NLP-Playlist 프로젝트 명세서

- 가칭 Oli (Only your vibe and Inspiration)

1. 개요

1.1 프로젝트명

NLP-Playlist: 자연어 기반 개인별 플레이리스트 생성 웹서비스

1.2 프로젝트 배경 및 필요성

기존 음악 스트리밍 플랫폼은 곡명, 아티스트명, 장르명 등 **정형화된 키워드 중심 검색 방식**만을 제공하고 있습니다. 그러나 사용자는 실제로 "한국 밴드 음악", "아이유 노래", "여름 느낌이 나는 청량한 노래" 등과 같이 **자신의 감성과 조건을 자연어로 표현**하여 음악을 탐색하길 원합니다.

또한 사용자는 개별적으로 자신만의 선곡 리스트(즐겨찾기/보관함/좋아요)를 보유하고 있으므로, 자신이 구성해놓은 음악들 중에서 **조건에 부합하는 곡을 빠르게 찾아 플레이리스트로 구성**할 수 있다면 훨씬 더 **개인화되고 직관적인 음악 감상 경험**을 제공할 수 있습니다.

1.3 프로젝트의 차별성

Oli 프로젝트는 사용자가 **직접 조건을 입력**하여 플레이리스트를 생성할 수 있다는 점에서 기존 서비스와 차별화됩니다. 많은 기존 서비스들이 단순히 사용자의 취향이나 라이브러리를 기반으로 **자동 추천**을 제공하는 데 그칩니다. 반면 Oli는 사용자가 직접 **세부 조건을 입력**하여 **정밀하게 맞춤화 된 검색**을 수행하고, 그 결과를 **개인별 맞춤 플레이리스트**로 즉시 제공함으로써 **완전히 개인화된 음악 탐색 경험**을 제공합니다.

---

2. 프로젝트 목표

본 프로젝트는 사용자가 구축한 개인별 음악 라이브러리(DB)를 기반으로, 입력된 자연어 질의(조건)를 이해하고 조건에 부합하는 곡을 자동으로 검색하는 웹서비스를 개발하는 것을 목표로 합니다.

검색된 곡들은 별도의 플레이리스트 형태로 제공되어 사용자가 원하는 조건에 맞는 음악을 편리하게 감상할 수 있도록 지원하여 개인화된 새로운 형태의 플레이리스트를 제공하고자 합니다.

---

3. 시스템 구조 및 주요 기능

3.1 전체 아키텍처 개요

1. **데이터 수집**
    - Spotify API를 이용해 사용자의 라이브러리 곡 메타데이터 수집
    - 수집 항목: 제목, 아티스트, 앨범, 발매연도, 장르, 오디오 피처(danceability, energy, tempo, valence 등)
2. **데이터 전처리 및 저장**
    - 메타데이터를 통합해 로컬 DB(SQLite/PostgreSQL 등)에 저장
    - 텍스트 정보(제목+아티스트+장르 등)와 수치적 오디오 피처를 함께 저장
3. **데이터 분석 및 자동 라벨링**
    - Spotify 오디오 피처 기반으로 군집 분석 (K-means / DBSCAN / GMM 등)
    - 각 곡에 군집 ID 및 감성 라벨(Tag)을 부여
    - PCA / t-SNE를 통해 곡 간 거리 관계를 시각화 및 검증
    - 추출된 군집 라벨을 사용해 곡 메타데이터에 추가 저장
4. **자연어 처리 (NLP)**
    - 사용자의 자연어 질의를 임베딩(문장 벡터)으로 변환 (sentence-transformers 모델 사용)
    - 곡 메타데이터(텍스트+라벨)도 동일한 임베딩 공간에 매핑
    - 질의 벡터와 곡 벡터 간 코사인 유사도 계산 → 상위 유사 곡 추출
5. **웹서비스**
    - 백엔드(FastAPI)에서 검색 및 결과 반환
    - 프론트엔드(React)에서 검색창과 플레이리스트 UI 구현
    - 결과 곡의 앨범 아트, 아티스트, Spotify 링크, 미리듣기 제공

4. 기능 명세

기능명	설명
사용자 라이브러리 등록	Spotify API를 통해 사용자 보관곡/좋아요 리스트 가져오기
데이터베이스 구축	각 곡의 메타데이터 + 오디오 피처 저장
자동 군집화	오디오 피처 기반 군집 분석 → 라벨링 생성
자연어 검색	자연어 질의를 임베딩 후 유사 곡 탐색
플레이리스트 생성	검색 결과를 새로운 플레이리스트로 구성
웹 UI	검색창, 결과 리스트, 플레이리스트 보기 구현

---

5. 데이터 분석 설계

5.1 데이터 구성

- 텍스트 정보: 곡명, 아티스트, 앨범, 장르, 가사(가능시)
- 수치 피처: danceability, energy, tempo, valence, acousticness, instrumentalness 등 (Spotify 오디오 피처)

5.2 라벨링 전략

- 오디오 피처 기반 K-means 클러스터링 (군집 수: 10~15개 가량 실험적으로 결정)
- 각 클러스터를 대표하는 라벨 지정
  (예: '청량', '몽환적', '신나는', '잔잔한', '밴드', '가수명' 등)
- 자연어 질의 시, 질의 임베딩과 각 곡의 메타데이터 임베딩을 함께 고려해 검색

5.3 임베딩 기반 검색

- sentence-transformers로 질의 및 곡 메타데이터 임베딩 생성
- cosine similarity 기반 검색
- 상위 N개 곡 반환 → 프론트엔드로 전송

---

6. 기술 스택

영역	기술
데이터 수집	Spotify Web API
데이터 분석	Python (pandas, scikit-learn, numpy, matplotlib), FAISS
NLP	sentence-transformers, HuggingFace Transformers
DB	SQLite (소규모), PostgreSQL (확장시)
백엔드	Java (Spring Boot)
프론트엔드	React (Next.js 가능)
배포	Vercel (프론트), Render/Heroku (백엔드)

---

7. 개발 흐름 (Java 기반 예시)

1. **데이터 수집 및 DB 구축**
    - Spotify API를 통해 사용자 라이브러리, 보관곡, 좋아요 곡 등 데이터를 수집
    - 곡 메타데이터와 오디오 피처를 정리하여 데이터베이스 구축
2. **데이터 분석 및 라벨링**
    - 오디오 피처 기반 군집 분석 수행 (Python 사용 가능)
    - 각 곡에 라벨링을 적용하여 검색 조건과 연결 가능한 구조 생성
3. **자연어 처리(NLP) 모델 구축**
    - 사용자가 입력한 문장을 임베딩 벡터로 변환
    - FAISS 등 벡터 검색 라이브러리(Python)로 조건 부합 곡 탐색
    - Java 백엔드에서 Python 분석 서버 호출 가능 (REST API)
4. **백엔드 개발 (Java)**
    - **Spring Boot** 기반 Java 서버로 검색 API 구현

- Python 분석 서버 또는 DB와 연동하여 검색 결과 반환
- 사용자 요청 처리, 결과 포맷팅, 프론트엔드 연결 담당

5. **프론트엔드 개발**
    - React를 사용해 검색창, 검색 결과 리스트, 플레이리스트 UI 구현
    - Java 백엔드 API와 연결하여 실시간 검색 결과 제공
6. **통합 테스트 및 배포**
    - 프론트엔드, Java 백엔드, 분석 서버 통합 테스트
    - Vercel, Render, Heroku 등 플랫폼에 배포
    - 문서화 및 사용 가이드 작성

---

8. 기대 효과

- **자연어 기반 검색**으로 사용자가 원하는 조건의 음악을 직관적으로 탐색 가능
- 사용자 개별 라이브러리에 특화된 **개인화 추천 및 분류**
- 음악 데이터 분석(군집화)와 NLP 기반 질의처리를 결합한 새로운 형태의 플레이리스트 생성 방식 제시
- 사용자 질의 기반의 맞춤형 플레이리스트 자동 생성으로 플레이리스트 구성 시간 및 노력 절감
- 사용자 음악 라이브러리 내 음악을 효율적으로 탐색하고 관리