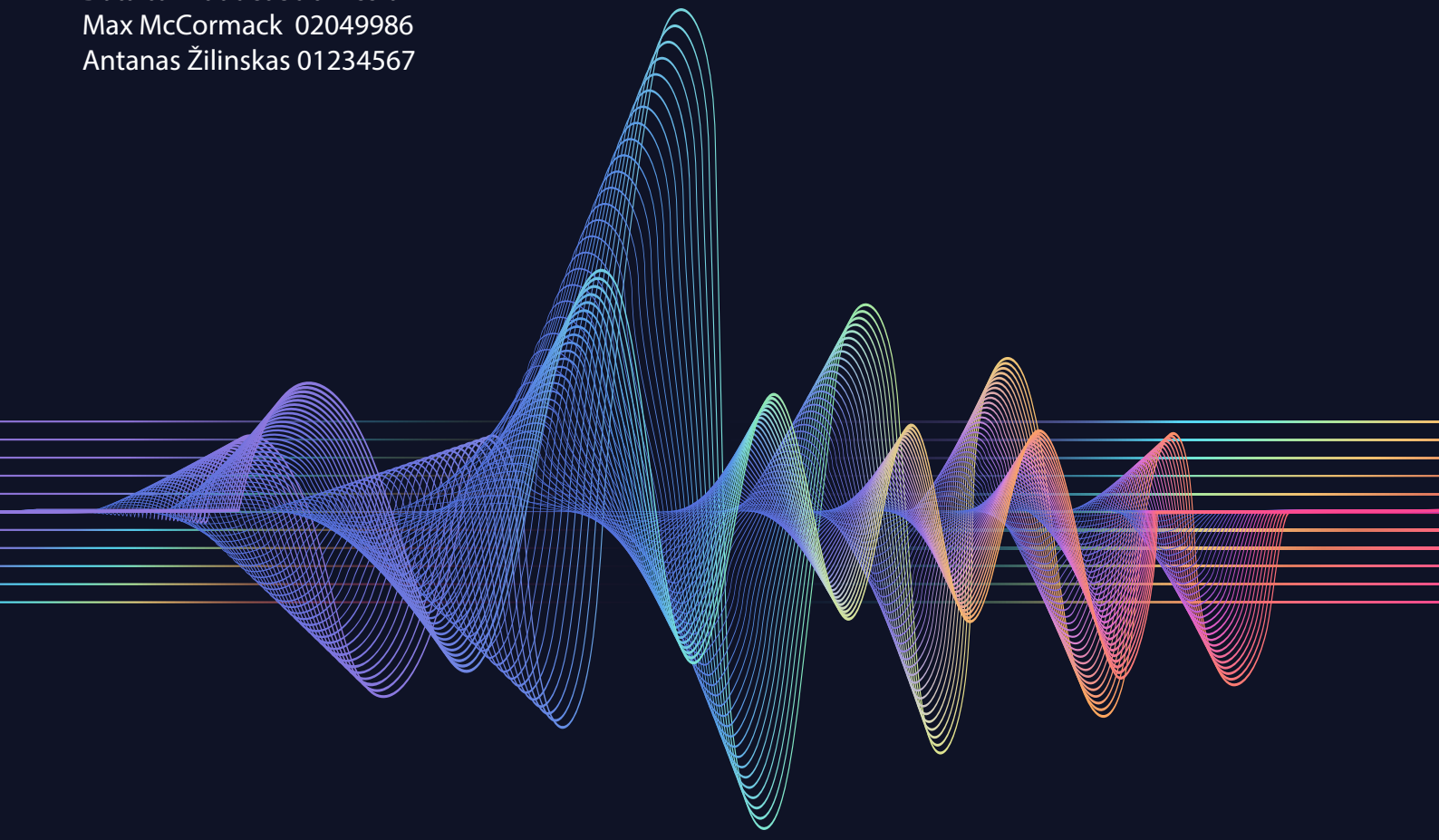
IMPERIAL

Harmonic, Rhythmic, and Melodic Score Feature Based Music Recommendations

Data to Product Submission

Max McCormack 02049986
Antanas Žilinskas 01234567



	Max	Antanas
Ideation	50%	50%
Data Sourcing	50%	50%
Data Curation	50%	50%
Data Product Canvas	50%	50%
Conceptual and System Design	50%	50%
Programming	50%	50%
Testing & Validation	50%	50%
Archiving & Documentation	50%	50%
Report Writing	50%	50%

Disclaimer: GenAI was used in this project for bugfixes and reviewing

Data to Product 2024

Harmonic, Rhythmic, and Melodic Score Feature Based Music Recommendations

Antanas Žilinskas 02054786
Max McCormack 02049986

Dyson School of Design Engineering
Imperial College London

***Abstract*—In this paper, we introduce *Harmonly*, a data product delivering personalised song recommendations by analysing the chordal, harmonic, and rhythmic structures of a user's favourite songs. Unlike existing recommendation systems that rely on genre, artist, or collaborative filtering, we leverage a novel musicological approach, drawing insights from a vast database of over 250,000 song scores sourced from MuseScore. By dissecting and featurising each song's harmonic progressions, rhythmic patterns, and structural nuances, the system identifies deeply rooted musical similarities that transcend surface-level attributes such as genre or artist.**

THE MUSIC RECOMMENDATION MARKET LACKS systems capable of chordal or note-by-note analysis. Existing algorithms, such as those used by Spotify, leverage a proprietary mix of genre-based and crowd-sourced groupings, convolutional neural networks (CNNs), and more recently, lyric similarity techniques. [5] While highly effective at recommending music within a user's preferred niche, these methods often fail to transcend genres and diversify the listening experience. Meanwhile, in the last 30 years, online tabulature-sharing services such as MuseScore and UltimateGuitar have amassed millions of music scores in a format conductive to computational analysis.

IDEATION

Music is inherently interconnected, with artists drawing inspiration from predecessors across diverse styles, and new genres being born of unlikely roots. This interconnectedness often leads to surprising links. For instance, heavy metal and rock frequently draw inspiration from classical music, resulting in the notable phenomenon of many metal fans appreciating baroque classical compositions [3]. Studies indicate heavy metal and classical music enthusiasts share more similarities than may be initially apparent. Beyond sharing a love of the theatrical (Wagner, Paganini, Kiss, Meatloaf), virtuoso performances (Liszt, Van Halen), and complex, metaphoric music (Bach, Mastodon), they may also share comparable personal-

ity traits [4].

These observations underpin our hypothesis that individuals are drawn to specific facets of music rather than particular genres. We posit that certain musical features, such as complexity, melody, timbre, tempo, and rhythm, can be empirically quantified and compared to reveal these connections. However, subjective elements such as cultural association, emotional evocation, or historical context, while influential, fall outside the scope of this project due to difficulties in objective quantification.

PRODUCT DESCRIPTION AND IMPACT

Our product aims to offer a novel perspective on music recommendations based on user preferences. To define our audience and ensure the final product caters to a wide user base, a series of distinct personas are considered:

1) **The casual music listener** who seeks further engagement with songs they might enjoy.
2) **The music enthusiast** who looks to expand their horizons and explore new styles.
3) **The professional curator** who is searching for music that compliments the tone, mood, or other vectors of harmony, of their existing tracklist.

To measure success and set clear objectives for the product's impact, we established the following Key Performance Indicators (KPIs):

1) The product must provide individualised music suggestions aligned with user preferences as defined by identifiable musical patterns.
2) The project must deliver tangible value to users beyond what current recommendation algorithms offer
3) It should foster increased user engagement with music and songs.

To achieve these objectives, the product's functionality was outlined as follows: Users provide a set of song preferences, and *Harmonly* analyses these preferences to parse their "harmonic DNA". By mapping musical features to songs in the database, *Harmonly* uncovers hidden connections, and suggests tracks across genres that share a similar musical essence to the user's input.

Given the abstract nature of musical featurisations used in our similarity search, a user-friendly explanation of how we came to our conclusions on recommendations is essential for user engagement. To address this, an element of of the product must visually explain the musical attributes identified in the user's preferences and illustrates how these features align with the recommended songs.

We aim for this product to empower users to explore music outside of their typical comfort zones, across genres and styles while retaining their personal musical preferences. Early feedback from musical professionals, including the conductor of Fulham Symphony Orchestra and soundtrack curators suggests potential impact in musical programme curation, where stylistic variety combined with thematic coherence is desirable.

THE DATASET

Our song database was derived from the large-scale PDMX dataset [1], which contains over 250,000 public domain MusicXML scores collected from the score-sharing platform MuseScore. These scores are formatted as JSON files and have well-organised metadata information. Although intended for music generative AI training, this dataset proved suitable for our purposes due to its scale, quality and readability.

To prepare the dataset for analysis, we pruned the data for unusable or low-quality files - such as those with missing titles, overly short scores, or duplicates. The cleaned dataset allowed us to extract features from each score and combine them into a single feature vector for each song, enabling efficient vector similarity searches.

The following list details the data points generated for each song from the original MusicXML JSON files:

- **Pitch class histogram**: A distribution of note occurrences across the 12 pitch classes (C, C#, D, etc.), representing tonal content.
- **Interval histogram**: A distribution of melodic intervals in the music, indicating the frequency of "gaps" in the song.
- **Melodic contour**: The general shape of the melody, simplified to the percentage of time the song is rising, falling, or remaining static in pitch.
- **Chord progressions**: The sequence of chords in the music, providing harmonic structure.
- **Key signature**: The musical key in which the piece is written, indicating the tonal center and accidentals, and roughly informing the general mood of the song.
- **Mode**: Whether the piece is in a major or minor

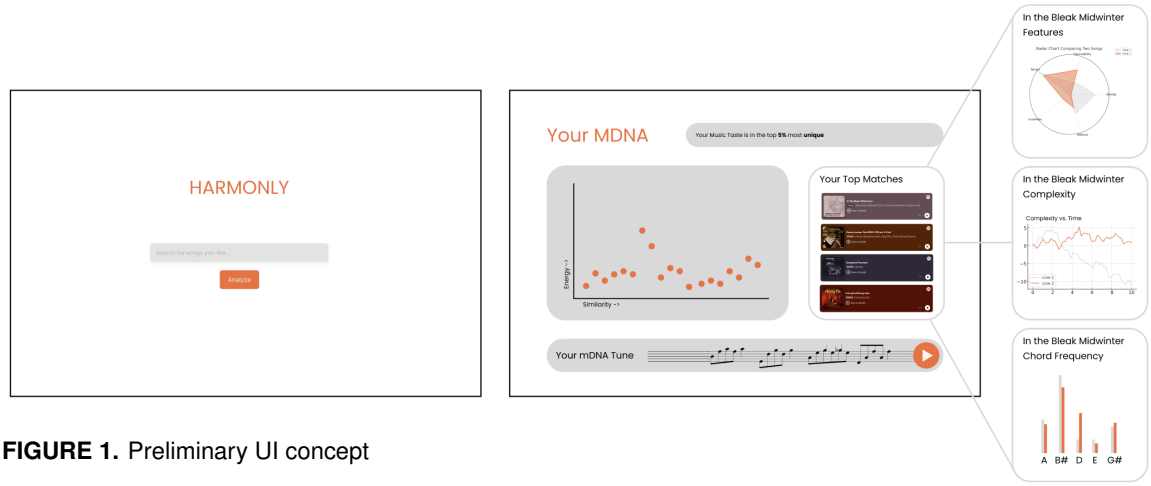


FIGURE 1. Preliminary UI concept

key, contributing to its emotional quality.

- **Note duration histogram**: A distribution of note durations in the music, showing rhythmic variety.
- **Average duration**: The average length of notes in the piece, reflecting tempo and rhythm.
- **Tempo**: The speed of the music, measured in beats per minute (BPM).
- **Measure**: The basic unit of time in a piece.
- **Time signatures**: The notational convention indicating the number of beats per measure and the beat unit.
- **Feature vector**: A numerical representation of the music's features for efficient computational analysis. This is a 128 element list of floats.
- **Title embedding**: A vectorized representation of the song title used for matching database entries to user input.

To ensure interpretability, we also developed a some higher level, more understandable metrics based on these features to communicate more abstract concepts of music.

- **Complexity Score**: A score from 0-1 evaluating a song's musical intricacy, pitch distribution, interval variability, melodic contour, chord progressions, and note durations. It calculates entropy (to measure unpredictability), variance, and unique chord ratios, combining them into a weighted score that reflects the overall complexity, balancing technical depth with practical interpretability.
- **Melodic Contour Score**: A representation of how much of the song is ascending or descending, on a scale from 0-1, and weighted logarithmically to more accurately represent how a human hears melodic contours.

CONCEPTUAL DESIGN

Our concept began with planning our user experience (UX) (Figure 2). Our aim was to ensure intuitive navigation while delivering a powerful, configurable tool that provides value. The input, as with any recommendation system, is a list of song preferences. Given the discretionary nature of song titles, we decided a search plus drop-down menu was the best user interface for this step. Selected titles are then matched to songs in the database, and their features are compared to others for similarities. Initial tests indicated any operations considering the entire dataset took a while, necessitating a loading indicator from the selection page to the recommendation page.

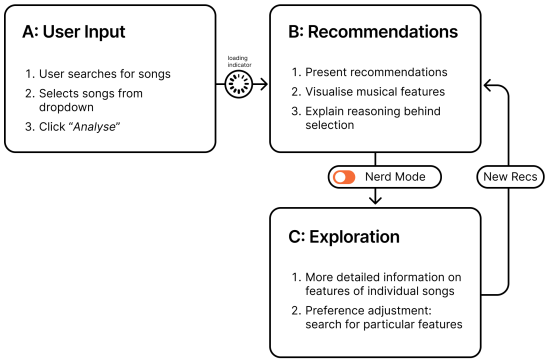


FIGURE 2. Preliminary UX plan

Colours for our User Interface (UI) were chosen to convey a clean, modern experience, and standardised interface elements such as a side menu and home page were used for familiarity and ease of navigation. With this UX and UI prototype, we rapidly produced a *Wizard-of-Oz* mockup MVP to test our concept with potential users. This mockup was presented to users to observe their interactions and iterate on our prototype.

Feedback from this process prompted us to relocate the ”Analyse” button to below the ”selected songs” box, as users repeatedly tried to click ”Analyse” once they had searched for a song, but not selected it.

We initially tested a playback feature that played scores with a MIDI synthesiser, but users noted it sounded somewhat mechanical and lacked depth, so Spotify links for each song were added instead.

A visualisation of recommended songs proved difficult to make intuitive, so we tested a number of charts with a variety of users. Less knowledgeable users preferred the simpler views, while enthusiasts enjoyed a variety of charts and graphs. Thus, a ”nerd mode” was added, which, when selected, presents the user with more technical, in-depth information about the recommendations. Some advanced users highlighted that they would like to see fine-tuning of preferences for tracklist curation, so we also developed an experimental feature that allows the user to use sliders to adjust the type of songs the algorithm returns. Due to the technical understanding required to utilise this feature, it was difficult to make intuitive, and only caters to a small user base, but remains an intriguing addition to our value proposition.

SYSTEM DESIGN

Due to the substantial size of the database (50 GB), it was necessary to store data on a centralised repository to enable the product to be portable and ensure accessibility across various devices. The data is hosted on Supabase as a PostGreSQL table, with access facilitated through a series of RESTful APIs. This ensures the local installation of our product is lightweight and doesn't rely on local processing power.

The song title search functionality was initially tackled by generating a *word2vec* text embedding using the Python `sentence_transformer` library for each user search query, and then doing a cosine similarity search on a database of title embeddings. This was functional, but slow. A fuzzy title search was used instead, using the PostgreSQL `pg_trgm` extension to generate similarity scores from database entries. This is executed via an edge function API on Supabase.

Once the user has identified their song preferences, they press ”Analyse”, triggering a loading icon. A runtime API request posts the song titles to Supabase, where the feature vectors of the preference songs are compared to those in the overall database. The matching function calculates a similarity for each song in the database score based on the Euclidiean distance between the query and database vectors, and the top 5 recommendations are returned.

Recommendations are communicated to the user on a ”target” style chart, showing how close a match to their preferences each one is, and displayed using dynamically loaded html elements. When one of these songs, or elements, is selected, API requests are sent to Supabase and Spotify, returning the feature vector for that song, so that a chart can be displayed, and the link to the song on Spotify.

DEVELOPMENT PROCESS

We relied heavily on user feedback during our development process. We hosted our initial functional prototype on *Render* to allow ease of testing with a variety of users. We continuously iterated our frontend based on observed interactions, and accordingly adjusting our backend only where absolutely necessary. We initially presented recommendations to users on a chart, where X represented complexity and Y represented energy. Our more casual audience found these terms confusing, and struggled to interpret the graph, while knowledgeable users enjoyed the presentation, but wanted more detail. Renaming ”energy” to ”danceability” did not improve interpretability, so the chart was simplified to a target conveying similarity. For advanced users, we decided to include a ”Nerd mode” switch to provide a deeper level of detail.

Additionally, presenting the user with a series of graphs on the recommendation page was deemed confusing and an ”information overload”, so we adjusted our UI to only give detail on a song when that song is selected. User feedback indicated both basic and advanced users liked the ”spider chart” highlighting higher level, more intuitive musical features for a particular song, so we implemented it to show in the song view pane, under the *Spotify* link.

Both team members contributed to the full stack, alternating responsibilities as needed. While dividing tasks by UI and server-side operations might have improved development efficiency, it would have limited our creative vision and flexibility. We used GitHub for version control, leveraging its project management features to track issues and features, organized into “To Do,” “In Progress,” and “Done” categories. To minimise merging conflicts and facilitate bug tracking, we created a separate branch for each feature.

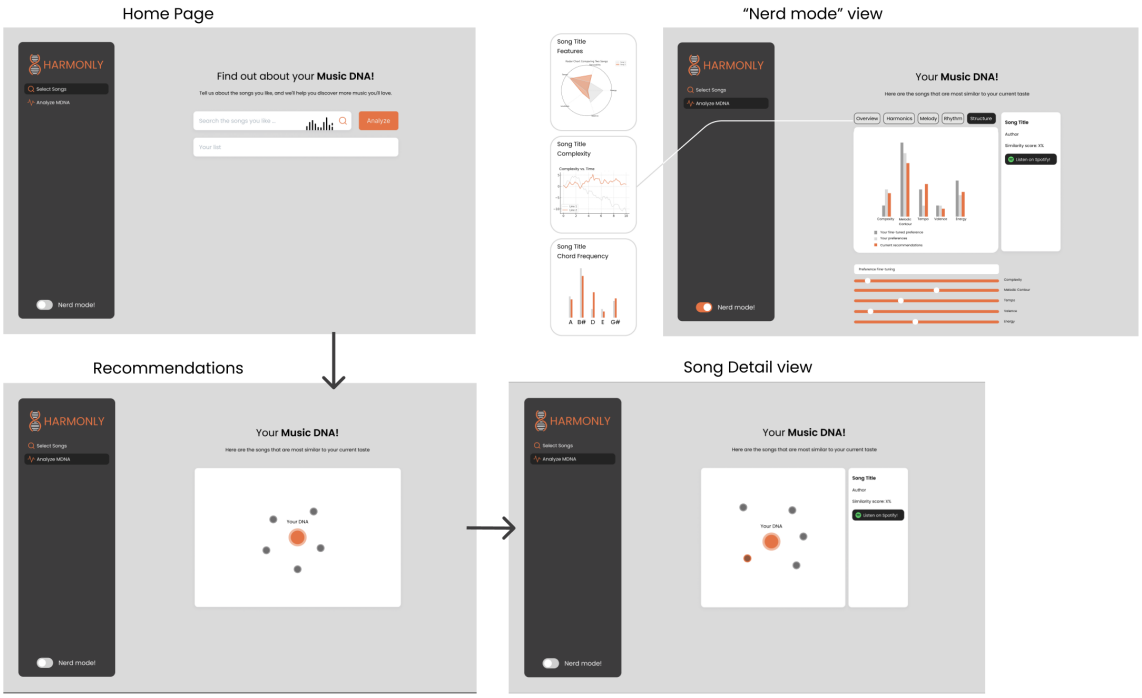


FIGURE 3. Preliminary UI concept

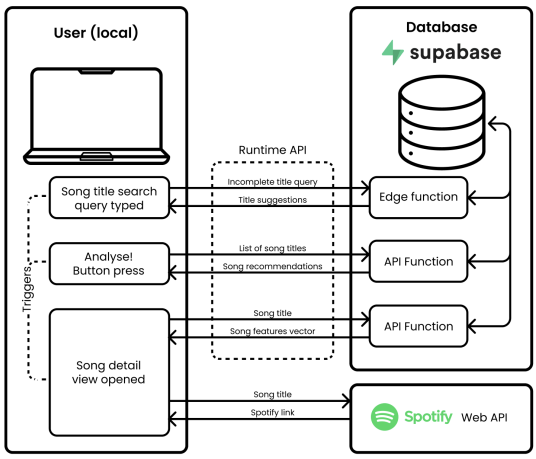


FIGURE 4. Interactivity between server and webapp, highlighting API triggers

Owing to our rapid development and redevelopment strategy, keeping up with changes sometimes proved difficult, resulting in some less than efficient pipelines on the backend. Some API calls have to be invoked multiple times to generate a single chart, as we didn't have time to completely redesign the function on the server side. Furthermore, some data that could be stored locally is re-retrieved on demand, further pushing up the number of API calls required to keep the app running. (Figure 4)

FINAL PRODUCT

The product is designed to provide value to all users, from casual listeners to music professionals. Graduated levels of technical complexity can be accessed through the advanced tools pane, catering to varying user needs.

For advanced users, fine-tune controls are available, enabling adjustments to preference parameters via sliders to generate more tailored recommendations. This feature is particularly beneficial for professional users who require greater specificity in their results, and could have applications in music programme curation. The final iteration can be run with R as long as we maintain our Supabase project, as is available at https://github.com/AntanasZilinskas/fd2p

PROGRAMMING

The frontend of the data product is an R shiny app. CSS is used for styling. Basic UI elements are defined in `ui.R`, while reactive elements are mostly defined in `server.R`, with some in `global.R`. Animations and some interactive elements, such as the loading wheel, are implemented with embedded javascript. The backend is mostly handled within the R shiny app, but most tasks involving the database are run on the server

side, with data being retrieved via a series of APIs (Figure 4). Data is stored in a PostGreSQL table, with most of the API functions being defined with SQL. Typescript was required to set up edge functions. Our repository also includes some python scripts, which were mostly used in prototyping and development.

REFLECTION

Considering our aims for the project, we believe the data product to be a success overall. The recommendation feature works well to introduce new and interesting songs to users, and feedback suggests that these recommendations do align well with testers preferences. However, our analytics, as presented to the user outside of nerd mode, do little to foster increased engagement with music and songs beyond providing the aforementioned recommendations. For nerd mode users, our charts present intriguing information, and the preference fine-tuning sliders may allow effective searching for songs that meld well together, but the audience is small. Given more time, we would have liked to find a way to present these ”nerd mode” features to a wider audience in an intuitive manner. Overall, we found that our musicological approach to song recommendations was very interesting to some of our more musically knowledgeable users, while less advanced testers weren't entirely sure what made us different to standard music recommendation algorithms. The main complaint was that our database didn't have all the songs the users wanted to choose from. This is due to our entire database being public domain songs, but our system could easily scale up given a commercial dataset.

We had a number of features that were planned, but not implemented due to time constraints:

1) **Feature Anchoring:** Highlight standout features that the user has a strong preference for, eg very upbeat or entropic music, or highly complex songs.
2) **Chord Characterisation:** A string of chords that defines the user's MDNA, being the most common sequence in their preferences.
3) **Entropy Mapping:** Charts showing the entropy over time
4) **Preference Tuning for Less Advanced Users:** An intuitive way to allow basic users to adjust the kind of recommendations they want.

Additionally, owing to the rapid development of our project, artifacts from previous iterations introduced inefficiencies that weren't patched due to time constraints, most notably calling APIs for data that has already been retrieved.

CONCLUSION

Overall we are happy with the outcome of the project. It represents a novel approach to music recommendation, advancing beyond current algorithms by incorporating a musicological note-by-note perspective. Recommendations are presented in an intuitive manner, designed to inspire intrigue and engagement. The basic user interface caters to users of all backgrounds, while additional levels of complexity provides value to more experienced users.

ACKNOWLEDGMENTS

We would like to thank all of the users, friends and family that allowed us to test our application and prototypes on them, and Pierre Pinson for ongoing support and consulting.

REFERENCES

1. Long, P., Novack, Z., Berg-Kirkpatrick, T., and McAuley, J. (2024) ‘PDMX: A Large-Scale Public Domain MusicXML Dataset for Symbolic Music Processing’, arXiv. Available at: https://arxiv.org/abs/2409.10831 (Accessed: 23 November 2024)
2. Cho, Y.-H., Lim, H., Kim, D.-W. and Lee, I.-K. (2016) ‘Music Emotion Recognition using Chord Progressions’, in 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2016), Budapest, Hungary, 9–12 October 2016. IEEE, pp. 002588–002593. doi:10.1109/SMC.2016.7844291.
3. Bauerle, E and Fletcher, K (2015) ’The Metal age: The use of Classics in Heavy Metal Music’, Society for Classical Studies. Available at: https://classicalstudies.org/scs-blog/parian/metal-age-use-classics-heavy-metal-music (Accessed 1 Dec 2024)
4. North, A (2010) ’Individual Differences in Music Taste’ The American Journal of Psychology Available at: https://www.jstor.org/stable/10.5406/amerjpsyc.123.2.0199 (Accessed 1 Dec 2024)
5. Wong, J. (2024) ‘Algorithmic Symphonies: How Spotify Strikes the Right Chord’, USC Viterbi School of Engineering, 21 January. Available at: https://illumin.usc.edu/algorithmic-symphonies-how-spotify-strikes-the-right-chord/ (Accessed: 1 December 2024).