CAHIER DES CHARGES

Application de synchronisation de playlists musicales

- BeatSync App -

1. Présentation du projet

Nom du projet : YouTube to Spotify - BeatSync App
Objectif principal :
Permettre à un utilisateur de **synchroniser automatiquement ses playlists YouTube vers Spotify**, avec la possibilité d'étendre ultérieurement le service à d'autres plateformes (Deezer, etc.).

2. Contexte et motivation

Les utilisateurs de plateformes musicales ont souvent des playlists éparpillées sur différents services. Recréer manuellement une playlist d'une plateforme à une autre est fastidieux.
Ce projet vise à simplifier cette opération en automatisant la migration des playlists via les **API officielles** (YouTube Data API, Spotify Web API…).

3. Objectifs fonctionnels

Fonctionnalités principales

1. **Connexion sécurisée via OAuth 2.0**
   - Authentification de l'utilisateur sur Spotify.
   - Stockage temporaire des tokens d'accès.
2. **Lecture des playlists YouTube**
   - Récupération du nom, de la description et de la liste des morceaux d'une playlist publique ou personnelle.
3. **Création d'une playlist Spotify correspondante**
   - Création automatique sur le compte utilisateur.
   - Recherche des morceaux correspondants sur Spotify (par titre / artiste).
   - Ajout des morceaux trouvés.
4. **Interface utilisateur simple et claire (React)**
   - Page d'accueil avec champ d'URL YouTube.
   - Page d'état de la synchronisation (progression, erreurs éventuelles).
5. **Notifications de succès / échec**
   - Message de confirmation une fois la synchronisation terminée.
   - Gestion des cas d'échec (morceaux introuvables, playlists privées, etc.).

Fonctionnalités secondaires (à venir)

- Sélection de plusieurs playlists à la fois.
- Support d’autres plateformes (Deezer, Tidal).
- Synchronisation bidirectionnelle (Spotify → YouTube).

4. Objectifs techniques

Stack technique

- **Frontend :** React.js
- **Backend :** Python (Flask)
- **Tests automatisés :** Pytest + Workflow GitHub Actions
- **Tests manuels :** Postman (vérification des endpoints REST (authentification, création de playlists, envoi de requêtes API)) et Interface utilisateur (tests manuels de parcours complets sur le frontend React)
- **APIs utilisées :**
  - YouTube Data API v3
  - Spotify Web API
- **Hébergement prévu :**
  - Frontend → Vercel
  - Backend → Render ou Railway

Architecture

- **Frontend (React)** : interface utilisateur, requêtes REST vers le backend.
- **Backend (Flask)** :
  - Gestion des appels API (YouTube / Spotify).
  - Conversion et synchronisation des playlists.
  - Gestion de l’authentification et des tokens.
- **Tests automatisés :** via GitHub Actions (CI/CD).

5. Contraintes

- Respect des politiques d’utilisation des API officielles.
- Sécurité des données (aucune conservation durable des tokens).
- Compatibilité navigateur : Chrome, Firefox, Edge.
- Code clair et documenté pour faciliter l’évolution future.

6. Design et ergonomie

- Palette de couleurs apaisante et moderne (vert / rose pastel / blanc).

- Logo circulaire symbolisant la musique et la synchronisation (flèches + disque).
- Animation de cœur battant (“Let’s Beat in Sync”) sur la page d’accueil.

7. Planning prévisionnel

Étape	Description	Statut
Analyse du besoin	Définition du périmètre et des APIs	✅ Fait
Conception technique	Architecture frontend/backend	✅ Fait
Développement backend	Flask + endpoints + tests	✅ Fait
Développement frontend	React + intégration API	🔄 En cours
Workflow CI/CD	Tests automatisés sur GitHub Actions	✅ Fait
Tests utilisateurs	Scénarios réels de synchronisation	⏳ À venir
Déploiement final	Mise en ligne sur Vercel/Render	⏳ À venir

8. Évolutivité et maintenance

Le code est conçu pour être extensible :

- Ajout facile de nouvelles plateformes grâce à une architecture modulaire (adapters).
- Tests unitaires pour assurer la stabilité du code.
- Utilisation de GitHub Actions pour détecter rapidement les régressions.

9. Auteure du projet

Nom : Ruken Dogan
Rôle : En formation Développeur Full-Stack
Statut : Projet individuel d’apprentissage