

---

# Rapport du Projet Système interactif d'Analyse des Feedbacks étudiants sur les Cours

---

Filière : Informatique et Ingénierie des Données



*Réalisé par :*

EL Warraqi IMANE  
Rhannouch NASSIMA  
Taftaf WIJDANE

*Encadre Par :*

Pr. Nidal LAMGHARI

Année universitaire  
2024/2025

# Table des matières

<b>Remerciements</b>	<b>3</b>
<b>Introduction</b>	<b>4</b>
0.1 Contexte et problématique . . . . .	4
0.2 Objectifs du projet . . . . .	4
<b>1 Analyse des besoins</b>	<b>5</b>
1.1 Étude des utilisateurs . . . . .	5
1.2 Analyse des besoins . . . . .	5
1.2.1 Besoins fonctionnels . . . . .	5
1.2.2 Besoins non fonctionnels . . . . .	5
<b>2 Conception du système</b>	<b>7</b>
2.1 Architecture générale du système . . . . .	7
2.2 Diagrammes UML . . . . .	9
2.2.1 Diagrammes de cas d'utilisation . . . . .	9
2.2.2 Diagramme de séquence . . . . .	9
2.2.3 Diagrammes de classes . . . . .	10
2.3 Modèle de base de données . . . . .	11
<b>3 Technologies et stack technique</b>	<b>13</b>
3.1 Choix technologiques et justifications . . . . .	13
3.2 Outils de développement et versioning (Git, GitHub) . . . . .	13
<b>4 Composant Machine Learning</b>	<b>14</b>
4.1 Dataset . . . . .	14
4.2 Modèles utilisés . . . . .	14
4.3 Prétraitement des données . . . . .	14
4.4 Entraînement des modèles . . . . .	15
4.5 Évaluation des modèles de machine learning . . . . .	15
4.5.1 Métriques : précision, rappel, F1-score (modèle français) . . . . .	15
4.5.2 Métriques : précision, rappel, F1-score (modèle anglais) . . . . .	16
4.6 Détection automatique de la langue pour classifier le sentiment . . . . .	17
<b>5 Fonctionnement du chatbot</b>	<b>19</b>
5.1 Entraînement du modèle de classification des feedbacks sérieux . . . . .	19
5.1.1 Objectif . . . . .	19
5.1.2 Dataset . . . . .	19
5.1.3 Prétraitement . . . . .	19
5.1.4 Détection de langue et routage . . . . .	19

5.2	Résultats et évaluation des modèles de filtrage . . . . .	20
5.2.1	Modèle français . . . . .	20
5.2.2	Modèle anglais . . . . .	20
5.3	Déroulement de l'interaction . . . . .	21
5.3.1	1. Collecte du feedback . . . . .	21
5.3.2	2. Analyse . . . . .	21
5.4	Architecture technique . . . . .	22
5.5	Intérêt . . . . .	22
<b>6</b>	<b>Génération automatique de recommandations par IA</b>	<b>23</b>
6.1	Idée générale de la recommandation IA . . . . .	23
6.2	conception de la réalisation de la recommandation IA . . . . .	23
6.2.1	Dataset de la recommandation . . . . .	24
6.2.2	Modèle entraîné de la recommandation . . . . .	25
6.2.3	Intégration dans Django : génération des recommandations et envoi automatique par mail . . . . .	26
<b>7</b>	<b>Implémentation</b>	<b>27</b>
7.1	Développement des interfaces . . . . .	27
7.1.1	Interface Étudiant / Dashboard . . . . .	27
7.1.2	Interface Chef de filière / Dashboard . . . . .	28
7.1.3	Interface Administrateur / Dashboard . . . . .	30
7.2	Développement Back-End avec Django . . . . .	33
7.2.1	Modélisation des données . . . . .	33
7.2.2	Interface d'administration Django . . . . .	38
<b>8</b>	<b>Tests et évaluation</b>	<b>40</b>
8.1	Tests fonctionnels . . . . .	40
<b>Difficultés rencontrées</b>		<b>48</b>
<b>Conclusion et perspectives</b>		<b>50</b>

# Remerciements

---

Nous souhaitons adresser nos remerciements les plus sincères à notre encadrant, **Pr. Nidal Lamghari**, pour son accompagnement précieux tout au long de ce projet. Sa disponibilité, sa patience et ses conseils clairs ont joué un rôle essentiel dans la réussite de notre travail. Il a toujours su nous orienter avec justesse, en nous apportant des explications accessibles et des remarques constructives, tout en nous laissant l'espace nécessaire pour réfléchir et avancer par nous-mêmes.

Grâce à son encadrement, nous avons pu acquérir des compétences solides, non seulement sur le plan technique (dans le domaine du machine learning, du traitement du langage naturel et du développement web), mais aussi sur le plan méthodologique et humain. Il nous a appris à travailler de manière rigoureuse, à gérer les difficultés avec méthode, et à rester motivés même lorsque certaines étapes du projet étaient plus complexes ou longues.

Nous avons particulièrement apprécié la confiance qu'il nous a accordée, sa bienveillance, ainsi que l'intérêt réel qu'il a porté à notre sujet. Son implication nous a motivés à donner le meilleur de nous-mêmes. Ce projet restera pour nous une expérience enrichissante, et une étape importante dans notre parcours d'apprentissage.

Encore une fois, nous lui exprimons toute notre reconnaissance pour son soutien constant, sa pédagogie, et la qualité de son encadrement.

# Introduction

---

## 0.1 Contexte et problématique

Dans un contexte pédagogique en constante évolution, l'amélioration continue de la qualité de l'enseignement constitue un enjeu majeur pour les établissements universitaires. Les feedbacks des étudiants représentent une source précieuse d'information pour évaluer l'efficacité des cours, la qualité de l'enseignement, et identifier les points d'amélioration. Cependant, la collecte et l'analyse de ces retours restent souvent manuelles, limitées, et peu exploitables à grande échelle.

La problématique centrale réside donc dans la mise en place d'un système capable de traiter automatiquement ces feedbacks, souvent formulés sous forme de texte libre, afin d'en extraire des informations pertinentes telles que le sentiment exprimé (positif, négatif, neutre) et les éléments fréquemment mentionnés. Cela nécessite l'intégration de techniques d'intelligence artificielle, notamment de traitement du langage naturel (NLP), ainsi qu'une interface interactive pour la visualisation des résultats par les différentes parties prenantes (étudiant, chefs de filière, administration).

## 0.2 Objectifs du projet

Ce projet a pour principal objectif la conception et la réalisation d'un système interactif d'analyse des feedbacks étudiants sur les cours. Les objectifs spécifiques sont les suivants :

- Automatiser la collecte des feedbacks via une interface dédiée aux étudiants.
- Analyser le contenu textuel des feedbacks à l'aide de techniques de NLP et de classification (détection de sentiment).
- Stocker les résultats dans une base de données structurée pour faciliter l'accès et la gestion.
- Développer des interfaces utilisateurs interactives (étudiant, admin, chef de filière) pour afficher les feedbacks et leurs analyses.
- Générer des tableaux de bord dynamiques permettant de visualiser les tendances et d'alerter les responsables pédagogiques en cas de feedbacks négatifs récurrents.

# 1. Analyse des besoins

---

Afin de concevoir un système interactif d'analyse des feedbacks étudiants sur les cours, il est essentiel d'identifier les besoins des différents acteurs impliqués ainsi que les fonctionnalités attendues pour répondre efficacement aux enjeux.

## 1.1 Étude des utilisateurs

Dans cette section, nous identifions les différents profils utilisateurs du système, leurs rôles et leurs attentes spécifiques.

- **Étudiants** : Soumission des feedbacks, consultation de leurs anciens avis et suivi des réponses.
- **Chefs de filière** : Analyse et visualisation des feedbacks.
- **Administrateurs** : modifier ou ajouter un chef de filière, suivi général.

## 1.2 Analyse des besoins

### 1.2.1 Besoins fonctionnels

Les besoins fonctionnels décrivent les services et fonctionnalités que le système doit offrir pour satisfaire les attentes des utilisateurs.

- **Soumission des feedbacks** : Les étudiants doivent pouvoir soumettre facilement leurs avis sur les cours suivis via une interface intuitive.
- **Catégorisation automatique des feedbacks** : Le système doit classifier les avis en différentes catégories (positif, négatif, suggestion d'amélioration, etc.) pour faciliter l'analyse.
- **Analyse et visualisation des feedbacks** : Les chefs de filière doivent avoir accès à des statistiques et graphiques leur permettant d'identifier les tendances et les points d'amélioration.
- **Notifications automatiques** : Le système doit alerter les professeurs par email en cas de feedbacks négatifs récurrents.
- **Gestion des utilisateurs** : Chaque utilisateur (étudiant, chef de filière, administrateur) doit disposer d'un accès sécurisé et d'un espace dédié avec des fonctionnalités adaptées à son rôle.

### 1.2.2 Besoins non fonctionnels

Ces besoins concernent la qualité du système, sa performance, sa sécurité et son expérience utilisateur.

- **Accessibilité et ergonomie** : L'interface doit être simple d'utilisation, ergonomique et accessible depuis différents appareils (ordinateurs, tablettes, smartphones).
- **Sécurité des données** : Les feedbacks et données utilisateurs doivent être stockés de manière sécurisée, avec des accès restreints aux personnes autorisées.
- **Scalabilité** : Le système doit pouvoir gérer un grand volume de feedbacks et d'utilisateurs sans perte de performance ni ralentissement.

## 2. Conception du système

---

### 2.1 Architecture générale du système

L'architecture du système repose sur une structure modulaire et orientée services, permettant de traiter les feedbacks étudiants depuis leur saisie jusqu'à leur analyse automatique, leur visualisation et leur exploitation pédagogique.

#### Interface utilisateur (Frontend)

Le système comporte trois espaces distincts :

- **Espace étudiant** .
- **Espace enseignant** .
- **Espace administrateur** .

Les interfaces sont développées en HTML/CSS/JavaScript.

#### Chatbot intelligent

Lorsqu'un étudiant soumet un feedback via l'interface, celui-ci est d'abord traité par un modèle de classification de sentiment. Si le feedback est identifié comme **négatif**, l'étape pour vérifier la **pertinence réelle du message**. Ce contrôle est effectué par le **chatbot**, intégré dans l'espace étudiant.

- recueille les messages des étudiants,
- détecte automatiquement la langue (fr/en),
- évalue la pertinence du message (sérieux ou non),

#### Backend Django

Le cœur du système est une application Django qui assure :

- la gestion des utilisateurs, des rôles et de l'authentification,
- la réception et le stockage des feedbacks dans MySQL,
- la communication avec les modèles IA via des endpoints internes ou API.

Une API REST permet la communication entre le frontend et le backend (`/predict`, `/submit`, etc.).

#### Modules NLP et Machine Learning

Deux pipelines de classification sont intégrés :

- **Classification de sérieux** : filtrage des feedbacks non pertinents à l'aide de modèles supervisés.

- **Classification de sentiment** : détection de tonalité (positif, neutre, négatif) selon la langue.

Chaque pipeline comprend :

- prétraitement avec TF-IDF,
- détection de langue (`langdetect`),
- un modèle ML (Naive Bayes ou Régression Logistique).

## Dashboards interactifs (Dash)

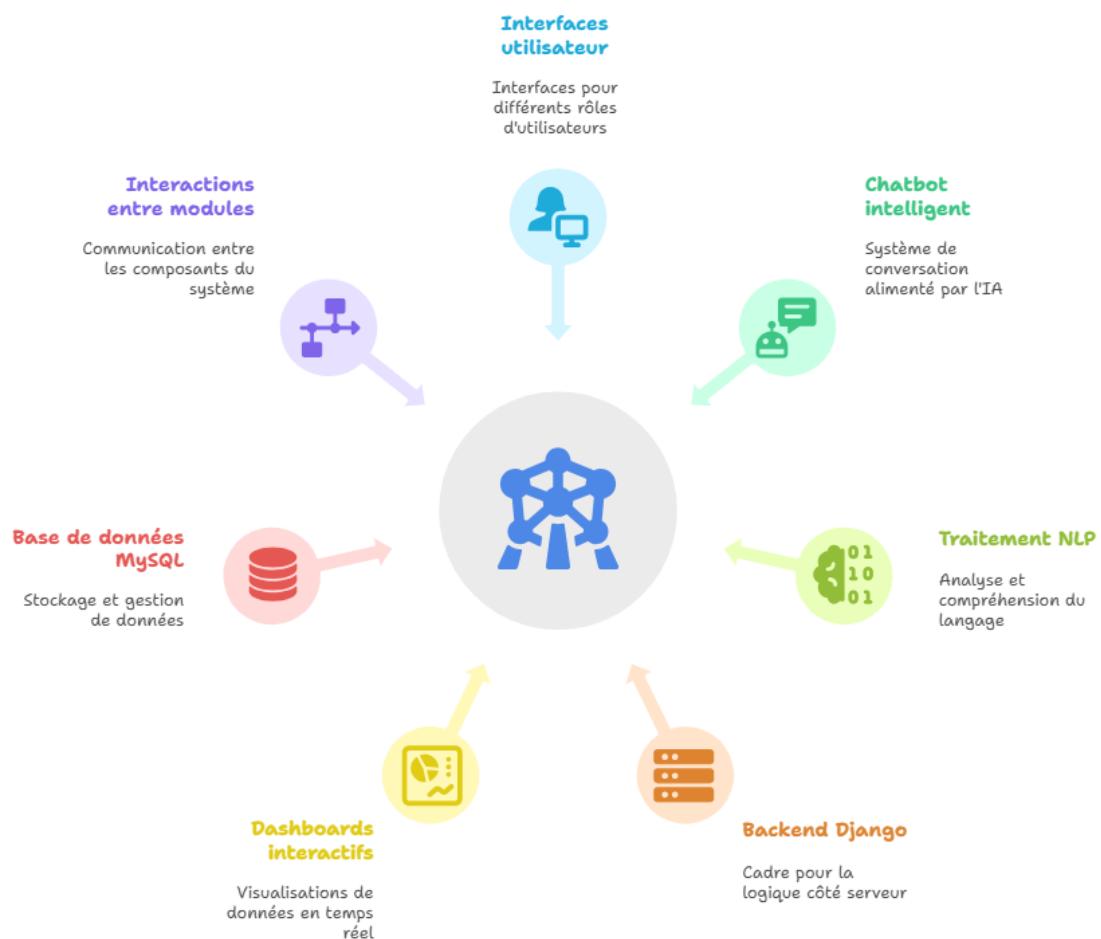
Développés avec la bibliothèque Dash en Python, les tableaux de bord permettent :

- la visualisation des feedbacks reçus,
- le suivi des tendances et des sentiments,
- la réception automatique de recommandations en cas de feedbacks négatifs.

## Base de données

Le système s'appuie sur une base MySQL pour stocker :

- les utilisateurs et rôles,
- les feedbacks et leurs annotations,
- l'historique des prédictions et des recommandations.

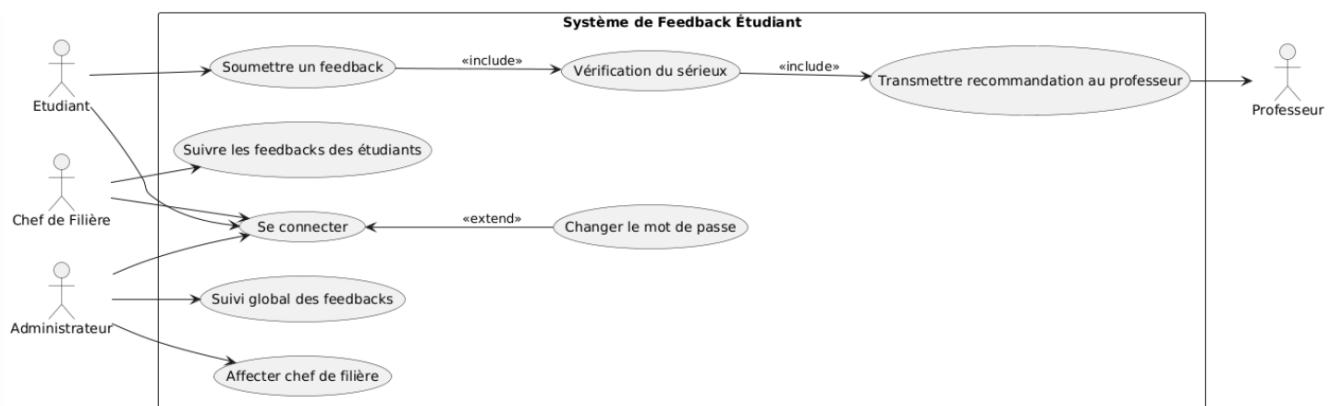


Cette architecture garantit une automatisation complète, un traitement intelligent des données, et une visualisation claire pour les utilisateurs finaux.

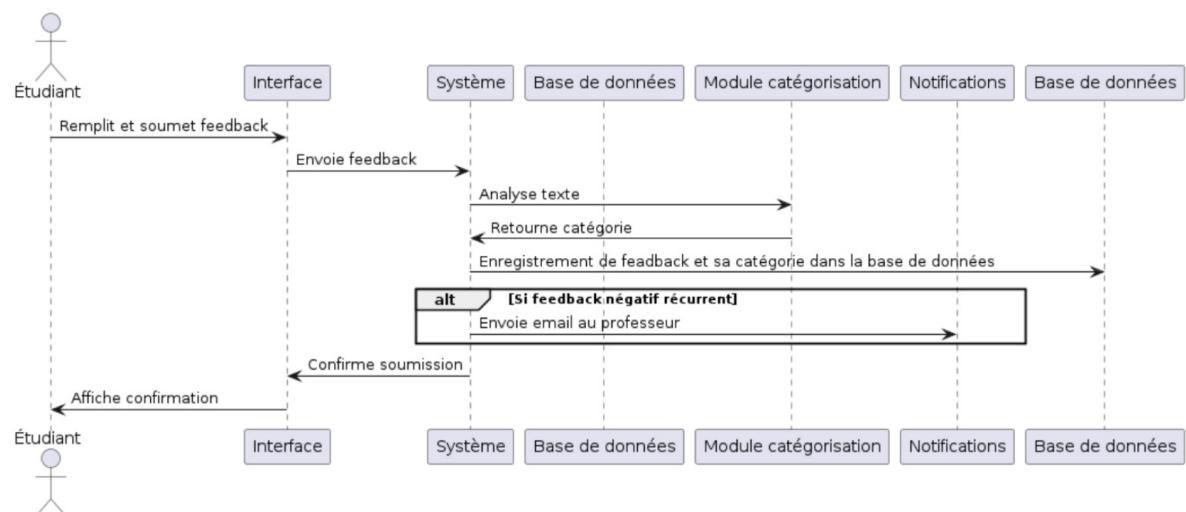
## 2.2 Diagrammes UML

### 2.2.1 Diagrammes de cas d'utilisation

Voici le diagramme de cas d'utilisation de notre application *FeedbackFlow* : Ce diagramme de cas d'utilisation représente les principales interactions entre les utilisateurs du système. Les étudiants peuvent se connecter et soumettre des feedbacks sur les cours, qui sont automatiquement analysés et catégorisés. Les chefs de filière ont la possibilité de consulter ces feedbacks, d'analyser les tendances et d'identifier les axes d'amélioration. De leur côté, les administrateurs gèrent les utilisateurs et les paramètres du système. Une fonctionnalité importante permet l'envoi automatique de notifications aux chefs de filière lorsqu'un volume élevé de feedbacks négatifs est détecté, afin de permettre une réaction rapide.



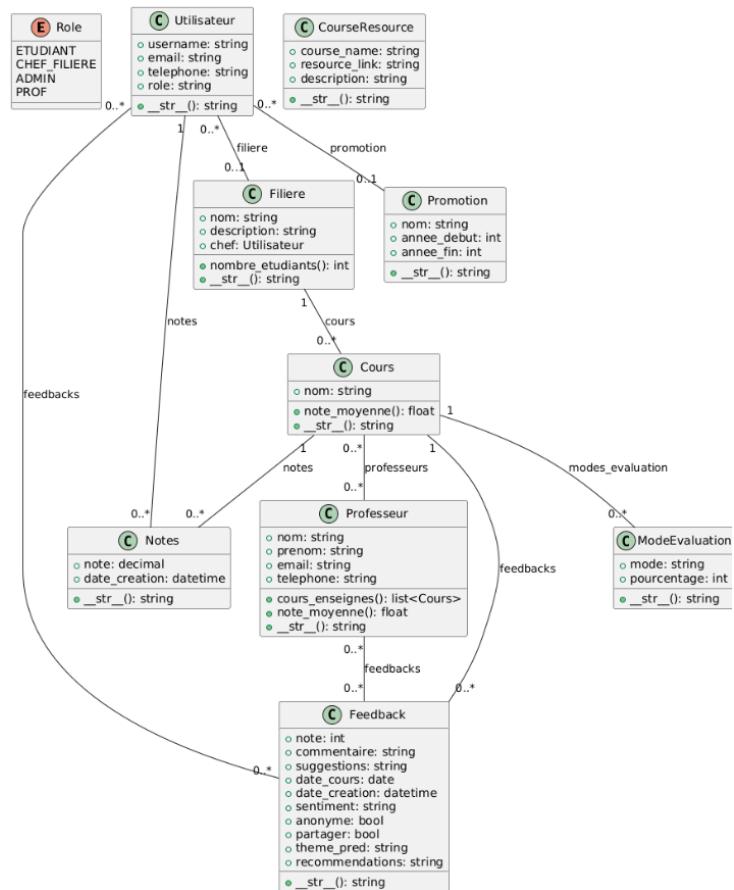
### 2.2.2 Diagramme de séquence



Le diagramme de séquence illustre les échanges entre les acteurs et composants du système lors de la soumission d'un feedback par un étudiant. Il montre comment le feedback est saisi, enregistré, analysé automatiquement pour être catégorisé, puis mis à jour en base de données. En cas de feedback négatif récurrent, une notification est envoyée au professeur. Enfin, une confirmation est affichée à l'étudiant. Ce diagramme met en évidence le déroulement logique et les interactions nécessaires à cette fonctionnalité.

### 2.2.3 Diagrammes de classes

Voici le diagramme de classes de notre application *FeedbackFlow* :



Ce diagramme modélise les différentes entités de l'application de gestion des feedbacks académiques ainsi que leurs relations. Il met en évidence les interactions entre les utilisateurs (étudiants, professeurs, chefs de filière, administrateurs) et les composantes du système éducatif, telles que les filières, promotions, cours et professeurs. Les étudiants peuvent soumettre des feedbacks sur les cours, comprenant des notes, commentaires, suggestions, sentiments analysés, thèmes prédictifs, et recommandations. Les cours sont également associés à des modes d'évaluation et à des ressources pédagogiques. Chaque filière est dirigée par un chef de filière et peut regrouper plusieurs promotions. Ce modèle de données favorise la centralisation et l'analyse qualitative et quantitative des retours étudiants, dans une optique d'amélioration continue des pratiques pédagogiques.

## 2.3 Modèle de base de données

Dans cette section, nous présentons la conception du modèle de données utilisé dans notre projet, ainsi que la structure de la base de données MySQL qui stocke les informations essentielles.

Le modèle de base de données de notre projet est conçu pour gérer les utilisateurs, les filières, les promotions, les cours, les professeurs, ainsi que les feedbacks et notes associées aux cours. Il est implémenté en Django et stocké dans une base de données MySQL.

**Utilisateur** : Ce modèle étend la classe `AbstractUser` de Django et inclut des champs spécifiques tels que l'email, le téléphone, le rôle (Étudiant, Chef de filière, Administrateur), la filière et la promotion. Il permet de gérer les différents types d'utilisateurs du système.

**Filiere** : Représente les filières de formation, avec un nom et une description. Chaque filière est associée à un chef de filière et plusieurs étudiants.

**Promotion** : Représente une promotion d'étudiants avec un nom et les années de début et de fin.

**Cours** : Contient les informations sur les cours proposés, liés à une filière spécifique. Chaque cours peut avoir plusieurs professeurs.

**Professeur** : Stocke les informations personnelles des professeurs ainsi que les cours qu'ils enseignent.

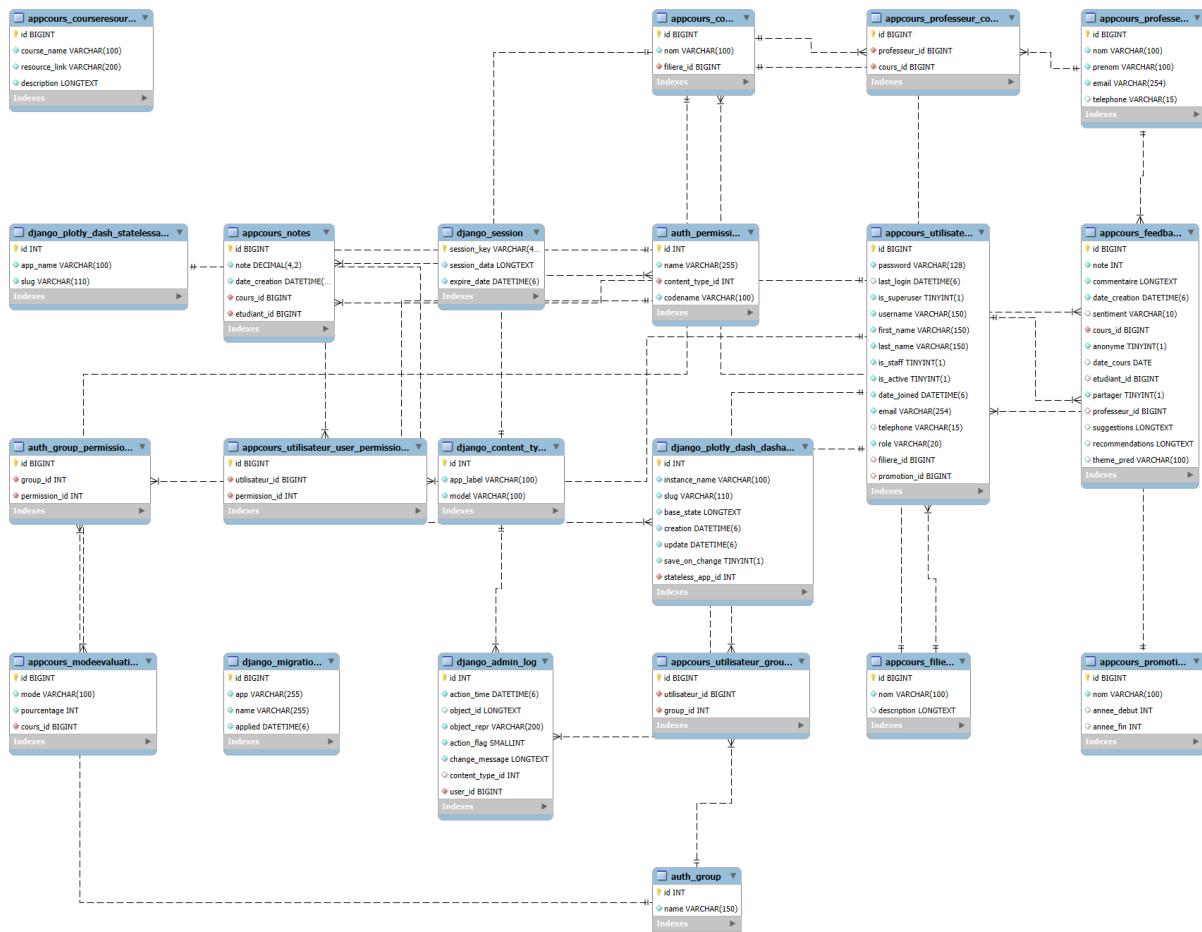
**Feedback** : Permet aux étudiants de laisser des retours (notes, commentaires, suggestions) sur les cours et les professeurs, avec un système d'analyse de sentiment (positif, neutre, négatif).

**Notes** : Gère les notes attribuées aux étudiants pour chaque cours, avec un suivi des dates.

**ModeEvaluation** : Définit les différents modes d'évaluation pour un cours (examen, projet, TP, etc.) et leur pondération dans la note finale.

**CourseResource** : (optionnel) Contient des ressources associées aux cours, telles que des liens et descriptions.

Cette structure relationnelle permet une gestion complète et cohérente des données académiques dans l'application, facilitant l'administration et l'analyse des informations liées aux filières, aux utilisateurs et à la qualité pédagogique.



## 3. Technologies et stack technique

---

### 3.1 Choix technologiques et justifications

Le choix des technologies repose sur des critères de performance, de simplicité d'intégration, de compatibilité avec les bibliothèques de Machine Learning, et de rapidité de développement.

- **HTML / CSS** : ces technologies standard du Web ont été choisies pour créer des interfaces utilisateurs simples, accessibles et compatibles avec tous les navigateurs. Leur large support communautaire facilite le développement.
- **JavaScript** : utilisé pour ajouter de l'interactivité aux pages (validation de formulaires, mise à jour dynamique, navigation fluide). Il permet une meilleure expérience utilisateur, en complément de Django côté serveur.
- **Django (Python)** : framework web rapide et sécurisé. Il est parfaitement compatible avec les bibliothèques de Machine Learning (scikit-learn, NLTK) et fournit un système complet de gestion des vues, modèles, formulaires et authentification.
- **Dash** : choisi pour construire des dashboards interactifs intégrés. Dash permet de générer des graphiques en temps réel, utiles aux enseignants et administrateurs pour visualiser les feedbacks.
- **MySQL** : système de base de données relationnelle open source. Son intégration avec Django est stable via le connecteur ‘mysqlclient’. Il est robuste pour stocker les utilisateurs, feedbacks, recommandations, etc.
- **Python** : langage principal du projet pour le back-end et l'analyse de données. Il est largement utilisé en data science et compatible avec les outils NLP et ML nécessaires à ce projet.

### 3.2 Outils de développement et versioning (Git, GitHub)

- **Git** : permet de versionner le code source, suivre les modifications, et faciliter la collaboration entre membres du groupe.
- **GitHub** : utilisé pour héberger le projet, documenter les tâches, et gérer les pull requests. Il offre également un suivi clair de l'évolution du code.

## 4. Composant Machine Learning

---

### 4.1 Dataset

Nous avons utilisé un jeu de données contenant des commentaires étudiants 2000 lignes. Chaque commentaire est associé à une étiquette de sentiment : *positif*, *négatif* ou *neutre*. Deux versions ont été utilisées : une en langue française, une autre en anglais.

<b>id</b>	<b>comment</b>	<b>sentiment</b>
1996	Not much stood out to me in this course.	neutral
1997	This class sparked my interest in the subject.	positive
1998	Instructions for tasks were often confusing.	negative
1999	Instructor's feedback was unclear and inconsis...	negative
2000	It was neither too easy nor too hard.	neutral

<b>id</b>	<b>commentaire</b>	<b>sentiment</b>
1	Le contenu est intéressant et bien expliqué.	positif
2	Formation neutre, pas très engageante.	neutre
3	Le professeur ne répondait pas aux questions.	négatif
4	Le cours était très clair et bien structuré.	positif
5	Le cours était très clair et bien structuré.	positif

### 4.2 Modèles utilisés

Pour classer les commentaires selon leur tonalité, nous avons utilisé deux modèles de classification :

- **Naïve Bayes** : simple, rapide et efficace pour les textes courts (utilisé pour les commentaires en français).
- **Régression Logistique** : plus robuste, souvent utilisée pour des problèmes de classification binaire ou multiclasse (utilisée pour les commentaires en anglais).

### 4.3 Prétraitement des données

Avant d'entraîner les modèles, les textes doivent être nettoyés et préparés :

- Conversion en minuscules

- Suppression de la ponctuation, des chiffres, des caractères spéciaux
- Suppression des mots vides (ex : *et, de, un, the, is*)
- Lemmatisation ou racinisation des mots
- Vectorisation avec **TF-IDF** : pour transformer le texte en chiffres exploitables par les modèles.

```
def preprocess_text(text):
    # Minuscule
    text = text.lower()
    # Supprimer la ponctuation
    text = ''.join([c for c in text if c not in string.punctuation])
    # Tokenisation
    tokens = word_tokenize(text, language='french')
    # Retirer les stop words
    tokens = [word for word in tokens if word not in stop_words]
    return ' '.join(tokens)
```

## 4.4 Entraînement des modèles

Les données prétraitées ont été séparées en deux groupes :

- **Train** : pour apprendre (80%)
- **Test** : pour vérifier les performances (20%)

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(stop_words=list(stop_words))),
    ('clf', MultinomialNB())
])

# Entraînement
pipeline.fit(X_train, y_train)
```

## 4.5 Évaluation des modèles de machine learning

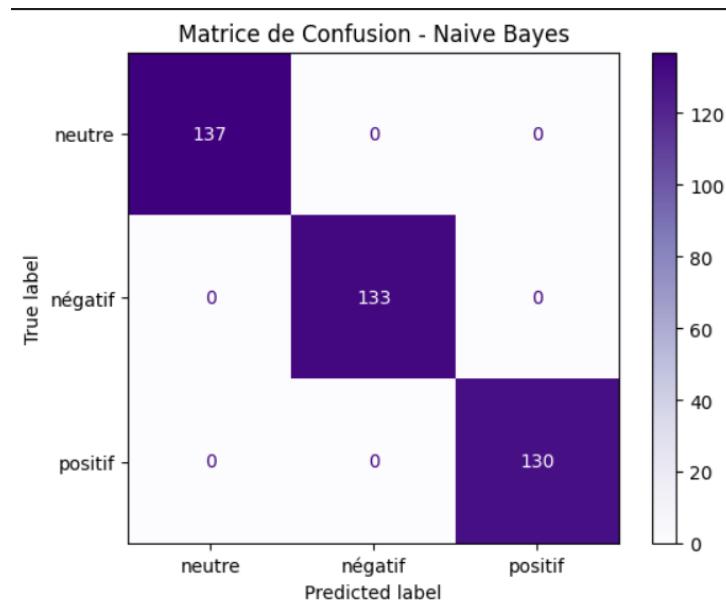
### 4.5.1 Métriques : précision, rappel, F1-score (modèle français)

Pour évaluer la performance du modèle Naive Bayes utilisé sur les feedbacks en langue française, nous avons calculé les métriques suivantes :

- **Précision** : taux de commentaires bien classés parmi ceux prédits comme tels.
- **Rappel** : taux de commentaires retrouvés parmi tous ceux réellement de cette classe.
- **F1-score** : moyenne harmonique entre précision et rappel.

	precision	recall	f1-score
neutre	1.00	1.00	1.00
négatif	1.00	1.00	1.00
positif	1.00	1.00	1.00
accuracy			1.00
macro avg	1.00	1.00	1.00
weighted avg	1.00	1.00	1.00

Les résultats montrent que le modèle obtient des scores parfaits (1.00) pour toutes les classes, ce qui indique une classification sans erreur sur le jeu de test.



La matrice de confusion montre que les classes *positif*, *négatif* et *neutre* ont été parfaitement prédites, sans erreurs de classification.

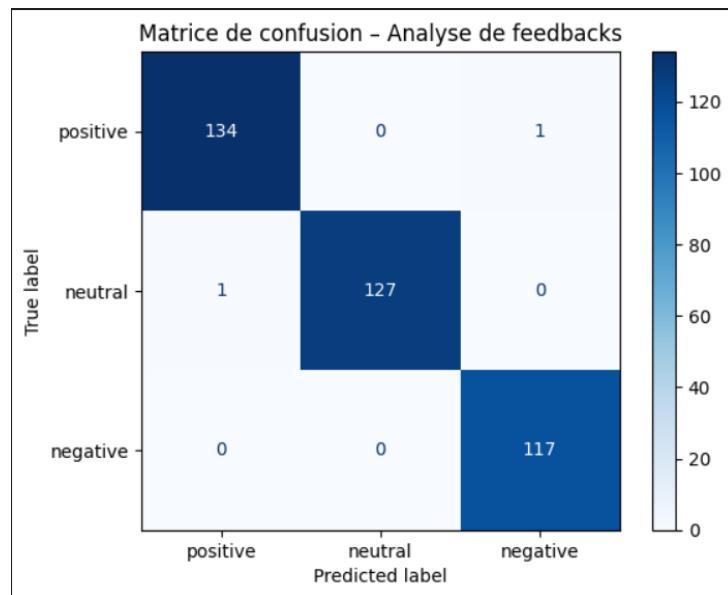
#### 4.5.2 Métriques : précision, rappel, F1-score (modèle anglais)

Le modèle de Régression Logistique, appliqué aux commentaires en langue anglaise, donne d'excellents résultats :

- **Précision moyenne** : 99% des prédictions sont correctes.
- **Rappel élevé** : les vrais commentaires sont bien identifiés.
- **F1-score proche de 1** : équilibre entre précision et rappel.

	precision	recall	f1-score
negative	0.99	1.00	1.00
neutral	1.00	0.99	1.00
positive	0.99	0.99	0.99
accuracy			0.99
macro avg	0.99	0.99	0.99
weighted avg	0.99	0.99	0.99

On constate que les trois classes (positif, neutre, négatif) sont bien traitées, avec un très faible taux d'erreur.



La matrice montre seulement deux erreurs mineures :

- Un commentaire positif prédit comme négatif.
- Un commentaire neutre prédit comme positif.

Cela confirme la robustesse du modèle sur les textes anglais.

## 4.6 Détection automatique de la langue pour classifier le sentiment

Afin de traiter automatiquement les feedbacks rédigés en français ou en anglais, une détection de la langue est réalisée en amont. Cette étape est essentielle pour appliquer le modèle de Machine Learning adapté à la langue du commentaire.

La bibliothèque `langdetect` est utilisée pour identifier la langue du texte. Une fois la langue détectée :

- Si la langue est le **français**, le texte est traité par un modèle **Naive Bayes**.

- Si la langue est l'**anglais**, un modèle de **Régression Logistique** est utilisé.
- Si la langue n'est ni française ni anglaise, le système indique que la langue n'est pas prise en charge.

```
from langdetect import detect

import joblib

model_fr = joblib.load("sentiment_model_fr_nb.pkl")
model_en = joblib.load("C://Users//WIJDANE TAFTA//Desktop//projet//backend//nlp_logistic_sentiment_model.pkl")

def detect_sentiment(text):
    lang = detect(text)

    if lang == 'fr':
        prediction = model_fr.predict([text])
        return f"Sentiment en français : {prediction[0]}"

    elif lang == 'en':
        prediction = model_en.predict([text])
        return f"[EN] Sentiment en anglais : {prediction[0]}"

    else:
        return "Langue non prise en charge."
```

Ce mécanisme rend le système intelligent et flexible : il s'adapte automatiquement à la langue d'entrée, sans que l'utilisateur ait besoin de sélectionner une option manuellement.

# 5. Fonctionnement du chatbot

---

## 5.1 Entraînement du modèle de classification des feed-backs sérieux

### 5.1.1 Objectif

Avant l'intégration dans le système, une phase d'apprentissage supervisé a été réalisée pour concevoir un modèle capable de distinguer un retour sérieux d'un message inutile ou moqueur. L'objectif est de permettre au chatbot de filtrer automatiquement les commentaires non pertinents avant qu'ils ne soient analysés ou transmis aux enseignants.

### 5.1.2 Dataset

Deux jeux de données ont été construits :

- Un en **français**, composé de messages étudiants simulés et réels, annotés manuellement.
- Un autre en **anglais**, à partir de traductions ou d'exemples similaires.

Chaque message est associé à une étiquette : *sérieux* (1) ou *non sérieux* (0). Les deux datasets ont été équilibrés pour que chaque classe soit représentée équitablement.

### 5.1.3 Prétraitement

Les données textuelles ont été nettoyées : suppression de ponctuation inutile, transformation en minuscules, suppression des mots vides, etc. Ensuite, les textes ont été convertis en représentations numériques adaptées à l'entraînement de modèles de classification. Les données ont été divisées en deux groupes : 80 % pour l'apprentissage, 20 % pour l'évaluation.

### 5.1.4 Détection de langue et routage

Pour que le système soit capable de gérer les deux langues, une détection automatique de la langue a été mise en place avec la bibliothèque `langdetect`. Selon la langue détectée :

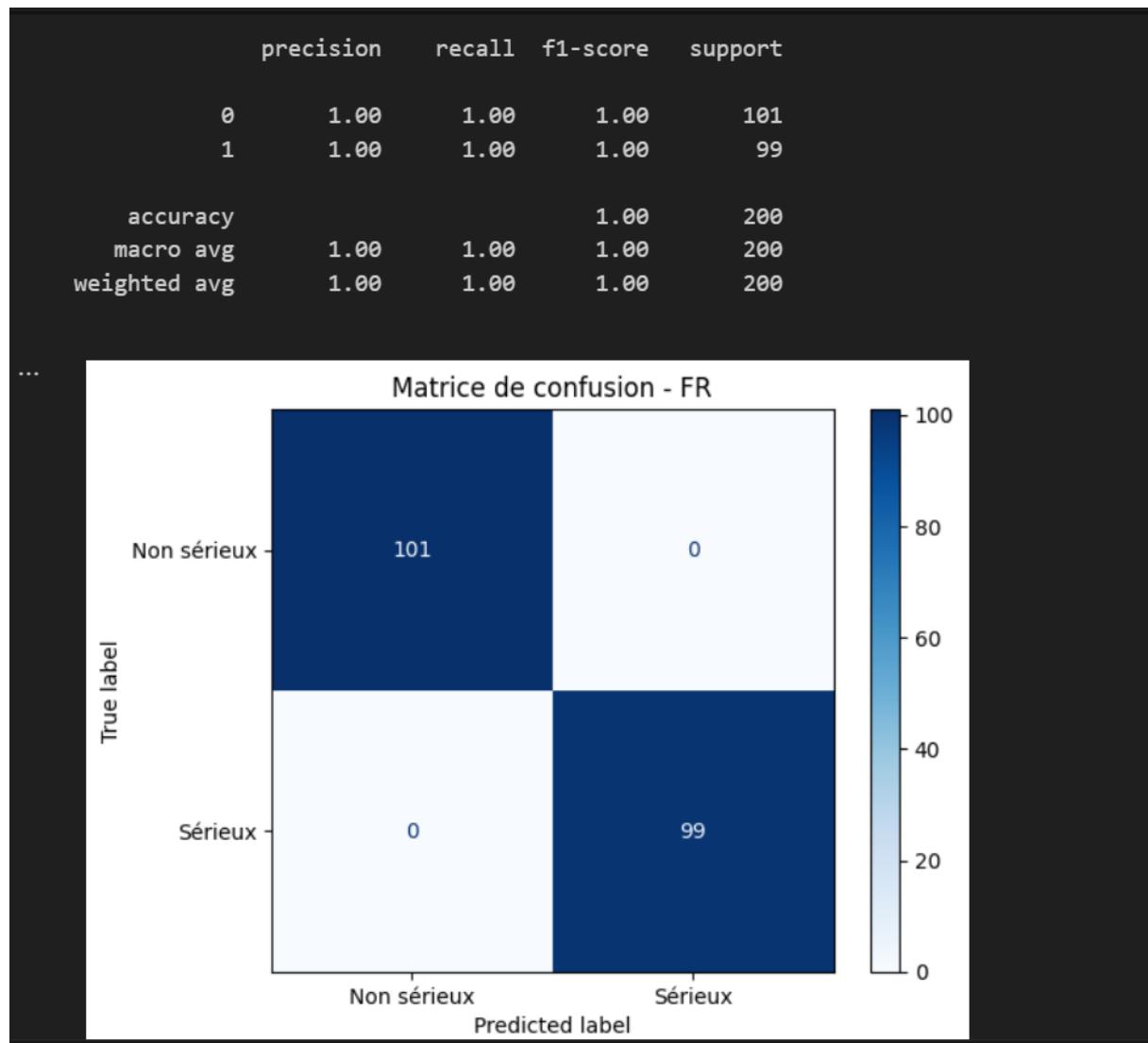
- le message est envoyé au modèle français si la langue est `fr`,
- ou au modèle anglais si la langue détectée est `en`.

## 5.2 Résultats et évaluation des modèles de filtrage

Après entraînement, les deux modèles (français et anglais) ont été testés sur des jeux de données distincts. Chaque modèle devait classer les commentaires comme **sérieux** ou **non sérieux**. Les résultats ont été mesurés à l'aide de deux outils classiques en apprentissage automatique : la **matrice de confusion** et le **rapport de classification** (précision, rappel, F1-score).

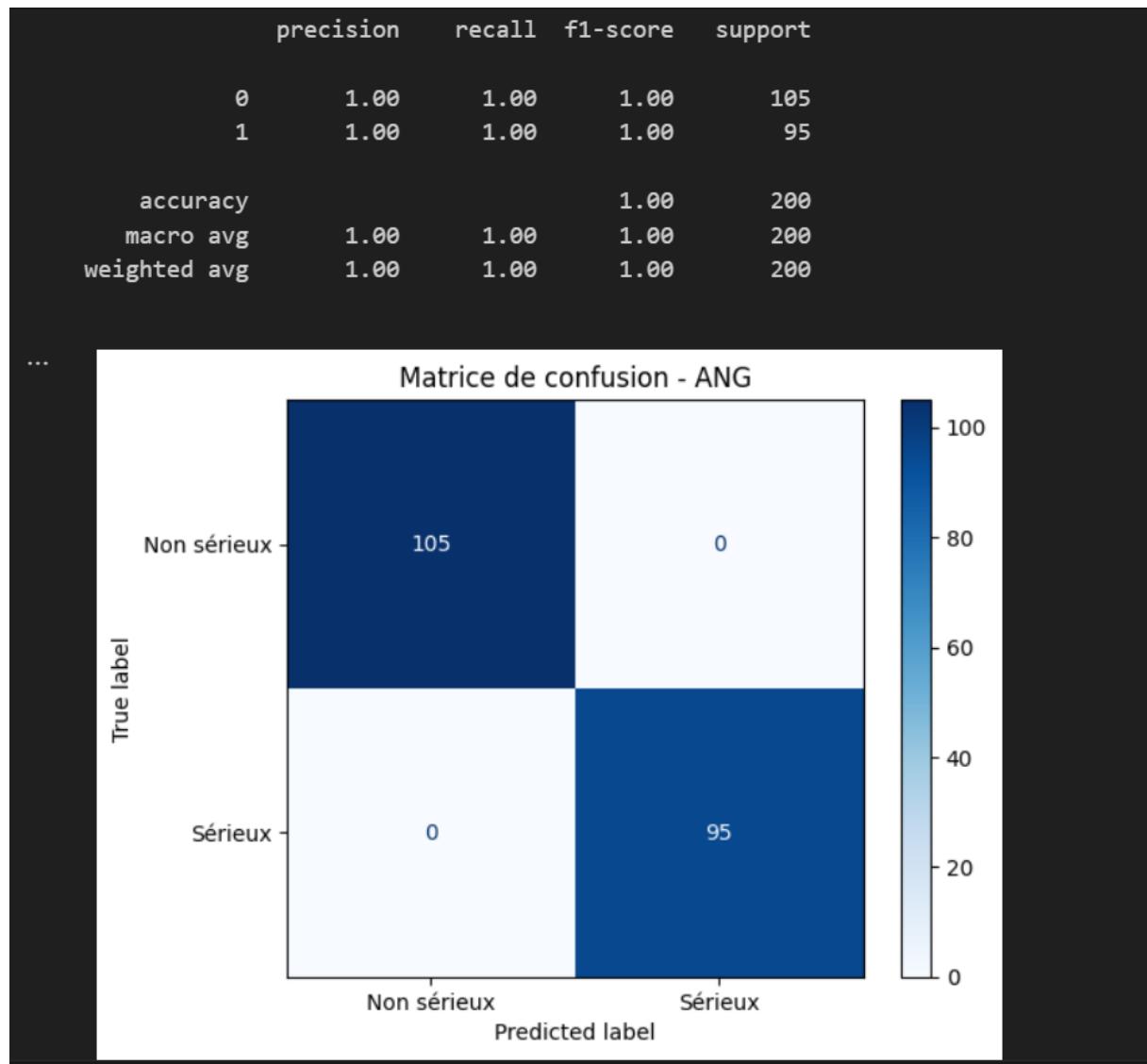
### 5.2.1 Modèle français

Le modèle entraîné sur les commentaires en langue française a obtenu un taux de précision de 1. Tous les commentaires ont été classés correctement, sans aucune confusion entre les classes.



### 5.2.2 Modèle anglais

Le modèle appliqué aux commentaires en anglais a lui aussi obtenu un score parfait de 1. Les prédictions sur les feedbacks rédigés en anglais sont donc tout aussi fiables.



Ces résultats montrent que la détection de commentaires non sérieux fonctionne parfaitement dans les deux langues. Cela permet au chatbot de filtrer efficacement les messages sans valeur ajoutée et de ne conserver que les retours utiles pour l'analyse pédagogique.

## 5.3 Déroulement de l'interaction

L'interaction suit une séquence logique en deux étapes principales :

### 5.3.1 1. Collecte du feedback

Le chatbot initie la conversation avec une question ouverte du type : « *Quelles sont les raisons pour lesquelles tu donnes ce feedback négatif ?* » L'étudiant saisit sa réponse dans une zone de texte libre.

### 5.3.2 2. Analyse

Le message est transmis automatiquement au serveur (Django), qui effectue les opérations suivantes :

- Détection automatique de la langue (français ou anglais) via la bibliothèque `langdetect`.
- Sélection du modèle de classification approprié (français ou anglais).
- Prédiction de la pertinence du message : **sérieux (1)** ou **non sérieux (0)**.
- Retour du résultat au client (interface).

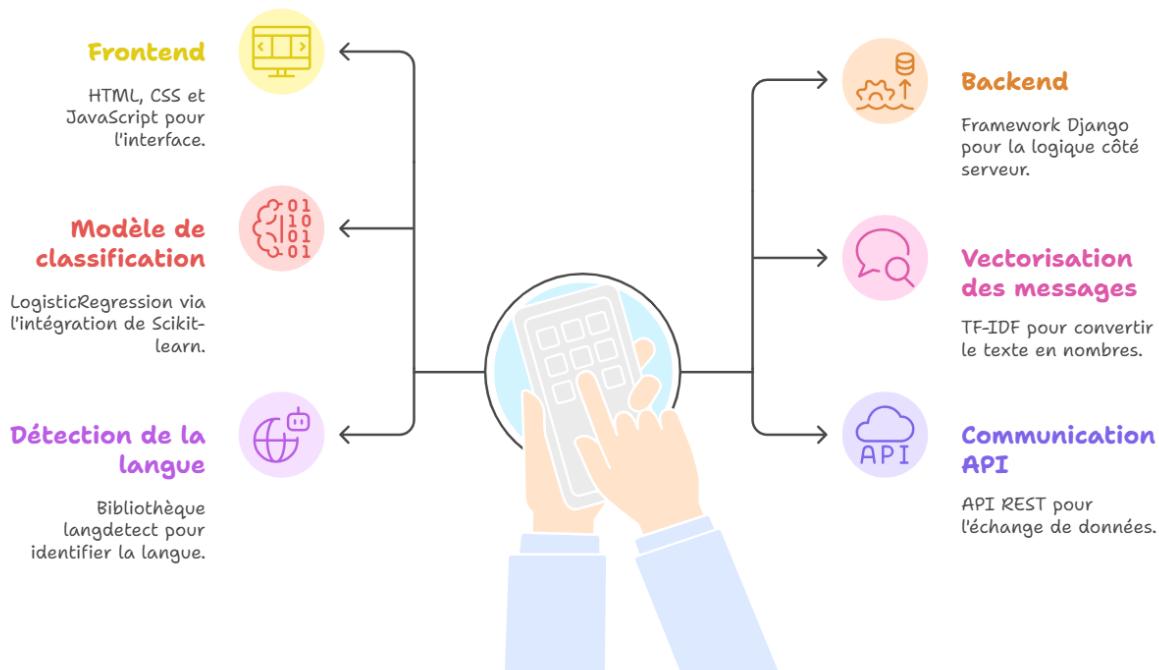
### 3. Réponse personnalisée

Selon le résultat de l'analyse, le chatbot affiche l'un des messages suivants :

- **Ce feedback est considéré comme sérieux.**
- **Ce feedback est considéré comme non sérieux.**

Un message final remercie ensuite l'étudiant pour sa participation.

## 5.4 Architecture technique



### 5.5 Intérêt

Ce chatbot intelligent contribue à :

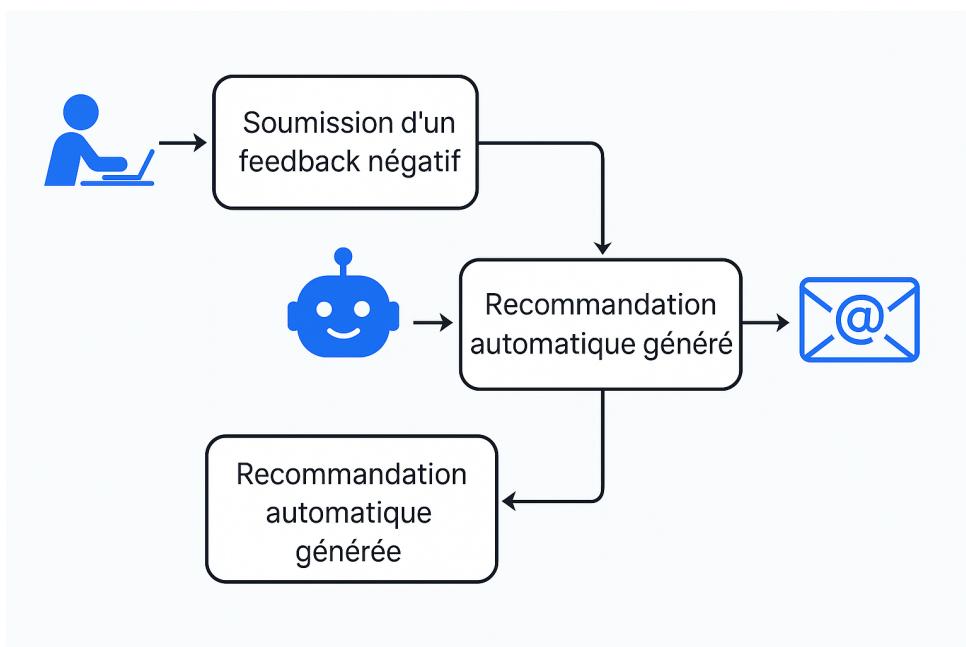
- Capter plus facilement des retours étudiants authentiques et spontanés.
- Éliminer automatiquement les messages non pertinents ou hors contexte.
- Fournir à l'enseignant un indicateur automatisé de la qualité des feedbacks reçus.

Il représente ainsi un outil moderne d'aide à la décision, en améliorant la fiabilité des retours et en facilitant leur exploitation pédagogique.

# 6. Génération automatique de recommandations par IA

## 6.1 Idée générale de la recommandation IA

Lorsqu'un étudiant soumet un feedback négatif sur un cours, et que ce dernier est validé par notre chatbot, une recommandation automatique est générée et envoyée par e-mail à l'enseignant responsable du cours concerné. Cette recommandation a pour objectif de fournir des suggestions d'amélioration basées sur l'analyse du feedback. Par exemple, si plusieurs étudiants signalent un manque de clarté dans les explications, l'enseignant recevra une suggestion d'utiliser davantage d'exemples concrets ou d'adapter le rythme du cours



## 6.2 conception de la réalisation de la recommandation IA

### 6.2.1 Dataset de la recommandation

L'idée est de générer un dataset à partir des données de feedbacks déjà collectées et traitées dans la partie dédiée aux composants de machine learning. Ce dataset servira à entraîner un modèle capable de produire automatiquement des recommandations aux enseignants.

La démarche consiste à extraire les mots-clés principaux de chaque feedback à l'aide d'un modèle de classification ou de règles simples, puis à associer chaque ensemble de mots-clés à une recommandation correspondante. Cela permet de construire un jeu de données supervisé pour l'entraînement d'un système de recommandation intelligent.

```
theme_keywords = {
    "workload": ["assignment", "project", "exam", "deadline"],
    "course content": ["material", "content", "outdated", "updated"],
    "teacher": ["teacher", "professor", "helpful", "supportive"],
    "organization": ["schedule", "organization", "time", "planning"]
}
df1['main_keywords'] = df1['tokens'].apply(lambda tokens: [word for word in tokens if word in sum(theme_keywords.values(), [])])

def assign_theme(keywords):
    for theme, words in theme_keywords.items():
        if any(word in keywords for word in words):
            return theme
    return "other"

df1['theme'] = df1['main_keywords'].apply(assign_theme)
```

Voici un exemple de la structure du dataset obtenu :

comment	theme
I didn't learn much from this course..	other
Everything was clear and easy to follow..	other
"Not bad, but not great either."	other
The professor was not responsive to questions..	teacher
I appreciated the detailed feedback on assignments..	workload
"Average experience, nothing special."	other
The professor was not responsive to questions..	teacher
Everything was clear and easy to follow..	other
Everything was clear and easy to follow..	other
I struggled a lot due to the fast pace..	other
"I learned some new things, but it could be better..	other
Didn't feel supported throughout the semester..	other
Class discussions were thought-provoking..	other
"Average experience, nothing special."	other
The instructor is knowledgeable and enthusiastic..	other
"The professor was decent, but not exceptional."	teacher
"Average experience, nothing special."	other
Assignments were irrelevant and poorly explained..	workload
Assignments were useful and relevant to the course..	workload

## 6.2.2 Modèle entraîné de la recommandation

Le modèle de classification automatique utilise une pipeline de machine learning pour prédire le thème d'un commentaire étudiant (feedback). Il commence par une étape de **prétraitement linguistique** à l'aide d'un composant personnalisé appelé `SpacyPreprocessor`, qui nettoie et normalise le texte. Ensuite, les commentaires textuels sont transformés en vecteurs numériques à l'aide de la méthode **TF-IDF** (`TfidfVectorizer`), en prenant en compte des **unigrammes** et **bigrammes** (mots seuls et paires de mots), avec une limite de 5000 caractéristiques.

Ces vecteurs sont ensuite utilisés pour entraîner un **modèle de régression logistique** (`LogisticRegression`) qui permet de classifier chaque commentaire dans l'un des thèmes prédefinis (par exemple : charge de travail, contenu du cours, enseignant, etc.). Le modèle est entraîné sur un sous-ensemble des données obtenu par séparation des jeux d'entraînement et de test (`train_test_split`). Ce processus permet d'automatiser l'analyse des feedbacks afin d'en extraire leur thématique principale et de servir de base à un système de recommandation intelligent.

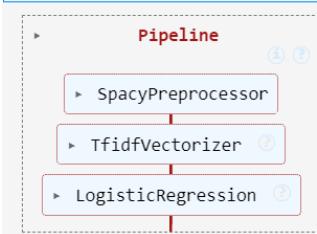
```
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

X = df['comment']
y = df['theme']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

pipeline = Pipeline([
    ('preprocess', SpacyPreprocessor()),
    ('tfidf', TfidfVectorizer(max_features=5000, ngram_range=(1, 2))),
    ('clf', LogisticRegression(max_iter=1000, class_weight='balanced'))
])

pipeline.fit(X_train, y_train)
```



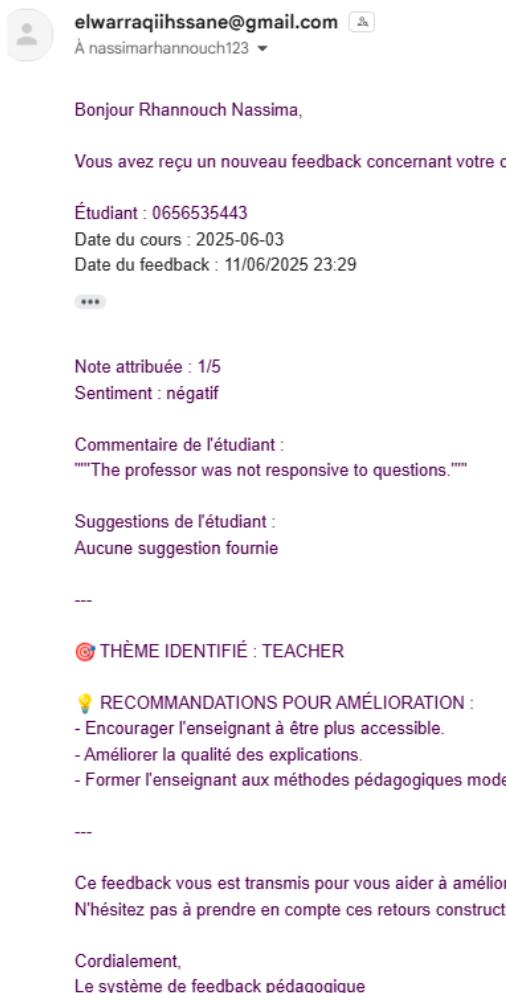
### 6.2.3 Intégration dans Django : génération des recommandations et envoi automatique par mail

Dans le framework **Django**, nous avons intégré le modèle de classification entraîné, capable de prédire le thème principal d'un feedback soumis par un étudiant.

Lorsqu'un feedback négatif est soumis par un étudiant via l'espace étudiant, il est d'abord vérifié et validé par notre **chatbot intelligent**. Une fois cette validation effectuée, le système procède de manière automatique comme suit :

- Le feedback est transmis au modèle, qui prédit le thème correspondant (par exemple : contenu du cours, rythme, communication, etc.).
- Une fonction personnalisée génère alors une **recommandation** spécifique en fonction du thème prédit.
- Cette recommandation est ensuite automatiquement envoyée par **mail** à l'enseignant responsable du cours concerné.

**Exemple de mail envoyé à l'enseignant :**



# 7. Implémentation

---

## 7.1 Développement des interfaces

Les interfaces développées dans cette partie sont les tableaux de bord (dashboards) destinés aux trois types d'utilisateurs principaux de notre application *FeedbackFlow* : l'étudiant, le chef de filière et l'administrateur. Ces dashboards ont été créés à l'aide de la technologie **Dash** de *Python*, qui permet la création d'interfaces web interactives et dynamiques adaptées à l'analyse de données.

### 7.1.1 Interface Étudiant / Dashboard

Le tableau de bord étudiant de l'application **FeedBackFlow** offre une vue complète et interactive sur les performances académiques d'un étudiant de la filière IID. Il commence par une section d'accueil personnalisée affichant l'identifiant de l'étudiant, sa filière et sa promotion. Trois cartes récapitulatives présentent : la moyenne générale de l'étudiant (15.63/20), la moyenne de la classe (15.61/20), ainsi que le nombre de cours suivis durant l'année (4). Une distinction claire est faite entre les performances personnelles et celles du groupe.

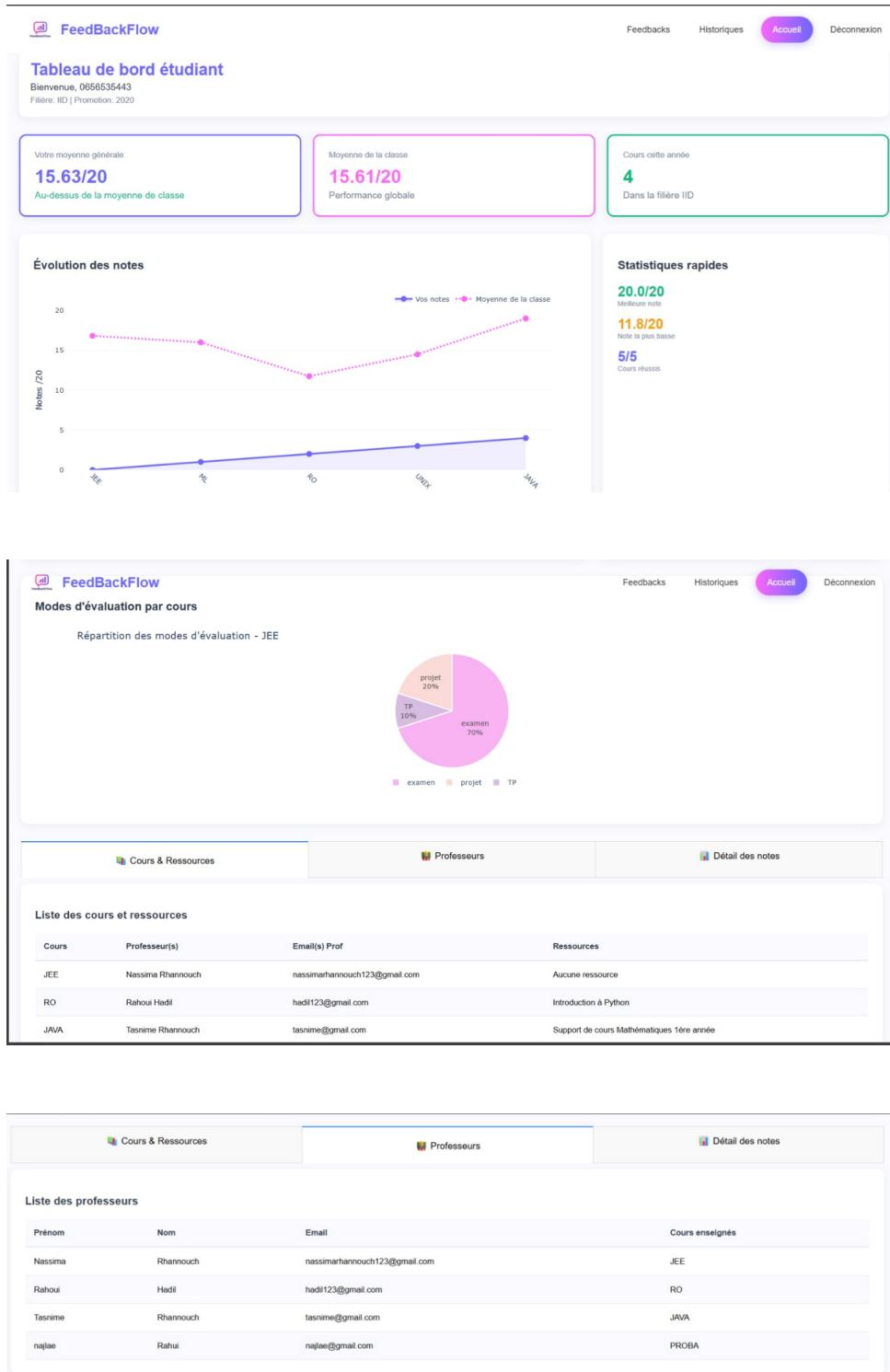
Le tableau de bord comprend également une courbe d'évolution des notes par matière (JEE, ML, RO, UNIX, JAVA), comparant les résultats de l'étudiant à la moyenne de la classe. Cela permet de suivre visuellement la progression académique et d'identifier les points faibles ou forts.

Sur la droite, une section de *Statistiques rapides* indique la meilleure note obtenue (20.0/20), la note la plus basse (11.8/20) et le nombre total de cours validés (5/5), offrant ainsi un résumé synthétique de la performance globale.

Une autre partie du tableau de bord est dédiée à la répartition des modes d'évaluation par cours, illustrée ici par un diagramme circulaire pour le module JEE : 70% d'examen, 20% de projet et 10% de travaux pratiques. Cela aide l'étudiant à comprendre le poids de chaque composante dans sa note finale.

Enfin, des onglets interactifs permettent d'accéder à la liste des cours et ressources, des professeurs, et le détail des notes. Chaque cours est associé à un ou plusieurs professeurs, dont le nom et l'adresse e-mail sont fournis, ainsi que les ressources pédagogiques disponibles.

Ce tableau de bord propose donc une interface conviviale, riche en données, facilitant le suivi académique et l'accès aux informations clés pour l'étudiant.



### 7.1.2 Interface Chef de filière / Dashboard

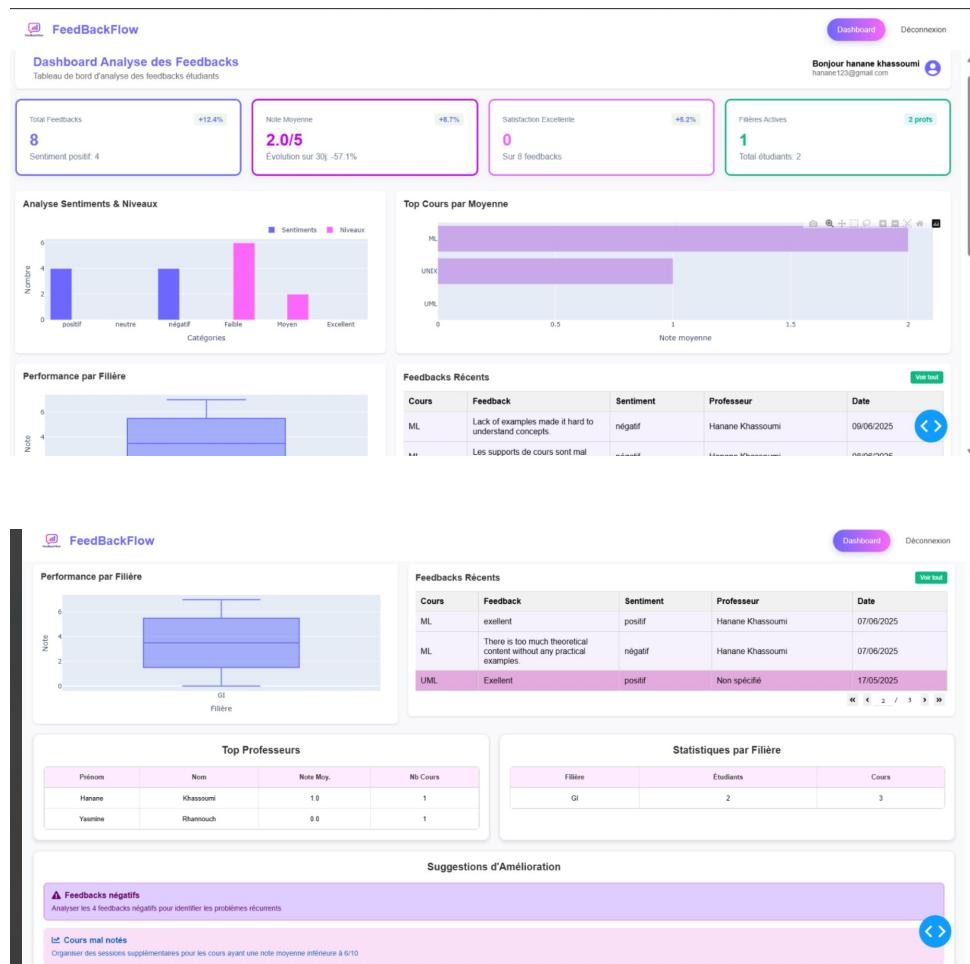
Le tableau de bord du chef de filière dans l'application **FeedBackFlow** offre une vue d'ensemble détaillée sur les performances des étudiants de sa filière. Il commence par un en-tête personnalisé affichant le nom de la filière (IID) et le nom du chef de filière. La partie centrale du tableau de bord présente des informations clés sous forme de cartes récapitulatives :

- Le nombre total de feedbacks reçus (+12.4)

- La note moyenne des étudiants (2.0/5), avec l'évolution sur les 30 derniers jours (-57.1)
- Le niveau de satisfaction des étudiants (Excellent, +5.2)
- Le nombre d'étudiants actifs (2)

Un graphique d'analyse des sentiments et des niveaux des feedbacks permet de visualiser la répartition des commentaires positifs, neutres et négatifs, ainsi que leur niveau (faible, moyen, excellent). Cela aide le chef de filière à identifier les points à améliorer. La section "Top Cours par Moyenne" présente les moyennes des différents cours de la filière (ML, UNIX, UML), permettant de repérer les enseignements les mieux notés. Enfin, la partie "Feedbacks Récents" détaille les derniers commentaires reçus, avec le cours concerné, le sentiment exprimé (positif ou négatif) et le nom du professeur évalué. Cela offre une vision concrète des retours des étudiants. Des fonctionnalités supplémentaires, telles que l'accès à la liste des étudiants, des professeurs et des statistiques plus approfondies, sont accessibles via des onglets dédiés.

Ce tableau de bord offre ainsi au chef de filière une interface complète et intuitive pour suivre les performances de sa filière, analyser les feedbacks des étudiants et prendre les mesures nécessaires pour améliorer la qualité de l'enseignement.



### 7.1.3 Interface Administrateur / Dashboard

Le tableau de bord administrateur de l'application **FeedBackFlow** offre une vue d'ensemble détaillée du système d'analyse des feedbacks étudiants pour la filière IID.

En haut de l'interface, l'administrateur peut sélectionner la filière à analyser à l'aide d'un menu déroulant. Cette fonctionnalité permet de basculer facilement entre les différentes filières gérées.

La partie centrale du tableau de bord présente des informations clés sous forme de cartes récapitulatives :

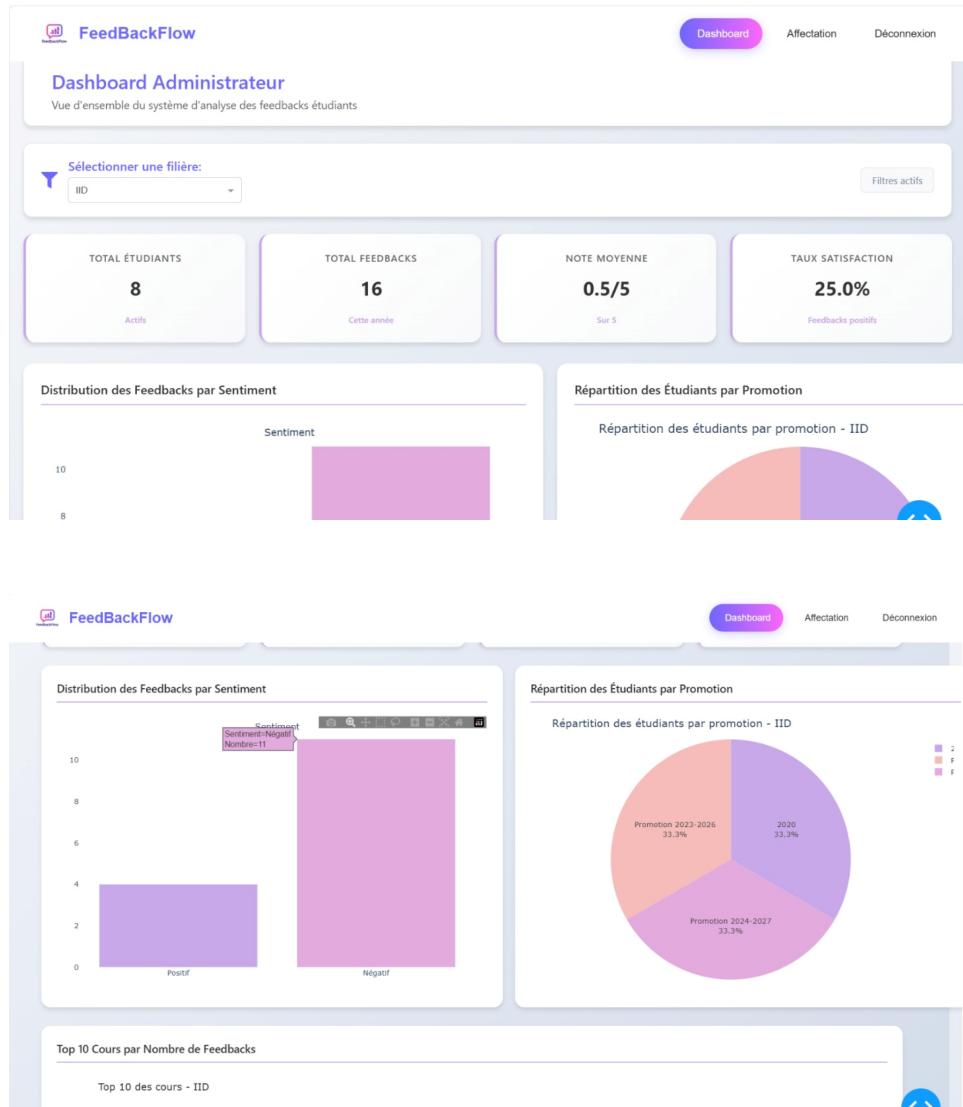
- Le nombre total d'étudiants actifs (8)
- Le nombre total de feedbacks reçus (16) pour l'année en cours
- La note moyenne des étudiants (0.5/5)
- Le taux de satisfaction des étudiants (25.0)

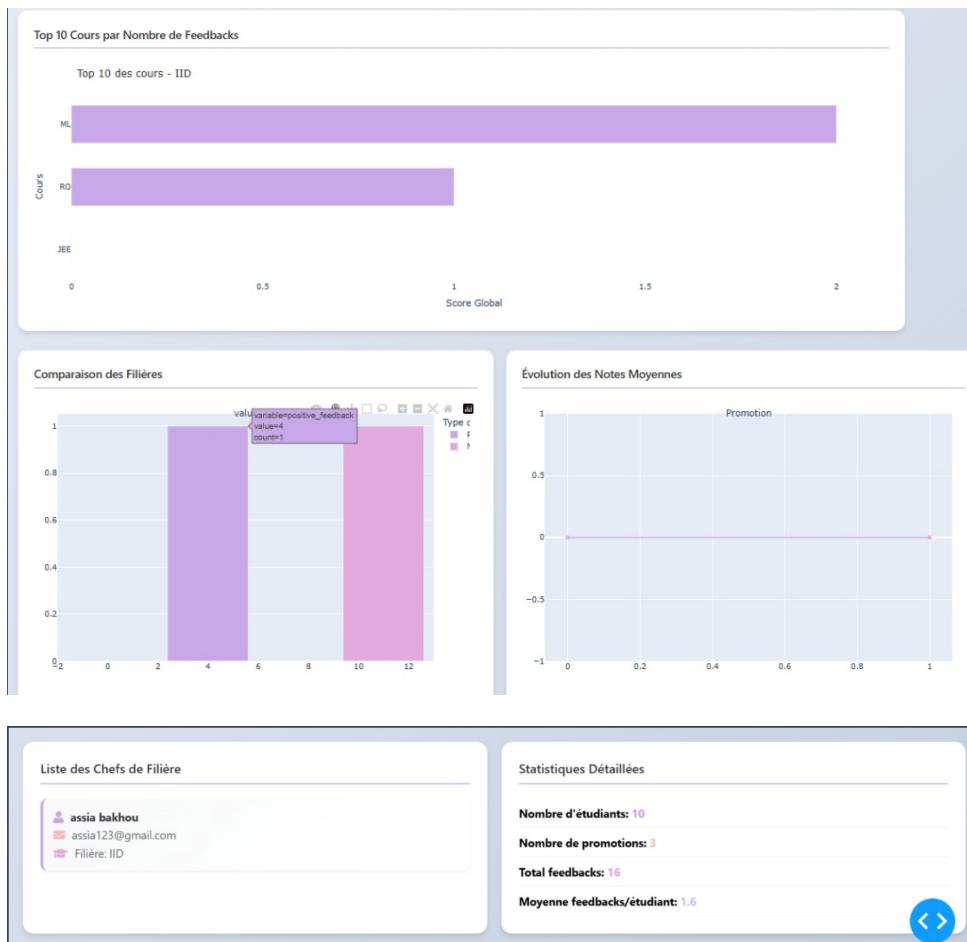
Deux graphiques détaillent la répartition des feedbacks par sentiment (positif, négatif) et la répartition des étudiants par promotion (2020, 2023-2026, 2024-2027). Ces visualisations permettent à l'administrateur d'identifier les tendances et les points à améliorer.

Une section "Top 10 Cours par Nombre de Feedbacks" présente les cours les plus commentés, avec leur score global moyen. Cela aide à cibler les enseignements qui nécessitent une attention particulière.

Enfin, des fonctionnalités supplémentaires, telles que l'accès à la liste détaillée des chefs de filière, sont accessibles via des onglets dédiés.

Ce tableau de bord offre ainsi à l'administrateur une interface complète et intuitive pour suivre les performances globales de la filière, analyser les feedbacks des étudiants et prendre les mesures nécessaires pour améliorer la qualité de l'enseignement.





## 7.2 Développement Back-End avec Django

Dans cette section, nous présentons l'implémentation du back-end de notre application à l'aide du framework Django. Le back-end gère la logique métier, la gestion des données et les interactions avec la base de données.

### 7.2.1 Modélisation des données

Nous avons défini plusieurs modèles Django pour représenter les entités principales du système, tels que les utilisateurs, les filières, les promotions, les cours, les professeurs, les feedbacks et les notes.

**Modèle Utilisateur :** Ce modèle étend la classe `AbstractUser` pour personnaliser les informations des utilisateurs avec des champs supplémentaires tels que l'email, le téléphone, le rôle, la filière et la promotion.

```

1 class Utilisateur(AbstractUser):
2     email = models.EmailField(unique=True)
3     telephone = models.CharField(max_length=15, null=True, blank=True)
4
5     ROLES = (
6         ('etudiant', 'tudiant'),

```

```

7     ('chef_filiere', 'Chef de Filière'),
8     ('administrateur', 'Administrateur'),
9 )
10    role = models.CharField(max_length=20, choices=ROLES, default
11      ='etudiant')
12    filiere = models.ForeignKey('Filiere', on_delete=models.
13      SET_NULL, null=True, blank=True, related_name='utilisateurs'
14      )
15    promotion = models.ForeignKey('Promotion', on_delete=models.
16      SET_NULL, null=True, blank=True, related_name='utilisateurs'
17      )
18
19    USERNAME_FIELD = 'username'
20    REQUIRED_FIELDS = ['email']
21
22    def __str__(self):
23        return f'{self.username} ({self.get_role_display()})'

```

**Modèle Filiere** : Ce modèle représente une filière de formation, associée à un chef de filière et à plusieurs étudiants.

```

1 class Filiere(models.Model):
2     nom = models.CharField(max_length=100)
3     description = models.TextField(blank=True, null=True)
4
5     def __str__(self):
6         return self.nom
7
8     @property
9     def chef(self):
10         return self.utilisateurs.filter(role='chef_filiere').
11             first()
12
13     @chef.setter
14     def chef(self, utilisateur):
15         # Retirer l'ancien chef
16         anciens_chefs = self.utilisateurs.filter(role='
17             chef_filiere')
18         for ancien in anciens_chefs:
19             ancien.role = 'etudiant'
20             ancien.save()
21             utilisateur.role = 'chef_filiere'
22             utilisateur.filiere = self
23             utilisateur.save()
24
25     @property
26     def nombre_etudiants(self):
27         return self.utilisateurs.filter(role='etudiant').count()

```

**Modèle Promotion** : Ce modèle représente une promotion d'étudiants avec un nom et les années de début et de fin.

```

1 class Promotion(models.Model):
2     nom = models.CharField(max_length=100)
3     annee_debut = models.IntegerField(null=True, blank=True)
4     annee_fin = models.IntegerField(null=True, blank=True)
5
6     def __str__(self):
7         return self.nom

```

**Modèle cours** : représente un cours lié à une filière. Il inclut une méthode calculant la note moyenne basée sur les feedbacks associés.

```

1 class Cours(models.Model):
2     nom = models.CharField(max_length=100)
3     filiere = models.ForeignKey(Filiere, on_delete=models.CASCADE
4                                 , related_name='cours')
5
6     class Meta:
7         verbose_name = "Cours"
8         verbose_name_plural = "Cours"
9
10    def __str__(self):
11        return f"{self.nom} ({self.filiere.nom})"
12
13    @property
14    def note_moyenne(self):
15        """Calcule la note moyenne du cours."""
16        feedbacks = self.feedbacks.all()
17        if not feedbacks:
18            return None
19        return sum(f.note for f in feedbacks) / feedbacks.count()

```

**Modèle professeur** : Ce modèle représente un professeur, avec ses informations personnelles et les cours qu'il enseigne. Il fournit aussi une méthode pour calculer la note moyenne issue des feedbacks sur ses cours.

```

1 class Professeur(models.Model):
2     nom = models.CharField(max_length=100)
3     prenom = models.CharField(max_length=100)
4     email = models.EmailField(unique=True)
5     telephone = models.CharField(max_length=15, null=True, blank=True)
6     cours = models.ManyToManyField(Cours, related_name='professeurs')
7
8     class Meta:
9         verbose_name = "Professeur"
10        verbose_name_plural = "Professeurs"
11
12    def __str__(self):
13        return f"{self.prenom} {self.nom}"
14

```

```

15 @property
16     def cours_enseignes(self):
17         """Retourne la liste des cours enseignés par ce
18         professeur."""
19         return self.cours.all()
20
21 @property
22     def note_moyenne(self):
23         """Calcule la note moyenne du professeur      partir des
24         feedbacks de tous ses cours."""
25         feedbacks = Feedback.objects.filter(cours__in=self.cours.
26         all())
27         if not feedbacks.exists():
28             return None
29         return feedbacks.aggregate(models.Avg('note'))['note_avg']
30

```

**Modèle Feedback :** Le modèle Feedback permet aux étudiants de laisser un retour sur un cours et/ou un professeur, avec un système d'analyse de sentiment, anonymat possible, et recommandations.

```

1 class Feedback(models.Model):
2     cours = models.ForeignKey(Cours, on_delete=models.CASCADE,
3                               related_name='feedbacks')
4     professeur = models.ForeignKey(Professeur, on_delete=models.
5                                    CASCADE, null=True, blank=True, related_name='feedbacks')
6     etudiant = models.ForeignKey(Utilisateur, on_delete=models.
7                                   SET_NULL, null=True, blank=True, related_name='feedbacks')
8     note = models.IntegerField()
9     commentaire = models.TextField()
10    suggestions = models.TextField(blank=True, null=True)
11    date_cours = models.DateField(null=True, blank=True)
12    date_creation = models.DateTimeField(auto_now_add=True)
13    sentiment = models.CharField(max_length=10, blank=True, null=
14        True, choices=[
15            ('positif', 'Positif'),
16            ('neutre', 'Neutre'),
17            ('n_gatif', 'N_gatif')])
18    anonyme = models.BooleanField(default=False)
19    partager = models.BooleanField(default=False)
20    theme_pred = models.CharField(max_length=100, blank=True,
21        null=True) # theme predict
21    recommendations = models.TextField(blank=True, null=True)
22
23    class Meta:
24        verbose_name = "Feedback"
25        verbose_name_plural = "Feedbacks"
26        ordering = ['-date_creation']
27
28    def __str__(self):

```

```

26     cours_nom = self.cours.nom if self.cours else "Cours
27         inconnu"
28     etudiant_str = "Anonyme" if self.anonyme else (self.
29         etudiant.username if self.etudiant else "Inconnu")
30     return f"Feedback #{self.id} - {cours_nom} par {
31         etudiant_str}"

```

**Modèle CourseResource** :Ce modèle stocke les ressources pédagogiques associées à un cours, avec un lien et une description.

```

1 class CourseResource(models.Model):
2     course_name = models.CharField(max_length=100)
3     resource_link = models.URLField()
4     description = models.TextField()
5
6     def __str__(self):
7         return f"{self.description} ({self.course_name})"

```

**Modèle Notes** :Modèle pour enregistrer les notes des étudiants par cours, avec contrainte d'unicité étudiant-cours.

```

1 class Notes(models.Model):
2     etudiant = models.ForeignKey('Utilisateur', on_delete=models.
3         CASCADE, related_name='notes')
4     cours = models.ForeignKey('Cours', on_delete=models.CASCADE,
5         related_name='notes')
6     note = models.DecimalField(max_digits=4, decimal_places=2)
7     date_creation = models.DateTimeField(auto_now_add=True)
8
9     class Meta:
10         verbose_name = "Note"
11         verbose_name_plural = "Notes"
12         unique_together = ('etudiant', 'cours')
13
14     def __str__(self):
15         return f"Note de {self.etudiant.username} pour {self.
16             cours.nom}: {self.note}/20"

```

**Modèle ModeEvaluation** :Ce modèle définit les différentes modalités d'évaluation d'un cours (examen, projet, TP, etc.) et leur pondération dans la note finale.

```

1 class ModeEvaluation(models.Model):
2     cours = models.ForeignKey('Cours', on_delete=models.CASCADE,
3         related_name='modes_evaluation')
4     mode = models.CharField(max_length=100) # Examen, Projet, TP
5         , etc.
6     pourcentage = models.IntegerField() # Pourcentage de la note
7         finale
8
9     class Meta:
10         verbose_name = "Mode d' valuation "
11         verbose_name_plural = "Modes d' valuation "

```

```

9
10     def __str__(self):
11         return f"{self.mode} ({self.pourcentage}%) - {self.cours.
12             nom}"

```

Ces modèles sont définis dans le fichier `models.py` et sont automatiquement mappés aux tables correspondantes dans la base de données MySQL via le système de migrations de Django.

### 7.2.2 Interface d'administration Django

Pour faciliter la gestion des données et permettre aux administrateurs du système d'effectuer les opérations CRUD sur les différentes entités, nous avons utilisé l'interface d'administration intégrée de Django. Cette interface fournit un panneau d'administration puissant, personnalisable et sécurisé. Chaque modèle principal (Utilisateur, Filière, Cours,

Professeur, Feedback, Notes, ModeEvaluation) a été enregistré dans le fichier `admin.py` avec une configuration adaptée pour afficher les champs pertinents, faciliter la recherche et le filtrage, et permettre une navigation aisée.

The screenshot shows the Django administration interface. At the top, there's a header bar with "Django administration" on the left and "WELCOME, HP. VIEW SITE / CHANGE PASSWORD / LOG OUT" on the right. Below the header, the main content area has a dark blue sidebar on the left containing links to various models: Cours, Course resources, Feedbacks, Filières, Modes d'évaluation, Notes, Professeurs, Promotions, and Users. Each link has a "+ Add" button and a "Change" link. To the right of the sidebar, there are two light gray boxes: "Recent actions" (empty) and "My actions" (also empty). The main content area is divided into three horizontal sections: "APPCOURS" (containing the list of models), "AUTHENTICATION AND AUTHORIZATION" (containing a single "Groups" entry), and "DJANGO PLOTLY DASH" (containing "Dash apps" and "Stateless apps"). Each section has a "+ Add" button and a "Change" link.

Grâce à cette interface, les administrateurs peuvent :

- **Créer** de nouveaux utilisateurs, filières, cours, professeurs, feedbacks, notes et modes d'évaluation.
- **Lire** et afficher la liste des enregistrements avec des possibilités de filtrage, recherche et tri.
- **Mettre à jour** les informations existantes en accédant aux formulaires d'édition.
- **Supprimer** des enregistrements via une interface sécurisée avec confirmation.

Cette interface est accessible uniquement aux utilisateurs ayant les permissions administratives et assure ainsi la sécurité et la cohérence des données. Nous avons également personnalisé certaines pages d'administration pour afficher des informations calculées, comme

la moyenne des notes d'un cours ou d'un professeur, facilitant ainsi le suivi pédagogique directement depuis l'interface admin.

The screenshot shows the Django admin interface for the 'Filières' model. The left sidebar has a 'Start typing to filter...' input field and lists several models: Cours, Course resources, Feedbacks, Filières (which is selected and highlighted in yellow), Modes d'évaluation, Notes, Professeurs, Promotions, and Users. The right panel title is 'Select filière to change'. It includes a search bar with a magnifying glass icon and a 'Search' button. Below the search is an 'Action:' dropdown set to '-----' and a 'Go' button. A message '0 of 2 selected' is displayed. A table lists two filières: 'GI' with 'geni info' as the chef, 'hanane (Chef de Filière)' as the chef, and 2 students; and 'IID' with 'informatique et ingenierie des donnee' as the chef, 'atoria (Chef de Filière)' as the chef, and 6 students. A total of '2 filières' is indicated at the bottom. A 'ADD FILIERE +' button is located in the top right corner of the main content area.

	NOM	DESCRIPTION	CHEF	NOMBRE ETUDIANTS
<input type="checkbox"/>	GI	geni info	hanane (Chef de Filière)	2
<input type="checkbox"/>	IID	informatique et ingenierie des donnee	atoria (Chef de Filière)	6

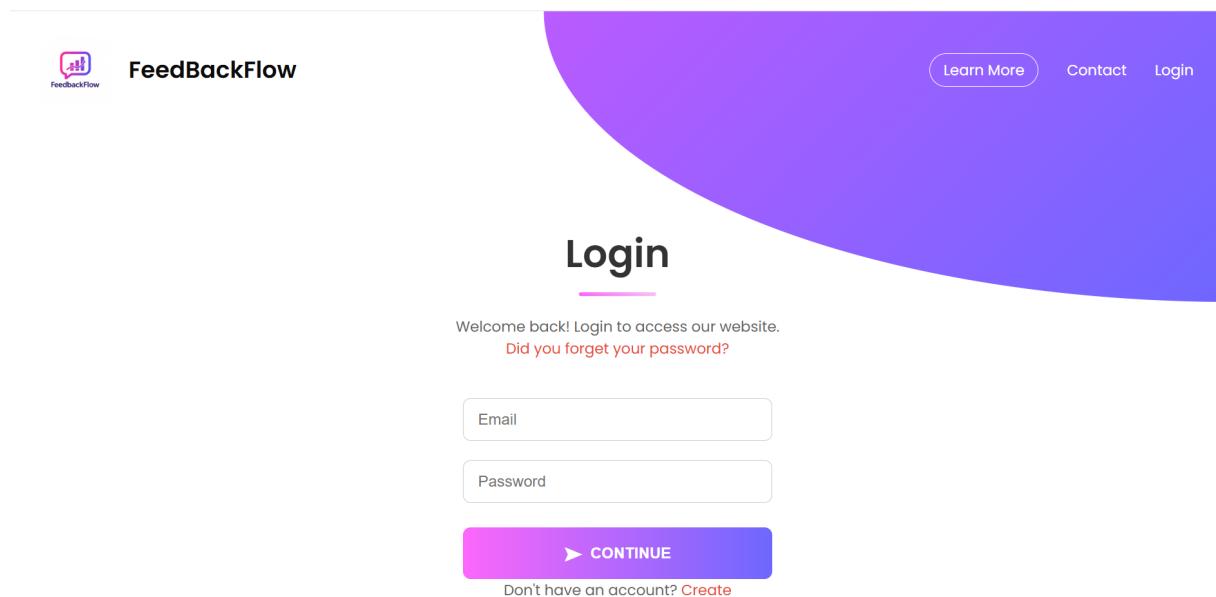
# 8. Tests et évaluation

## 8.1 Tests fonctionnels

L'objectif de cette phase est de valider les fonctionnalités principales de l'application selon les exigences du cahier des charges. Voici un résumé des tests effectués :

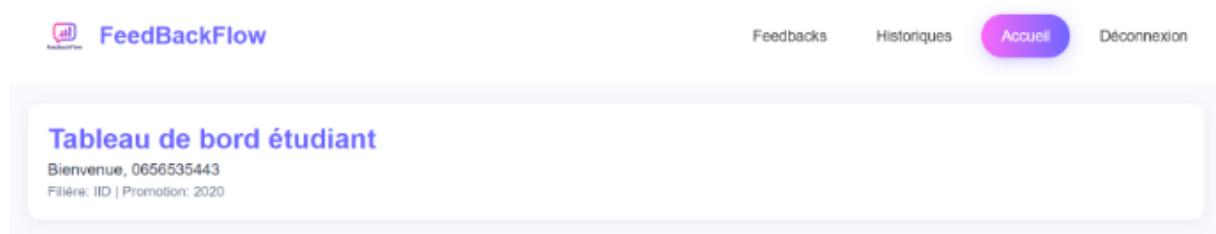
### 1. Authentification multi-rôle

- **Fonction testée :** Connexion avec un seul formulaire pour les rôles (admin, étudiant, chef de filière).



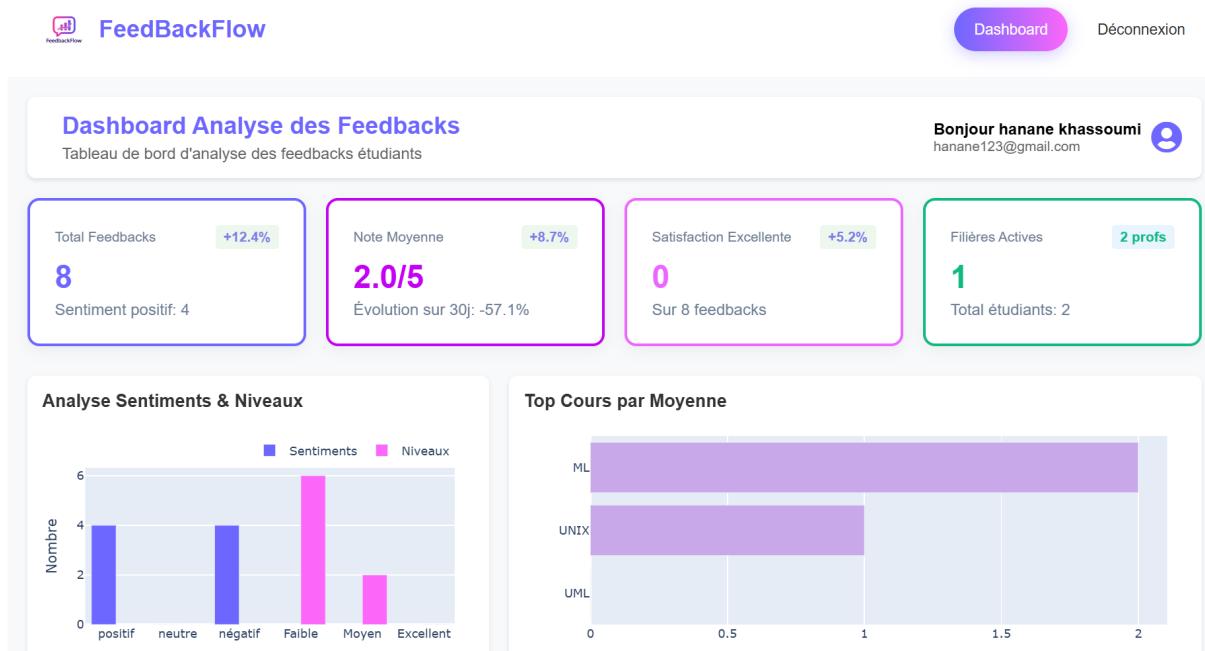
The screenshot shows the login page of the FeedBackFlow application. At the top left is the logo 'FeedBackFlow'. At the top right are three buttons: 'Learn More', 'Contact', and 'Login'. The main title 'Login' is centered above a horizontal line. Below the line, the text 'Welcome back! Login to access our website.' is displayed, followed by a red link 'Did you forget your password?'. There are two input fields: 'Email' and 'Password'. A large purple button labeled '► CONTINUE' is positioned below the inputs. At the bottom, a small note says 'Don't have an account? [Create](#)'.

- **Résultat :** Après authentification, l'utilisateur est redirigé vers l'interface correspondant à son rôle.  
étudiant :

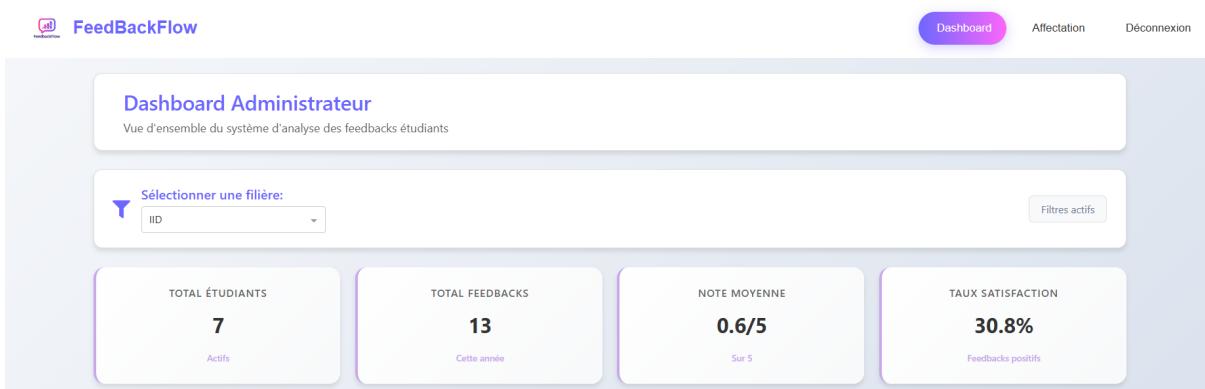


The screenshot shows the student dashboard ('Tableau de bord étudiant'). At the top left is the logo 'FeedBackFlow'. At the top right are five buttons: 'Feedbacks', 'Historiques', 'Accueil' (which is highlighted in pink), and 'Déconnexion'. The main area has a light gray background and displays the text 'Bienvenue, 0656535443' and 'Filière: IID | Promotion: 2020'.

chef de filière :



admin :



— Statut : Réussi.

## 2. Interface Étudiant

— **Soumission de feedback :** Le formulaire permet à l'étudiant de soumettre un feedback lié à un cours ou professeur.

 **Nassima**

Cours concerné :

Enseignant :

Date du cours :  

Note : 

Votre feedback détaillé :  
Décrivez ce que vous avez apprécié ou ce qui peut être amélioré...

Suggestions d'amélioration :  
Avez-vous des idées concrètes pour améliorer ce cours ?

Options supplémentaires :

Soumettre ce feedback de façon anonyme  
 Partager ce feedback avec les autres étudiants

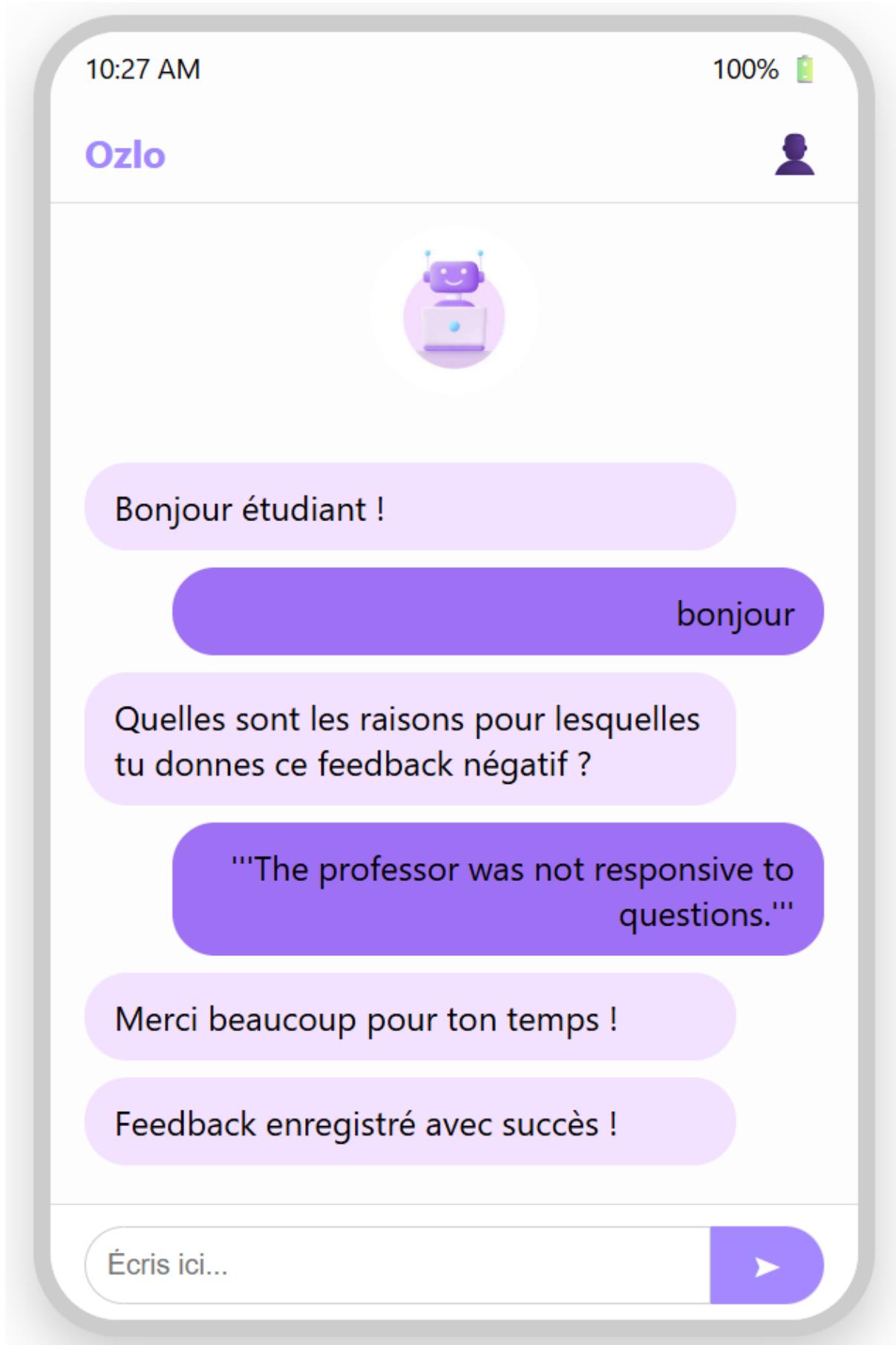
 Envoyer mon feedback  Réinitialiser

Vos feedbacks nous aident à améliorer la qualité des cours [Politique de confidentialité](#)

- **Analyse automatique de sentiment** : Le feedback est analysé (positif, neutre, négatif) via un modèle pré-entraîné.

ML (GI)	Hanane Khassoumi	Nassima (Étudiant)	1	June 9, 2025, 3:16 p.m.	Négatif		
JEE (IID)	Nassima Rhannouch	Nassima (Étudiant)	1	June 8, 2025, 1:14 p.m.	Négatif		
ML (GI)	Hanane Khassoumi	Nassima (Étudiant)	1	June 8, 2025, 12:24 a.m.	Négatif		
ML (GI)	Hanane Khassoumi	Nassima (Étudiant)	1	June 7, 2025, 3:08 p.m.	Négatif		
ML (GI)	Hanane Khassoumi	Nassima (Étudiant)	1	June 7, 2025, 3:07 p.m.	Positif		
JEE (IID)	Nassima Rhannouch	Nassima (Étudiant)	1	June 7, 2025, 2:33 p.m.	Négatif		
ML (GI)	Hanane Khassoumi	Nassima (Étudiant)	1	June 7, 2025, 2:02 p.m.	Négatif		
JEE (IID)	Nassima Rhannouch	Nassima (Étudiant)	3	June 7, 2025, 2:01 p.m.	Positif		
JEE (IID)	Nassima Rhannouch	Nassima (Étudiant)	1	June 7, 2025, 12:43 p.m.	Négatif		

- **Déclenchement du chatbot en cas de feedback négatif :** Le système lance une conversation pour vérifier le sérieux du feedback.



- **Recommandation automatique** : Une recommandation est générée automatiquement et envoyée au professeur concerné.
- **Historique des feedbacks** : L'étudiant peut consulter tous ses anciens feedbacks avec les sentiments associés.

### Historique des Feedbacks

The screenshot shows a user interface for viewing historical feedback. At the top, there are dropdown menus for 'Cours' (set to 'Tous les cours') and 'Sentiment' (set to 'Tous'), and a search bar. Below this, two feedback entries are listed in a card-based format.

**JEE**

Nassima Rhannouch 04/06/2025 11/06/2025 à 11:38

'The professor was not responsive to questions.'

**Suggestions d'amélioration**  
Aucune suggestion fournie.

**Négatif** **Partageable** ★ 1/5

**ML**

Hanane Khassoumi 04/06/2025 09/06/2025 à 15:16

Lack of examples made it hard to understand concepts.

**Suggestions d'amélioration**  
améliorer

**Négatif** **Partageable** ★ 1/5

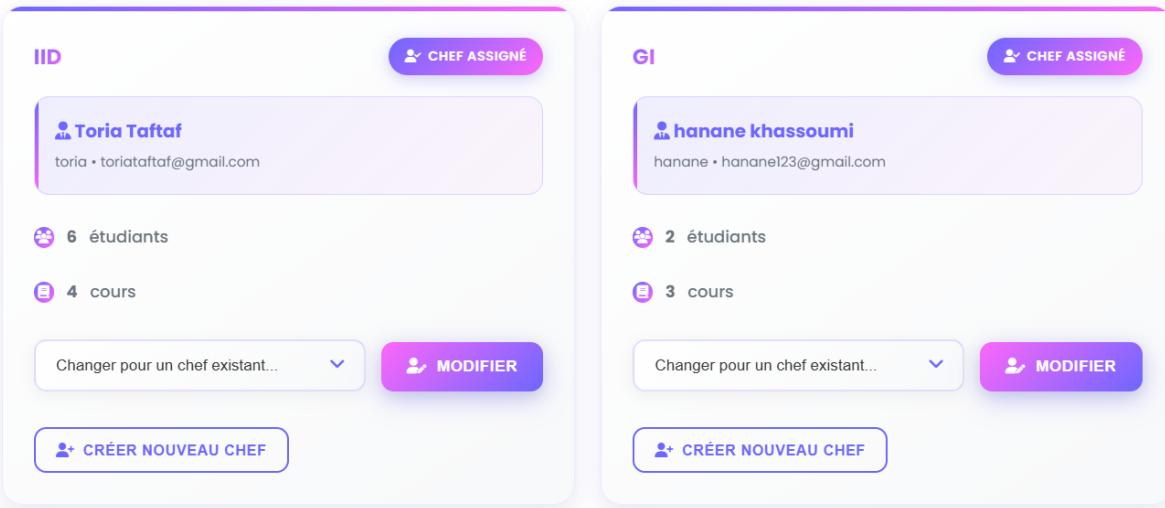
- **Statut** : Tous les tests sont réussis.

## 3. Interface Chef de filière

- **Analyse des feedbacks** : Le chef de filière a accès à un dashboard analytique contenant des statistiques sur les feedbacks par professeur, par cours, et par sentiment.
- **Filtrage et exploration** : Possibilité de filtrer par module, période ou type de feedback.
- **Statut** : Réussi.

## 4. Interface Administrateur

- **Gestion des chefs de filière** : L'administrateur peut :
  - Modifier un chef de filière existant.

 Filières avec Chef de Filière Assigné

Filière	Chef Assigné	Étudiants	Cours
IID	Toria Taftaf	6 étudiants	4 cours
GI	hanane khassoumi	2 étudiants	3 cours

- Affecter un nouveau chef en choisissant soit un professeur existant soit en créant un nouvel utilisateur. les profs et les chefs qui sont exsiste :

## ✓ Filières avec Chef de Filière Assigné

Changer pour un chef existant...

**Chefs de filière**

- hanane khassoumi (hanane)
- Toria Taftaf (toria)

**Professeurs**

- Nassima Rhannouch (nassimarhannouch123@gmail.com)
- Rahoui Hadil (hadil123@gmail.com)
- Hanane Khassoumi (hanane123@gmail.com)
- Yasmine Rhannouch (yassmine@gmail.com)
- Tasnime Rhannouch (tasnime@gmail.com)
- najlae Rahui (najlae@gmail.com)

Changer pour un chef existant... ▾

**MODIFIER**

**CRÉER NOUVEAU CHEF**

interface montre le rôle est modifiée :

USERNAME	EMAIL	ROLE	FILIERE	PROMOTION
hp	hp123@gmail.com	Étudiant	-	-
toria	toriataftaf@gmail.com	Chef de Filière	IID	-
yasmine	yassmine123@gmail.com	Étudiant	IID	-
hanane	hanane123@gmail.com	Chef de Filière	GI	Promotion 2023-2026
chef2025	chef@example.com	Étudiant	GI	-
admin	nassimarhannouch123@gmail.cm	Étudiant	IID	2020
Khadija	khadija@gmail.com	Étudiant	-	-
Nassima	nassimarhannouch123@gmail.com	Étudiant	IID	2020

### — Statut : Réussi.

Tous les cas d'usage définis dans les spécifications ont été testés manuellement. Les erreurs détectées ont été corrigées au fur et à mesure du développement.

# Difficultés rencontrées

---

Au cours de la réalisation de ce projet, plusieurs difficultés techniques et organisationnelles ont été rencontrées. Voici les principales :

## 1. Intégration de Dash dans Django

L'une des premières difficultés a été la tentative d'intégrer **Dash** (qui fonctionne comme une application Flask autonome) au sein du framework **Django**. Cela a nécessité :

- Une configuration spéciale des URLs pour inclure les dashboards dans les vues Django.
- L'utilisation de bibliothèques tierces comme `django-plotly-dash`, parfois mal documentées.
- La gestion des sessions utilisateurs entre les deux environnements, pour conserver la logique des rôles (étudiant, enseignant, admin).

## 2. Accès et qualité des jeux de données

L'obtention de jeux de données annotés pour les commentaires étudiants a été un défi :

- Peu de datasets publics existent dans le domaine éducatif avec des labels de sentiment.
- La nécessité de créer ou d'adapter manuellement des jeux de données de test (en français et en anglais).
- Des jeux déséquilibrés entre sentiments (trop de positifs ou neutres), affectant l'apprentissage.

## 3. Modélisation des relations dans la base de données

La conception du modèle relationnel a présenté des contraintes techniques :

- Gérer les liens entre utilisateurs, feedbacks, recommandations, et modèles.
- Concevoir des tables normalisées tout en gardant des performances correctes.
- Intégrer des relations plusieurs-à-plusieurs entre feedbacks et modèles NLP (en cas de reclassification).

## 4. Limitations matérielles et mémoire

Le traitement de texte (vectorisation TF-IDF, entraînement des modèles) est exigeant en ressources :

- Entraînement long sur des machines peu performantes (RAM limitée).
- Nécessité de sauvegarder les modèles et de les charger dynamiquement pour ne pas surcharger le serveur Django.
- Difficulté à tester sur des jeux volumineux en local.

Malgré ces obstacles, des solutions progressives ont été mises en place, avec des compromis entre performance, précision et faisabilité.

# Conclusion et perspectives

---

Ce projet de système interactif d'analyse des feedbacks étudiants vise à automatiser la collecte, structurer l'analyse et fournir des insights pertinents aux acteurs pédagogiques. En s'appuyant sur une architecture modulaire combinant Django, Dash, et des modèles de Machine Learning (Naive Bayes pour le français et Régression Logistique pour l'anglais), et en appliquant des techniques NLP comme TF-IDF et la suppression des *stop words*, nous avons pu classifier efficacement les sentiments exprimés dans les commentaires. Cette approche a permis de rationaliser la gestion des retours, de détecter les tendances critiques en temps réel, et d'améliorer la réactivité des enseignants via un tableau de bord dynamique. Malgré certains défis techniques (intégration de Dash dans Django, traitement de jeux de données multilingues, contraintes mémoire), le système s'est révélé stable, évolutif et efficace. Les perspectives incluent l'adoption de modèles NLP avancés (type BERT), l'extension multilingue, et l'optimisation pour des environnements mobiles. Nous exprimons enfin notre sincère gratitude à notre encadrant, **Pr. Nidal Lamghari**, pour son accompagnement constant, ses conseils précieux et son implication bienveillante tout au long du projet.