BABEȘ-BOLYAI UNIVERSITY CLUJ-NAPOCA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
SPECIALIZATION COMPUTER SCIENCE

DIPLOMA THESIS

"Beat With It": Design and Implementation of a Full-Stack Music Recommendations, Insights and Discovery Platform

Scientific supervisor
lect. dr. Vladiela Petrașcu

Author
Eduard Adrian Lupu

2024

ABSTRACT

Music apps provide a wide range of platforms tailored to different aspects of the music experience. TikTok, Shazam, and Spotify are renowned for their unique characteristics and functionalities that shape consumers' interactions with music. Opening from Chapter 2, the first theoretical chapter, we conduct a comprehensive analysis of the prominent characteristics of these platforms in the theoretical section.

This analysis aims to understand how these attributes collectively contribute to Beat With It, an integrated application that simplifies music discovery and access. Furthermore, Beat With It offers valuable insights and recommendations in an engaging and exploratory style.

In Chapter 3, the second theoretical chapter, we examine Music Recommendation Systems, specifically discussing collaborative filtering, content-based filtering, and hybrid systems. We analyze their unique functionalities, advantages, and disadvantages. Subsequently, the focus of this investigation is directed towards Spotify's Music Recommendation System and its Application Programming Interface.

Within Chapter 4, we explore a detailed analysis of the architecture and development process of Beat With It, focusing on practical aspects and including in-depth explanations of design and implementation. Beat With It is an innovative advancement in the field of music technology that combines theoretical knowledge and practical application to provide consumers with an effortless and unified platform for enhancing their music discovery, recommendations, insights and enjoyment.

We set out to turn the theoretical convergence of these platforms into an actuality as we dig deeper into the architecture, design, and execution of Beat With It. This thesis seeks to further music technology by combining theoretical understanding with real-world application development, thereby bringing in a new era of innovation and ease for music lovers everywhere.

Contents

List of Figures

Chapter 1

Introduction

"Where Words Fail, Music Speaks"
- Hans Christian Andersen

Music has become an essential component of our everyday lives in the digital age, enhancing experiences, expressing emotions, and forging bonds between various communities. Concurrently, the rising of music platforms has provided unrivaled accessibility to extensive song archives, allowing users to find, explore, and exchange music in ways never possible before. These platforms are redefining how we engage with music, and TikTok, Shazam, and Spotify stand out as leaders in this regard. They each offer special features and functions that appeal to distinct aspects of the music experience.

Driven by a love of music and an aim to improve user experiences, Beat With It becomes a original platform that combines the features of Spotify, Shazam, and TikTok into a unified, streamlined application. Beat With It was born out of anger at not being able to quickly recognize music used in TikTok videos. While Shazam and similar services are great at pinpointing certain songs, organizing and retrieving data for a large number of songs that are included in different TikTok videos can be difficult. Beat With It simplifies the process of finding and enjoying music from TikTok videos by giving users access to extensive information about song performance, details and direct links to Spotify.

The design, architecture, and implementation of Beat With It are examined in this thesis, together with the underlying technologies, specifications, needs, and features that drive this cutting-edge platform. This thesis seeks to contribute to the changing field of music technology by fusing theoretical understanding of current music applications with hands-on application development, providing a fresh approach to improve users' digital music engagement.

1.1 Structure of the thesis

The process of writing ”Beat With It” and the related thesis was enjoyable and involved a great deal of investigation and learning. I read a ton of research papers to grasp the theoretical foundations. The thesis is structured into three chapters: the first two are theoretical, and the last one is practical, detailing the development of ”Beat With It.”

In the first theoretical chapter, entitled ”State-of-the-Art in Music Applications”, I explored the landscape of music applications, with a focus on industry leaders such as TikTok, Shazam, and Spotify. I analyzed the innovative features that have propelled these apps to the forefront of the music industry.

The second theoretical chapter, which forms the theoretical core of the thesis, delves into Music Recommendation Systems. I examined the critical components and methodologies of these systems, with a detailed case study of Spotify's Music Recommendation System, renowned for its accuracy and sophistication.

The last chapter is the practical segment, where I developed ”Beat With It” by incorporating the theoretical understandings from the previous chapters. The full-stack application's engineering is described in this chapter using the Software Development Life Cycle (SDLC) methodology.

1.2 Declaration of Generative AI and AI-assisted technologies in the writing process

This work is the result of my own effort. I affirm that I have not deliberately solicited or obtained any illegal assistance in the course of carrying out this assignment. I only used QuillBot Premium to boost my academic writing by addressing grammatical errors and improving clarity and coherence via necessary rephrasing. In addition, I used ChatGPT to efficiently arrange my thesis, develop ideas, provide examples, and offer explanations on intricate subjects. After reviewing the content, I edited the changes as needed.

Chapter 2

State-of-the-Art in Music Applications

Nowadays, listening to music is almost exclusively done through streaming services, which provide consumers with never-before-seen access to enormous song libraries, tailored suggestions, and engaging experiences. This section explores the cutting-edge music applications that have completely changed how people find, listen to, and interact with music. Among these, Spotify, Shazam, and TikTok stick out as industry leaders since they each provide distinctive features and functions made to accommodate the wide range of user preferences across the globe.

2.1 The Leaders: TikTok, Shazam, and Spotify

Through short films and viral challenges, TikTok has become a cultural phenomenon, changing how consumers engage with music. TikTok has emerged as a dominant force in music discovery, catapulting songs and artists into viral stardom in a matter of hours with its user-friendly interface and algorithm-driven content recommendations. TikTok is an important player in the music industry landscape because it has established a dynamic platform where creativity flourishes and trends are born by seamlessly mixing music with user-generated content.

Shazam invented the idea of music identification technology, enabling users to recognize songs just by hearing a brief sample. Shazam's sophisticated audio fingerprinting system allows it to recognize millions of songs with accuracy in real-time, giving users fast access to artist bios, lyrics, and streaming platform links. Beyond only identifying songs, Shazam provides useful functions like trending chart discovery and personalized suggestions for music lovers looking to discover new songs to like.

Spotify is a well-known music streaming service that has gained prominence due to its vast repertoire, tailored suggestions, and social media integration capabilities.

Spotify, which puts millions of songs at users' fingertips, provides a customized listening experience based on personal tastes by using machine learning algorithms to create playlists and suggestions that are unique to each user. Furthermore, Spotify's social features promote community and engagement among its user base by enabling users to follow friends, exchange playlists, and find new music through collaborative playlists.

2.2 Exploring Innovative Features and Functionalities

These businesses' creative ideas helped them rise to the position of industry leaders. We should examine some of their key features.

2.2.1 TikTok

According to the official TikTok 2019 Creator Ecology Report [Dou30], "In the era of short videos, the creative forms and genres of short videos are extensive, and any details of life can be used as materials for creative expression." TikTok offers a variety of content materials that might have addressed any topic of interest, such as short, original sitcoms, current events, vlogs, education, unique abilities and more. A complex and rational label classification system is the cornerstone for accurately matching the corresponding audience to the vast amount of content available on the TikTok platform.

Figure 2.1: TikTok's most popular labels. [Zha21]

Using this method, an interest label system tree diagram is constructed from the interest label set. The inclusion and hierarchical link between the data may be seen quite clearly in the tree diagram. The entire TikTok content tree is represented as the root node in this hierarchical label categorization tree. Sports, technology, entertainment, and other general categories are among the meta-classifiers in the first layer below. Assuming a parent node—a node with a branch—represents sports, its child nodes denote more specific sports classifications, such basketball, football, and swimming. Every branch can be further divided into more focused and limited areas. In addition, TikTok promotes content providers to focus on a single area rather than trying to cover a lot of ground.

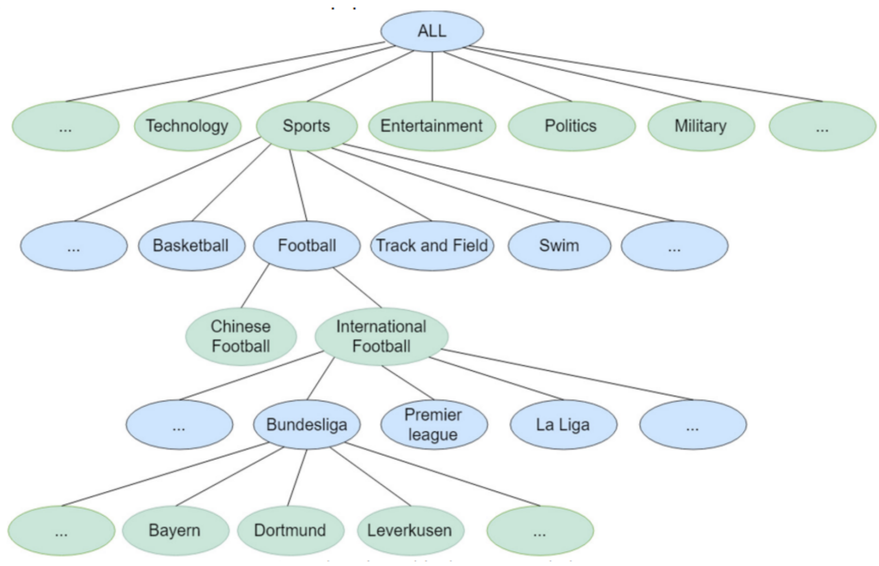


Figure 2.2: TikTok's hierarchical interest label tree diagram. [Zha21]

"The key to the data pool is to get more data through the continuous operation of the original data," according to Yang Fei, the originator of the "data pool marketing" theory [Yan18], in his book. This theory is mirrored in TikTok's algorithmic logic. When releasing fresh content, TikTok employs the partitioned data bucket technique in an effort to reach a larger audience.

Users are divided by the platform into multiple small batch buckets based on a random factor (e.g., groupings with the same ID tail number). The platform will first evaluate the content recommendation effect using one of the small batch buckets when it introduces new content. The amount of views, likes, replays, shares, and completion rates are the four fundamental measures; each has a distinct weight. This short film has the potential to become well-known when the total score of these assessment markers hits a certain threshold. The test is then repeated with this footage overflowing into a medium batch bucket. Content in buckets of varying sizes will receive views at varying intensities.

The emotional resonance that is sought in the broader testing scope forms the basis of these explosive content pieces. This method of content evaluation may work well. Given the vast quantity of content available on the TikTok platform, multilayer screening seems like a very sensible way to ensure that the content is of a high caliber. Ultimately, TikTok can only hold onto users if its content is high quality. Furthermore, as per the findings of Cao Huanhuan's paper [Cao18], TikTok employs a mix of partitioned data buckets and A/B testing to evaluate the efficacy of the recently introduced recommendation technique.

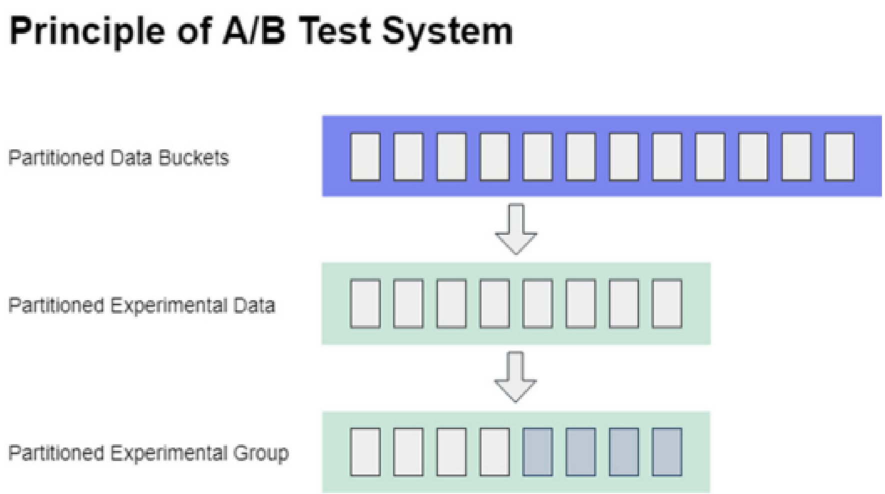


Figure 2.3: Principle of A/B Test System used in TikTok algorithm. [Zha21]

In actuality, this is comparable to the content recommendation and test. All users are first split up into multiple small batch buckets. One of the buckets is then taken out, and half of it is identified as the experimental group. The A/B Test makes use of quantitative techniques to confirm the strategy's viability and efficacy. Start an experiment with 10% of the data, for instance, with 5% representing the experimental group using the new technique and the remaining 5% representing the baseline using the old strategy. Compare the pertinent data from the two groups during the same time frame, in the same test environment, and with the same user group (no discernible attribute differences). They can evaluate the effectiveness of the new plan in this way. [Zha21]

2.2.2 Shazam

Other than mobile phone music recognition, there are numerous other uses for the Shazam algorithm. The ability to detect subtle differences in noise enables us to discern music that may be obscured by a loud narration, like in a radio commercial. However, the method is also extremely quick and can be applied to copyright monitoring at a real-time search speed of more than 1000 times, allowing a small server to monitor a sizable number of media streams. The method can also be used for content-based indexing and cueing in archives and libraries.

Through a technique known as "fingerprinting," replicable hash tokens are extracted from each audio file. The same analysis is applied to "sample" and "database" audio files. A sizable collection of fingerprints taken from the music database is compared to the fingerprints from the unidentified sample. The accuracy of the matches made by the candidate matches is then assessed. The following criteria should be followed while choosing fingerprint characteristics: they should be sufficiently entropic, resilient, translation-invariant, and temporally confined. [Wan03]

In order to prevent distant events from affecting the fingerprint hash, the temporal locality rule recommends that each fingerprint hash be computed using audio samples close to a corresponding point in time. Because of the translation-invariant feature, fingerprint hashes generated from corresponding matched content can be repeated anywhere in an audio file as long as the temporal locality where the data used to compute the hash is present. This makes sense because any part of the original audio file could contain an unidentified sample. Robustness is the ability to produce hashes from a degraded audio copy using the original, pristine database track. [Wan03]

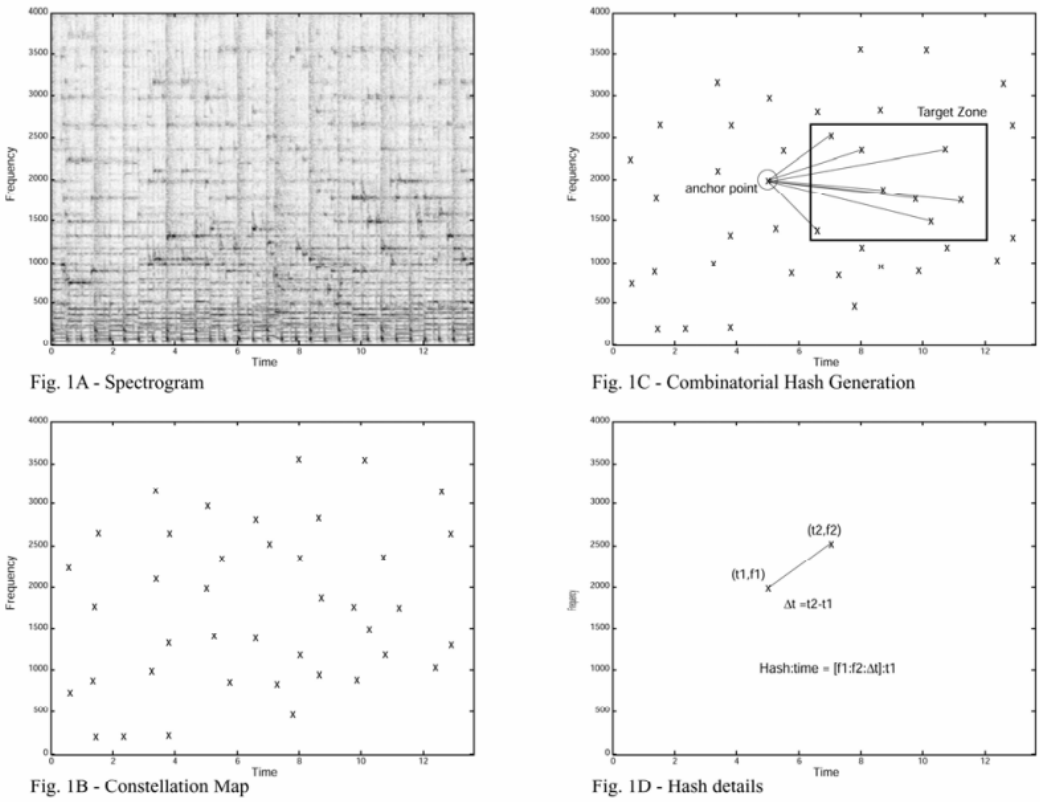


Figure 2.4: Fingerprinting details of an audio. [Wan03]

2.2.3 Spotify

Spotify gave customers unmatched versatility in how they connect with their favorite songs, revolutionizing the way people interact with music. It was born out of a desperate need to stop the widespread illicit music downloading that was happening at the time via torrents. Acting as a central hub for music consumption, Spotify sought to offer a vast song catalog with quick and easy accessibility for its consumers. [FGN22]

The way people interact with music has changed dramatically as a result of streaming services. Gone are the days of private, digital libraries on personal devices and in favor of shared profiles that can be accessed through a cloud-based catalog and provide unmatched customization. For instance, Spotify uses a complex combination of deep learning and machine learning models along with human curation to assess user behavior and musical qualities in order to provide personalized suggestions. [GS19] Spotify builds a thorough ”taste profile” for each listener by combining acoustic, user, contextual, behavioral, and semantic music knowledge data. This allows for automated curation and tailored suggestions. [Só16]

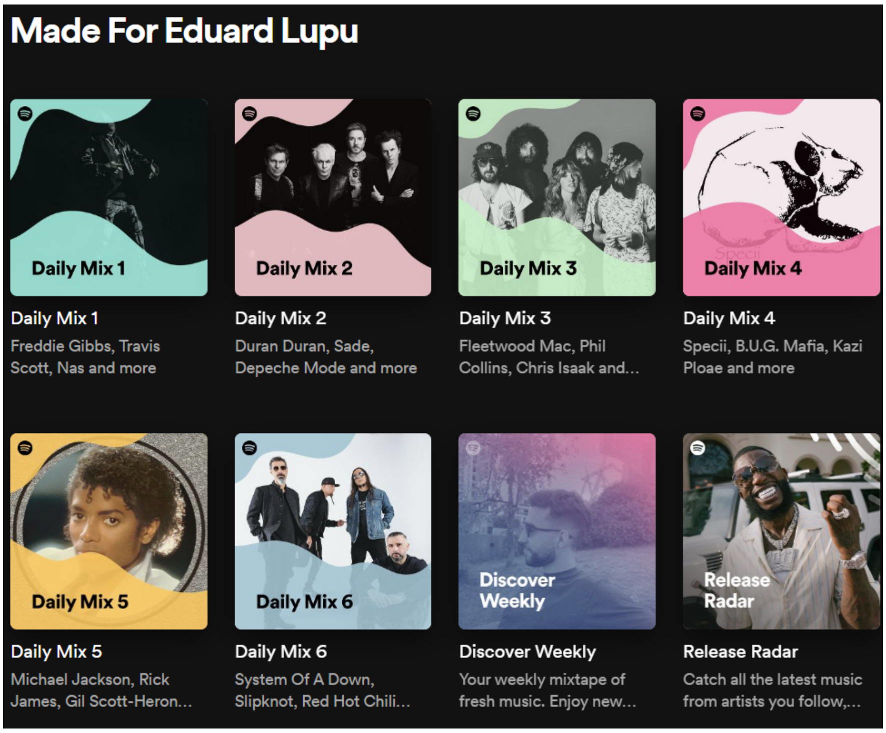


Figure 2.5: Playlists with automated song suggestions based on collaborative filtering and content-based filtering approaches abound on my customized homepage.

Another key feature that makes a Spotify a global leader is the high quality of the music playback and low-latency. Spotify uses both server and peer-to-peer network to achieve great speeds. [KN10]

The mp3 player's UI is comparable to that of desktop players. Tracks can be arranged by users into playlists that they can share via links. The two main methods for finding music are searching and browsing. In addition to browsing, the user can search for songs, albums, or artists. For example, by clicking on an artist's name, a page showcasing all of that artist's albums is displayed to the user. [KN10]

Ogg Vorbis is used to encode audio streams, with a default quality of q5, and a configurable bitrate that averages about 160 kbps. Alternatively, users with a premium membership can opt to receive Ogg Vorbis in q9 quality, which averages around 320 kbps, via a client setting. Peer-to-peer networks and servers both provide access to these kinds of files. Peers do not do reencoding, so a peer who has the q9 version of a track cannot provide it to a peer who wants the q5 version. [KN10]

The Spotify client keeps an eye on the sound card buffers while it is playing music. The client interprets a stutter as having happened if the buffers are underrun. Network effects or a lack of local resources at the client's disposal to quickly decode and encrypted data can both cause stutters. The cause of this kind of local hunger is usually the client host performing other (often I/O-intensive) operations. The protocol is not appropriate for live broadcasts because it is meant to offer on-demand access to a sizable music collection. For example, unless a client receives the entire track, it cannot upload the music. This is because it eliminates the overhead of having to communicate which portions of a track a client has, therefore simplifying the protocol. [KN10]

Spotify employs TCP as opposed to UDP, which is the most widely used transport protocol in streaming apps. First off, designing and implementing protocols is much easier when there is a dependable transport protocol. Second, TCP is beneficial to the network since it enables stateful firewalls with explicit connection signaling and is friendly to itself and other TCP-using applications. Thirdly, the application benefits from the resending of dropped packets since streamed content is shared via a peer-to-peer network. A single TCP connection is utilized between two hosts, and messages are multiplexed over the connection via the application protocol. A client maintains a TCP connection to a Spotify server while it is operating. [KN10]

Spotify low-latency is also due to caching. Caching is the third secret ingredient that makes this company a top leader in music industry.. First of all, it is typical for users to listen to the same song multiple times. By caching the song, this eliminates the need to download it again. Second, the client may provide music files that have been cached in the peer-to-peer overlay. When a client downloads only a portion of a track, that portion will typically be cached because the cache can hold partial tracks. Other players are unable to access encrypted content that has been cached.

The clients' default configuration limits the maximum cache size to 10% of available disk space (not including the cache's size), with a minimum of 50 MB and a maximum of 10 GB. The user can also set the size to be in increments of 1 GB, ranging from 1 to 100 GB. Due to this strategy, the majority of client installs have sizable caches—56 percent of them may hold up to 1000 tracks and have a maximum capacity of 5 GB or more.

A policy known as Least Recently Used (LRU) is used to execute cache eviction. Cache efficiency is not greatly affected by the choice of cache eviction policy when caches are large, according to simulations employing playback logs and cache size data. This is similar to Huang et al.'s [HFC+08] discovery that the PPLive system became significantly more efficient when it switched from a simpler, LRU evaluation procedure to a more involved, weight-based review process. But in their configuration, the items in the cache are movies, and the size of their caches means that a client can only store one or a small number of movies.

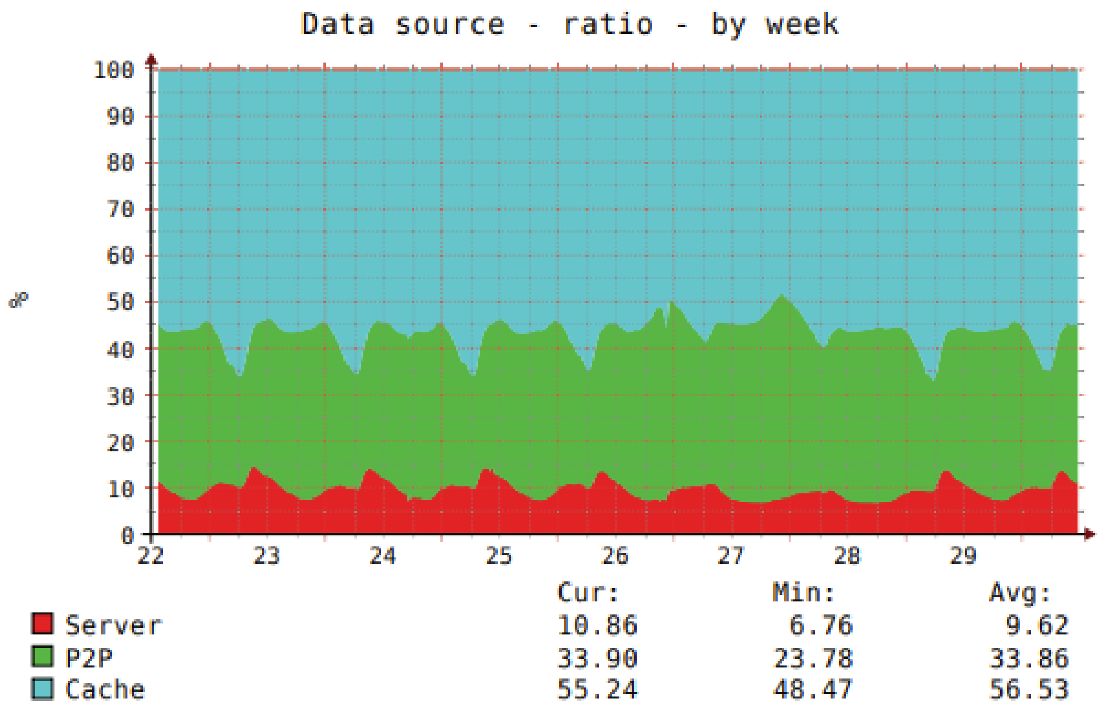


Figure 2.6: Sources of data used by clients [KN10]

Chapter 3

Theoretical analysis on Music Recommendation Systems

Music recommendation systems are sophisticated algorithms and technology created to propose music songs, artists, or albums to listeners according to their preferences, listening history, and other relevant factors. These systems can offer highly customized listening experiences by utilizing machine learning, data mining, and information retrieval techniques. The significance of music recommendation systems rests in their capacity to augment user experience by providing personalized suggestions, thereby facilitating the discovery of new songs and artists that are in line with their preferences. Customization is essential for maintaining customer interest and ensuring their continued happiness and loyalty to music streaming companies. Moreover, efficient recommendation algorithms facilitate content exploration, providing lesser-known artists with opportunities to reach new audiences, and can enhance revenue generation for streaming services by boosting user subscriptions and increasing platform engagement. [Sur22]

The development of music recommendation systems originated in the 1990s with the process of manual curation by DJs and music critics, as well as the utilization of basic rule-based algorithms. During the early 2000s, collaborative filtering became prominent, exemplified by systems such as Last.fm. These systems utilized user data to recommend music based on the preferences of users with similar habits. Simultaneously, content-based filtering algorithms, like Pandora's Music Genome Project, examined audio qualities to suggest comparable tunes. In the 2010s, there was an emergence of hybrid systems that integrated collaborative and content-based methods. Spotify's advanced algorithms serve as a prime example of this. The progress in machine learning and big data analytics has significantly improved these systems, as deep learning models and neural networks have become crucial in comprehending intricate consumer preferences. [Sur22]

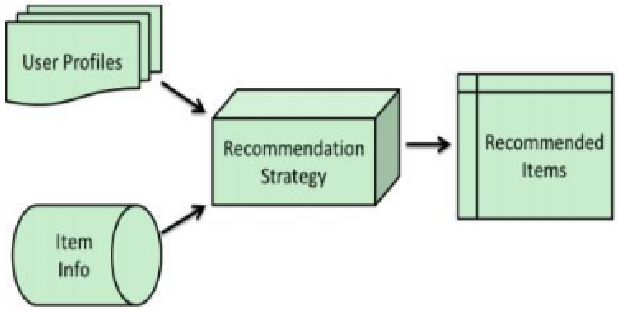


Figure 3.1: An illustration of Music Recommendation System [Kat21]

3.1 Fundamentals of Music Recommendation Systems

This section will provide clear definitions of key terms and concepts that are crucial for comprehending music recommendation systems. The methods encompass collaborative filtering, content-based filtering, and hybrid systems.

3.1.1 Collaborative filtering

This type of recommendation is derived from an analysis of both the conduct of the listeners and the conduct of all other users on the site. The fundamental concept is that the opinions of other users can be used to accurately forecast the preferences of a user for an item that they have not yet rated. In this approach, a user is provided with recommendations based on other users who have similar tastes. For many years, we have relied on suggestions from our friends, family, and coworkers for guidance on music, restaurants, movies, and other forms of entertainment. This mechanism is the one that is being attempted to be replicated here. Netflix first popularized this strategy, which relies on user ratings, and it has since gained widespread use, particularly in the case of Spotify's Discover Weekly. [KG22]

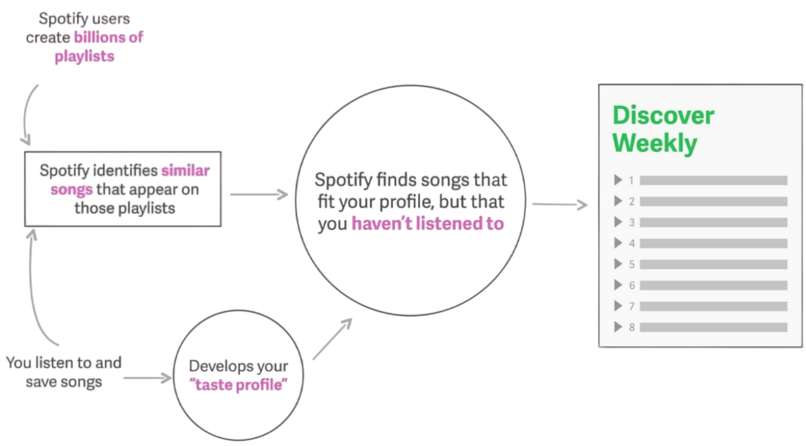


Figure 3.2: Flow chart for Discover Weekly playlist

The system often employs the "K-nearest neighbor (KNN)" method, which relies on the user's past data. The music preferences of users are used to measure the distances between various users. The collaborative filtering recommendation method utilizes the "nearest neighbor user" of the target user to assess and assign weight to the product's worth. Collaborative filtering recommendations forecast the level of the user's affinity for a particular target product. [DL15]

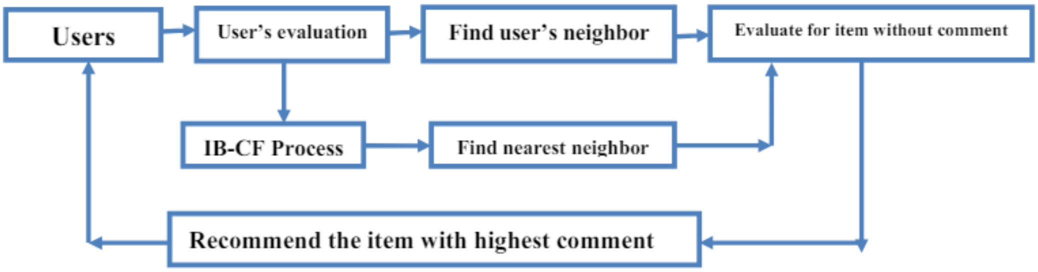


Figure 3.3: Collaborative filtering system flow. [DL15]

The system utilizes usage statistics and user interests to identify "neighbor users" who have similar interests to the user. Subsequently, the recommendation engine suggests information that aligns with the interests of "neighboring users" to the user. The fundamental concept behind collaborative filtering systems is readily comprehensible. Put simply, this technique is widely embraced in everyday life, as individuals often rely on their friends' suggestions when making choices. [DL15]

Advantages	Disadvantages
• Prior understanding of the subject matter is unnecessary	• The quality is contingent upon a substantial history dataset.
• Can accurately identify niches	• Prone to statistical irregularities in data.
• Quality improves over time	• Reacts slowly to drift
• Personalized recommendations.	

Table 3.1: Collaborative filtering systems trade-offs. [DL15]

3.1.2 Content-based filtering

Content-based recommendation is an effective way of providing personalized recommendations by analyzing the attributes of items and user profiles. This approach employs information filtering technology and applies machine learning to predict customer preferences by assessing the characteristics of the objects without relying on the user's opinion. [DL15]

One significant benefit of content-based filtering is that it does not suffer from the new item problem, which is the difficulty of a system recommending new items that have not been rated by any user. This means that both well-known and less-known artists have an equal opportunity to be recommended. Nevertheless, a significant limitation of the content-based filtering strategy is its assumption that each user is autonomous and unrelated to others. Its lack of consideration for the social aspect of human behavior restricts it to excessive specialization, resulting in recommendations that are only similar to items previously listened to by the user. [LTCC14]

In the context of music recommendation systems, content-based filtering utilizes the characteristics of songs, such as genre, tempo, instrumentation, and lyrical themes, to produce recommendations. It utilizes metadata linked to music recordings, including artist data, album specifics, and user listening history, to precisely locate songs that align with a user's interests. Furthermore, improvements in audio feature extraction methods, such as spectrogram analysis and audio fingerprinting, contribute to the increased precision of content-based music suggestions. Although content-based filtering is effective, it may encounter difficulties in effectively capturing the subjective and nuanced components of musical taste preferences in music recommendation systems. To address these problems and improve the recommendation experience for users, it is beneficial to incorporate user input mechanisms and utilize hybrid systems that mix content-based and collaborative filtering techniques.

Advantages	**Disadvantages**
• Excluding other user information simplifies the process of giving early recommendations.	• Extracting content and identifying useful attributes for analysis can be challenging. Content-based recommendation relies on organized content and the capacity to articulate user preferences in a distinctive manner.
• It can provide recommendation service to users with special interests.	• The content-based filtering technique is ill-suited for assessing media content such as music and video, as it relies on machine learning to acquire and discern the user's preferences.
• Suggests both new and "non-mainstream" items.	• The value of the recommendations cannot be evaluated.
• It is relatively simple to implement.	

Table 3.2: Content-based filtering trade-offs. [DL15]

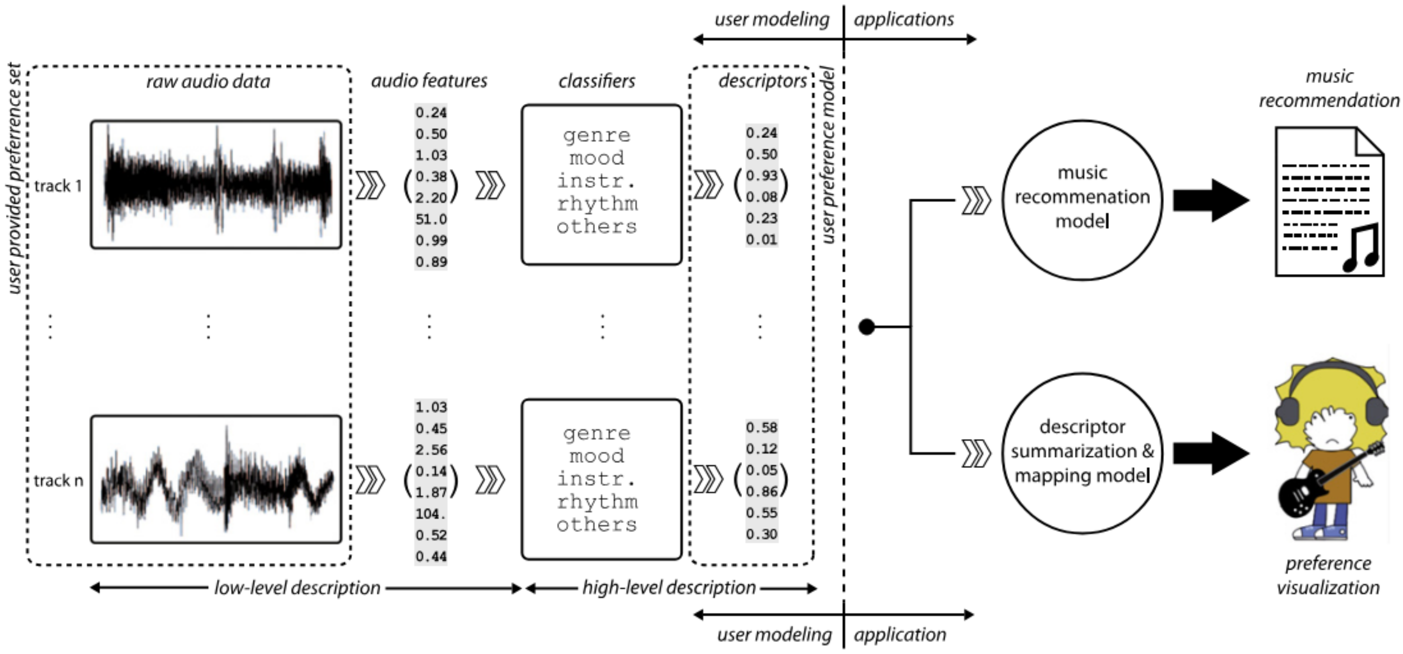


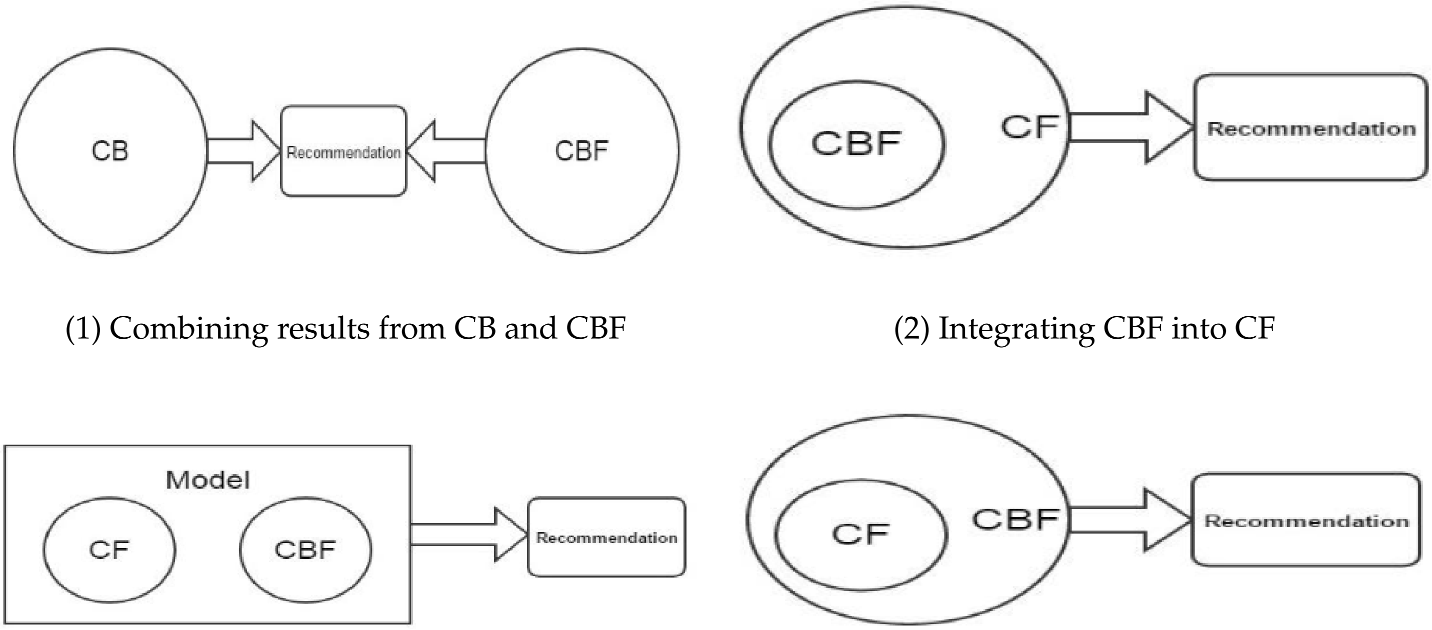
Figure 3.4: Content-based filtering system flow in the context of music. [BHF+13]

3.1.3 Hybrid system

Recent research has shown that hybrid techniques can be quite beneficial in specific situations. Collaborative filtering (CF) and content-based filtering (CBF) are the predominant techniques used in information filtering applications. Nevertheless, every strategy possesses its unique advantages and disadvantages. The main goal of hybrid techniques is to merge collaborative filtering (CF) with content-based filtering (CBF) in order to improve the accuracy of recommendations. Hybrid approaches provide versatility in their implementation, since they can be utilized in various ways according to unique needs and circumstances. [TGB15]

In this regard, several approaches can be employed:

1. Implement both collaborative and content-based approaches separately and combine their forecasts.
2. Incorporate content-based attributes into a collaborative methodology.
3. Create a comprehensive consolidative model that combines the features of both content-based and collaborative approaches.
4. Incorporate collaborative attributes into a content-focused methodology. [TGB15]



(1) Combining results from CB and CBF (2) Integrating CBF into CF

(3) Unified model using CF and CBF (4) Incorporate CF into CBF

Figure 3.5: Hybrid systems approaches flow charts [TGB15]

3.2 Case Study: Spotify's Music Recommendation System

The acquisition of The Echo Nest by Spotify marked an important milestone in its efforts to improve and expand its talents in recommending and discovering music. The Echo Nest was established in 2005 by Tristan Jehan and Brian Whitman as a firm specializing in music intelligence. The company focused on music data analysis, offering a variety of services and APIs to music firms, developers, and researchers.

Spotify provides a multitude of features, such as the ability for users to search for their preferred music, the Browse function which offers a curated selection of popular or mood-based music, the Radio function, and the Discover function. Spotify also showcases itself on social networks, allowing users to explore the collections of their friends, other musicians, or celebrities. The fundamental recommendation function of Spotify comprises the Discover, Related Artists, and Radio lists. Spotify is additionally incorporated with last.fm.

3.2.1 Spotify's recommendation engine

Spotify primarily started improving its recommendation engine by focusing on collaborative filtering. Collaborative filtering works by deducing users' preferences through the analysis of their past usage data. For example, when two users, A and B, exhibit comparable listening patterns, the algorithm deduces that they have similar musical preferences. Moreover, if a particular set of users repeatedly listens to both songs, the system can infer that both songs share comparable characteristics. [DL15]

Although collaborative filtering provides useful insights into user preferences, it also has inherent limits. A major obstacle arises when it comes to suggesting new or less popular music, as collaborative filtering mainly depends on usage statistics, which makes it more convenient to recommend popular tunes that have been widely used in the past. In addition, collaborative filtering fails to include variables such as tone, genre, and lyrics, so restricting its capacity to provide varied and subtle recommendations. [DL15]

Spotify recognized the limitations of collaborative filtering and decided to take a new direction after acquiring The Echo Nest. Spotify aimed to enhance its music discovery and recommendation abilities by incorporating content-based filtering into its recommendation algorithm. Content-based filtering is assessing several attributes of music, including genre, tone, lyrics, and artists, in order to produce individualized suggestions. By utilizing data mining and digital signal processing techniques provided by The Echo Nest, Spotify is now able to utilize content analysis to enhance its recommendation engine. [DL15]

Spotify's recommendation system combines both collaborative and content-based filtering techniques into one unified hybrid system. An example of this is the "Discover" feature, which mostly relies on collaborative filtering. It analyzes users' past usage habits to provide music recommendations. In contrast, the "Radio" feature mostly relies on content-based filtering, suggesting music that share comparable features with a user-selected tune. [DL15]

By employing a hybrid approach, Spotify is able to utilize the advantages of both collaborative and content-based filtering. This allows Spotify to provide users with a wide range of personalized recommendations that are specifically tailored to their individual musical preferences and inclinations. [DL15]

3.2.2 Spotify's recommendation API

The recommendation API offered by Spotify equips developers with robust tools for integrating personalized music recommendations into their applications and services. It utilizes different algorithms and data analysis approaches to empower developers in providing consumers with personalized music recommendations that match their individual tastes and preferences.

Track's Audio Features endpoint

Spotify's API incorporated the Track's Audio Features endpoint after acquiring The Echo Nest, which enhanced Spotify's powers in data mining and digital signal processing. By utilizing The Echo Nest's specialized knowledge, Spotify broadened its API offerings to encompass comprehensive analysis of the acoustic characteristics of individual music. This API endpoint grants developers access to extensive metadata, allowing for the retrieval of crucial musical components such as tempo, key, and mode. By utilizing this metadata, developers can optimize content-based recommendation systems, providing more customized music suggestions that are specifically matched to consumers' individual preferences. Spotify's integration emphasizes its dedication to enhancing the experience of discovering music for its users and showcases the platform's continuous use of innovative technologies to improve its offerings.


```json
{
    "acousticness": 0.2,
    "analysis_url": "https://api.spotify.com/v1/audio-analysis/6WK9dVrRABMkUXFLNlgWFh",
    "danceability": 0.663,
    "duration_ms": 255133,
    "energy": 0.838,
    "id": "6WK9dVrRABMkUXFLNlgWFh",
    "instrumentalness": 0.000177,
    "key": 0,
    "liveness": 0.0782,
    "loudness": -5.744,
    "mode": 0,
    "speechiness": 0.0286,
    "tempo": 112.788,
    "time_signature": 4,
    "track_href": "https://api.spotify.com/v1/tracks/6WK9dVrRABMkUXFLNlgWFh",
    "type": "audio_features",
    "uri": "spotify:track:6WK9dVrRABMkUXFLNlgWFh",
    "valence": 0.752
}
```


Figure 3.6: Track's Audio Features endpoint response for "Enjoy the Silence" by Depeche Mode

Track's Audio Analysis endpoint

In addition to the Track's Audio Features endpoint, Spotify's API also offers the Track's Audio Analysis endpoint. The purpose of this endpoint is to provide developers with access to in-depth audio analysis data for music in the Spotify catalog. This data includes comprehensive information about rhythm, pitch, and timbre. The output of this endpoint includes a thorough examination of the musical composition of the piece, offering useful observations about its arrangement and substance. Developers can employ this data to construct sophisticated music analysis tools, such as visualization programs or music transcription software. In addition, the Track's Audio Analysis endpoint allows developers to improve content-based recommendation systems by integrating comprehensive audio analysis into their algorithms, resulting in more precise and personalized music suggestions that cater to customers' preferences.

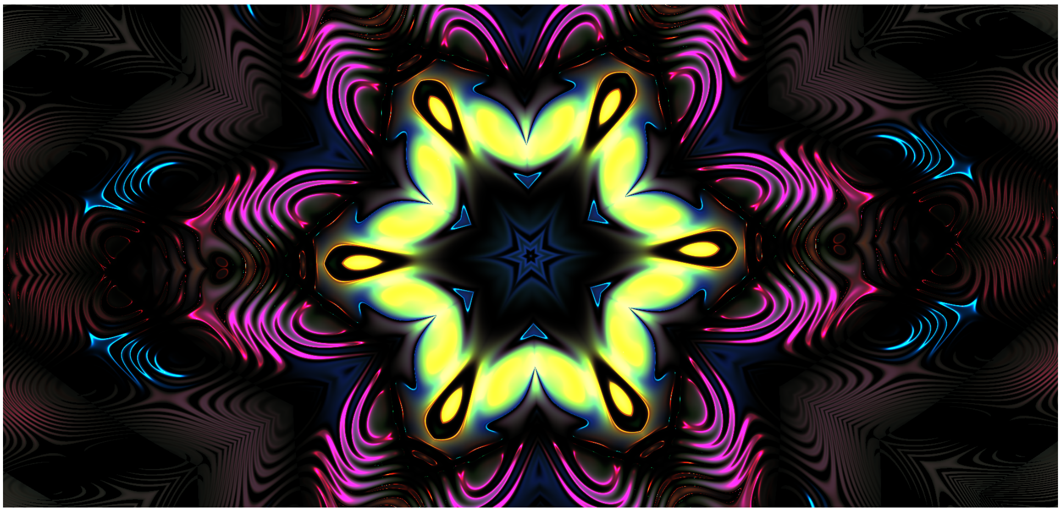


Figure 3.7: A visualization of the song "Dreams" by Fleetwood Mac using Track's Audio Analysis endpoint response

Get Recommendations

The "Get Recommendations" endpoint in Spotify's API is a potent tool specifically created to provide personalized music recommendations for consumers. This endpoint utilizes a combination of collaborative filtering and content-based filtering to recommend tracks that are in line with a user's distinct musical preferences and listening patterns.

The recommendations are improved by incorporating the user's listening history and preferences. This enhances the precision and pertinence of the recommendations, providing a customized music exploration experience.

The endpoint offers up-to-date recommendations that are continuously refreshed using the most recent data. This dynamic methodology guarantees that customers are provided with up-to-date and pertinent music recommendations, so ensuring their listening experience remains captivating.

Developers can define a range of criteria to customize recommendations. The criteria encompass a maximum of 5 seeds in total, which can be a mixture of seed artists, seed genres, and seed tracks. By supplying these seeds, the endpoint can generate a suggestion list that closely aligns with the user's preferences. [Sur22]

In addition to the seeds, the recommendations can be fine-tuned using parameters from the Track's Audio Features endpoint:

1. Popularity - This value can vary from 0 to 100, with 100 representing the maximum frequency or occurrence. The popularity of an artist is contingent upon the collective popularity of all their tunes.

2. Acousticness - This value quantifies the probability that a track is acoustic, ranging from 0.0 to 1.0. A rating of 1.0 indicates a high level of certainty that the track is acoustic.

3. Danceability - Ranging from ranging from 0.0 to 1.0, this parameter is evaluated using a combination of musical elements including pace, rhythm consistency, bar strength, and overall uniformity. A grade of 0.0 signifies the lowest level of danceability, while a rating of 1.0 signifies the highest level of danceability.

4. Duration - The duration of the song in milliseconds.

5. Energy - This parameter utilizes a numerical scale ranging from 0.0 to 1.0 in order to quantify the perceptual assessment of a track's intensity and activity level. The feature is influenced by factors like as dynamic range, perceived loudness, timbre, speed, and total entropy.

6. Instrumentality - This parameter is used to determine if a track has vocal elements. A higher instrumental score, closer to 1.0, indicates a greater likelihood that the track is solely instrumental. Values exceeding 0.5 signify the presence of instrumental content, with higher values suggesting a higher level of certainty in the absence of vocal material.

7. Key - The song's tone. A number of -1 denotes the absence of tonality, but values ranging from 0 to 11 signify the tonality using Standard Pitch Class notation.

8. Liveness - This parameter is used to detect the presence of audiences in the recording. Tracks with higher liveness scores suggest a higher probability that the tune was performed live. A number exceeding 0.8 indicates a substantial probability that the track was executed in a live performance. This parameter has a range of values from 0 to 1, which is represented as a floating-point number.

9. Loudness - The track's total volume, quantified in decibels (dB), spans from -60 dB to 0 dB. The sound values are calculated by taking the average of the sound levels throughout the track. These values can then be used to compare the relative sound levels of different songs.

10. Mode - The mode of a track indicates its modality, either major or minor, or the specific sort of scale used for its melodic material. A value of 1 signifies the highest level of modality, whilst a value of 0 signifies the lowest level.

11. Speechiness -The higher the value of the attribute, the more predominantly spoken the recording is. Varies between 0.0 and 1.0.

12. Tempo - The worldwide tempo of a track, expressed in beats per minute (BPM). It is directly generated from the mean duration of time periods.

13. Time signature - An estimated time signature denotes the frequency at which each bar is reiterated. The time signature ranges from 3 to 7, encompassing time signatures that range from "3/4" to "7/4."

14. Valence - The musical positiveness conveyed by a track is quantified on a scale from 0.0 to 1.0. Tracks with high valence exhibit a more positive emotional tone, such as happiness, cheerfulness, and euphoria. Conversely, tracks with low valence convey a more negative emotional tone, such as sadness, depression, and anger.

The heatmap below indicates that the strongest association is found between loudness and energy, with a correlation coefficient of 0.82. Furthermore, there are significant associations between popularity and attributes like as loudness (0.36), danceability (0.26), and energy (0.25). Valence exhibits significant positive associations with danceability (0.55), energy (0.44), and loudness (0.40). In contrast, acousticness has the lowest correlation. The findings indicate that popular songs on Spotify are typically characterized by high danceability, loudness, and intensity, potentially suggesting genres such as hip-hop, electronica, or dance music. Nevertheless, it is important to recognize that these associations may not comprehensively reflect the popularity of alternative music genres like classical, rock, or blues, which still have substantial admiration from audiences. [Sur22]

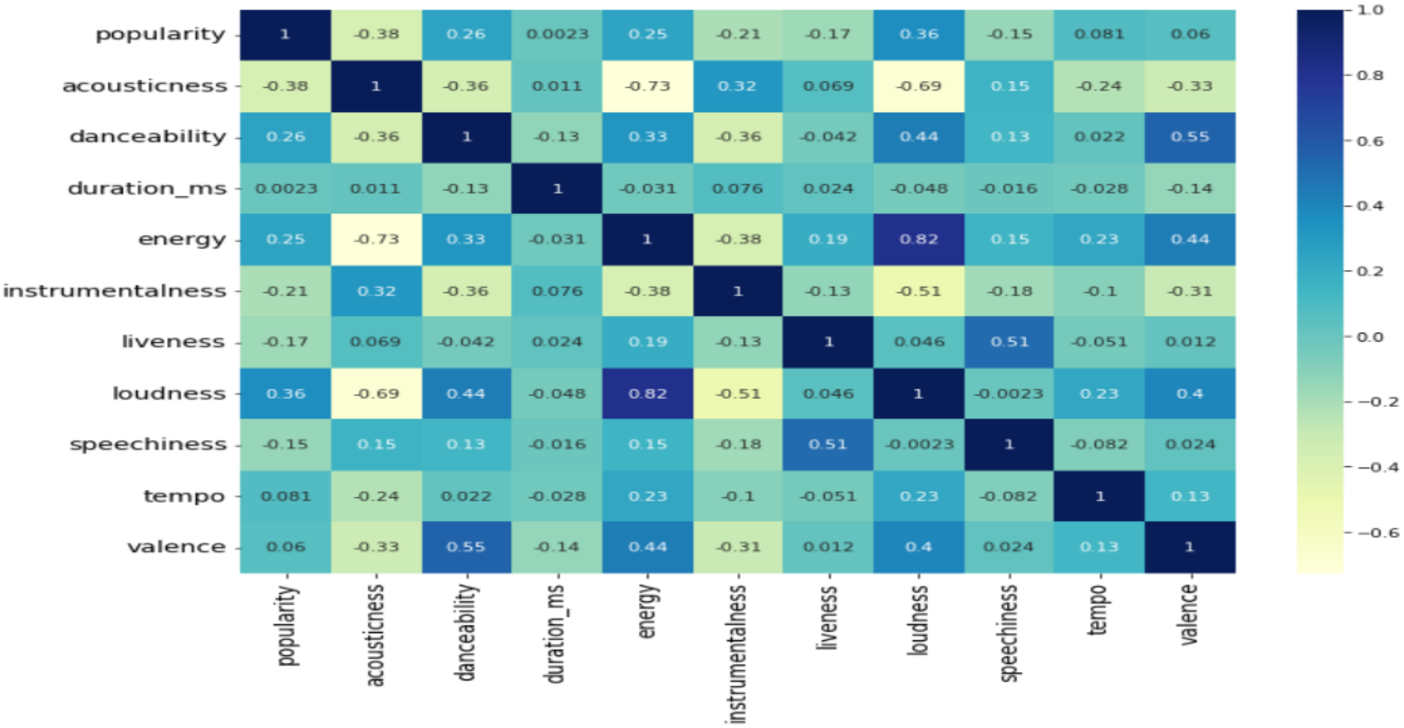


Figure 3.8: A heatmap of popular songs audio features. [Sur22]


```
{
  "genres": ["acoustic", "afrobeat", "alt-rock", "alternative", "ambient", "anime",
"black-metal", "bluegrass", "blues", "bossanova", "brazil", "breakbeat", "british",
"cantopop", "chicago-house", "children", "chill", "classical", "club", "comedy", "country",
"dance", "dancehall", "death-metal", "deep-house", "detroit-techno", "disco", "disney",
"drum-and-bass", "dub", "dubstep", "edm", "electro", "electronic", "emo", "folk", "forro",
"french", "funk", "garage", "german", "gospel", "goth", "grindcore", "groove", "grunge",
"guitar", "happy", "hard-rock", "hardcore", "hardstyle", "heavy-metal", "hip-hop",
"holidays", "honky-tonk", "house", "idm", "indian", "indie", "indie-pop", "industrial",
"iranian", "j-dance", "j-idol", "j-pop", "j-rock", "jazz", "k-pop", "kids", "latin",
"latino", "malay", "mandopop", "metal", "metal-misc", "metalcore", "minimal-techno",
"movies", "mpb", "new-age", "new-release", "opera", "pagode", "party", "philippines-opm",
"piano", "pop", "pop-film", "post-dubstep", "power-pop", "progressive-house", "psych-rock",
"punk", "punk-rock", "r-n-b", "rainy-day", "reggae", "reggaeton", "road-trip", "rock", "rock-
n-roll", "rockabilly", "romance", "sad", "salsa", "samba", "sertanejo", "show-tunes",
"singer-songwriter", "ska", "sleep", "songwriter", "soul", "soundtracks", "spanish", "study",
"summer", "swedish", "synth-pop", "tango", "techno", "trance", "trip-hop", "turkish", "work-
out", "world-music"]
  }
```


Figure 3.9: Retrieving available genre seeds using Spotify API.

Chapter 4

Engineering "Beat With It"

In order to develop Beat With It successfully, it is crucial to have a comprehensive grasp of its requirements and desired functionality. What goals should we strive for, and what is the underlying significance of our work? Beat With It strives to provide a user-friendly solution that enables the effortless connection between TikTok music clips and their associated Spotify songs.

However, doing this on a single device has difficulties, since it requires the simultaneous operation of both Shazam and music playback, which I have found to be a demanding process. Expanding the capacity to process 100 clips makes it impractical to manually identify songs using Shazam.

Additionally, I like the possibility of engaging in participatory music exploration. I am fascinated by the process of identifying the highest-rated songs. I am also interested in discovering lesser-known songs that are similar in quality to these popular favorites. Categorizing TikTok sounds instead of having them in a single repository is preferable, since it improves accessibility and facilitates maintenance. Uncovering concealed revelations inside TikTok videos enhances the depth of research. Assessing the quality and widespread acceptance of TikTok songs amplifies the platform's allure.

The many issues that Beat With It aims to tackle are emphasized by these concerns. As we begin this project, it becomes evident that these problems have great importance. They are not only hurdles to overcome, but rather chances to create new ideas and transform the way we find and appreciate music.

The engineering of Beat With It followed the Software Development Life Cycle (SDLC) paradigm, specifically using the Waterfall model. This methodology guaranteed a step-by-step progression, where each stage relied on the successful completion of the preceding one.

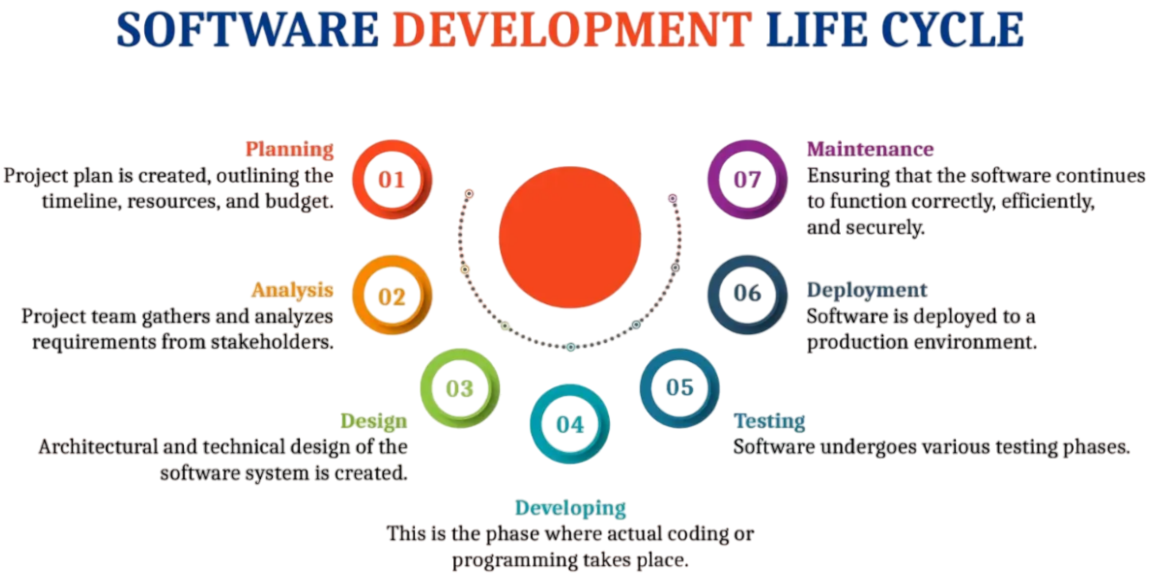


Figure 4.1: Software Development Life Cycle (SDLC) diagram. [Dis24]

4.1 Beat With It requirements

To ensure optimal functionality of the application, it is essential to establish clear and succinct requirements as a first step. When documenting product requirements, it is crucial to differentiate between functional and non-functional needs.

In essence, functional requirements delineate the specific actions and functionalities that the product must possess, but non-functional requirements impose limitations on the manner in which the product should execute those actions and functionalities.

They may be represented in the following format:

- Functional requirement: "The application must do [requirement]."
- Non-functional requirement: "The application shall be [requirement]."

4.1.1 Functional requirements

We need to establish explicit user activities and what he can do in our application:

- Users should be provided with the functionality to create accounts, which will provide them access to the platform. Additionally, they should be able to login in to their accounts and recover their password in case it is forgotten.
- Upon entering, the user should be presented with the homepage and a list of the most recent assets in the system. The user should possess the capability to engage with them, perceive their details, actively listen to them, include them in their watchlists, and conceal them if desired.
- The user should be able to link their Spotify account to the platform. Subsequently, he will have the capability to use the tailored recommendations feature and add songs to his "Saved Songs" playlist on Spotify. The user should possess the capacity to disassociate their account at any given point.
- The user should have the ability to see, edit, and update their profile information, as well as change their password. He should have the capability to log out at any point. In addition, he should possess the capability to see the concealed assets and then reveal them to their original listings.
- The user must have the capability to see, create, modify, and remove the following entities: Search Jobs, Alerts, Digital Scout, and Watchlists. These resources should be restricted to authorized clients who have logged in. Only an administrator should possess the capacity to access information on the assets that were uploaded to the system by the Digital Scout of a user.
- The customer may access Spotify Songs without the need to connect his Spotify account to Beat With It. In addition, he has the ability to access the Charts from TikTok, Shazam, and Spotify, but he is unable to do any actions on them. In addition, he has the ability to see the Spotify Playlists that have been added to the system, add other playlists to be monitored, and input different Spotify User Handles for the system to track. However, it is important to note that he does not have the authority to remove these playlists, since this privilege is reserved just for administrators.
- The administrators should possess the capability to perform all client actions, as well as have the authority to globally conceal assets, modify application settings, and manage users by banning, updating their information, changing their passwords, enabling or disabling 2FA, banning them or deleting them.

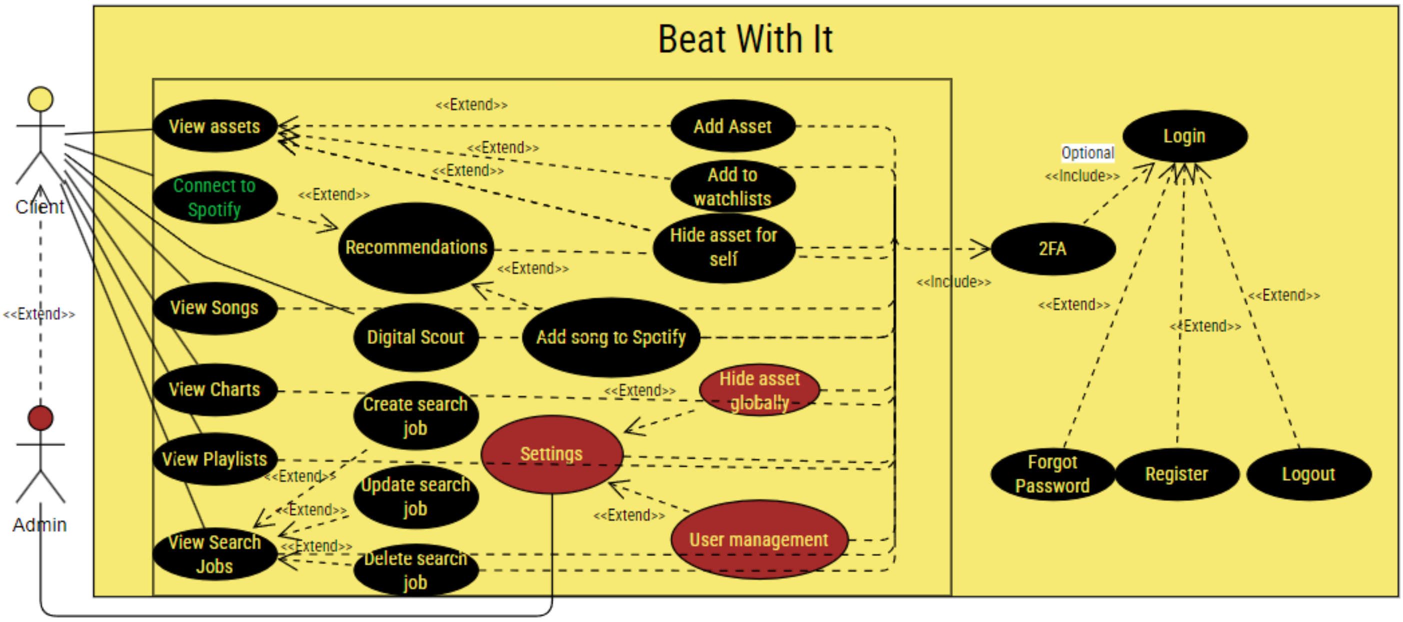


Figure 4.2: Use Case Diagram for Beat With It

4.1.2 Non-functional requirements

We should explore some non-functional needs that should be implemented for every online application:

- Performance: The platform should exhibit fast loading times and promptly react to user activities without any noticeable delay. Any subsequent page loads or navigations should take no more than 1 second. Search queries must provide results within a maximum of 500 milliseconds.

- Security: User data must be kept and sent in a secure manner, using safeguards to prevent unauthorized access and defend against attacks such as SQL injection and cross-site scripting. Confidential data such as passwords must go through a process called hashing. It is advisable to include additional security measures such as two-factor authentication (2FA).

- Usability: The platform should have an intuitive user experience, defined by clear navigation and user-friendly features. The platform should provide users an interface that amplifies their pleasure in utilizing it. It should also be adaptable to any device.

All of these non-functional needs should be applicable to Beat With It. In addition to them, we must use additional customized non-functional requirements:

- The platform must ensure the reliability of user activities by preventing clients from performing administrative tasks or bypassing authorization protections such as two-factor authentication (2FA).

- The system must guarantee that the data it provides to the user is relevant, engaging, and accurate. It should only show data that is accessible to us via the platforms we are utilizing as third-parties. This means that if TikTok prohibits a certain song, the platform should refrain from displaying it.
- The system should provide an alert service that promptly notifies the user of the outcome of their activities, whether they were successful or resulted in an error. If there is any data retrieval process occurring, the user should be notified that the system is actively retrieving data and a loading screen should be shown.
- Only logged-in users should have access to see all activities on the platform and all accessible information.
- The application must have a mechanism for enforcing a restriction on the number of requests (rate-limiter).

4.1.3 Constraints

Given that Beat With It relies on third-party platform to get data, it is essential to define the boundaries and limits that the system faces.

TikTok

TikTok has an official API, however, acquiring an authorization key for this thesis application proved to be difficult due to the many prerequisites that must be met throughout the application procedure. Beat With It utilizes many techniques to extract data from TikTok. However, it depends on the specific way TikTok has been implemented. Therefore, if TikTok were to make any changes to these particulars, it might possibly result in Beat With It not functioning properly, therefore requiring prompt adjustments.

Shazam

Following Apple Inc.'s acquisition of Shazam, Shazam now offers an official application programming interface (API) called ShazamKit. However, this API is only accessible for iOS devices. Beat With It utilizes a Linux-exclusive open-source reverse-engineering library called SongRec.

Spotify

Thankfully, Spotify is quite accommodating to developers and offers an official API for seamless integration. This API is used by the system for a multitude of purposes.

However, several features of Beat With It, such as collecting music streams, depend on parsing data from Spotify public websites since the official Web API does not provide an endpoint for this particular data.

Rate limiting

The recommendations feature provided by Beat With It relies on the use of the Spotify API ”Get Recommendations” endpoint. Spotify does not explicitly disclose the exact number of requests it handles every second. Instead, it employs an internal formula based on a rolling 30-second window to rate its performance. This approach makes it difficult to estimate the precise number of requests.

4.2 System Architecture and detailed design

The system architecture of ”Beat With It” offers a complete framework that delineates the structure and interaction of different components inside the application. This section provides a comprehensive explanation of the high-level architecture, which encompasses the client-server paradigm. In this model, the client is responsible for managing user interactions, while the server oversees business logic and data processing.

Additionally, it encompasses the incorporation of TikTok, Shazam, Spotify, as well as the development of a database architecture that optimizes data storage and retrieval. The design guarantees the capacity to manage a large amount of traffic, achieve high speed, and provide strong security measures.

4.2.1 Frontend - User Interface (UI) and User Experience (UX)

Developing a design for Beat With It proved to be a challenging task, but with some careful deliberation, I managed to conceive a design for the User Interface. I comprehended the specifications set by myself, necessitating a user interface that is user-friendly, engaging, responsive, and instinctive. I want to provide the facts in an optimal manner without causing the user to feel overwhelmed.

The software was designed with a focus on User Experience, ensuring high responsiveness. The user receives toast feedback, quick page loads, intuitive iconography, tooltips, and modals for confirmations in response to every action they do. The app has a cohesive and consistent design across its whole. The home screen assets list table is a versatile component that is used in several sections of the program, enhancing its usability.

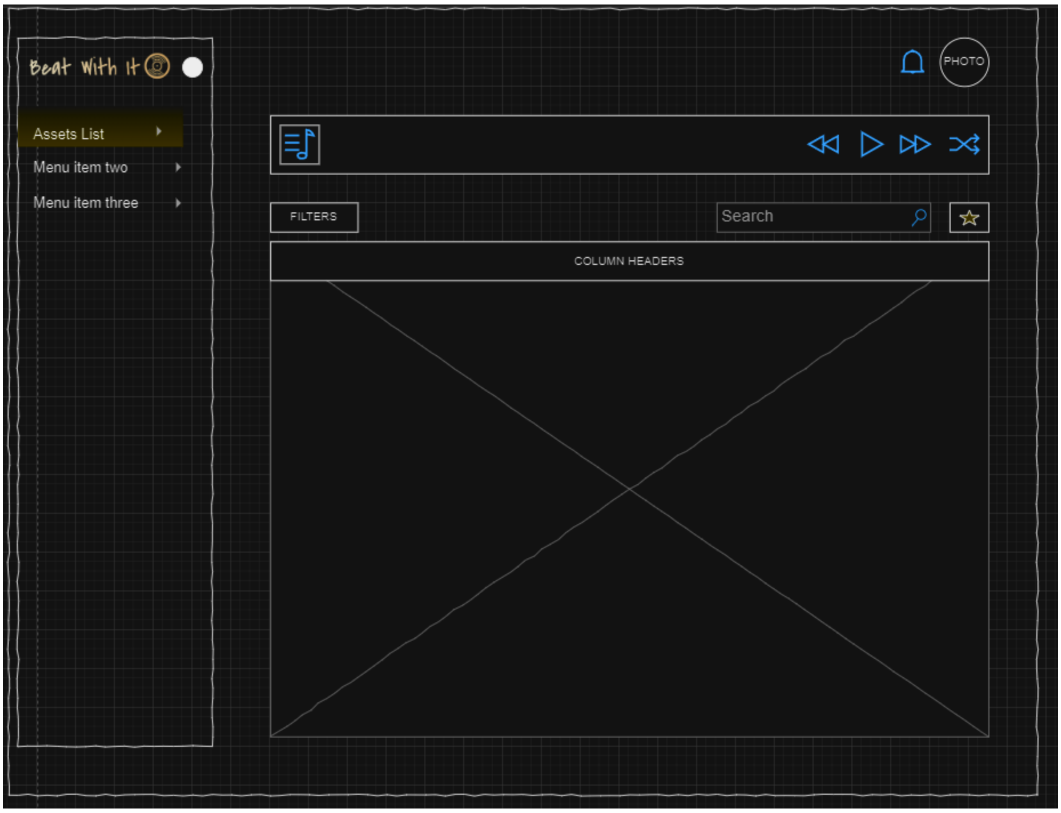


Figure 4.3: Home screen wireframe of Beat With It

4.2.2 Backend - Server and Business Logic

Once we have clearly established and comprehended the requirements, developing the business logic for Beat With It becomes a straightforward task. Currently, in Beat With It, there are three methods by which assets from TikTok may be integrated into the system:

- Add Asset - the user gives the link to a TikTok music sound and the system proceeds to scan the webpage to extract relevant information.
- Digital Scout - the user provides his TikTok account to the system and the system parses his ForYou feed, adding every music sound it finds.
- TikTok Charts - the system crawls the official charts from TikTok in order to find the best performing songs, then adds them to the platform.

Nevertheless, the data collected via TikTok is insufficient. We want a suitable music that corresponds to the Original Sounds from TikTok. Here we present Shazam. Each asset that is added to the system is placed in a Shazam Queue. In this queue, the song is downloaded and the audio is fingerprinted, with the assistance of an external open-source, reverse-engineered, Linux library, called SongRec. The library essentially replicates the fingerprinting process used by Shazam and provides the associated information as output.

Subsequently, a request is sent to the Shazam server with the obtained metada, essentially simulating a person using the Shazam app to identify a sound with their phone. Shazam provides a response including details about the identified music.

There is also the possibility of receiving an empty answer, indicating that Shazam was unable to discover a matching song for the given asset. This suggests that the asset might potentially be an unreleased musical composition or a recording of someone speaking, since these types of TikTok videos are very popular and have the potential to become viral. When an asset is successfully matched, it provides an ISRC (International Standard Recording Code), which serves as a unique identifier for a published song, same to how an ISBN is used for books.

Following that, we may use the ISRC (International Standard Recording Code) to conduct a search on Spotify based on the ISRC, or alternatively, we can search for it in the database, since there is a possibility that we may own the song by accessing it via Spotify Charts. The probability of finding zero results is rather small, considering that Spotify provides an collection of 100 million songs. Therefore, we have obtained all the essential information.

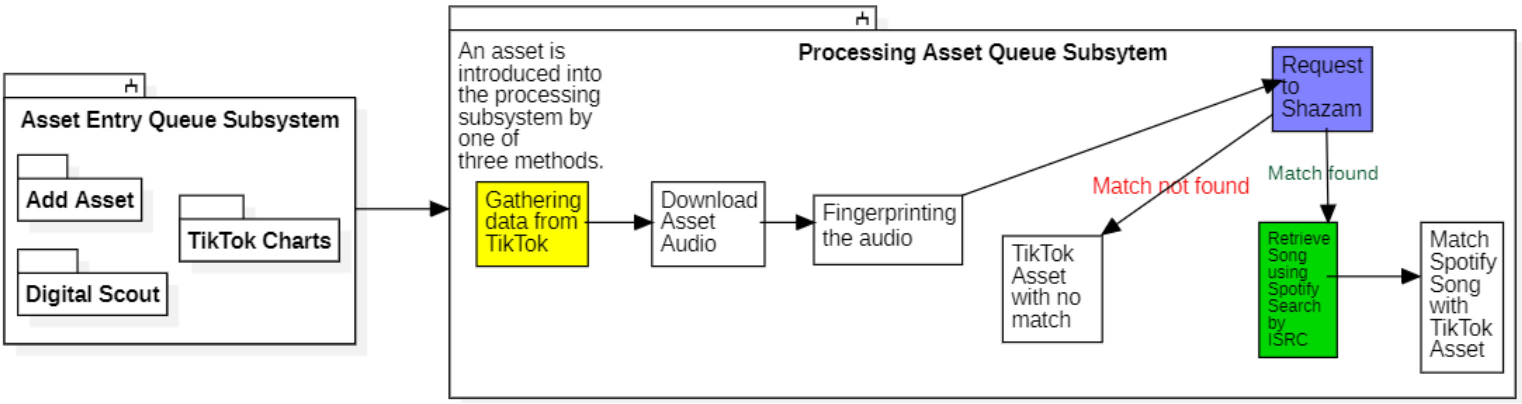


Figure 4.4: Asset Processing Workflow Diagram

To monitor the performance of each asset and track, we employ a Daily Monitoring Queue. This queue follows a similar process as described, but it excludes re-shazaming matched assets and does not attempt to find a Spotify matched song.

Instead, it focuses on tracking the performance in terms of streams for the already matched songs. Unmatched assets continue to be process by the Shazam Queue because an asset has the potential to transition from being unreleased to released. This is because some artists often share their music on TikTok before officially releasing it. If we eventually find a match, we may begin monitoring the performance on Spotify as well.

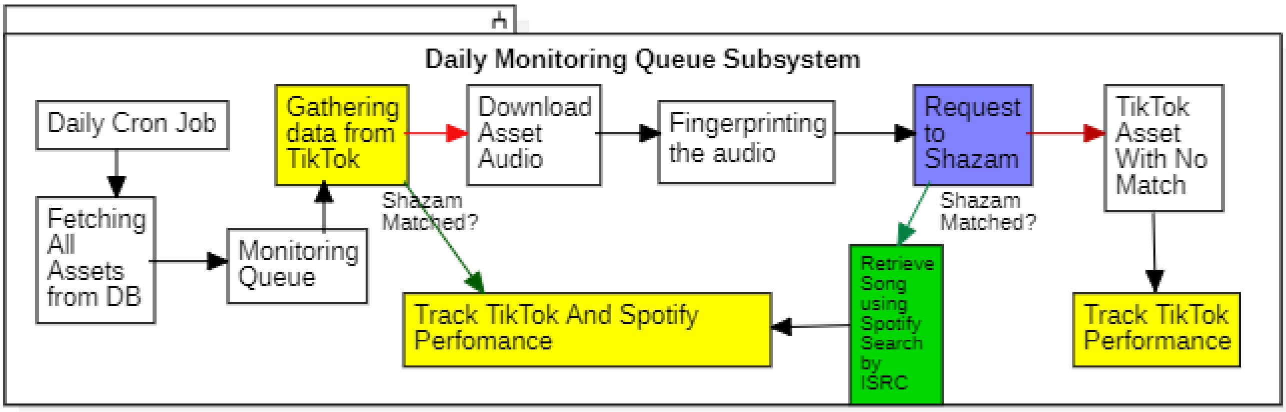


Figure 4.5: Daily Monitoring Queue Subsystem Workflow Diagram

4.2.3 Authentication and Authorization

For session management, authentication is made easier with JSON Web Tokens (JWT), which offer a safe way to confirm users' identities. Furthermore, before being kept in the database, passwords are safely hashed using bcrypt, a popular cryptographic hashing technique. By utilizing bcrypt for password hashing and JWT for session management, strong security measures are implemented to safeguard user credentials and reduce the possibility of unwanted access and data breaches.

I have added two-factor authentication (2FA) with email verification in addition to password security. By asking customers to confirm their identification with a one-time code delivered to their registered email address, this provides an additional degree of protection. In the event that a password is compromised, the 2FA procedure helps prevent unwanted access and improves account security.

4.2.4 Database Schema

The database schema for "Beat With It" is optimized to effectively handle and structure the application data, guaranteeing scalability and ease of maintenance. Due to the need for several joins, a relational database is the best option for this application.

This schema provides a clear representation of the organization and connections between important elements of our data, ensuring that data can be retrieved quickly and accurately while maintaining its integrity. It was designed to be scalable, allowing it to effectively manage increasing volumes of data.

Primary and foreign keys are indexed, which improves the efficiency of query execution and ensures consistent performance, especially when dealing with huge data sets. Due to the extensive querying performed on several columns, it is necessary to establish a significant number of indexes.

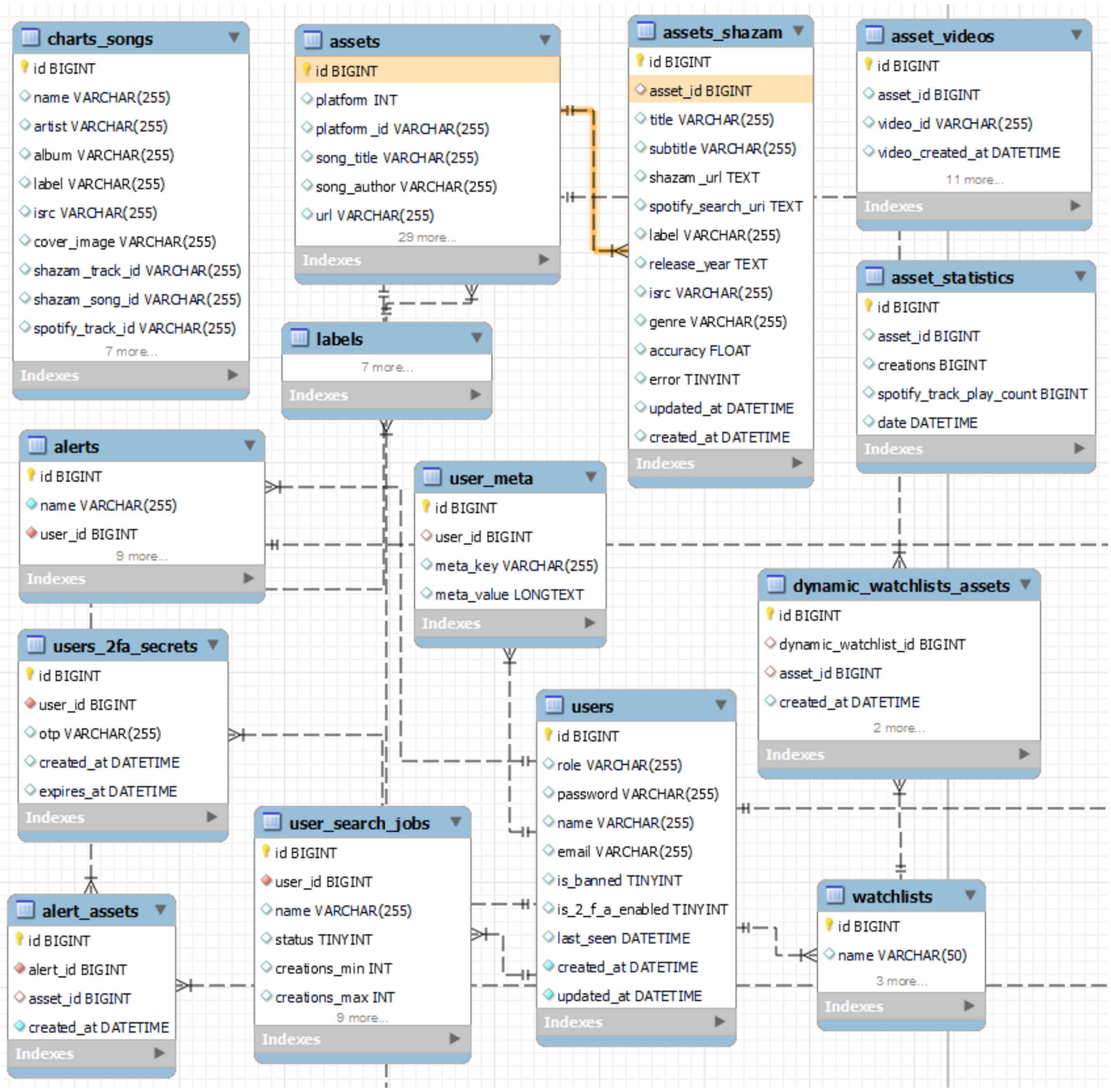


Figure 4.6: Some of the most import tables in the database schema of Beat With It

4.2.5 Data Privacy and Security

Role-based access control (RBAC) has been used in authorization to specify and impose access permissions according to user roles and privileges. Administrators can safely manage users, content, and platform settings thanks to a separate Admin Panel. I put user security and data protection first by integrating bcrypt and JWT for authentication with strong authorization procedures, building user base trust and confidence.

In addition, the platform offers a comprehensive feature set to facilitate user account maintenance, which includes the ability to reset forgotten passwords. By using a validated email link, users may safely reset their passwords, lowering the possibility of nefarious actors resetting passwords without authorization. In addition, email verification is added to the registration procedure, meaning that users must validate their email addresses in order to use the site. This maintains the integrity of user data and aids in the prevention of fraudulent account creation.

4.3 Technologies used

The development of Beat With is supported by a diverse array of modern technologies, meticulously chosen to enhance scalability, security, and performance. Node.js, TypeScript, and Express form the fundamental stack, providing a robust foundation for developing web applications that are both feature-rich and responsive. The project is enhanced by using Next.js robust routing system and efficient development process. When used in conjunction with Sequelize and MySQL, Beat With It ensures reliable data storage and easy database management. Redis and BullMQ provide robust task queuing and caching capabilities that improve user experience and maximize performance, making them a valuable addition to these technologies.

Collectively, this comprehensive stack acts as the basis for Beat With It, enabling the development of a cutting-edge music discovery platform that has the capacity to revolutionize the music business entirely. The technologies have been selected based on the guidance and endorsement of research articles.

4.3.1 Frontend: Next.js, Materio, Material UI, Axios, Chart.js

Within contemporary front-end web designs, Next.js appears as a key framework that provides developers with a powerful and effective toolkit for creating dynamic and scalable web applications [LA22].

Next.js allows developers to take advantage of server-side rendering (SSR) and static site generation (SSG) features, improving user experience and speed, by integrating smoothly with React.js. Developers can concentrate on creating immersive user experiences without sacrificing efficiency or scalability thanks to the framework's simple routing mechanism and integrated API routes.

At the very beginning of the project, I began the implementation using Next.js and JavaScript. However, during more investigation, I found that including TypeScript significantly enhances the whole experience.

The ability to enable smooth data fetching is one of Next.js's key capabilities. This allows developers to pre-render pages with dynamic information while maintaining optimal performance and search engine optimization. Next.js provides unmatched versatility in handling the rendering and data retrieval operations, supporting both client-side and server-side data fetching.

In addition, Next.js supports a thriving ecosystem of libraries and plugins, which promotes extensibility and lets developers alter and expand the framework's features to meet the demands of particular projects. Because of this, Next.js stands out as a flexible and strong framework for creating contemporary front-end web architectures, enabling developers to easily create online apps that are high-performing and search engine friendly.

As the application need both a client and admin perspective, I have also included Materio, a Next.js admin template built using Material UI. Materio offers a straightforward method for implementing a frontend architecture that accommodates both client and admin themes. The software has several features that greatly facilitated the development of Beat With It. These include a vertical menu and essential views such as login, registration, and lost password.

Moreover, it provided me with the ability to simply personalize the application and enhance its responsiveness. The Material UI, specifically the DataGridPro component, greatly facilitated the process of showing data in tables inside this application. This enhanced the development experience and made it a more enjoyable and stress-free activity. It offers a high level of customization and enables you to implement any idea you have with little coding.

When it came to retrieving data, I reflected between using the useSWR hook from Next.js and the Axios package. In the application, I used both options, but with a focus on Axioss, as I found it to be more user-friendly and comprehensible. In addition, I believe it offers more configurability and enables the writing of more concise code. Due to its high level of responsiveness to user inputs, Axios facilitated the handling of mistakes by providing more customization options. Besides, I used it on the backend, which streamlined the process significantly due to my existing familiarity with it.

In addition to them, I used Chart.js, an open-source JavaScript library employed for the purpose of visualizing data. The tool allowed me to effortlessly generate a pie chart that displays the distribution of videos by country and it enabled the presentation of bar and line graphs to represent TikTok and Spotify information.

4.3.2 Backend - Node.js, TypeScript, Express

Strong user management systems are essential for contemporary web applications based on Node.js, as highlighted in the article "Web Security Challenges in Node.js Applications" by Kaul et al. [KBB+18]. Considerable thought is necessary to avoid security vulnerabilities regarding the complexities of user registration processes. The authors clarify the need of putting in place safe registration procedures that thoroughly verify user inputs to prevent hostile exploits such cross-site scripting (XSS) assaults and SQL injection, which are common in online contexts.

Along with to its security features, I have selected Node.js for its straightforward installation process, ability to run on several platforms, and extensive customization options. An important benefit is the capability to use the same programming language for both frontend and backend development, resulting in a more efficient development process and easier code maintenance. This integrated approach is beneficial in facilitating smooth interactions across components of the application.

By using Express, a Node.js framework, developers can easily build powerful and scalable RESTful APIs. Express offers a plethora of capabilities and middleware that streamline typical activities, such as routing, processing requests, and managing errors. The versatility of this technology enables developers to organize their applications based on their unique needs, making it an excellent option for constructing web apps and APIs. Node.js and Express provide a potent combo for creating contemporary and effective server-side applications.

Node.js includes npm (Node Package Manager), a powerful tool for managing dependencies and packages for Node.js applications. It streamlines the procedure of installing, updating, and uninstalling packages from your project, enabling developers to smoothly include third-party libraries, frameworks, and tools into their applications.

4.3.3 Database: MySQL, Sequelize ORM

Pereira explores the nuances of using SQL databases in Node.js apps, clarifying the guiding ideas and optimal procedures for a smooth transition. The chapter emphasizes how crucial it is to use SQL databases to manage and store data persistently in order to meet the scalability and dependability requirements of contemporary web applications. [Per16]

Pereira describes several methods for connecting from Node.js to SQL databases, emphasizing the usefulness of ORMs like Sequelize for database communication. The author enables users to fully utilize SQL databases in their Node.js projects by providing comprehensive code examples and step-by-step instructions, thereby promoting strong data management capabilities. [Per16]

The decision to use MySQL as the relational database for the project was made due to a number of straightforward but persuasive factors. My knowledge of MySQL greatly influenced the decision-making process. Utilizing a database system that I am already skilled with enables more streamlined development, debugging, and maintenance procedures, eventually resulting in time and effort savings.

On top of that, the fact that MySQL is open-source was a crucial element. MySQL is an open-source DBMS that provides a cost-effective solution while maintaining high performance and a wide range of capabilities. The dynamic community around MySQL consistently contributes to its enhancement and offers comprehensive documentation, tutorials, and support tools.

Another important factor was the scalability of MySQL. MySQL's scalability guarantees that the database system can effectively manage larger volumes of data and user traffic as the project expands and develops, without compromising speed or reliability.

I opted on Sequelize as the ORM (Object-Relational Mapping) because of its capacity to provide complete control over the database and querying procedures, eliminating the need to directly engage with SQL. Sequelize enhanced the development process by simplifying the intricacies of SQL, resulting in streamlined and more effective operations.

Sequelize has the ability to generate database tables using predefined models, which is a significant benefit. This efficient method of managing database schema reduced the need of manually composing SQL queries for table construction, hence decreasing the possibility of mistakes and making the development process more straightforward.

Moreover, Sequelize greatly simplified the process of creating connections and implementing logical links between several models. I was able to establish connections between models and execute intricate queries with no exertion using the user-friendly API, hence improving productivity and the manageability of code.

4.3.4 Queue System: BullMQ, Redis

Message queues are systems that facilitate asynchronous communication between various application components, granting them the autonomy to generate and receive messages. This decoupling permits components to function independently of one another, which improves flexibility, scalability, and dependability. Asynchronous processing, load balancing, improved dependability via automated task retries, and simpler component scaling are some of the advantages. Asynchronous notifications, managing workload spikes, and background task processing are examples of common use cases. [Kum24]

Utilizing Redis as its message broker, BullMQ is a powerful and efficient message queue module for Node.js. Along with advanced features like work delays, retries, and rate limitation, it provides comprehensive job lifecycle management and permanent task storage. A dashboard for monitoring is included into BullMQ, and it allows distributed queues across several workers. Its design makes it easier to handle background jobs, plan out recurrent chores, and effectively handle unsuccessful job retries.

The choice to use BullMQ as the message queue for the system was made easily, mainly since there are few open-source libraries especially designed for this purpose in Node.js. BullMQ has emerged as a viable alternative in the market, delivering a feature-rich solution specifically designed for Node.js settings, despite the dominance of large competitors like Amazon SQS.

Nevertheless, a minor limitation arose during the development process: BullMQ is only compatible with Redis, which is mostly accessible on Linux platforms. In order to surmount this challenge, I choose to install Redis on my Windows Subsystem, which allows for smooth interaction with BullMQ inside the Windows operating system.

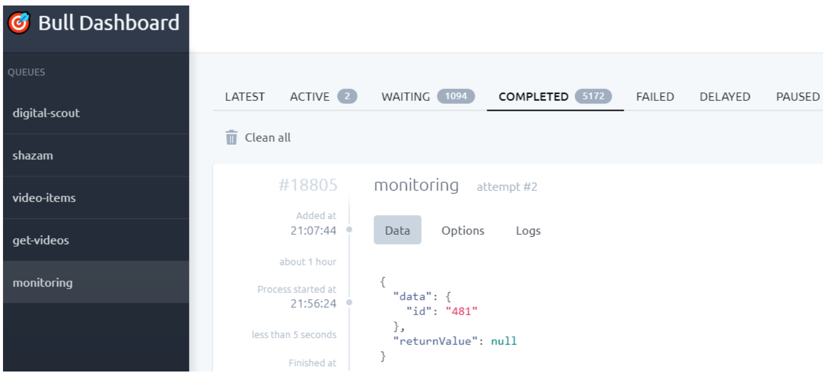


Figure 4.7: Bull MQ Dashboard for Beat With It.

4.3.5 Docker

As stated above in the constraints, the system depends on a Linux-compatible library named SongRec for audio fingerprinting. In order to include this library into the project, I choose to develop a Docker image that is pre-configured with a Linux operating system.

By packaging the SongRec library and its dependencies in a Docker container, I insured that it would run consistently and reliably, independent of the host machine it is deployed on. This technique allowed for the smooth incorporation of the audio fingerprinting capabilities into the project, without requiring significant alterations to the existing infrastructure.

4.3.6 Other Technologies

The selection of NodeMailer as the email delivery service for the project was based on its strong and user-friendly capabilities. NodeMailer, an open-source solution, provides extensive support for many email transport protocols such as SMTP, OAuth2, and direct transfer. This versatility allows it to easily adapt to diverse email delivery requirements. The integration of this software with Node.js apps is smooth, enabling fast administration and modification of emails straight from the server-side code. NodeMailer's capacity to manage attachments, HTML content, and diverse email templates guarantees its ability to meet intricate email sending needs while upholding dependability and efficiency.

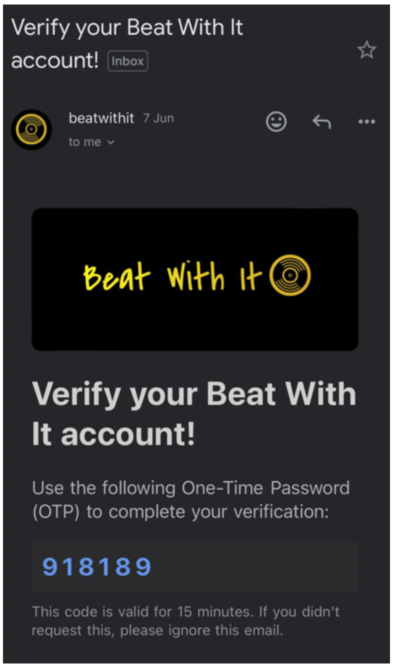


Figure 4.8: 2FA email sent with the help of NodeMailer.

Furthermore, the express-rate-limit feature was included to regulate the frequency of incoming requests and safeguard against misuse or excessive stress on the system. This middleware is crucial for bolstering the security and reliability of the application by implementing request rate restrictions for various endpoints. By implementing the express-rate-limit configuration, the system may efficiently reduce the likelihood of brute-force assaults and other harmful operations. It enables the implementation of customizable rate restriction rules, which includes the ability to specify the maximum number of requests allowed within a certain time frame (180 for Beat With It).

To extract data from TikTok's For You section, we had to create our own solution since there is no official API available for this specific task. In order to do this, we used the Puppeteer library in conjunction with Playwright. Puppeteer is a Node framework that offers a sophisticated API for managing Chrome or Chromium using the DevTools Protocol. It enables us to systematically browse TikTok's online interface, engage with items, and retrieve the required data. Playwright enhances Puppeteer by providing cross-browser automation features, guaranteeing that our solution is both efficient and flexible across many browser contexts. By harnessing the power of Puppeteer and Playwright, we successfully developed a resilient and dependable system for extracting TikTok data, a crucial component for the optimal performance of Beat With It.

Beat With It utilizes the "axios-retry" package, which serves as a wrapper for axios and provides additional functionality. It connects to an axios instance and offers a simple method to retry a request with minimum code. This feature is very valuable in our application since we regularly come across 429 status codes from Spotify, which indicate rate restrictions. "axios-retry" manages the duration of waiting for us and thereafter attempts the request again.

In addition, the project made use of dotenv to handle environment variables. dotenv is a module that has no dependencies and is used to import environment variables from a '.env' file into the 'process.env' object. This method enables the safe and easy customization of the application's environment-specific parameters, such as API keys, database connection strings, and other confidential data.

Moreover, Beat With It utilizes the Node.js library "node-cron" to automate the daily updates of various pieces of information. Given the large volume of data that needs regular updating, these cron jobs are essential for handling tasks programmatically and ensuring the system remains current and accurate.

4.4 Features, implementation details and user manual

This section will go over each Beat With It feature in depth, looking at how it was built. We will also talk about the connections between each part and the others to construct the entire Beat With It system. Detailed feature descriptions, technical implementation details, and user manuals will all be included in this extensive overview.

4.4.1 Detailed Assets Overview

The focal point of the program is its assets, which are mainly audios from TikTok clips paired with Shazam and Spotify data. Since the majority of these assets contain unique data that isn't available on other platforms, this entity serves as the platform's cornerstone. An extensive summary of assets and their importance within the application ecosystem is given in this subsection.

Assets Homescreen

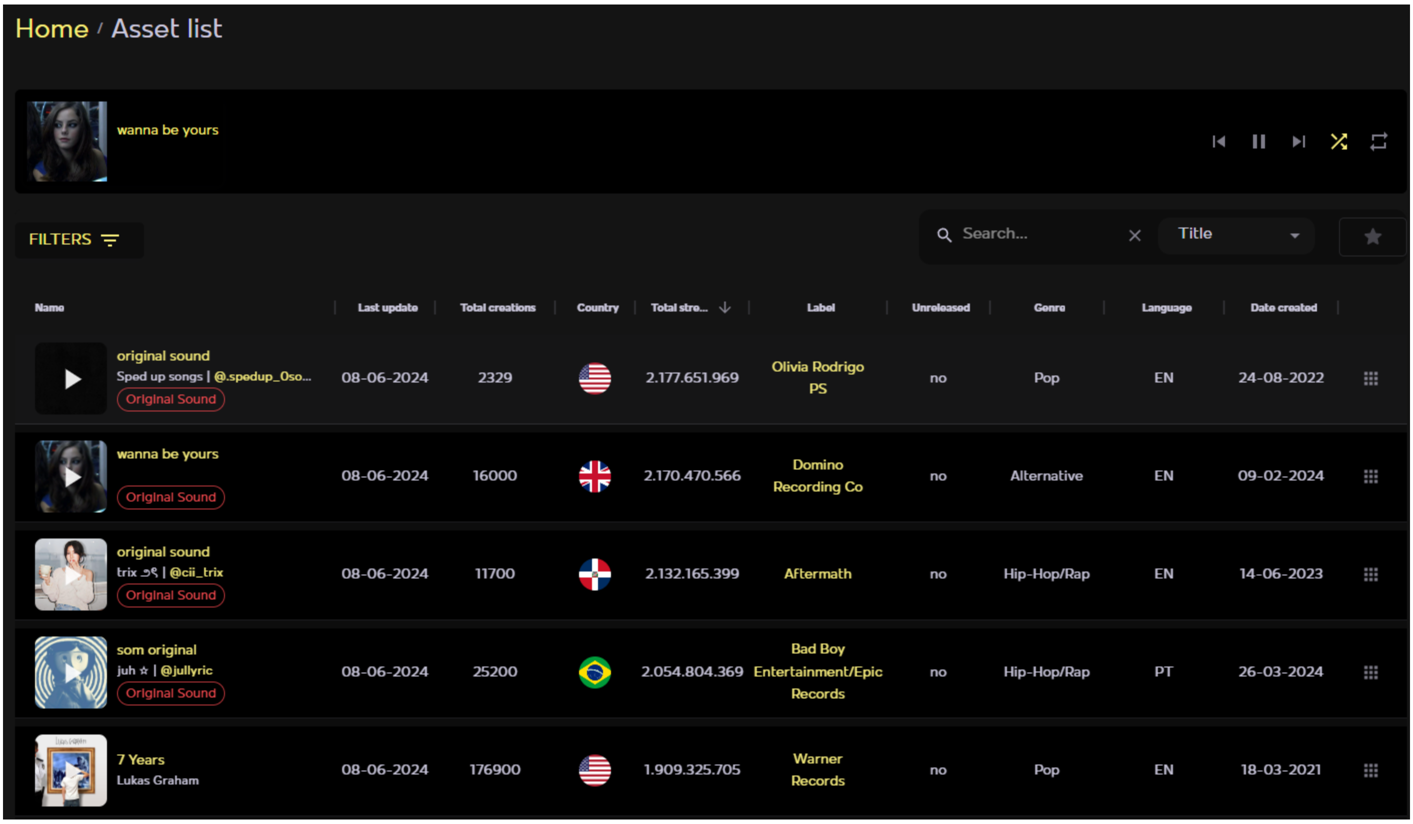


Figure 4.9: Beat With It assets in descending order by the total number of streams.

The user can play music either from the Audio Player either from the Play Button. He can also search, filter, hide, add to watchlists or open the Asset Details.

To better understand how assets work, we need to examine the information they hold in their internal representation:

- Platform ID: This column holds the TikTok ID for that asset.
- Title: The title of the asset from TikTok
- Author: The author's nickname of the asset from TikTok
- Last Update: A datetime column that indicates when the asset was last crawled and updated.
- Total creations: It represents the total number of videos from TikTok that use that audio
- Country: The country associated with that specific asset. The country is deduced using two fallback methods:
  - from the region of the author who created the audio on TikTok
  - if the region of the author is not found, the country is determined from the ISRC (International Standard Recording Code), if the song is matched. The first two characters of the ISRC represent the country where the song was released
- Original Sound: This column designates if the sound in the asset is an original sound or a delivered sound. An original sound is one that is recorded by the user just for the TikTok clip and is not taken from an already-existing audio file. A delivered sound comes from a source or a label, such as an audio file supplied by a music label or an already-existing song.
- Date Created: A column that approximates the date when the audio was created. There are currently four fallback methods for determining this date:
  - if the original video is still available, we take the date from it.
  - if the song is matched, we check if Shazam provides a release date.
  - if Shazam does not provide a match or a release date, we try to find the oldest video using that sound.
  - We use the resource-intensive TikTok Mobile API endpoint that we established as a last resort if none of the earlier techniques work.
- Label ID: The label's ID that is associated with the asset. A label is assigned to an asset only if a Shazam match has been found.
- Owner Handle: The handle of the author to the TikTok platform

- User Language: The author's language. It helps identifying the language in which the asset is.
- Spotify Track ID: The Spotify ID of the song linked to the asset.
- Genre: This column indicates the genre of the matched Shazam song. The genre is only available if there was a match between Shazam and the asset's audio.
- Play URL: This column contains the URL to the asset's audio from TikTok.
- Growths: We discussed about how Beat With It offers information on how the songs and assets are performed. At the moment, the program computes six growth measures in total:
  - TikTok Growths: We determine how many new videos were made with the asset during the previous week and the previous day.
  - Spotify Growths: We figure out how many new streams there have been of the music in the past week and the previous day.
  - Spotify Percentage Growths: We compute the growth percentage of the streams over the last week and the last day.

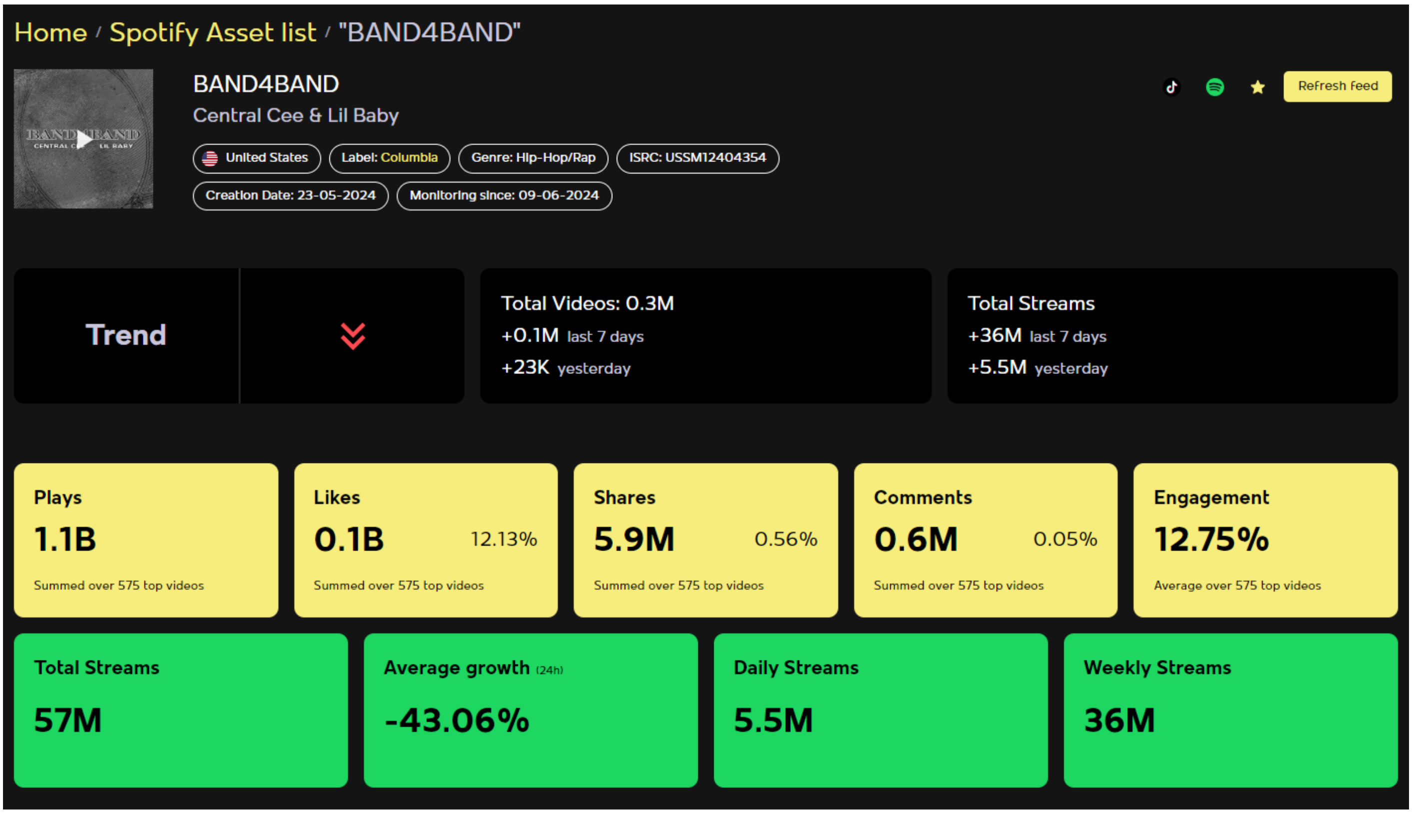


Figure 4.10: Asset Details Page.

We can see the information about the asset as well as the data presented in Figure 4.10. The system alerts the user that it is presently retrieving video information for the particular asset when they visit the details page of an asset that has never been seen by anyone.

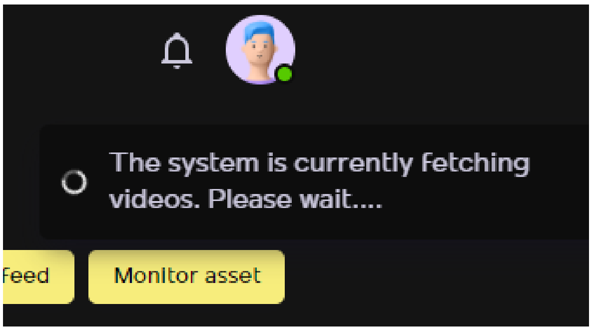


Figure 4.11: Toast message when viewing a never-opened asset.

The asset enters the "Get Videos Queue Subsystem" in the backend, where the process starts. Here, the total of all the columns of the videos—including play counts, likes, shares, comments, and like count—is entered into the yellow cards following the completion of the work and the fetching of a batch of films. The frontend updates as a result of receiving this data from the backend.

Subsequently, each video fetched enters the "Videos Queue Subsystem", where the system attempts to determine the region where the video was created. This allows for the creation of the country breakdown pie chart. However, the information is not sent automatically to the frontend due to the potential volume of data – sending more than 400 video information for each location found would be impractical. Instead, the asset countries are saved in the database. Users can then refresh the page or wait, as the asset information will be updated after all jobs for videos of that asset are completed.

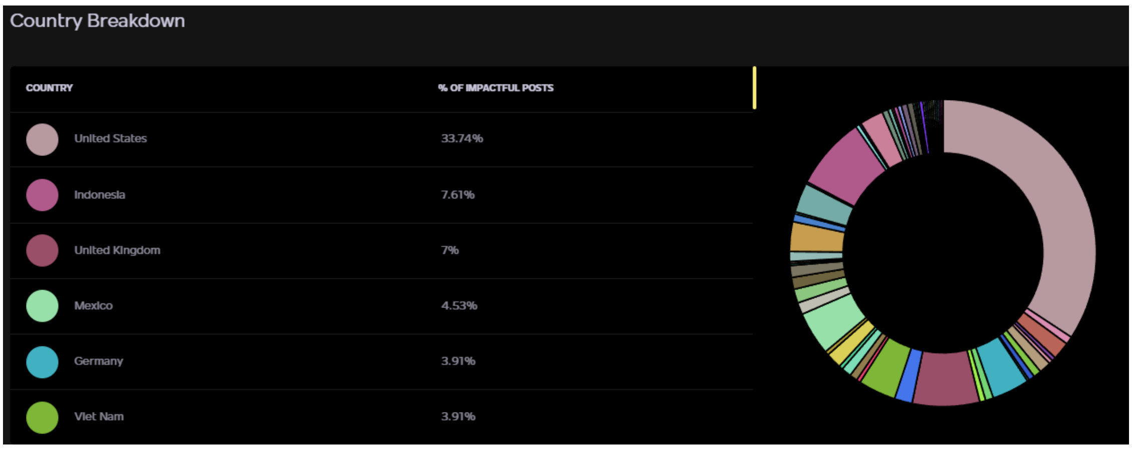


Figure 4.12: Asset usage by countries graph.

```
static async maybeFetchIfNeverFetched(asset: Asset): Promise<boolean> {
  let fetchedCommandSent : boolean  = false

  if (asset.last_creations > 0) {
    const hasBeenFetched : boolean  = await this.hasBeenNeverBeenFetched(asset.id) /// We check if there were any previous fetches for this asset
    if (!hasBeenFetched) {
      await this.create( values: { asset_id: asset.id }) // We create a record to mark that we are fetching videos for this asset

      const c : number  = await AssetVideo.count( options: { where: { asset_id: asset.id } }) // We check if there are any videos for this asset
      if (c === 0) {
        fetchedCommandSent = true
        const queue : Queue<any, any, string>  = new QueueService().getQueue(Queues.GET_VIDEOS);
        queue.add(JobType.GET_VIDEOS,  data: { asset_ids: [asset.id] }) // We send a command to fetch videos for this asset
      }
    }
  }

  return fetchedCommandSent // We return whether we sent a command to fetch videos for this asset or not
}
```

Figure 4.13: Adding the asset to the "Videos Queue Subsystem".

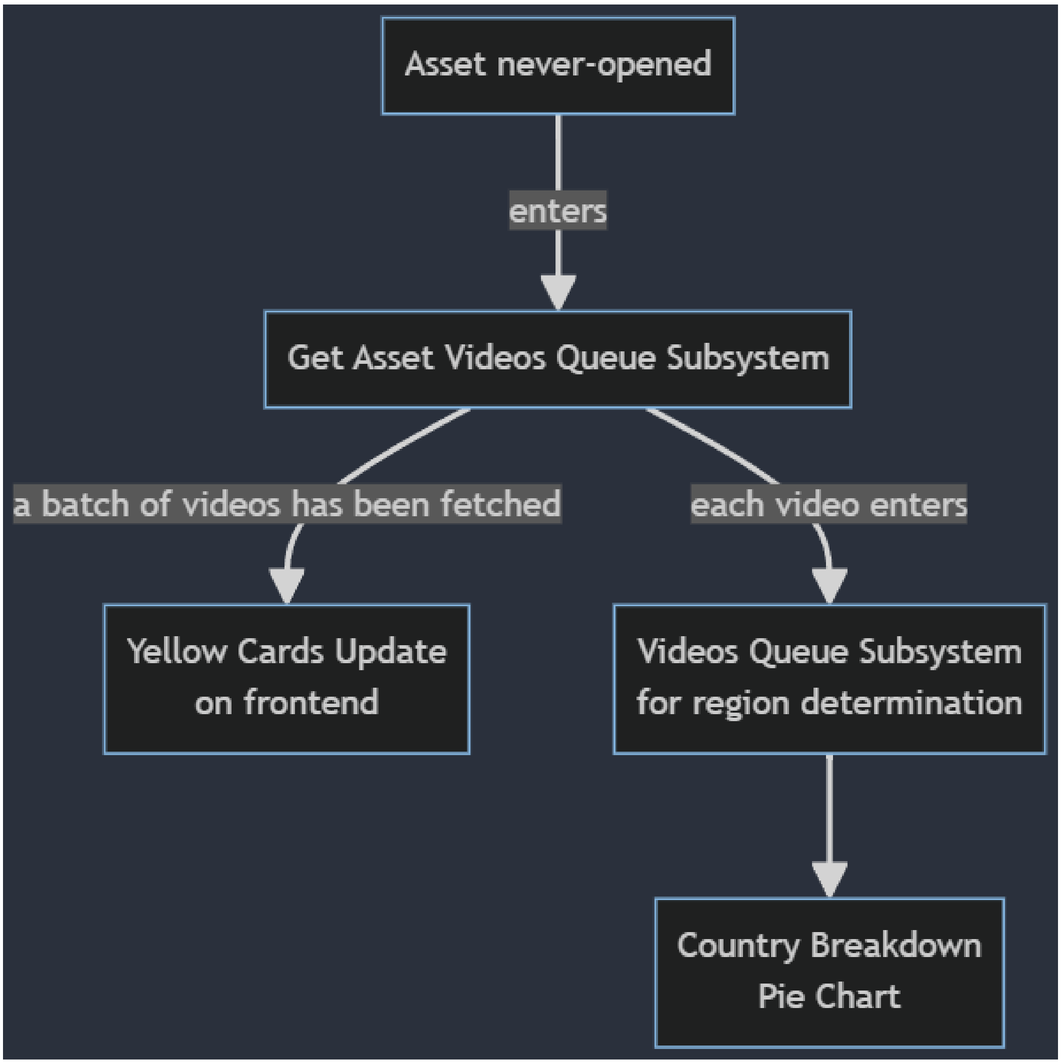


Figure 4.14: Videos Queue Subsystem Workflow Diagram

A key component of the Asset Details are the Performance Graphs, which are separated into "new" and "total" subcomponents. The "new" graph offers insights into current trends and changes by displaying the daily growths from Spotify and TikTok, meanwhile, Spotify streams and TikTok video creations cumulative totals are shown on the "total" graph.

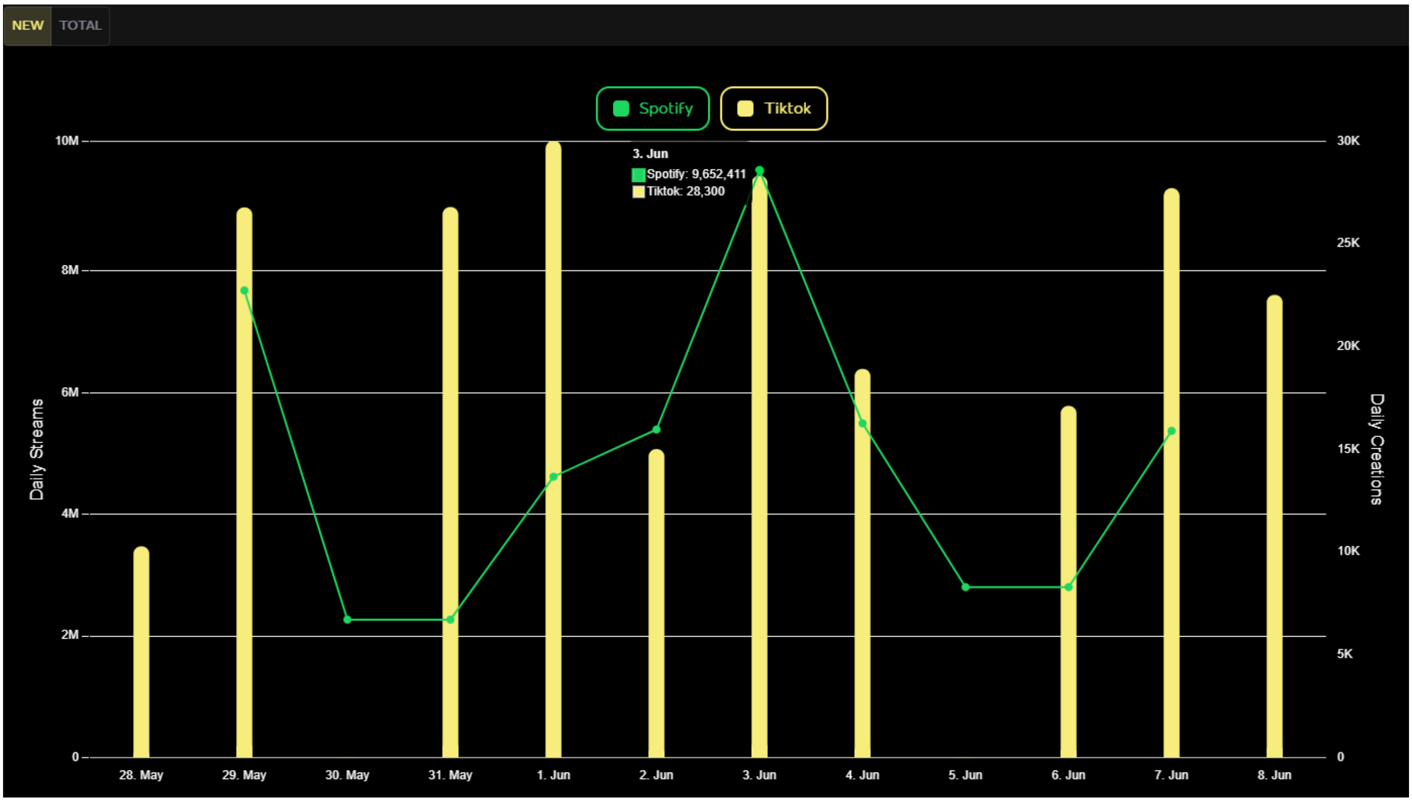


Figure 4.15: An asset's "new" graph, based on daily growths.

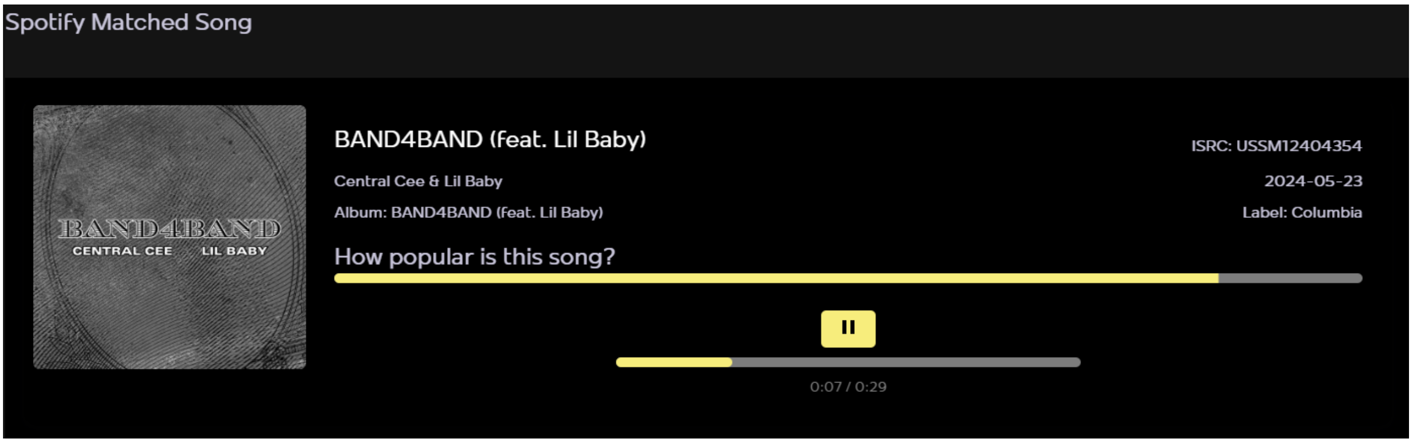


Figure 4.16: Spotify Matched Song of an asset.

Only when an asset is linked to a Spotify Track ID can the Spotify Matched Song component be accessed on the Asset Details page. Moreover, if the user is connected with his Spotify Account, he can add the song to his "Saved Songs" Playlist.

Asset actions: Search, Filtering, Hide, Add to watchlists

The assets are used by the user for various purposes. Searching, filtering, hiding, and adding to watchlists are the most crucial ones. He has the option to search for several fields inside the asset, such as Title, Author, Platform ID, and ISRC.

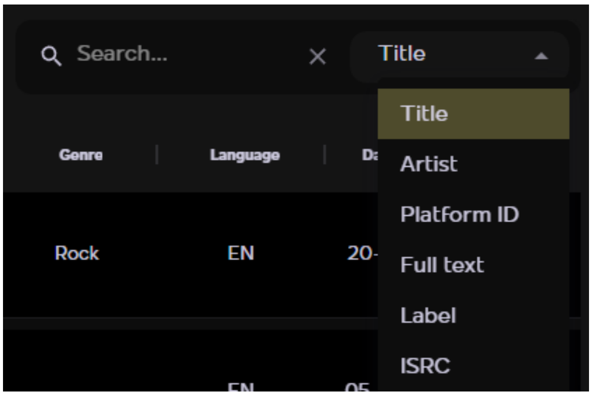


Figure 4.17: Assets Search Toolbar.

Additionally, he has the option to filter the assets using a variety of filters. Every filter contains a tooltip that describes the filter's operation to the user. Moreover, he may add, modify, and remove filter presets. Because you can save a ton of filters and apply them with just one click, they are quite helpful.

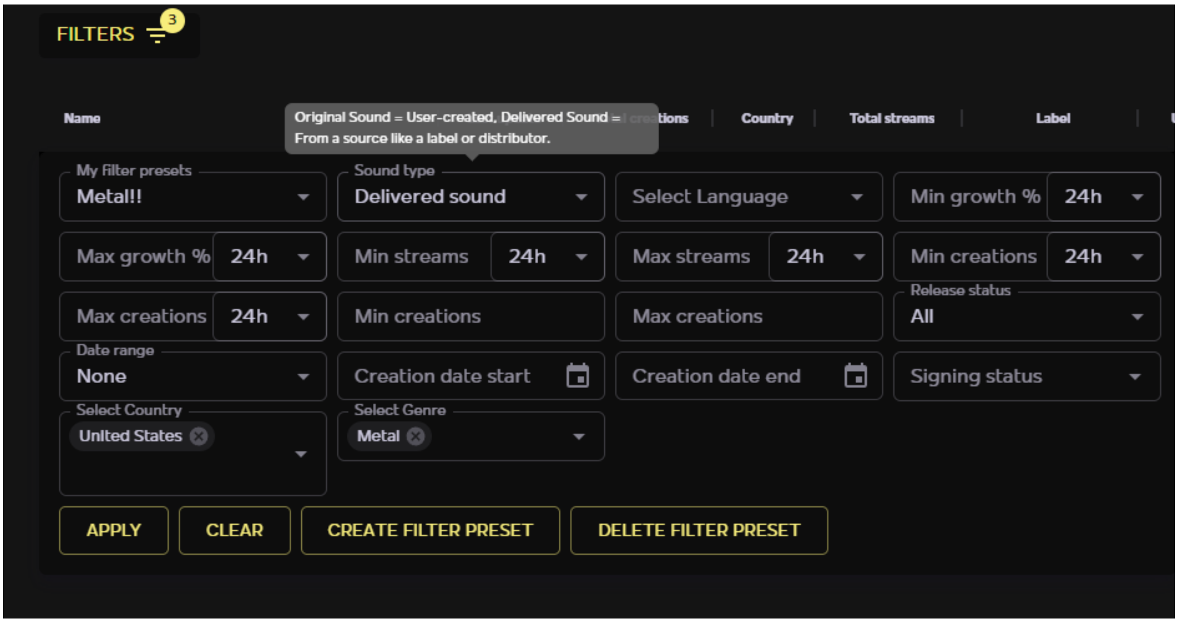


Figure 4.18: Assets Filters Toolbar.

The user can choose to add an asset to one or more of his existing watchlists if he finds it interesting or if it meets his requirements, but he can also instantly establish new watchlists. Moreover, by selecting the star button from the toolbar, the user can decide to add all the filtered assets to a particular watchlist after applying a filter.

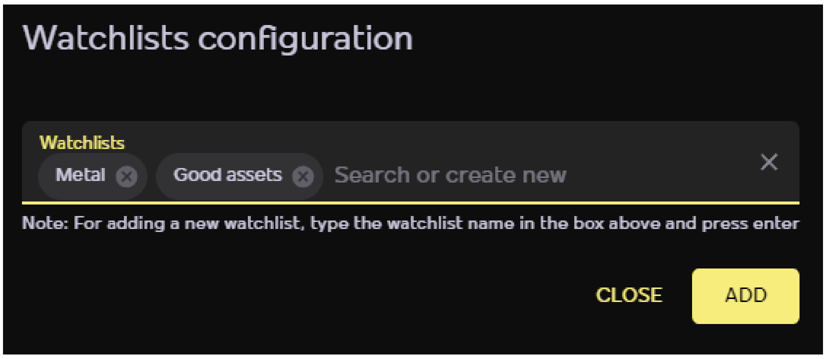


Figure 4.19: Add an asset to multiple or new watchlists.

The user has the option to conceal an asset if he cannot find any relevance for it. In the unlikely event that the user decides they are interested in that item once more, they may view all of their hidden assets by going to Profile -¿ My Hidden Assets and even disclose them to their original lists.

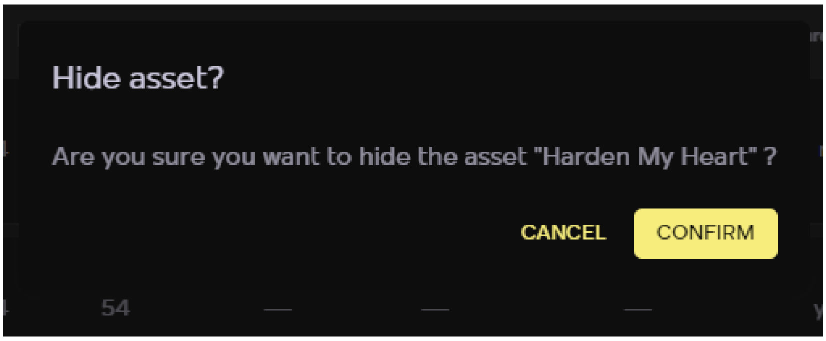


Figure 4.20: Before hiding an asset, a confirmation modal appears.

Add Asset

The user has the possibility to contribute to the platform by adding his favorite TikTok music. The process outlined in section 4.2.2 is followed by the new asset.

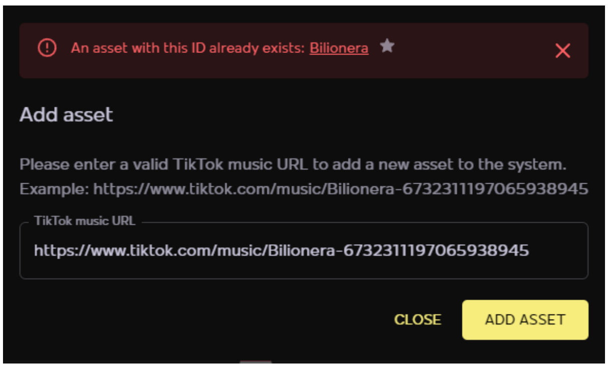


Figure 4.21: Adding an existing asset to the system.

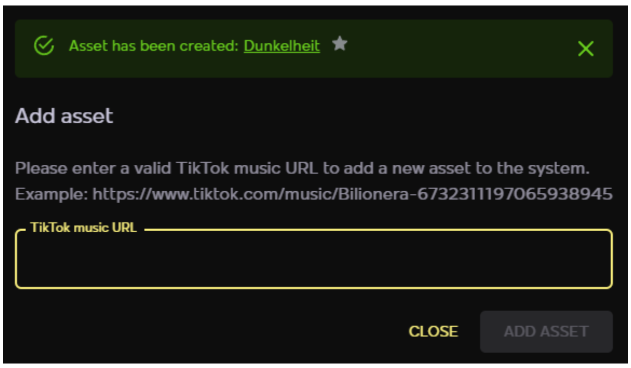
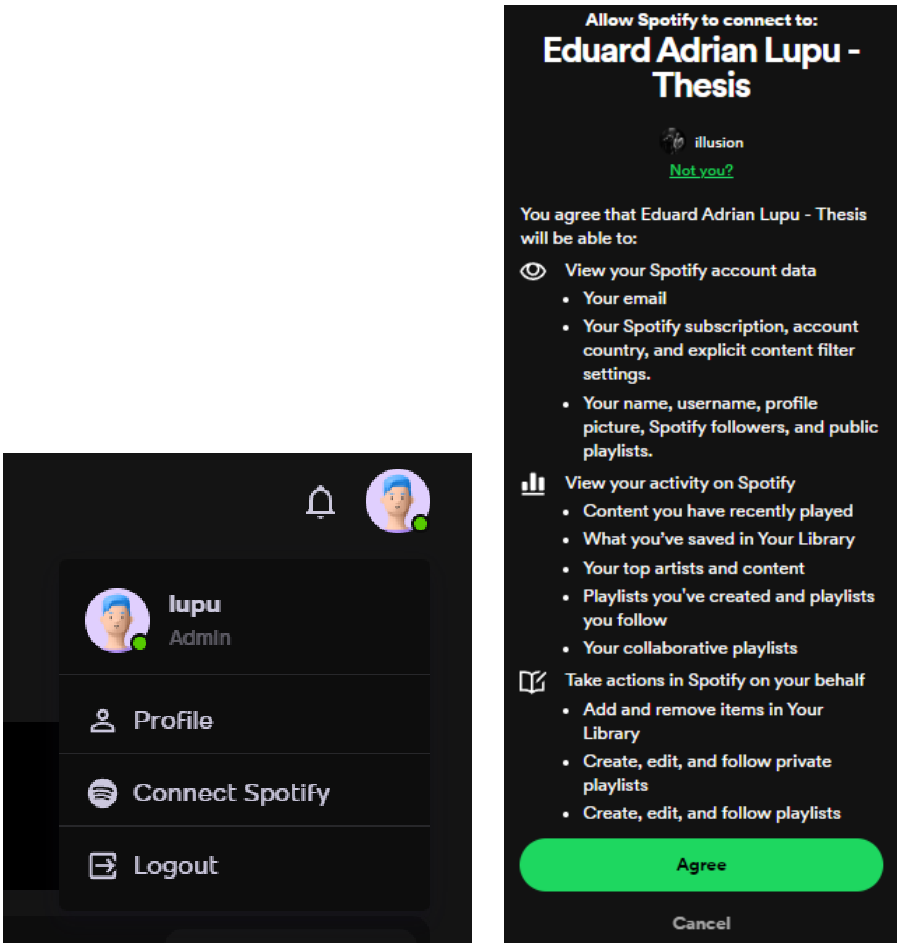


Figure 4.22: Adding a new asset to the system.

4.4.2 Recommendations

Another crucial component of Beat With It is the recommendations component. For recommendations to function, the user must first connect their Spotify account to Beat With It.



(1) Press "Connect Spotify" (2) Allow Spotify to connect

Figure 4.23: Connect Spotify account to Beat With it instructions

Once connected, the user can navigate to the Recommendations component from the vertical menu. Here, they will find all the possible parameters described in Chapter 3, Section 3.2.2.

Alternatively, users can seek recommendations by visiting a Spotify Asset Details Page. In this case, automatic recommendations for that song will be generated, as the seed will contain the Spotify Track ID of the asset.

If the user is connected to Spotify, he also has the possibility to add the recommended tracks or any Spotify Song from Beat With It to his "Saved Song" Playlist on Spotify.

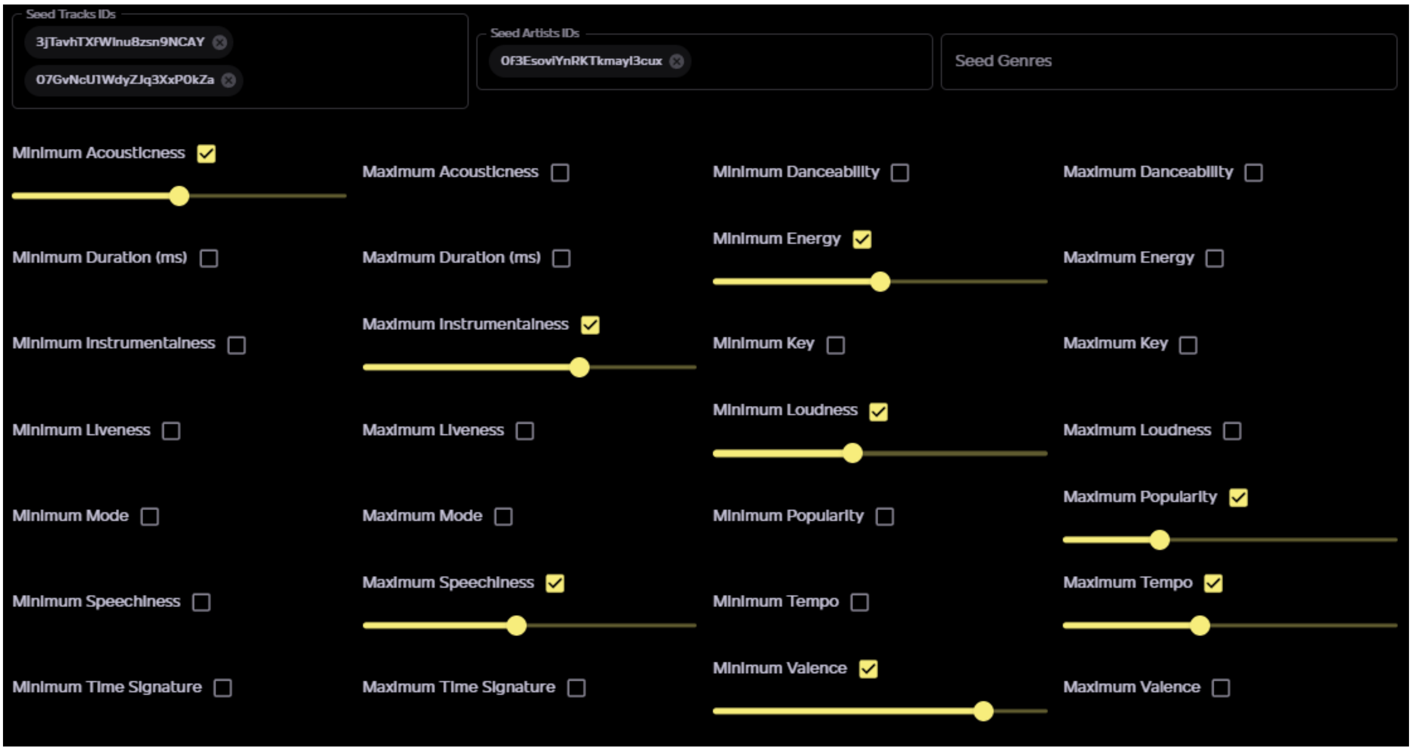


Figure 4.24: Recommendations filters from Beat With It.

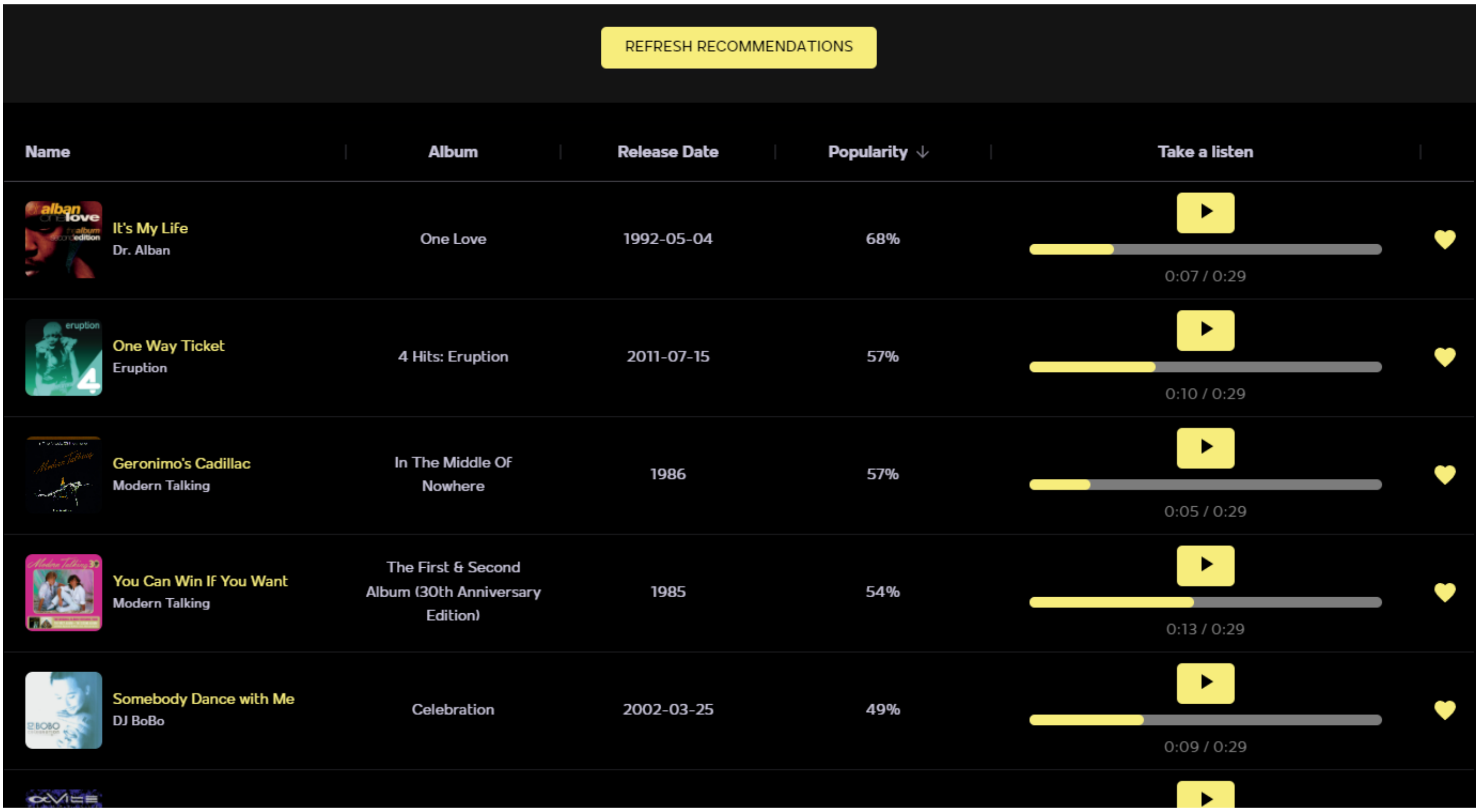


Figure 4.25: Suggested songs for "Pretty Young Girl" by Bad Boys Blue, sorted descendingly based on popularity, with a minimum energy of 0.68.

Since Spotify manages the recommendations, no backend logic is needed for this capability, hence the recommendations are only implemented in the frontend.

```
const RecommendationsSongsTable: React.FC<RecommendationsSongsTableProps> = ({ props : SpotifyRecommendationsProps , refreshValue : boolean  }) => {
    const [loading : boolean , setLoading : React.Dispatch<React.SetStateA... ] = useState( initialState: false);
    const [songs : ChartSong[] , setSongs : React.Dispatch<React.SetStateA... ] = useState<ChartSong[]>( initialState: []);
    const [rowCount : number , setRowCount : React.Dispatch<React.SetStateA... ] = useState<number>( initialState: 0);
    const [rowsPerPage : number , setRowsPerPage : React.Dispatch<React.SetStateA... ] = useState<number>( initialState: 25);

    useEffect( effect: () : void  => {
        if (props.seed_tracks?.length !== 0 || props.seed_artists?.length !== 0 || props.seed_genres?.length !== 0) {
            setLoading( value: true);
            getSpotifySongRecommendations(props).then((response: any) : void  => {
                const tracks = response.data.tracks.map((track: any) : {...}  => {
                    return {
                        id: track.id ?? null,
                        name: track.name ?? null,
                        artist: track.artists?.map((artist: any) => artist.name).join(", ") ?? null,
                        album: track.album?.name ?? null,
                        release_date: track.album?.release_date ?? null,
                        cover_image: track.album?.images?.[0]?.url ?? null,
                        spotify_track_id: track.id ?? null,
                        preview_url: track.preview_url ?? null,
                        popularity: track.popularity ?? null
                    }
                })
                setSongs(tracks);
                setRowCount(tracks.length);
                setLoading( value: false);
            });
        }
    }, deps: [refreshValue]);
```

Figure 4.26: Implementation of the RecommendationsSongsTable component, written in Next.js with TypeScript, used in the menu item and in Asset Details Page.

4.4.3 Charts

In Beat With It, we feature charts from TikTok, Spotify, and Shazam. These charts are updated daily using a cron job to identify the most popular songs. The TikTok chart is unique as it not only displays the most relevant songs but also automatically adds new assets to the system, with the process described in Section 4.2.2.

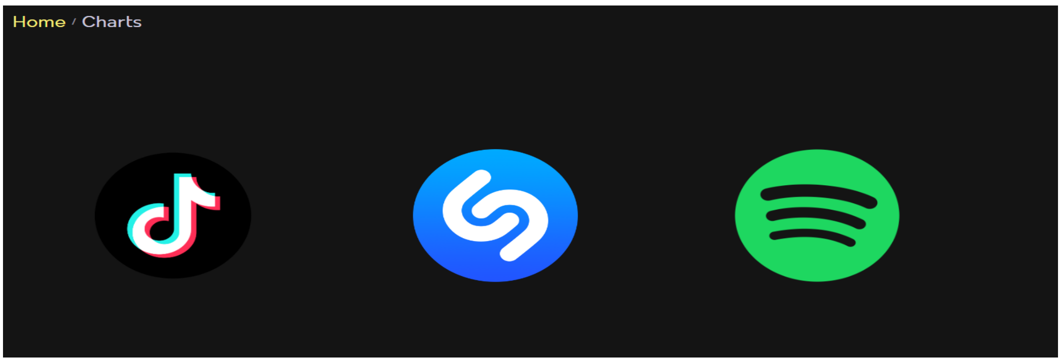


Figure 4.27: Charts Homepage.

As the charts run per country, managing charts for all countries (3 charts $\times$ 195 countries) would result in 585 configurations, which could overload the system. To address this issue, the Charts feature is configured to run only for specific countries set by the admin. This setting can be adjusted in the Settings tab, accessible exclusively to the admin.

```
{
    schedule: '0 0 21 * * *', // Every day at 21:00
    options: {
        timezone: 'Europe/Bucharest'
    },
    task: async () : Promise<void>  => {
        await getChartInstance( key: 'spotify').updateChart()
    }
},
{
    schedule: '0 0 3 * * *', // Every day at 3:00
    options: {
        timezone: 'Europe/Bucharest'
    },
    task: async () : Promise<void>  => {
        await getChartInstance( key: 'shazam').updateChart()
    }
},
{
    schedule: '0 0 5 * * *', // Every day at 5:00
    options: {
        timezone: 'Europe/Bucharest'
    },
    task: async () : Promise<void>  => {
        await getChartInstance( key: 'tiktok').updateChart()
    }
```

Figure 4.28: Cron Configurations for the Charts.

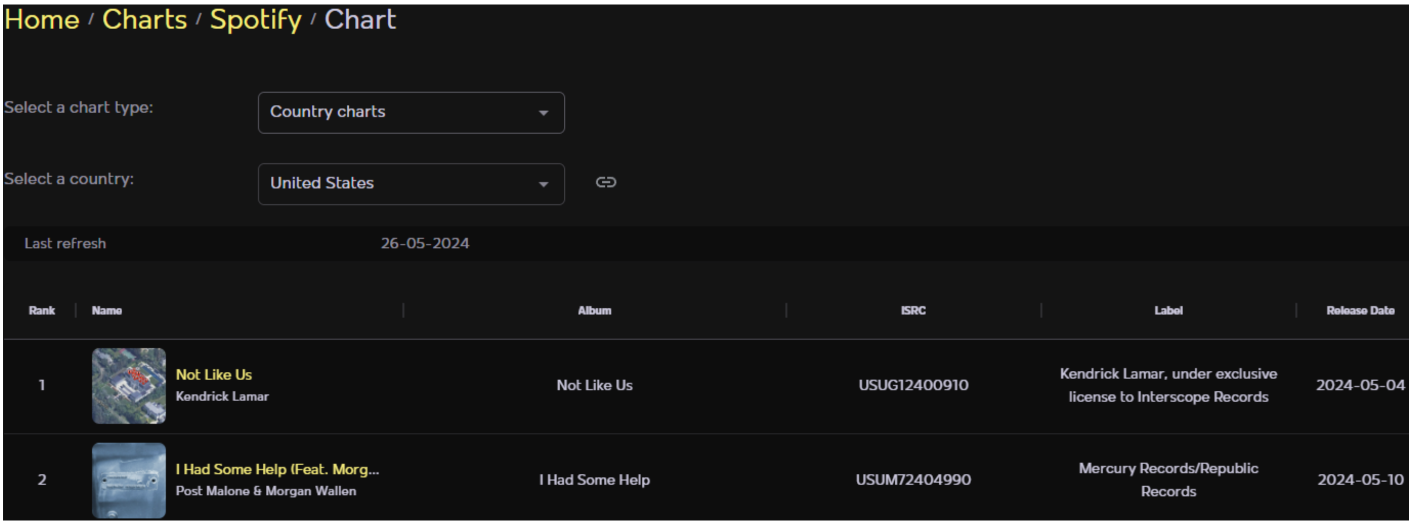


Figure 4.29: Spotify Chart.

Each chart also includes a link to its corresponding page on each platform. When a user clicks on an asset, a page will load displaying the assets from Beat With It that share the same ISRC.

4.4.4 Digital Scout

Digital Scout is a key component of Beat With It, serving as the third method by which the system adds assets. It functions similarly to manually scrolling through TikTok and adding every music audio you find into Beat With It, but the system automates this process for you.

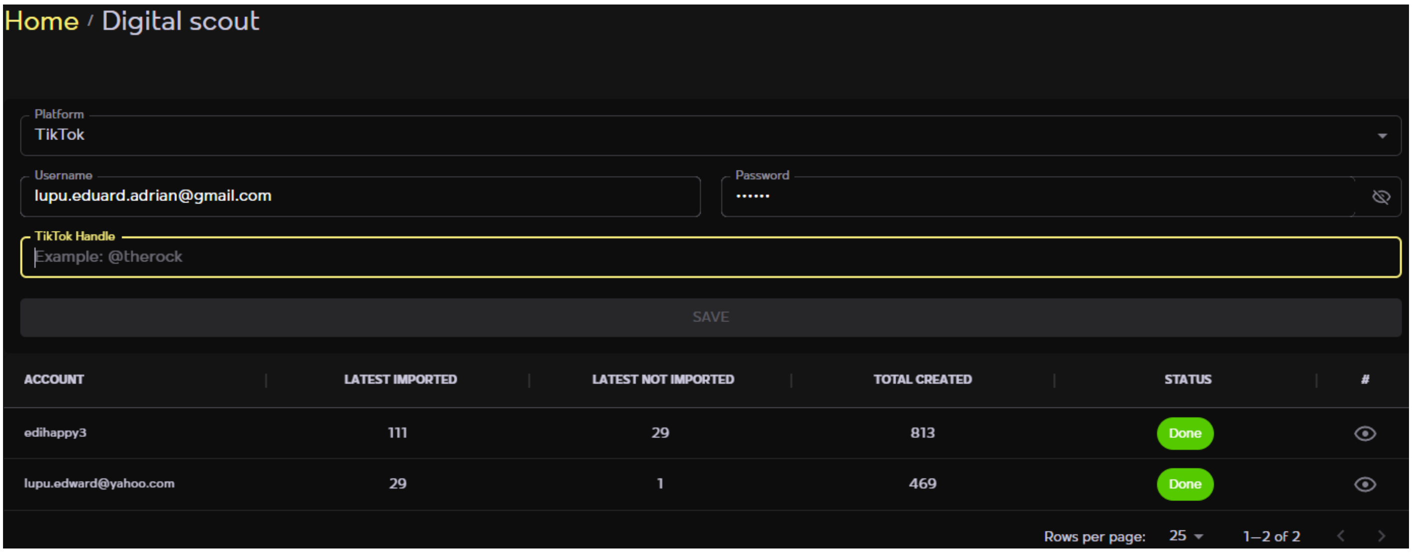


Figure 4.30: Digital Scout Page View.

You can view the assets added by your account, track the total number accumulated, and more. While you can use your Digital Scout as many times a day as you like, the system will automatically use your account once daily with a cron job to add assets to the platform. The Digital Scout process operates as described in Section 4.2.2.

4.4.5 User Search Jobs and Alerts

User Search Jobs and Alerts are similar in concept but differ in their scope and application. Both act as dynamic filters on assets, but User Search Jobs operate on all assets within the system, whereas Alerts are configured specifically for the assets in the selected watchlists during the Alert creation process.

They are updated daily using two separate cron jobs, one for each. This allows users to create User Search Jobs and Alerts, receiving notifications in the application and via email about new assets that match their configured filters.

This feature keeps users engaged with Beat With It by continuously providing them with assets that meet their criteria. Users can add, edit, delete, pause, and view their Alerts and Search Jobs.

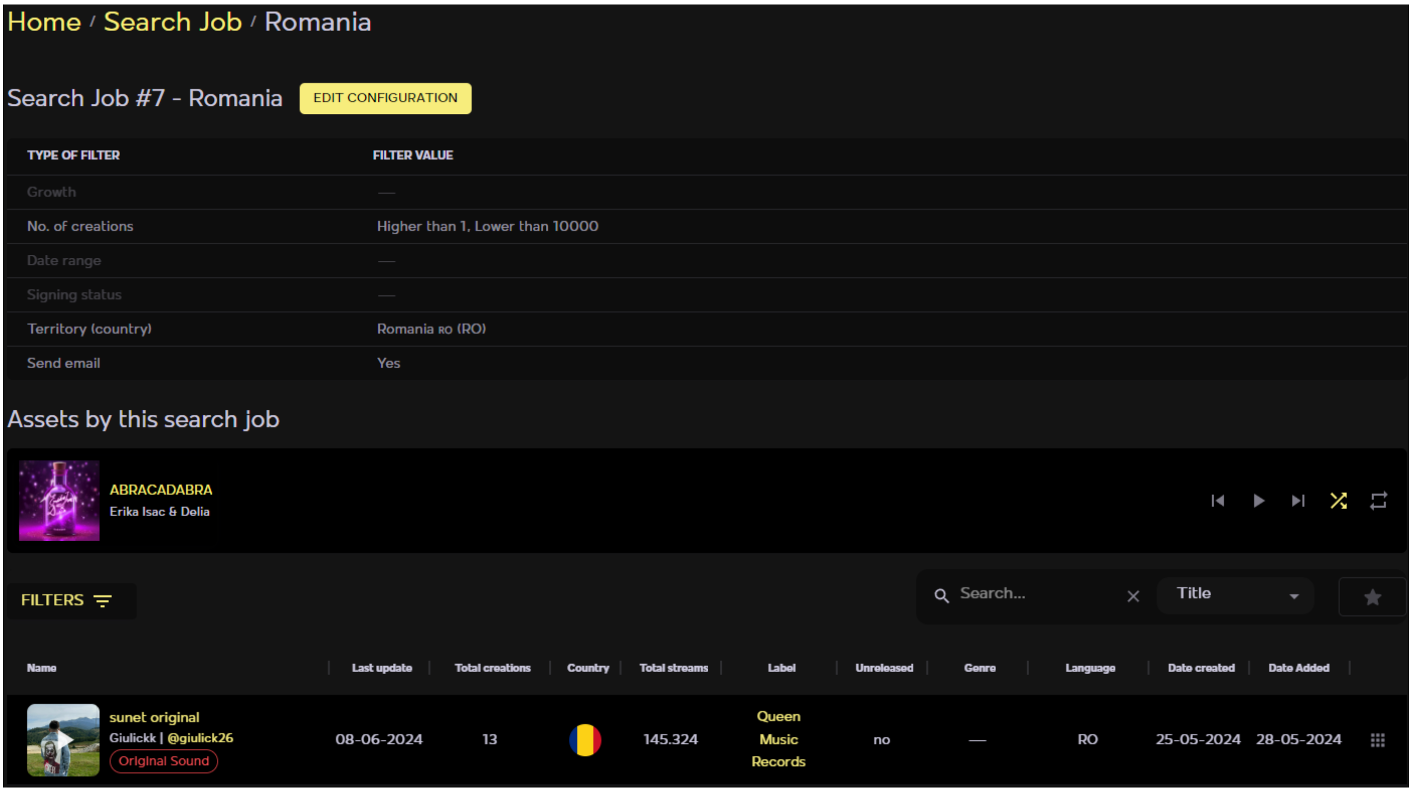


Figure 4.31: Search Job Details Page.

Figure 4.32: Editing the configuration of a Search Job.

4.4.6 User Management

The application is divided into two distinct layers, the admin layer and the client layer, by the installation of the User Management capability. Administrators can successfully manage user accounts, access restricted data, and carry out administrative activities thanks to the admin layer's complete access to all information and operations within the program. In contrast, the client layer protects the system's security and confidentiality by preventing regular users from accessing private or restricted information.

Users get access to key functions like user registration, login, profile management, and basic account settings in the client layer. Users are guided through their interactions with the platform with ease because to the intuitive, user-friendly, and responsive design of the user interface. But administrators only have access to specific functions and data, guaranteeing that private and sensitive information is kept safe.

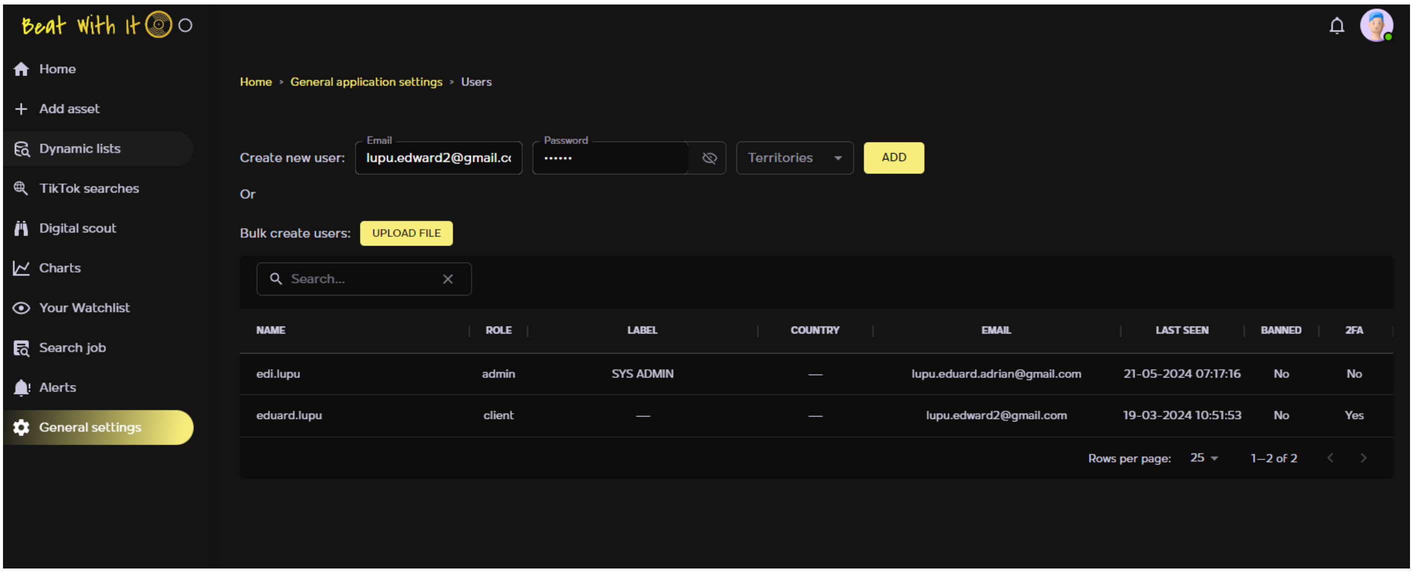


Figure 4.33: Admin Panel for user management

Administrators have complete visibility and control over the data and features of the application thanks to the admin layer. Administrators are able to create and remove users in bulk, set roles and permissions, view comprehensive activity logs for each user, and manage user accounts. They may also prohibit users as needed. The User Management functionality prioritizes user privacy and security by dividing the program into separate client and admin layers. It also gives administrators the necessary tools and access to manage the platform efficiently.

User Authentication and Security

The system securely manages the registration process for users by securely hashing new account passwords and verifying necessary inputs. The Forgot Password function emphasizes user convenience and account security by sending out a secure password reset email in the event that the user forgets their password. Users can safely reset their passwords using the Reset Password service, preserving the accessibility and security of their accounts.

Login Functionality ensures secure validation of user credentials, checks for account bans, and manages the 2FA process to protect user accounts from unauthorized access.

Strong security measures are built into the system to guarantee user account integrity and safety. OTP Verification securely verifies OTP codes during the two-factor authentication (2FA) procedure, improving account security. The Resend OTP tool ensures quick authentication access by facilitating the prompt distribution of OTP verification emails, hence improving user experience.

```javascript
module.exports.login = async (req, res) => {
  try {
    const user = await User.findOne({ where: { email: req.body.email } })
    if (!user || !user.id) {
      return res.status(401).json({ message: 'Invalid email / password' })
    }
    if (user.isBanned()) {
      return res.status(401).json({ message: 'Your account has been banned!' })
    }
    const isPasswordValid = await bcrypt.compare(req.body.password, user.password)
    if (!isPasswordValid) {
      return res.status(401).json({ message: 'Invalid email / password' })
    }
    if (!user.is2FAEnabled()) {
      const accessToken = jwt.sign({ id: user.id, role: user.role }, jwtConfig.secret, {
        expiresIn: jwtConfig.expirationTime
      })
      return res.send(
        JSON.stringify({
          accessToken,
          userData: {
            ...user.toJSON(),
            password: undefined
          },
          redirect_url: user.role === 'admin' ? '/asset-list' : '/tiktok-assets'
        })
      )
    } else {
      await email.sendOTPVerificationEmail(user.id, user.email)
      return res.status(200).json({
        userData: {
```

Figure 4.34: Login function implementation

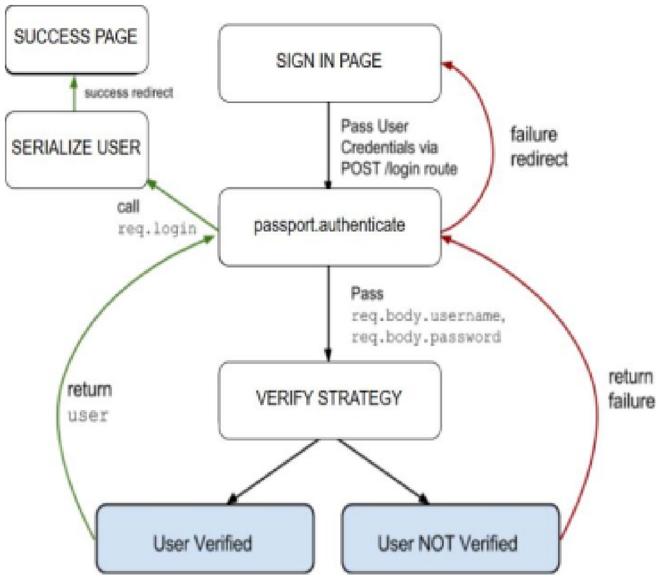


Figure 4.35: Authentication Workflow used in Beat With It. [KBB+18]

User Profile Management

With the help of the admin panel, administrators can easily manage user data. They can update passwords, enable/disable two-factor authentication (2FA), modify user profiles, keep an eye on user activity, ban or delete accounts, and send account activation emails, among other things. This extensive feature set prioritizes privacy and security protocols while guaranteeing that user information is handled effectively.

The administrator can also update the countries which an client can see in Asset List. This is effectively similar to every platform "Not available in your country". This provides better control on what the client sees in the application and reduces the risk of displaying assets which are not of interest.

Additionally, administrators have the capability to update the list of countries that clients can view in the Asset List. This functionality is akin to the "Not available in your country" feature commonly found on various platforms. By offering this level of control, administrators can ensure that clients only see assets relevant to their interests, thus minimizing the risk of displaying irrelevant content.

4.4.7 General application settings

This component is exclusively accessible to administrators and governs the operational parameters of Beat With It. It enables the modification of chart countries and the adjustment of the number of pages processed by each Digital Scout. This level of control grants administrators greater flexibility in managing the application's behavior. Rather than hardcoding these settings, administrators can dynamically observe and adjust the system's performance based on the current configuration.

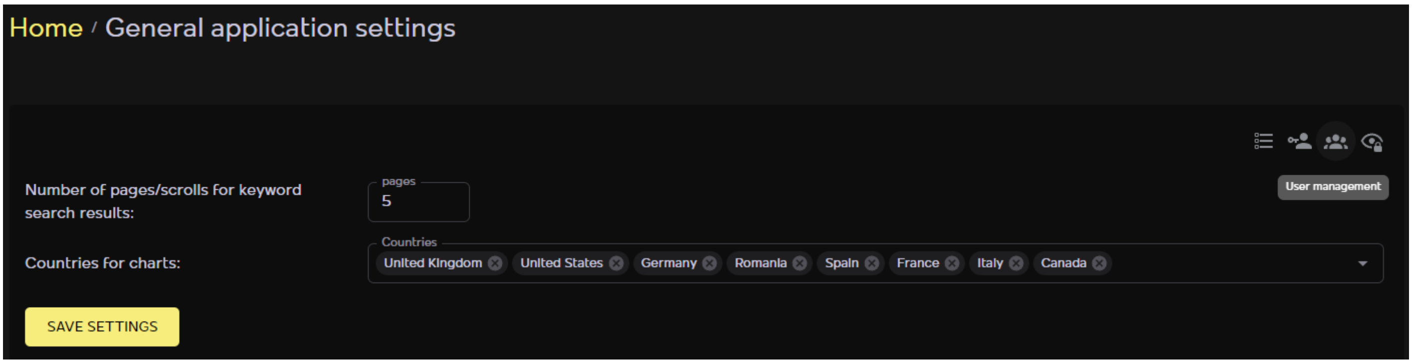


Figure 4.36: Settings Page of Beat With It.

In the Settings component, admins can access the Labels section, which offers full control over the labels identified by the system. Since labels can be categorized as Major or Indie (with Major labels typically associated with industry giants like Sony Music, Universal Music, and Warner Music), determining the classification solely by name can be challenging. This complexity arises from the fact that Major labels often encompass numerous sub-labels. Therefore, manual updates to label information may be necessary at times to ensure accuracy and relevance within the system.

The Hidden Assets component is located within the settings, providing administrators with visibility into assets that have been globally hidden. Administrators have the option to review these hidden assets and, if deemed necessary, restore them to their original lists by revealing them. This functionality offers administrators greater control over the visibility and management of assets within the system.

4.5 Evaluation and Testing

To verify and evaluate the functionalities of Beat With It, several evaluation methods were employed. In this section, we will delve into the most important ones.

4.5.1 Manual Testing

Manual testing was conducted extensively throughout the development process. This involved:

- Verifying user interactions with the system.

- Ensuring that assets could be added, searched, hidden, and filtered correctly.
- Testing the asset details view to confirm that all information is displayed accurately and that the listening functionality works as intended.

4.5.2 Automated Testing with Mocha

Automated tests were implemented using Mocha, focusing on:

- Unit tests to validate individual functions and methods.
- Integration tests to ensure that different parts of the system work together

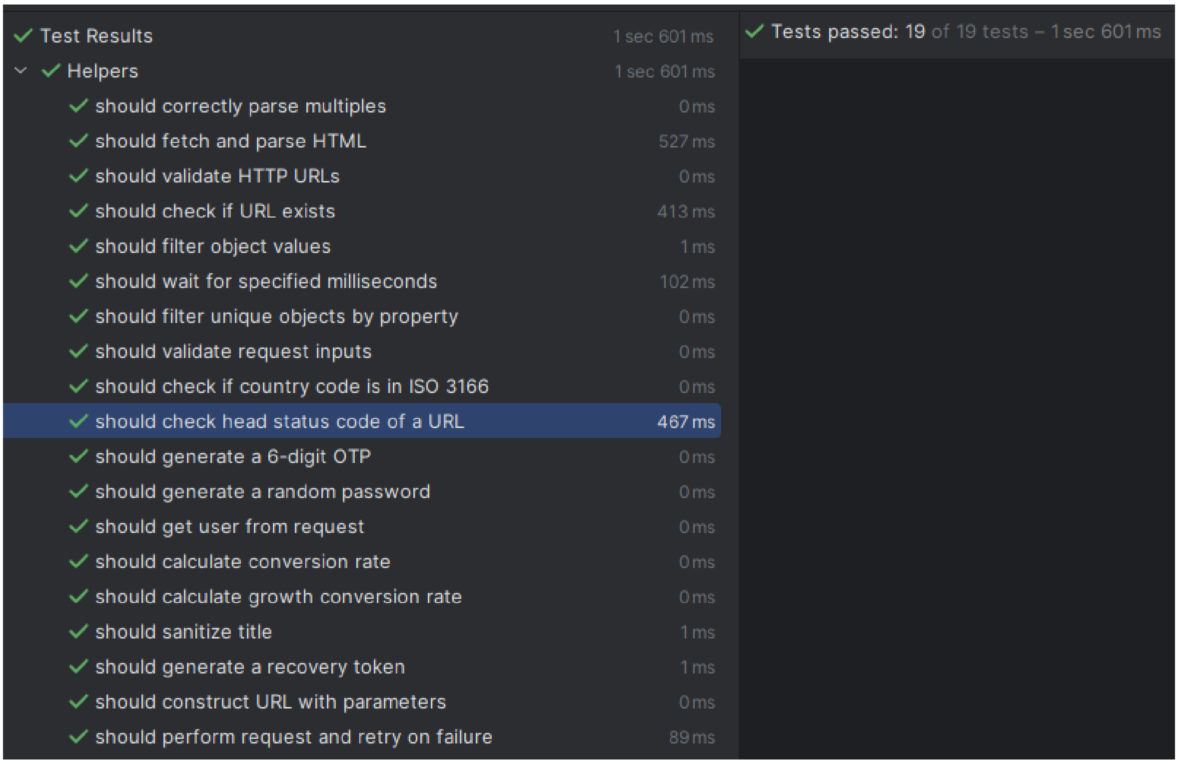


Figure 4.37: Helpers function tested using Mocha and unit tests.

4.5.3 Security and Request Handling with Burp Suite

Burp Suite was utilized to enhance the security testing of the system. This involved:

- Intercepting and analyzing HTTP requests and responses to verify that data is handled securely.
- Testing for vulnerabilities such as SQL injection, cross-site scripting (XSS), and other common security threats.

- Ensuring that proper authentication and authorization mechanisms are in place and functioning correctly.

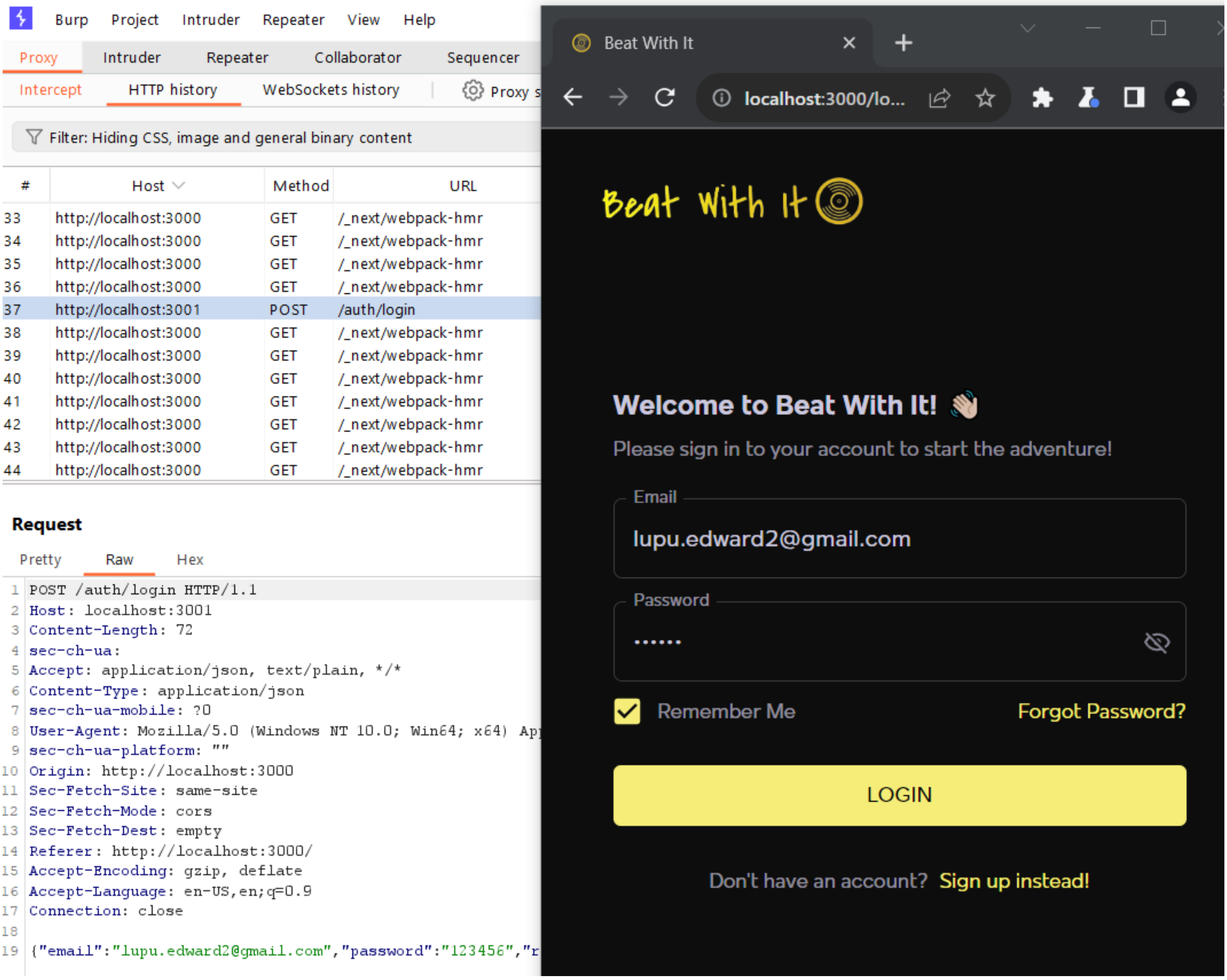
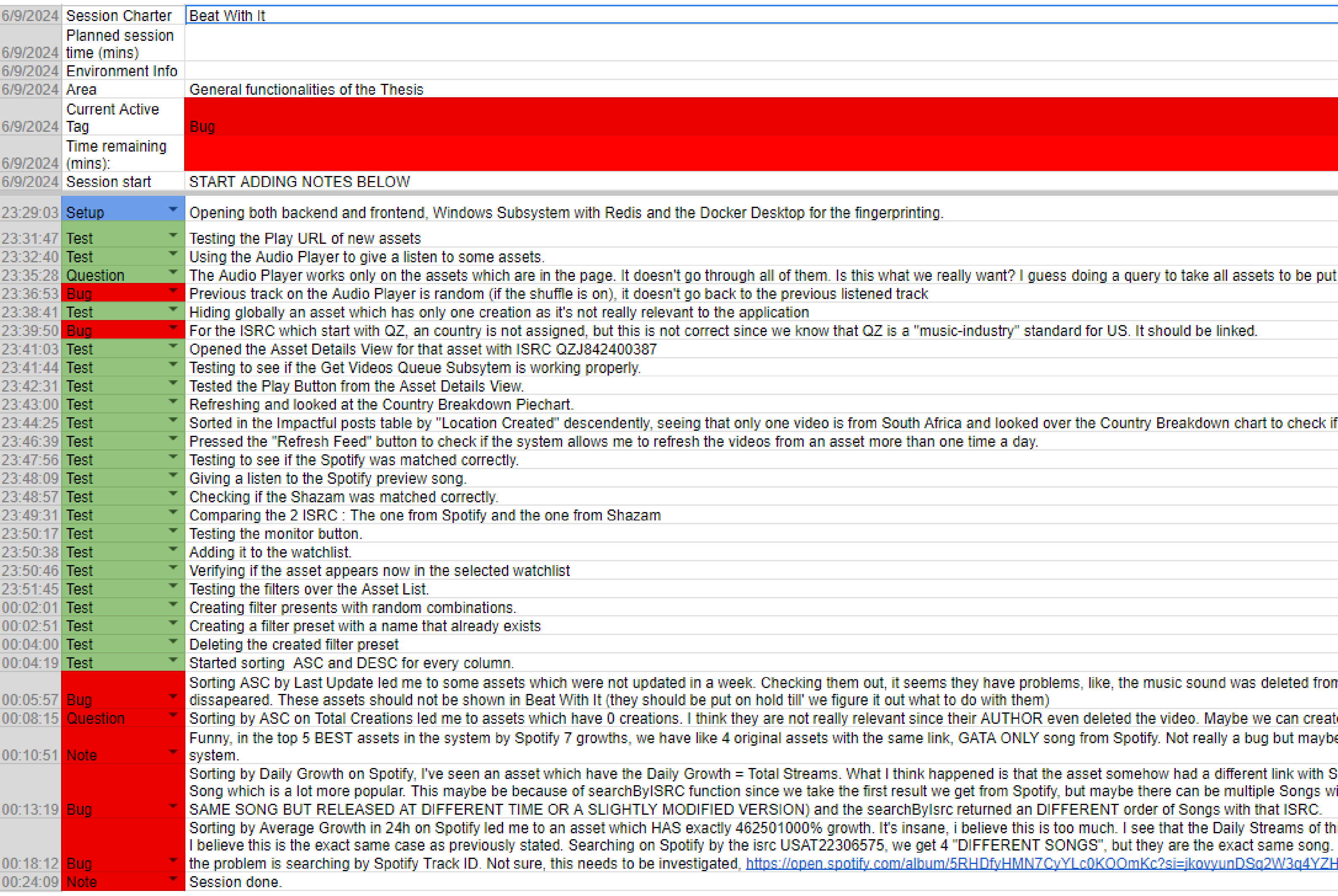


Figure 4.38: Burp Suite intercepting requests from Beat With It

4.5.4 Exploratory Testing using SBTM Method

One of the most potent and successful testing techniques is exploratory testing. It combines tasks like designing, carrying out, and documenting tests; its main goal is to teach the application being tested.

Additionally, exploratory testing requires a great deal of thought, which makes it very advantageous for the person conducting it. Because it focuses on learning the application being tested, this testing method requires users to use a variety of cognitive processes, including conscious, logical, and documentation thinking.



6/9/2024	Session Charter	Beat With It
6/9/2024	Planned session time (mins)	
6/9/2024	Environment Info	
6/9/2024	Area	General functionalities of the Thesis
6/9/2024	Current Active Tag	Bug
6/9/2024	Time remaining (mins):	
6/9/2024	Session start	START ADDING NOTES BELOW
23:29:03	Setup	Opening both backend and frontend, Windows Subsystem with Redis and the Docker Desktop for the fingerprinting.
23:31:47	Test	Testing the Play URL of new assets
23:32:40	Test	Using the Audio Player to give a listen to some assets.
23:35:28	Question	The Audio Player works only on the assets which are in the page. It doesn't go through all of them. Is this what we really want? I guess doing a query to take all assets to be put
23:36:53	Bug	Previous track on the Audio Player is random (if the shuffle is on), it doesn't go back to the previous listened track
23:38:41	Test	Hiding globally an asset which has only one creation as it's not really relevant to the application
23:39:50	Bug	For the ISRC which start with QZ, an country is not assigned, but this is not correct since we know that QZ is a "music-industry" standard for US. It should be linked.
23:41:03	Test	Opened the Asset Details View for that asset with ISRC QZJ842400387
23:41:44	Test	Testing to see if the Get Videos Queue Subsytem is working properly.
23:42:31	Test	Tested the Play Button from the Asset Details View.
23:43:00	Test	Refreshing and looked at the Country Breakdown Piechart.
23:44:25	Test	Sorted in the Impactful posts table by "Location Created" descendently, seeing that only one video is from South Africa and looked over the Country Breakdown chart to check if
23:46:39	Test	Pressed the "Refresh Feed" button to check if the system allows me to refresh the videos from an asset more than one time a day.
23:47:56	Test	Testing to see if the Spotify was matched correctly.
23:48:09	Test	Giving a listen to the Spotify preview song.
23:48:57	Test	Checking if the Shazam was matched correctly.
23:49:31	Test	Comparing the 2 ISRC : The one from Spotify and the one from Shazam
23:50:17	Test	Testing the monitor button.
23:50:38	Test	Adding it to the watchlist.
23:50:46	Test	Verifying if the asset appears now in the selected watchlist
23:51:45	Test	Testing the filters over the Asset List.
00:02:01	Test	Creating filter presents with random combinations.
00:02:51	Test	Creating a filter preset with a name that already exists
00:04:00	Test	Deleting the created filter preset
00:04:19	Test	Started sorting ASC and DESC for every column.
00:05:57	Bug	Sorting ASC by Last Update led me to some assets which were not updated in a week. Checking them out, it seems they have problems, like, the music sound was deleted fron dissapeared. These assets should not be shown in Beat With It (they should be put on hold till' we figure it out what to do with them)
00:08:15	Question	Sorting by ASC on Total Creations led me to assets which have 0 creations. I think they are not really relevant since their AUTHOR even deleted the video. Maybe we can creat
00:10:51	Note	Funny, in the top 5 BEST assets in the system by Spotify 7 growths, we have like 4 original assets with the same link, GATA ONLY song from Spotify. Not really a bug but maybe system.
00:13:19	Bug	Sorting by Daily Growth on Spotify, I've seen an asset which have the Daily Growth = Total Streams. What I think happened is that the asset somehow had a different link with S Song which is a lot more popular. This maybe be because of searchByISRC function since we take the first result we get from Spotify, but maybe there can be multiple Songs wi SAME SONG BUT RELEASED AT DIFFERENT TIME OR A SLIGHTLY MODIFIED VERSION) and the searchByIsrc returned an DIFFERENT order of Songs with that ISRC.
00:18:12	Bug	Sorting by Average Growth in 24h on Spotify led me to an asset which HAS exactly 462501000% growth. It's insane, i believe this is too much. I see that the Daily Streams of th I believe this is the exact same case as previously stated. Searching on Spotify by the isrc USAT22306575, we get 4 "DIFFERENT SONGS", but they are the exact same song. the problem is searching by Spotify Track ID. Not sure, this needs to be investigated. https://open.spotify.com/album/5RHDfyHMN7CyYLc0KOOmKc?si=jkovyunDSg2W3q4YZH
00:24:09	Note	Session done.

Figure 4.39: Exploratory Testing done using SBTM Method.

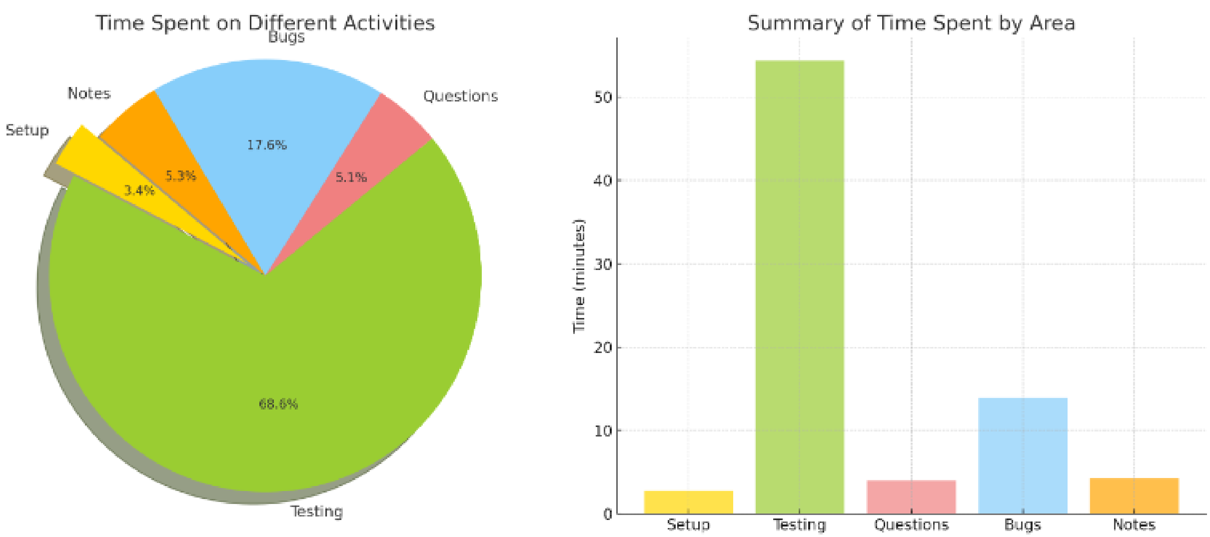


Figure 4.40: SBTM Reports.

4.6 Future Work on Beat With It

For Beat With It, a lot of new additions can be implemented. My top implementations would be:

- Developing new techniques to add assets to the system by enabling users to add author handles and hashtags is another crucial factor to take into account. The system would do the job to add these assets into the platform. This approach not only helps influencers monitor their presence on TikTok but also allows them to keep an eye on potential competitors. For example, this feature can be used by up-and-coming musicians to monitor how their music is performed on TikTok. They can determine their level of popularity and room for expansion by tracking how their songs are received on the platform both before and after their official release. This thorough examination, which incorporates information from TikTok and Spotify, offers insightful information about the progression of their musical careers.
- Scaling the application horizontally and vertically to accommodate increased user demand and ensure smooth performance under heavy loads.
- Monitoring the quantity of ”shazams” that every song has allows us to measure the growths in correlation with the Spotify ones and determine whether they grow similarly after a normalization. This provides insightful information about trends in music popularity.
- Making it more Spotify-friendly, allowing users to perform all desired actions, including music streaming, playlist creation, and searching for tailored playlists, directly within the Beat With It interface.
- We may concentrate on using advanced data analytics to reveal more nuanced information and offer more tailored suggestions. This could entail a number of significant efforts, such as the analysis of user behavior and preferences using machine learning algorithms. Through the utilization of machine learning, Beat With It is able to acquire a more profound comprehension of user interactions and adjust its recommendations appropriately. Furthermore, by putting predictive analytics into practice, Beat With It might be able to foresee user needs and recommend suitable actions in advance. The platform can improve the overall user experience by providing timely and relevant suggestions based on patterns in user behavior and consumption habits.

Chapter 5

Conclusion

Beat With It's growth and possibilities represent an evolution in how we listen to and interact with music. With its potential to transform music discovery, storage, insights, and playing, this app will become a vital resource for music lovers everywhere. Beat With It provides an unmatched experience customized to your tastes, regardless of your level of passion for music or for TikTok.

As we covered in the section on Future Works, there are countless ways to improve and expand Beat With It. It has the ability to precisely and creatively meet the needs of particular users by carving out a niche in fields like music recommendations and insights. Its flexibility also makes it possible for smooth integration with other platforms, such as SoundCloud, YouTube Music, YouTube Shorts, and others, enhancing user experience and reaching a wider audience.

When I think back on it, I'm glad that I have been a consistent Beat With It user. Not only has the app I've had the honor of developing changed the way I listen to music, but it has also made it easier for me to discover, organize, and analyze my tastes in music. Beat With It is proof of the influence that technology and innovation can have on our musical journeys.

In conclusion, Beat With It is more than just an app; it's a doorway to an infinite musical universe that allows users to discover, connect, and find their own "beat" in life.

Bibliography

[BHF$^+$13] Dmitry Bogdanov, Martín Haro, Ferdinand Fuhrmann, Anna Xambó, Emilia Gómez, and Perfecto Herrera. Semantic audio content-based music recommendation and visualization based on user preference examples. *Information Processing & Management*, 49(1):13–33, 2013.

[Cao18] H. Cao. Today's headline algorithm principle, 01 2018.

[Dis24] DishaaPro. Software development life cycle explained, 2024. Accessed: 2024-06-05.

[DL15] Yiwen Ding and Chang Liu. Exploring drawbacks in music recommender systems: the spotify case, 2015.

[Dou30] Douyin Official. 2019 creator ecology report, 2019-08-30.

[FGN22] Sophie Freeman, Martin Gibbs, and Bjørn Nansen. 'don't mess with my algorithm': Exploring the relationship between listeners and automated curation and recommendation on music streaming services. *First Monday*, 27(1), Jan. 2022.

[GS19] K.E. Goldschmitt and N. Seaver. *Shaping the stream: Techniques and troubles of algorithmic recommendation*, pages 63–81. Cambridge University Press, Cambridge, 2019.

[HFC$^+$08] Y. Huang, T. Z. Fu, D.-M. Chiu, J. C. Lui, and C. Huang. Challenges, design and analysis of a large-scale p2p-vod system. *SIGCOMM Comput. Commun. Rev.*, 38(4):375–388, 2008.

[Kat21] Sheela Kathavate. Music recommendation system using content and collaborative filtering methods. *International Journal of Engineering Research & Technology (IJERT)*, 10(02):167–171, 2021.

[KBB$^+$18] Shashank Kaul, Hrittik Bhattacharjee, C. K. Barry, Gaurav Verma, and J. Gowthamy. Web security challenges in node.js applications. *International Journal of Emerging Technologies in Engineering Research (IJETER)*, 6(5):35, May 2018.

[KG22] Aniket Katoch and Deepak Gupta. Music recommendation system. 2022.

[KN10] Gunnar Kreitz and Fredrik Niemelä. Spotify – large scale, low latency, p2p music-on-demand streaming. Technical report, KTH – Royal Institute of Technology, and Spotify, Stockholm, Sweden, 2010.

[Kum24] Suneel Kumar. Message queue in node.js with bullmq and redis. *Medium*, 2024. Accessed: 2024-05-21.

[LA22] Mochammad Fariz Syah Lazuardy and Dyah Anggraini. Modern front end web architectures with react.js and next.js. *International Research Journal of Advanced Engineering and Science*, 7(1):132–141, 2022.

[LTCC14] Ning Lin, Ping-Chia Tsai, Yu-An Chen, and Homer H Chen. Music recommendation based on artist novelty and similarity. In *2014 IEEE 16th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6. IEEE, 2014.

[Per16] Caio Ribeiro Pereira. *Building APIs with Node.js*. Apress, São Vicente - SP, São Paulo, Brazil, 2016. Copyright © 2016 by Caio Ribeiro Pereira.

[Sur22] Akshat Surolia. Recommendation system based on artist and music embeddings. *GLS KALP: Journal of Multidisciplinary Studies*, 2(3):8–15, 2022.

[Só16] Sónar +D. Spotify presents discover weekly and taste profiles, June 16 2016.

[TGB15] Poonam B Thorat, Rajeshwari M Goudar, and Sunita Barve. Survey on collaborative filtering, content-based filtering and hybrid recommendation system. *International Journal of Computer Applications*, 110(4):31–36, 2015.

[Wan03] Avery Li-Chun Wang. An industrial-strength audio search algorithm. Internal Report, 01 2003. Shazam Entertainment, Ltd.

[Yan18] F. Yang. *Data Pool*. 4 2018.

[Zha21] Zhengwei Zhao. Analysis on the "douyin (tiktok) mania" phenomenon based on recommendation algorithms. *E3S Web of Conferences*, 235:03029, 2021.