# Music Selector Dashboard (AlgoDJ)

Renzhi Chan, Michael Lutter, Stuart Mals, Ryan Porter, Wael Shabana

**Abstract**

AlgoDJ is a song recommendation application that bases playlist suggestions on a variety of input factors including the user's searched song, the user's past search history, as well as other demographic factors such as the user's age, gender, and search language. Current systems recommend based on play history or trending songs, but our approach is more holistic, including the user's previous music interests and demographic information as factors influencing the playlist recommendation. Additionally, AlgoDJ innovates by recommending songs based on a cosine similarity metric determined using other factors such as tempo and danceability. Last, AlgoDJ sets itself apart from other music recommendation systems by visually displaying the results in a fashion that is easy for the user to interpret and further investigate his or her own music interests further. As illustrated in our app, AlgoDJ makes use of a table, charts, a node network, and a wordcloud to not only recommend a playlist, but to also show the user why and how those recommendations were determined. Success for AlgoDJ was measured via both qualitative (user studies of satisfaction) and quantitative (relevance ratings) methods, as discussed in the Results and Evaluation section below. The project took approximately eight weeks to complete, including testing, and the work was spread out evenly among all team members in three parts: Data Scraping, Modeling, and Visualization (Figure 1). The final result of the project is a python-based recommendation system, displayed in graphics using Dash and plotly in python, as discussed in the Survey section below.

**Keywords**

Visualization, Recommender System, Plotly, PySpark, Music

## 1. Introduction

Working with a dataset of more than 1.2 million records of songs from which to generate playlist recommendations, AlgoDJ strives to improve the field of music recommendation not only by providing better recommendations to the user based on factors like their past search history, age, and search language, but also to help users visualize the playlist recommendations with easy to understand graphics to make the app visually appealing to consumers. Making use of the cosine similarity ranking system, AlgoDJ goes beyond recommending generically popular hits in a genre or mood and caters to a specific user's attributes. Bar charts, node networks, tables, and a wordcloud were all chosen for their visual appeal to users and their relative ease of interpretation. AlgoDJ thus allows users to learn more about their own music interests, how their interests compare to those of similar demographics to the user, and even to explore new songs and artists that they may not have come across in a more generic "popular hits" recommendation system that is based on the results of the cosine similarity ranking system.

## 2. Problem Definition

With a dataset containing 24 features (such as artist, tempo, release date) of more than 1.2 million songs from Spotify's platform, the first challenge for AlgoDJ was scraping, parsing, cleaning, and storing data for later use in the modeling and visualization phases. Furthermore, it is critical with an app like AlgoDJ to have fast processing speeds so that the user does not get bored and leave the app before AlgoDJ has generated the recommendation. With regards to modeling, the largest challenge was finding a way to effectively recommend songs to users in a way that presented a clear and distinct advantage over existing recommendation systems. Though other techniques may focus more on a user's similarity to other users, our method focuses on the user's catalogued past interests, his or her age and gender, and the features of the song or artist, which we think sets us apart from the competition in a unique way. The final challenge came on the visualization side in the form of displaying results in an easy to follow way that allowed the

user to understand the logic behind the recommendation in an eye-catching fashion. By applying the concepts learned in this class and through additional research, our team was able to address these problems to achieve our desired end product, as will be discussed further below.

## 3. Survey

The following survey section dives into what existing literature out there has to say about the three components of our project: Backend data scraping, Modeling, and Front End Visualization. The research below informed our decisions for improving upon existing methods to create AlgoDJ.

### 3.1 Backend/Data Scraping

Web scraping offers a solution to the problem of fragmented datasets by collecting data from where search engines cannot[11]. Web scraping collects data too niche for an existing web API to cover. Although many webpages are designed for human interaction, scraping accounts for most of the network activity[8]. Web scraping is a fitting technique for this project, since many Python packages have been created[8]. REST architecture gained popularity by its enforcement of API queries to use HTTP methods explicitly and constraining HTTP operations to be stateless[15]. Despite the benefits, REST web APIs often return data objects that are over-fetched with redundant information[6]. GraphQL addresses this issue by building bespoke queries, leading to improved retrieval performance and shorter development time[6]. In 2022, 89% of web services support REST, while only 28% do so for GraphQL[1].

Hadoop is an open-source framework for handling Big Data processing[5]. Hadoop Distributed File System provides a framework to store data across many computers. A "map" procedure filters and parses data, then a "reduce" process summarizes the output[5]. Hadoop can process music data, but MapReduce requires low level code development, slowing the development process. Apache Pig accelerates development via a scripting language (Pig Latin) to perform data analysis tasks[18]. However, Pig lacks other functionality needed. Apache Spark is also a framework for big data, but uses Resilient Distributed Dataset (RDD) for efficient data sharing. It also contains libraries for SQL, machine learning, and API's for several different programming languages[16]. Spark runs in-memory which allows it to run faster than Hadoop[13]. With the faster processing speeds and python API, Spark will be used for the backend data processing.

### 3.2 Modeling

**Audio-Based Genre Classification** discusses songs recommendation systems that match the music genre selected by a user, based on multiple factors to generate a list from different genres. This is applicable because AlgoDJ will recommend songs based on an audio clip, after identifying the song of the clip. In this program, the data could only include music or artists belonging to a single genre. Additionally, the data could only be used in either the test set or the training set but not in both. Our data will include artists that belong to multiple genres of music, since artists often have songs grouped under different genres. [12]

**Smarter than Genius?** compares Genius, a commercial music recommender system, to two other similar systems. One system recommends songs based on artist similarity and the other two base recommendations on acoustic content. This gives insight into the benefits of using commercial programs and suggests which features we might want to include in our application. These systems used more popular songs in their database and did not include any examples of using lesser-known artists. [4]

**Music recommender systems** outlines how to use knowledge-graph-based and feature-based explanations to improve a music recommender system. Best results are seen when a system uses multiple attributes to choose the most accurate recommendation without over-classifying. We will have to avoid over-classifying the model or using too many features in the algorithm because this would limit the number of recommendations. [3]

### 3.3 Front End Visualization

Users are more likely to use a given recommendation system if they can visualize the output and understand how the system generated the given recommendation[14]. Among many interactive visualization tools, Plotly is chosen due to its scalability and ease of use.

**Plotly-Dash** is an open-source library that interfaces with R, Python and MATLAB[7]. This library creates fully interactive, web-based applications, allowing users to focus on analytical aspects of development[2].

Plotly-Dash uses Flask for the backend and React for handling front end components, all rendered in a single-page code[10].

**Tableau** is a well-known visualization tool, and its drag-and-drop functionality allow for short learning curves prior to use. However, it often requires data compression in the form of extracts to process the Big Data sets quickly[19]. Tableau will not be the best tool due to its latency with large data sets.

**R Shiny** is R-based industry-leading tool for visualization. R's tidyverse ecosystem is successful because it is easy to understand and can "limit cognitive load without removing the benefits of performing analysis via code"[17]. Despite its advantages, we will work with Plotly in Python for better compatibility and scalability.

## 4. Proposed Method

### 4.1 Intuition

Personalized playlist curation is the result of AlgoDJ's cosine similarity score, which looks at the factor's of a song provided in Spotify's API, including tempo, danceability, and other factors. Furthermore, with the user's search history and demographic factors, AlgoDJ comes up with a recommendation that is tailored to the user's specific interests, allowing the user to create playlists that are most suitable for an event or taste in music.

By factoring in the language of the user's search, AlgoDJ allows users to compare personal playlists with the top 10 hits in their spoken language, giving them insight into how similar or different their music tastes are from their neighbors and friends. This feature also allows users to discover fresh, local music while travelling, further broadening their music horizon.

Most music recommendation applications fall short of providing highly personalized recommendations for users while also providing a visual interface to explore new music. AlgoDJ attempts to address these deficiencies by developing a visual-centric, bespoke music curation and exploration experience. Our team believes that some of the greatest joys in appreciating music lies with discovery. It is this intuition about music recommendations and our understanding of the importance of visuals that informs our design decisions for AlgoDJ. Some of the key visualizations and their intuition are described below.

The user's recommendations will be visualized with the Song and Artist Network Graph (SANG). Users of AlgoDJ can visualize their song recommendations in a network graph and see how the songs in their playlist are connected to other songs in terms of sound or artists. Network graphs allow the users not only to see which songs are most similar to their song of interest (as illustrated by the distance between nodes), but also to further explore their own music interests by exploring those nodes which are two- or three-degrees separated from their original song of interest.

In addition to the Song and Artist Network Graph, AlgoDJ makes use of other visualizations, including a tabular display of the recommended playlist, a word cloud of the popular artists included in the playlist, and various charts illustrating the frequency distribution of songs by year or release in order to help the user filter down to particular time periods of interest to them.

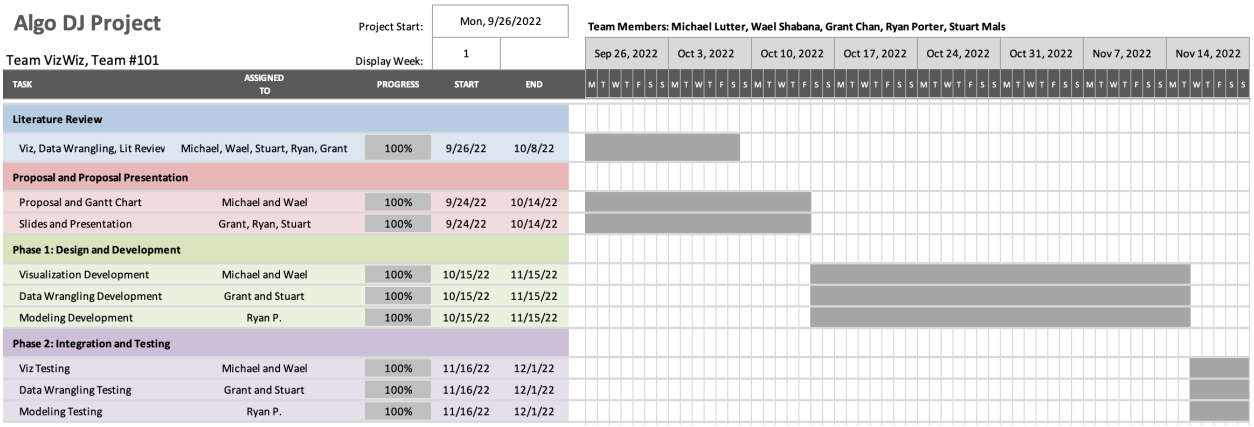
### 4.2 Algorithms

AlgoDJ's uses two algorithms: one at the app level and another at the model level. At the app level, AlgoDJ's UI sends three instances of User, Song and Playlist classes instantiated based on a user's input. All instances are sent to the model to be used in its search algorithm, using the user's provided information to generate a satisfactory playlist. The model, thereafter, instantiates three instances of a Playlist and as many instances of Song and Artist, required for the playlist, all returned to the UI for display. The algorithm at the model level is a cosine similarity method where the similarity between two vectors is measured. A playlist is generated from the top specified number of songs with the closet output to 1. Plotly-Dash then takes the output from the model and constructs different visualizations of the playlist to display to the user. This gives the user greater insight into what kind of songs make up the playlist.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

### 4.3 User Interface

AlgoDJ's user interface (UI) offers its users the flexibility of using the app without signing up for an account, though they do have the option of entering age and gender for more tailored recommendations.



Algo DJ Project — Team VizWiz, Team #101 — Project Start: Mon, 9/26/2022 — Display Week: 1 — Team Members: Michael Lutter, Wael Shabana, Grant Chan, Ryan Porter, Stuart Mals

| TASK | ASSIGNED TO | PROGRESS | START | END |
|---|---|---|---|---|
| **Literature Review** | | | | |
| Viz, Data Wrangling, Lit Reviev | Michael, Wael, Stuart, Ryan, Grant | 100% | 9/26/22 | 10/8/22 |
| **Proposal and Proposal Presentation** | | | | |
| Proposal and Gantt Chart | Michael and Wael | 100% | 9/24/22 | 10/14/22 |
| Slides and Presentation | Grant, Ryan, Stuart | 100% | 9/24/22 | 10/14/22 |
| **Phase 1: Design and Development** | | | | |
| Visualization Development | Michael and Wael | 100% | 10/15/22 | 11/15/22 |
| Data Wrangling Development | Grant and Stuart | 100% | 10/15/22 | 11/15/22 |
| Modeling Development | Ryan P. | 100% | 10/15/22 | 11/15/22 |
| **Phase 2: Integration and Testing** | | | | |
| Viz Testing | Michael and Wael | 100% | 11/16/22 | 12/1/22 |
| Data Wrangling Testing | Grant and Stuart | 100% | 11/16/22 | 12/1/22 |
| Modeling Testing | Ryan P. | 100% | 11/16/22 | 12/1/22 |



**Figure 1.** VizWiz (Team#101) Gantt Chart - all members contributed equally

UI allows users to choose the size of their playlists, from 2 to 20 songs in length. To use the app, users must enter a title of a song of their choosing. AlgoDJ's UI will display the recommended playlist in a list box, which will have a scroll bar to enable users to go through a list of songs, each of which listed with its title, artist name, album, and year of release. Users can hover over a song in a playlist to get more information than what is displayed in the list box via a tool-tip.

Dash-Plotly (D-P) is used in developing the UI. D-P is used in the development of the UI, using HTML and Dash Core Components, arranged in a layout[7]. Plotly is used for rendering charts and React is used in the background by Dash for handling all components, rendered in all Python-based development. Dash Bootstrap Components handles components in the layout and the screen[2]. Bootstrap is a set of tools that abstract away details pertaining to handling elements and components on the layout of the app. Bootstrap achieves this through thematic effects, adoption of flexbox system, responsiveness, encoded colors, and the use of prebuilt systems. UI uses a combination of flexbox-based layout, providing patterns, and low-level, screen-based attributes to achieve desired aesthetics and communicate vibrant content. Bootstraps also helps build a responsive app, displaying the layout dynamically with changing screen size. [7].

## 5. Experiments/Evaluation

### 5.1 Testing

Evaluation of AlgoDJ is categorized into two segments, qualitative and quantitative performance. Quantitative performance was assessed by means of accuracy (correctly displaying the top x number of songs based on their cosine similarity rating score), whereas qualitative performance was determined by a combination of focus group survey results and A/B testing. For the focus group which consisted of 10 peers, we showed them AlgoDJ's layout and song recommendation platform, and asked them what they did and did not like about the app. For the A/B testing, we showed another set of 10 peers two different versions of the dashboard - one with the visualizations, and one without visualizations (tabular results only) - and recorded their preferences. The summary results of the accuracy tests are discussed below in the Recommendation Accuracy subsection, whereas the evaluation of the qualitative performance ratings are discussed in the Usability Evaluation section.

### 5.2 Results and Evaluation

#### 5.2.1 Scalability Evaluation

AlgoDJ's design is scalable as it uses loose coupling principles, providing no dependencies between its modules, data, model, and visualization[9]. The scalability of the application was evaluated during several parts of the workflow. First, the data was gathered from web API's based on user input. Initial testing showed that some API's have limitations on the number of API calls over a specific time period. In order to find the optimal amount of data, several different methods had to be tested on how much data was required to build an accurate recommendation model by comparing the output of each. Once all of the raw data was acquired, it then needed to be processed using feature engineering, filtering, aggregation, etc. Baseline tests were performed to see processing time using pandas. The same tests were then performed using Apache Spark to test speed increases using clus-

ters and parallel computing. Last, once the prototype of the application became available, additional testing was used to see how it scaled with the number of users and if there were any potential bottlenecks. Upon completion of project, all final scalability tests were successful and no potential bottlenecks were identified.

#### 5.2.2 Recommendation Accuracy

When considering how to assess the accuracy of AlgoDJ, our team considered a few different methods of evaluation. First, one method of evaluation would be to conduct user studies in which users would try each of the different functions of the program and answer a few questions assessing how it performed. All but one question could be either yes or no and the other questions would rate overall performance on a scale of 1 to 10. If most answers are in yes or no format, we would be able to use precision and recall by computing the true and false positives/negatives. The final question would ask how well the program worked on a scale, making it is easy to determine how useful the users found our program. Another way of finding the accuracy of the model is using the cosine model, which takes two songs and computes the dot product between the two. Therefore, we can determine how similar both of the songs are to each other. Due to the subjectivity of individual respondents interpretation of song recommendations, we decided to use the cosine similarity metric as a our preferred method of identifying the accuracy of AlgoDJ's recommendations. However, we recognize that relevance, as determined by individual users, is a very important metric that would be crucial to commercial viability of an application such as AlgoDJ. In future experimentation, our team recommends further research into how to assess the relevance of songs to the users as a means of improving AlgoDJ's song recommendation.

#### 5.2.3 Usability Evaluation

In order to assess the usability of our UI, we used both a focus group and A/B testing with classmates. The focus group consisted of a set of 10 peers who tested AlgoDJ and provided qualitative feedback on the visual appearance and the relevance of our recommended playlist. A second set of classmates participated in A/B testing, whereby we showed the individual our recommendation system side-by-side with a generic recommendation system that does not contain many of the factors that make AlgoDJ unique, such as the node network visualization and the use of cosine similarity to create accuracy scores for the relevance of songs in our recommended playlist. The results of both the focus group and our A/B testing allowed us to refine our AlgoDJ recommendations to best cater to individual's interests. The most notable takeaways from these evaluations were as follows:

**A/B Testing Results:** While the majority of respondents to our survey (sample size of 10 peers) agreed that each individual visualization added to their understanding of the playlist results, some visualizations clearly performed better than others. The artist graph network was the best performing visualization, with all but one of the survey respondents saying that they preferred this visualization be included in the dashboard. This indicates that users value the ability to see the interconnectedness of the various artists in the generated playlist. Similarly, users liked the methodology used for generating the playlist (the cosine similarity algorithm) and preferred this method to a more general "top trending hits" playlist recommendation system. The worst performing visualization was the timeline visualization, with only half of respondents saying they preferred this visualization to be included in the dashboard.

**Focus Group Results:** The focus group results provided our team with a variety of consumer perspectives that should be considered for further improvements to AlgoDJ. A few suggestions stood out in particular. First, someone expressed not liking the dark mode layout as the only option and suggested that there be the option to toggle between light and dark mode. Second, one respondent suggested adding an exportable table of the playlist recommendation, for ease of the user's use or later reference, or even adding a Spotify hyperlink to each song on the playlist table. Last, another respondent suggested that it might be helpful if AlgoDJ gave the user the ability to select which features would be factored into the cosine similarity ranking system, thus giving the user the power to choose features that are most important to them when determining the similarity of songs.

## 6. Conclusion and Discussion

AlgoDJ innovates music recommendation by improving upon the curation and the visualization of playlists generated from music recommender systems. Unlike

more traditional recommender systems, AlgoDJ generates song suggestions based on a variety of factors including the user's search history, his or her age and gender, and implicit factors of the song such as tempo and danceability when creating the cosine similarity ranking scores. Additionally, the visualizations and display of the playlist help the user understand the logic behind why each song was selected for the playlist, and allows the user to explore other songs and artists visually that may be hard to gather from a tabular recommendation output alone, as indicated by the results of our focus group responses. Overall, the project was a success for Team AlgoDJ, and future expanded research is encouraged to improve AlgoDJ for further implementation, as discussed below.

## 7. Suggestions for Further Research

Given the time constraints of the project in a single semester, there are a few components of AlgoDJ for which the team encourages future research. Those areas of recommended future study are as follows:

### 7.1 Additional Consumer Studies

As mentioned above, there was a limited timeframe and pool of potential candidates for gathering survey responses from potential users on the layout and recommendation accuracy and visualizations of AlgoDJ. In order to further improve AlgoDJ's mass appeal, more extensive consumer research surveys should be conducted to improve the product further.

### 7.2 Geographic Feature Engineering

While AlgoDJ does incorporate the user's search language, one additional improvement could be to make use of the user's IP address in order to filter results based on both the user's language of searched song and their current location. For example, if a user is located in a Spanish-speaking country, but has a history of searching exclusively for songs in English, the results could be weighted more heavily towards the language of their search instead of their geographic location. On the other hand, the default for a user without search history would have a larger weighting on songs that users in similar geographic locations preferred, in order to maximize the likelihood of recommending songs that people in the user's proximal network would be interested in.

### 7.3 Shazam Song Recognition Incorporation

Given that users are very busy in today's modern world, one feature that would make AlgoDJ more readily usable for consumers would be allow the user simply to record a clip of the song of interest, and rely on Shazam's song recognition, in order to generate the recommended playlist. Though our team initially set out to incorporate this audio-recognition component into our app, time did not allow us to fully incorporate. However, we recognize that additional value to the user would be available through such an additional feature.

## References


[1] *2022 State of the API Report*. Postman.com, 2022. URL: https://web.postman.com/state-of-api/api-technologies/#api-technologies.

[2] C. M. Adam Schroeder. *The Book of the Dash*. No Starch Press Inc. 2022.

[3] D. Afchar et al. "Explainability in music recommender systems". In: *AI Magazine* 43.2 (2022), pp. 190–208.

[4] L. Barrington, R. Oda, and G. Lanckriet. "Smarter than Genius? Human Evaluation of Music Recommender Systems." In: Jan. 2009, pp. 357–362.

[5] R. Beakta. "Big Data And Hadoop: A Review Paper". In: *international journal of computer science information te* 2 (Jan. 2015).

[6] G. Brito and M. T. Valente. "REST vs GraphQL: A Controlled Experiment". In: *2020 IEEE International Conference on Software Architecture (ICSA)*. 2020, pp. 81–91.

[7] E. Dabbas. *Interactive Dashboards and Data Applications*. Packt Publishing Ltd. May 2021.

[8] *Approaching the largest 'API': extracting information from the Internet with Python* code4lib Journal.39 (2018). URL: https://journal.code4lib.org/articles/13197.

[9] D. Hillard. *Practices of the Python Pro*. Simon and Schuster. 2019.

[10] M. B. Jovan Veljanoski. "Interactive and scalable dashboards with Vaex and Dash". In: *Medium* (June 2020). URL: https://medium.com/plotly/interactive-and-scalable-dashboards-with-vaex-and-dash-9b104b2dc9f0.

[11] R. Mitchell. $Web_Scraping_with_Python.bibtex$. 2018.

[12] E. Pampalk, A. Flexer, and G. Widmer. "Improvements of audio-based music similarity and genre classification". In: *ISMIR* (Jan. 2005).

[13] S. Pan. *The performance comparison of hadoop and spark*. Culminating Projects in Computer Science and Information Technology. 7., 2016. URL: https://repository.stcloudstate.edu/csit_etds.

[14] C. Richthammer, J. Sänger, and G. Pernul. *Interactive Visualization of Recommender Systems Data*. Neuchâtel, Switzerland, 2017.

[15] A. Rodriguez. *Introduction to RESTful Web services*. ENGLISH. IBM. IBM, Feb. 2015. URL: https://developer.ibm.com/articles/ws-restful/.

[16] S. Salloum et al. "Big data analytics on Apache Spark". In: *International Journal of Data Science and Analytics* 1.3 (2016), pp. 145–164.

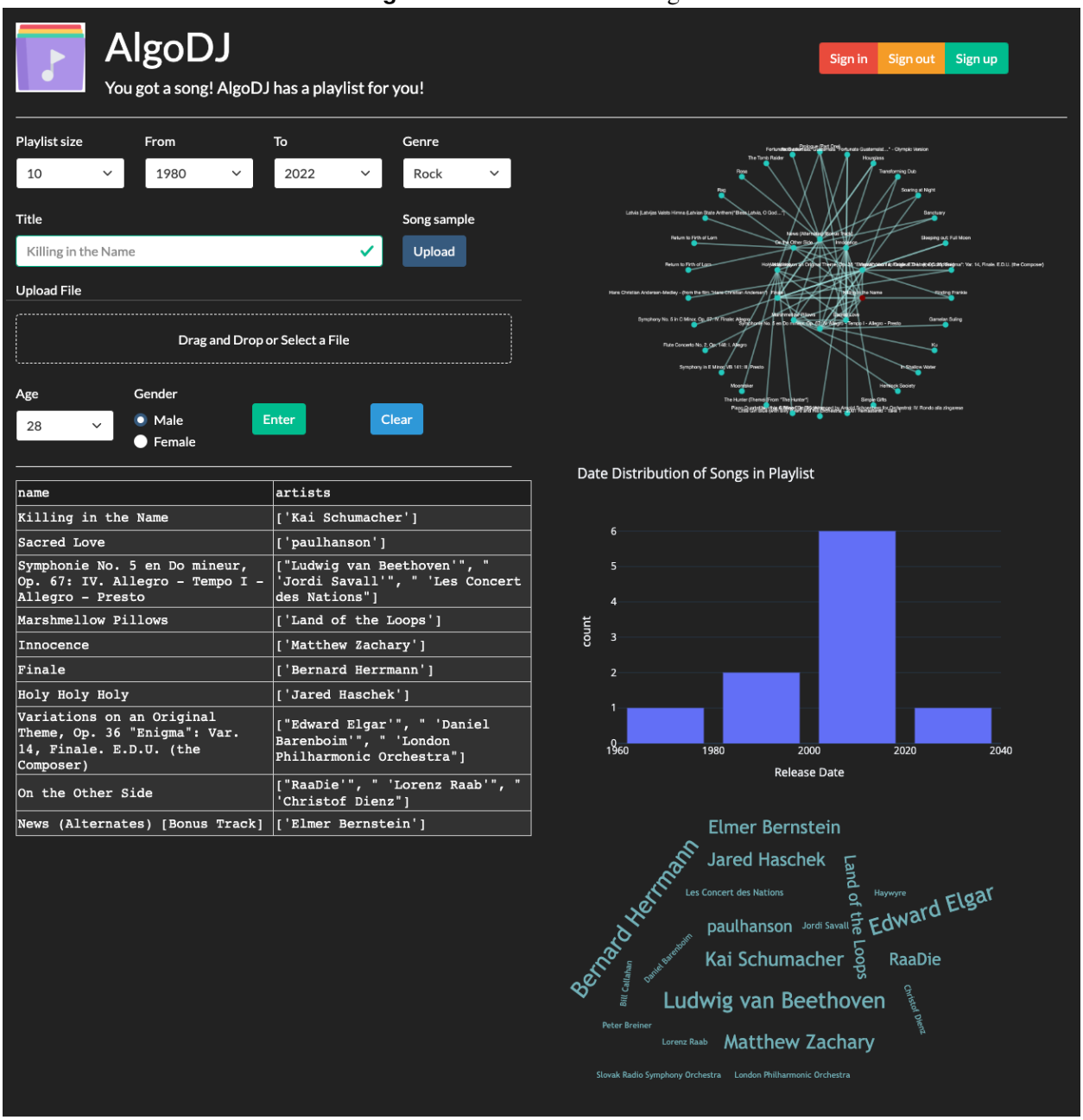
[17] C. Sievert. *Interactive Web-based Data Visualization with R, Plotly, and Shiny*. Chapman & Hall/CRC the R series. Research rep. 2020.

[18] C. Swa and Z. Ansari. "A Data Flow Framework Based on Hadoop Map Reduce". In: *International Journal of Engineering Trends and Technology* 50 (Aug. 2017), pp. 271–275.

[19] R. M. G. Wesley and P. Terlecki. *Leveraging Compression in the Tableau Data Engine*. Snowbird, Utah, USA, 2014.

## 8. Appendix

**Figure 2.** User Interface of AlgoDJ




| name | artists |
|---|---|
| Killing in the Name | ['Kai Schumacher'] |
| Sacred Love | ['paulhanson'] |
| Symphonie No. 5 en Do mineur, Op. 67: IV. Allegro - Tempo I - Allegro - Presto | ["Ludwig van Beethoven'", " 'Jordi Savall'", " 'Les Concert des Nations"] |
| Marshmellow Pillows | ['Land of the Loops'] |
| Innocence | ['Matthew Zachary'] |
| Finale | ['Bernard Herrmann'] |
| Holy Holy Holy | ['Jared Haschek'] |
| Variations on an Original Theme, Op. 36 "Enigma": Var. 14, Finale. E.D.U. (the Composer) | ["Edward Elgar'", " 'Daniel Barenboim'", " 'London Philharmonic Orchestra"] |
| On the Other Side | ["RaaDie'", " 'Lorenz Raab'", " 'Christof Dienz"] |
| News (Alternates) [Bonus Track] | ['Elmer Bernstein'] |

**Figure 3.** Artist Network example from AlgoDJ

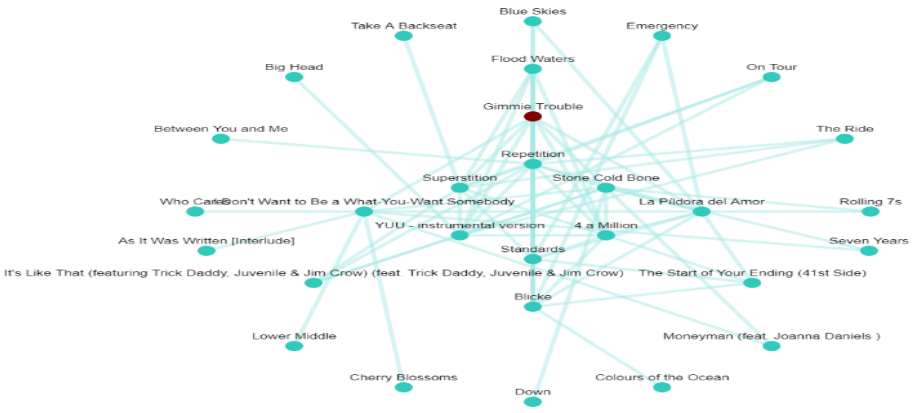



**Figure 4.** GitHub Repository