FINAL REPORT

Song recommendation app

Songfox

SONGFOX

GROUP 2

ABHIGNA MALLEPALLY | ANISHKA AGARWAL | EDWIN DANIELS | SACHIN SHARMA

March 2025

Contents

[Executive Summary 3](#_Toc193064009)

[Introduction 4](#_Toc193064010)

[Business Context & Market Need 4](#_Toc193064011)

[Problem Statement 5](#_Toc193064012)

[Project Objectives 6](#_Toc193064013)

[Approach 8](#_Toc193064014)

[Data Engineering & Pipeline 8](#_Toc193064015)

[AI-Powered Recommendation Engine 13](#_Toc193064016)

[System Architecture 22](#_Toc193064017)

[User Experience and Engagement 25](#_Toc193064018)

[Detailed UI design and features 25](#_Toc193064019)

[User Engagement Metrics and Performance Targets 27](#_Toc193064020)

[Experimentation and Results 29](#_Toc193064021)

[Evaluation Methodologies 29](#_Toc193064022)

[Evaluation Metrics 29](#_Toc193064023)

[Continuous Improvement and Model Iteration 32](#_Toc193064024)

[Financial Model & ROI Analysis 32](#_Toc193064025)

[Revenue Streams 32](#_Toc193064026)

[Cost Analysis 33](#_Toc193064027)

[ROI Projection & Breakeven Analysis 35](#_Toc193064028)

[Market Validation & Strategic Positioning 36](#_Toc193064029)

[Future Roadmap 37](#_Toc193064030)

[Expansion & Growth 37](#_Toc193064031)

[AI Model Improvements 37](#_Toc193064032)

[Long-Term Vision 38](#_Toc193064033)

[Conclusion & Final Recommendations 38](#_Toc193064034)

[References 40](#_Toc193064035)

[Appendix 42](#_Toc193064036)

# Executive Summary

The music streaming industry has undergone tremendous growth, with streaming contributing over 67% of the global music industry’s revenue. Despite this success, existing recommendation systems suffer from limitations such as popularity bias, lack of personalization, and the cold-start problem, leading to user dissatisfaction and limited music discovery. SongFox is an AI-powered music recommendation system designed to address these challenges by employing a hybrid recommendation approach that integrates collaborative filtering, content-based filtering, and deep learning techniques. Our project focuses on improving user engagement, optimizing recommendation accuracy, and ensuring sustainable monetization.

SongFox employs cloud-based architecture featuring a React/Next.js front end, a Flask API backend, and PostgreSQL database storage. The system ensures real-time updates by integrating user listening behavior and external metadata from the Spotify API. Key performance indicators (KPIs) include recommendation accuracy, session duration, churn rate, and monetization effectiveness.

SongFox’s key differentiator lies in its multi-faceted recommendation engine, which utilizes both explicit user interactions (likes, skips, listening history) and implicit content features (tempo, energy, mood) to deliver highly personalized recommendations. Unlike traditional models that reinforce mainstream trends, SongFox promotes diverse music discovery by balancing user preferences with exploratory suggestions. Our platform includes an AI-powered chatbot, powered by Google Gemini Flash 2.0, which enhances user engagement by providing dynamic playlist creation, artist discovery insights, and music history knowledge. This interactive feature makes SongFox more than just a recommendation engine—it becomes a digital music assistant tailored to individual user needs.

By balancing technical innovation with business sustainability, SongFox establishes itself as a next-generation platform in the music streaming industry.

# Introduction

## Business Context & Market Need

The global music streaming market has experienced remarkable growth over the past decade. According to IFPI’s Global Music Report, streaming platforms account for ± 65 percent of recorded music revenue worldwide (IFPI 2022). This surge reflects a shift in consumer preferences, driven by widespread smartphone adoption, improved internet access, and on-demand digital content. Platforms such as Spotify, Apple Music, YouTube Music, have emerged as the industry’s major players, collectively serving millions of users and delivering music catalogs containing millions of tracks (MIDiA Research 2021).

Despite these achievements, the way in which these platforms deliver personalized recommendations is pivotal to user engagement and retention. Research shows that over 70 percent of streaming users cite algorithmic discovery as a key avenue for finding new music (Nielsen Music 2020). Machine learning (ML), and artificial intelligence (AI) has been key to enhancing personalization, customizing curated playlists and suggestion feeds. However, existing recommender systems face significant limitations, particularly in music discovery, personalization, and monetization. Many engines rely predominantly on collaborative filtering, reinforcing mainstream hits rather than guiding users to personalized content.

This bias toward globally popular content reduces the diversity of recommendations, a phenomenon sometimes described as the “popularity echo chamber.” Listeners who seek fresh or niche musical experiences are frequently left with repetitive suggestions that fail to help them explore beyond their existing preferences. As a result, music discovery becomes constrained, hindering both user satisfaction and the potential for lesser-known artists to build audiences on major streaming platforms. Additionally, first-time users with limited listening histories suffer from generic default recommendations, a phenomenon known as the cold‑start problem, which impacts their early impressions of the service and can contribute to elevated churn rates.

Monetization further complicates these challenges. While subscription-based revenue models have been adopted globally— Spotify’s premium subscriber base reached 205 million in early 2023 (Spotify 2023), many users, especially in cost-sensitive markets, remain hesitant to pay monthly fees. Consequently, a large portion of listeners opt for ad-supported access, generating a fraction of the revenue per user that premium subscribers do (Deloitte 2022). At the same time, licensing costs for music catalogs have escalated. In addition to paying substantial royalties to major labels, streaming platforms often confront rising expenses in cloud infrastructure, user acquisition, and marketing. These pressures underscore the need for hybrid monetization strategies that balance free-tier engagement and creative revenue streams, including artist promotions, affiliate partnerships, and event-based marketing.

Given these complexities, there is a clear demand for an AI-driven music recommendation system that addresses both user-centric challenges in discovery and personalization, and the business imperative of sustainable revenue. This capstone project seeks to meet that demand by proposing a music recommendation platform designed with advanced algorithms for content curation, a strong emphasis on user engagement, and pathways to revenue generation.

This capstone project, undertaken at Northwestern University, involves building a full-stack music recommendation system that delivers tailored song suggestions to users. The system integrates data from the Spotify API, captures user listening behavior, and employs a hybrid machine learning model to recommend music aligning with each user's tastes.

## Problem Statement

*“How can we enhance the music streaming experience by providing recommendations that cater to individual preferences?”*

Traditional music recommendation engines rely heavily on collaborative filtering, which examines patterns of user engagement (plays, likes, skips) to suggest similar music. While collaborative filtering excels in recommending songs that are already popular among one’s “neighbors,” it can inadvertently trap users in repetitive loops of mainstream content, leaving independent or niche artists underexposed. This approach also struggles in the cold‑start scenario, wherein new users lack any significant listening history to guide personalization. As a result, they often receive generic recommendations, reducing the likelihood that they become active and loyal subscribers.

A closely related concern is the inability of many existing algorithms to accurately profile the actual musical attributes that make certain songs appealing to specific listeners. If a user’s listening choices stem from nuanced preferences—such as a fondness for acoustic, lyric-driven tracks or an affinity for certain cultural or language-specific genres—this might not be well captured in purely collaborative methods. Without a content-based or hybrid mechanism that considers the unique acoustic fingerprint or thematic qualities of songs, the system overlooks hidden commonalities between seemingly disparate tracks. This myopia deprives users of meaningful recommendations that might genuinely broaden their musical horizons.

In addition to these algorithmic gaps, business challenges loom large. The global appetite for subscription‑based music streaming is tempered by pricing barriers, especially outside higher-income markets. Many users opt to remain on free tiers, forcing streaming services to rely on advertising revenue, which can be inconsistent and demands sophisticated ad targeting to remain viable (MIDiA Research 2021). Affiliate event partnerships, sponsored content, and direct artist promotions offer viable alternatives, but integrating them seamlessly into the user experience requires advanced personalization to avoid alienating listeners with excessive or irrelevant ads. The tension between maximizing revenue per user and maintaining a user-friendly platform is a delicate one, particularly in an industry where switching costs are low, and competition is fierce.

## Project Objectives

The project objectives are broken down into three primary aims:

### Objective 1: Develop a Personalized Music Recommendation App

A fundamental step involves the creation of an AI-powered streaming service that tailors song suggestions to individual users. Here, the focus is on user-driven and real-time interactions:

1. Collaborative filtering models will identify shared patterns among users, ensuring that high-level behavioral similarities guide initial recommendations.
2. Content‑based filtering analyzes acoustic features and music attributes—tempo, energy, instrumentation—to align each recommended track with the user’s true preferences.
3. A hybrid ML framework will merge collaborative and content-based insights, allowing the system to continuously learn from new interactions, and user feedback
4. Playlist generation capabilities will not only improve convenience for users but also serve as a testing ground for the system’s personalization. By offering curated or auto‑generated playlists around genres, moods, or themes, the platform cements its role as a user’s day-to-day music destination.

This multi-layered approach aims to reduce the cold‑start problem for new users and address the mainstream bias that limits discovery on existing services.

### Objective 2: Optimize User Engagement & Retention

To cement a music streaming platform as a daily habit rather than a passing novelty, it is essential to retain active users. The second objective focuses on strategies to keep listeners engaged:

1. Interactive feedback loops will ask users to like/dislike tracks. This direct input is assimilated by the recommendation engine to refine future suggestions, giving individuals a sense of control and fostering loyalty.
2. Quantifiable metrics such as session duration, daily active users (DAU), and churn rates will guide iterative improvements. The system will hypothesize that improved personalization correlates with higher session lengths and lower monthly churn.
3. Special emphasis on new user onboarding addresses the cold-start dilemma by gathering essential preference data early—through short surveys or demographic signals.

### Objective 3: Propose a Sustainable Monetization Strategy

The third objective revolves around ensuring financial viability. Without a diversified revenue model, streaming platforms often struggle to turn user engagement into profitability:

1. Ad-supported content remains a staple for users unwilling to pay a monthly subscription fee. By leveraging targeted advertising, the system can generate revenue from a broad user base, provided the AI can deliver relevant and minimally intrusive ads.
2. Affordable premium tiers offer an alternative for those seeking an ad‑free environment, higher-quality audio, or exclusive content, appealing especially to more dedicated listeners who demand additional features.
3. Artist promotions open another revenue channel, wherein up-and-coming musicians pay for preferential placement or sponsored playlists. A fair but effective approach ensures that recommended tracks are still a match for the user while giving emerging artists a platform to be discovered.
4. Event Affiliate partnerships, for instance, with concert or festival organizers—further diversify revenue. These collaborations allow the platform to integrate ticket sales or brand sponsorship into the listening experience without undermining user satisfaction.

By effectively balancing user experience and business viability, the project aspires to demonstrate measurable improvements in user engagement (e.g., higher session durations), lower churn rates, and robust revenue generation.

# Approach

## Data Engineering & Pipeline

The foundation of the recommendation system is a robust data pipeline that collects, processes, and stores information from multiple sources. Data sources include Kaggle's 1 million+ Spotify songs and similar public datasets, Spotify Web API, which provides rich metadata on songs and artists, as well as user interaction logs from our application and supplemental artist information. Both the Spotify and user interaction datasets contain 20 columns consisting of 12 audio features along with relevant track and artist information.

The Spotify API is a RESTful service returning JSON metadata for music artists, albums, and tracks (SpotifyDevelopers 2023a). Through endpoints such as “Get Track” and “Get Artist,” the system retrieves track details (title, album, duration), audio features (tempo, key, valence, etc.), and artist attributes (genre tags, popularity), among others (MusicTomorrow 2022c; SpotifyDevelopers 2023b). These calls require authorization and are polled regularly to keep the music catalog updated. Since November 2024 the API has been deprecated.

In addition to third-party data, the application collects user interaction data—implicit feedback signals such as songs a user plays, likes/dislikes, or adds to playlists. Each interaction is timestamped and linked to the user and tracked in the database. These behavioral logs form the core of collaborative filtering data, reflecting user preferences and listening patterns over time. Finally, artist metadata (beyond what Spotify provides) can be incorporated from external sources or the Spotify API itself, e.g., artist biographies or related-artist graphs, to enrich content-based features. All incoming data (whether from API or user activity) is first captured in raw form before undergoing cleaning and transformation.

*Table 1: Column level information of the datasets*

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Description** | **User Data** | **Training Data** |
| User\_name | Name of the user | Checkmark with solid fill |  |
| Artist\_name | Name of the artist | Checkmark with solid fill | Checkmark with solid fill |
| Track\_name | Track or record name | Checkmark with solid fill | Checkmark with solid fill |
| Track\_id | Unique id from Spotify for a track | Checkmark with solid fill | Checkmark with solid fill |
| Popularity | Track popularity (0 to 100) | Checkmark with solid fill | Checkmark with solid fill |
| Year | Year released (2000 to 2023) | Checkmark with solid fill | Checkmark with solid fill |
| Played\_at | Time when the user last played the track | Checkmark with solid fill |  |
| Genre | Genre type of the song | Checkmark with solid fill | Checkmark with solid fill |
| Danceability | Track suitability for dancing (0.0 to 1.0) | Checkmark with solid fill | Checkmark with solid fill |
| Energy | The perceptual measure of intensity and activity (0.0 to 1.0) | Checkmark with solid fill | Checkmark with solid fill |
| Key | The key, the track is in (-1 to -11) | Checkmark with solid fill | Checkmark with solid fill |
| Loudness | Overall loudness of track in decibels (-60 to 0 dB) | Checkmark with solid fill | Checkmark with solid fill |
| Mode | Modality of the track (Major '1'/ Minor '0') | Checkmark with solid fill | Checkmark with solid fill |
| Speechiness | Presence of spoken words in the track | Checkmark with solid fill | Checkmark with solid fill |
| Acousticness | Confidence measure from 0 to 1 of whether the track is acoustic | Checkmark with solid fill | Checkmark with solid fill |
| Instrumentalness | Whether tracks contain vocals (0.0 to 1.0) | Checkmark with solid fill | Checkmark with solid fill |
| Liveness | Presence of audience in the recording (0.0 to 1.0) | Checkmark with solid fill | Checkmark with solid fill |
| Valence | Musical positiveness (0.0 to 1.0) | Checkmark with solid fill | Checkmark with solid fill |
| Tempo | Tempo of the track in beats per minute (BPM) | Checkmark with solid fill | Checkmark with solid fill |
| Time\_signature | Estimated time signature (3 to 7) | Checkmark with solid fill | Checkmark with solid fill |
| Duration\_ms | Duration of track in milliseconds | Checkmark with solid fill | Checkmark with solid fill |

To enhance recommendations with diverse locale information, the system utilizes public datasets (containing lyrics or song language data) in addition to the Kaggle Million Songs Spotify dataset. Language detection is performed using the ‘langdetect’ Python package, or alternatively, the data is processed using the LLaMA 3.1-8B Instruct model, which infers the geographical and cultural relevance of tracks based on artist names. Given an input of songs, artists, and release years, the model generates locale-based metadata, associating each track with its most likely region of origin or influence. The LLaMA model runs batched inference over a large CSV file with over a million records, efficiently predicting regional affinity based on linguistic cues, artist origin, and genre patterns. These inferences supplement existing metadata, allowing the system to provide locale-aware recommendations that enhance user discovery of music aligned with their cultural and regional preferences.

### Exploratory Data Analysis

The purpose of the exploratory data analysis was to gain insight into the user's music listening habits using data from their Spotify playlist. The analysis explored various dimensions such as song genres, popularity, audio features, and temporal listening patterns. Various visualization methods were employed to make data insights more comprehensible, such as bar plots, histograms, scatter plots, line plots, and pie charts. Each visualization aimed to highlight specific aspects of the dataset, from genre distribution to trends over time.

The exploratory data analysis provided a comprehensive view of user music listening habits on Spotify. The analysis identified prevalent genres and preferred artists, with trends in song popularity and audio features such as danceability and loudness. The time-series and cluster analyses revealed listening patterns and preferences, offering insights into peak listening times and distinct music taste patterns. The 1 million dataset analysis further enriched these insights by highlighting historical trends in track releases, genre popularity, and artist success. This detailed exploration helped in understanding user preferences and evolving trends, which informed music recommendations and potential industry strategies.

### Data Preprocessing

Once data is collected, an ETL (Extract, Transform, Load) process ensures it is cleaned and structured for efficient use. In the pre-processing stage, the pipeline addresses any data gaps (e.g., retrieves missing information for genre, years, popularity and languages using Spotify and Llama APIs) and filters out incomplete or noisy records (e.g., removing tracks with missing metadata or filtering obvious spam in user-created playlists). It standardizes fields (converting date/time formats, normalizing text cases) and performs deduplication so each unique song or artist is represented once. Relevant features are extracted and organized—for instance, converting Spotify’s raw audio feature JSON into structured columns for danceability, energy, tempo, etc.

The data is then loaded into a relational database with a schema optimized for queries by the recommendation engine. We designed a schema with separate tables for Users, Liked/ Disliked Tracks, Playlists, and Playlist Songs linked by foreign keys. This normalization avoids data redundancy while supporting complex queries. Indexes are added on key fields used in lookups (such as user ID, track ID, genre) to speed up query performance (DataCamp 2021a).

### Database Architecture

Early in development, a local SQLite database was used for simplicity and zero configuration, enabling quick iteration on the data model. SQLite is lightweight and suitable for initial testing, but has limitations in production, notably concurrency constraints and single-file architecture. As the project scaled, we migrated to PostgreSQL hosted in the cloud (Docker Container in free GCP VM Instance). PostgreSQL is a powerful client-server RDBMS designed for large-scale applications, capable of handling high concurrency and complex queries (DataCamp 2021a; DataCamp 2021b). This migration addressed scalability: whereas SQLite performs adequately up to a few gigabytes and minimal write concurrency, PostgreSQL can manage hundreds of gigabytes to terabytes of data and handle heavy concurrent access. Moving to a cloud-hosted PostgreSQL instance also improves reliability—managed infrastructure provides automated backups, replication, and failover.

The cleaned data is stored in a cloud PostgreSQL database (after initial prototyping with SQLite), which provides reliability and scalable storage for the large volume of streaming data. We configured connection pooling on the Flask backend to efficiently reuse DB connections. Performance optimizations occurred during and after migration: adding indexes on frequently queried columns (e.g., user\_id in the liked/dislike table) and analyzing execution plans to find bottlenecks. PostgreSQL features like parallel query execution and caching help reduce query latency. We also enabled JSONB columns in some tables to store unstructured data (like the raw Spotify JSON), while using relational columns for structured attributes—combining the best of both worlds.

### Feature Engineering

Feature engineering steps were crucial for enabling the models to effectively learn from the data and provide accurate recommendations. As specified below, the following feature engineering methods were performed.

1. Standardization was applied to numerical features to ensure that all features contributed equally to the model. ‘StandardScaler’ was used to transform the numerical features (danceability, energy, loudness, etc.) so that they had a mean of 0 and a standard deviation of 1. This transformation streamlined the input data for the KNN and SVD models to compute distances or similarities without bias towards certain features.
2. Label Encoding was then performed to convert categorical features into numerical values. ‘LabelEncoder’ was used to transform each categorical feature into a set of integers. Each ‘genre’ or ‘artist’ was assigned a unique integer label. This allowed the models to incorporate the categorical variables into their calculations, considering them alongside the numerical features.
3. Combined features were developed to create a unified feature space that included both numerical and categorical data after standardization and label encoding. The combined feature set was then used by the KNN model to find similar songs, and by the SVD model to predict user preferences.
4. Interaction matrix capturing user-song interactions for recommendation was created using ‘coo\_matrix’, where rows represented users, columns represented songs, and the matrix values represented interactions (such as listening history or song popularity). This matrix allowed LightFM to learn relationships between users and items, which was then used to recommend new songs based on user preferences.

## AI-Powered Recommendation Engine

### Hybrid Recommendation System

The app implements a hybrid recommendation system that integrates collaborative filtering, content-based filtering, and embedding-based methods to generate personalized music suggestions. Instead of relying on a single approach, the system merges multiple techniques: it captures collaborative signals from user behavior, incorporates audio features of songs, and learns latent representations (embeddings) for users and items. This enables the recommendation engine to retrieve songs that “sound similar” based on content features while also prioritizing tracks favored by users with similar preferences.



*Figure 1: Recommender systems methods*

#### LightFM (Hybrid Latent Factor Model)

A key component of the system is LightFM, a hybrid recommendation model that combines both collaborative and content-based filtering and learns latent factor embeddings for both users and items while incorporating content-based features. LightFM extends traditional matrix factorization by representing users and songs as feature-driven latent vectors. The model is trained using a sparse user-item interaction matrix and song metadata, such as genre, mood, and locale.

##### Latent Factor Representation

Each user u and item (song) i are represented as latent factor vectors pₙ and qᵢ respectively. These vectors capture user preferences and item attributes in a shared embedding space.

##### Feature Representation

Instead of using only user-item interaction data, LightFM incorporates content-based features. Users and items are represented as a combination of multiple features:

pu​=​∑​f∈Fuwf​xf​, qi​=​∑​g∈Givg​yg​

where:

* Fu​ and Gi are sets of user and item features.
* wf and vg are learnable feature weights.
* xf and yg ​are feature embeddings.

##### Prediction Function

The preference score r̂ᵤᵢ between user u and song i is computed as:  
 r̂ᵤᵢ = pT​u qi​

This dot product measures the alignment between the user's preference vector and the song's feature-driven embedding.

##### Loss Function

LightFM is typically trained using a ranking-based loss function such as Bayesian Personalized Ranking (BPR) or Weighted Approximate-Rank Pairwise (WARP) loss. The general objective is:

min​ ∑​ (u,i,j)∈D  ​log σ(r̂ᵤᵢ−r̂ᵤj​) + λ∣∣Θ∣∣2

where,

* D is the set of observed positive interactions and negative samples.
* σ is the sigmoid function.
* λ∣∣Θ∣∣2 is a regularization term to prevent overfitting.

Cold-Start Handling  
 Since LightFM incorporates content features, it can generate recommendations even for users or items with no prior interactions. By leveraging song metadata such as genre, mood, and locale, the model can infer preferences based on content similarity.

##### Optimization and Training

The embeddings are learned through stochastic gradient descent (SGD) or adaptive methods like Adagrad. The training process involves iteratively updating parameters to minimize the ranking loss and improve recommendation accuracy. During training, LightFM learns an embedding space where similar users and songs are positioned closer together. The dot product of a user vector with a song’s feature-derived vector predicts the likelihood of user engagement with that song. The advantage of LightFM is that it effectively balances collaborative filtering and content-based filtering, making it suitable for handling both cold-start recommendations and established user profiles (Kula 2015).

##### Implementation Details

* **Loss Function:** We utilized the Weighted Approximate-Rank Pairwise (WARP) loss function, which is specifically designed for ranking tasks. WARP optimizes the ranking of items, prioritizing those that are most likely to be of interest to the user.
* **Model Training:** LightFM was trained using stochastic gradient descent (SGD) with adaptive learning rates. The model was iteratively updated to minimize the WARP loss, ensuring that the most relevant items were ranked higher in the recommendation list.
* **Parameter Tuning:** We conducted a series of experiments to fine-tune hyperparameters such as learning rate, the number of components, and the number of epochs. Cross-validation was employed to validate the model’s performance, ensuring that it generalized well to unseen data.

##### Model Results

LightFM delivered robust recommendations that were both personalized and contextually relevant. The hybrid nature of the model allowed it to generate playlists that reflected the user’s genre and artist preferences while introducing new and diverse tracks. The model’s flexibility in handling both collaborative and content-based data made it a versatile tool in our recommendation system, providing a well-rounded listening experience for the users.

##### Advantages

* Handles new items and users well by incorporating content features alongside interaction data.
* Supports implicit feedback using ranking-based loss functions such as WARP, which optimizes for personalized ranking instead of numerical rating prediction.
* Learns a unified representation of users and items in a shared latent space, making it robust for recommendation scenarios with mixed data sources.

##### Trade-offs

* Requires feature engineering to optimize content-based signals. The quality of the recommendations depends on how well features such as genre, mood, and tempo capture meaningful user preferences.
* Computational cost: Training LightFM on large datasets requires optimization to ensure fast inference.
* Interpretability challenges: Since LightFM combines multiple feature sources into a single latent space, recommendations are driven by a combination of explicit and implicit feature interactions, making it harder to provide a simple “why this song was recommended” explanation (Kula 2015).

#### Singular Value Decomposition (SVD) – Collaborative Filtering

To capture collaborative filtering signals, the system employs Singular Value Decomposition (SVD) on the user–song interaction matrix. SVD decomposes the sparse interaction matrix into lower-dimensional representations, learning hidden latent factors that explain variations in user preferences (Koren et al. 2009). These latent factors represent abstract taste dimensions that influence which songs a user is likely to enjoy.

Given an interaction matrix R of size m×n (where m is the number of users and n is the number of songs), SVD decomposes it into three matrices:

R≈UΣVT

where:

* U∈Rm×k represents user latent factors (user preferences).
* Σ∈Rk×k is a diagonal matrix containing singular values, representing the strength of each latent factor.
* VT∈Rk×n represents item (song) latent factors (song characteristics).
* *k* is the number of latent factors, chosen such that *k*≪min⁡(m,n) to achieve dimensionality reduction.

##### Prediction Formula

Once SVD has been computed, the predicted user-song interaction score is given by:

R^ = UΣVT

where R^ is the reconstructed matrix that estimates the missing values in R. Each predicted rating r^ui ​for user *u* and song *i* is computed as:

r^ui ​= ∑j=1k​ Uuj​ Σjj​VjiT​

where:

* Uuj represents the affinity of user *u* for latent factor *j.*
* VjiT represents the strength of latent factor *j* in song *i*.

Σ*j​* scales the importance of factor *j*.

##### Implementation Details

* **Matrix Decomposition:** The interaction matrix was decomposed into three matrices: U (user latent features), Σ (diagonal matrix of singular values), and V^T (item latent features). The decomposition allowed us to capture complex patterns in user preferences and song characteristics.
* **Training Process:** We focused on personalizing recommendations based on the user’s last 50 songs, assuming that recent listening history is a strong indicator of current preferences. The model was trained to minimize the reconstruction error of the interaction matrix, thereby ensuring accurate predictions.
* **Regularization:** To prevent overfitting, we applied L2 regularization to the user and item matrices. This added a penalty for large weights in the latent factors, promoting simpler and more generalizable models.
* **Hyperparameter Tuning:** Key hyperparameters, such as the number of latent factors and regularization strength, were optimized using grid search and cross-validation. This ensured that the model balanced complexity with predictive accuracy.

##### Model Results

The SVD model excelled at providing personalized music recommendations. By focusing on the user’s recent listening history, it was able to recommend songs that closely matched the user’s current taste. The model’s ability to generalize from sparse data (e.g., few interactions) made it highly effective for users with limited listening history. Furthermore, the SVD model was computationally efficient, making it suitable for real-time recommendation tasks.

##### Advantages

* Discovers hidden patterns in user preferences beyond surface-level similarity.
* Generalizes well with sufficient interaction data, allowing the system to recommend songs that users may not have discovered otherwise.

##### Trade-offs

* Cold-start problem: SVD relies on historical user–item interactions, meaning it struggles to recommend new songs or cater to new users without sufficient data.
* Static factorization: SVD assumes a fixed training dataset. Incremental updates for new interactions require recomputing latent factors, which can be computationally expensive (Koren et al. 2009).

#### k-Nearest Neighbors (Content-Based Filtering)

For content-based recommendations, the app uses k-Nearest Neighbors (KNN), which identifies songs with similar audio features based on numerical characteristics such as tempo, valence, energy, and danceability. The system computes the cosine similarity as shown below between feature vectors and retrieves nearest-neighbor songs to recommend to the user (Gallo 2020) .

sim(A,B)=A⋅B​/∥A∥∥B∥

where:

* A and B are feature vectors of two songs (e.g., tempo, valence, energy, danceability).
* ∥A∥ and ∥B∥ are the Euclidean norms of these vectors.

The result ranges from -1 (completely dissimilar) to 1 (identical songs).

##### Implementation Details

* **Distance Metric:** We used the Euclidean distance to measure the similarity between songs in the feature space. The Euclidean distance was chosen for its simplicity and effectiveness in handling continuous variables.
* **Choosing K:** The number of nearest neighbors (k) was a critical hyperparameter. We experimented with different values of k, ranging from 3 to 10, and found that k=5 provided the best trade-off between bias and variance. The model was validated using cross-validation techniques to ensure robustness.
* **Computational Considerations:** Since KNN is computationally intensive during the prediction phase, we implemented optimizations such as KD-trees to speed up the nearest neighbor search. This allowed us to handle the large dataset of 1 million Spotify tracks efficiently.

##### Model Results

The KNN model successfully identified similar songs for any given track, enabling users to explore songs that match their taste. The model performed well in terms of precision and recall, particularly for genres and styles with distinctive audio features. However, the model’s performance was sensitive to the choice of k and the quality of the feature space.

##### Advantages

* + Cold-start resilience: Works even when collaborative data is unavailable.
  + Transparency: Recommendations are based on direct feature similarity, making them easier to explain than latent factor models.

##### Trade-offs

* + Risk of overfitting to user preferences, potentially creating a filter bubble where users only receive very similar songs and lack diversity in recommendations.
  + Computational cost for large-scale similarity calculations. To optimize performance, approximate nearest-neighbor search methods such as Annoy or Faiss can be used to retrieve similar songs in sublinear time (Gallo 2020).

By combining multiple strategies, the system creates a context-aware recommendation experience, where song recommendations can be influenced by listening history, user preferences, and mood-based classification. This method aligns with best practices in modern recommender systems, where blending multiple algorithms typically yields higher accuracy and adaptability compared to single-method approaches (Kula 2015).

### Mood-Based Classification

A unique aspect of the recommendation engine is its mood-based classification system, which categorizes songs into happy, sad, energetic, or romantic based on valence and energy scores.

The classification logic follows a simple rule-based approach:

* + Happy: High valence and danceability (> 0.6) with medium-high energy (0.5–0.9).
  + Sad: Low valence and danceability (< 0.3) and low energy (< 0.5).
  + Energetic: High energy, loudness and tempo (> 0.75, -5, 120).
  + Romantic: Medium valence, high accousticness and low energy (< 0.6).
  + Chill: High accousticness and low energy and danceability (<0.5).
  + Angry: High energy (>0.8) and loudness with low valence (<0.4)
  + Party: High energy, danceability, tempo and valence.

These mood labels influence recommendations by allowing users to filter playlists by mood.

### AI Chat

Our LLM-powered chat feature in SongFox, built using Google Gemini Flash 2.0, revolutionizes the way users interact with music by providing a highly intelligent and dynamic music assistant. This AI-driven chat system is designed to understand natural language queries, enabling users to seamlessly generate personalized playlists, receive highly curated song recommendations, and explore in-depth information about their favorite artists and albums. Unlike traditional music recommendation engines, our chat feature retains context throughout conversations, allowing it to remember user preferences, past interactions, and musical tastes to deliver even more relevant and personalized suggestions over time. Whether a user is looking for new songs based on mood, discovering underrated artists, or diving deep into the history of an album, the AI adapts and responds with insightful, real-time information. Additionally, the system integrates direct Spotify links, making it incredibly convenient for users to instantly listen to the recommended tracks without leaving the platform. With Google Gemini Flash 2.0’s advanced natural language processing capabilities, SongFox ensures that conversations feel intuitive, engaging, and deeply personalized, making music discovery not just easy but truly enjoyable. By bridging the gap between AI and music intelligence, our chat feature transforms how users engage with their favorite tunes, providing an immersive, context-aware, and seamlessly interactive music discovery experience like never before.

## System Architecture

*A diagram of a phone

AI-generated content may be incorrect.Figure 2. System architecture overview*

### Frontend: React/Next.js User Interface

We developed the front-end using React and Next.js, providing an interactive user interface for music discovery. Next.js supports server-side rendering, improving performance and SEO while preserving the single-page application feel. Users see recommended tracks on a homepage feed (fetched via calls like `GET /recommendations?user\_id=123`), can search for songs or artists, and provide feedback (like/skip). The UI logs user interactions and sends them back to the server as JSON, fueling the data pipeline.

Because Next.js can pre-render pages, initial loads are quick, and subsequent interactions happen client-side. CORS policies allow the front-end (e.g., on `myapp.com`) to communicate with the API domain. The interface is fully responsive, working well on mobile or desktop. Interaction events, such as “user X skipped track Y,” are recorded by asynchronous calls to an endpoint like `/log\_interaction`, so the system can continually refine recommendations.

### Backend API: Flask and PostgreSQL

The back-end is a RESTful API built with Flask (Python), coupled with PostgreSQL for data persistence. We organized the Flask code into modules (blueprints) for authentication, recommendation logic, data ingestion, etc. Example endpoints include:

* `GET /recommendations?user\_id`: Returns recommended track IDs/details for users
* `GET /track/{id}`: Returns metadata about the specified track from the DB.
* `POST /feedback`: Logs user feedback (like/dislike) on recommendations.

The Flask layer interacts heavily with our PostgreSQL database, using connection pooling (via psycopg2 or SQLAlchemy). Common read/write operations, such as retrieving a user’s top genres or storing new interaction logs, occur here. Database schemas for users, songs, artists, and interactions are normalized to avoid duplication and are indexed for quick lookups (DataCamp 2021a). This API also integrates with external services (Spotify) for data updates or track previews. Authentication/authorization ensures only valid tokens can access user-specific endpoints, and HTTPS secures data in transit.

### Latency Optimization and API Integration

To keep the user experience responsive, we employ caching at multiple layers. Frequent queries or results—like top-trending songs—can be cached in memory or a distributed cache. We also store previously computed recommendations for short periods, so repeated requests within minutes need not rerun the entire model. We push heavier computations (model training, large analytics) to background workers. For any calls to external APIs (Spotify), we set timeouts and rate-limit to avoid blocking. These approaches reduce round-trip times and keep the platform smooth, even as usage grows.

# User Experience and Engagement

Building a compelling user experience (UX) is critical for maximizing user engagement and retention. This section discusses the design principles, guiding the interface—personalization, interactivity, and accessibility—and outlines the metrics and targets used to measure engagement effectiveness. By aligning UX decisions with clear performance indicators, the platform ensures that users not only enjoy the service but also become satisfied long-term users.

## Detailed UI design and features

Our design approach focuses on personalized content, interactive social features, and offline accessibility—all aimed at enhancing engagement and user satisfaction.

### Personalized Homepage

The SongFox homepage is tailored to each user, allowing them to engage with content at both a global and country-specific level. Users can search for songs, view daily top songs, discover trending tracks in their city, and explore upcoming music events. The system dynamically populates the homepage with relevant recommendations based on user behavior and interaction history. This ensures a personalized experience that makes content discovery effortless. Research has shown that personalized recommendations significantly improve user retention and engagement (Smith, 2021). By presenting trending songs and “For You” suggestions together, the homepage balances real-time trends with individualized discovery, keeping users actively engaged.

### Curated Playlists & Discovery

The Playlist page enables users to generate and save playlists by filtering songs based on genre, artist, locale, and mood. SongFox provides both editorial playlists (hand-picked by experts) and algorithmic mixes based on user preferences, offering a seamless listening experience. Studies indicate that curated playlists drive deeper engagement, with 75% of Spotify users regularly tuning into editorial selections.

To enhance discovery, SongFox’s “Featured” page highlights top artists and the most popular daily songs, ensuring users always have something fresh to explore. Additionally, an “Up Next” queue and autoplay features seamlessly continue playback, reducing idle time and increasing session duration—an approach proven effective in platforms like Netflix’s autoplay system.

### Social Sharing and Community Features

The Chat page serves as an interactive hub where users can ask music-related questions and receive instant responses. This feature makes SongFox more than just a streaming platform—it becomes a space for music discovery, learning, and community engagement. Social interaction is further encouraged through curated viral playlists and insights into what’s currently trending among the SongFox community. Collaborative playlist features have been shown to boost engagement by 20% on platforms like Spotify. By allowing users to share music and collaborate on playlists, SongFox fosters a network effect that strengthens user retention and encourages organic growth.

### Events & Live Music Integration

The Events page provides users with detailed information about music events, ensuring they stay updated on concerts and shows. Integrated with the homepage, this feature allows users to explore events relevant to their location or favorite artists, bridging the gap between digital music discovery and real-world experiences.

### Personalized User Experience

On the Profile page, users can access their playlists and manage their liked/disliked songs. This section ensures that SongFox remains a truly personalized platform, adapting to user preferences over time.

### Offline Mode and Accessibility

Considering users with limited or inconsistent internet connectivity (a key finding in our competitive analysis), the application includes an Offline Mode. Users can download songs or playlists when online and later access them without an internet connection. This feature ensures the service remains accessible anytime, removing connectivity as a barrier to usage. From a UX perspective, offline capability greatly improves reliability and user satisfaction. Research indicates that approximately 70% of users expect apps to work offline, and apps with robust offline features exhibit significantly higher retention (up to 3×) and engagement (up to +45%) than those without (Anderson, 2022). Our implementation (detailed in Section 2) uses local caching to store user-selected songs and playlists for offline playback, with clear UX cues (e.g., an offline toggle or icon) to manage downloads.

By focusing on personalized content, seamless discovery, social interaction, event integration, and offline accessibility, SongFox creates an engaging and user-friendly music experience that keeps listeners coming back and makes the platform more inclusive and ensure that engagement can continue uninterrupted, ultimately supporting better user retention across diverse usage conditions.

## User Engagement Metrics and Performance Targets

To measure the effectiveness of our UX enhancements, we can track several core user engagement metrics. These metrics can provide quantitative insight into how users interact with the platform and inform ongoing UX improvements.

### Session Duration (Avg. Time per Session)

This metric captures how long, on average, a user spends active in the app during a single session. Currently, our goal is to achieve 60+ minutes per session on average. A high session duration indicates that users are finding enough value to continue listening, browsing, or interacting without leaving. In the context of media streaming, long session times are a strong sign of engagement—leading video platforms report that their subscribers watch on the order of ~2 hours per day as “healthy” engagement levels (Netflix, 2022). Our 60-minute target aligns with these industry benchmarks. We use analytics tools to track session lengths and identify drop-off points.

### AI Recommendation Accuracy

It is a critical factor in improving user retention and engagement. A well-tuned recommendation system ensures that users consistently discover songs and playlists that align with their tastes, leading to increased time spent on the platform. Our goal is to achieve 70%+ user approval for recommendations, ensuring that most suggested content resonates with users. A high recommendation accuracy leads to greater user satisfaction and retention, ultimately boosting subscription conversion rates. Since personalized recommendations directly impact daily user engagement, we leverage AI models that analyze user listening patterns, song metadata, and contextual factors to continuously refine the accuracy of recommendations.

### User Stickiness (DAU/MAU Ratio)

User stickiness measures the frequency of return usage, defined as the ratio of Daily Active Users to Monthly Active Users (DAU/MAU). This ratio indicates what portion of monthly users engage with the app on a daily basis and is a key indicator of habitual use. Our goal is to reach a 50% stickiness rate (0.5 DAU/MAU), meaning that on any given day, half of the monthly active user base is using the platform. Reaching this level would put us on par with top-tier engagement platforms, indicating very strong user retention and loyalty. To drive stickiness, our UX strategy focuses on delivering continuous value and reasons to return. Personalized content updates (e.g., fresh daily song recommendations or new trending tracks each day) give users something new each time they open the app.

### Subscription Conversion Rate (Free-to-Paid)

Beyond just usage, a critical business metric is the conversion of free users into paid subscribers. Our platform employs a freemium model, where basic features are free with ads, while premium features (such as ad-free listening, high-quality audio, and offline downloads) are available to paid subscribers. We aim for a 10% conversion rate of active free users to paid subscribers. A well-engaged user base provides a pool of candidates for conversion. Users who are highly engaged and satisfied with the free service are more inclined to see value in subscribing. By tracking engagement metrics, we can iteratively improve the UX. Our ultimate goal is to create an engaging, user-centric platform that not only attracts users but keeps them active and satisfied in the long run.

### Average Revenue Per User (ARPU)

ARPU (Average Revenue Per User) is a key metric for measuring the platform’s financial performance. It reflects the total revenue generated per user, including both ad-supported and premium subscribers. Our goal is to achieve an ARPU of over $5, which supports projected revenues of $540K+ with 100K users. ARPU is influenced by several factors, including subscription conversion rates, engagement-driven ad revenue, and upselling premium features. By optimizing the user experience to increase engagement and retention, we aim to drive higher spending per user, ensuring long-term revenue growth.

# Experimentation and Results

## Evaluation Methodologies

* Offline Evaluation: We use historical data to simulate recommendations and compare them to what the user actually listened to. Metrics such as Precision, Recall can gauge how well the model recovers songs the user liked (NeptuneAI 2021).
* User Feedback: We can directly gather user feedback in the app (explicit likes/dislikes, short surveys). This yields real-time signals on recommendation quality.
* A/B Testing: The best measure of effect on engagement is controlled A/B testing. A subset of users sees the new algorithm (Group A), others see the older version (Group B). We compare approval rates, churn, session length. If Group A outperforms, we confirm causation.
* Real-World Performance Monitoring: We do cohort analysis to see how new user cohorts behave over time, checking for algorithmic drift. We also run exploration strategies to discover new user tastes, learning from successes and failures (AWA 2022).

## Evaluation Metrics

The models were assessed on their ability to deliver accurate, personalized, and contextually relevant song recommendations to users.

### KNN for Song Similarity

* **Precision@K:** We measured the precision of the KNN model by calculating the proportion of recommended songs that were truly similar to the user’s favorite tracks. Precision@5 was used as a key metric, reflecting the model's ability to recommend relevant songs within the top 5 results.
* **Recall@K:** Recall@5 was calculated to determine how well the model retrieved all possible similar songs from the dataset.
* **Results:** The KNN model achieved a Precision@5 of 78% and a Recall@5 of 65%. These results indicate that the model was highly effective in recommending songs that aligned with user preferences.
* **Speed and Efficiency:** The optimizations implemented, such as KD-trees, significantly reduced the computation time, making it feasible to run KNN on a large-scale dataset with over 1 million tracks. However, the model still faced challenges with scalability as the size of the dataset increased.
* **User Feedback:** Users reported a high level of satisfaction with the recommendations, particularly appreciating the ability to discover new songs similar to their favorites. The model's effectiveness in capturing subtle nuances in the audio features contributed to the positive user experience.

### SVD for Personalized Recommendations

* **Root Mean Square Error (RMSE):** The RMSE between the predicted and actual user ratings was used as a primary metric to evaluate the accuracy of the SVD model. A lower RMSE indicates better predictive performance.
* **Top-N Accuracy:** We also assessed the accuracy of the top-N recommendations provided by the SVD model, focusing on how well the model predicted the user’s actual preferences within the top 10 recommended songs.
* **Results:** The SVD model achieved an RMSE of 0.86, which is considered quite competitive in the context of music recommendation systems. This low RMSE suggests that the model was able to accurately predict user preferences based on their listening history. The model’s Top-10 Accuracy was recorded at 82%, reflecting its strong ability to recommend songs that were highly relevant to the user’s current taste
* **Scalability and Efficiency:** The SVD model demonstrated excellent scalability, handling the large dataset efficiently due to its matrix factorization approach. The model was able to generate personalized recommendations in near real-time, making it suitable for dynamic user interactions.
* **User Feedback:** Users appreciated the personalized recommendations, noting that the model effectively captured their evolving music tastes. The SVD model’s ability to adapt to recent listening history made it a favorite among users seeking a fresh and personalized music experiences.

### LightFM for Hybrid Recommendations

* **Mean Reciprocal Rank (MRR):** MRR was used to evaluate the ranking quality of the recommendations. It measures the average of the reciprocal ranks of the relevant items, with higher values indicating better performance.
* **AUC (Area Under the ROC Curve):** AUC was employed to evaluate the model's ability to distinguish between relevant and non-relevant songs for the user. A higher AUC value indicates better classification performance.
* **Results:** The LightFM model achieved an MRR of 0.72, demonstrating its capability to rank the most relevant songs higher in the recommendation list. The AUC for the model was recorded at 0.89, indicating a strong ability to differentiate between songs that the user would like and those they would not.
* **Playlist Diversity:** The hybrid nature of LightFM allowed it to generate playlists that were both diverse and aligned with the user’s preferences, successfully combining familiar and new tracks in a balanced manner.
* **Scalability and Efficiency:** LightFM was highly scalable and performed well even with the large dataset of 1 million tracks. The model's hybrid approach allowed it to effectively leverage both collaborative filtering and content-based features, providing a comprehensive recommendation experience.
* **User Feedback:** Users responded positively to the diversity of the playlists generated by LightFM, noting that the model introduced them to new genres and artists while still reflecting their core preferences. The combination of genre and artist-based recommendations was particularly well-received.

## Continuous Improvement and Model Iteration

Recommender systems thrive on iterative refinement. We schedule regular retraining as new interaction data arrives and incorporate user feedback loops. If music fans show low approval, we need to investigate and fine-tune. We also allow “exploratory” recommendations to occasionally test the user’s openness to new genres. A well-structured experimentation platform (with A/B deployments) will ensure safe rollouts: new algorithm variants must prove they outperform the baseline before we adopt them widely. This approach fosters ongoing innovation and a user-centric, metrics-driven culture.

# Financial Model & ROI Analysis

The analysis details revenue streams, cost structures, and financial projections that guide the path to profitability, including breakeven analysis and user growth assumptions. The overall goal is to validate the feasibility of the project by estimating expected revenues, operational expenses, and return on investment (ROI) over a three-year period.

## Revenue Streams

Monetizing a music streaming platform requires a diversified revenue model that balances affordability with financial sustainability. The revenue strategy combines subscription-based premium services, advertising revenue from free-tier users, artist promotions, and affiliate partnerships.

1. **Ad Revenue:** A freemium model underpins the platform’s monetization strategy, where majority of users access content for free while generating revenue through ad placements. The ad model follows a cost-per-thousand-impressions (CPM) pricing structure, where advertisers pay based on the number of times their ad is shown. With an estimated 90% of users opting for the free tier, and an average CPM of $0.20 per user per month, the potential annual ad revenue for 1 million users reaches approximately $2.16 million.
2. **Premium Subscriptions:** A low-cost subscription model is introduced at $0.99 per month for premium users. This tier offers AI-powered music curation, ad-free listening, and exclusive chatbot access. Given affordability constraints in target markets like Africa and India, the price point remains competitive, aligning with local streaming services such as JioSaavn and Gaana ($1/month in India) (IFPI, 2022). With a projected 10% conversion rate, premium subscriptions contribute $1.2 million annually per 100K premium users.
3. **Artist Promotions:** Independent artists and labels benefit from sponsored promotions, allowing them to showcase their music to targeted audiences. Artists can run promotion campaigns for $50 each, placing their tracks in discovery playlists or featuring them on user dashboards. Assuming 1% of onboarded artists opt for promotions, the model projects $100K annually from 2,000 participating artists.
4. **Affiliate Partnerships:** The platform taps into music-related affiliate partnerships, including music festivals, telecom providers, and brand collaborations. These partnerships provide cross-promotional opportunities where music events and telecom companies bundle services (e.g., free premium subscriptions with data plans). With a modest revenue expectation, affiliate partnerships contribute $50K per year.

By leveraging these revenue streams, the financial model forecasts:

A screen shot of a computer

AI-generated content may be incorrect.

*Figure 3: Revenue Breakdown*

## Cost Analysis

To achieve sustainable growth, the financial model factors in development, infrastructure, licensing, marketing, and operational expenses.

1. **App Development & Maintenance:** Developing and maintaining a robust streaming service incurs costs related to backend and frontend engineering, continuous updates, and bug fixes. Estimated costs for Year 1 are $35K, reflecting initial setup and ongoing improvements.
2. **Cloud Infrastructure & AI Costs:** The platform requires high-performance cloud infrastructure for content delivery, AI recommendations, and chatbot functionalities. Hosting services (e.g., AWS, GCP) and AI-related expenses (e.g., model training, API calls) result in annual cloud costs of $50K.
3. **Licensing & Music Rights:** A significant cost factor is music licensing, ensuring compliance with rights holders and fair compensation for artists. Licensing fees are estimated at $100K per year, covering both independent artists and potential label partnerships.
4. **Marketing & Customer Acquisition Costs (CAC):** To reach the target of 100K users in Year 1, the marketing strategy focuses on organic and paid user acquisition via Social media campaigns, influencer marketing and offline events and partnerships. The estimated customer acquisition cost (CAC) is $0.80 per user, with total Year 1 marketing expenses at $80K.
5. **Operational Costs (Support, Compliance, Legal, Payments):** Operations include customer support, compliance (data protection, regional laws), and transaction fees. These costs amount to $30K annually.
6. **Staffing & Payroll:** A lean team of five core members (including Product Management, Engineering, Marketing, and Business Development) ensures efficient execution. Staffing costs total $250K per year.

A screen shot of a computer

AI-generated content may be incorrect.

*Figure 4: Total Estimated Costs (Year 1)*

## ROI Projection & Breakeven Analysis

To assess the return on investment (ROI) and breakeven timeline, the financial model projects user growth and revenue targets over a three-year period.

1. **User Growth Model:** The platform follows a combined organic + paid marketing approach, aiming for a monthly growth rate of ~5.9%.

A black and blue screen with black text

AI-generated content may be incorrect.

*Figure 5:User Growth Model*

1. **Breakeven Point: Year 2 (~500K Users):** By Year 2, the platform reaches 500K users, marking the breakeven point where revenue covers expenses. This milestone confirms financial sustainability and positions the business for long-term growth.

## Market Validation & Strategic Positioning

The financial model aligns with key market opportunities, particularly in Africa and India, where music streaming adoption is growing rapidly.

### Africa: A High-Growth Market

* + Africa’s music streaming market is projected to grow 3× by 2030 (IFPI, 2023).
  + Local genres (Afrobeats, Amapiano) remain underserved, offering an opportunity for regional focus.

### India: High Demand, Low Localization

* + 200M+ Indian music streamers exist, but global platforms struggle with localization.
  + Services like JioSaavn and Gaana charge $1/month, validating our pricing strategy.

### Challenges & Considerations

* + Affordability Constraints → The model ensures pricing stays within local economic limits (e.g., Nigeria’s $0.50-$1 range).
  + Low Credit Card Penetration (5.5%) → Payments via Google Pay, UPI, and mobile billing address this barrier.

This financial analysis outlines a sustainable revenue model that leverages ads, subscriptions, artist promotions, and partnerships. With controlled costs and an achievable growth trajectory, the platform reaches profitability within two years. By strategically targeting emerging markets, it establishes a competitive position in a rapidly growing industry, ensuring long-term scalability and financial viability.

# Future Roadmap

As SongFox continues to evolve, our future roadmap focuses on scalability, AI advancements, and long-term market positioning. Our goal is to enhance user engagement, introduce innovative monetization strategies, and establish SongFox as a leading music discovery platform in high-growth markets.

## Expansion & Growth

SongFox will execute a regional expansion strategy targeting high-potential markets such as India, Africa, and Southeast Asia to achieve 10 million users. These regions have rapidly growing music streaming audiences but remain underserved in terms of personalized recommendation technology.

Key initiatives include:

* **Localized Content & Partnerships:** Expanding our database with regional music catalogues, collaborating with local artists, and integrating multi-language support.
* **Telecom Bundling & Carrier Partnerships:** Offering free trials and bundled premium plans through telecom providers, reducing payment friction in regions with low credit card penetration.
* **Offline Mode Enhancements:** Optimizing song caching for users with limited internet access, increasing accessibility in emerging markets.

Beyond user growth, SongFox will diversify monetization through:

* **Event-Based Partnerships:** Collaborating with concert promoters, festivals, and music venues to integrate event-based recommendations and ticketing within the app.
* **Merchandise & Fan Engagement:** Partnering with artists to sell exclusive merchandise, direct fan interactions, and artist-sponsored content.

## AI Model Improvements

To enhance the accuracy and contextual relevance of recommendations, SongFox will integrate next-generation AI capabilities, including:

* **Multilingual NLP for Lyrics Analysis:** Expanding our Natural Language Processing (NLP) models to analyze song lyrics across multiple languages, allowing for more context-aware and culturally relevant recommendations.
* **Voice-Based Recommendations:** Implementing voice recognition AI, enabling users to request recommendations via voice commands, create playlists through spoken preferences, and interact with the AI chatbot using natural conversation.
* **Real-Time Sentiment Analysis:** Incorporating emotion recognition algorithms that adjust recommendations based on a user’s mood, inferred from voice tone, listening patterns, and user feedback.

## Long-Term Vision

SongFox aspires to become the leading music discovery and AI-powered recommendation platform in high-growth markets. Our long-term vision includes:

* Becoming the dominant AI-driven music platform in Africa & India, offering an alternative to global competitors that lack local market adaptation.
* Expanding beyond music streaming, integrating podcasts, audiobooks, and live music experiences, evolving into a comprehensive digital entertainment hub.
* Pioneering ethical AI in music recommendations, ensuring fair exposure for independent artists, genre diversity, and bias-free personalization.

By continually innovating and adapting to user needs, SongFox will redefine the future of music discovery, engagement, and monetization.

# Conclusion & Final Recommendations

The development of our music recommendation application, built on the Spotify dataset and user profiles, aimed to create a highly personalized music discovery experience. By leveraging K-Nearest Neighbors (KNN), Singular Value Decomposition (SVD), and LightFM, we developed a system that tailors recommendations based on inferred user preferences. Ensuring high-quality, comprehensive datasets was critical to achieving accuracy and relevance, underscoring the importance of data integrity in machine learning applications.

Each algorithm serves distinct app features: KNN identifies similar songs based on audio features, SVD personalizes recommendations from recent listening history, and LightFM curates playlists aligned to user preferences. By integrating these methods efficiently, we achieved strong performance with minimal computational cost. Future iterations may explore deep learning models and autoencoders to enhance personalization further.

Beyond traditional recommendations, we incorporated locale-based personalization for culturally relevant music, mood-driven playlists that adapt to user emotions, and event-aware recommendations connecting users to trending music and live concerts. Our AI-powered chatbot, using Google Gemini Flash 2.0, enables intuitive music discovery through conversational interactions, helping users generate playlists and refine their listening experience dynamically.

Personalized recommendation systems drive user loyalty, engagement, and brand association while reducing churn. As users interact with SongFox, their preferences continuously refine, leading to increasingly accurate recommendations. The platform's commercialization strategy includes value-based pricing, ad-supported tiers, exclusive content licensing, and strategic partnerships to ensure sustainable growth.

By integrating machine learning, context-aware recommendations, and interactive AI-driven features, we have built a scalable and innovative MVP. With ongoing refinement based on user feedback and industry trends, SongFox aims to become a leading music discovery platform, delivering intelligent, immersive, and personalized experiences worldwide.

# References

Joshi, A. (2023) "Spotify 1Million Tracks." Kaggle. Last modified May 21, 2023. https://www.kaggle.com/datasets/amitanshjoshi/spotify-1million-tracks/data

Deloitte. Digital Media Trends Report 2022. https://www2.deloitte.com/content/dam/Deloitte/tw/Documents/technology-media-telecommunications/2022-digital-media-trends-report-en.pdf

MIDiA Research. Music Subscriber Market Shares 2021. https://www.midiaresearch.com/blog/music-subscriber-market-shares-q2-2021

Nielsen Music. Music 360 Report, 2020. https://www.musicbusinessworldwide.com/files/2021/01/MRC\_Billboard\_YEAR\_END\_2020\_US-Final.pdf

Spotify. Spotify Company Reports and Investor Relations, 2023. https://newsroom.spotify.com/2024-02-06/spotify-reports-fourth-quarter-2023-earnings/

AMT Lab. 2022. “The Necessary Diversification of Pervasive Recommendation Algorithms.” https://amt-lab.org/blog/2022/4/recommendation-algorithms-are-pervasive-now-they-need-to-diversify

MusicAlly. 2021. “Study Explores Gender Bias of Music Recommendation Algorithms.” https://musically.com/2021/04/07/study-explores-gender-bias-of-music-recommendation-algorithms/

AnalyticsIndiaMag 2022a. “Collaborative Filtering vs. Content-Based Filtering for Recommender Systems.” https://analyticsindiamag.com/ai-trends/collaborative-filtering-vs-content-based-filtering-for-recommender-systems/

Mukhopadhyay, S., Kumar, A., Parashar, D., Singh, M. (2024). Enhanced music recommendation systems: A comparative study of content-based filtering and K-Means clustering approaches. Revue d'Intelligence Artificielle, Vol. 38, No. 1, pp. 365-376. https://doi.org/10.18280/ria.340138

DataCamp 2021a. “SQLite vs PostgreSQL: A Detailed Comparison.” DataCamp Blog. https://www.datacamp.com/blog/sqlite-vs-postgresql-detailed-comparison

AWA 2022. “The Impact of AB Testing on User Retention.” AWA Digital Blog. https://www.awa-digital.com/blog/the-impact-of-ab-testing-on-user-retention/

MusicTomorrow 2022b. “How Spotify’s Algorithm Works? A Complete Guide to Spotify Recommendation System (2022).” Music Tomorrow Blog. https://www.music-tomorrow.com/blog/how-spotify-recommendation-system-works-a-complete-guide-2022

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30-37. https://ieeexplore.ieee.org/document/5197422

Pazzani, M. J., & Billsus, D. (2007). Content-based recommendation systems. In The Adaptive Web (pp. 325-341). Springer, Berlin, Heidelberg. https://link.springer.com/chapter/10.1007/978-3-540-72079-9\_10

Lenk, Kadri. 2021. “Mood of Music Streamed on Spotify.” https://www.kadrilenk.com/spotify.

SpotifyDevelopers 2023a. “API Calls: Spotify for Developers.” https://developer.spotify.com/documentation/web-api/concepts/api-calls

NeptuneAI 2021. “Recommender Systems Metrics and Business Metrics.” Neptune Blog. https://neptune.ai/blog/recommender-systems-metrics

Anderson, K. (2022). Offline-first apps: Why they matter for user engagement. UX Research Journal. https://www.uxresearch.com/offline-first](https://www.uxresearch.com/offline-first

IFPI (2022). Global Music Report. https://www.ifpi.org/global-music-report](https://www.ifpi.org/global-music-report

IFPI (2023). Africa Music Market Insights. https://www.ifpi.org/africa-music-market](https://www.ifpi.org/africa-music-market

Mulligan, Mark. 2019. “Why India Matters to Spotify.” MIDiA Research. https://midiaresearch.com/why-india-matters-to-spotify/.

# Appendix

**Code**

The team used Github to store and manage all code for the project.

Here is a link to the repository: <https://github.com/sunnysharma03/msds_NU/tree/main/Capstone>

**Detailed EDA**

Extensive EDA has been performed on the user dataset and the Spotify dataset. The notebook is stored on Github.

Here is the link to the EDA: <https://github.com/sunnysharma03/msds_NU/tree/main/Capstone/EDA>

**UI**

Home Page

A screenshot of a video

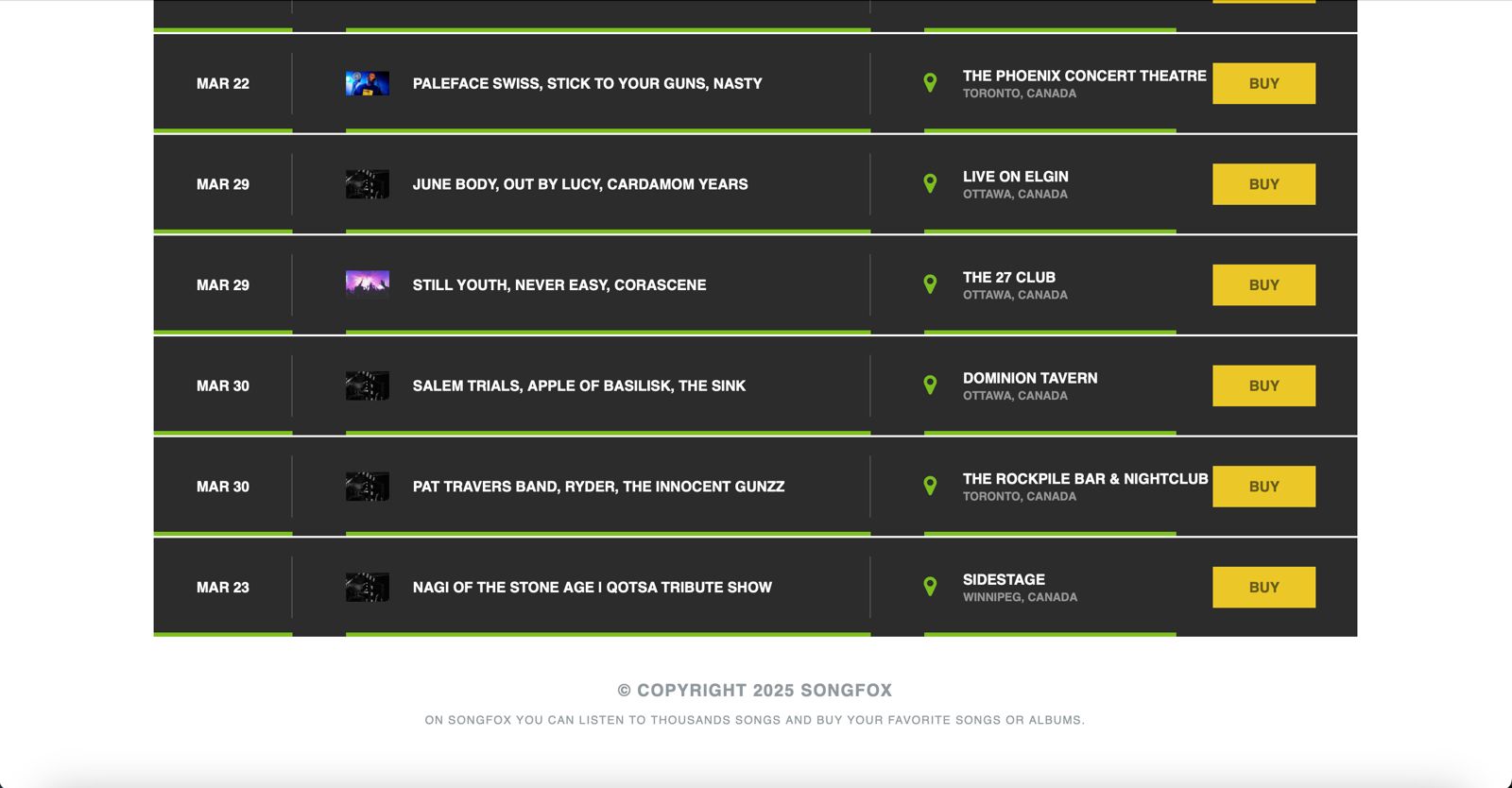
AI-generated content may be incorrect.

A screenshot of a music album

AI-generated content may be incorrect.

A screenshot of a website

AI-generated content may be incorrect.



Playlist Page

A screenshot of a computer

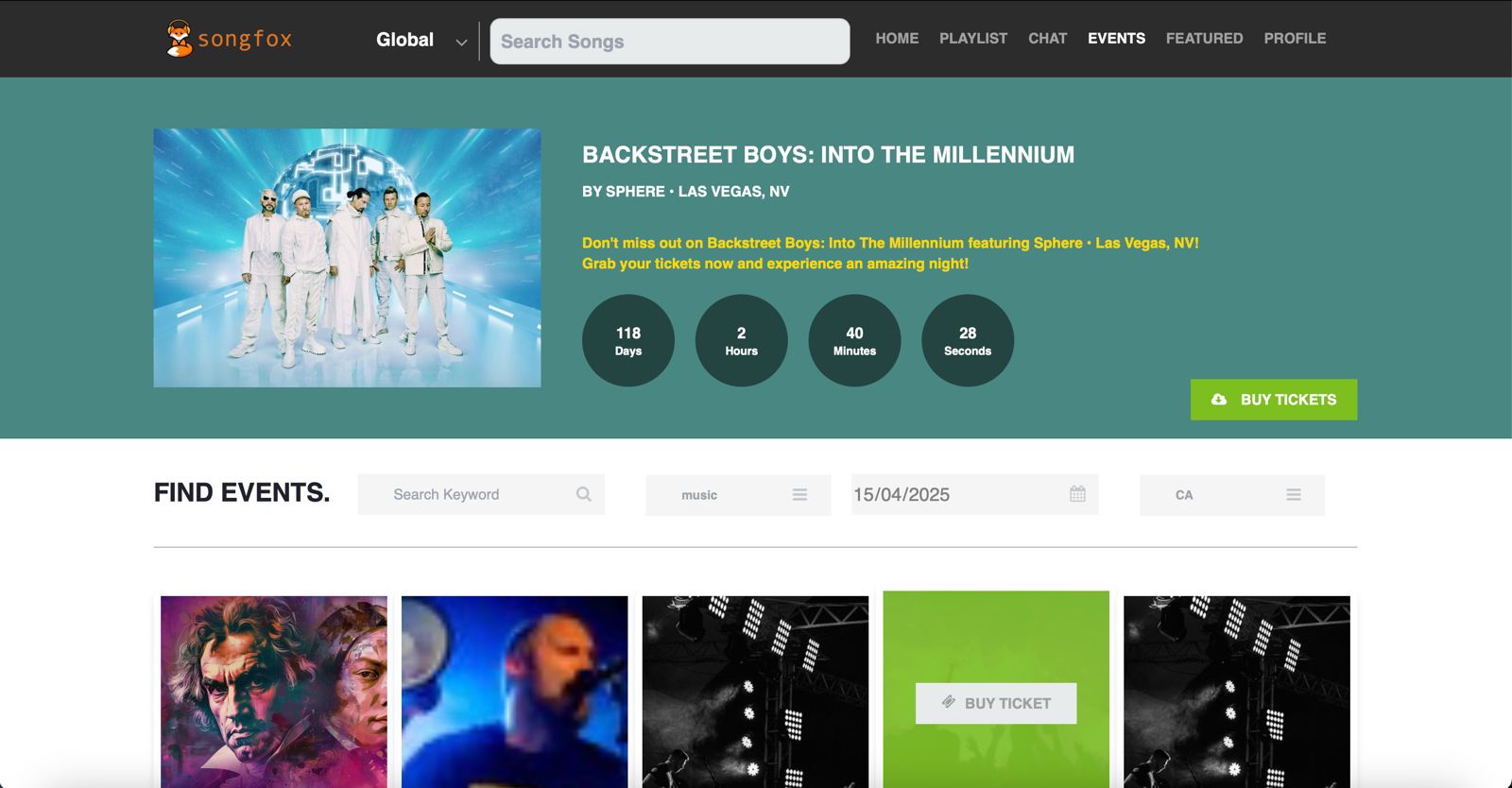
AI-generated content may be incorrect.

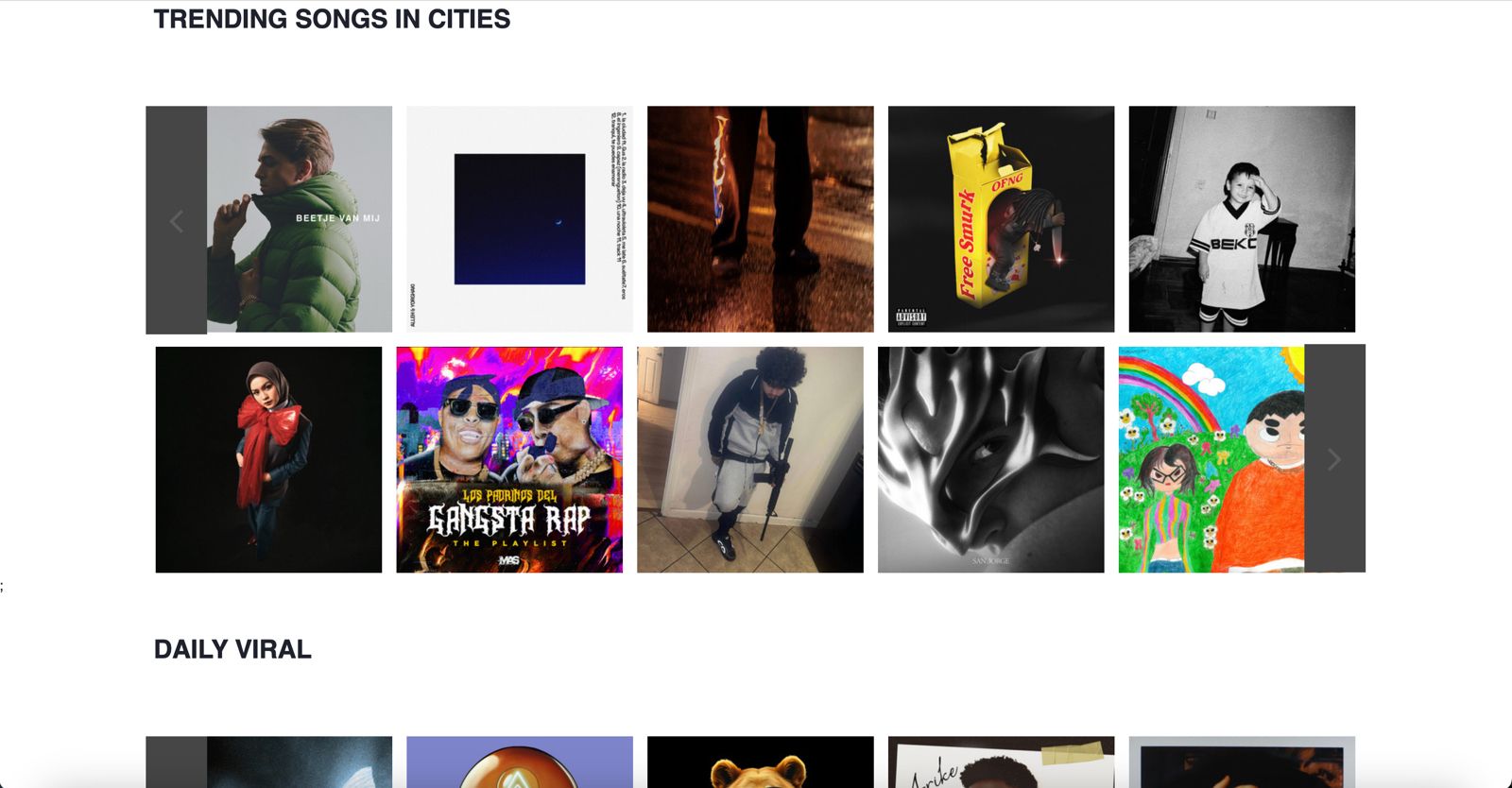
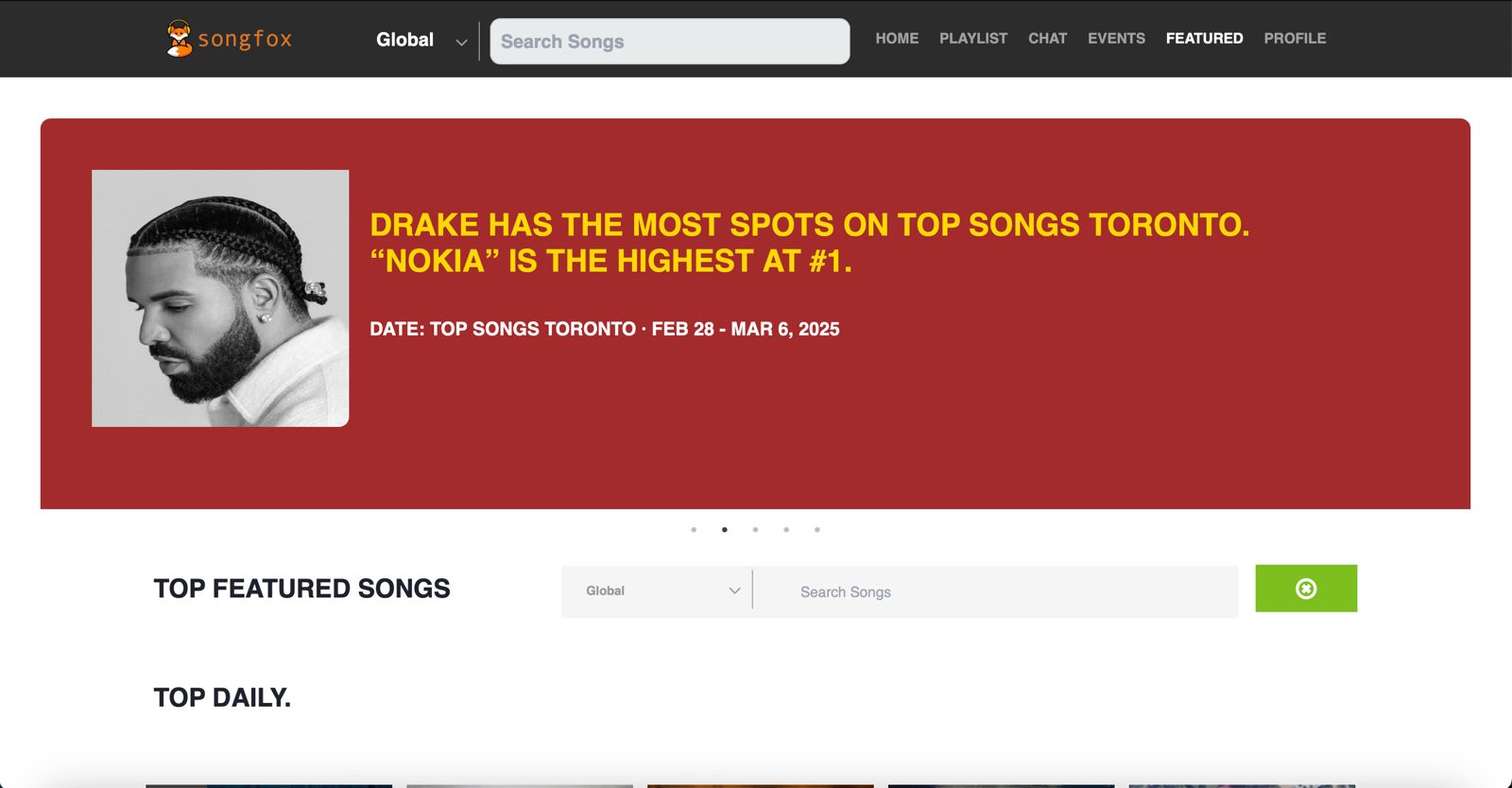
Chat Page

A screenshot of a chat

AI-generated content may be incorrect.

Events Page



Featured Page

User Profile Page

A screenshot of a website

AI-generated content may be incorrect.

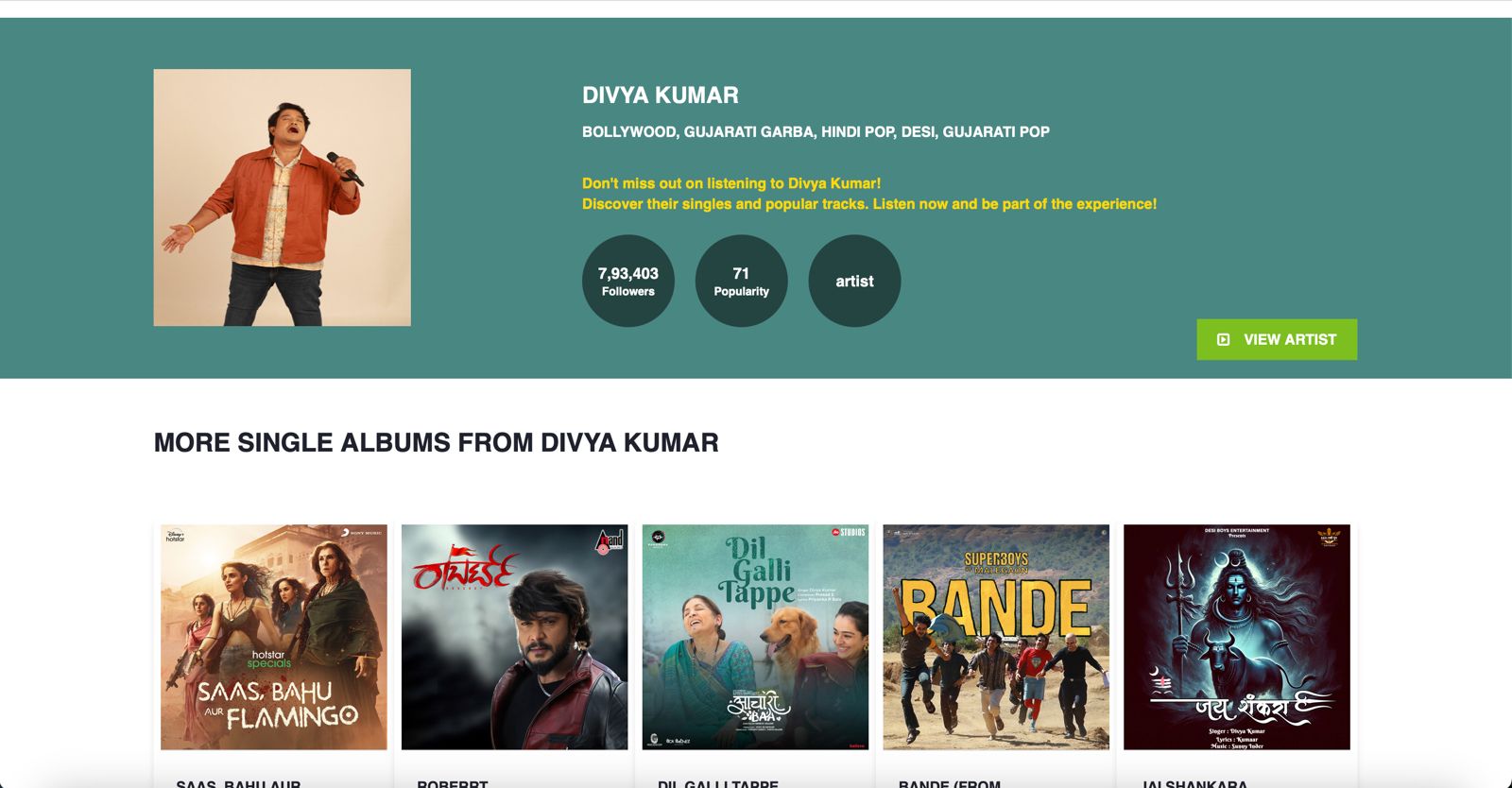
Search Songs

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.



A screenshot of a music player

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.