CS310 Final Report

Harry Verhoef

March 26, 2020

Abstract

This project entails the creation of a mobile application that allows users to democratically elect tracks to be pushed to a queue on a host device. Alongside this, a hybrid recommendation system consisting of genre and artist inference models is developed to recommend tracks to groups of users, which can then be voted for and consequently elected. People in a social setting must typically rely on a single device to control music playback, that is to say it is unifocal, leading to potential conflicts of preference. The application developed [Insert findings from evaluation]

Keywords— Hello, whats up, yolo

Contents

Chapter 1

Definitions of Terms

API
    Native Bridging Module?
    Hybrid Recommendation System

Chapter 2

Introduction

Too often in any situation where multiple people are listening to music from the same source, there is only one person/device controlling what music playback. As a result, many people are subject to music they are not particularly fond of. This project aims to give everyone in this situation a voice in the form of a vote that can be cast suggesting the next track to be played from a set of recommendations. The song with the most votes will be enqueued at the end of the current song. The idea is to introduce the most democratic music environment possible, one where the majority of people listening are happy with what they're listening to. With this in mind, users will also have the ability to up-vote and down-vote a current song. A hybrid recommendation system will be used to shortlist the tracks put up for election, from the universe of all possible tracks that Spotify give access to through their API. A good model in this situation will be one that provides recommendations likely to be voted for by the users in the lobby.

Chapter 3

Review of Existing Solutions

There have been previous attempts at making music playback a democratic process. A good project will review these attempts, and ultimately deduce what can be done to maximise the democracy of music playback.

3.1 Spotify Social Listening

In the June of 2019, Spotify introduced a social listening system [reference] that allows users to join a ”party” and once in, each user has complete control over the party queue. This system is a feature of the Spotify application.

The mechanism by which users can join parties is through activating their on-device camera and scanning a bespoke Spotify barcode as shown in figure []. This is an efficient mechanism that allows users to easily join the party, very little effort is spared. Seeing as the system is a feature of the Spotify app itself, the user-interface will be very familiar to the user and the system, as a result, very easy to use. It’s as if the user is using the regular Spotify application, except that other users have complete control over the queue too.

There are many limitations to this system. For one, all users part of the party must have a Spotify account, meaning that any users without one control the music playback. Moreover, the system is more anarchic than it is democratic. There are no regulations or limitations as to what each user can do, they can all delete tracks off the queue, add as many as they want, change the order completely, etc.

3.2 Festify

Festify [reference] is a web application that allows users to create parties in-browser, then connect to Spotify, and from there users can join the party by entering a party ID. Once in the party, users have the ability to vote for the tracks they like and the most voted for will be played first. Festify relies on users to add songs to the queue (to be put up for voting), the creator of the party must specify fallback playlists which will be played in the event where no tracks have been queued.

The concept of allowing users to vote for tracks is inherently democratic; The track that gets the most votes and is therefore played next is only so as a direct consequence of relative user preference. Users can quickly join parties using the party ID, since it's short and memorable. By the same token, it's easy for users who are already in the party to reveal the party ID to users who aren't.

However, users can vote for an unlimited number of tracks at a time, with different users voting more or less frequently, making it hard to ascertain the value of a single vote. This undoubtedly makes the process less democratic, since some users may vote very generously, and others more seldom, resulting in a bias. Moreover, Festify provide the party creator with the ability to enter "Admin Mode", this is a mode that gives the party creator permission to completely override the decisions made by the group of users. This is completely anti-democratic and self-defeating, it doesn't matter how seemingly popular a track may be among the group of users, if the party creator doesn't favour it they could skip or remove it from the queue. In fact, it can also be argued that the mechanism by which tracks are shortlisted is un-democratic, since it is possible that a small minority of users are the only ones who shortlist the entirety of tracks put up for voting.

3.3 OutLoud

OutLoud [reference] is a mobile application that allows users to create parties by uploading a single playlist. Users can then join this party using a hyperlink. Similar to Festify, users can add and vote for tracks once they're in the party. Users can see explicitly what other users have voted for, and users who add tracks that get plenty of votes are shown on a leaderboard as the "top DJs".

Seeing as OutLoud is a mobile application, available on iOS App Store and the Google Play Store, it is available for a relatively large percentage of the population, and making it more available directly makes it more democratic. Users can down-vote tracks too, making the queue more malleable to user preference and thus more democratic. The principle of uploading one playlist as the base playlist means that

The OutLoud application falls victim to many of the aforementioned pitfalls of Festify. Specifically, there are no limits on the number of tracks the users can vote for or against on, and the shortlisting mechanism can introduce bias. Furthermore, the up-votes aren't anonymous, rendering them as potentially less honest. The hyperlink that allows users to join the party must be sent to new users, and is not a particularly graceful procedure.

3.4 Deductions

A good solution will take the positives from the existing solutions and attempt to omit the negatives. In the context of this project, a positive feature is one that contributes to increasing the level of democracy in the decisions made regarding music playback, and a negative feature functions contrarily. 8 key deductions can be made from the 3 highlighted existing solutions, as to specifically what will help this project be a success.

1. Both barcodes and IDs to be used to allow users to join a party.
2. A similar user-interface to that which the user is already familiar with.
3. An anonymous voting system where users can vote for only one track at a time.
4. Deployment of the application to a variety of platforms.
5. Functionality that allows users to give negative feedback on tracks.
6. A party-joining system that doesn't require third-party authentication.
7. Functionality that ensures no user (even the party creator) has complete control over the queue.
8. An objective shortlisting mechanism that takes into account every user's preferences.

Chapter 4

Methodology

4.1 Choice of Development Methodology

The principle of agile development will be used for the development of this project, namely since one of the fundamental aspects of agile development is a dynamic specification. This is important since the developer has little to no experience in the development of such a project and as a result, requirements will likely need to be adapted throughout the course of development as the developer learns. If the development of the project were to be completed using a waterfall methodology, it is likely that the developer may outline infeasible requirements, which are much more difficult to amend since the whole development procedure would have been planned.

Agile development itslef is a principle that defines a set of methodologies [cs261 slides], the methodology of choice in this case is scrum. Compared to the likes of extreme programming, scrum is a more general agile methodology and one that is perhaps more suited for a standalone developer. Moreover, the developer is much more familiar with scrum development than with any other methodology.

4.2 Requirements Analysis

As part of the scrum development lifecycle, it is important to outline general project requirements. These requirements were compiled as part of the project specification document given to the customer on 09/10/2019.

4.3 Sprint Cycles

4.4

Chapter 5

Design

5.1 Overview

5.2 Native Bridging Module

5.3 REST API

5.4 WebSocket API

5.5 Database

5.6 Recommendation System

5.6.1 Genre Inference Model

5.6.2 Artist Inference Model

5.7 User Interface

Chapter 6

Implementation

technologies used and why...

6.1 Overview

6.2 Native Bridging Module

6.3 REST API

6.4 WebSocket API

6.5 Database

6.6 Recommendation System

6.6.1 Genre Inference Model

6.6.2 Artist Inference Model

6.7 User Interface

Chapter 7

Testing and Results

7.1 Unit Testing

7.2 Integration Testing

7.3 System Testing

7.4 User Acceptance Testing

Chapter 8

Evaluation

8.1 Functional Evaluation

8.2 Non-Functional Evaluation

8.3 Legal, Social and Ethical Evaluation

8.4 Project Management Evaluation

Chapter 9

Conclusions

Chapter 10

Bibliography

Chapter 11

Appendices