NEXUS Machine Learning Hackathon

Stage-2: Builder Round

VibeCheck Recs

Spotify leads the music streaming industry largely due to the strength of its recommendation system. By combining user listening behavior, playlist context, audio features, and collaborative patterns across millions of users, Spotify delivers highly personalized and engaging music suggestions. Effective recommendations improve discovery, retention, and user satisfaction, making them a core differentiator.

Your job in this round is to design a similar system. Given a Spotify playlist link containing **around 40 tracks**, your system must analyze the playlist to infer the listener's musical preferences and return a ranked list of **10 recommended tracks** that best match the user's taste.

The recommendations should be generated using the information available from the playlist, such as

Submissions will be evaluated by comparing the recommended tracks produced by each system with Spotify's own recommendations for the same playlist. A predefined similarity metric will be used to quantify how closely a team's output aligns with Spotify's recommendations, ensuring objective and consistent evaluation across all submissions.

API Usage (Important)

Using large language models (LLMs) to generate recommendations or using external recommendation systems via APIs is **not prohibited** but will result in **penalties in the final score**.

However, teams **are free to use Spotify's API solely to retrieve audio-related information and metadata (such as artists, genres, popularity, and other available attributes)**about tracks (such as artists, genres, popularity, and audio features). All recommendations must be generated using the team's own logic or models. **(Spotify's recommendation endpoint is not allowed to use and will result in disqualification)**.

Spotify API

Candidates can go to https://developer.spotify.com/ and generate an API for development. This API can provide you with the following **raw features** for a song.

Feature Name	Level	Data Type	Description	Usage
track_name	Track	Categorical (string)	Name of the track	Display / identification
track_duration_ms	Track	Numeric (integer)	Duration of the track in milliseconds	Basic temporal info
track_explicit	Track	Boolean	Indicates whether the track is explicit	Content filtering
track_popularity	Track	Numeric (0–100)	Spotify popularity score of the track	Core similarity signal
artist_name	Artist	Categorical (string)	Name of the primary artist	Identification / diversity
artist_popularity	Artist	Numeric (0–100)	Spotify popularity score of the artist	Mainstream vs niche signal
artist_genres	Artist	Multi-label categorical (list)	Genres associated with the artist	Content-based similarity
aggregated_g	Genre	Categorical	Genre counts aggregated	Genre profiling

enres		frequency map	for the track	
playlist_count	Playlist Co-occurrence	Numeric (integer)	Number of playlists containing the track	Collaborative signal
playlist_name	Playlist Co-occurrence	Categorical (string)	Name of each playlist containing the track	Context metadata
playlist_owner	Playlist Co-occurrence	Categorical (string)	Owner of each playlist	Context metadata
playlist_tracks_total	Playlist Co-occurrence	Numeric (integer)	Number of tracks in each playlist	Playlist scale/context

The derived and computed features include track duration in minutes, number of artist genres, primary genre, genre overlap score, popularity gap between artist and track, normalized playlist co-occurrence score, average/min/max playlist size, playlist owner diversity, explicit-content preference rate, and normalized (z-scored) duration or popularity values.

Submission Guidelines

Teams must submit a **GitHub repository** containing all source code. The repository must include a **README** that clearly explains the approach, system design, and instructions to run the application.

The final interface may be implemented as either:

- a **Command Line Interface (CLI)**, or
- a **Web Application**.

Evaluation Methodology

Submissions will be evaluated by measuring how closely a team's recommended tracks align with Spotify's own recommendations for the same playlist.

For each test playlist, Spotify's recommendations will be treated as the reference set. Relevant track information provided by Spotify—such as audio-related attributes and metadata—will be extracted for both the reference tracks and the tracks recommended by each team. Each track is represented as a feature vector for comparison.

To account for unordered recommendation lists, evaluation will be performed using **pairwise cosine similarity** between predicted and reference tracks, followed by an **optimal matching** between the two sets that maximizes overall similarity. The final score for a playlist is computed as the average similarity across all matched pairs.

For accurate and consistent evaluation, teams are **strongly encouraged to output Spotify track IDs (rather than song names or URLs)** as part of their final output.

Evaluation will be conducted on multiple hidden playlists using a fixed and deterministic pipeline to ensure fairness, reproducibility, and consistency across all submissions.

Bonus Points

Teams may earn **bonus points** by implementing the following optional features. Bonus points are awarded independently of the core recommendation score and are intended to reward deeper modeling, system design, and robustness.

1. Explanation Bonus (+2 points)

2 points will be given if the system provides a clear explanation for **why each recommended track was selected**. Explanations may include, but are not limited to:

- similarity in artists or musical style
- closeness in audio features (e.g., tempo, energy, valence)
- proximity in learned or latent embedding space

A simple **JSON or text-based explanation** is sufficient.

Eg: `"explanation": "Artist 'Queen' appears in your playlist. Matches the overall vibe. Popular track in this genre"`

2. Fast Inference Bonus (+2 points)

2 points will be awarded to the top **three** systems that demonstrate the **fastest end-to-end inference time** during evaluation, measured from playlist input to final recommendations.

3. Cold-Start Robustness Bonus (+2 points)

2 points will be awarded if the system maintains strong recommendation quality (<5% drop in similarity score) when **50% of the input playlist tracks are hidden**.

Eg: if a 50 song playlist results in a similarity score of 60%, then for a 25 song playlist, if similarity scores range from 55-65 % then 2 points will be given.

Best of luck