Rapport de Présentation du Projet : Système de Gestion de Bibliothèque

1. Introduction

1.1 Contexte et Objectifs du Projet

Dans le cadre de ce projet, nous avons développé un **système de gestion de bibliothèque** en Python. L'objectif principal est de fournir une solution simple et automatisée pour gérer les livres, les utilisateurs, et les emprunts au sein d'une bibliothèque. En effet, la gestion manuelle des livres, des retours, et des emprunts devient rapidement inefficace, surtout lorsqu'il y a un grand nombre de livres et d'utilisateurs à suivre. Ce projet vise donc à réduire cette charge administrative en automatisant certaines opérations et en offrant une interface utilisateur simple pour interagir avec le système.

Objectifs spécifiques :

- **Gestion des livres** : Ajouter, modifier et supprimer des livres dans la bibliothèque.
- **Gestion des utilisateurs** : Créer des comptes utilisateurs, gérer les emprunts et retours de livres.
- **Gestion des emprunts** : Suivre les dates de prêt et de retour des livres, et notifier les retards.
- **Système d'alertes pour les retards** : Le système calcule les retards et peut notifier les utilisateurs ou appliquer des amendes.

1.2 Problématique

La gestion manuelle des livres dans une bibliothèque peut rapidement devenir problématique. Les erreurs humaines peuvent entraîner des pertes de livres, des retards non signalés, ou des erreurs dans les comptes des utilisateurs. Ce projet vise à résoudre ces problèmes en offrant une solution automatisée et fiable. L'objectif est de créer un environnement où les bibliothécaires et les utilisateurs peuvent facilement interagir avec le système, garantissant ainsi une gestion fluide et efficace.

1.3 Public Cible

Le public cible de ce système inclut :

- **Les bibliothécaires** : Qui peuvent facilement gérer les livres, les utilisateurs et les emprunts.
- **Les utilisateurs de la bibliothèque** : Qui peuvent consulter les livres disponibles, emprunter ou retourner des livres, et consulter leur historique.
- **Les administrateurs** : Qui supervisent la gestion des utilisateurs et des livres.

2. Objectifs du Projet

L'objectif principal de ce projet est de concevoir un système simple mais complet pour gérer les livres et les emprunts dans une bibliothèque. Plus spécifiquement, nous avons cherché à réaliser les objectifs suivants :

- Permettre l'ajout, la suppression et la modification des livres dans la base de données.

- Gérer les emprunts de manière efficace, en enregistrant les dates d'emprunt et de retour.
- Mettre en place un système de gestion des retards, avec des alertes pour les utilisateurs en retard.
- Fournir une interface simple pour l'interaction avec le système, principalement basée sur la ligne de commande (CLI), mais extensible à une interface graphique si nécessaire à l'avenir.

3. Description Fonctionnelle

Le projet repose sur les principales fonctionnalités suivantes :

- **Gestion des Livres** : Le système permet de gérer les informations des livres : titre, auteur, ISBN, statut de disponibilité, etc. Chaque livre peut être ajouté, supprimé, ou modifié à tout moment.
- **Gestion des Utilisateurs** : Un utilisateur peut être inscrit avec des informations de base (nom, identifiant, etc.) et peut emprunter des livres. Il est possible de suivre l'historique des livres empruntés ainsi que les retours effectués ou non.
- **Gestion des Emprunts** : Lorsqu'un utilisateur emprunte un livre, le système enregistre la date de prêt et la date de retour prévue. Si le livre n'est pas retourné à temps, une alerte est envoyée. Le système gère aussi les retours de livres et met à jour la disponibilité du livre.
- **Gestion des Retards** : Le système vérifie régulièrement les livres non retournés et calcule les retards. Des rappels peuvent être envoyés à l'utilisateur, et des amendes peuvent être appliquées si nécessaire.

---

4. Technologies Utilisées

- **Langage de Programmation** : Le projet est développé en **Python**, un langage populaire pour les applications de gestion grâce à sa simplicité et sa richesse en bibliothèques.
- **Modules Python Utilisés** :
    - datetime : Pour la gestion des dates d'emprunt et de retour des livres.
    - os : Pour interagir avec le système de fichiers (par exemple, créer des fichiers pour sauvegarder les données).
    - csv : Pour stocker les données des livres et des utilisateurs de manière persistante, ce qui permet au système de se souvenir des informations même après un redémarrage.
    - **Gestion des Erreurs** : Utilisation de structures try/except pour gérer les erreurs de saisie de l'utilisateur ou les erreurs de fonctionnement du système.

5. Architecture du Système

Le système est conçu de manière à séparer les différentes responsabilités pour simplifier la maintenance et l'évolution du projet. Il repose sur une architecture simple et modulaire, avec les composants suivants :

- **Classes de Base** :
  - **Livre** : Une classe qui contient les informations d'un livre, comme le titre, l'auteur, l'ISBN et son statut de disponibilité.
  - **Utilisateur** : Une classe pour gérer les utilisateurs, leur nom, identifiant, et leur historique d'emprunts.
  - **Bibliothèque** : Une classe principale qui regroupe toutes les fonctions de gestion des livres et des utilisateurs.
- **Flux de Données** :
  - Lorsqu'un utilisateur souhaite emprunter un livre, le programme vérifie la disponibilité du livre, enregistre l'emprunt, et met à jour le statut du livre.
  - À la date prévue pour le retour, le système calcule si l'utilisateur est en retard et prend les actions nécessaires (alerte, amende, etc.).

6. Détail du Code Source

Voici quelques parties du code qui illustrent le fonctionnement du système. Ces extraits montrent comment les livres et les utilisateurs sont gérés :

Exemple de la classe Livre :

```
class Livre:
    def __init__(self, titre, auteur, isbn):
        self.titre = titre
        self.auteur = auteur
        self.isbn = isbn
        self.disponible = True  # Indique si le livre est disponible ou non

    def afficher(self):
        return f"Titre : {self.titre}, Auteur : {self.auteur}, ISBN : {self.isbn}, Disponible : {self.disponible}"
```

Exemple de la classe Utilisateur :

```
class Utilisateur:
    def __init__(self, nom, id_utilisateur):
```

```
        self.nom = nom
        self.id_utilisateur = id_utilisateur
        self.emprunts = []  # Liste des livres empruntés

    def emprunter(self, livre):
        if livre.disponible:
            livre.disponible = False
            self.emprunts.append(livre)
            print(f"{self.nom} a emprunté {livre.titre}.")
        else:
            print(f"{livre.titre} n'est pas disponible.")

    def retourner(self, livre):
        if livre in self.emprunts:
            livre.disponible = True
            self.emprunts.remove(livre)
            print(f"{self.nom} a retourné {livre.titre}.")
```

7. Test du Système

Le système a été testé pour vérifier plusieurs fonctionnalités :

- **Emprunt d'un livre disponible** : Lorsqu'un utilisateur emprunte un livre qui est disponible, le livre devient non disponible et l'utilisateur peut consulter son historique.
- **Retour d'un livre en retard** : Le système vérifie automatiquement les retards et peut envoyer un rappel à l'utilisateur.
- **Affichage des livres et utilisateurs** : Les livres et les utilisateurs peuvent être affichés avec leurs informations actualisées, montrant leur statut.

8. Limitations et Améliorations

Limitations actuelles :

- **Pas d'interface graphique** : Le projet utilise une interface en ligne de commande, bien qu'il soit possible d'ajouter une interface graphique (par exemple, avec Tkinter) pour améliorer l'expérience utilisateur.

- **Pas de base de données réelle** : Les données sont actuellement sauvegardées dans des fichiers csv. Une future amélioration pourrait intégrer une vraie base de données relationnelle comme SQLite.

Améliorations possibles :

- Ajouter une fonctionnalité de recherche pour que les utilisateurs puissent facilement trouver des livres.
- Implémenter un système de réservation de livres pour les utilisateurs.
- Ajouter une interface graphique pour une expérience utilisateur plus fluide.

9. Conclusion

Ce projet de gestion de bibliothèque en Python offre une solution simple mais fonctionnelle pour la gestion des livres, des utilisateurs et des emprunts. Il a permis d'automatiser des tâches essentielles, comme le suivi des emprunts et des retards, tout en garantissant une gestion efficace des ressources. Bien qu'il soit encore en développement, ce système constitue une base solide pour une bibliothèque numérique automatisée.