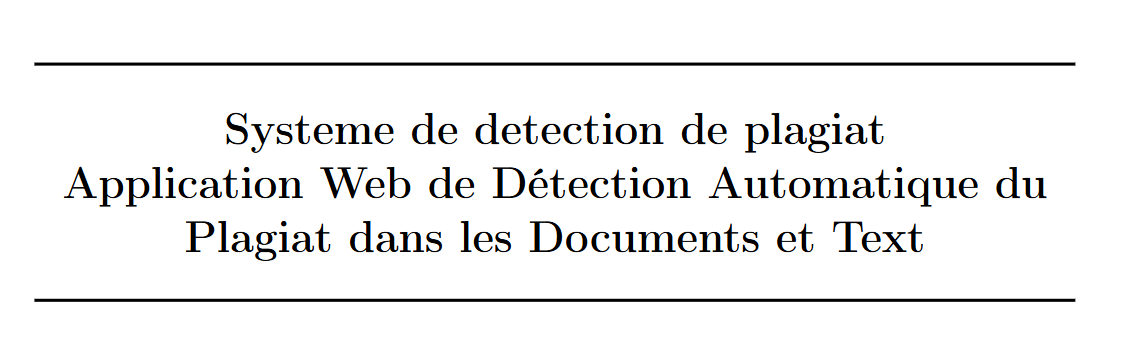


**Ecole Marocaine des sciences de l’ingénieur**

****

***Réalisé Par :***

***Abderrahmane Nassif***

***Badr Boulaid***

Remerciement

Tout d'abord, je tiens à exprimer ma sincère gratitude à Loubna Chaibi pour sa disponibilité, ses conseils et son soutien tout au long de ce projet. Ses connaissances et ses orientations ont été très utiles pour mener à bien ce travail.  
  
Je tiens également à remercier l'ensemble du corps professoral de L'école Marocaine de science de l'ingénieur pour leur soutien et leur engagement, ainsi que pour la qualité élevée de l'enseignement que j'ai reçu durant ma formation.  
  
Je voudrais exprimer ma gratitude à mes collègues et conseillers en promotion pour les conversations enrichissantes, le partage de connaissances et le soutien moral tout au long de ce parcours.  
  
Enfin, je voudrais exprimer ma sincère gratitude à ma famille pour leur soutien indéfectible, leur patience et leurs encouragements tout au long de ma carrière académique.

Résumé

Le plagiat est désormais un problème majeur dans le monde académique et professionnel, mettant en danger la qualité du travail et l'intégrité intellectuelle. Dans ce contexte, notre projet vise à concevoir et à mettre en œuvre un système de détection de plagiat accessible via une application web contemporaine.  
  
Les utilisateurs peuvent soumettre des documents ou des extraits de texte à l'application pour faire vérifier leur originalité. Une fois le contenu extrait, le système utilise des algorithmes de similarité textuelle pour comparer le texte soumis avec une base de données existante de documents. Après cela, il affiche un rapport complet qui détaille le pourcentage de plagiat trouvé ainsi que les passages similaires.  
  
De plus, le système inclut un historique des analyses effectuées, un système de gestion des utilisateurs sécurisé, et une interface d'administration pour la gestion.

Abstract

This report presents the development of a web application dedicated to the automatic detection of plagiarism in texts and documents. The main objective of this project is to provide students, professors, and institutions with a simple, fast, and efficient tool for analyzing and comparing textual content to ensure academic integrity.

The work begins with a detailed analysis of user needs, followed by a comprehensive design phase based on UML diagrams. The technical part focuses on the implementation of the web application using Python, Django, MySQL, HTML/CSS, technologies. Various modules such as file upload, text extraction, similarity comparison, and results display are explained in detail.

This project highlights the importance of academic honesty and proposes an intelligent and accessible solution to support plagiarism detection efforts within educational environments.

Introduction générale

À une époque où l'accès instantané et généralisé à l'information est rendu possible par Internet, la question de l'originalité des productions intellectuelles est posée avec une précision croissante. Le plagiat, qu'il soit intentionnel ou non, est un problème sérieux dans les milieux académiques et professionnels. Cela met l'accent sur la crédibilité des institutions concernées ainsi que sur l'intégrité du travail accompli.  
  
À la lumière de cela, il devient nécessaire de mettre en place des outils permettant une détection efficace du plagiat. Particulièrement, les établissements d'enseignement supérieur nécessitent des solutions fiables, automatisées et conviviales pour maintenir la qualité et l'authenticité des travaux des étudiants.  
  
Dans ce cadre, se trouve notre projet de fin d'année, qui consiste à créer une application en ligne pour la détection automatique.

Table de Matières

**Chapitre I : Contexte du projet**

* Introduction
* I.1 Présentation générale du projet
  + I.1.1 Problématique
  + I.1.2 Étude de l’existant
* I.2 Objectifs du projet
* I.3 Solution proposée
* Conclusion

**Chapitre II : Étude des besoins**

* Introduction
* II.1 Besoins fonctionnels
* II.2 Besoins non-fonctionnels
* II.3 Description des acteurs
* Conclusion

**Chapitre III : Analyse et conception**

* Introduction
* III.1 Outils de modélisation utilisés
* III.2 Diagramme de cas d’utilisation
* III.3 Diagrammes de séquence
* III.4 Diagramme de classes
* Conclusion

**Chapitre IV : Étude technique**

* Introduction
* IV.1 Langages utilisés
  + IV.1.1 PYTHON / DJANGO / HTML / CSS
  + IV.1.2 MySQL
* IV.2 Logiciels et Serveur utilisés
  + IV.2.1 Visual Studio Code
  + IV.2.2 Serveur XAMPP / PHPMyAdmin
  + IV.2.3 API Utilisé
* Conclusion

**Chapitre V : Réalisation**

* Introduction
* V.1 Interfaces d’accueil
* V.2 Interfaces de création
* Conclusion

**Conclusion générale**  
**Webographie**

Table de Figures

* Figure 1.1 : diagramme de cas d'utilisation
* Figure 1.2 : diagramme d'inscription d'un utilisateur
* Figure 1.3 : diagramme d'analyse de texte de plagiat
* Figure 1.4 : diagramme de classe
* Figure 2.1 : Logo HTML 5
* Figure 2.2 : Logo Langage CSS
* Figure 2.3 : Logo Langage Python
* Figure 2.4 : Logo Framework Django
* Figure 2.5 : Logo Langage SQL
* Figure 3.1 : Interface d'accueil d'application
* Figure 3.2 : Interface d'inscription d'application
* Figure 3.3 : Interface d'inscription d'application
* Figure 4.1 : Interface d'accueil d'utilisateur
* Figure 4.2 : Interface d'importer un document
* Figure 4.3 : Interface de coller un texte manuellement
* Figure 4.4 : Interface de l'historique de plagiat
* Figure 4.5 : Interface d'accueil d'administrateur

Chapitre I : **Contexte du projet**

**Introduction**

Il est crucial de bien comprendre le contexte dans lequel une solution informatique est intégrée avant de la concevoir et de la développer. Cette étape initiale permet d'identifier clairement le problème à résoudre, d'analyser les besoins réels des utilisateurs et d'établir les bases d'un projet cohérent et pertinent.  
  
Dans un cadre académique où des rapports, des mémos et des articles sont constamment produits, la question du plagiat prend une importance considérable. De nos jours, les outils de recherche permettent de copier et coller du contenu en quelques clics, rendant plus difficile la détection du plagiat sans l'aide de logiciels. C'est pourquoi il y a un besoin croissant d'automatiser ce processus en utilisant des solutions technologiques efficaces.  
  
Par conséquent, les objectifs de ce chapitre sont de décrire le cadre général du projet, de révéler le problème qui l'a conduit, et de mener une analyse.

**I.1 Présentation générale du projet**

**I.1.1 Problématique**

Le plagiat devient un problème de plus en plus important dans un monde où l'information est instantanément disponible et facilement copiée et partagée, en particulier dans les milieux académiques et professionnels. La facilité de copier-coller depuis Internet ou d'autres sources numériques rend de plus en plus difficile la détection du plagiat, en particulier lorsqu'elle repose uniquement sur des méthodes humaines ou manuelles.  
  
En revanche, le plagiat a plusieurs effets négatifs, notamment une crédibilité diminuée, une atteinte à la propriété intellectuelle et une compromission de l'intégrité des auteurs. À la lumière de ce problème, il devient impraticable de mettre en œuvre des outils numériques capables d'analyser automatiquement les documents et de trouver des similitudes avec des textes préexistants.  
  
Comment créer une application web facile à utiliser, rapide et accessible afin que chaque utilisateur puisse vérifier l'unicité d'un

**I.1 Présentation générale du projet**

**I.1.2 Étude de l’existant**

Un certain nombre de solutions numériques ont émergé en réponse au problème croissant du plagiat, permettant l'analyse des documents et l'identification des similitudes avec le contenu déjà publié. Parmi les plateformes les plus connues, on trouve :

**Turnitin** : largement utilisé dans les universités, il offre une détection très poussée, mais il est payant et souvent réservé aux établissements abonnés. De plus, son interface est complexe pour un utilisateur non initié.

**Plagscan** et **Urkund** : solutions similaires à Turnitin, également orientées vers les institutions académiques, avec des coûts parfois élevés et un accès limité.

**Scribbr**, **Quetext** ou **SmallSEOTools** : des outils accessibles au grand public, mais souvent limités en nombre de mots, en qualité d’analyse ou en profondeur de comparaison.

Malgré leur efficacité, ces solutions présentent plusieurs **limites** :

* L’accès est souvent **payant ou restreint**.

Les interfaces ne sont pas toujours **conviviales**.

* Il n’y a souvent **aucune possibilité de personnalisation** ou de gestion d’une base de textes interne.

Face à ce constat, il est pertinent de proposer une application web :

 Gratuite ou accessible librement.

 Permettant à l’administrateur d’ajouter ses propres textes de référence.

* Et offrant à l’utilisateur une **interface intuitive** pour déposer un fichier ou coller un texte, avec un retour instantané sur le taux de plagiat.

**I.2 Objectifs du projet**

L'objectif principal de ce projet est de concevoir et de développer une application web intuitive qui permettra aux utilisateurs de vérifier l'originalité d'un texte ou d'un document en utilisant un système automatisé de détection de plagiat.

* **Faciliter la détection du plagiat** pour tout utilisateur, via une interface simple et accessible, sans besoin de formation préalable.
* Permettre à l’utilisateur de **soumettre un document (PDF, Word)** ou de **coller un texte brut**, afin d’effectuer une analyse rapide de similarité.
* Afficher un **taux de plagiat clair (%)**, accompagné d’un aperçu des correspondances détectées.
* Offrir à l’administrateur une interface de gestion permettant :

 L’ajout **de textes de référence** à la base de comparaison.

 La **gestion des comptes utilisateurs** (création, modification, suppression).

* Garantir la **sécurité et la confidentialité** des fichiers déposés.

**I.3 Solution proposée**

Afin de résoudre le problème du plagiat et de réduire les lacunes trouvées dans les recherches existantes, nous proposons de créer une application en ligne pour la détection automatique du plagiat, accessible à la fois au grand public et aux étudiants.  
  
Cette solution se distingue par sa facilité d'utilisation, sa rapidité d'analyse et sa personnalisation par l'administrateur. Elle repose sur deux interfaces principales :

### Interface Utilisateur

* Permet à tout utilisateur de **créer un compte** et de **se connecter** de manière sécurisée.
* Offre deux modes d’analyse :
  + **Téléversement de document** (formats : PDF, DOC, DOCX).
  + **Saisie manuelle de texte** via une zone dédiée.
* Fournit un **taux de similarité (%)**, accompagné d’un **rapport détaillé** sur les passages suspects et les sources correspondantes.
* Donne accès à l’**historique des analyses précédentes**, avec possibilité de retélécharger les rapports.

### Interface Administrateur

* Permet de **gérer les comptes utilisateurs** : ajout, modification, suppression.
* Donne la possibilité d’**ajouter, modifier ou supprimer** des **textes de référence** dans la base de données interne du système.
* Fournit un **tableau de bord** pour suivre l’activité du système (nombre d’analyses effectuées, taux moyens, etc.).

### Fonctionnalités techniques clés

* Interface **responsive**, adaptée à tous les supports (ordinateur, tablette, smartphone).
* Système de **comparaison automatique** basé sur la détection de similarité textuelle (matching de chaînes, ou algorithme de type Cosine Similarity, etc.).
* **Rapports téléchargeables** pour archivage ou preuve d’analyse.

**Chapitre I : Contexte du projet**

**Conclusion**

Nous avons posé les bases de la compréhension du projet dans ce premier chapitre. Tout d'abord, nous avons fourni un aperçu du contexte général dans lequel notre travail s'inscrit en soulignant la prévalence croissante du plagiat dans un environnement numérique où l'accès à l'information est plus facile.  
  
L'analyse des solutions existantes a permis d'identifier un certain nombre de solutions déjà présentes sur le marché tout en mettant en évidence leurs lacunes en termes d'accessibilité, de personnalisation et d'ouverture au public. Cette analyse nous a aidés à définir un problème clair, auquel nous avons répondu par des objectifs spécifiques et mesurables.  
  
Enfin, nous avons présenté la solution suggérée : une application web simple, efficace et innovante qui permet l'analyse automatique de textes ou de documents afin d'identifier des similitudes en utilisant une base de référence

**Chapitre II : Etude des besoins**

**Introduction**

Avant toute étape de la conception ou du développement d'un système informatique, il est essentiel d'identifier précisément les besoins que le système doit satisfaire. Cette étape est essentielle à la réussite de tout projet car elle permet de définir les attentes des utilisateurs finaux et d'identifier les fonctionnalités à mettre en œuvre.  
  
Dans le cadre de ce projet, nous avons créé une application en ligne pour la détection automatique du plagiat. Celle-ci est basée sur deux interfaces principales : une interface administrateur qui est utilisée pour gérer les utilisateurs et les textes de référence utilisés pour l'analyse, et une interface utilisateur qui permet aux utilisateurs de soumettre du contenu (tel que des fichiers ou du texte brut) pour analyse.  
  
Ainsi, ce chapitre décrit les exigences fonctionnelles qui définissent les services que le système doit

II.1 Besoins fonctionnels

**Acteur : Utilisateur**

* Créer un compte.
* Se connecter à l’espace utilisateur.
* Choisir entre :
  + **Uploader un document** (PDF, Word).
  + **Coller manuellement du texte**.
* Lancer l’analyse de plagiat sur le contenu soumis.
* Obtenir un **taux de similarité (%)** avec les textes de la base.
* Consulter le détail des similarités (surlignage ou texte associé).
* Télécharger un rapport PDF de détection.
* Consulter l’historique de ses analyses passées.

**Acteur : Administrateur**

* Se connecter à l’espace administrateur.
* Gérer les comptes utilisateurs :
  + Ajouter, modifier, désactiver ou supprimer un compte.
* Ajouter, modifier ou supprimer des **textes de référence** dans la base de données (sources contre lesquelles les textes soumis seront comparés).
* Consulter les statistiques globales (nombre d’analyses, taux moyens de plagiat, utilisateurs actifs...).

II.1 Besoins non-fonctionnels

### ****Sécurité****

* Authentification sécurisée (mot de passe hashé).
* Gestion des droits d’accès selon le rôle (admin ou utilisateur).
* Protection des fichiers et textes stockés.

### ****Ergonomie & accessibilité****

* Interface intuitive, adaptée aux non-techniciens.
* Interface responsive compatible mobile / tablette / desktop.
* Navigation claire avec boutons explicites.

### ****Performance****

* Temps de traitement court (< 15 secondes pour un document standard).
* Système capable de gérer plusieurs utilisateurs en simultané sans ralentissement.

### ****Compatibilité****

* Prise en charge des formats : PDF, DOC, DOCX.
* Possibilité de copier/coller du texte brut dans une zone dédiée. **Fiabilité**
* Détection correcte et précise des similarités.
* Messages clairs en cas d'erreurs (format incorrect, taille dépassée...).
* Sauvegarde automatique des analyses réalisées.

### ****Évolutivité****

* Possibilité future d’intégrer :
  + Une base de données externe ou un moteur de recherche documentaire.

II.3 Description des acteurs

Les différents acteurs qui interagissent avec le système sont présentés dans cette section. Chaque acteur est défini par son rôle, les actions qu'il peut entreprendre et les fonctionnalités auxquelles il a accès. Deux acteurs clés ont été identifiés pour notre application de détection de plagiat :

**Utilisateur :**

L’utilisateur représente toute personne souhaitant utiliser l’application pour vérifier l’originalité d’un texte. Il peut s’agir d’un étudiant, d’un professionnel ou de tout individu désirant analyser un document. Ce rôle possède une interface dédiée lui permettant d’interagir avec le système.

#### **Actions possibles :**

* Créer un compte personnel via un formulaire d’inscription.
* Se connecter à son espace utilisateur.
* Choisir une méthode d’entrée :
  + Importer un document (PDF, Word).
  + Coller un texte manuellement.
* Lancer l’analyse de plagiat.
* Consulter les résultats de l’analyse (taux de similarité, passages similaires).
* Accéder à l’historique de ses analyses précédentes.

**Administrateur :**

L’administrateur est responsable de la gestion globale du système. Il dispose d’un accès étendu qui lui permet de gérer les utilisateurs ainsi que les sources textuelles utilisées comme base de comparaison pour la détection de plagiat.

#### **Actions possibles :**

* Se connecter à l’interface d’administration.
* Gérer les comptes utilisateurs (ajout, modification, suppression).
* Ajouter des textes de référence à la base de données.
* Supprimer ou modifier les textes existants.
* Consulter les activités globales de l’application (analyses réalisées, utilisateurs actifs, etc.).
* Configurer certains paramètres du système (formats autorisés, taille maximale des fichiers...).

**Chapitre II : Etude des besoins**

**Conclusion**

Ce chapitre a permis de définir précisément les différentes exigences que notre application web de détection de plagiat doit respecter. Tout d'abord, nous avons identifié les besoins fonctionnels, qui décrivent les caractéristiques clés que les utilisateurs administrateurs et finaux souhaitent. Ces exigences incluent la gestion des comptes, l'ajout de citations, la soumission de documents ou de textes, et la création d'analyses de plagiat.  
  
Ensuite, nous avons examiné les exigences non fonctionnelles, qui définissent les normes de qualité à respecter. Cela inclut la sécurité des données, la performance du système, la compatibilité avec divers formats et l'ergonomie de l'interface.  
  
Enfin, les descriptions des acteurs ont permis de clarifier les rôles et les interactions potentielles au sein du système en différenciant

**Chapitre III : Analyse et conception**

**Introduction**

Ce chapitre est consacré à l'analyse et à la conception du système afin d'assurer une transition structurée entre la phase d'analyse des besoins et la phase de développement. Il est basé sur des outils de modélisation comme le langage UML, qui permet de représenter visuellement les différentes composantes du système. Nous commencerons par présenter les outils sélectionnés avant d'introduire le diagramme de cas d'utilisation qui décrit les interactions entre les acteurs du système et les fonctionnalités. Les diagrammes de séquence seront ensuite utilisés pour modéliser comment les scénarios fonctionnels se dérouleront, tandis que les diagrammes de classes peuvent être utilisés pour représenter la structure du système en identifiant les principales entités, leurs caractéristiques, leurs méthodes et les relations qui les lient.

**III.1 Outils de modélisation utilisés**

Nous avons décidé d'utiliser le Langage de Modélisation Unifié (UML), qui est actuellement une norme dans le domaine de l'ingénierie logicielle, pour modéliser notre système et représenter visuellement les différentes interactions, structures et comportements.  
  
UML permet la représentation graphique des composants d'un système logiciel en utilisant divers types de diagrammes, chacun ayant un but spécifique. Dans le cadre de ce projet, nous avons principalement utilisé les diagrammes suivants :

 Diagramme **de cas d’utilisation**: il permet de modéliser les interactions entre les acteurs (utilisateur, administrateur) et les fonctionnalités principales du système.

 Diagrammes **de séquence**: ils illustrent la chronologie des échanges entre les objets du système lors de l’exécution de scénarios fonctionnels.

 Diagramme **de classes**: il représente la structure statique du système, en identifiant les entités (classes), leurs attributs, leurs méthodes, ainsi que les relations entre elles (associations, héritages...).

Pour la conception de ces diagrammes, nous avons utilisé l’outil de modélisation StarUML qui offre une interface intuitive et conforme aux standards UML.

L'utilisation de ces outils nous a permis d'anticiper les interactions futures, de clarifier la structure du système et d'améliorer la communication entre les membres de l'équipe de développement.

**III.2 Diagramme de cas d’utilisation**

**Definition** :

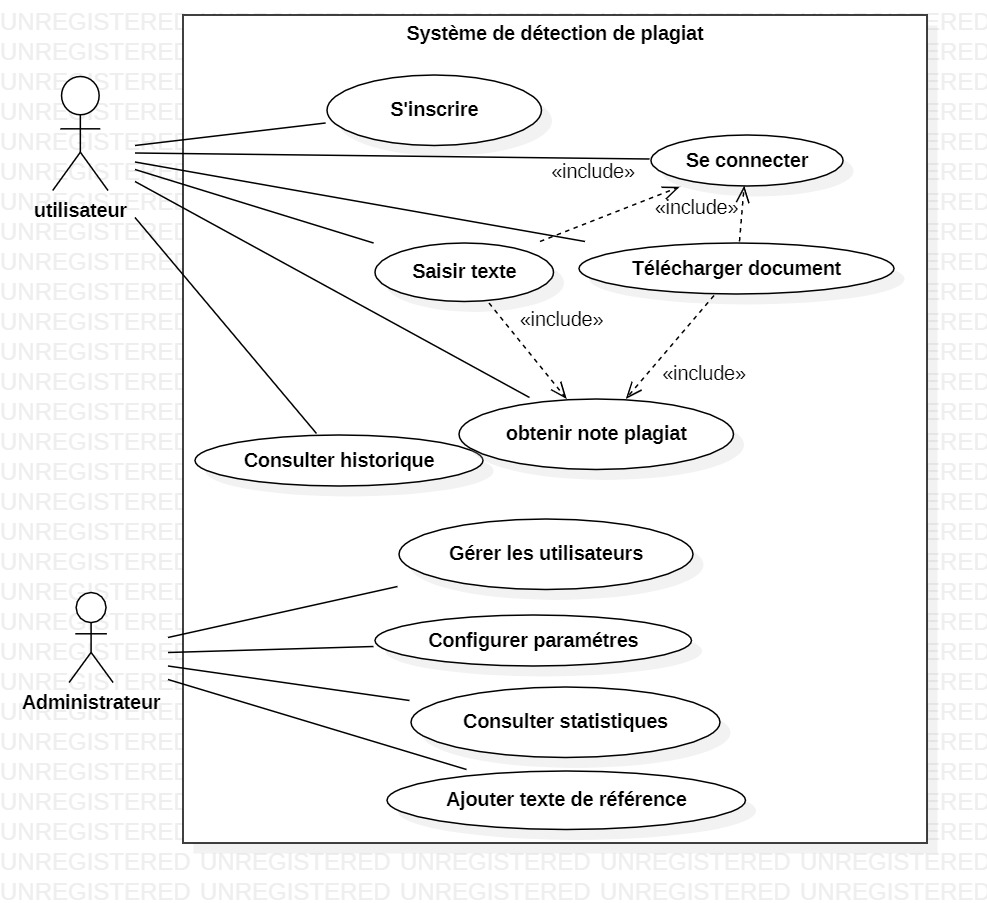
Le diagramme de cas d’utilisation, ou Use Case Diagram, est un diagramme UML qui permet de représenter les différentes interactions entre les utilisateurs (ou autres acteurs externes) et le système. Il met en évidence les fonctionnalités principales sous forme de cas d’utilisation, sans entrer dans les détails techniques de leur implémentation. Ce type de diagramme offre une vue d’ensemble fonctionnelle du système, facilitant la compréhension des besoins métier et la communication entre les parties prenantes du projet.

Figure 1.1 : Diagramme de cas d'utilisation

**Description du diagramme de cas d’utilisation de notre projet** :

Le diagramme de cas d’utilisation de notre projet représente les différentes interactions entre les deux principaux acteurs du système : l’utilisateur et l’administrateur. Il permet de visualiser les fonctionnalités essentielles offertes par l’application, telles que l’inscription, la connexion, le dépôt de documents, le lancement de l’analyse de plagiat, ainsi que la consultation des résultats et de l’historique pour l’utilisateur. Du côté administrateur, le diagramme met en évidence les actions de gestion des comptes utilisateurs et l’ajout de textes de référence servant de base à la détection. Cette modélisation fonctionnelle offre une vue claire et structurée des services proposés par le système, facilitant ainsi la phase de conception technique.

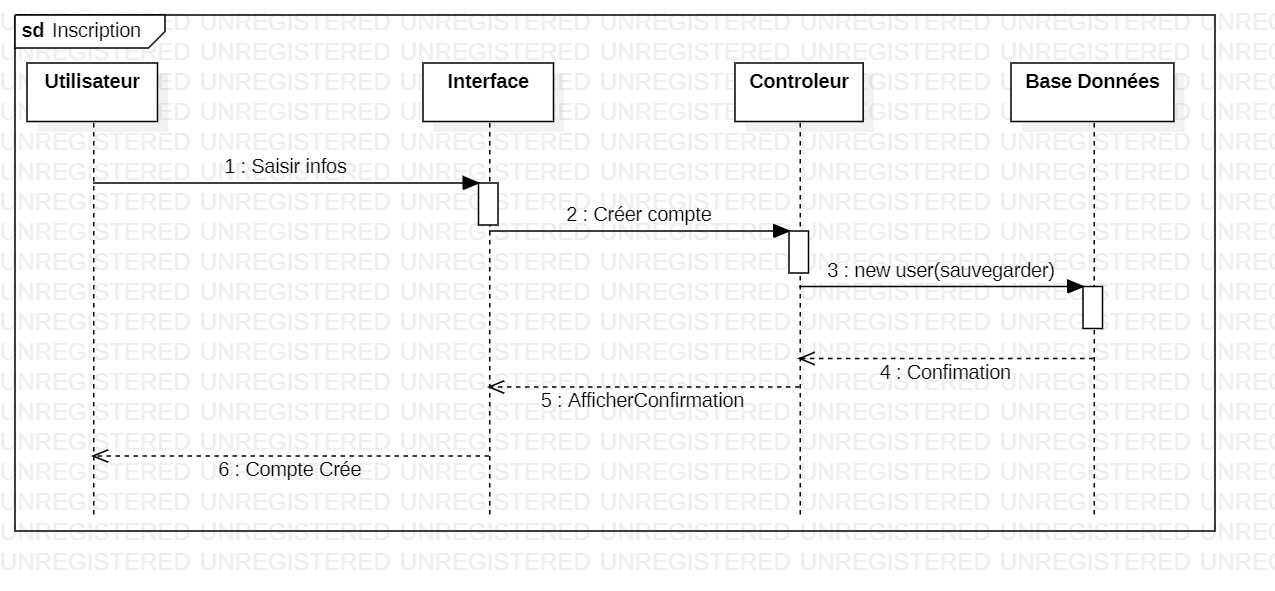
**III.3 Diagrammes de séquence**

**Définition** :

Le diagramme de séquence est un diagramme UML qui permet de représenter le déroulement temporel des interactions entre les différents objets ou composants d’un système, dans le cadre d’un scénario donné. Il met l’accent sur l’ordre d’envoi des messages, la chronologie des actions, ainsi que les relations de cause à effet entre les entités impliquées. Ce type de diagramme est particulièrement utile pour visualiser la logique d’exécution d’une fonctionnalité, en montrant comment les objets communiquent entre eux pour atteindre un objectif précis

**Description du diagramme de séquence de notre projet** :

Pour notre système, deux diagrammes de séquence ont été réalisés afin d’illustrer les scénarios clés.



.

Figure 1.2 : Inscription d'un utilisateur

**Description** :

Ce diagramme de séquence montre comment se déroule le processus d'ajout d'un nouvel utilisateur à l'application. Il explique étape par étape comment les messages sont envoyés entre les différents composants du système, y compris l'utilisateur, l'interface, le contrôleur et la base de données.

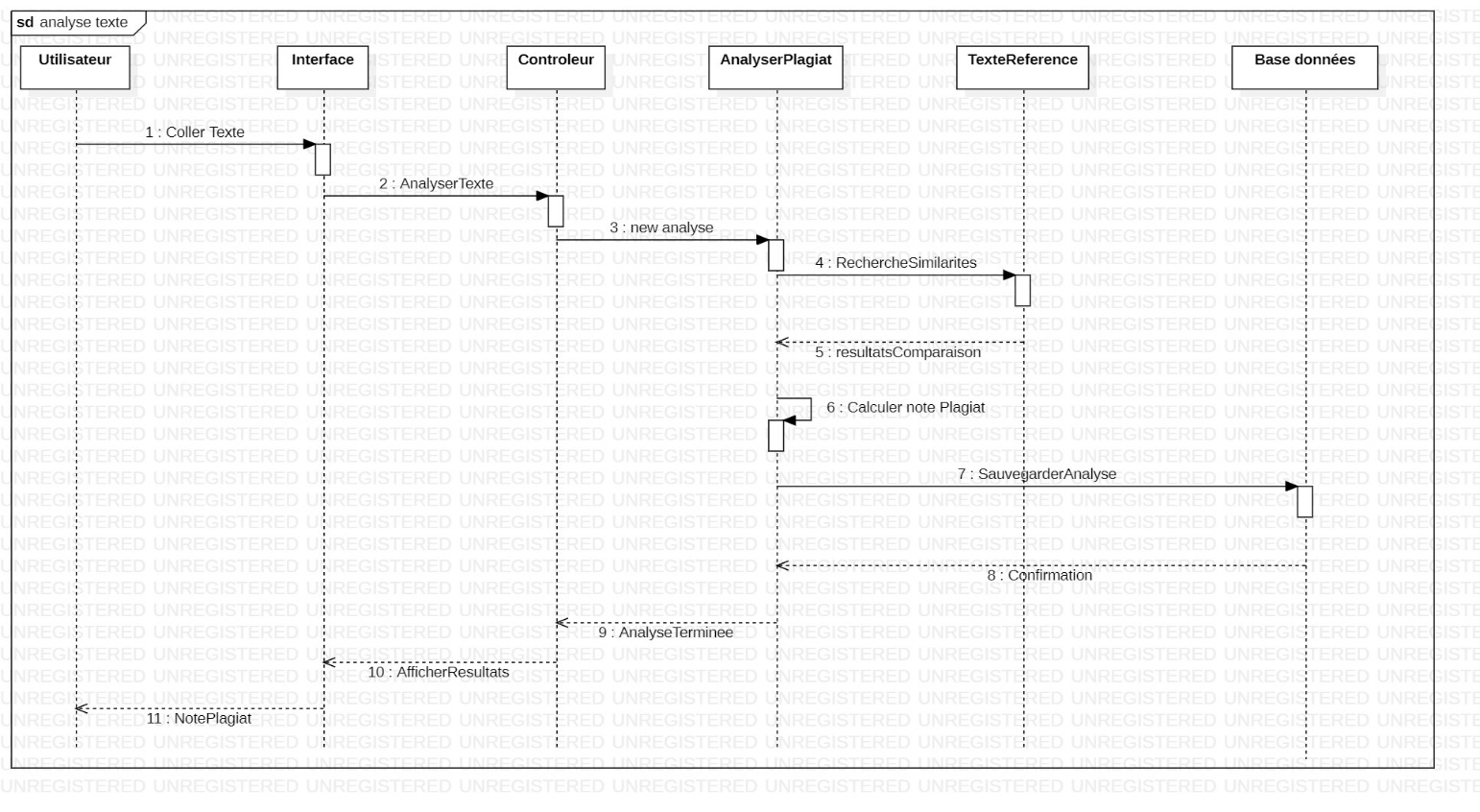


Figure 1.3 : Analyse de texte plagiat

**Description** :

Ce diagramme de séquence explique l'ensemble du processus de soumission et d'analyse d'un texte collecté par l'utilisateur dans l'application afin de déterminer son taux de plagiat. Il met en évidence les interactions entre l'utilisateur du système, l'interface, le contrôleur, le module d'analyse, les textes de référence et la base de données.

**III.4 Diagramme de classes**

**Définition** :

Le diagramme de classes est un diagramme structurel du langage UML qui permet de représenter la structure statique d'un système. Il met en évidence les classes du système (entités), leurs caractéristiques, leurs méthodes (ou opérations), et les relations entre elles (association, héritage, agrégation, composition, etc.). Ce type de diagramme est crucial lors de la phase conceptuelle car il offre une vue d'ensemble des composants fonctionnels du système et de leurs interactions, facilitant ainsi la mise en œuvre cohérente et modulaire du code source.

**Description du diagramme de classe de notre projet** :

Le diagramme de classes du système de détection automatique de plagiat présente les principales entités, leurs attributs, leurs méthodes, ainsi que les relations qui les lient. Il modélise les utilisateurs (inscription, connexion), les analyses de plagiat effectuées, les textes de référence utilisés pour la comparaison, ainsi que les rôles d’administration. Ce diagramme reflète la structure statique du système en représentant, de manière formelle, l’organisation des objets métiers, les interactions entre les composants (utilisateur, module d’analyse, base de référence), et les responsabilités de chaque entité dans le processus d’analyse.

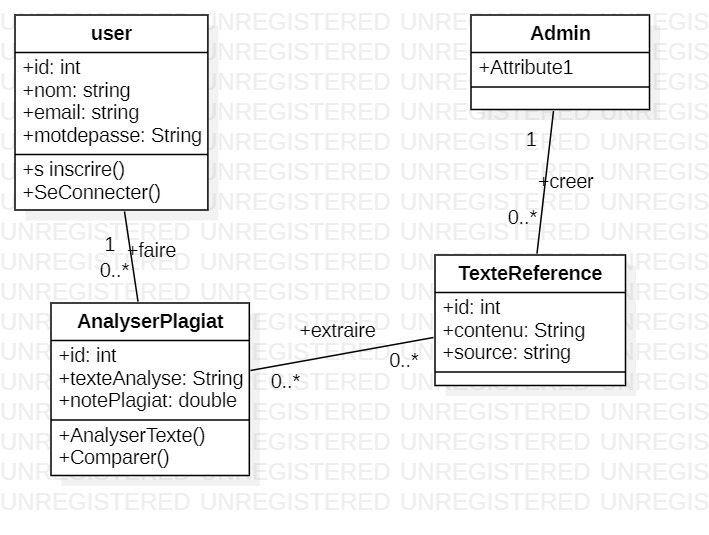


Figure 1.4 : Diagramme de classe

**Description** :

Ce diagramme de classes représente la structure statique de l’application web de détection de plagiat. Il identifie les entités principales du système, leurs attributs, leurs méthodes, ainsi que les relations qui existent entre elles. On y retrouve quatre classes principales : User, Admin, AnalyserPlagiat et TexteReference.

**Chapitre III : Analyse et conception**

**Conclusion**

Ce chapitre nous a permis d'utiliser le langage UML pour traduire les exigences du système en un modèle clair et structuré. Tout d'abord, nous avons présenté les outils de modélisation que nous avons utilisés pour développer notre application de manière rigoureuse et conforme aux normes. Ensuite, nous avons pu identifier les principales interactions entre les acteurs et le système à l'aide du diagramme de cas. Les diagrammes de séquence ont montré comment divers scénarios fonctionnels se déroulent dynamiquement en mettant en évidence l'échange de messages entre les composants. Enfin, en définissant les entités, leurs attributs, leurs méthodes et leurs relations, le diagramme de classes a permis de représenter la structure statique de l'application.  
  
Toutes ces représentations ensemble fournissent une base solide pour entrer dans la phase de

**Chapitre IV : Étude technique**

**Introduction**

Le succès d'un projet de développement logiciel dépend non seulement d'une analyse approfondie des besoins et d'un concept rigoureux, mais aussi de choix techniques cohérents et appropriés aux objectifs visés. Ces choix concernent principalement les langages de programmation, les outils de développement et l'environnement technique dans lequel l'application sera créée, testée et potentiellement déployée.  
  
Ce chapitre décrit les différentes technologies que nous avons utilisées pour mettre en œuvre notre application de détection de plagiat en ligne. Nous décrirons en partie les langages de développement utilisés pour le front-end, le back-end et la gestion des données, et en partie les serveurs et logiciels qui ont facilité la mise en œuvre du projet, tels que les éditeurs de code, les environnements de test et les plateformes de serveur.

**IV.1 Langages utilisés**

La création de notre application web pour la détection automatique du plagiat repose sur une collection de langages complémentaires, chacun jouant un rôle spécifique dans la structure du projet. Ces langages ont été sélectionnés en raison de leur facilité d'utilisation, de leur efficacité et de leur correspondance avec les objectifs de notre application.

**IV.1 Langages utilisés**

**IV.1.1 HTML** (**HyperText Markup Language)**

Les pages web de l'application sont structurées en utilisant le langage HTML. Il permet la définition de divers éléments de l'interface utilisateur, tels que des zones de texte, des boutons, des formulaires de dépôt et des sections de résultats. Ainsi, HTML est la base de toutes les pages destinées aux clients.

**Utilisation dans notre projet :**

Dans notre application, HTML a été utilisé pour construire toutes les interfaces visibles par l’utilisateur. Cela inclut :

* Le formulaire d’inscription et de connexion
* La page d’envoi de documents ou de texte brut
* L’affichage des résultats d’analyse (score de similarité, texte surligné, messages d’alerte)
* La page de consultation de l’historique des soumissions  
  Grâce à HTML, nous avons pu structurer chaque vue de façon cohérente et lisible pour l’utilisateur.



Figure 2.1 : Logo HTML 5

**IV.1.2 CSS (Cascading style sheets)**

En plus de HTML, CSS est utilisé pour améliorer le style visuel de l'interface. Cela permet de personnaliser les couleurs, les polices, les marges, l'alignement et l'assemblage pour les rendre réactifs sur différents écrans. Nous avons pu créer une interface esthétiquement plaisante, ergonomique et adaptable grâce au CSS.

**Utilisation dans notre projet :**

CSS a permis d’améliorer l’expérience utilisateur en rendant l’interface :

* Plus esthétique (utilisation d’une palette cohérente, typographie agréable)
* Plus lisible (espacements, alignements, sections bien délimitées)
* Plus intuitive (effets visuels sur les boutons, messages de validation ou d’erreur)
* Adaptée à tous les supports : mobile, tablette et ordinateur  
  En combinant HTML et CSS, nous avons pu offrir une interface claire, moderne et agréable à utiliser.



Figure 2.2 : Logo Langage CSS

**IV.1.3 Python**

Le principal langage utilisé sur les serveurs est Python. Il dispose de nombreuses bibliothèques puissantes, d'une syntaxe claire et d'une excellente lisibilité. Python est utilisé dans notre application pour gérer la logique métier, les contrôleurs, le traitement des documents, l'analyse de similarité et la génération de résultats.

**Utilisation dans notre projet :**

Python a joué un rôle central dans le développement côté serveur. Il a été utilisé pour :

* Définir les modèles représentant les utilisateurs, les documents soumis, les textes de référence, et les résultats d’analyse
* Écrire les vues qui gèrent les actions métier (soumission, traitement, affichage)
* Implémenter l’algorithme de détection de plagiat en comparant le texte soumis aux textes de référence (calcul du taux de similarité)
* Générer et stocker les rapports d’analyse
* Gérer les routes, les erreurs et les retours côté client  
  Grâce à Python, nous avons pu intégrer des traitements complexes tout en conservant un code maintenable et évolutif.

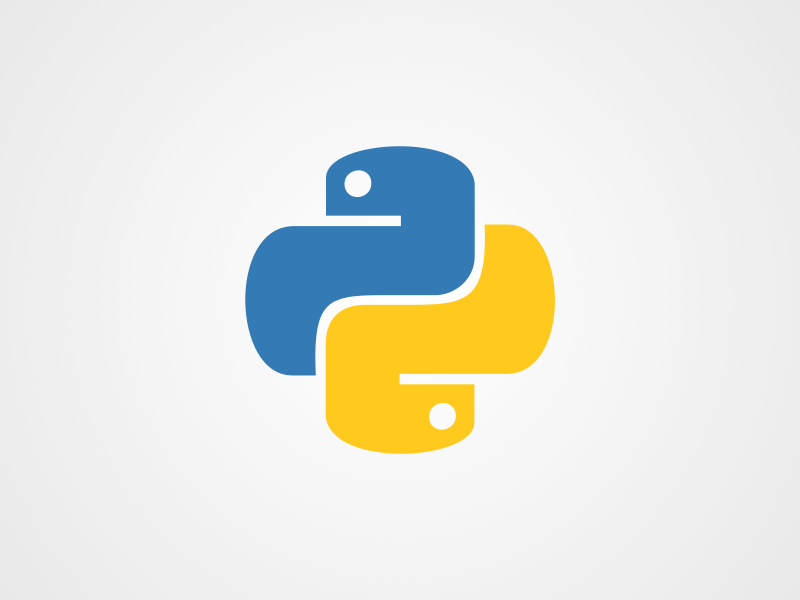
****

Figure 2.3 : Logo Langage Python

**IV.1.4 Django**

Django est un Framework web open-source construit sur Python. Il adhère au modèle MVC (Modèle-Vue-Contrôleur) et facilite le développement rapide d'applications web sécurisées

**Utilisation dans notre projet :**

Django a été l’épine dorsale de notre application. Il a permis de :

* Structurer le projet en modules clairs (app utilisateur, app analyse, app administration…)
* Créer l’ensemble des routes pour naviguer entre les pages
* Gérer l’inscription, l’authentification et les sessions utilisateurs
* Créer des formulaires sécurisés pour l’envoi de fichiers ou de texte
* Connecter facilement notre logique Python aux bases de données via des modèles Django
* Afficher dynamiquement les résultats d’analyse à l’utilisateur
* Gérer les privilèges d’accès entre les utilisateurs et l’administrateur  
  En résumé, Django nous a permis de gagner du temps de développement tout en garantissant un haut niveau de sécurité et de cohérence dans l’ensemble du système.



Figure 2.4 : Logo Framework Django

**IV.1 Langages utilisés**

**IV.1.2 SQL** (Structured Query Language)

Une langue standardisée appelée SQL (Structured Query Language) est utilisée pour communiquer avec les bases de données relationnelles. Il permet l'exécution d'opérations telles que la création de tables, l'insertion, la modification, la suppression et la recherche de données. Presque toutes les applications qui nécessitent un stockage de données structuré et un accès rapide à celles-ci utilisent SQL.

**Utilisation dans notre projet :**

Dans notre application, SQL est utilisé de manière indirecte via le **système ORM de Django**, mais les commandes SQL sous-jacentes sont essentielles pour :

* Créer les tables correspondantes aux modèles définis (Utilisateur, Document, Résultat-Analyse, Texte-De-Référence)
* Insérer et récupérer les données lors des actions utilisateur (connexion, soumission de document, affichage des résultats)
* Filtrer les données selon différents critères (exemple : afficher uniquement les documents de l’utilisateur connecté)
* Supprimer les documents ou utilisateurs si nécessaire
* Stocker les résultats d’analyse et les lier aux documents correspondant

****Grâce à SQL, notre application peut **gérer les interactions avec la base de données de manière fiable**, assurer la **cohérence des données**, et garantir un **accès rapide** aux informations nécessaires au bon fonctionnement du système

Figure 2.5 : Logo Langage SQL

**IV.2 Logiciels et Serveur utilisés**

Dans le cadre du développement de notre application web de détection automatique de plagiat, nous avons utilisé plusieurs logiciels, serveurs et bibliothèques pour assurer une réalisation efficace et structurée.

**Logiciels, serveurs et bibliothèques utilisés :**

#### **XAMPP :**

XAMPP est un serveur local gratuit qui regroupe Apache (serveur HTTP), MySQL (base de données), et d'autres outils.  
**Utilisation** : Il a permis de simuler un environnement de serveur web en local, notamment pour exécuter MySQL et accéder à phpMyAdmin.

#### **PhpMyAdmin :**

phpMyAdmin est une interface web permettant de gérer visuellement des bases de données MySQL.  
**Utilisation** : Elle a été utilisée pour consulter, vérifier et gérer les tables de la base de données générées par Django.

#### **StarUML :**

StarUML est un outil de modélisation UML utilisé pour créer des diagrammes standards (cas d’utilisation, séquence, classe, etc.).  
**Utilisation** : Nous l’avons utilisé pour concevoir les diagrammes UML du projet dans la phase de conception.

**Librairies et modules utilisés dans le projet Python/Django :**

#### **Scikit-learn**

scikit-learn est une bibliothèque Python pour le machine learning et le traitement de données.  
 **Utilisation** : Elle a été utilisée pour vectoriser les textes et calculer la similarité entre le texte soumis et les textes de référence.

#### **Python-docx**

python-docx est une bibliothèque Python qui permet de lire, modifier et extraire le contenu des fichiers Word (.docx).  
**Utilisation** : Elle a servi à extraire le texte brut depuis les fichiers DOCX déposés par les utilisateurs.

#### **PyPDF2**

PyPDF2 est une bibliothèque Python qui permet de lire et manipuler des fichiers PDF.  
 **Utilisation** : Elle a permis l’extraction du contenu textuel des fichiers PDF soumis à l’analyse.

#### **Dotenv**

dotenv permet de gérer les variables d’environnement via un fichier .env.  
 **Utilisation** : Elle a été utilisée pour protéger les informations sensibles comme les identifiants de base de données.

#### **Pathlib, os, Sys, re**

Ce sont des bibliothèques standards Python pour la manipulation de fichiers, de chemins système et d’expressions régulières.  
 **Utilisation** :

* Pathlib et os : gestion des chemins et fichiers.
* Sys : interaction avec l’environnement d’exécution.
* Re : traitement de texte via des expressions régulières.

#### **Modules internes du projet**

Ce sont les fichiers Python développés pour structurer l’application : modèles, formulaires, vues, fonctions d’analyse, etc.  
 **Utilisation** : Organisation de la logique métier, exécution des analyses de plagiat, gestion des utilisateurs et enregistrement des résultats.

**Chapitre IV : Étude technique**

**Conclusion**

Ce chapitre nous a permis de présenter toutes les méthodes de sélection qui ont aidé au développement de notre application web pour la détection automatique du plagiat. Les langages utilisés ont d'abord été décrits en détail, y compris HTML et CSS pour la structure et l'apparence de l'interface, Python pour la logique métier, et Django comme principal Framework pour la gestion du back end, l'authentification, la gestion de la base de données et l'architecture globale du projet. Ensuite, nous avons présenté les outils et logiciels qui ont facilité la mise en œuvre, tels que Visual Studio Code pour le développement, XAMPP et phpMyAdmin pour la gestion du serveur et de MySQL, et StarUML pour la modélisation UML. De plus, nous avons inclus un certain nombre de bibliothèques spécifiques à Python, y compris

**Chapitre V : Réalisation**

**Introduction**

La phase de réalisation est une étape cruciale dans le cycle de développement de notre système de détection automatique du plagiat. Avec la réalisation de diverses interfaces fonctionnelles, cela signifie la transition du concept théorique à la mise en œuvre pratique de l'application. Ce chapitre présente les principales interfaces créées pour ce projet, en mettant en évidence leurs fonctions, leur ergonomie et leurs contributions à l'expérience utilisateur.  
  
Nous commencerons par les interfaces de bienvenue, qui constituent le principal point d'entrée de la plateforme. Ils permettent à l'utilisateur de s'inscrire, se connecter et accéder à son espace personnel de manière facile et sécurisée. Ensuite, nous décrirons les interfaces de création, y compris celles pour soumettre des documents ou des textes, au début de

**V.1 Interfaces d’accueil**

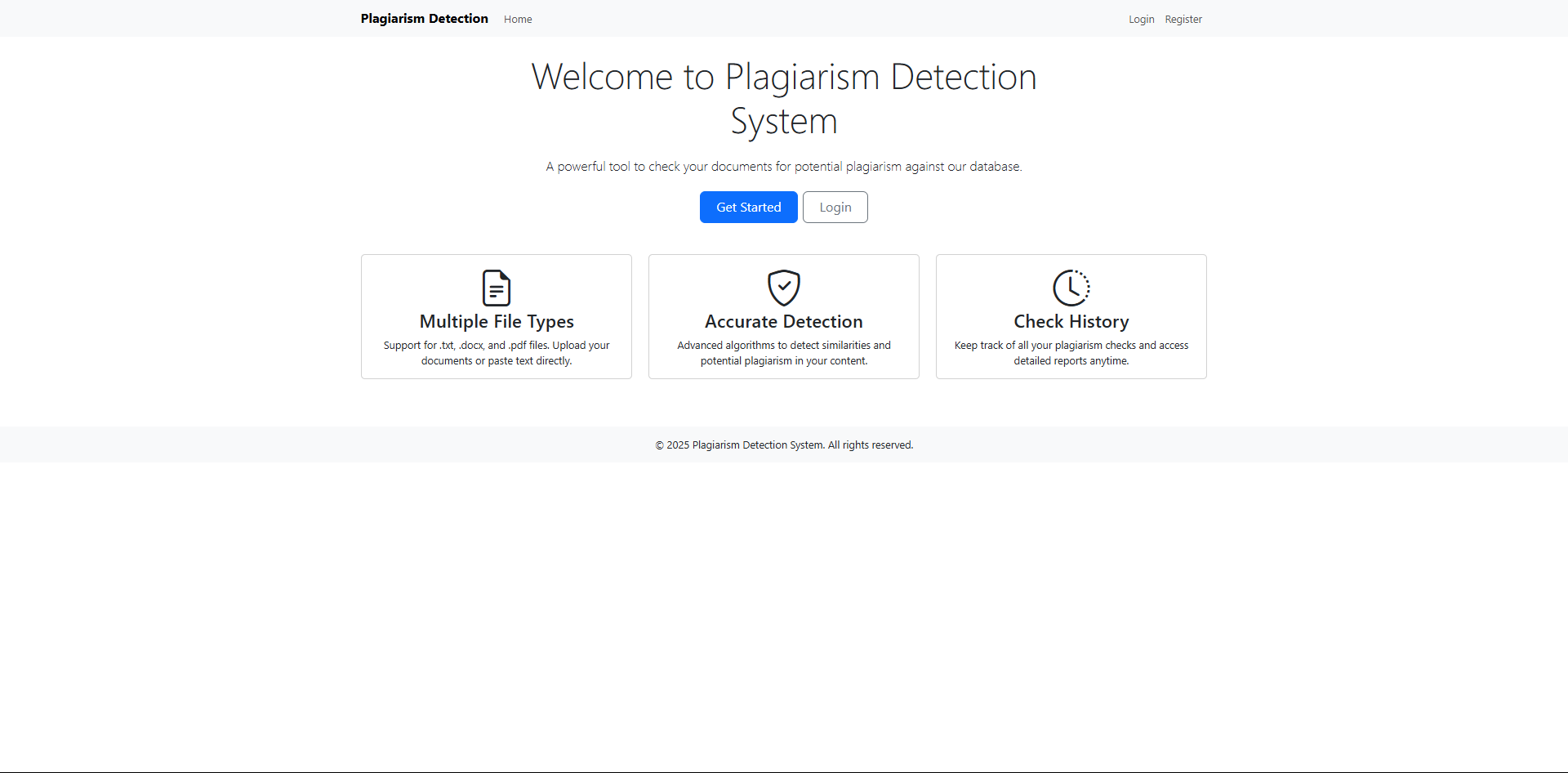
****

Figure 3.1 : Interface d'accueil d'application

**Description :**

Le point d'entrée principal de notre application de détection de plagiat est l'interface utilisateur, qui a été conçue pour offrir une expérience utilisateur claire, simple et divertissante. Elle décrit les caractéristiques essentielles du projet, y compris la gestion de plusieurs formats de fichiers (.txt, .docx et .pdf), la détection des fichiers avec précision grâce à des algorithmes sophistiqués, et l'accès à l'historique des analyses. Les boutons "Commencer," "Se connecter," et "S'inscrire" sur cette interface facilitent le démarrage des utilisateurs tout en mettant en avant l'objectif du système grâce à un design clair et soigné. Elle joue donc un rôle crucial dans l'engagement initial et la navigation de l'utilisateur.

**V.1.2 Interface D'inscription**

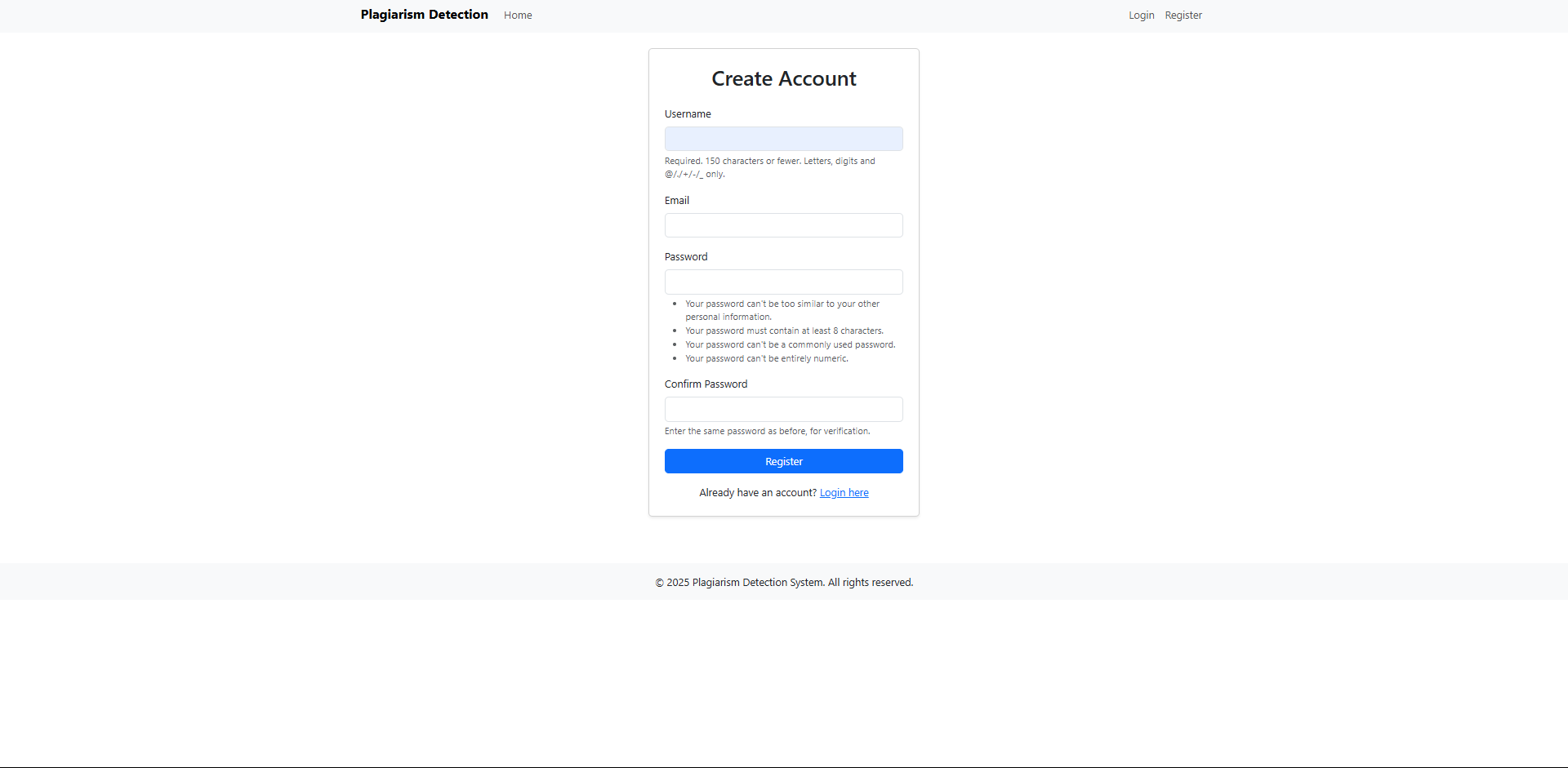
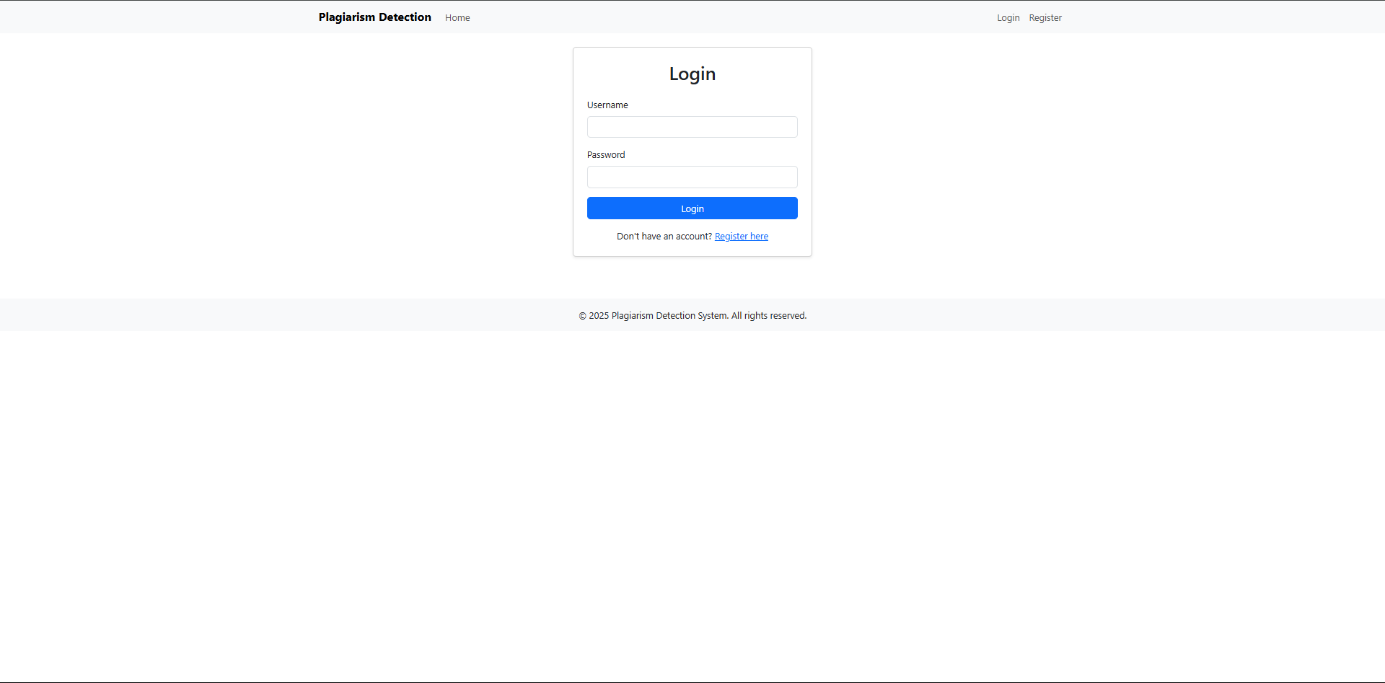
****

Figure 3.2 : Interface d'inscription d'application

**Description :**

En s'inscrivant simplement et en toute sécurité, l'interface de création de compte permet à tous les nouveaux utilisateurs d'accéder au système de détection de plagiat. Elle inclut des champs pour le nom de l'utilisateur, l'adresse e-mail, le mot de passe et la confirmation, ainsi que des instructions claires pour garantir la sécurité du mot de passe. Le design épuré et ciblé de cette interface garantit la validation des données avant l'enregistrement tout en favorisant l'accessibilité et la fluidité pour l'utilisateur. Par conséquent, c'est une étape cruciale pour intégrer les nouveaux utilisateurs à la plateforme.

**V.1.2 Interface De connexion**



. Figure 3.3 : Interface d'inscription d'application

**Description :**

En fournissant leur nom d'utilisateur et leur mot de passe, l'interface de connexion permet aux utilisateurs enregistrés d'accéder à leur espace personnel. Son design simple et ciblé garantit la sécurité de l'accès tout en favorisant une expérience utilisateur facile et intuitive. En cas de non-inscription, un lien redirige les utilisateurs vers la page d'inscription. Par conséquent, cette interface est une étape cruciale pour authentifier les utilisateurs et leur fournir un accès sécurisé aux fonctionnalités du système de détection de plagiat.

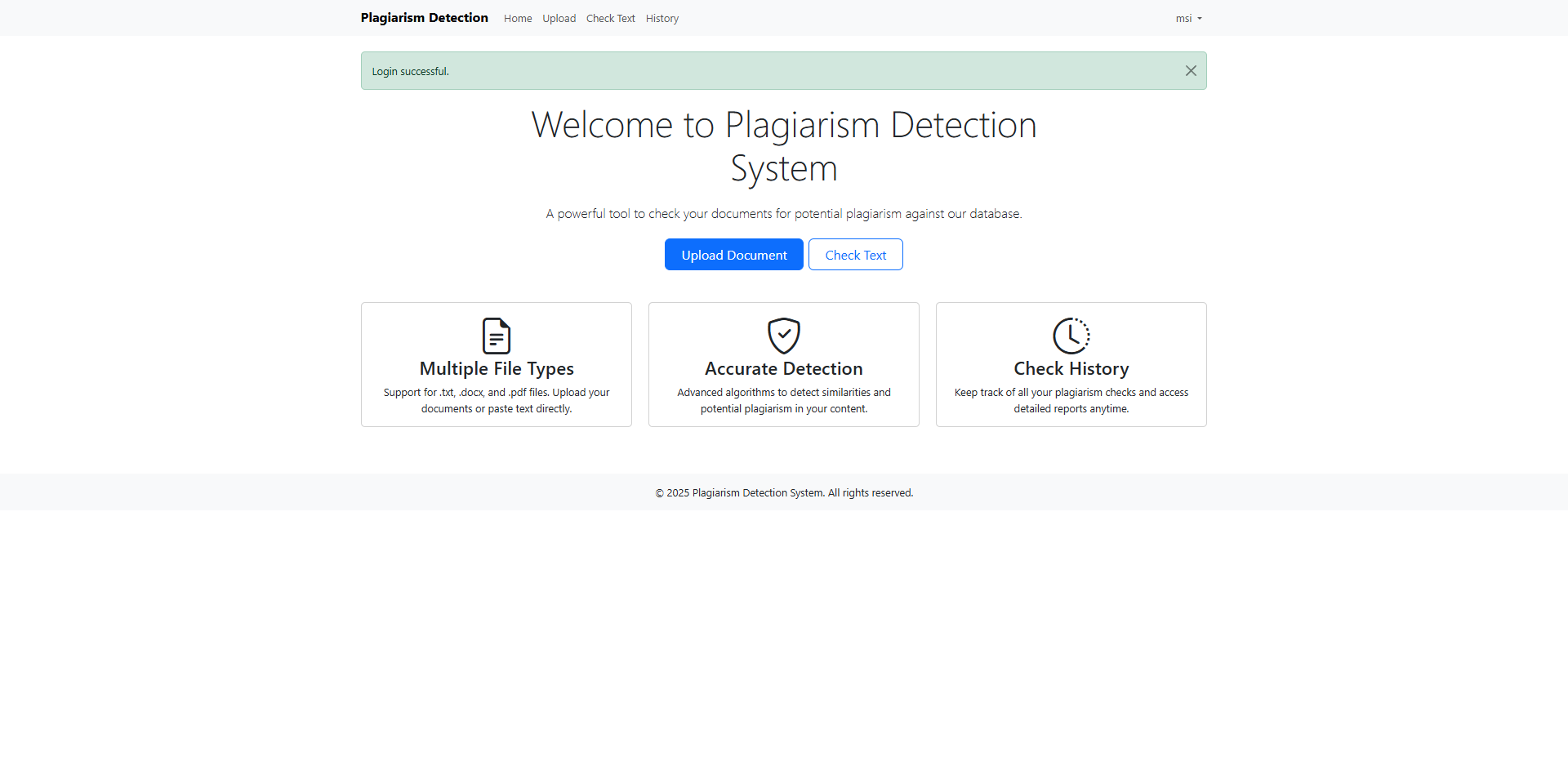
**V.1 Interfaces de creation**

Figure 4.1 : Interface d'accueil d'utilisateur

**Description :**

Une fois qu'un utilisateur se connecte au système de détection de plagiat, cette interface représente sa page d'accueil personnalisée. Elle confirme le succès de la connexion avec un message de validation tout en suggérant deux actions principales : envoyer un document pour analyse ou vérifier directement un texte. Avec des blocs informatifs sur les types de fichiers pris en charge, la précision de la détection et l'historique des vérifications, elle préserve les avantages de l'interface publique tout en offrant un menu de navigation amélioré adapté à l'utilisateur authentifié. Ce tableau améliore l'expérience utilisateur et facilite l'accès aux fonctionnalités clés.

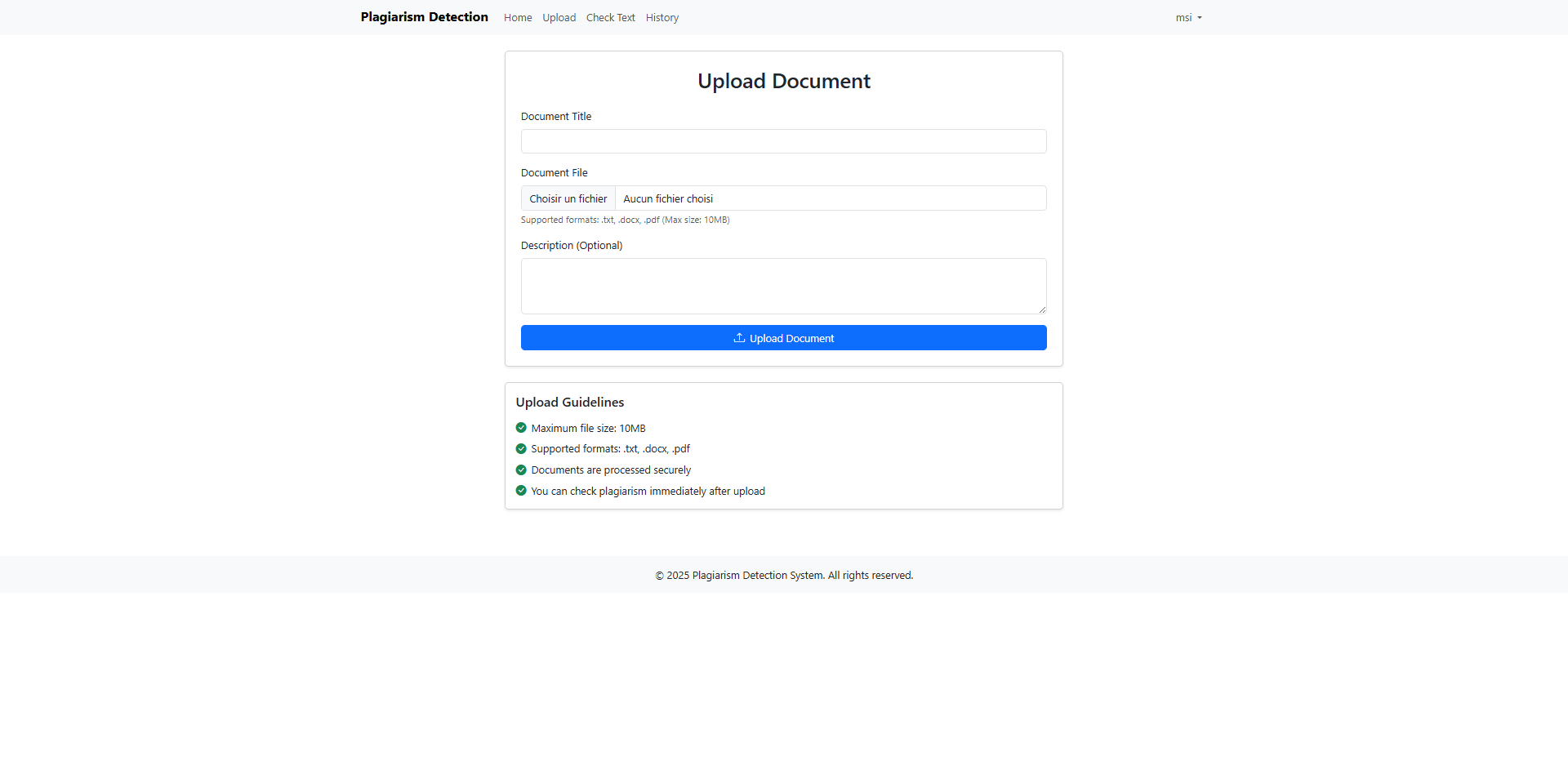


Figure 4.2 : Interface d'importer un document

**Description :**

Cette interface sert de page de soumission de documents, permettant aux utilisateurs de soumettre un fichier pour une analyse de plagiat. Elle fournit un formulaire clair avec des champs pour le titre du document, la sélection du fichier (les types de fichiers acceptés sont .txt, .docx et .pdf), et une description optionnelle. L'envoi peut être lancé avec un bouton unique. Les consignes utiles sont affichées dans la section ci-dessous : taille maximale, formats pris en compte, sécurité du traitement et capacité d'analyse immédiate. Cette interface est nécessaire pour démarrer le processus de détection dans le système de manière rapide, facile et sécurisée.

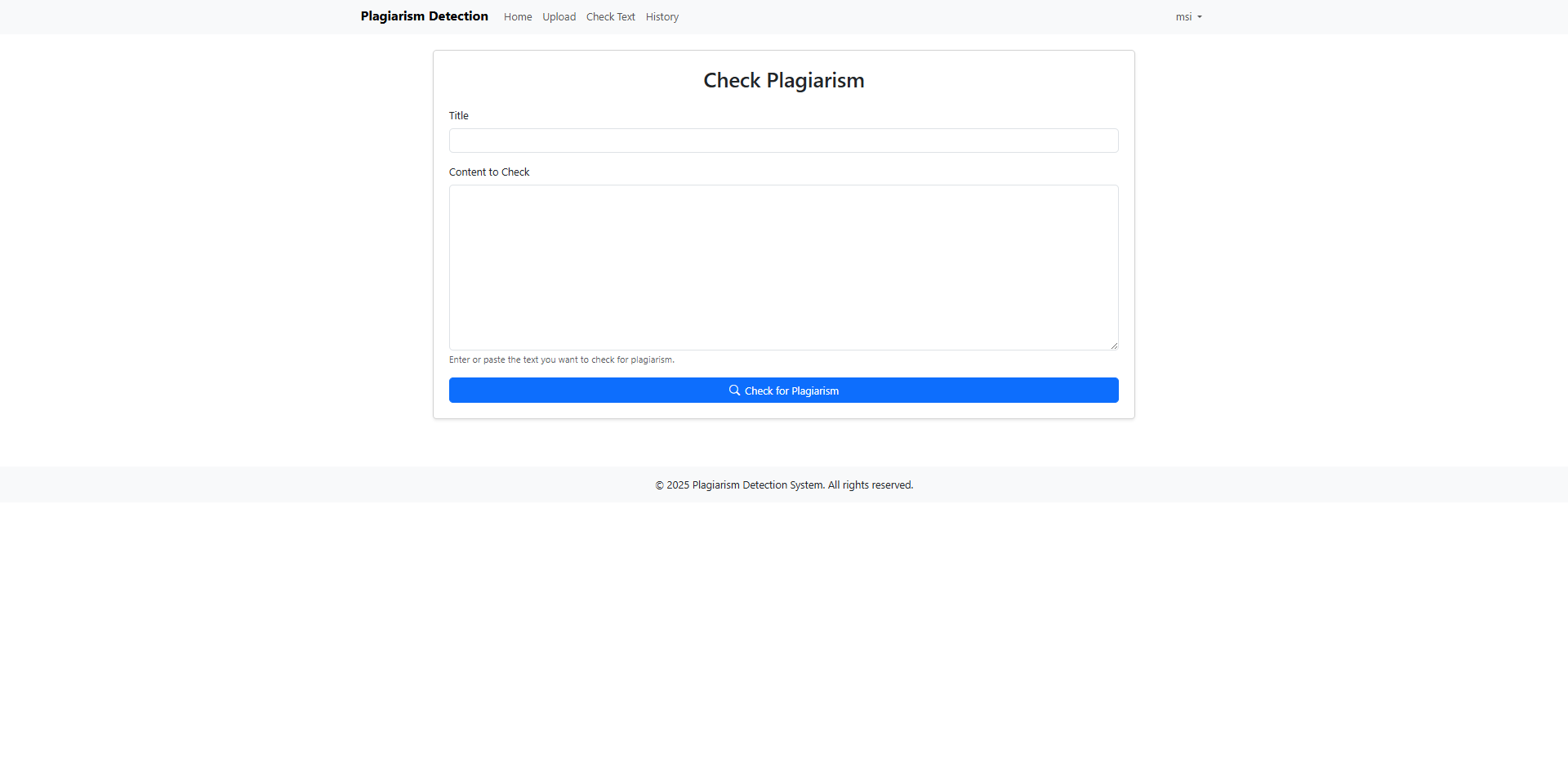


Figure 4.3 : Interface de coller un texte manuellement

**Description :**

En se repliant directement dans la zone désignée à cet effet, cette interface permet à l'utilisateur de vérifier manuellement le contenu textuel. Elle propose deux champs : un pour le titre du texte et un autre pour le contenu à examiner. Après avoir rempli ces champs, l'utilisateur peut commencer le processus de détection de plagiat en cliquant sur le bouton "Vérifier le plagiat". Cette fonctionnalité est utile lorsqu'un utilisateur ne souhaite pas ou ne peut pas télécharger un fichier, offrant un moyen rapide et facile de confirmer l'originalité d'un texte saisi directement dans l'application.

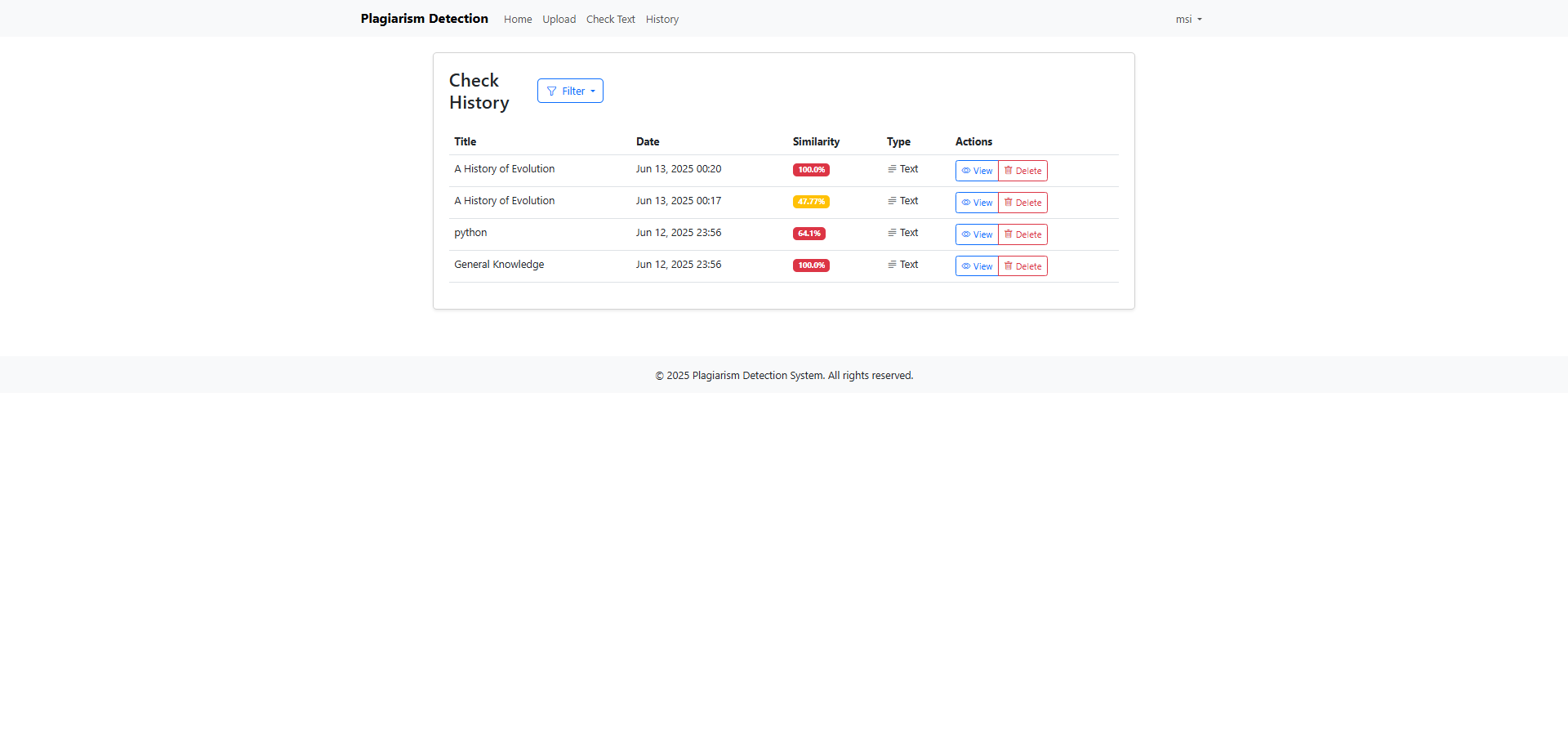


Figure 4.4 : Interface de l'historique de plagiat

**Description :**

Cette interface sert de section Vérifications Historiques de Plagiat, où les utilisateurs peuvent consulter toutes leurs recherches antérieures. Chaque entrée affiche le titre du texte ou du document, la date de vérification, le pourcentage de similarité trouvé, le type de contenu (texte ou fichier), et deux options : voir les résultats détaillés ou supprimer l'entrée. Cette interface offre une vue simplifiée et organisée qui permet à l'utilisateur de suivre l'évolution de ses analyses, d'identifier les documents problématiques ou de conserver un enregistrement de ses activités précédentes.

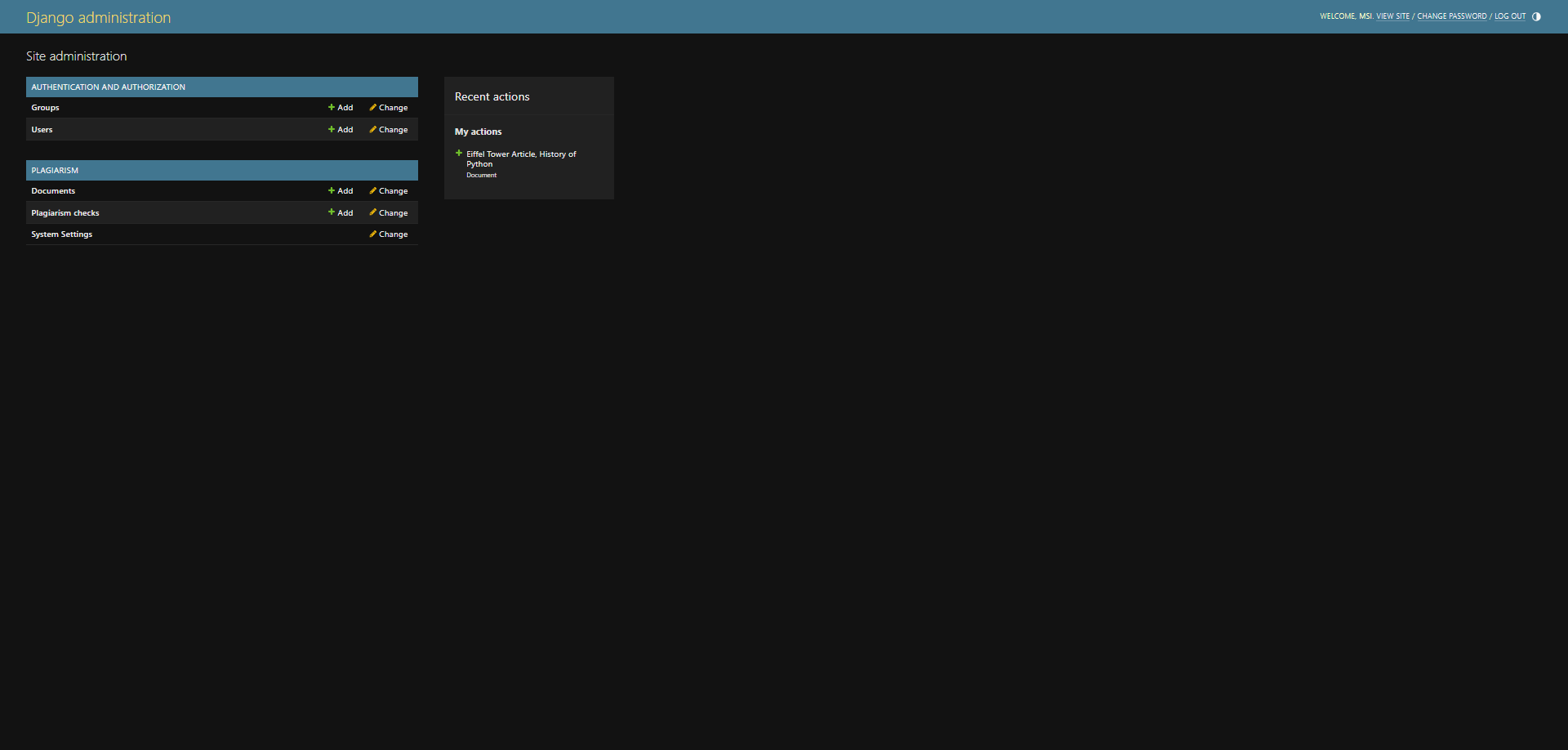
****

Figure 4.5 : Interface d'accueil d'administrateur

**Description :**

Cette interface représente la **page d’accueil de l’administration Django**, accessible uniquement aux utilisateurs disposant des droits d’administration. Elle permet de gérer l’ensemble des entités du système, telles que les **utilisateurs**, **groupes**, et **modules de plagiat**. Les sections affichées comprennent notamment :

* **AUTHENTICATION AND AUTHORIZATION** pour gérer les utilisateurs et les groupes,
* **PLAGIARISM** pour gérer les documents, les résultats des vérifications, et les paramètres du système.  
  La colonne de droite affiche les **actions récentes** de l’administrateur connecté. Cette interface, simple et structurée, offre une vue complète et centralisée sur les éléments à gérer dans le système.

**Chapitre V : Réalisation**

**Conclusion**

En conclusion, ce chapitre a permis de fournir un aperçu détaillé de toutes les interfaces créées pour notre système de détection de plagiat. De l'écran d'accueil aux pages d'inscription et de connexion, en passant par les modules de dépôt de documents, la vérification manuelle de texte, l'analyse historique, et enfin l'espace administratif, chaque élément a été conçu en tenant compte de la facilité d'utilisation, de la clarté et de la navigation. Ces interfaces offrent une expérience utilisateur intuitive et complète tant pour les utilisateurs traditionnels que pour les administrateurs, et elles traduisent de manière tangible les exigences fonctionnelles identifiées lors des phases précédentes. Ainsi, cette phase de réalisation est une étape cruciale pour mettre en production un système fonctionnel, efficace et accessible.

**Conclusion générale**

Dans le cadre de ce projet de fin d'année, nous avons pu concevoir, développer et mettre en œuvre un système complet de détection de plagiat, répondant à une demande croissante dans les domaines académique et professionnel. À travers les différents chapitres, nous avons d'abord examiné les exigences fonctionnelles et techniques du système avant de passer à un processus de modélisation approfondi en utilisant des outils UML. Nous avons ensuite commencé la phase de mise en œuvre, où chaque fonctionnalité a été soigneusement mise en place, en commençant par l'authentification des utilisateurs et en terminant par l'analyse automatique de similarité, suivie de la gestion des documents, de l'interface administrative et de l'historique des vérifications. Tout cela a été intégré dans une interface intuitive et ergonomique conçue pour faciliter l'utilisation tout en garantissant fiabilité et performance. Ce projet nous a permis non seulement de

**Webographie**

[**https://docs.python.org/3/library/difflib.html**](https://docs.python.org/3/library/difflib.html)

**https://openclassrooms.com/fr/courses/4425126-debutez-dans-leback-**

**end-avec-python**

[**https://developer.mozilla.org/fr/docs/Web/JavaScript**](https://developer.mozilla.org/fr/docs/Web/JavaScript)

[**https://stackoverflow.com/**](https://stackoverflow.com/)

**https://docs.python.org/3/library/difflib.html**