# Rapport du projet:

## 1. Étude et Correctifs du Code Fourni

Le code initial comportait une base très simple avec des classes qui représentaient des éléments d'une bibliothèque tels que des livres, des DVD, des CD, etc. Voici les principales observations du code fourni au début :

- **Manque d'interactivité et d'intégration :**
    - Le code ne contenait qu'un menu minimaliste imprimant des messages sans logique associée.
    - Les classes livre, dvd, cd..., étaient uniquement définies par des attributs, mais sans méthodes pour gérer les actions de la bibliothèque (emprunt, retour, disponibilité…).
- **Correctifs appliqués :**
    - Les objets et classes ont été restructurés pour utiliser un modèle basé sur une base de données, facilitant ainsi la gestion des emprunts et des retours.
    - Les classes initiales comme “livre, dvd, cd” ont été remplacées par une seule classe `Media` plus flexible dans le modèle Django, avec un champ “type_media” pour les différencier.
    - Ajout des méthodes dans la classe Emprunt pour gérer les emprunts et la validation des règles d'emprunt.

## 2. Mise en Place des Fonctionnalités Demandées

**Fonctionnalités développées :**

- **Gestion des emprunts et des retours :**
    - La création d'emprunts avec des règles strictes pour vérifier que :
        - Un membre ne peut pas emprunter plus de 3 médias simultanément.
        - Un membre ayant un emprunt en retard ne peut pas en effectuer de nouveaux.
        - Les jeux de plateau ne peuvent pas être empruntés, uniquement consultés sur place.
    - Retour d’un emprunt avec la mise à jour automatique de la disponibilité du média.
- **Validation des emprunts :**
    - Chaque emprunt est validé via la méthode clean() dans le modèle Emprunt, ce qui garantit que les règles d'emprunt sont respectées.
- **Mise à jour automatique de la disponibilité des médias :**
    - Lors de la création d'un emprunt, le média est automatiquement marqué comme indisponible.
    - Lorsqu'un emprunt est retourné, le média est marqué comme disponible.
- **Tests unitaires :**

- Chaque fonctionnalité a été accompagnée de tests unitaires pour garantir le bon fonctionnement du système. Les tests incluent :
    - La création d'un emprunt.
    - Le retour d'un emprunt.
    - La gestion des règles de limitation à 3 emprunts par membre.

### 3. Stratégie de Tests

Les tests ont été mis en place pour s'assurer que le système fonctionne comme prévu et respecte les contraintes des emprunts. La stratégie de test se divise en trois parties :

**Tests de Création d'Emprunt :**

- Créer un emprunt et vérifier si l'objet est correctement créé dans la base de données.
- Vérifier que le média emprunté devient indisponible après l'emprunt.

**Tests de Retour d'Emprunt :**

- Retourner un emprunt et vérifier que le média devient à nouveau disponible.
- Vérifier que la date de retour est bien enregistrée dans la base de données.

**Tests de Contrainte sur les Emprunts :**

- Limiter à 3 emprunts par membre : Si un membre a déjà 3 emprunts, tout nouvel emprunt doit être rejeté.
- Gérer les emprunts en retard : Si un membre a des emprunts non retournés depuis plus de 7 jours, il ne peut pas en créer de nouveaux.

### 4. Base de Données avec des Données Test

Une base de données SQLite est utilisée, ce qui permet une exécution locale sans nécessiter de serveur de base de données complexe.

**Exemple de données dans la base :**

- **Membres :**
    - Martin Lucas
    - Durand Claire
    - Simon Thomas
- **Médias :**
    - "Les Misérables" de Victor Hugo (livre)
    - "Angélique " de Guillaume Musso (livre)
    - "Sang d’encre" de Robert Galbraith  (cd)
- **Emprunts :**
    - Bernard Julien a emprunté "La Nuit Des Temps".
    - Claire Durand a emprunté "Les Misérables".

## 5. Instructions pour Exécuter le Programme

Pour exécuter ce programme sur n'importe quelle machine, suivez les étapes ci-dessous :

1. **Installation de Python et Django :**
    - Assurez-vous que Python (version 3.8 ou supérieure) est installé.
    - Installez Django en utilisant pip :

      **pip install django**

2. **Clonage du projet :**
    - Clonez ou téléchargez le projet dans un répertoire local:
      **git clone https://github.com/Abdel-74/Django_Project**

3. **Exécuter le serveur :**
    - Pour lancer l'application en local, utilisez la commande suivante :

      **python manage.py runserver**

4. **Accès à l'application :**
    - Ouvrez votre navigateur et allez à l'adresse suivante :

      **http://127.0.0.1:8000**