ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

Ecole Marocaine des Sciences de l'Ingénieur de Tanger

RAPPORT DE PROJET FIN D'ANNEE

Filière
Ingénierie Informatique et Réseaux

StudySmart : Espace Intelligent pour la Gestion de Cours, Notes, Examens et Aide IA

Réalisé par
Abdelmajid AFAKI
Aya EL BAYADE
Amine EL OUAFI
Marwan OUAFI

Encadré par
M. Ahmed RABHI
M. Mouad MANSOURI

Membres de jury
M. Ahmed RABHI
M. Mouad MANSOURI
M. Houssam BAZZA

4IIR

2er Année Cycle d'Ingénieur

Juin 2025
Année Universitaire 2024-2025

Remerciements

*Nous tenons à exprimer notre sincère gratitude à **Monsieur Ahmed RABHI**, notre encadrant, pour son accompagnement constant, ses conseils avisés et sa disponibilité tout au long de ce projet. Son expertise technique et sa rigueur pédagogique ont été d'un grand apport à la réussite de notre travail.*

*Nous adressons également nos vifs remerciements à **Monsieur Mouad MANSOURI**, pour son encadrement attentif, sa bienveillance et ses orientations précieuses qui nous ont guidés à chaque étape du projet.*

*Nos remerciements vont aussi à **l'ensemble de l'administration de l'EMSI**, pour les moyens mis à notre disposition et pour le cadre pédagogique favorable à l'épanouissement de nos compétences.*

*Enfin, nous remercions chaleureusement **tous les membres de notre équipe** pour leur engagement, leur collaboration et leur esprit d'initiative. C'est grâce à la contribution de chacun que ce projet a pu être mené à bien dans les meilleures conditions.*

Résumé

Ce rapport présente le développement de StudySmart, une application web éducative conçue spécialement pour les étudiants de l'EMSI. L'application regroupe, dans une seule interface moderne, les outils essentiels à la gestion de la vie académique : cours, notes, fichiers, examens, tâches et calendrier.

Un module d'intelligence artificielle intégré permet aux étudiants de poser des questions, d'obtenir des explications personnalisées à partir de fichiers, ou encore de générer des quiz pour réviser. Cette assistance intelligente favorise une meilleure compréhension et un apprentissage autonome.

En complément, l'application propose un système d'entraide étudiante : les utilisateurs peuvent publier ou répondre à des demandes d'aide, échanger entre eux, et ainsi construire une dynamique collaborative au sein de la communauté EMSI.

StudySmart offre une solution complète de gestion académique, centralisant toutes les fonctionnalités utiles dans un tableau de bord simple, fluide et accessible. Elle améliore l'organisation quotidienne des étudiants tout en intégrant des technologies modernes garantissant sécurité, performance et évolutivité.

Mots-clés : Application web, IA, Gestion de cours, Planification.

Abstract

This report presents the development of StudySmart, an educational web application specifically designed for EMSI students. The application brings together, in a single modern interface, all the essential tools for managing academic life: courses, notes, files, exams, tasks, and calendar.

An integrated artificial intelligence module allows students to ask questions, receive personalized explanations from files, and generate quizzes for self-study. This intelligent support enhances understanding and promotes independent learning.

Additionally, the application includes a student support system where users can publish or respond to help requests, exchange messages, and build a collaborative dynamic within the EMSI student community.

StudySmart provides a complete academic management solution, centralizing all useful features into a clear, fluid, and accessible dashboard. It improves students' daily organization while integrating modern technologies to ensure security, performance, and scalability.

Keywords: Web application, AI, Course management, Planning.

Table des matières

Liste des figures

Liste des tableaux

Introduction Générale

Dans le cadre de notre quatrième année d'études en informatique à l'École Marocaine des Sciences de l'Ingénieur (EMSI), notre équipe a eu l'opportunité de réaliser un projet innovant répondant à une problématique concrète rencontrée par les étudiants dans leur quotidien académique. Ce projet, mené sur plusieurs mois, nous a permis de mobiliser nos compétences techniques tout en expérimentant les différentes étapes du cycle de développement logiciel, depuis l'analyse des besoins jusqu'aux tests, dans un contexte réaliste et professionnalisant.

Face à la multiplication des ressources pédagogiques, cours, fichiers, notes, révisions, tâches à gérer et échanges entre étudiants, nous avons constaté l'absence d'une solution unique et intelligente capable de centraliser et d'organiser efficacement ces éléments. C'est ainsi qu'est née l'idée de créer **StudySmart**, une application web éducative, intuitive et complète, conçue spécifiquement pour améliorer la gestion de la vie académique des étudiants de l'EMSI. Elle vise à renforcer la productivité, favoriser la planification, simplifier la collaboration entre pairs, et optimiser le suivi pédagogique.

StudySmart propose un ensemble de fonctionnalités intégrées dans un espace sécurisé : authentification, gestion des cours et des fichiers, prise de notes, planification des tâches et des examens, génération automatique de questions via l'IA, demande d'aide entre étudiants, entre autres. Pour concrétiser cette vision, nous avons adopté un stack technologique moderne et performante : **Next.js** pour l'interface utilisateur, **FastAPI** pour le backend, **Supabase** pour la base de données et l'authentification, et **Groq** pour les services d'intelligence artificielle. Ce choix technologique nous a permis de garantir une application rapide, fiable, évolutive et facile à maintenir.

Ce rapport détaille l'ensemble des étapes du projet à travers quatre chapitres principaux. Le **chapitre 1** présente une étude bibliographique incluant une analyse des solutions existantes, les apports spécifiques de StudySmart et une comparaison fonctionnelle. Le **chapitre 2** est consacré au cahier des charges, définissant le contexte, les objectifs et les spécifications du projet. Le **chapitre 3** aborde la conception du système à travers des diagrammes UML, l'architecture générale et le modèle de base de données. Enfin, le **chapitre 4** décrit en profondeur le développement de l'application, les technologies utilisées, les fonctionnalités implémentées, les tests effectués ainsi que les résultats obtenus.

.

Chapitre 1 : Étude bibliographique

1

Étude bibliographique

Résumé : Les outils actuels pour gérer les études sont nombreux mais souvent fragmentés et peu adaptés aux besoins spécifiques des étudiants. L'intelligence artificielle améliore l'apprentissage, mais sans être intégrée à la gestion académique. StudySmart propose une application unique qui combine gestion, IA et entraide. Cette solution facilite l'organisation et la collaboration des étudiants de l'EMSI.

1. Analyse des solutions existantes

1.1. Applications de gestion académique

Plusieurs applications visent à accompagner les étudiants dans la gestion de leurs cours et devoirs. Parmi les plus connues, on peut citer :

a. Google Drive

- Plateforme de stockage en ligne proposée par Google.
- Permet de stocker, organiser, partager des fichiers et collaborer en ligne.
- Intégration avec Google Docs, Sheets, Slides.
- Utilisation simple, mais peu personnalisable pour un contexte scolaire structuré [1].

b. Microsoft OneDrive

- Service de stockage de Microsoft, avec synchronisation automatique.
- Intégration étroite avec la suite Office.
- Fonctionne bien pour le travail collaboratif, mais peu adapté à un usage académique spécifique [2].

c. Moodle

- Plateforme de gestion de l'apprentissage (LMS) utilisée par de nombreuses institutions.
- Supporte les cours en ligne, le dépôt de devoirs, et la gestion des fichiers.
- Complexité de navigation, particulièrement pour des tâches simples comme consulter ou déposer un fichier [3].

d. WeTransfer

- Plateforme spécialisée dans l'envoi rapide de fichiers volumineux.
- Pas de gestion de stockage ou d'organisation, ni de comptes utilisateurs [4].

- Très limité dans un contexte scolaire ou universitaire.

1.2. Intelligence artificielle dans l'éducation

L'intégration de l'**IA** dans le domaine éducatif est en pleine expansion. Plusieurs services utilisent l'IA pour améliorer l'apprentissage :

- **ChatGPT** (OpenAI) ou **Bard** (Google) permettent de poser des questions complexes, mais n'offrent pas de contextualisation académique liée aux documents personnels ou aux cours.
- **Quizlet** intègre des suggestions automatiques de questions, mais reste limité à des formats d'exercices simples.
- **Khan Academy** expérimente des tuteurs IA, mais ne propose pas une gestion complète de l'organisation académique.

2. Apports de StudySmart

StudySmart regroupe dans une seule application tous les outils utiles aux étudiants de l'EMSI: gestion des cours, des fichiers, des notes, des examens, aide avec une IA, et entraide entre camarades. Cela permet aux étudiants de tout gérer facilement depuis un seul espace.

L'application se distingue par sa simplicité, son tableau de bord clair, et l'intégration de l'intelligence artificielle. Elle ne se contente pas de rassembler des outils, elle améliore réellement l'organisation quotidienne et encourage la collaboration entre étudiants. C'est une nouvelle façon, plus pratique et plus intelligente, de vivre ses études.

3. Comparaison fonctionnelle

Fonctionnalité	Google Drive	Notion	Moodle	WeTransfer	SmartStudy
Stockage de fichiers	Oui	Oui	Oui	Non	Oui
Téléversement par étudiants	Oui	Oui	Oui	Oui	Oui
Interface simple et intuitive	Oui	Oui	Non	Oui	Oui
Structuration par modules/semestres	Non	Non	Oui	Non	Oui
Spécialisation éducation	Non	Non	Oui	Non	Oui
Intégration de l'IA	Oui	Oui	Oui	Non	Oui
Planification et gestion des emplois du temps	Non	Non	Non	Non	Oui

Tableau 1 : Comparaison fonctionnelle

4. Conclusion

L'étude bibliographique et l'analyse des solutions existantes a mis en évidence un manque de simplicité ou d'adaptation aux besoins spécifiques des étudiants dans les outils actuellement disponibles. SmartStudy a été conçu pour offrir une alternative ciblée : simple, fonctionnelle et parfaitement intégrée à l'environnement universitaire. Ce positionnement constitue la force principale de notre solution.

Chapitre 2 : Cahier des Charges du Projet

2

Cahier des Charges

Résumé : Ce cahier des charges présente les bases du développement d'une application éducative destinée aux étudiants de l'EMSI. Il décrit les objectifs, les modules fonctionnels (cours, notes, planning, IA, entraide...), les contraintes techniques et la planification du projet. L'application vise à centraliser la gestion des études dans une interface moderne.

1. Introduction

Dans un environnement académique en constante évolution, les étudiants de l'EMSI doivent faire face à de nombreuses tâches : suivre plusieurs cours, prendre des notes, organiser des fichiers, préparer des examens, et parfois solliciter l'aide de leurs camarades. Pour répondre à ces besoins croissants, ce projet propose le développement d'une application web intelligente, collaborative et centralisée, conçue spécifiquement pour améliorer l'organisation, la productivité et la réussite des étudiants de notre école à travers une interface intuitive et moderne.

2. Contexte et Problématique

Les étudiants utilisent aujourd'hui une multitude d'outils fragmentés pour gérer leurs études : applications de prise de notes, plateformes de stockage de fichiers, calendriers, forums d'entraide, etc. Ce manque d'unification entraîne une perte de temps, de concentration et d'efficacité. De plus, peu d'outils éducatifs intègrent des fonctionnalités d'intelligence artificielle ou de collaboration directe entre étudiants.

3. Objectifs du Projet

L'objectif principal du projet est de développer une application éducative intelligente qui aide les étudiants à :

- Gérer leurs cours, notes et fichiers de manière centralisée.
- Planifier efficacement leurs examens et révisions.
- Poser des questions à une IA pour mieux comprendre leurs cours.
- Coopérer avec leurs pairs via un système d'entraide.
- Accéder à un tableau de bord clair de leur activité.

4. Spécifications Fonctionnelles

Authentification : Ce module permet aux étudiants de créer un compte personnel sécurisé avec une adresse email et un mot de passe afin d'accéder à l'application. Une fois connectés, ils peuvent gérer leur profil en modifiant à tout moment leurs informations personnelles telles que le nom, l'email ou le mot de passe, dans un environnement protégé.

Gestion des cours : Ce module permet aux étudiants d'ajouter, modifier ou supprimer leurs cours selon les matières ou le semestre. Ils peuvent y associer des fichiers (PDF, images, etc.) pour organiser leurs supports, et utiliser un moteur de recherche ou des filtres pour retrouver un cours rapidement.

Gestion des notes : Les étudiants peuvent créer et modifier des notes liées à chaque cours, ce qui facilite l'organisation de leurs révisions. Grâce à un système de recherche par mots-clés, ils peuvent retrouver facilement une note précise parmi celles enregistrées.

Planification d'études : Ce module permet aux étudiants d'ajouter des examens ou des tâches dans un calendrier intégré et d'organiser leur charge de travail. En fonction des échéances, l'application génère automatiquement un plan de révision personnalisé pour optimiser leur temps d'étude.

Assistant IA : L'intelligence artificielle intégrée aide les étudiants à mieux comprendre leurs cours en répondant à leurs questions. Elle peut également analyser un fichier pour en fournir une explication ou un résumé, et générer automatiquement des questions d'entraînement à partir du contenu des cours.

Système d'entraide : Ce module favorise la collaboration entre étudiants. Chacun peut publier une demande d'aide sur un sujet spécifique et consulter celles des autres. Il est aussi possible de discuter directement avec un autre étudiant pour proposer ou recevoir de l'aide.

Tableau de bord : Le tableau de bord centralise toutes les informations importantes : il offre une vue d'ensemble des cours, notes, fichiers, examens et tâches à venir, afin de permettre à l'étudiant de suivre facilement sa progression et son organisation.

Gestion des fichiers : Les étudiants peuvent importer, visualiser et classer leurs fichiers selon les cours. L'interface permet également de partager facilement un document avec un autre utilisateur de l'application, rendant la gestion documentaire simple et collaborative.

5. Spécifications Non Fonctionnelles

Sécurité : L'application garantit une authentification sécurisée et un stockage protégé des données personnelles. Les informations des utilisateurs sont traitées de manière confidentielle, afin d'assurer la protection de leur vie privée.

Performance : L'interface est optimisée pour offrir des temps de chargement rapides. Les actions principales (connexion, consultation de cours, enregistrement de notes...) doivent s'exécuter en moins d'une seconde afin d'assurer une expérience fluide.

Accessibilité : L'application est conçue pour s'adapter à tous les types d'écrans, que ce soit sur ordinateur, tablette ou smartphone. L'interface est intuitive et facile à utiliser, même pour les utilisateurs peu expérimentés.

Fiabilité : Les données saisies par les utilisateurs sont sauvegardées automatiquement dans la base de données Supabase. Cette sauvegarde régulière garantit que les informations ne sont jamais perdues, même en cas de panne ou de déconnexion.
Scalabilité : L'architecture du projet est conçue pour supporter une augmentation du nombre d'utilisateurs sans compromettre les performances. Ainsi, l'application peut évoluer dans le temps selon les besoins de la communauté étudiante.
Intégration IA : L'intégration de l'intelligence artificielle Groq se fait via une API efficace et rapide, permettant de fournir des réponses pertinentes en temps réel aux questions posées par les étudiants, ainsi que des fonctionnalités avancées comme la génération de résumés ou de quiz.

6. Diagramme de Gantt

Le diagramme de Gantt ci-dessous présente la planification complète du projet, répartie sur plusieurs semaines, de mars à juillet 2025. Chaque tâche est associée à une date de début, une durée estimée, et un membre responsable.

Figure 1 : Diagramme de GANTT

ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

PFA | 4IIR

N Groupe

Légende :

Membre de groupe

Date de début du projet : 22/03/2025

Incrément de défilement : 1

Description du jalon	Catégorie	Attribué à	Progression	Début	Jours
Recherche d'idees de sujet	En bonne voie	Aya	100%	22/03/2025	2
Analyse de faisabilite	En bonne voie	Afaki	100%	24/03/2025	2
Conception de l'architecture generale	Risque faible	Marwan	100%	26/03/2025	2
Répartition des taches	Risque faible	Amin	100%	28/03/2025	1
Authentification par e-mail	Risque moyen	Afaki	100%	29/03/2025	3
Authentification via Google	Risque moyen	Marwan	100%	01/04/2025	3
Gestion de session / cookies	Risque moyen	Aya	100%	04/04/2025	2
Creation du profil utilisateur	En bonne voie	Aya	100%	06/04/2025	2
Creation de dossiers par cours	En bonne voie	Amin	100%	12/04/2025	3
Interface d'upload de fichiers	En bonne voie	Marwan	100%	15/04/2025	4
Prise en charge des extensions	En bonne voie	Afaki	100%	19/04/2025	2
UI de visualisation des fichiers	En bonne voie	Aya	100%	21/04/2025	3
Telechargement / suppression des fichiers	En bonne voie	Amin	100%	24/04/2025	2
Categorisation manuelle	En bonne voie	Afaki	100%	26/04/2025	3
Categorisation par IA	Risque élevé	Marwan	100%	29/04/2025	4
Interface edition de fichiers	Risque élevé	Aya	100%	03/05/2025	3
Amelioration UI/UX fichiers	Risque élevé	Amin	100%	06/05/2025	2
UI du planning	En bonne voie	Aya	100%	10/05/2025	3
Creation plan par horaire dispo	En bonne voie	Afaki	100%	13/05/2025	2
Saisie des examens	Risque élevé	Amin	100%	15/05/2025	2
Ajout de notes aux fichiers	En bonne voie	Marwan	100%	17/05/2025	3

Figure 2 : Tableau de GANTT (1)

31	UI pour visualiser notes	Risque moyen	Aya	100%	20/05/2025	2
32	Enregistrement et liaison aux fichiers	Risque moyen	Afaki	100%	22/05/2025	2
33	Systeme de recherche dans les notes	Risque moyen	Amin	100%	24/05/2025	4
34	Affichage des resultats filtres	En bonne voie	Marwan	100%	28/05/2025	3
35	Creation de l'interface de profil	En bonne voie	Aya	100%	31/05/2025	2
36	Chargement & affichage des infos utilisateur	En bonne voie	Afaki	100%	02/06/2025	1
37	Formulaire de modification + validation	En bonne voie	Amin	100%	03/06/2025	2
38	Amelioration ergonomie et responsive	En bonne voie	Marwan	100%	05/06/2025	1
39	Sécurité des mises à jour	Risque moyen	Afaki	100%	06/06/2025	1
40	Notification rappels d'examen	Risque moyen	Afaki	100%	07/06/2025	4
41	Alertes sur nouveaux fichiers	Risque moyen	Aya	100%	11/06/2025	3
42	Generation QCM a partir des fichiers	En bonne voie	Marwan	100%	14/06/2025	3
43	Interface d'examen pour etudiants	En bonne voie	Aya	100%	17/06/2025	2
44	Correction auto + feedback IA	En bonne voie	Amin	100%	19/06/2025	2
45	Tests fonctionnels	En bonne voie	Afaki	100%	21/06/2025	2
46	Optimisation UI et perfs	En bonne voie	Marwan	100%	23/06/2025	3
47	Corrections des bugs	En bonne voie	Aya	100%	26/06/2025	2
48	Redaction technique	En bonne voie	Afaki	100%	28/06/2025	2
49	Description fonctionnelle	En bonne voie	Aya	100%	30/06/2025	1
50	Conclusion & annexes	En bonne voie	Marwan	100%	01/07/2025	1
51	Démo	En bonne voie	Amin	100%	04/07/2025	1
52	Repetition + revision	En bonne voie	Tous	100%	05/07/2025	1

Figure 3 : Tableau de GANTT (2)

7. Conclusion

Ce cahier des charges a permis de définir de manière structurée les fondations du projet de développement d'une application éducative intelligente, pensée pour répondre aux besoins réels des étudiants de l'EMSI. À travers une organisation claire en modules fonctionnels, une planification rigoureuse, et des objectifs bien ciblés, le projet vise à centraliser et simplifier la gestion des études, tout en intégrant des outils modernes comme l'intelligence artificielle et la collaboration entre étudiants.

La démarche adoptée garantit une solution cohérente, évolutive et accessible, avec une expérience utilisateur fluide. Grâce à cette base, l'équipe peut désormais passer à la phase de développement en s'appuyant sur une vision claire, des priorités définies, et une répartition équilibrée des responsabilités. Ce projet représente une réelle valeur ajoutée pour les étudiants de l'EMSI, en leur offrant un outil numérique complet et adapté à leur quotidien académique.

Chapitre 3 : Conception du système

3

Conception du système

Résumé : Ce chapitre présente la conception technique de l'application, en détaillant les diagrammes de cas d'utilisation, de séquence et de classes, qui illustrent les interactions entre les utilisateurs et le système. Il décrit également l'architecture générale de l'application, structurée en frontend, backend, base de données et intelligence artificielle. Enfin, il expose le modèle logique et physique de la base de données, avec les entités principales et leurs relations, implémentées via Supabase.

1. Digramme de cas d'utilisation

Le diagramme ci-dessous représente les principales interactions entre les utilisateurs de l'application StudySmart et les fonctionnalités offertes. Ce diagramme est centré sur l'acteur principal **« Étudiant »**, qui est le principal utilisateur du système, et l'acteur secondaire **« IA »**, représentant les services d'intelligence artificielle intégrés à l'application.

Les acteurs :

- **Étudiant** : peut accéder à toutes les fonctionnalités proposées par l'application, notamment la gestion des cours, des fichiers, des examens, des tâches, des notes et des demandes d'aide.
- **IA** : intervient dans certaines fonctionnalités spécifiques comme la génération de plans de révision ou la consultation intelligente de fichiers.

Cas d'utilisation principaux :

- **Authentifier** : permet à l'étudiant de se connecter à la plateforme de manière sécurisée.
- **Planification d'étude** : aide l'étudiant à organiser ses sessions d'étude.
- **Gérer les tâches**, **gérer les cours**, **gérer les notes** : fonctionnalités classiques de suivi académique.
- **Créer un cours** : fonctionnalité permettant à l'étudiant d'ajouter ses propres cours. Elle **inclut** implicitement « Gérer les cours » et dépend de « Gérer les fichiers ».
- **Gérer les fichiers** : permet d'ajouter, modifier ou supprimer les fichiers liés aux cours.
- **Consulter un fichier avec l'IA** : fonctionnalité qui permet d'analyser un fichier grâce à l'IA pour en extraire des informations ou poser des questions. Elle **dépend** de la gestion des fichiers et implique également l'acteur IA.
- **Demandes d'aide** : permet aux étudiants de demander de l'aide à d'autres utilisateurs.

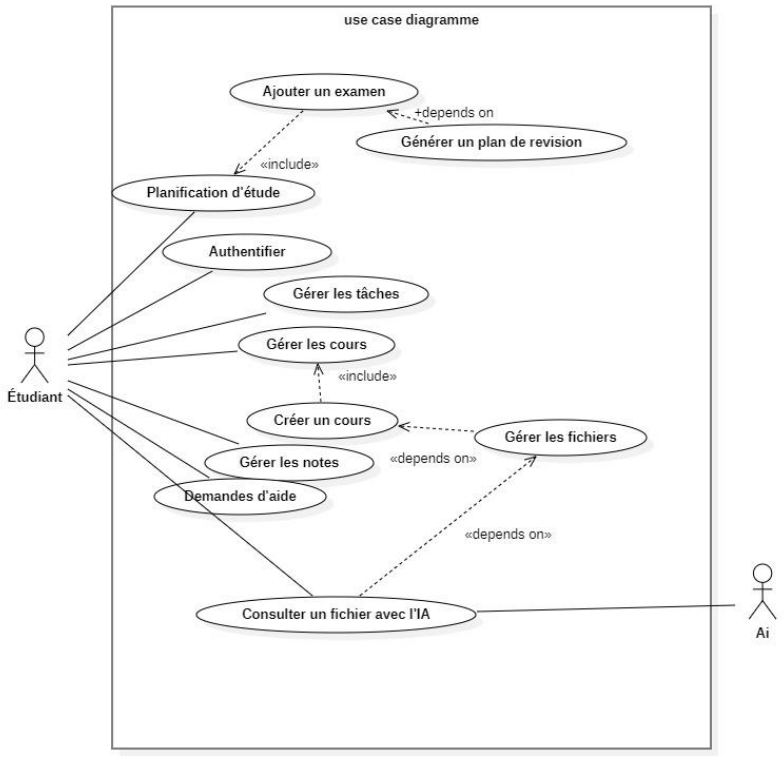

Figure 4 : Diagramme de cas d'utilisation

2. Diagramme de classe

Ce diagramme de classes UML illustre l'architecture principale d'un système, probablement lié à une plateforme éducative ou de gestion de contenu académique. Il met en évidence les entités clés, leurs responsabilités et comment elles interagissent.

Classes Principales et Leurs Rôles :

- **Utilisateur** : Représente toute personne utilisant le système, avec des fonctions de connexion et d'authentification de base.
- **Profil** : Une spécialisation de l'utilisateur, contenant des informations détaillées comme le nom complet, le niveau académique et la spécialisation. Un utilisateur possède un profil.
- **Note** : Permet de gérer des annotations ou des évaluations, pouvant être associées à un utilisateur ou à un profil.
- **AnnonceAide** : Gère la création et la gestion d'annonces pour offrir ou demander de l'aide.
- **Cours** : Représente des modules d'enseignement avec leur titre et description.
- **Examen** : Permet de définir et de gérer les examens, incluant leur date et priorité.
- **Fichier** : Gère les documents et ressources téléchargeables, comme les supports de cours.
- **ServiceAI** : Intègre des fonctionnalités d'intelligence artificielle pour traiter des messages ou analyser des fichiers.
- **PlanEtude** : Représente un plan d'étude, qui peut être généré.

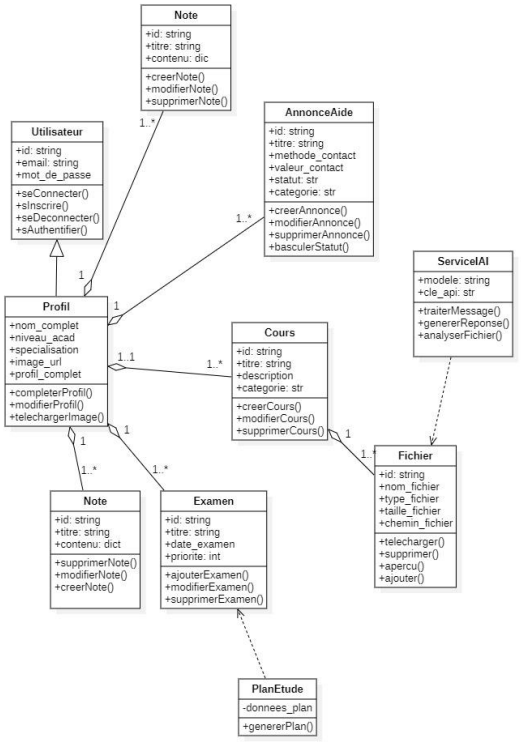

Figure 5 : Diagramme de classe

3. Architecture générale de l'application

L'architecture choisie pour notre application repose sur un modèle client-serveur moderne, structuré en plusieurs couches bien distinctes. Ce type d'architecture est largement utilisé dans le développement web, car il permet de séparer clairement les responsabilités de chaque composant, d'améliorer la lisibilité du code, de faciliter la maintenance et de rendre l'ensemble du système plus évolutif et sécurisé.

La première couche est le frontend, c'est-à-dire l'interface utilisateur, qui a été développée à l'aide du framework Next.js. Ce framework basé sur React permet de construire des interfaces web interactives et performantes. Il offre plusieurs avantages importants : rendu côté serveur (Server-Side Rendering), génération statique de pages, optimisation automatique des performances et support natif du routage. Grâce à Next.js, nous avons pu concevoir une interface fluide, responsive (adaptée à tous les types d'écrans) et facile à utiliser. L'étudiant peut ainsi interagir avec l'application de manière intuitive : consulter ses cours, gérer ses fichiers, poser des questions à l'IA ou encore accéder à son tableau de bord.

La deuxième couche est le backend, qui a été développé avec FastAPI, un framework Python moderne, rapide et léger. FastAPI nous permet de créer des API REST sécurisées et performantes. Cette couche gère toute la logique métier de l'application : traitement des données, vérification des permissions, gestion des sessions utilisateur, envoi des requêtes vers la base de données, et intégration avec les services externes comme l'intelligence artificielle. FastAPI facilite également la documentation automatique des routes API, ce qui a rendu le travail en équipe plus fluide et organisé.

La troisième couche est celle de la base de données, qui est gérée à l'aide de Supabase, une alternative open source à Firebase, reposant sur le moteur PostgreSQL. Supabase permet de stocker toutes les informations liées aux utilisateurs, aux cours, aux notes, aux fichiers et aux

demandes d'aide. Il propose un système d'authentification intégré, la gestion des rôles et des permissions, un stockage de fichiers, ainsi qu'une API générée automatiquement à partir des tables créées. Supabase propose également la synchronisation en temps réel, ce qui permet d'actualiser certaines données instantanément dans l'interface, sans devoir recharger la page.
Enfin, une quatrième couche vient enrichir notre architecture : celle de l'intelligence artificielle, intégrée grâce à l'API Groq. Ce module IA permet aux étudiants de poser des questions, d'obtenir des explications personnalisées sur des fichiers ou des sujets d'étude, ou encore de générer automatiquement des exercices d'entraînement. Cette couche donne à notre application une dimension intelligente et interactive, en offrant un assistant virtuel qui soutient les étudiants dans leur apprentissage quotidien. L'intégration de l'IA se fait via des requêtes API sécurisées depuis le backend, avec des réponses structurées envoyées à l'interface utilisateur.

Cette architecture globale, répartie sur quatre couches principales, permet de séparer efficacement les fonctions de présentation, de logique métier, de gestion des données et d'intelligence. Elle offre une meilleure maintenabilité, une évolutivité simplifiée en cas de mise à jour ou d'ajout de fonctionnalités, ainsi qu'un haut niveau de sécurité et de performance. Grâce à ce choix d'architecture, notre application peut évoluer avec les besoins des utilisateurs tout en gardant une structure solide et claire.

4. Modèle logique et physique de la base de données

4.1. Modèle logique

Le modèle logique constitue une étape essentielle dans la conception d'une base de données relationnelle. Il permet de représenter, de manière abstraite et indépendante des considérations techniques, la structure des informations à gérer dans l'application. Ce modèle est centré sur l'identification des entités clés, de leurs attributs et des relations logiques entre elles. Dans notre cas, nous avons défini plusieurs entités fondamentales qui reflètent les fonctionnalités principales de l'application : Utilisateur, Cours, Note, Fichier, Demande aide et Réponse IA. Chaque entité correspond à une table de la base de données, avec des attributs adaptés à son rôle. Par exemple, l'entité Utilisateur regroupe toutes les informations liées à un étudiant ou un membre inscrit sur la plateforme, comme son nom, prénom, adresse e-mail et mot de passe. L'entité Cours permet de stocker les cours créés, tandis que Note et Fichier sont respectivement utilisés pour enregistrer les notes personnelles de l'étudiant et les fichiers associés aux cours. Demande aide modélise les questions posées dans le cadre du système d'entraide entre utilisateurs, tandis que Réponse IA (facultative) conserve l'historique des échanges avec l'intelligence artificielle. Les relations entre ces entités sont définies à l'aide de clés primaires (souvent un identifiant unique) et de clés étrangères pour établir des connexions solides entre les différentes tables. Ce modèle logique constitue donc le socle sur lequel repose toute la structure de données de l'application.

4.2.Modèle physique

Le modèle physique est la concrétisation technique du modèle logique dans un système de gestion de base de données. Il vise à définir, de manière détaillée, comment chaque table, champ et relation va être implémenté concrètement. Cela comprend notamment le choix des types de données (texte, entier, booléen, date…), la définition des clés primaires et étrangères, la mise en place des contraintes d'intégrité (comme NOT NULL, UNIQUE, ou encore DEFAULT) et l'optimisation de l'accès aux données à travers la création d'index.

Dans notre projet, chaque entité définie dans le modèle logique a été traduite en table PostgreSQL avec des champs typés et des relations strictement encadrées. Par exemple, la table utilisateur possède une clé primaire auto-incrémentée et des champs avec des contraintes sur le format de l'adresse e-mail ou la sécurité du mot de passe. Les tables comme cours, note ou fichier sont liées entre elles via des clés étrangères, garantissant ainsi une cohérence logique entre les données. Le modèle physique inclut également la création d'index sur certaines colonnes fréquemment utilisées dans les requêtes, ce qui permet d'accélérer les recherches et d'améliorer les performances générales de l'application. Ce travail de structuration fine est indispensable pour assurer la fiabilité, la rapidité et la sécurité de la base de données

4.3.Implémentation dans Supabase

Pour la mise en œuvre de notre modèle physique, nous avons opté pour **Supabase**, une plateforme open source basée sur PostgreSQL, qui offre à la fois une interface simple d'utilisation et une grande puissance fonctionnelle. Supabase permet de créer facilement des tables, de gérer les relations entre elles, et d'ajouter des règles de sécurité adaptées aux besoins de l'application. Ce choix s'est imposé naturellement car Supabase propose des fonctionnalités intégrées comme l'authentification sécurisée, le stockage de fichiers, la génération automatique d'API REST, et même la synchronisation des données en temps réel.

Dans notre projet, Supabase nous a permis de créer rapidement l'ensemble des tables définies dans notre modèle logique, de tester les requêtes SQL (insertion, mise à jour, suppression), et d'interagir avec la base à travers une API générée automatiquement. L'un des avantages majeurs de Supabase est son système de sécurité basé sur des **politiques RLS (Row Level Security)**, qui permet de définir des règles précises sur qui peut lire, écrire ou modifier chaque donnée selon le contexte de l'utilisateur. Nous avons également profité du module de stockage pour héberger les fichiers pédagogiques (PDF, images) et les lier facilement aux cours concernés. Enfin, la compatibilité de Supabase avec des frameworks modernes comme FastAPI et Next.js a grandement facilité l'intégration avec le reste de notre application. En résumé, Supabase s'est révélé être un outil complet, fiable et performant pour gérer l'ensemble du back-end de notre base de données.

Chapitre 4 : Développement de l'application

4

Développement de l'application

Résumé : Ce chapitre présente l'implémentation technique de la plateforme. Il regroupe les technologies utilisées, la démonstration des principales fonctionnalités de l'application, l'explication de certains extraits de code importants, ainsi que les tests unitaires réalisés pour garantir la fiabilité du système.

1. Introduction

Le développement de l'application de gestion de fichiers pour étudiants représente une étape cruciale dans la concrétisation de notre vision éducative. Cette phase technique combine des technologies modernes et des pratiques de développement robustes pour créer une plateforme intuitive et performante. L'architecture choisie, basée sur Next.js pour le frontend et FastAPI pour le backend, avec Supabase comme solution backend-as-a-service, permet de développer rapidement tout en garantissant scalabilité et sécurité. L'intégration d'outils avancés comme TipTap pour l'édition de texte riche, l'assistant IA via Groq, et un système de gestion de fichiers sophistiqué, démontre notre engagement envers l'innovation technologique au service de l'éducation. Ce chapitre détaille les choix technologiques, l'architecture logicielle, les fonctionnalités clés et les stratégies de test qui sous-tendent cette plateforme éducative moderne.

2. Technologies utilisées

2.1. Architecture Générale

L'application suit une architecture moderne basée sur le **pattern client-serveur** avec une séparation claire entre le frontend et le backend :

- **Frontend** : Application web moderne développée avec Next.js 15
- **Backend** : API REST développée avec FastAPI
- **Base de données** : Supabase (PostgreSQL avec authentification intégrée)

- **Stockage** : Supabase Storage pour la gestion des fichiers

2.2. Stack Technologique Frontend

L'architecture frontend de l'application repose sur un écosystème moderne de technologies web, centré autour de Next.js et React, qui permettent de créer une interface utilisateur réactive et performante. Cette stack combine des outils de développement robustes avec des bibliothèques spécialisées pour offrir une expérience utilisateur optimale, tout en maintenant une codebase maintenable et évolutive.

2.2.1. Framework principal

Les technologies de base qui constituent le socle de l'application frontend :

- **Next.js 15.3.1 :** Framework React avec rendu côté serveur (SSR) et génération statique [5].

Figure 6 : Logo de Next.js

- **TypeScript 5 :** Typage statique pour une meilleure maintenabilité [6].

Figure 7 : Logo de TypeScript

2.2.2. Gestion d'État et Données

Les bibliothèques pour la gestion des données et l'interaction avec l'API :

- **TanStack React Query 5.74.4** : Gestion du cache et synchronisation des données
- **Axios 1.9.0** : Client HTTP pour les requêtes API [7].

Figure 8 : Logo d'Axios

2.3. Stack Technologique Backend

L'infrastructure backend de la plateforme est construite sur FastAPI, un framework Python moderne qui offre des performances exceptionnelles et une documentation automatique des APIs. Cette architecture s'appuie sur Supabase comme solution backend-as-a-service, fournissant une base de données PostgreSQL robuste et un système de stockage cloud intégré, tout en intégrant des services d'authentification sécurisés et des capacités d'intelligence artificielle pour enrichir l'expérience éducative.

2.3.1. Langage de programmation

Python est un langage de programmation interprété, reconnu pour sa simplicité. Il a été utilisé pour développer le backend, gérer la logique métier et assurer la communication avec la base de données [8].

Figure 9 : logo de python

2.3.2. Framework API

Les technologies de base qui constituent le socle de l'API backend :

- **FastAPI 0.115.12** : Framework Python moderne pour APIs REST [9].

Figure 10 : Logo de FastAPI

2.3.3. Base de Données et Stockage

L'infrastructure de données et de stockage cloud :

- **Supabase 2.15.0** : Plateforme backend-as-a-service [10].

Figure 11 : Logo de Supabase

2.3.4. Intelligence Artificielle

L'intégration de l'IA pour l'assistance éducative :

- **Groq SDK 0.22.0** : Intégration avec l'API Groq pour le chat IA [11].

Figure 12 : Logo de Groq SDK

2.3.5. Génération de Documents

Les outils pour la création de documents PDF :

- **ReportLab** : Génération de PDF pour les plans d'étude
- **FPDF** : Alternative pour la création de documents PDF

2.4. Environnement de développement

Les outils et processus qui assurent la qualité et la maintenabilité du code.

L'environnement de développement principal utilisé pour le projet :

- **Visual Studio Code 1.85+** : Éditeur de code moderne avec support natif pour TypeScript, Python et React, offrant des extensions spécialisées pour Next.js, Tailwind CSS et Supabase, ainsi que des outils de débogage intégrés et une intégration Git native pour un workflow de développement optimisé [12].

Figure 13 : Logo de Visual Studio Code

3. Démonstrations

Cette section présente les fonctionnalités principales de l'application à travers des captures d'écran et des descriptions détaillées de l'interface utilisateur. Les démonstrations illustrent l'expérience utilisateur complète, depuis l'authentification jusqu'aux fonctionnalités avancées, montrant comment la plateforme répond aux besoins spécifiques des étudiants et des établissements éducatifs.

3.1. Interface utilisateur

3.1.1. Page de connexion et d'acceuil

- **Système d'authentification** : Inscription, connexion et gestion des profils
- **Design responsive** : Adaptation automatique aux différentes tailles d'écran
- **Animations fluides** : Transitions et micro-interactions

- Les fenêtres de connexion et d'inscription

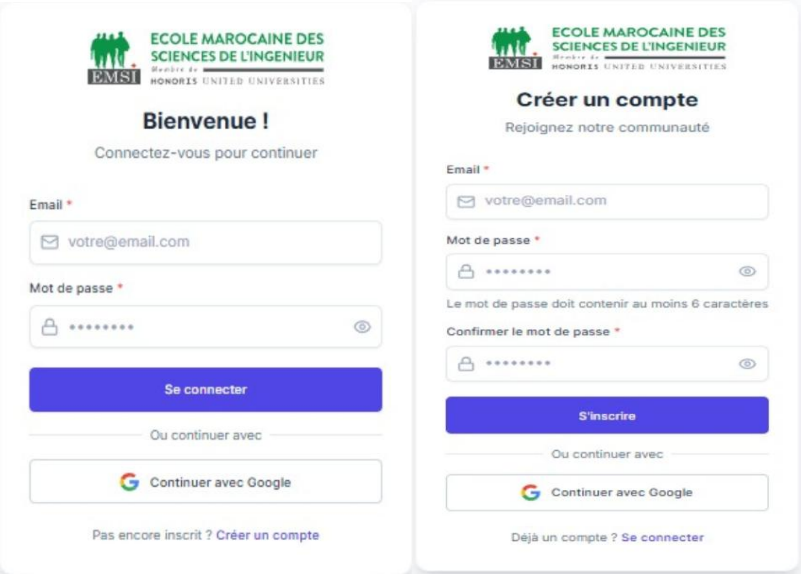

Figure 13 : Pages de connexion et de création de compte

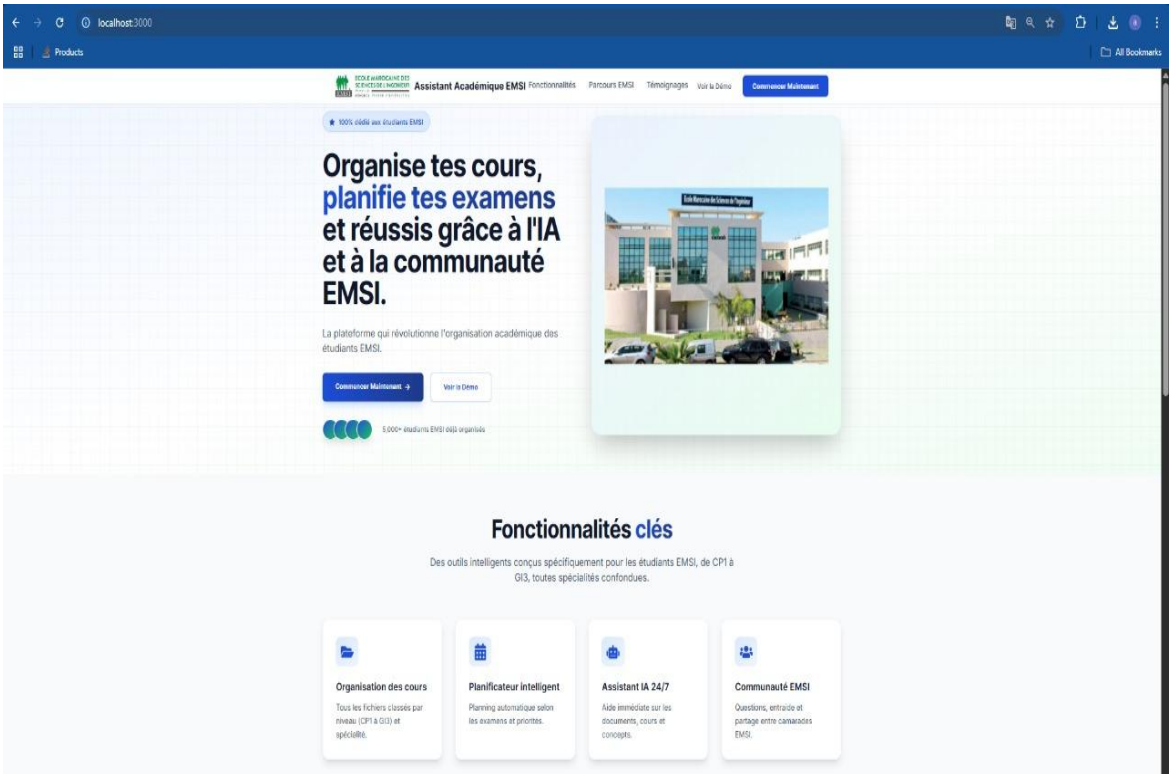

Figure 14 : Page d'Accueil

3.1.2. Tableau de bord Principal

- **Vue d'ensemble** : Statistiques des cours, fichiers et tâches
- **Navigation intuitive** : Menu latéral avec accès rapide aux fonctionnalités
- **Cartes interactives** : Affichage des informations clés

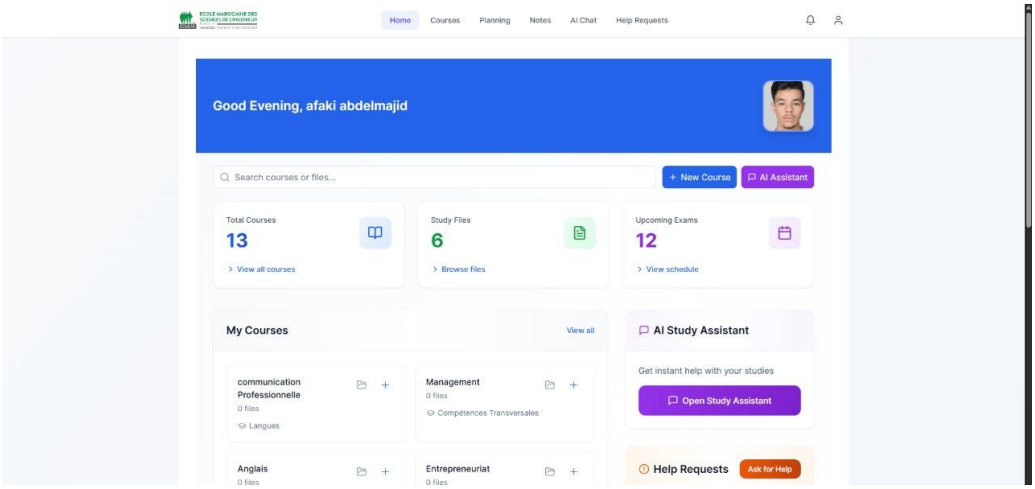

Figure 15 : Tableau de bord

3.1.3. Gestion des Cours

- **Création et édition** : Formulaires dynamiques pour la gestion des cours
- **Catégorisation** : Organisation par spécialités académiques
- **Interface drag-and-drop** : Upload de fichiers simplifié

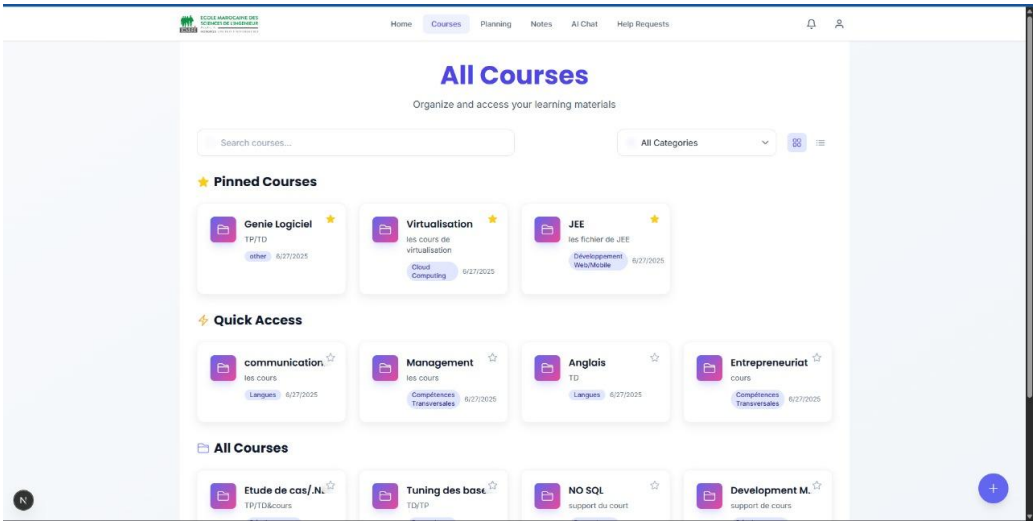

Figure 16 : La page de gestion des cours

3.1.4. Système de Fichiers

- **Organisation par catégories** : Images, PDFs, documents, présentations, archives
- **Prévisualisation** : Affichage en ligne des fichiers supportés
- **Gestion des permissions** : Accès sécurisé aux fichiers

La page de détail d'un cours, où l'étudiant peut consulter un aperçu du cours ainsi que les fichiers classés par catégorie.

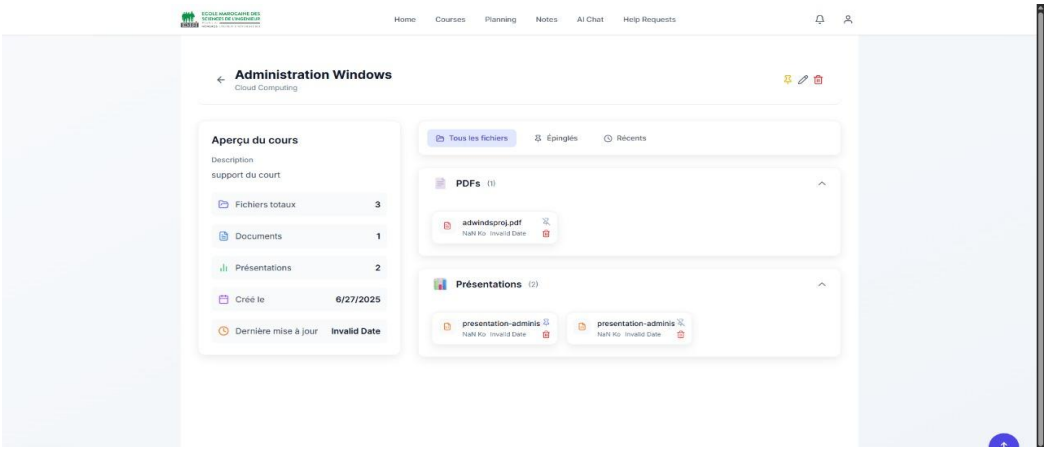

Figure 17 : Page de détail du cours – aperçu et fichiers catégorisés

La fonctionnalité de prévisualisation d'un fichier PowerPoint directement dans la plateforme, permettant à l'étudiant de consulter ses présentations sans téléchargement préalable.

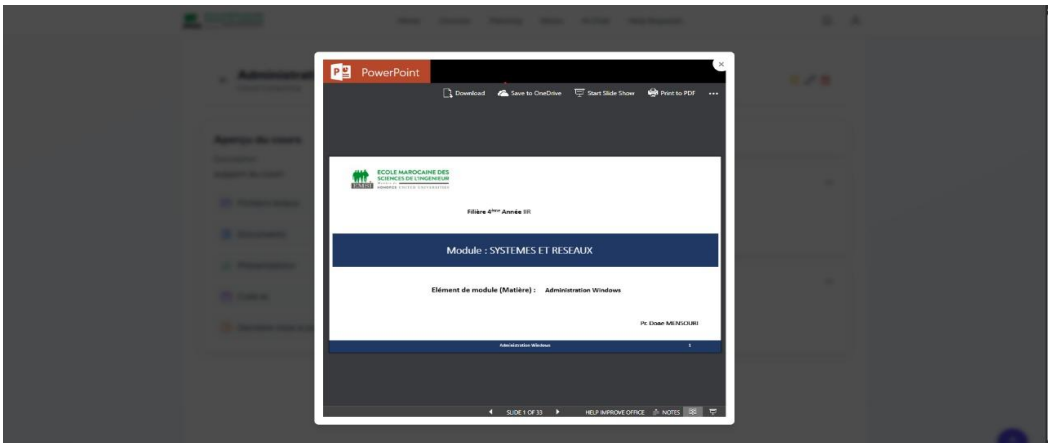

Figure 18 : Prévisualisation d'une présentation PowerPoint dans l'interface utilisateur

3.1.5. Éditeur de Notes

- **Éditeur riche TipTap** : Mise en forme avancée du texte
- **Support multimédia** : Insertion d'images et liens
- **Sauvegarde automatique** : Persistance des modifications

La page de gestion des notes, où l'utilisateur peut rechercher, filtrer et organiser ses notes de cours de manière centralisée.

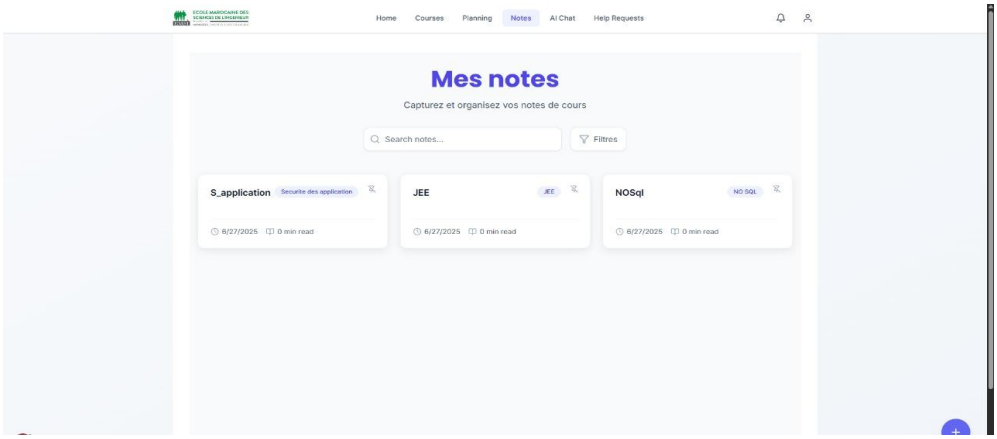

Figure 19 : Interface de gestion et d'organisation des notes de cours

La fenêtre de création d'une nouvelle note,
permettant à l'utilisateur de saisir un titre et de lier la note à un cours spécifique.

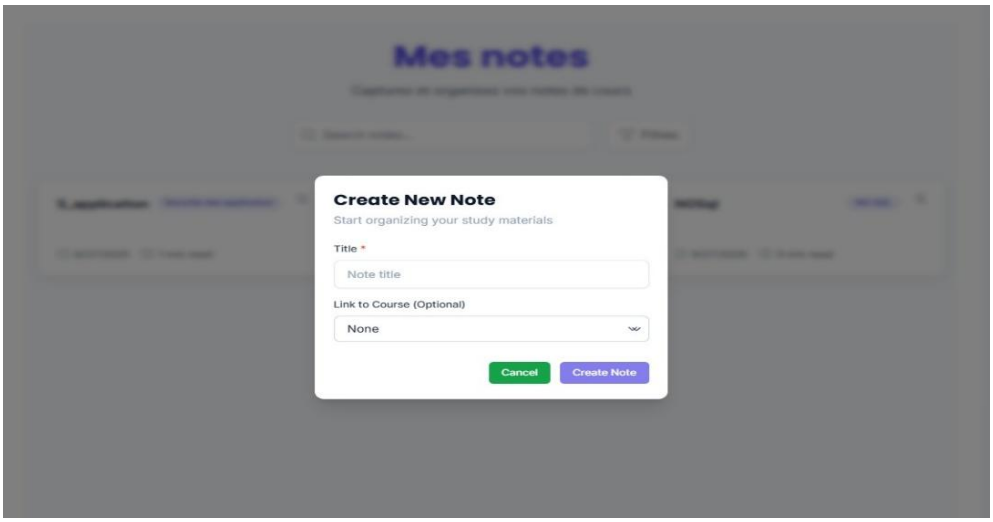

Figure 20 : Fenêtre de création d'une nouvelle note

L'éditeur de notes riche, permettant à l'utilisateur de structurer, formater et enrichir ses contenus pédagogiques avec des outils avancés.

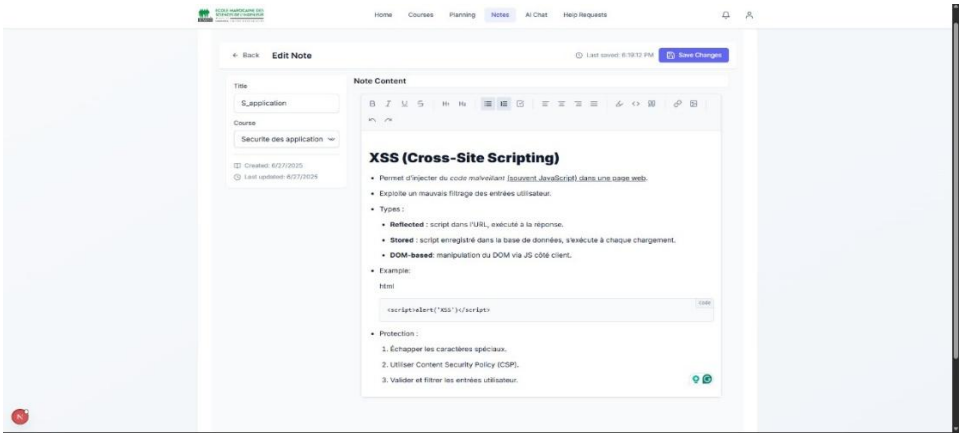

Figure 21 : Page d'éditeur de notes

3.1.6. Assistant IA

- **Chat intelligent** : Interface de conversation avec l'IA
- **Historique des conversations** : Organisation par dates
- **Réponses contextuelles** : Assistance personnalisée

L'assistant d'études IA, où l'étudiant peut poser des questions, discuter avec l'IA et obtenir des réponses instantanées pour faciliter son apprentissage.

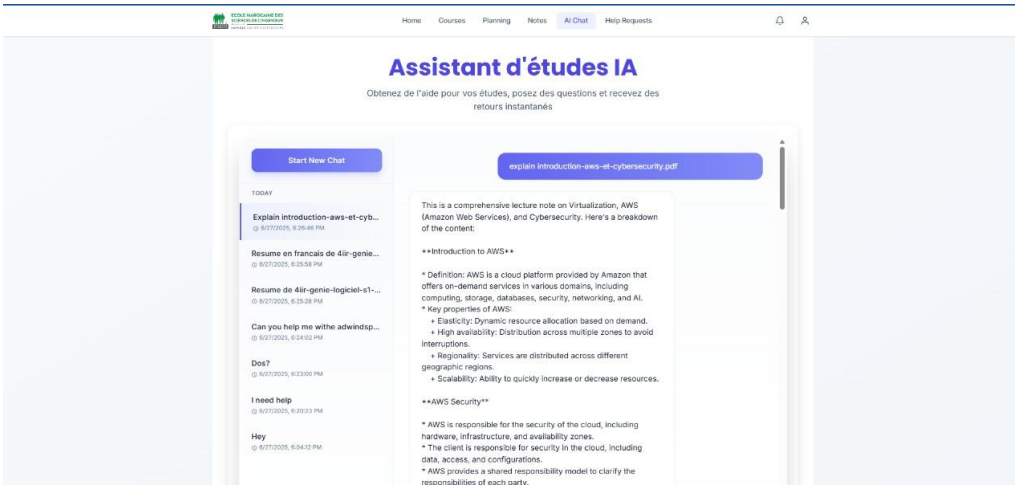

Figure 22 : Interface de l'assistant d'études IA

3.1.7. Planification

- **Calendrier interactif** : Gestion des examens et échéances
- **Génération de PDF** : Plans d'étude automatisés
- **Système de priorités** : Organisation des tâches

La page de planning d'études, permettant à l'étudiant de visualiser son calendrier, d'organiser ses tâches et de suivre les examens à venir.

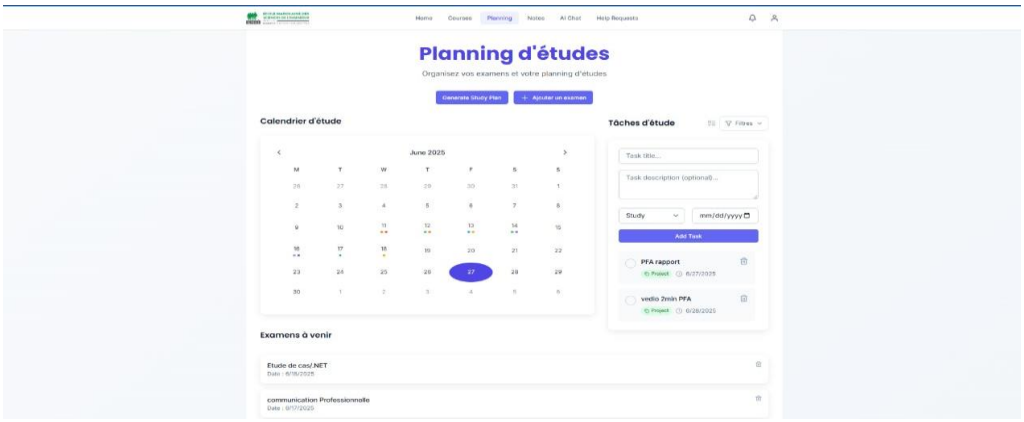

Figure 23 : Interface de gestion du planning d'études avec calendrier

Un exemple de plan d'étude hebdomadaire généré automatiquement au format PDF, facilitant l'organisation du temps de travail de l'étudiant.

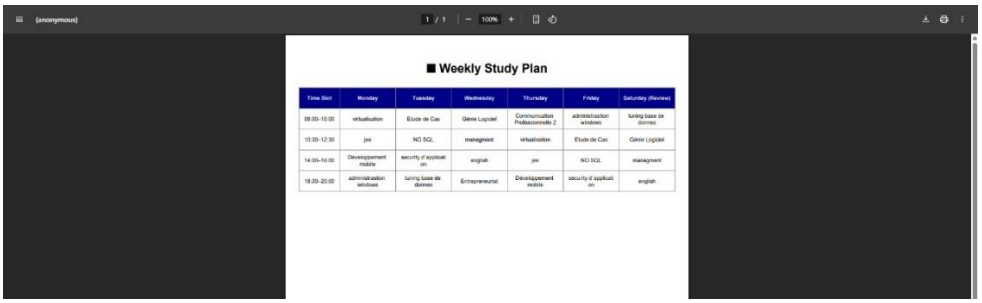

Figure 24 : Exemple de plan d'étude hebdomadaire généré en PDF

La fenêtre ci-dessous permet à l'étudiant d'ajouter plusieurs examens à son planning, en précisant le cours, la date et la priorité de chaque épreuve.

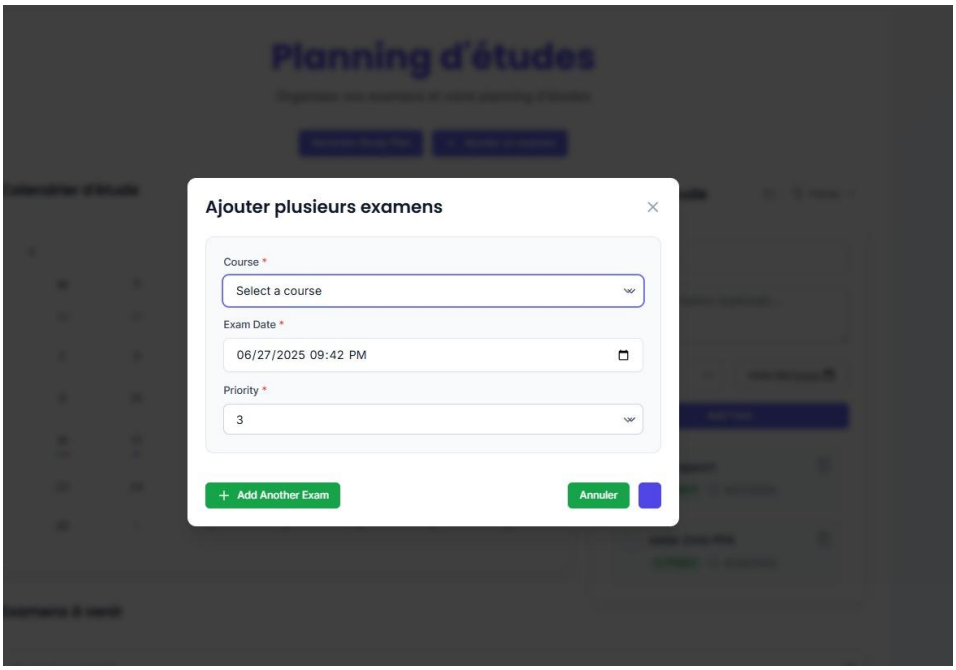

Figure 25 : Fenêtre d'ajout d'examens au planning

3.2. Fonctionnalités Avancées

3.2.1. Internationalisation

- **Support multilingue** : Français et anglais
- **Traduction dynamique** : Changement de langue en temps réel

Mise en évidence la fonctionnalité d'internationalisation, permettant à l'utilisateur de changer la langue de l'interface directement depuis la page de profil.

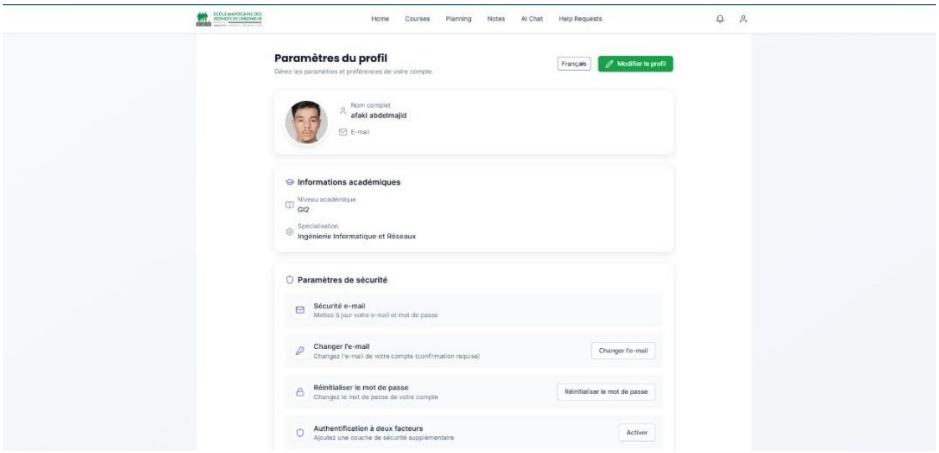

Figure 26 : Page de paramètres du profil avec sélecteur de langue

3.2.2. Gestion des fichier

- **Upload multiple** : Support de plusieurs fichiers simultanément
- **Prévisualisation** : Affichage des images et PDFs
- **Téléchargement sécurisé** : URLs signées pour l'accès aux fichiers

L'interface de gestion des fichiers d'un cours, avec une organisation claire par type de document et un accès rapide à l'ensemble des ressources.

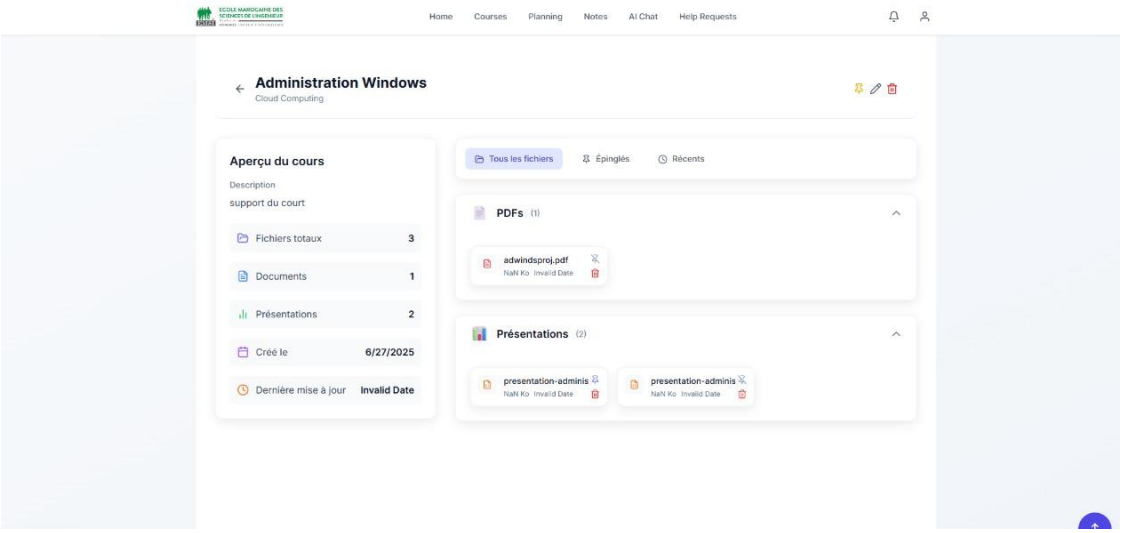

Figure 27 : Interface d'organisation des fichiers d'un cours

3.2.3. Système d'Entraide

- **Création d'annonces** : Publication de demandes d'aide par catégorie
- **Méthodes de contact** : Choix entre email et téléphone pour les réponses
- **Gestion des statuts** : Suivi des annonces ouvertes et fermées
- **Recherche et filtrage** : Navigation facilitée dans les demandes d'aide

Fenêtre de la création d'une demande d'aide, où l'étudiant peut spécifier le titre, la catégorie, le mode de contact et les informations nécessaires

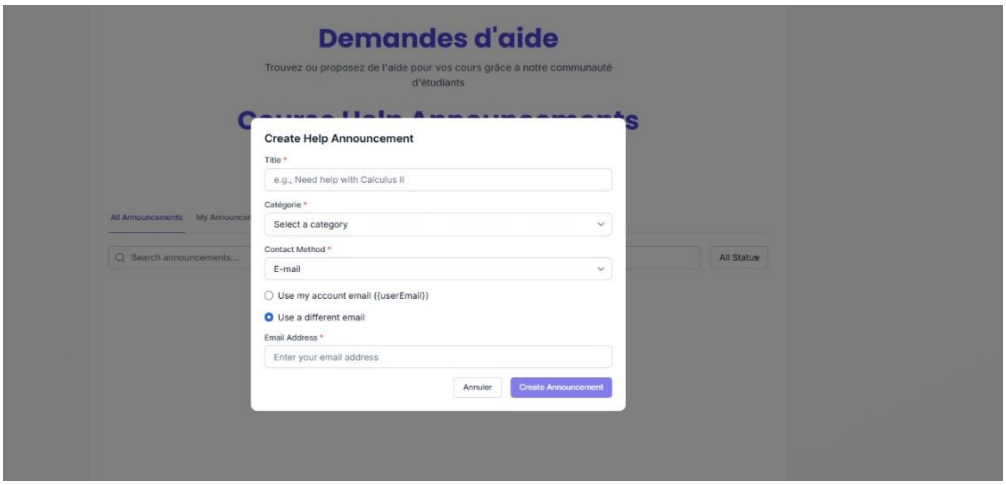

Figure 28 : Formulaire de création d'une annonce

La liste des demandes d'aide publiées, permettant aux étudiants de consulter, filtrer et répondre aux annonces selon leurs besoins.

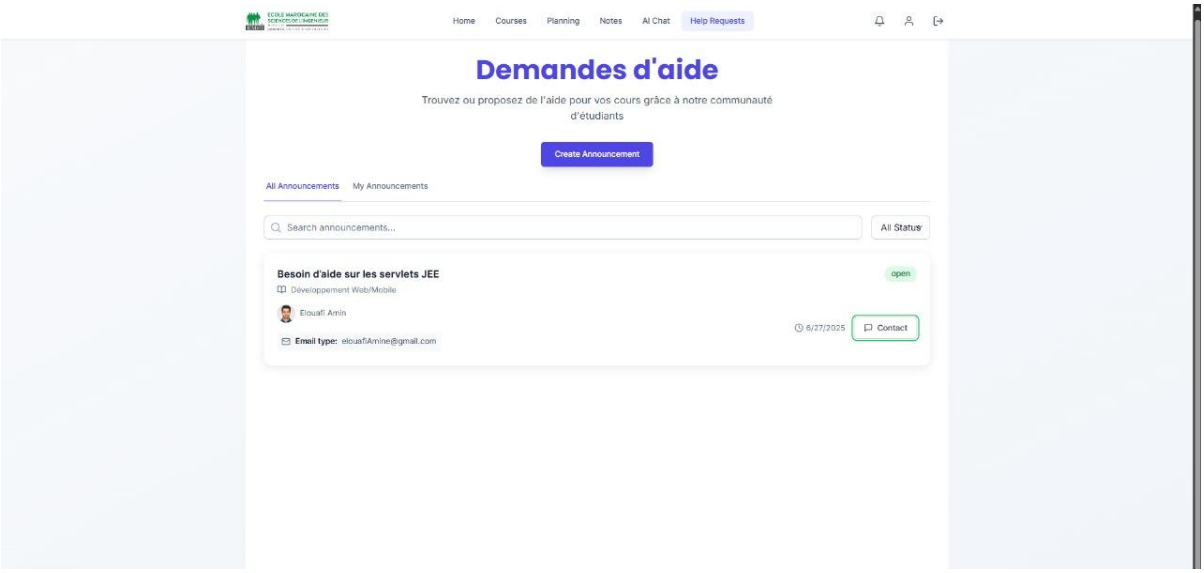

Figure 29 :Interface de gestion et de consultation des demandes d'aide

4. Parties du code importantes

4.1. Gestion des fichiers

Catégorisation des Fichiers

Ce code permet de classer les fichiers en différentes catégories (images, PDF, etc.) en fonction de leur type MIME ou de leur extension.

```typescript
// src/utility/FileCategory.ts
export function categorizeFiles(files: FileOut[]) {
  return {
    imageFiles: files.filter(file =>
      file.file_type.includes('image') ||
      file.file_name?.match(/\.(jpg|jpeg|png|gif|bmp|webp|svg)$/i)
    ),
    pdfFiles: files.filter(file =>
      file.file_type === 'application/pdf' ||
      file.file_name?.match(/\.pdf$/i)
    ),
    // ... autres catégories
  };
}
```

Figure 30 : Code de catégorisation des fichiers

4.2. Architecture Backend

Configuration FastAPI

Ce code initialise l'application StudySmart avec FastAPI, configure le CORS pour permettre la communication avec le frontend, et intègre les routeurs liés à l'authentification, à la gestion des fichiers et aux autres modules de l'application.

```python
# backend/main.py
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware

app = FastAPI(title=settings.PROJECT_NAME)

app.add_middleware(
    CORSMiddleware,
    allow_origins=settings.BACKEND_CORS_ORIGINS,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

# Inclusion des routeurs
app.include_router(auth_router, prefix="/auth", tags=["auth"])
app.include_router(file_router, prefix="/files", tags=["files"])
# ... autres routeurs
```

Figure 31 : Code de configuration de FastAPI

Intégration d'intelligence artificielle capable de répondre aux questions des étudiants.

```python
@router.post("/ai-chat")
async def ai_chat(messages: list[ChatMessage]):
    try:
        last_message = messages[-1].content.lower()

        try:
            file_name = extract_file_name_from_message(last_message)
            file_result = supabase.table("files").select("file_name").eq("file_name", file_name).execute()
            if file_result.data:
                return await explain_file(file_name)
        except HTTPException as fe:
            if fe.status_code != 400:
                raise fe

        chat_completion = client.chat.completions.create(
            messages=[msg.dict() for msg in messages],
            model="llama-3.3-70b-versatile",
        )
        reply = chat_completion.choices[0].message.content
        return {"reply": reply}
    except Exception as e:
        raise HTTPException(status_code=400, detail=f"AI request failed: {str(e)}")


@router.post("/explain_file")
async def explain_file(file_name: str):
    try:
        file_result = supabase.table("files").select("*").eq("file_name", file_name).execute()
        if not file_result.data:
            raise HTTPException(status_code=404, detail="File not found")

        file_path = file_result.data[0]["file_path"]
        signed_url = supabase.storage.from_("filesb").create_signed_url(file_path, 3600)
        if isinstance(signed_url, dict) and "error" in signed_url:
            raise HTTPException(status_code=400, detail=signed_url["error"])

        file_content = await download_file(signed_url['signedURL'])
        extracted_text = extract_text_by_file_type(file_name, file_content)

        ai_response = await ai_chat([
            ChatMessage(role="user", content=f"Explain this content: {extracted_text}")
        ])
        return {"reply": ai_response['reply']}
    except Exception as e:
        raise HTTPException(status_code=500, detail=f"Failed to explain the file: {str(e)}")
```

Figure 32 : Code pour intégration de l'IA

Gestion des Fichiers Backend

Ce code gère l'envoi de fichiers dans StudySmart : il vérifie les permissions de l'utilisateur, téléverse le fichier vers Supabase Storage, puis enregistre ses informations dans la base de données.

```python
# backend/api/files/routes.py
@router.post("/upload_file/{course_id}")
async def upload_file(
    course_id: str,
    file: UploadFile = File(...),
    user=Depends(get_current_user)
):
    try:
        # Vérification des permissions
        course_result = supabase.table("courses").select("user_id").eq("id", course_id).execute()
        if course_result.data[0]['user_id'] != user["id"]:
            raise HTTPException(status_code=403, detail="Access denied")

        # Upload vers Supabase Storage
        file_path = f"{course_id}/{file.filename}"
        upload_result = supabase.storage.from_('filesb').upload(file_path, file.file)

        # Enregistrement en base de données
        file_data = {
            "course_id": course_id,
            "file_name": file.filename,
            "file_path": file_path,
            "file_type": file.content_type,
            "file_size": file.size,
        }

        result = supabase.table("files").insert(file_data).execute()
        return result.data[0]

    except Exception as e:
        raise HTTPException(status_code=400, detail=str(e))
```

Figure 33 : Code de gestion des fichiers côté backend

Ce code gère les requêtes HTTP côté client, en configurant automatiquement les en-têtes, l'authentification et le traitement des erreurs pour communiquer avec l'API de la plateforme.

```typescript
async function client<T = any>(
  endpoint: string,
  {
    data,
    body,
    token,
    headers: customHeaders,
    responseType = 'json', // Default to JSON
    ...customConfig
  }: ClientOptions = {}
): Promise<T> {
  const finalBody = body ?? (data ? JSON.stringify(data) : undefined);

  const config: RequestInit = {
    method: finalBody ? 'POST' : 'GET',
    body: finalBody,
    credentials: 'include',
    headers: {
      ...(finalBody instanceof FormData ? {} : { 'Content-Type': 'application/json' }),
      ...(token ? { Authorization: `Bearer ${token}` } : {}),
      ...customHeaders,
    },
    ...customConfig,
  };

  if (typeof window !== 'undefined') {
    config.credentials = 'include';
  }

  try {
    const response = await fetch(`${BASE_URL}${endpoint}`, config);

    if (!response.ok) {
      let errorData;
      try {
        errorData = await response.json();
      } catch (e) {
        errorData = { message: response.statusText };
      }

      const error: ApiClientError = new Error(
        errorData.detail || errorData.message || 'Request failed'
      );
      error.status = response.status;
      error.data = errorData;
      throw error;
    }
```

Figure 34 : Fonction utilitaire pour la gestion centralisée des requêtes API

5. Tests

5.1. Stratégie de Tests

Tests Backend

Les tests backend englobent plusieurs niveaux de validation. Des tests unitaires sont réalisés pour vérifier la conformité des modèles Pydantic et la logique métier. Des tests d'intégration assurent le bon fonctionnement des différents endpoints de l'API, tandis que des tests de sécurité valident l'authentification et la gestion des permissions d'accès.

- **Tests unitaires** : Validation des modèles Pydantic et logique métier
- **Tests d'intégration** : Vérification des endpoints API
- **Tests de sécurité** : Validation de l'authentification et des permissions

Tests Frontend

Pour le frontend, la qualité de l'interface et des interactions est garantie par des tests de composants, qui valident le rendu et les comportements des éléments graphiques. Des tests d'intégration simulent des parcours utilisateur complets, et des tests de régression permettent de détecter rapidement toute anomalie introduite lors des évolutions du code.

- **Tests de composants** : Validation du rendu et des interactions
- **Tests d'intégration** : Vérification des flux utilisateur complets
- **Tests de régression** : Détection automatique des régressions

5.2. Outils de Test

Backend (Python)

Le backend est développé avec FastAPI, incluant une documentation interactive via Swagger. Pour les tests, FastAPI fournit des outils natifs permettant de tester les endpoints directement sans avoir recours à Pytest.

FastAPI : Framework principal pour le backend

Swagger : Documentation interactive des endpoints REST

Tests d'API : Validation des endpoints avec les outils natifs de FastAPI

Frontend (JavaScript/TypeScript)

Côté frontend, la validation repose sur des tests manuels approfondis des fonctionnalités utilisateur, des vérifications de compatibilité multi-navigateurs, et des tests de performance visant à optimiser les temps de chargement.

- **Tests manuels** : Validation des fonctionnalités utilisateur
- **Tests de compatibilité** : Vérification multi-navigateurs
- **Tests de performance** : Optimisation des temps de chargement

5.3. Couverture de Tests

Fonctionnalités Testées

La couverture des tests s'étend à toutes les fonctionnalités essentielles de la plateforme. Sont ainsi testés : l'authentification (inscription, connexion, gestion des sessions), la gestion des fichiers (upload, téléchargement, suppression), les opérations CRUD sur les cours, l'éditeur de notes (sauvegarde, formatage, export) et l'assistant IA (génération de réponses, gestion des conversations).

- **Authentification** : Inscription, connexion, gestion des sessions
- **Gestion des fichiers** : Upload, téléchargement, suppression
- **CRUD des cours** : Création, lecture, mise à jour, suppression
- **Éditeur de notes** : Sauvegarde, formatage, export
- **Assistant IA** : Génération de réponses, gestion des conversations

Tests de Sécurité

Des tests de sécurité spécifiques sont également menés pour valider la gestion des tokens JWT, le contrôle des permissions, la validation des fichiers uploadés et la protection contre les attaques par injection SQL.

- **Validation des tokens** : Authentification JWT
- **Permissions** : Accès aux ressources utilisateur
- **Upload de fichiers** : Validation des types et tailles
- **Injection SQL** : Protection contre les attaques

5.4. Tests de Performance

Métriques Mesurées

Enfin, la performance de la plateforme est évaluée à travers plusieurs métriques : le temps de réponse des API, le temps de chargement du frontend (optimisé par le lazy loading et le code splitting), l'utilisation mémoire et la capacité du système à gérer la montée en charge, notamment lors d'uploads multiples de fichiers.

- **Temps de réponse API** : Optimisation des requêtes
- **Temps de chargement frontend** : Lazy loading et code splitting
- **Utilisation mémoire** : Gestion efficace des ressources
- **Scalabilité** : Tests de charge pour les uploads multiples

6. Conclusion

Le développement de cette plateforme de gestion de fichiers éducative marque une étape significative dans l'évolution des outils numériques dédiés à l'apprentissage. L'architecture technique choisie, combinant les dernières technologies web avec des pratiques de développement modernes, a permis de créer une solution robuste et évolutive qui répond aux besoins spécifiques des étudiants et des établissements éducatifs. L'intégration réussie de fonctionnalités avancées telles que l'édition de texte riche, l'assistant IA, et la gestion intelligente des fichiers démontre la capacité de l'application à s'adapter aux exigences contemporaines de l'éducation numérique. Les choix technologiques, de Next.js et FastAPI à Supabase et TipTap, garantissent non seulement une performance optimale mais aussi une maintenabilité à long terme. L'approche centrée sur l'expérience utilisateur, couplée à un système de sécurité robuste et des tests complets, assure que la plateforme peut être déployée avec confiance dans des environnements éducatifs réels. Cette implémentation technique constitue une base solide pour l'évolution future de l'application, permettant l'ajout de nouvelles fonctionnalités et l'adaptation aux besoins émergents du secteur éducatif, tout en maintenant les standards de qualité et de sécurité établis.

Conclusion générale

Dans un contexte où les étudiants doivent gérer un grand volume d'informations, d'activités et de ressources tout au long de leur parcours académique, la mise en place d'un outil numérique intelligent et centralisé s'est imposée comme une solution pertinente et nécessaire. C'est dans cette optique que notre équipe a conçu et réalisé une application éducative innovante, combinant plusieurs fonctionnalités essentielles à la vie étudiante : gestion des cours, organisation de fichiers, prise de notes, planification d'examens et de tâches, entraide entre étudiants, et assistance personnalisée grâce à l'intelligence artificielle.

Tout au long du projet, nous avons adopté une approche méthodique, en commençant par la rédaction d'un cahier des charges précis, suivi d'une phase de conception à l'aide de diagrammes UML (cas d'utilisation, classes, séquences) pour bien structurer notre solution. L'architecture mise en place repose sur un modèle client-serveur, avec une séparation claire entre l'interface utilisateur (Next.js), la logique métier (FastAPI), la base de données (Supabase) et le module IA (Groq). Cette architecture modulaire garantit une bonne maintenabilité, une scalabilité future et une sécurité renforcée des données.

Le développement technique de l'application nous a permis de mettre en œuvre nos compétences en programmation web full-stack, en gestion de base de données relationnelle, en sécurisation des accès et en intégration de services intelligents. Nous avons aussi réalisé des tests unitaires et fonctionnels afin de vérifier le bon comportement de l'application et d'assurer une expérience utilisateur fluide.

Sur le plan pédagogique, ce projet nous a permis d'approfondir notre compréhension des méthodes de gestion de projet, de la modélisation logicielle, de l'interopérabilité entre outils modernes et de la communication entre les différents composants d'une application. Il a également renforcé notre capacité à travailler en équipe, à prendre des décisions techniques argumentées et à nous adapter aux contraintes et aux évolutions du développement logiciel.

En définitive, notre application se veut être une plateforme utile, moderne et évolutive, au service des étudiants. Elle peut constituer une base solide pour des améliorations futures, comme l'ajout d'un espace collaboratif en temps réel, l'envoi de notifications intelligentes, ou encore la publication sur les stores mobiles. Ce projet représente pour nous une étape importante, marquant la transition entre l'apprentissage académique et l'application concrète de nos compétences dans un environnement technologique actuel et orienté utilisateur.

Références Bibliographiques

[1] «Google Drive,» [En ligne]. Available: https://drive.google.com.

[2] «Microsoft OneDrive,» [En ligne]. Available: https://onedrive.live.com.

[3] «Moodle,» [En ligne]. Available: https://moodle.org.

[4] «WeTransfer,» [En ligne]. Available: https://wetransfer.com.

[5] «Next.js Documentation,» Vercel, [En ligne]. Available: https://nextjs.org/docs..

[6] «TypeScript,» [En ligne]. Available: https://www.typescriptlang.org.

[7] «Axios,» [En ligne]. Available: https://axios-http.com/docs/intro.

[8] «python,» [En ligne]. Available: https://www.python.org.

[9] «FastAPI Documentation,» FastAPI, [En ligne]. Available: https://fastapi.tiangolo.com.

[10] «Supabase Documentation,» Supabase, [En ligne]. Available: https://supabase.com/docs.

[11] «groq ai,» [En ligne]. Available: https://console.groq.com/docs/prompting .

[12] «Visual Studio Code documentation,» Microsoft, [En ligne]. Available: https://code.visualstudio.a/docs. [Accès le 31 Mars 2025].

[13] «GitHub Documentation,» GitHub, [En ligne]. Available: https://docs.github.com/fr. [Accès le 2 Avril 2025].

[14] M. Mythily, A. Samson Arun Raj, T. Joseph et Iwin, «An Analysis of the Significance of Spring Boot in The Market,» chez *2022 International Conference on Inventive Computation Technologies (ICICT)*, 2022.

[15] «PostgreSQL: Documentation,» [En ligne]. Available: https://www.postgresql.org/docs/. [Accès le 10 Avril 2025].

[16] C. Babu et G. Gunasingh, «DESH: Database evaluation system with hibernate ORM framework,» chez *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2016.

[17] A. Bonteanu et C. Tudose, «Performance Analysis and Improvement for CRUD Operations in Relational Databases from Java Programs Using JPA, Hibernate, Spring Data JPA,» *Applied Sciences,* n° %12743, 2024.

[18] «Introduction to JSON Web Tokens,» JWT , [En ligne]. Available: https://jwt.io/introduction. [Accès le 01 Avril 2025].

[19] «WebSocket in Spring,» Spring, [En ligne]. Available: https://docs.spring.io/spring-framework/reference/web/websocket.html. [Accès le 19 Avril 2025].

[20] «Python 3 Documentation,» Python Software Foundation, [En ligne]. Available: https://docs.python.org/3/.

[21] «Google Classroom,,» Google for Education, [En ligne]. Available: https://edu.google.com/intl/ALL_fr/products/classroom/.

[22] G. Siemens, «Connectivism: A Learning Theory for the Digital Age,» [En ligne]. Available: https://www.learningdevelopmentinstitute.org.

[23] P. Education, Intelligence Unleashed: An argument for AI in Education, 2016.

[24] M. B. e. C. F. W. Holmes, Artificial Intelligence in Education: Promises and Implications for Teaching and Learning, Center for Curriculum Redesign, 2019.

[25] T. R. e. H. M. Asghar, «Technology use, self-directed learning, student engagement and academic performance: Examining the interrelations,,» *Computers in Human Behavior, vol,* 2016.

[26] «tiptap,» [En ligne]. Available: https://tiptap.dev/docs/editor/extensions/nodes/document.