Cahier des Charges - Projet Musify

1. Contexte et présentation du projet

Le projet **Musify** est une plateforme musicale permettant aux utilisateurs de découvrir, écouter et gérer leurs morceaux préférés en exploitant les données de l'API Deezer. Il s'agit d'une application web où les utilisateurs non connectés pourront rechercher de la musique, écouter des extraits, et consulter des albums et artistes. Les utilisateurs inscrits bénéficieront de fonctionnalités supplémentaires telles que la création de playlists, la gestion des favoris, l'historique d'écoute et des suggestions personnalisées.

2. Objectifs du projet

L'objectif principal de **Musify** est de fournir une interface intuitive et conviviale pour que les utilisateurs puissent :

- Rechercher et découvrir de la nouvelle musique.
- Écouter des extraits de morceaux via l'API Deezer.
- Gérer leurs morceaux favoris et créer des playlists personnalisées.
- Accéder à une expérience utilisateur étendue grâce à la gestion de profil et l'intégration de fonctionnalités supplémentaires pour les utilisateurs connectés.

Le projet vise à offrir une expérience fluide, à la fois sur desktop et mobile, et à rendre la navigation facile et agréable à travers une interface moderne et responsive.

3. Périmètre fonctionnel

3.1 Fonctionnalités pour les utilisateurs non connectés

- **Recherche de musique** : Les utilisateurs peuvent rechercher des morceaux, albums ou artistes via une barre de recherche (appel à l'API Deezer).
- **Consultation des résultats** : Les utilisateurs peuvent consulter les résultats sous forme de liste et accéder aux détails d'un morceau, album ou artiste.
- **Écoute d'extraits** : Les utilisateurs peuvent écouter un extrait de 30 secondes par morceau (fonctionnalité fournie par l'API Deezer).
- **Navigation et découverte** : Parcourir les titres populaires et les nouveautés musicales.

3.2 Fonctionnalités pour les utilisateurs connectés

- **Système de favoris** : Les utilisateurs peuvent enregistrer des morceaux, albums ou artistes dans une playlist personnelle.
- **Création de playlists** : Les utilisateurs peuvent créer, gérer et organiser leurs playlists.

- **Historique d'écoute** : Les utilisateurs peuvent voir l'historique de leurs morceaux récemment écoutés.
- **Suggestions personnalisées** : Des recommandations sont faites en fonction des morceaux et albums favoris de l'utilisateur.

3.3 Administration de l'application (c'est pas sûr que l'on fasse cette étape)

- Gestion des utilisateurs et des playlists via une interface d'administration Django.

4. Contraintes et spécifications

4.1 Contraintes techniques

- **Langage** : Le projet sera développé en **Python** avec le framework **Django** pour le back-end.
- **Base de données** : Utilisation de **MySQL** (via **WAMP**) pour gérer les informations relatives aux utilisateurs, aux favoris, aux playlists et à l'historique d'écoute.
- **API externe** : L'application repose sur les appels à l'**API Deezer** pour la recherche de musique, la récupération des informations des morceaux et l'écoute des extraits.
- **Front-end** : Utilisation de **HTML**, **CSS** et **JavaScript** pour créer une interface responsive. Les templates Django seront utilisés pour le rendu des pages.
- **Outils de développement** : Développement sur **Visual Studio Code (VS Code)**.
- **Collaboration et gestion de projet** : Utilisation de **GitHub** pour le contrôle de version et **Trello** pour la gestion des tâches.
- **Communication** : Utilisation de **Discord** pour la coordination entre les membres de l'équipe.

4.2 Contraintes fonctionnelles

- **Performance** : L'application doit pouvoir charger les résultats de recherche et les playlists rapidement. L'intégration de l'API Deezer doit être fluide.
- **Sécurité** : Utilisation des standards de sécurité pour protéger les informations des utilisateurs, notamment lors de l'inscription et de la connexion.
- **Responsivité** : L'application doit être responsive et s'adapter à divers formats (desktop, tablette, mobile).
- **Gestion des erreurs** : Mise en place d'un système de gestion des erreurs et des messages utilisateurs pour assurer une navigation sans accroc (erreurs API, connexions invalides, etc.).

4.3 Contraintes organisationnelles

- **Nombre de développeurs** : Le projet sera développé par deux personnes.
- **Planning** : Le projet devra être découpé en sprints, chaque sprint couvrant une fonctionnalité ou un ensemble de fonctionnalités.

- **Documentation** : Il est important de documenter chaque étape du projet (code et processus) afin de garantir une bonne maintenabilité.

4.4 Contraintes légales

- **Droit d'utilisation** : L'utilisation des extraits musicaux est limitée aux fonctionnalités offertes par l'API Deezer (30 secondes d'extrait). Il faudra respecter les termes d'utilisation de l'API Deezer.

5. Ressources nécessaires

5.1 Humaines

- **Deux développeurs** ayant les compétences nécessaires pour le développement web (Python, Django, MySQL, HTML, CSS, JavaScript) et la gestion des APIs.

5.2 Techniques

- **Environnement de développement** :
    - **Visual Studio Code** : Environnement de développement utilisé pour le projet.
    - **WAMP** : Serveur local pour tester l'application avec **MySQL** comme base de données.
    - **Django** : Framework back-end utilisé pour construire l'application.
    - **MySQL** : Base de données pour stocker les informations des utilisateurs, playlists, et favoris.
    - **API Deezer** : Source de données externe pour la recherche musicale et les extraits.
- **Hébergement et gestion des versions** :
    - **GitHub** : Contrôle de version et collaboration via des branches pour la gestion du code source.

5.3 Outils de gestion de projet

- **Trello** : Pour organiser les tâches en sprints, suivre l'avancement, et attribuer les tâches aux développeurs.
- **Discord** : Pour la communication en temps réel et la gestion des discussions relatives au projet.

6. Livrables

- **Livrable 1** : Cahier des charges (décrit ici).
- **Livrable 2** : Code source de l'application hébergé sur GitHub.
- **Livrable 3** : Documentation technique (README et guides d'installation).
- **Livrable 4** : Application fonctionnelle.

- **Livrable 5** : Plan de tests et rapport de validation des fonctionnalités.

7. Organisation du projet et répartition des tâches

7.1 Phases du projet

1. **Phase 1** : Initialisation du projet
    - Création de l'environnement de développement, configuration de Trello et mise en place de la base du projet sur GitHub.
2. **Phase 2** : Conception et implémentation du back-end
    - Modélisation de la base de données.
    - Mise en place des modèles Django (Utilisateurs, Favoris, Playlists, Historique).
    - Intégration des appels à l'API Deezer.
3. **Phase 3** : Développement des fonctionnalités principales
    - Recherche de musique, gestion des playlists et favoris, affichage des résultats.
4. **Phase 4** : Développement du front-end
    - Création des templates HTML, intégration du CSS, mise en place de la responsivité.
5. **Phase 5** : Tests et débogage
    - Tests des fonctionnalités, gestion des erreurs API, tests unitaires et validation du projet.
6. **Phase 6** : Déploiement et documentation
    - Déploiement sur un serveur, finalisation de la documentation technique.

7.2 Répartition des tâches

- **Développeur 1** : Gestion du back-end, intégration API, base de données, sécurité.
- **Développeur 2** : Gestion du front-end, création des templates, responsivité, interaction utilisateur.

8. Suivi et validation

Le projet sera suivi à travers **Trello** pour le découpage des tâches et l'avancement des fonctionnalités. Les points de validation seront définis à la fin de chaque phase du projet, avec une révision des fonctionnalités clés sur **Discord** et une revue de code sur **GitHub**.