



University of Arkansas – CSCE Department

Capstone I – Preliminary Report – Fall/Spring 2019

JukeBox - A playlist importer/exporter

Aaron Tucker, Marc Aranibar, Rhett Brandon, Noah Holt, Tyler Gay

Abstract

Many people face the problem of needing to move from one streaming platform to another, or maintaining a playlist across multiple streaming platforms. The objective of this application would be to help users move and maintain their playlists across multiple streaming services. The application will be a website that stores playlists, and integrates with multiple streaming platform's APIs. The significance of this application is that it allows for easy movement from one platform to another, and it allows independent (not affiliated with the streaming services) playlist creators to establish a presence across multiple services.

1.0 Problem

Switching from or between music streaming services can be frustrating when you have large playlists you want to recreate on other platforms. It is also cumbersome maintaining a playlist between different platforms like Youtube, Spotify, Soundcloud, etc. when some platforms have certain tracks and others do not. Sometimes users want to know their most listened to songs or what genres they listen to the most so they know where to branch out and explore their tastes. Many of these platforms have this information but it is hidden most likely to reduce clutter.

2.0 Objective

Our web app would be a simple intermediary between these platforms where you can rip information from your playlists in other apps and create these same playlists in other platforms

provided they have intersecting content. We would also like to see if we could play music from the webapp using content from multiple platforms at once creating a playlist from multiple platforms. If possible we also have plans to give the user interesting data metrics about their listening habits overall and their tastes on individual platforms like Spotify that otherwise obfuscate data used to provide you listening suggestions, as well as giving a way for users to interact by sharing playlists or collaborating to create playlists.

3.0 Background

3.1 Key Concepts

A playlist is a list of media to access in a particular order that can be modified and shared by the user. In the context of music, they can be a passive stimuli, or “going on in the background”. So someone may be listening to a playlist while jogging, studying, or playing video games. Playlists can also be an active experience that allows people to discover new content. With the rise of streaming platforms, playlists have become the primary way people organize their music.

A streaming platform is an on demand online entertainment source for media. The revenue model is that listeners can either listen to advertisements, or pay for an ad-free experience. A factor specific to music streaming services is that content creators are paid per stream of one of their tracks. This gives content creators the incentive to upload their music for passive revenue.

An API is a set of routines, protocols, and tools for building software applications. These may be related to hardware, such as the arduino, or related to other software. Several music streaming services have APIs that allow developers to read / write to content such as playlists. These APIs will require a credential, normally called an API token. We intend to leverage these APIs to create a seamless playlist experience.

An ETL (Extract, Transform, Load) is an automated pipeline from one data source to another data source. First it extracts data from a source, normally a database. Then it transforms this data into another format suited for another source. Then it loads that transformed data into the other data source. An ETL will exist for each streaming service to take a playlist from them, and store it on our database.

OAuth defines a delegation protocol for conveying authorization decisions across a network of web-enabled applications and APIs. OAuth is used in a wide variety of applications, including providing mechanisms for user authentication. This has led many developers and API providers to incorrectly conclude that OAuth is itself an authentication protocol and to mistakenly use it as such. (“User Authentication with OAuth 2.0”). In the context of our application, OAuth allows for the connection to streaming services. Upon authorization, our application will receive a token that allows for access to the streaming service’s API.

3.2 Related Work

What other researchers or developers have accomplished in this project area including references in [ref number] format, e.g., [1]. What are the problems with those implementations

that yours will solve or why will yours be better or different?

<https://soundiiz.com/>

Very good platform, but doesn't have any user interaction or music analytics.

<https://playlistconverter.net/>

Allows sharing but does not have a wide range of services and no user analytics.

We want to design an application with a wide range of services to transfer music between as well as giving the user the ability to share their curated content lists and giving them analytics/suggestions on their listening habits to promote music discovery.

4.0 Design

4.1 Requirements and/or Use Cases and/or Design Goals

- User wants to import data from a playlist on a streaming service into JukeBoxer
- User wants to export data from playlist on JukeBox onto a playlist on a streaming service
- User wants to construct playlists for DJing, and uses the data we aggregate to construct a better playlist
- User wants to share playlist from JukeBoxer with friend
- User wants to collaborate and build playlist with friend on JukeBoxer
- User wants to know more about their listening habits

4.2 [High Level / Detailed] Architecture

Landing page:



JUKEBOXER

Playlist Import Export
My Music
Metrics

My Music:

JUKEBOXER



My Music:

My new playlist

Dolor sit amet - vitae consequat ipsum
Consectetur adipiscing - nunc semper elementum
I'm a snort - bork
Uberrimas fides - lync
"Return the slab" - King Ramses
Deathgrips is online - Carlie Ray Jepsen

Playlist Import/Export: (Import)

JUKEBOXER

CONNECTED ACCOUNTS | SIGN IN



Import Playlist from:

- Spotify
- Apple Music
- Soundcloud

Export Playlist to:

- Spotify
- Apple Music
- Soundcloud

Export Playlist Name:

My new playlist

SELECTED IMPORT PLAYLIST

NEW EXPORT PLAYLIST

Dolor sit amet - vitae consequat ipsum
Consectetur adipiscing - nunc semper elementum
I'm a snort - bork
Uberrimas fides - lync
"Return the slab" - King Ramses
Deathgrips is online – Carlie Ray Jepsen

Playlist Import/Export: (Export)

JUKEBOXER

CONNECTED ACCOUNTS | SIGN IN



Import Playlist from:

- Spotify
- Apple Music
- Soundcloud

Export Playlist to:

- Spotify
- Apple Music
- Soundcloud

Export Playlist Name:

SELECTED IMPORT PLAYLIST

NEW EXPORT PLAYLIST

Lorem ipsum - Phasellus dignissim at quam et euismod
✓ Dolor sit amet - vitae consequat ipsum
✓ Consectetur adipiscing - nunc semper elementum
Uberrimas fides - lync
✓ I'm a snort - bork
✓ "Return the slab" - King Ramses
Ok Boomer - Splitknot
✓ Deathgrips is online – Carlie Ray Jepsen

My Metrics:

JUKEBOXER

CONNECTED ACCOUNTS | SIGN IN



Metrics:

Title: "Return the slab"
Artist: King Ramses

Low Level information:

Key: C# major (42.3%)
Chords Key: C# major
Danceability: 0.34
Bpm: 60
Beat count: 156

High Level information:

Voice: Egyptian god
Gender: male

SELECTED IMPORT PLAYLIST

Lorem ipsum - Phasellus dignissim at quam et euismod
Dolor sit amet - vitae consequat ipsum
Consectetur adipiscing - nunc semper elementum
Uerrimas fides - lync
I'm a snort - bork
"Return the slab" - King Ramses
Ok Boomer - Splitknot
Deathgrips is online - Carlie Ray Jepsen

Connected Accounts:

JUKEBOXER

CONNECTED ACCOUNTS | Marc Aranibar



Connected Accounts:



Spotify:
marcaranibar



Apple Music:
aranibar1998



Soundcloud:
[link](#)

The application will allow you to import your playlists from multiple music subscription services, export your desired playlist to the platform of your choice, and show analytics about the tracks in your playlist. The services supported include Spotify, Apple Music, and

SoundCloud. These services were chosen because their APIs allow read / write access to playlists. We intend to integrate with each of these services via OAuth, an open-source standard that gives access to a third-party service without giving credentials such as a password.

This service will be a website with an Express.js backend and React.js frontend. We chose these technologies because of our team's familiarity with JavaScript from work experience and the software engineering class. Using a website also avoids the need to create a mobile app for both Android and iOS, by providing a mobile website. Both our backend API and our frontend will be hosted on the same Heroku instance to avoid issues with CORS (Cross-Origin Resource Sharing). We chose Heroku because of its simplicity and cheapness compared to other cloud services.

For the importing of the playlist from each service, there will be an ETL process written in NodeJS that extracts the playlist data from the endpoint they expose. After taking this data, it will transform it into a data format for storage in our PostgreSQL database. Then each track in the playlist will have analytics added to them. Note that this requires us to persist the playlist to the database, and not just send the playlist data to the other streaming service. This also avoids any need to create an ETL procedure for each pair of streaming services. Instead only one format will need to be considered when exporting the playlist to another streaming service.

The exporting of the playlist will be accomplished using another ETL procedure written in NodeJS, that extracts the data from our database, and does any necessary transformations to load it into the streaming services' API. In more detail, the ETL will create a playlist with the specified name in that streaming service. Then ETL will take each track in the playlist on the database, and use an endpoint in the service's API to check if the track exists. If the track exists, the ETL will add it to the playlist in the streaming service using the streaming services' API. If the track does not exist, it will be added to a notification in the end displaying what songs could not be imported.

The analytics of each track will be sourced from AcousticBrainz, a research project that crowdsources the acoustic analysis of songs. This analysis includes data such as danceability, tempo, genre, and mood. Upon the importing of a playlist into our database, there will be a script written in NodeJS that goes through each song in the playlist. The script will check if the analytics for the song already exist in our database. If the analytics already exist, the script will link the analysis to that song. If the analytics do not exist, it will attempt to import the data from the song into our database. There is a possibility that the song is not in the AcousticBrainz database, in which case we cannot provide analytics about that song. If analytics about that song are in the AcousticBrainz database, the script will import them into our database, and link them to the song. The AcousticBrainz project does provide a database dump we could import, however, we decided against that because of the costs associated with storing such a large amount of data.

4.3 Risks

Risk	Risk Reduction
Copyright Lawsuit	Terms and Agreements policy that says we are not

	<p>responsible for any copyright infringement.</p> <p>We do not provide the actual content or host any of it, just move it from one place to another. BitTorrent gets around with a similar strategy.</p>
Privacy of information and linked accounts	Encrypt user data & use OAuth to avoid storing credentials from linked accounts.
Deprecation of Streaming Services' API	There is nothing that can be done to prevent this from happening. However, we can inform users when it occurs, and add support for additional streaming services.

Tasks –

1. Gain background knowledge of all service provider APIs.
2. Design the basic implementation of what API calls we will need to make, as well as how they will feed into our website.
3. Get basic server architecture and database structure running.
4. Program the home page of our application.
5. Make the web page for playlist import and export.
6. Connect to each of the streaming services via OAuth
7. Import playlists from each of the streaming services via their API
8. Export playlists on the website to each of the streaming services via their API
9. Pull information from AcousticBrainz about tracks in playlists
10. If time allows, create functionality to allow for the following and creation of a friends list.

4.5 Schedule

Tasks	Dates	Total Time
1. Background investigation of APIs	01/02/2020 - 01/09/2020	1 week
2. Design of basic API calls	01/09/2020 - 01/23/2020	2 weeks
3. Get basic server architecture & database structure running	01/09/2020 - 01/23/2020	2 weeks

4. Program the home page of the website	01/23/2020 - 01/30/2020	1 week
5. Make the web page for playlist import / export	01/30/2020 - 02/13/2019	2 weeks
6. Add the connection to the streaming services	01/30/2020 - 02/27/2020	4 weeks
7. Import playlists from the streaming services	02/27/2020 - 03/20/2020	3 weeks
8. Export playlists to streaming services	03/20/2020 - 04/10/2020	3 weeks
9. Pull & display analytic information from AcousticBrainz	04/10/2020 - 04/24/2020	2 weeks
10. Add friends list	04/24/2020 - 05/07/2020	2 weeks

Deliverables –

- Design Document: Contains a listing of each major hardware and software component.
- Database scheme and initial data: The DB schema is for a PostGreSQL database.
- Website code: The NodeJS, HTML, and CSS associated with the website and it's functionality
- Final Report

5.0 Key Personnel

Aaron Tucker - Is a senior Computer Science major pursuing a Chinese minor at the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Programming Foundations I and II, Programming Paradigms, and Software Engineering. He was a software engineer for JB Hunt working with REACT NodeJS platform as well as the HTML/CSS languages.

Marc Aranibar - Is a senior Computer Engineer major at the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Programming Foundations I and II, Programming Paradigms, and Software Engineering. He is a professional

YouTuber focusing on building computers and reviewing commercial computer parts.

Rhett Brandon - Is a senior Computer Science major at the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Programming Foundations I and II, Programming Paradigms, and Software Engineering. He is a Database Administrator for Wal-mart, working on automating daily tasks using Ansible, Python and Korn Shell.

Noah Holt - Is a senior Computer Science major pursuing a Philosophy minor at the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Programming Foundations I and II, Programming Paradigms, and Software Engineering. He was a Software Engineer for Wal-mart doing work with security working in the NodeJS and Python languages.

Tyler Gay - Is a senior Computer Science major at the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Programming Foundations I and II, Programming Paradigms, and Software Engineering. He is a Software Engineer for Pivotal, Inc., building a website for integration solutions and tools used internally as well as by technicians in the field.

5.1 Facilities and Equipment

- Subscription services for Spotify, Soundcloud, Pandora, etc. for testing
- Database server
- Application server

6.0 References

Serra, Xavier. "Welcome to AcousticBrainz!" AcousticBrainz, 31 July 2019, acousticbrainz.org/.

Soundiiz. "Transfer Playlists and Favorites between Streaming Services." *Soundiiz*, soundiiz.com/.

"Playlist Converter - Convert & Share Playlists and Transfer Them between Spotify, Napster, Deezer." *Http://Playlistconverter.net/*, playlistconverter.net/.