1. Thème de l'application : "Musify"

L'application "Musify" sera une plateforme de musique permettant aux utilisateurs de découvrir, écouter et gérer leurs morceaux préférés. Elle exploitera les données de l'API Deezer pour récupérer des informations sur les titres, artistes, albums, et plus encore. L'application sera divisée en deux parties principales : une partie publique pour les utilisateurs non connectés et une partie privée avec des fonctionnalités étendues pour les utilisateurs inscrits.

---

2. Fonctionnalités de l'application

L'application offrira différentes fonctionnalités basées sur le statut des utilisateurs (connecté ou non).

2.1. Fonctionnalités pour les utilisateurs non connectés

- **Recherche de Musique** : Les utilisateurs peuvent rechercher des morceaux, albums, ou artistes via une barre de recherche.
    - *Appel API* : Utilisation de l'endpoint de recherche de l'API Deezer : https://api.deezer.com/search?q={query}.
- **Affichage des résultats de recherche** : Liste de chansons, albums, ou artistes correspondant aux requêtes des utilisateurs.
- **Lecture d'extraits de musique** : Les utilisateurs peuvent écouter un extrait de 30 secondes d'un morceau (fonctionnalité fournie par l'API Deezer).
- **Consultation des albums et artistes** : Détails supplémentaires des albums et artistes récupérés via l'API Deezer.
- **Navigation et découverte** : Possibilité de naviguer dans des listes prédéfinies de musiques populaires ou les derniers titres (en utilisant les appels API pertinents).

2.2. Fonctionnalités pour les utilisateurs connectés

En plus des fonctionnalités disponibles pour les utilisateurs non connectés, les utilisateurs inscrits auront accès à des options supplémentaires :

- **Système de favoris** : Enregistrer des morceaux, albums ou artistes dans une playlist personnelle.
    - *Base de données interne* : Enregistrement des favoris dans une base de données locale pour chaque utilisateur à l'aide de l'ORM Django.
- **Historique d'écoute** : Voir les morceaux récemment écoutés.
- **Création de playlists** : Les utilisateurs peuvent créer et gérer leurs propres playlists.

- **Profil personnalisé** : Une page utilisateur affichant ses playlists, albums, et artistes favoris.
- **Suggestions basées sur les favoris** : L'application pourrait offrir des suggestions personnalisées basées sur les favoris enregistrés.

---

3. Fonctionnement de l'application

3.1. Utilisation de l'API Deezer

L'application exploitera différents endpoints de l'API Deezer pour récupérer et afficher les données musicales. Voici les principales fonctionnalités gérées par l'API :

- **Recherche de musique** : Appel à https://api.deezer.com/search?q={query}.
- **Récupération des informations sur les artistes** : Utilisation de https://api.deezer.com/artist/{id}.
- **Affichage des albums** : Utilisation de https://api.deezer.com/album/{id}.
- **Lecture d'extraits** : Les extraits sont fournis directement par Deezer dans la réponse API.

3.2. Structure de l'application Django

L'application sera organisée autour du framework Django. Voici quelques éléments clés :

- **Modèles (Models)** : Les modèles définis géreront les utilisateurs, leurs favoris, playlists, et historique d'écoute.
    - *Utilisateur* : Stockage des informations personnelles, favoris, playlists.
    - *Playlist* : Chaque utilisateur peut avoir plusieurs playlists contenant des morceaux récupérés via l'API Deezer.
- **Vues (Views)** : Les vues serviront à gérer les différentes pages de l'application.
    - *Page d'accueil* : Présente des titres populaires ou suggérés.
    - *Page de recherche* : Affiche les résultats des requêtes de recherche des utilisateurs.
    - *Page de profil* : Accessible uniquement aux utilisateurs connectés, elle affiche leurs favoris et playlists.
    - *Pages de connexion/inscription* : Permet aux utilisateurs de s'inscrire ou de se connecter.
- **Templates** : Utilisation des templates Django pour afficher les données récupérées via l'API Deezer et les informations locales (playlists, favoris).

3.3. Fonctionnement pour les utilisateurs connectés

- **Enregistrement des favoris** : Lorsqu'un utilisateur clique pour ajouter un morceau à ses favoris, l'application enregistre l'ID du morceau dans une table "Favoris" associée à l'utilisateur, en utilisant l'ORM Django pour interagir avec la base de données.

- **Consultation du profil** : Lors de la consultation de la page de profil, un appel à la base de données interne est effectué pour récupérer les morceaux, albums ou artistes enregistrés par l'utilisateur.
- **Stockage des playlists** : Les playlists personnelles sont stockées dans la base de données. Lorsqu'un utilisateur souhaite écouter un morceau, l'application interroge l'API Deezer pour récupérer les informations à jour concernant ce morceau.

---

4. Base de données : Structure et organisation

La base de données sera principalement utilisée pour stocker des informations locales (utilisateurs, playlists, favoris) et sera gérée via l'ORM de Django.

Modèles de la base de données :

- **Utilisateur** : id, nom, email, mot de passe (haché), date de création.
- **Favoris** : id_utilisateur, id_musique_deezer, type (morceau, album, artiste).
- **Playlists** : id_utilisateur, nom_playlist, date_de_creation, description.
- **Historique d'écoute** : id_utilisateur, id_musique_deezer, date.

L'authentification et la gestion des utilisateurs se feront via le système d'authentification de Django, en utilisant des modèles et formulaires intégrés pour l'inscription et la gestion des sessions utilisateurs.

---

5. Design et charte graphique

L'interface utilisateur sera moderne et intuitive, avec une attention particulière portée à la simplicité et à la fluidité de la navigation.

- **Palette de couleurs** : Couleurs sombres avec des accents lumineux pour un effet moderne et agréable (inspiré par le design de Deezer).
- **Logo** : Un logo simple reflétant le thème musical de l'application.
- **Disposition des pages** : Utilisation de cartes (cards) pour afficher les morceaux et albums, avec des boutons clairs pour écouter, enregistrer en favoris ou ajouter à une playlist.
- **Responsivité** : L'application sera responsive et accessible sur mobile, tablette et ordinateur.

---

6. Organisation du projet et répartition des tâches

Le projet sera divisé en trois grands axes pour faciliter le développement en parallèle :

1. **Appels API et intégration avec Django** :
    - Gestion des appels à l'API Deezer.

- Intégration des données récupérées dans les vues Django.

2. **Back-end (Base de données et gestion des utilisateurs)** :
    - Mise en place de la base de données avec l'ORM de Django.
    - Gestion des utilisateurs (inscription, connexion, favoris, playlists).
3. **Front-end (Design et templates)** :
    - Création des templates Django pour l'affichage des pages.
    - Gestion de l'interface utilisateur avec un design moderne et des interactions fluides.