Rapport sur le projet Notre Livre Notre Média.

1. L’étude et les correctifs du code fourni.

L’ancien code comprenait une bonne structure globale mais n’était pas du tout pratique ni lisible.

Pour commencer j’ai modifié ceci :

```
def menuBibliotheque() :
    print("c'est le menu de l'application des bibliothéquaire")



def menuMembre():
    print("c'est le menu de l'application des membres")
    print("affiche tout")
```

Créer les 2 parties (bibliothécaires et membres) avec des fonctions auraient été beaucoup trop complexe et illisible.
En effet (majoritairement dans la partie bibliothécaires) j’avais besoin d’une interface afin de pouvoir créer mes fichiers de vues, modèles, templates, etc…
Je ne voyais pas cela possible ou alors ce serait beaucoup trop compliqué et illisible, de ce fait, j’ai préféré créer 2 applications à part.

Grâce à cela, que ce soit du côté utilisateur ou développeur, chaque parti détient une interface visuelle organisée et compréhensible.

Ensuite, les modèles de médias :

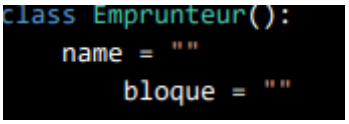
```
class livre():
    name = ""
    auteur = ""
    dateEmprunt = ""
    disponible = ""
    emprunteur = ""

class dvd():
    name = ""
        realisateur = ""
        dateEmprunt = ""
        disponible = ""
    emprunteur = ""


class cd():
    name = ""
    artiste = ""
    dateEmprunt = ""
    disponible = ""
    emprunteur = ""

class jeuDePlateau :
    name = ""
    createur = ""
```

Sur le code d'origine, l'unique modèle plus ou moins bien construit concerne les jeux de plateau. En effet, n'ayant besoin de lui assigner des emprunts le modèle ne requiert qu'un nom et un auteur.
Pour les classes Livre, DVD et CD, plutôt que faire 3 classes différentes comme ci-dessus, j'ai opté pour une classe mère qui contiendrait les 3 médias.

En plus des noms et des auteurs, j'ai ajouté des types de médias ainsi qu'un champ de disponibilité en fonction des emprunts.

L'ancien développeur avait directement ajouté les dates d'emprunts ainsi que les emprunteurs à la classe des médias mais j'ai opté pour une autre manière.

La classe « Emprunteur » :



Cette classe manque cruellement de champs. Pour commencer j'ai supprimé le champ « bloque ». La manière pour qu'un emprunteur ne puisse plus emprunter a été revisitée dans une fonction que l'on verra plus tard dans le dossier.

Tout d'abord la table « Emprunteur » est devenue une table relationnelle. J'ai utilisé les classes membres et médias comme clé étrangère et j'ai ajouté des champs pour les dates d'emprunts et pour les dates retour.

Cette table contient la fonction pour bloquer le membre à partir de 3 emprunts actifs.

2. La mise en place des fonctionnalités demandées.

En tout premier, j'avais pour consigne de sécuriser la base de données par mot de passe, et une autre m'indiquant que la partie « bibliothécaires » ne devait être accessible qu'au personnel de la médiathèque.

Je ne sais pas si j'ai développé la bonne manière, mais je me suis dit qu'en fusionnant les 2 consignes je pouvais régler ceci. J'ai donc créer une page de connexion, qui requiert un utilisateur et un mot de passe afin que seuls les personnes autorisées puissent accéder à la partie des employé.e.s et donc de la base de données. J'ai créé un utilisateur avec les droits super admin pour mon développement, en attente d'autres utilisateurs, afin d'accéder au projet je vous donne les informations de connexion : Nom d'utilisateur : Yann – Mot de passe : ProjetPython1.

Pour les membres, j'ai créé une fonction simple pour afficher la liste des membres en base de données. Elle récupère l'entièreté des membres et les affiches dans une page HTML.

Pour leurs ajouts, une autre page HTML contenant un formulaire avec les champs « nom » et « prénom », à la soumission de ce formulaire les données s'inscrivent directement en base de données et sont retrouvables directement dans la page de la liste.

Pour leurs modifications, en allant sur la page de la liste, à côté de chaque membre il y a un lien nommé « Modifier » qui ouvre une autre page avec un formulaire, celui-ci déjà rempli avec les informations, il est directement possible de modifier ces champs et sauvegarder les modifications avec le bouton.
Les modifications sont instantanées.

Pour leurs suppressions, tout comme la fonction pour modifier, on retrouve un lien « Supprimer » à côté de chaque membre. Au simple clique de ce lien, la suppression est instantanée et vous ramène sur la page de la liste. La suppression fonctionne donc avec l'id de chaque membre.

La liste des médias fonctionne comme la liste des membres avec 2 ajouts supplémentaires. Tout d'abord les types, chaque média est trié par type afin de s'y retrouver plus facilement. Ensuite, à côté de chaque média se trouve les mots « Disponible (en vert) » et « Indisponible (en rouge) ». Cela a pour but d'informer en vitesse élevée, les employé.e.s quant à la disponibilité des médias.

Finalement pour les emprunts, c'est une fonctionnalité principale qui contient plusieurs petites fonctionnalités.

Pour commencer, la création d'emprunts peut se faire via une page et est très intuitive.
Sous forme de formulaire vous avez un champ pour les médias, aucunement besoin d'écrire le nom du média puisque en cliquant sur ce champ une liste déroulante apparait avec les médias uniquement DISPONIBLE.
Second champ pour les membres, de la même manière que pour les médias, au clique de celui-ci une liste déroulante apparait avec les membres AUTORISÉS à emprunter. Ceux qui ont des retards de retour, déjà 3 emprunts actifs ne seront pas disponibles.

3. Stratégie de tests.

Cette partie est un peu plus délicate. Pour chacune des fonctions, des tests unitaires sont construits dans un fichier externe, après multiple changement, correction, (dolipranes et migraines), je n'arrive pas à faire fonctionner mes tests.

Ils sont bien construits, les imports sont présents et la librairie (Pytest) que j'utilise est bien installée.

Au début j'avais des problèmes liés à mon environnement virtuel, après suppression de celui-ci puis création d'un nouvel environnement ainsi que la réinstallation de toutes mes dépendances nécessaires ce problème était réglé.

Ensuite Pytest n'arrivait pas à trouver les paramètres Django, j'ai donc dû refaire une configuration de paramètres fin que cela fonctionne.

Pour finir, le dernier problème que je n'arrive malheureusement pas à résoudre concerne ma table « Membre ». La console m'indique qu'aucune table de ce nom n'existe dans la base de données. Après multiples recherche dans la base de données et de commandes en console, ma table était bien présente et le nom de la table était également bien écrit.

Cela faisait 2 jours que je bloquais sur ce problème, malgré mes nombreuses heures de recherches sur internet, et même l'utilisation d'une IA je n'ai malheureusement pas réussi à le régler. Le fichier de tests est bien présent et rempli mais à mon désespoir les tests, eux, ne fonctionnent pas.

4. Base de données avec des données test.

La base de données contient de multiples données de tests, que ce soit pour les membres, les médias et même les emprunts.

5. Instruction d'utilisation

Pour utiliser ce programme facilement, et sans prérequis, il est possible d'aller sur mon lien **GitHUb** : https://github.com/Teakz9/N-L-N-M, de cliquer sur le petit bouton vert « Code » et de cliquer sur « Download ZIP ». Cela vous permet de télécharger tout le dossier du projet sans passer par des commandes Git.

Pour son utilisation et naviguer à travers le site, je ne considère pas qu'un IDE (Environnement de Développement Intégré) soit un prérequis, donc il faudra :

- Ouvrir le projet avec un IDE
- Ouvrir la console et taper les commandes suivantes :
    - 'cd mediatheque' afin de naviguer dans le dossier contenant le fichier serveur, puis :
        - 'python manage.py runserver'

Ces 2 commandes permettront de lancer le programme sur un serveur et d'accéder à toutes les fonctionnalités qu'il possède.