SoundTransfer: Data Portability Between Music Streaming Services

Abhyudaya Sharma
Brown University

Grant Lee
Brown University

Muskaan Patel
Brown University

Abstract

This project emphasizes the absence of a direct method of migrating user data between music streaming services, as required by the GDPR. We devise a comprehensive rubric to evaluate the accessibility, completeness and security of existing (indirect) data migration methods and identify the gaps in these methods.

We then built a web app, which makes three main contributions. Firstly, it provides a novel standardized schema to represent user data from all music streaming services. Secondly, it is built upon an extensible framework for ingesting and exporting data using data format adapters. Thirdly, it emphasizes the importance of performing the data migration process on a secure and private environment. The combination of these features provides greater assurances of security, privacy, accuracy and accessibility in the data migration process.

1 Introduction

Article 20 of the European Union's General Data Protection Regulation (GDPR) emphasizes that data subjects should have the right to have their personal data *transmitted directly from one controller to another, where technically feasible* [1]. Unfortunately, in many contexts, it is unlikely for a data controller to perform a direct transfer of user data to another controller. Music streaming is one such context. Table 1 describes some types of user data in the context of music streaming services. Account-related data is likely to be sensitive to specific streaming services and hence should *not* be transferred to other services. On the other hand, music-related data tends to be the data type of concern that users are interested to transfer between streaming services.

Despite the popularity of music streaming services such as Apple Music [2] and Spotify [3], there is no common framework that allows for a direct transfer of user data from one service to another. The lack of collaboration between music streaming service providers on this front leads to the status quo, in which (1) the responsibility of performing a transfer of user data is taken on by the data subjects instead, and (2) the nature of the transfer is no longer direct from one controller to another.

Account-related data	**Music-related data**
Account Settings	Saved Songs
Payment Information	Saved Albums
Profile Information	Followed Artists
	Saved Playlists
	Listening History

Table 1: Music streaming service user data

Relying on manual processes for data migration between music streaming services tends to be excessively cumbersome and time-consuming for data subjects. There exists third-party services that facilitate this process, which gives an initial impression of convenience, but in fact comes at a compromise of user privacy and security.

In this study, we first evaluate 3 existing methods of *indirectly* transferring one's user data between music streaming services. To do so, we propose a comprehensive rubric that considers 12 factors contributing to a complete, secure and accessible data migration process. Using our evaluation, we are able to identify gaps in these existing methods. These learning points inspire the design for *SoundTransfer*, our own data transfer application. The result is a more secure, complete, accessible and extensible data transfer system that makes the *indirect* nature of this data migration process feel more *direct*.

SoundTransfer makes three main contributions:

1. A standardized schema to represent user data from music streaming services
2. An extensible framework for ingesting and exporting music streaming user data via data format adapters
3. A secure and private environment for the data migration process to happen, so that data subjects do not have to

trust any other party besides the streaming services as well as their own local system

2 Background/Related Work

2.1 Applied Terminology

In the realm of data privacy and user rights, the concept of data portability stands as a fundamental principle. Data portability refers to the right of individuals to access and move their personal data between different services or platforms, enabling users to exercise control over their information. [1] This principle is a key component of regulations like the GDPR implemented by the European Union. The GDPR requires organizations to allow users to transfer their data from one platform to another seamlessly, fostering competition and empowering users to manage their data effectively.

Spotify and Apple Music, in compliance with the GDPR, [1] allow users to download their personal data and listening history. [4, 5] This "right to data portability" allows users to request a *data export*, which is essentially a compressed archive containing user-specific information like playlists, saved songs, listening history and even search queries. However, since there is no standardized format for these data export files, they can be provided in a multitude of file formats, including JSON, CSV or even HTML.

REST APIs [6] allow different services to communicate over the internet and exchange data. Music streaming platforms like Spotify and Apple Music provide public APIs that allow users (or other services) to fetch data they have access to. By making enough API calls, users can, in most circumstances, fetch almost all of their personal data. Although there is no legal obligation for a platform to provide APIs, there is a strong business argument for having a complete API since a good API can enable profit-making integrations with other services.

In terms of data storage, MongoDB is a widely used NoSQL database that provides a flexible and scalable solution for storing diverse types of data. It offers the capability to store and manage data necessary for our project, accommodating our universal schema while also allowing us to store data in different formats from different music streaming services efficiently. [7]

Third-party services like TuneMyMusic[8] or Soundiiz[9] have emerged as intermediaries, attempting to bridge the gap between different music streaming services by offering tools for users to transfer playlists and tracks between platforms. However, despite the convenience, these services pose concerns regarding user privacy and data security, as they require users to trust an external party with their personal data. Additionally, these services come at a monetary cost to the users.

The International Standard Recording Code (ISRC) [10] is a unique identifier assigned to individual sound recordings and music videos. It consists of a 12-character alphanumeric code that helps in identifying and distinguishing different tracks or audiovisual recordings worldwide. ISRCs enable the tracking of usage, distribution, and sales of specific recordings across various platforms and territories, allowing for effective royalty payments and copyright management within the music industry.

The Universal Product Code (UPC) [11] is a barcode symbology widely used for retail purposes to identify products. In the context of the music industry, UPC barcodes are assigned to album or single releases, providing a standardized way to track and manage physical and digital music sales. These codes contain a unique 12-digit numerical identifier and are essential for inventory management, sales tracking, and reporting across retail and digital platforms, aiding in effective distribution and marketing of music products.

2.2 Existing Methods

We evaluated three existing methods for data migration: (a) third-party services like TuneMyMusic [8] and Soundiiz [9], (b) manual song-by-song transfer, and (c) generating GDPR-compliant exports combined with manual playlist and library recreation. Understanding the need for an objective assessment, a comprehensive rubric was devised, encompassing crucial factors essential for a user-friendly data portability tool. This rubric is provided in Appendix A (Table 2).

a Third-Party Data Migration Services

Engaging a third-party service for data transfer offers several advantages. These platforms often excel in user-friendliness, providing interfaces that demand minimal effort and ensuring compatibility across various platforms. They streamline the transfer process, offering convenience and accessibility to users, thereby simplifying the sharing and distribution of music content across different channels and audiences. However:

1. **Inefficiency**: Some third-party services have delays or lapses in data transfer and management, impacting productivity.

2. **Costly**: Certain services can be expensive, involving substantial fees for access to premium features or extensive usage.

3. **Security Vulnerabilities**: Third-party platforms exhibit significant security vulnerabilities, exposing users to the potential risk of data breaches or unauthorized access to sensitive information.

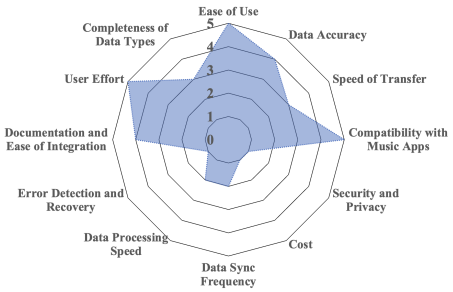

Figure 1: Evaluation of Third Party Services

b Manually Transferring Data from Platform to Platform

The manual import-export method refers to one in which the data subject manually recreates their library and playlists by interacting with the streaming services' apps. This manual process provides complete security and compatibility. Security is guaranteed as users are directly interacting with their data without involving third-party services. Additionally, as long as the user has access to whichever streaming services they are interested in, this method would be compatible across all music streaming services. However:

1. **Inefficiency**: This method is highly inefficient, especially for individuals managing extensive playlists, as it demands substantial user effort and time and is prone to human error.

2. **Scalability**: Long-time users of streaming services are likely to have an increasingly large volume of data (songs, playlists and listening history). This manual process becomes gradually cumbersome and challenging to scale efficiently.

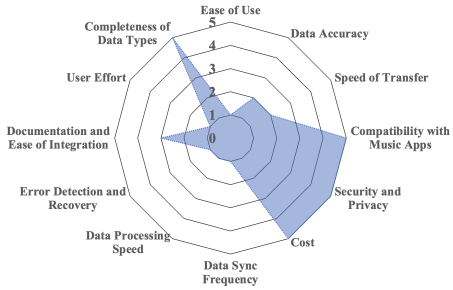

Figure 2: Evaluation of Manual Data Import-Export

c Generating GDPR-Compliant Exports from Previous Service, Manually Recreating Data in New Service

Another method is to generate GDPR-compliant data exports from one's streaming platform, and manually recreate the library and playlists on the other platform. This method offers enhanced data completeness and assurance that the data transferred maintains its integrity and accuracy. However:

1. **Inefficiency:** One significant drawback lies in its sluggish nature, particularly due to the extended duration required for streaming services to generate the data exports, which can range from 1 to 4 weeks. This prolonged timeline makes it a slow and less viable option, especially in scenarios where quick data transfer or synchronization is crucial.

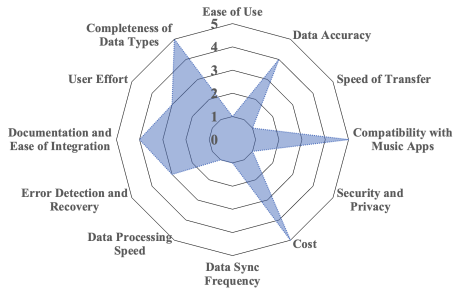

Figure 3: Evaluation of GDPR Exports and Manual Import

3 Design

3.1 System Architecture

Figure 4 describes the overall architecture of the *SoundTransfer* system. Note that the entirety of our system operates entirely within the user's local device, instead of an online environment hosted on a third-party cloud server. This design choice inherently guarantees that no one besides the data subjects themselves (and the streaming services) have access to the user data. This translates to greater security and protection from potential data breaches.

Orange-colored arrows denote the flow of user data from the music streaming service provider to the user's local system. This ingestion process is associated with the scenario of a user migrating away from a specific music streaming service. Data can be ingested from the service provider via two methods: (1) GDPR-compliant data exports directly requested by the user, and (2) developer APIs provided by the service provider. This data is transformed from its original format into the schema dictated by *SoundTransfer*, and subsequently stored in the user's local MongoDB database.

Blue-colored arrows denote the flow of user data from the user's local system to a music streaming service provider. This exporting process is associated with the scenario of a user migrating to a specific music streaming service. Data can only be exported to the service provider via developer APIs. Once again, this data is transformed from our schema-defined format into the formats required by these APIs.

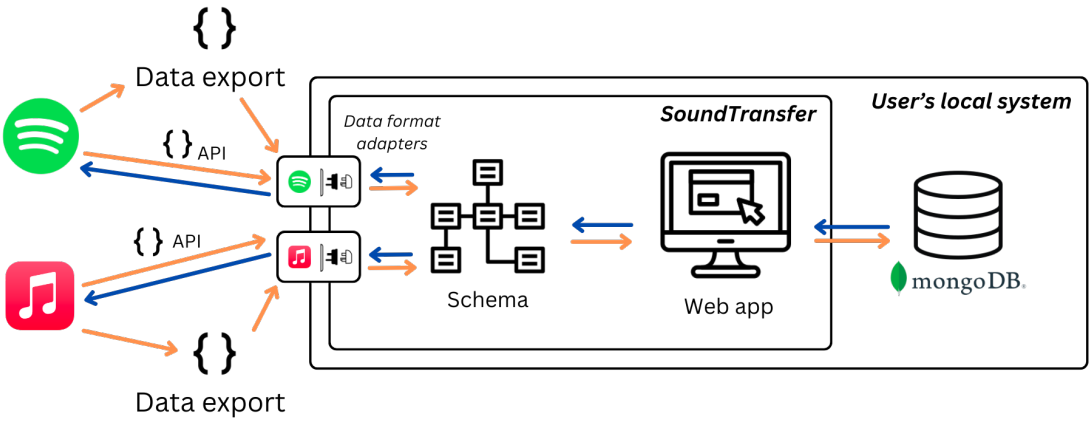

Figure 4: System Architecture of *SoundTransfer*

3.2 Standardized Schema

The core of the *SoundTransfer* system is the standardized schema that is used to represent the user's data. The implementation of such a schema is necessary due to the fact that every streaming service formats its data in a different way. The differences include (but are not limited to) the organization of the data and types of data that are provided.

In particular, each streaming service primarily uses their own internal identification code to identify songs. Instead of storing these internal identifiers, our schema stores the ISRC of the song instead. The ISRC of each song allows us to retrieve the internal identifier for every music streaming service as and when they are required. The conversion between ISRCs and internal identifiers is handled by the data format adapters, and usually involves additional requests to the streaming services' APIs. Similarly, we use the UPCs of the albums as global identifiers, converting between them and internal identifiers using developer APIs.

Additionally, we designed this schema to be a union (rather than intersection) of the data types that are provided by each streaming service. Doing so provides us with a number of benefits.

Firstly, we gain greater completeness. Some music streaming platforms may have include certain features that do not (yet) exist on other platforms. For example, at time of writing, Spotify allows the creation of collaborative playlists while Apple Music does not.[1] Nevertheless, given the union nature of our schema, should a data subject ingest their playlists from Spotify, we retain information about whether or not a given playlist is collaborative. By doing so, should Apple Music implement a similar feature in the future, that information already exists on the user's database and is ready to be exported to Apple Music.

Greater completeness also gives us greater extensibility. The schema allows us to store information that may not be immediately useful, but could be so in the future. For example, the GDPR-compliant data exports for both Spotify and Apple Music include the user's listening history. However, this can currently be considered unactionable data, since the developer APIs for both platforms do not provide functionality to export listening history into their systems. We imagine that listening history data could potentially be very useful, in a way that streaming platforms can provide users with accurate song and/or playlist recommendations immediately upon account creation and data migration. Should this happen, the user's local database would already contain this data, ready to be exported to the target streaming service.

To ensure that we populate the local database with maximum completeness (as per the schema), each ingestion function usually involves at least one additional request to the streaming services' APIs to retrieve additional information.

3.3 Web App

Another integral part of our system is the implementation of a web app that acts as a graphical user interface (GUI) for data subjects to visualize their data and perform data ingestions and exports. This is necessary to ensure that the data migration process is accessible and approachable for the layman (who is assumed to be familiar with web apps in general).

Another assumption is that user of the web app is knowledgeable about the process of generating developer tokens from the music streaming service providers. These developer tokens are necessary in order to authorize the user to make API requests. We understand that this assumption is unlikely to be true, hence we provide relevant instructions and documentation along with the web app.

The web app consists of 6 pages of data visualization and ingest/export actions. Figure 6 depicts two such pages.

[1] Collaborative playlists are scheduled to arrive in Apple Music in iOS 17.3 which is slated for a 2024 release

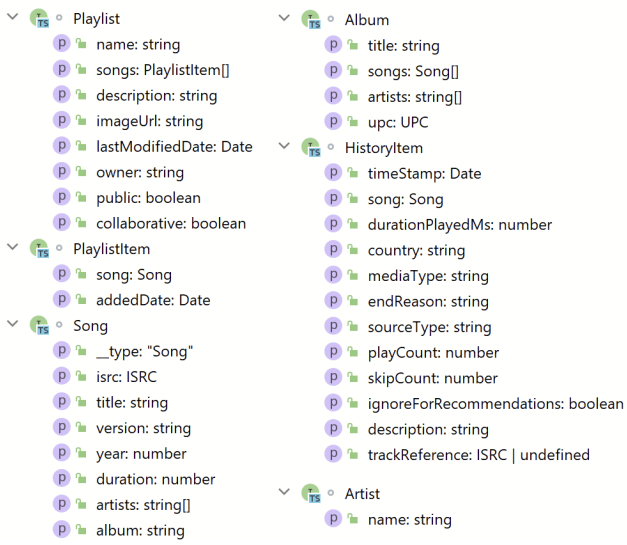

Figure 5: A subset of the types from our universal schema

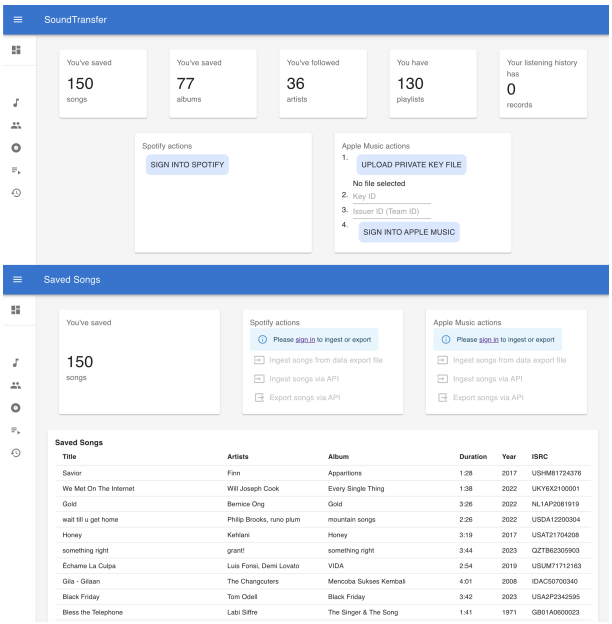

Figure 6: Dashboard of web app

The first page is the dashboard, providing a summary of the number of saved songs, albums, artists, playlists as well as the records in the user's listening history. The dashboard also provides an interface for users to log into their music streaming services and provide the aforementioned developer tokens. After logging in, buttons are provided to allow users to ingest and export data of all 5 types (songs, albums, artists, playlists, listening history).

Each of the 5 other pages are individual visualizations for each data type in the format of a table. Buttons are also provided for users to ingest and export only data of the corresponding type.

4 Implementation

The *SoundTransfer* project has been made open-source, accessible on GitHub.[2]

We implemented our application in TypeScript running on Node.js using `ts-node` which *type-checks* and *transpiles* the code to JavaScript on the fly. The application stores ingested user data in a locally hosted MongoDB database, and is supported by a web app user interface, written in JavaScript with React.js and served by Express.js. The frontend interacts with the backend using REST APIs.

Authorization is handled completely in the browser and the backend never stores any user credentials, though the necessary credentials are sent to the backend with every API call. For Apple Music, we delegate the authentication implementation to MusicKit.js, while for Spotify, we implemented it ourselves.

Once authorized, users choose to ingest their data into the local database either by uploading their GDPR-compliant data export files, or by requesting their data via the music streaming platforms' APIs. Once data has been ingested, it can be exported to other platforms. Our app also includes visualizations for the ingested data including but not limited to songs, playlists, albums, artists and listening history.

In our implementation, we also discovered a number of interesting limitations about the platform APIs, for which we had to find workarounds. The Apple Music API, for instance, allows users to get artists in their library but does not allow users to add new artists. For importing library playlists, we need to first make an API call to get all playlists in the user's library. For each playlist in that list, we need to make an API call to get the tracks in that playlist. The returned data includes a 'library identifier' for each track which cannot be directly used to fetch the ISRC, the unique identifier we use to map songs across platforms. Hence, for each of these songs, we create another API call which would then include the ISRC.

5 Evaluation

Using the same rubric as before, we evaluate the effectiveness of *SoundTransfer* as two separate methods. The first involves data ingestion using the GDPR-compliant data export provided by the music streaming service and uploaded by the user. The second involves data ingestion using the streaming service's APIs. The former method introduces a time delay, but provides more complete information than the latter.

[2] https://github.com/grrrrnt/soundtransfer

5.1 Data Ingestion using User's GDPR-Compliant Data Exports

The first method suffers from low scores for 'Speed of Transfer' and 'Data Processing Speed', because the user has to manually initiate a data export request from their current music streaming service, wait up to a month to receive the data export files, then use our application to process these files. As discussed previously, the *SoundTransfer* platform provides greater ease of use, accuracy and completeness, while also providing useful diagnostic messages.

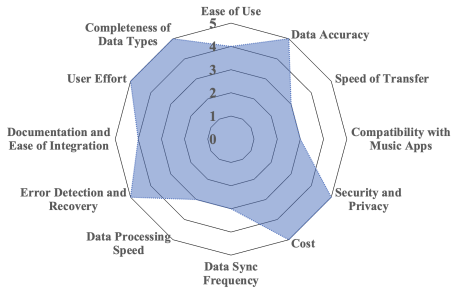

Figure 7: Evaluation of Manual GDPR Exports and Automated Import

5.2 Data Ingestion via Music Streaming Service APIs

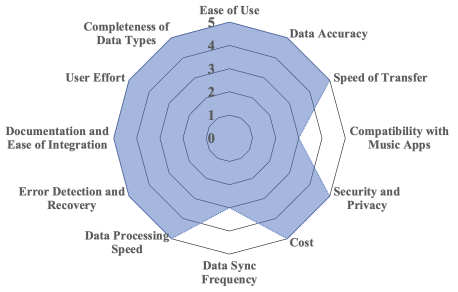

Figure 8: Evaluation of Automated Import and Export

The second method requires the user to perform no additional steps other than providing credentials to our application. The platform suffers slightly in the 'Compatibility with Music Apps' section because we would have to implement ingestors and exporters for each streaming platform based on the available APIs. This method also scores low in the 'Data Sync Frequency' criterion because the user has to manually initiate the ingest or export process. However, we believe, with a little work and a background process, we can fetch and export new data periodically.

5.3 Discussion

Our platform meets or exceeds existing solutions in almost all features considered in our rubric. It provides a fast, easy-to-use, and well-documented application for users to transfer their data from Apple Music to Spotify, or vice versa.

The effectiveness of the *SoundTransfer* system can be attributed to the extent of the functionality made available to us by the APIs of the streaming platforms. However, this also means that *SoundTransfer* is only as functional as the APIs provided by the streaming platforms.

Additionally, *SoundTransfer* requires users to provide developer credentials to access these APIs. For Spotify, the user needs to register their account on Spotify's developer portal, and for Apple Music, the user needs to register as an Apple developer by paying a $99 fee. Even with these prerequisites fulfilled, there may be situations where users may hit rate limits when fetching data using the APIs.

Our solution to match songs and albums across platforms using ISRCs and UPCs works well and produces accurate results, successfully matching our entire libraries across the two platforms. However, we imagine that there are some situations in which using these identifiers may not work. ISRCs may not be portable across countries, while UPCs may not work in Europe or other parts of the world, because they compete with EAN (European Article Numbers) [12] which is another unique product identifier. Furthermore, not all songs or albums in a user's library may be available on all streaming platforms. For now, we handle such situations by printing out a warning.

Finally, we must say that *SoundTransfer*'s design is possible only because of the existence of ISRCs, UPCs and developer APIs provided by streaming services. Without ISRCs and UPCs, there would be no reliable method to identify songs across platforms. Without developer APIs, and without resorting to unreliable and insecure techniques like screen readers and user interface automation, a data portability tool like ours would not exist.

6 Future Work

Our solution proves that it is *not* technically infeasible for music streaming services to allow transferring user data to other services. Our universal schema presents a solution that can be adapted by music streaming platforms to facilitate easy data interchange. This also raises a fascinating question about whether music streaming services are in violation of GDPR Article 20(2), which requires data controllers to allow users to transfer their data when *technically feasible*.

In the future, we would like to make our solution more complete by adding support for other streaming platforms like Tidal, YouTube Music, Amazon Music and Deezer. Thankfully, *SoundTransfer* is built to be extensible. Adding support for a new streaming platform is as simple as implementing a

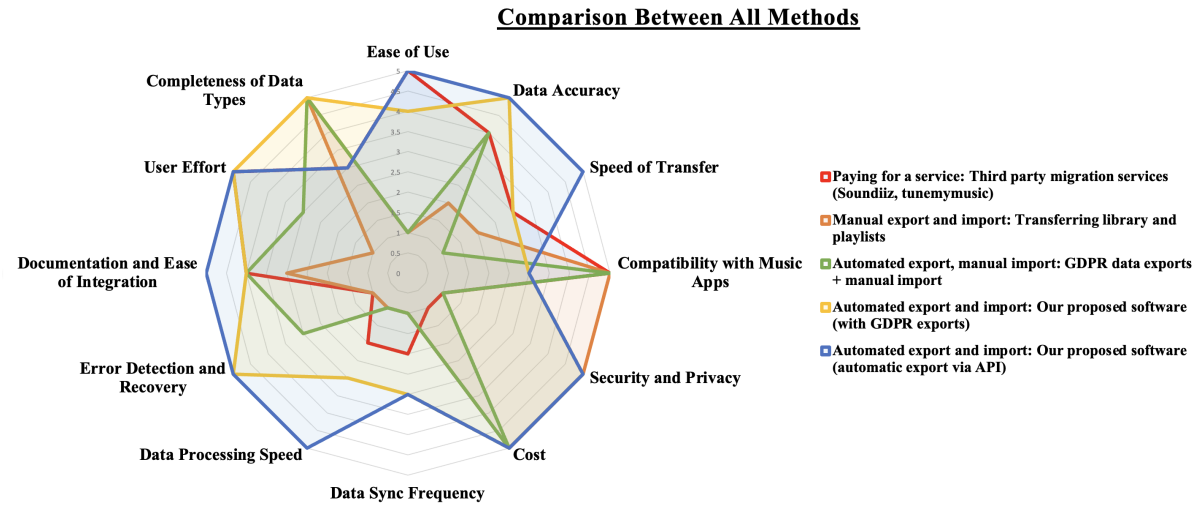

Figure 9: Comparison between all the methods

new data format adapter to facilitate the ingestion and exporting of data.

Another idea that came up during our research was that because of existing globally unique identifiers (like UPCs, EANs, and ISBNs), similar data portability tools can be created for services like eBook marketplaces, video streaming services, e-commerce websites and other services that do not currently have any such existing data portability tools. For example, UPCs can be used to help Amazon Prime customers move their shopping recommendation-related data to Walmart+ (or vice versa).

Our application operates as an open-source project (link to our GitHub repository can be found in Section 4), welcoming developers to contribute and enhance its functionality. With our inclusive schema, developers can fork the project, leveraging its comprehensive structure to build and customize their features. Our extensible and adaptable schema facilitates the addition of various music streaming services, enabling their seamless integration and utilization within our platform. Developers can utilize our robust database infrastructure to implement their own projects. Above all, we highly encourage for music streaming services to adopt our schema, to promote collaboration and interoperability across music streaming platforms.

References

[1] GDPR Article 20. URL https://gdprhub.eu/index.php?title=Article_20_GDPR#(1)_Right_to_data_portability.

[2] Apple. Apple Music, . URL https://www.apple.com/apple-music/.

[3] Spotify. Spotify, . URL https://open.spotify.com/?

[4] Apple. Apple Music - Download your Data, . URL https://support.apple.com/en-us/102208.

[5] Spotify. Spotify - Download your Data, . URL https://www.spotify.com/us/account/privacy/?_ga=2.53014496.998691067.1702675676-473638565.1701298919.

[6] Roy Thomas Fielding. *REST: Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine, 2000. URL http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

[7] MongoDB. What is MongoDB? URL https://www.mongodb.com/docs/manual/.

[8] Transfer Playlists Between Music Services | Tune My Music. URL https://www.tunemymusic.com.

[9] Soundiiz - Transfer playlists and favorites between streaming services. URL https://soundiiz.com.

[10] The International Standard Recording Code. URL https://isrc.ifpi.org/en/.

[11] Carla Johnson. What Is UPC? URL https://www.musicgateway.com/blog/music-industry/what-is-upc.

[12] GS1. EAN/UPC barcodes. URL https://www.gs1.org/standards/barcodes/ean-upc.

A Appendix: Evaluation Rubric

Table 2 contains the rubric we used for evaluating existing data migration methods as well as the *SoundTransfer* system.

B Individual Contributions

All members contributed equally. Grant worked on the Spotify data format adapter. Abhyudaya worked on the Apple Music data format adapter. Muskaan worked on evaluations of existing services and of our own system.

Features	Highly Unsatisfactory	Unsatisfactory	Average	Satisfactory	Highly Satisfactory
Ease of Use	Very complex and non intuitive interface	Very complex but manageable	Average UI	User-friendly with room for improvement	Extremely easy to use and intuitive
Data Accuracy	Migrated data was completely inaccurate	Migrated data mostly inaccurate, some correctness	Moderately accurate	Highly accurate migrated data with minor discrepancies	100% accurate data with no discrepancies
Speed of Transfer	More than 1 week	Within 1 week	Within a day	Within half a day to ensure efficient transfers	Almost instantaneous, ensuring swift data transfer
Compatibility across streaming services	Not compatible with any popular apps	Partial compatibility with 1 popular app	Compatible with 1-2 popular existing apps	Compatible with 3-4 popular existing apps, ensuring broad use	Compatible with all music apps, ensuring universal accessibility
Security and Privacy	Highly insecure, high risk of data leaks	Somewhat secure, with a moderate risk of data leaks	Somewhat secure, low risk of personal data leak	Secure, with minimal risk of non-personal data leaks	Highly secure, no data leaks, ensuring complete privacy
Cost	Very expensive (more than $100)	Somewhat expensive ($50-100)	Slightly expensive ($20 - 50)	Within $20, ensuring affordability for users	Free, offering a cost-effective solution
Data Sync Frequency	Manual synchronization requiring over one hour of user effort	Manual synchronization requiring between 1 hour and 1 minute of effort	Manual synchronization requiring less than 1 minute of effort	Frequent automatic sync	Real-time synchronization
Data Processing Speed	Very slow processing speed	Slow processing speed	Moderate processing speed	Fast processing speed	Very fast processing speed
Error Detection and Recovery	No error handling or recovery mechanism	Limited error handling and recovery capabilities	Basic error handling and recovery	Robust error handling and recovery capabilities	Advanced error handling and immediate recovery
Documentation and Ease of Integration	No documentation and challenging integration	Limited documentation and somewhat challenging integration	Adequate documentation and integration	Good documentation and ease of integration	Comprehensive documentation and seamless integration
User Effort	Extremely high user effort, complex interactions	High user effort, requiring significant user input	Moderate user effort, some user input required	Low user effort, intuitive interactions	Minimal user effort, seamless and automated interactions
Completeness of Data Types	Severe limitations in supporting essential music transfer data types	Partial support for necessary data types	Basic coverage of fundamental data types	Comprehensive support for essential data types	Exceptional support for diverse data types

Table 2: Our Proposed Rubric (for this rubric, all figures are based on a user library with 1000 songs, 10 artists, 50 playlists, 15 albums and 10,000 items in the listening history)