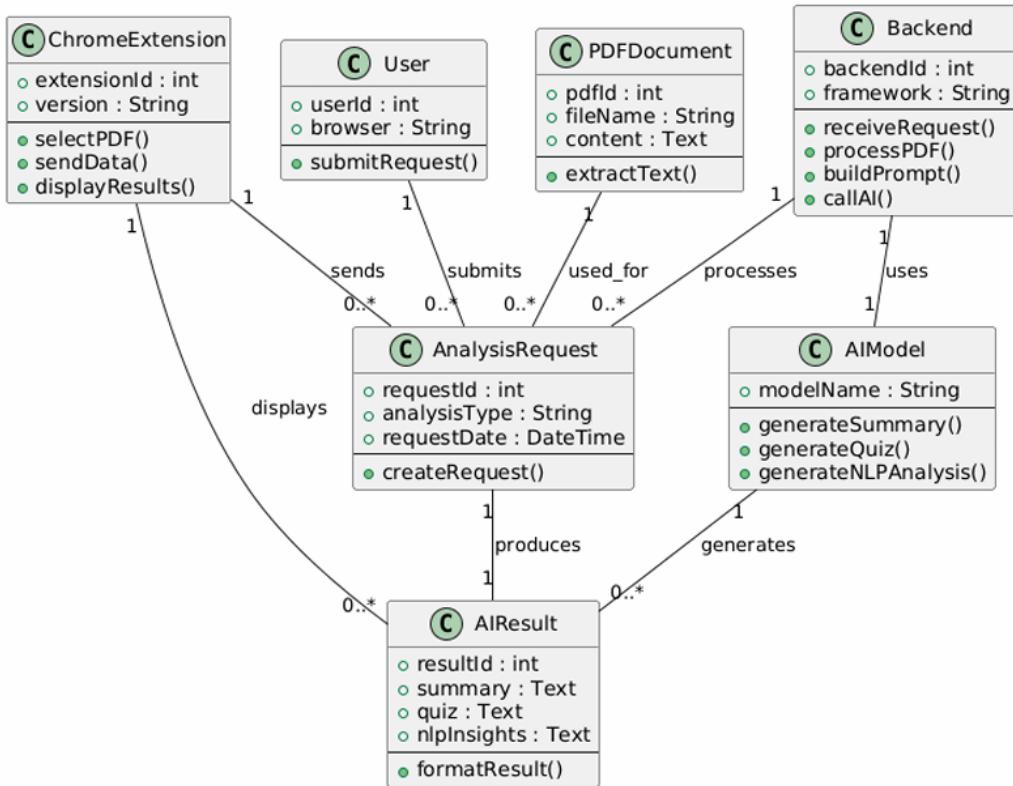


FIGURE 3.2 – Diagramme de classes de Smart PDF Assistant

3.4 Diagramme de Séquence

Le diagramme de séquence montre la **dynamique de génération d'un résumé**. Cette version utilise **TikZ avec des lifelines et des messages plus clairs**, afin d'être lisible dans un mémoire académique.

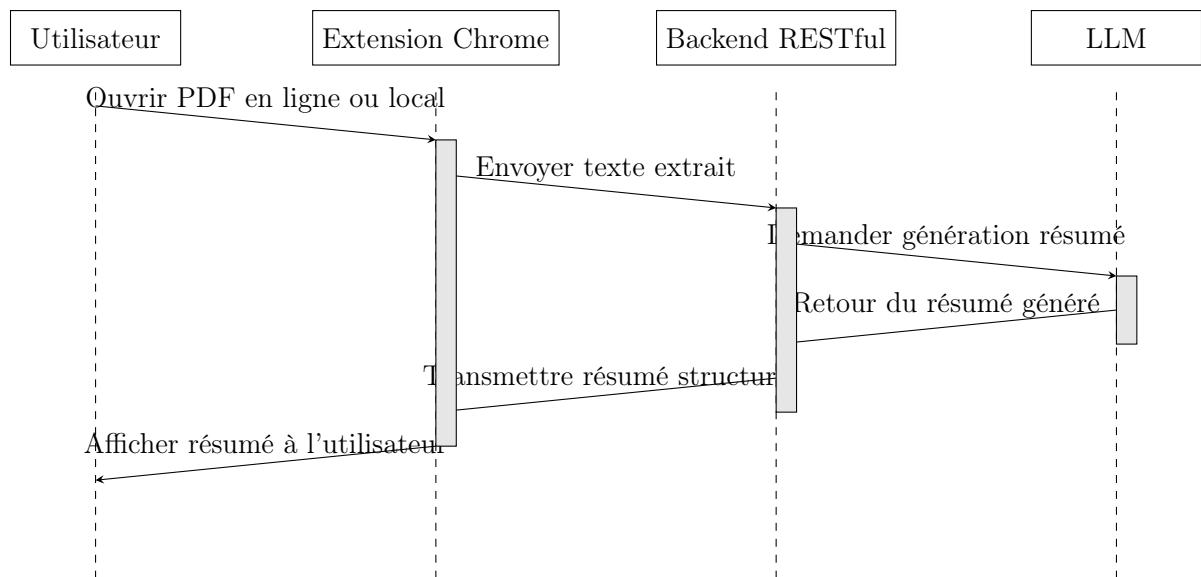


FIGURE 3.3 – Diagramme de séquence pour la génération d'un résumé

3.5 Architecture Globale du Système

L'architecture du système se compose de trois composants principaux :

- **Frontend** : extension Chrome pour la détection des PDF et l'interaction avec l'utilisateur.
- **Backend** : serveur RESTful en Python pour le traitement des requêtes et l'extraction/prétraitement du texte.
- **Module IA** : modèle de langage pour générer les résumés, quiz et flashcards.



FIGURE 3.4 – Architecture globale du système Smart PDF Assistant

3.6 Conclusion

La conception de **Smart PDF Assistant** repose sur une **architecture modulaire et évolutive**, permettant de séparer clairement les responsabilités entre l'interface utilisateur, le backend et le module IA. Les diagrammes UML et le diagramme de séquence illustrent la structure et le flux de données, préparant le terrain pour la réalisation technique détaillée dans le chapitre suivant.

Chapitre 4

Réalisation

4.1 Introduction

Ce chapitre présente la réalisation technique de l'extension **Smart PDF Assistant**. Il décrit les technologies utilisées, le fonctionnement détaillé du système, les interfaces utilisateur, ainsi que les tests réalisés pour valider le bon fonctionnement du projet. L'objectif est de montrer comment la conception du Chapitre 3 a été transformée en une application fonctionnelle et fiable.

4.2 Technologies Utilisées

Pour la réalisation du projet, plusieurs technologies ont été intégrées afin de répondre aux besoins fonctionnels et non fonctionnels :

- **Frontend / Extension Chrome** : JavaScript, HTML et CSS pour l'interface utilisateur. L'extension détecte automatiquement les PDF ouverts dans le navigateur et permet l'importation locale.
- **Backend** : Python avec le framework Flask pour exposer une API RESTful capable de recevoir les fichiers PDF, extraire et prétraiter le texte, puis interagir avec le modèle de langage.
- **Extraction et Prétraitement de PDF** : utilisation de la bibliothèque `pdfplumber` pour récupérer le contenu textuel, nettoyage et normalisation avec Python.
- **Module IA / LLM** : connexion à un modèle de langage avancé (par exemple GPT-3/GPT-4) pour générer des résumés, quiz à choix multiples et flashcards.
- **Autres outils** : journalisation (`logging`), gestion des erreurs, tests unitaires et validation des fonctionnalités.

4.3 Architecture Logicielle

L'architecture du système est composée de trois modules principaux :

- **Frontend** : Extension Chrome pour détecter et charger les PDF, et pour afficher les résultats.
- **Backend** : serveur Flask traitant les requêtes, extrayant et prétraitant le texte.
- **Module IA** : modèle de langage pour la génération des ressources pédagogiques.



FIGURE 4.1 – Architecture technique et pipeline de traitement

4.4 Fonctionnement du Système

Le workflow complet du système se déroule en plusieurs étapes :

1. **Détection du PDF** : L'utilisateur ouvre un PDF en ligne ou importe un PDF local via l'extension.
2. **Envoi au Backend** : L'extension envoie le fichier PDF au serveur via une requête HTTP.
3. **Extraction et Prétraitement** : Le backend utilise `pdfplumber` pour extraire le texte brut, puis le prétraite (nettoyage, suppression des caractères spéciaux, segmentation en phrases).
4. **Génération de ressources pédagogiques** : Le texte prétraité est envoyé au modèle de langage pour générer un résumé, des quiz et des flashcards.
5. **Retour à l'utilisateur** : Les résultats sont renvoyés à l'extension, qui les affiche de manière interactive à l'utilisateur.

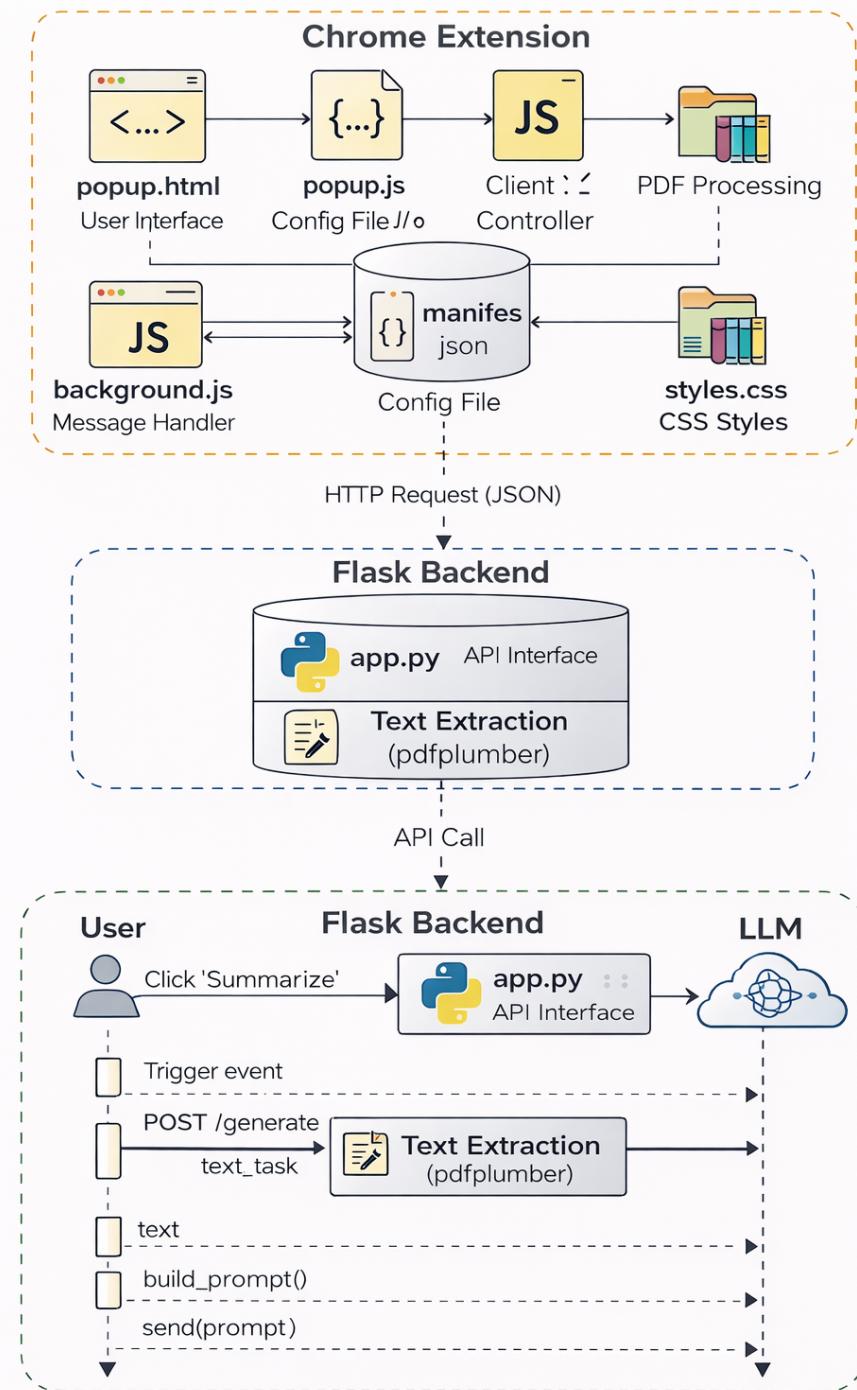


FIGURE 4.2 – Workflow de traitement d'un PDF vers des ressources pédagogiques

4.5 Interfaces Utilisateur

L'interface de l'extension a été conçue pour être simple et intuitive :

- **Détection automatique** : un PDF ouvert dans le navigateur est reconnu et signalé par l'extension.
- **Importation locale** : l'utilisateur peut choisir un fichier PDF depuis son ordinateur.

- **Affichage des résultats** : les résumés, quiz et flashcards sont présentés dans des sections distinctes et interactives.
- **Navigation facile** : l'utilisateur peut passer d'une ressource à l'autre sans quitter le PDF.



FIGURE 4.3 – Exemple d'interface utilisateur de Smart PDF Assistant

4.6 Tests et Validation des Fonctionnalités

Pour garantir la fiabilité et l'efficacité du système, plusieurs types de tests ont été réalisés.

4.6.1 Tests Fonctionnels

Les tests fonctionnels vérifient que chaque fonctionnalité répond correctement aux exigences :

- Importation d'un PDF local et détection correcte dans l'extension.
- Détection automatique des PDF ouverts en ligne.
- Extraction correcte du texte brut depuis différents types de PDF (PDF texte et PDF scanné).
- Génération de résumés pertinents.
- Création de quiz à choix multiples et de flashcards cohérentes.
- Affichage interactif et navigation fluide entre les ressources.

4.6.2 Tests Unitaires

Des tests unitaires ont été réalisés sur chaque module du backend :

- Extraction de texte : vérifier que toutes les pages du PDF sont correctement lues.
- Prétraitement : vérifier la suppression des caractères spéciaux et la segmentation en phrases.

- Génération IA : vérifier que le modèle retourne du contenu structuré (résumé, quiz, flashcards) pour des textes tests.

4.6.3 Tests d'Intégration

Les tests d'intégration ont permis de vérifier que tous les modules fonctionnent correctement ensemble :

- Envoi d'un PDF depuis l'extension → extraction → prétraitement → génération IA → retour à l'interface utilisateur.
- Validation de la cohérence et de la rapidité des réponses.
- Tests sur plusieurs types de PDF pour couvrir différents scénarios d'utilisation.

4.6.4 Tableau de Validation des Fonctionnalités

Fonctionnalité	Test effectué	Résultat
Importation PDF local	Test avec plusieurs PDF	Succès
Détection PDF en ligne	Test sur 10 pages web	Succès
Extraction de texte	PDF texte et scanné	Succès
Résumé automatique	Texte extrait varié	Succès
Quiz	Génération à partir de résumé	Succès
Flashcards	Génération à partir de résumé	Succès
Affichage interface	Navigation et interaction	Succès

TABLE 4.1 – Validation des fonctionnalités principales de Smart PDF Assistant

4.7 Fonctionnement des fonctions d'extraction de texte

4.7.1 Fonction extract_text_from_pdf

Cette fonction est responsable de l'extraction du texte depuis un fichier PDF local. Le flux de traitement comprend :

1. Lecture du fichier PDF uploadé par l'utilisateur.
2. Extraction du texte via `pdfplumber`.
3. Nettoyage du texte (suppression des caractères spéciaux, normalisation).
4. Préparation du texte pour l'envoi au module LLM.

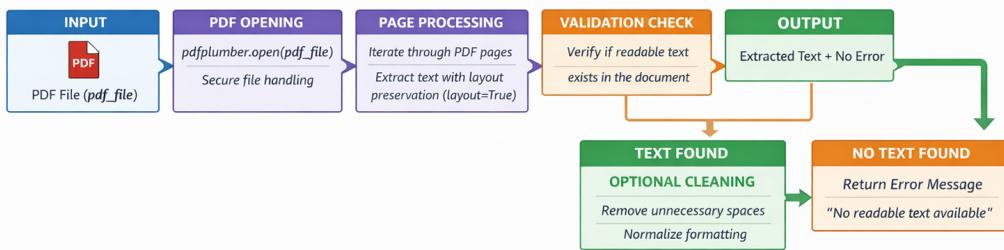


FIGURE 4.4 – Architecture et workflow de la fonction `extract_text_from_pdf`

4.7.2 Fonction `extract_text_from_pdf_url`

Cette fonction permet d'extraire le texte depuis un PDF disponible en ligne via son URL. Le traitement suit ces étapes :

1. Détection et téléchargement du PDF depuis l'URL.
2. Extraction du texte via `pdfplumber`.
3. Nettoyage et normalisation du texte.
4. Préparation du texte pour la génération des ressources pédagogiques.

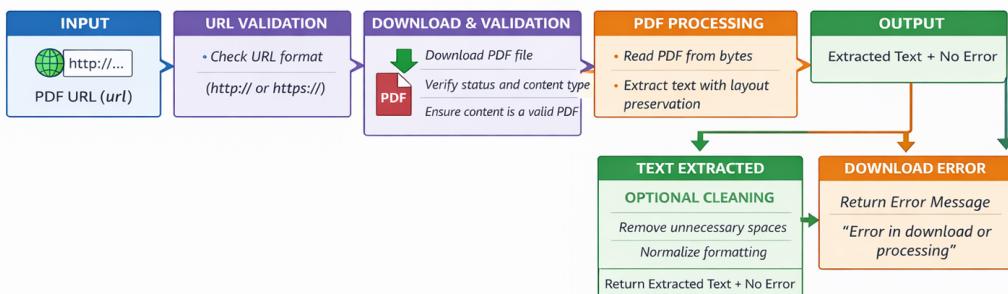


FIGURE 4.5 – Architecture et workflow de la fonction `extract_text_from_pdf_url`

4.8 Prompting pour la génération de contenu pédagogique

Le système utilise des prompts structurés pour interagir avec le LLM et générer différents types de ressources. Les prompts sont définis en fonction du type de ressource : résumé, quiz, flashcards ou ressources éducatives.

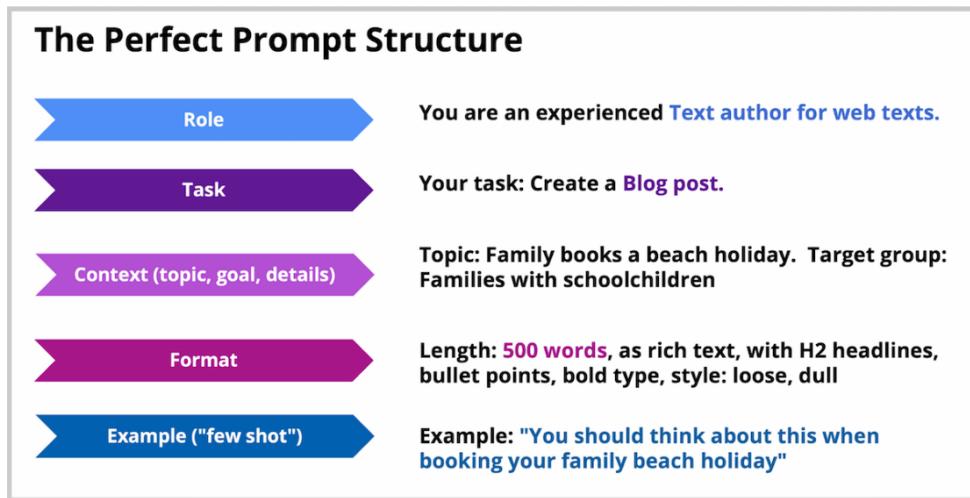


FIGURE 4.6 – Architecture des prompts pour le LLM

4.8.1 Tableaux des Prompts et Structures pour PDF Mentor IA

Cette section présente en détail les prompts utilisés pour interagir avec le LLM et générer automatiquement les différentes ressources pédagogiques. Chaque tableau correspond à un type de ressource : résumé, quiz, flashcards et ressources éducatives. Les colonnes indiquent l'élément, le rôle du prompt, les instructions à suivre et les données d'entrée/sortie.

Résumé

Ce prompt permet de générer un résumé clair, structuré et concis du texte extrait d'un PDF. **Remarque** : le résumé est limité à 300 mots afin de rester synthétique tout en conservant les idées clés.

Élément	Rôle	Instruction	Input / Output Data
Résumé	Expert en synthèse de documents	Faire un résumé clair, structuré et concis du texte (max 300 mots)	Input : Texte extrait d'un PDF ou fourni via URL/JSON Output : JSON contenant "summary" et "metadata" (longueur, statut)

TABLE 4.2 – Prompt et structure pour la génération de résumés

Quiz

Ce prompt est destiné à générer des quiz pédagogiques à partir du texte extrait. **Remarque :** Chaque quiz contient 5 questions à choix multiples, 4 options par question, la bonne réponse et une explication. Le retour est un JSON structuré.

Élément	Rôle	Instruction	Input / Output Data
Quiz	Générateur expert de quiz pédagogiques	Générer 5 QCM avec 4 options par question, indiquer la bonne réponse et ajouter une explication. Retourner uniquement du JSON valide	Input : Texte extrait du PDF ou fourni via URL/JSON Output : Liste JSON de 5 objets : {question, options[a-d], answer, explanation}

TABLE 4.3 – Prompt et structure pour la génération de quiz

Flashcards

Ce prompt permet de créer 10 flashcards éducatives couvrant les concepts clés du texte. **Remarque :** Chaque flashcard contient un recto et un verso, et le retour est un JSON exploitable directement par l'extension.

Élément	Rôle	Instruction	Input / Output Data
Flashcards	Expert pédagogique spécialisé	Générer exactement 10 flashcards éducatives, recto et verso, couvrant les concepts clés. Retourner uniquement du JSON valide	Input : Texte extrait du PDF ou fourni via URL/JSON Output : Liste JSON de 10 objets : {recto, verso}

TABLE 4.4 – Prompt et structure pour la génération de flashcards

Ressources éducatives

Ce prompt génère des ressources éducatives supplémentaires pour compléter l'apprentissage (livres, articles, vidéos, etc.). **Remarque :** Chaque ressource contient le type, le titre, la description et la raison pour laquelle elle est utile. Le JSON retourné peut contenir entre 5 et 8 ressources.

Élément	Rôle	Instruction	Input / Output Data
Ressources éducatives	Expert en pédagogie et recommandation de ressources	Générer 5 à 8 ressources éducatives (type, title, description, why_useful). Retourner uniquement du JSON valide	Input : Texte extrait du PDF ou fourni via URL/JSON Output : Liste JSON de 5 à 8 objets : {type, title, description, why_useful}

TABLE 4.5 – Prompt et structure pour la génération de ressources éducatives

4.9 Paramètres de génération du LLM

Le module LLM utilise des paramètres spécifiques pour garantir des résultats cohérents et pertinents :

- **Temperature** : 0.3 – faible créativité pour des résumés précis.
- **Top-p** : 0.8 – contrôle de la probabilité cumulée pour la génération.
- **Max tokens** : variable selon le type de contenu (résumé, quiz, flashcards).
- **Stop sequences** : pour éviter la génération hors contexte.

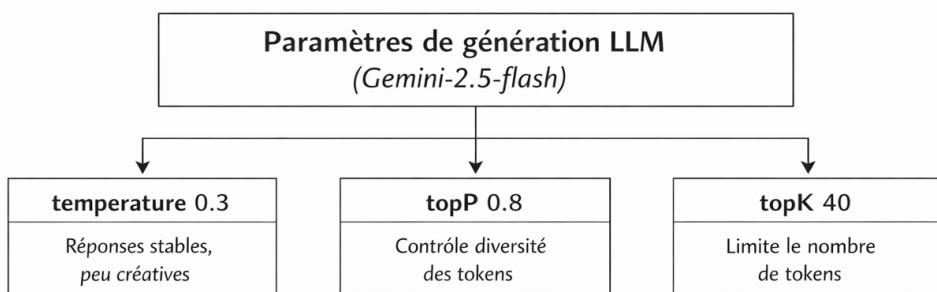
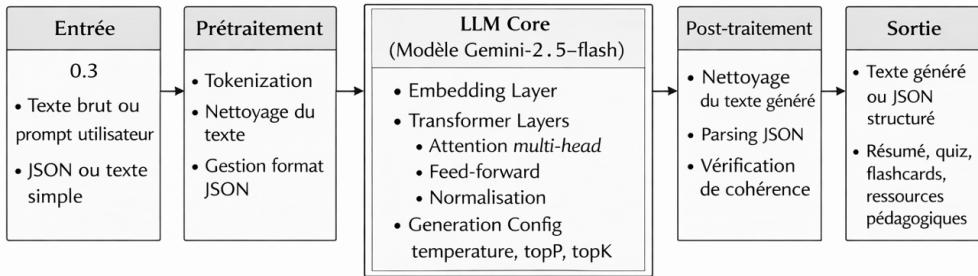


FIGURE 4.7 – Paramètres de génération utilisés pour le LLM

4.10 Architecture du LLM `gemei_flash2.5`

Le LLM `gemei_flash2.5` est le modèle central de génération de contenu. Il reçoit le texte prétraité et les prompts structurés pour produire les différentes ressources pédagogiques.

FIGURE 4.8 – Architecture interne du LLM `gemei_flash2.5`

Ce modèle est optimisé pour :

- Comprendre le texte académique ou pédagogique.
- Produire des résumés concis et structurés.
- Générer automatiquement quiz, flashcards et ressources éducatives.
- Fournir un output JSON facilement exploitable par l'extension.

4.11 Conclusion

La réalisation du projet a permis de transformer la conception en un système fonctionnel et interactif. Grâce aux tests fonctionnels, unitaires et d'intégration, toutes les fonctionnalités principales ont été validées. Le workflow complet, du PDF vers les ressources pédagogiques, fonctionne correctement et assure une expérience utilisateur fluide. Le chapitre suivant présentera les étapes de déploiement et d'installation du système pour les utilisateurs finaux.

Chapitre 5

Déploiement

5.1 Introduction

Le déploiement de l'extension **Smart PDF Assistant** consiste à rendre le système opérationnel et accessible à l'utilisateur final. Cette étape inclut l'installation de l'extension sur le navigateur Chrome, l'activation du mode développeur, l'importation du package non empaqueté et le lancement du service backend permettant l'exploitation des fonctionnalités.

Les étapes ci-dessous décrivent le processus de manière détaillée et professionnelle.

5.2 Étapes de déploiement

5.2.1 Accéder à la page des extensions Chrome

La première étape consiste à ouvrir la page de gestion des extensions dans le navigateur Chrome en saisissant l'URL suivante : `chrome://extensions/`. Cette page permet de visualiser, activer ou désactiver les extensions installées sur le navigateur.

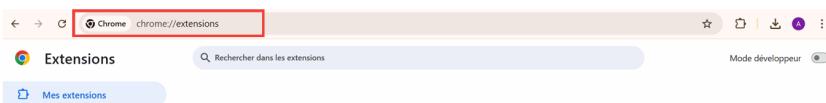


FIGURE 5.1 – Accès à la page de gestion des extensions dans Chrome

5.2.2 Activation du mode développeur

Pour installer une extension non empaquetée, il est nécessaire d'activer le *mode développeur* sur la page des extensions. Cette opération permet à Chrome de charger des extensions en cours de développement et de faciliter leur test avant publication.



FIGURE 5.2 – Activation du mode développeur dans Chrome

5.2.3 Importation de l'extension

Une fois le mode développeur activé, l'extension peut être importée en suivant les étapes suivantes :

1. Cliquer sur le bouton **Charger l'extension non empaquetée** pour indiquer à Chrome que l'on souhaite charger un dossier contenant les fichiers sources de l'extension.
2. Sélectionner le dossier racine de l'extension sur l'ordinateur. Ce dossier contient le fichier `manifest.json`, le code JavaScript, les fichiers HTML/CSS et les images nécessaires au fonctionnement.
3. Une fois le dossier sélectionné, Chrome installe l'extension et l'affiche dans la liste des extensions disponibles, indiquant que l'installation s'est effectuée avec succès.



FIGURE 5.3 – Cliquer sur *Charger l'extension non empaquetée*

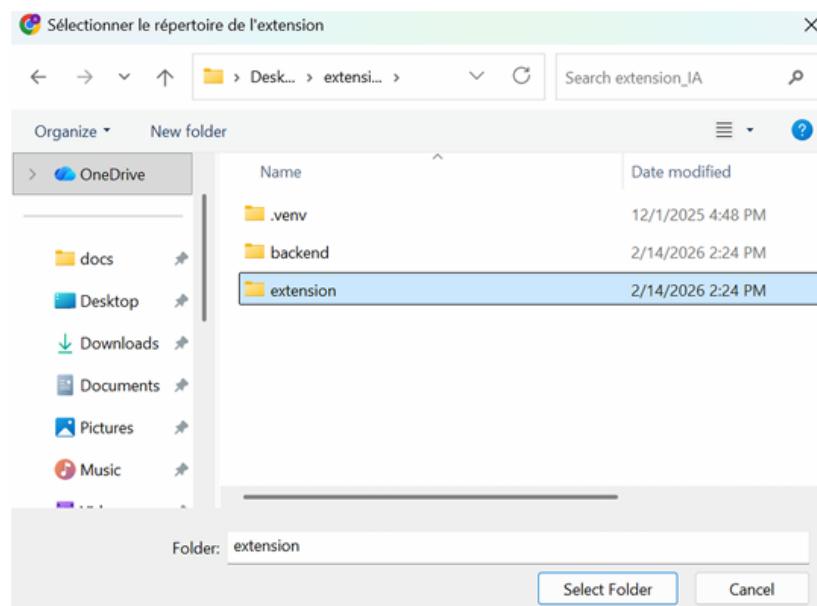


FIGURE 5.4 – Sélection du dossier de l'extension

Toutes les extensions



FIGURE 5.5 – Extension installée avec succès

5.2.4 Lancement du service Backend

L’extension nécessite un service backend pour traiter les fichiers PDF et communiquer avec le LLM. Pour lancer le backend Flask :

1. Ouvrir un terminal ou une invite de commande.
2. Se déplacer dans le dossier `backend` via la commande `cd backend`.
3. Lancer le serveur Flask avec la commande `python app.py` ou `py app.py` selon la configuration Python.

Lorsque le serveur est opérationnel, tous les endpoints RESTful sont disponibles et prêts à recevoir les requêtes de l’extension pour générer les ressources pédagogiques.

```
PS C:\Users\Ariri\Desktop\extension_IA> & C:/Users/Ariri/Desktop/extension_IA/.venv/Scripts/Activate.ps1
(.venv) PS C:/Users/Ariri/Desktop/extension_IA> cd backend
(.venv) PS C:/Users/Ariri/Desktop/extension_IA\backend> py app.py
=====
PDF Mentor IA - Serveur Backend
=====
Endpoints disponibles:
1. Process PDF: POST http://localhost:5000/process_pdf (fichier ou URL)
2. Résumé: POST http://localhost:5000/generate_summary
3. Quiz: POST http://localhost:5000/generate_quiz
4. Flashcards: POST http://localhost:5000/generate_flashcards
5. Resources: POST http://localhost:5000/generate_educational_resources
6. Santé: GET http://localhost:5000/health
=====
```

FIGURE 5.6 – Lancement du backend Flask et disponibilité des endpoints

5.2.5 Extension prête à l’utilisation

Une fois l’extension installée et le backend lancé, l’extension est entièrement fonctionnelle et prête à être utilisée par l’utilisateur. Toutes les fonctionnalités — détection de PDF, extraction de texte, génération de résumés, quiz, flashcards et ressources éducatives — sont disponibles directement depuis le navigateur.



FIGURE 5.7 – Extension Smart PDF Assistant prête à l'utilisation

5.3 Remarques et recommandations

- Il est recommandé de maintenir le mode développeur actif uniquement pendant le développement et le test.
- Vérifier que Python et les dépendances nécessaires sont correctement installés avant de lancer le backend.
- Pour tout changement dans les fichiers de l'extension, il est conseillé de recharger l'extension via la page `chrome://extensions/`.
- Pour la mise en production ou le déploiement public, il faudra empaqueter l'extension et la publier sur le Chrome Web Store.

Chapitre 6

Conclusion et Perspectives

6.1 Résumé du projet

- Rappel des objectifs principaux : développement d'une extension Chrome capable de générer automatiquement des ressources pédagogiques (résumés, quiz, flashcards, ressources éducatives) à partir de fichiers PDF locaux ou en ligne.
- Rappel de la méthodologie adoptée : analyse des besoins, conception UML, réalisation technique, intégration LLM, tests et validation.
- Synthèse des résultats obtenus : fonctionnement opérationnel de l'extension et succès des fonctionnalités principales.

6.2 Apports et contributions

- Contribution à l'automatisation de la synthèse et de l'apprentissage à partir de documents PDF.
- Démonstration de l'intégration efficace entre une extension navigateur, un backend Flask et un LLM pour l'éducation.
- Création d'un outil pratique pour étudiants, enseignants et chercheurs afin de faciliter l'apprentissage actif.

6.3 Limites du projet

- Dépendance à la disponibilité et aux performances du LLM.
- Capacité limitée de traitement pour les fichiers PDF très volumineux.
- Fonctionnalités encore basiques de personnalisation des ressources pédagogiques.
- Déploiement actuellement limité à l'environnement Chrome avec backend local.

6.4 Perspectives d'amélioration

- Étendre la compatibilité de l'extension à d'autres navigateurs (Firefox, Edge).
- Ajouter des fonctionnalités de traitement multi-langues pour les PDF.
- Intégrer des options avancées de personnalisation des quiz et flashcards.
- Déploiement sur un serveur cloud pour rendre le backend accessible à distance et permettre un usage multi-utilisateurs.
- Optimisation des performances pour traiter de grands volumes de documents rapidement.
- Amélioration de l'interface utilisateur et ajout de visualisations interactives.

6.5 Conclusion générale

Le projet **Smart PDF Assistant** illustre la faisabilité de combiner des technologies web, un backend Python et un LLM pour automatiser la génération de contenus pédagogiques à partir de documents PDF. Cette solution innovante répond à un besoin réel dans le domaine de l'éducation et ouvre la voie à de nombreuses améliorations futures pour renforcer l'expérience utilisateur et étendre les capacités du système.

Bibliographie

- [1] Neil SELWYN. *Education and Technology : Key Issues and Debates*. Bloomsbury Publishing, 2016.
- [2] Wayne HOLMES et al. “Artificial Intelligence in Education”. In : *Computers and Education* (2019).
- [3] Tom B. BROWN et al. “Language Models are Few-Shot Learners”. In : *Advances in Neural Information Processing Systems* (2020).
- [4] Ramesh NALLAPATI et al. “Abstractive Text Summarization using Sequence-to-Sequence RNNs”. In : *arXiv preprint arXiv :1602.06023* (2016).
- [5] Ashish VASWANI et al. “Attention is All You Need”. In : *Advances in Neural Information Processing Systems* (2017).
- [6] Ghadah KURDI et al. “A Systematic Review of Automatic Question Generation for Educational Purposes”. In : *International Journal of Artificial Intelligence in Education* (2020).
- [7] Rishi BOMMASANI et al. “On the Opportunities and Risks of Foundation Models”. In : *arXiv preprint arXiv :2108.07258* (2021).
- [8] Enkelejda KASNECI et al. “ChatGPT for Good ? On Opportunities and Challenges of Large Language Models for Education”. In : *Learning and Individual Differences* (2023).