UNIVERSITY OF SOUTHAMPTON

Faculty of Physical Sciences and Engineering

A project report submitted for the award of
Bachelor's in Computer Science

Supervisor: David Millard
Examiner: Zehor Belkhatir

Project Audyssey - A Platform for Smooth Multidimensional Journeys Through Song Libraries

by Kathirvelan Arounassalam

April 10, 2025

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING

A project report submitted for the award of Bachelor's in Computer Science

by Kathirvelan Arounassalam

Statement of Originality

- I have read and understood the ECS Academic Integrity information and the University's Academic Integrity Guidance for Students.
- I am aware that failure to act in accordance with the Regulations Governing Academic Integrity may lead to the imposition of penalties which, for the most serious cases, may include termination of programme.
- I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.

We expect you to acknowledge all sources of information (e.g. ideas, algorithms, data) using citations. You must also put quotation marks around any sections of text that you have copied without paraphrasing. If any figures or tables have been taken or modified from another source, you must explain this in the caption and cite the original source.

I have acknowledged all sources, and identified any content taken from elsewhere.

If you have used any code (e.g. open-source code), reference designs, or similar resources that have been produced by anyone else, you must list them in the box below. In the report, you must explain what was used and how it relates to the work you have done.

I have not used any resources produced by anyone else.

You can consult with module teaching staff/demonstrators, but you should not show anyone else your work (this includes uploading your work to publicly-accessible repositories e.g. Github, unless expressly permitted by the module leader), or help them to do theirs. For individual assignments, we expect you to work on your own. For group assignments, we expect that you work only with your allocated group. You must get permission in writing from the module teaching staff before you seek outside assistance, e.g. a proofreading service, and declare it here.

I did all the work myself, or with my allocated group, and have not helped anyone else.

We expect that you have not fabricated, modified or distorted any data, evidence, references, experimental results, or other material used or presented in the report. You must clearly describe your experiments and how the results were obtained, and include all data, source code and/or designs (either in the report, or submitted as a separate file) so that your results could be reproduced.

The material in the report is genuine, and I have included all my data/code/designs.

We expect that you have not previously submitted any part of this work for another assessment. You must get permission in writing from the module teaching staff before re-using any of your previously submitted work for this assessment.

I have not submitted any part of this work for another assessment.

If your work involved research/studies (including surveys) on human participants, their cells or data, or on animals, you must have been granted ethical approval before the work was carried out, and any experiments must have followed these requirements. You must give details of this in the report, and list the ethical approval reference number(s) in the box below.

My work did not involve human participants, their cells or data, or animals.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction (870 words)

1.1 Problem

As technology has advanced over the years, people's personal music collections have become more and more digital [?].

Initially, if a listener wanted to listen to a song or some music, it would have to be performed live, usually by the creator of said music. As such music started to be performed in concerts, where many people could listen to songs from one or more artists.

Radio allowed people to listen to music together and in the comfort of their own home. However, there wasn't full control over what to listen to.

Vinyls and CDs allowed for listening to any song in any place, with the right equipment. This made it possible for people to start treating music as a collectible physical item and build personal music collections. However these physical collections were limited by purchase cost and storage space.

As technology progressed further, we entered the streaming era, where physical collections were replaced by digital collections. These digital collections are significantly cheaper, and are much less constrained by storage space.

As such music collections have the capability to be significantly larger than their physical counterpart. The process of adding songs is much simpler digitally, allowing these digital collections to grow at a significantly faster rate than physical collections.

To navigate the landscape of a personal music collection the predominant method employed by streaming applications is to simply arrange the items in a list. This method works well on a small scale, but falls apart with the large scale that digital music collections can reach.

As such, as a workaround, most users split their mental collections into sub-collections (playlist)s, folders that allow for a lesser mental load and better maintainability.

Whilst playlists help manage the scale of songs, they themselves' have the issue "when a user's playlists become overwhelmingly numerous, streaming services begin to appear inefficient and unmanageable as collection systems"[1].

This hierarchical system allows for the better handling of large scale collections that grow at the pace most collections do **CITE** but if these playlists grow too large, then they require reorganising which can become a chore.

As such the organisational benefit provided by these playlists is lost due to the 'perceived' high mental load required.

Whilst the capability and format of personal music collections have changed dramatically with technological advancements, the structure of these collections has stagnated in the streaming era.

This project will investigate different structures to see how they can improve either of the following key aspects for personal music collections:

- **Complete Knowledge of the Collection** - understanding the entire contents of the collection (to not lose/forget about songs in the collection over time) without exacting a heavy mental load
- **Replayability/Queue Building** - being able to quickly and frictionlessly create song queues to listen to (where the order of the song queue exists on a spectrum between fixed and random)

1.2 Method

First we will try to understand people's mental organisational models for their personal music collections and how they create song queues using their organisational model.

Then we will see the relationship between their mental understanding and their digital music collection.

1.2.1 Complete Knowledge over the Collection

For song organisation, the current method employed by streaming applications is a folder-based list structure (using playlists as folders).

Other methods using graphical spatial methods have been researched

- there has been research into creating 2d and 3d visualisations of song Libraries
- although there hasn't been research into making these usable from a software application point of view

This project will employ three different organisational models:

- **Clustered Table** →
- **Static Graph** → each song is mapped onto a cartesian grid (both 2D and 3D) where the co-ordinates of the song is determined by the numerical attribute for that axis
- **Dynamic Graph** → each song is spaced out from every other song depending on how similar they are to each other, based on both metadata (artist, genre, etc.) and attributes

The aforementioned attributes are mid-level features analysed from the audiofile of each song → these include the instrumentalness, loudness, energy, etc. of a track

1.2.2 Replayability/Queue Building

When listening to songs, the next song to be played is chosen somewhere on a spectrum between fully deterministically (manually selected) or fully non-determinstically (randomly selected):

- fixed deterministic choices occur when the listener knows which song they want to listen to next
- when the listener doesn't know which song to listen to next then the next song should be randomly chosen (such as with Spotify's shuffle feature on a playlist)

With the advent of software now being the medium with which songs are listened to, song queues can now be generated randomly from a set of songs, such as a listener's playlist.

However, there exists no framework for creating song queues where their order is elsewhere on the spectrum than near the two ends. Essentially there is no way to control this randomness without significant overhead (such as having to create a new playlists with the desired songs).

By using the graph-based views as the foundation, this project will investigate a song-queue building algorithm which allows for more of a guided randomness, that is allowing for songs to be randomly selected under the constraints of song metadata and attributes.

This project will use a software application to act as a vehicle for testing and evaluating the aforementioned new organisational structures.

The problem is that I want to be able to see my full collection, not just see it in bits, even if it has been broken down structurally.

Every time i see the collection (rather part of it) my mental experience and idea of it is influenced by what I saw. I want full knowledge over the whole collection, not just vague notion of the parts

When listening to my songs I want to be able to have more movement on the spectrum - fully ordered/active - manually choose songs - guided random - listen randomly within a radius of a song - fully random/passive - just hit shuffle

Main issue is sometimes I want to only listen to a certain region of my collection - if I know the region - either this is already a playlist (issues of creating and maintaining this region) - can draw a region/create radius - if don't know the region - should be able to guide trajectory to song that I'm currently vibing with

Chapter 2

Background

Here we will have a systematic literature review of the prior approaches to this problem.

And some more definitions and explanation

2.1 Graphical Visualisation of Music Collections

2.2 Song Similarity

There has been lots of research into analysing song similarity by performing analysis on the audio itself. This leads to extractable features that can then be used to compare songs to find their **sonic distance**

- low-level →
- mid-level →
- high-level →

2.2.1 EchoNest's Music Database

One of the most prominent cases of this is the EchoNest Music Database.

This is primarily fueled by streaming services need to suggest similar songs for the user to listen to.

Once the user has found the songs the like, they need a way of listening to it easily again.

There does not exist a way to do this in a user-facing way.

This song similarity exists within the proprietary music databases that Big Music don't want you to know/use.

- Exportify's Python Notebooks
-

2.3

Chapter 3

Design

The instrument for testing and evaluation these new models of digital music collection organisation and song queue creation was decided to be a software application.

3.1 Non-Functional Requirements

This software application was designed with the following guidelines in mind:

- **Integration with existing software** → the features in the application should be able to be inserted in existing music streaming applications without issue (such as Spotify, Apple Music, etc.)
- **Desktop only** → to simplify the development process, the application was only developed for use on desktop
- Users can access their library in 3 clicks or less
- All controls should be intuitive and comfortable to use

3.2 Functional Requirements

-

3.3 User Stories

3.4 Activity Network Diagram

3.5 UI Wireframes

As one of the key non-functional requirements is for the developed features to integrate into existing streaming applications, the UI was designed using Spotify's desktop interface as an inspiration.

Chapter 4

Development

Chapter 5

Testing

5.1 Results

Chapter 6

Evaluation

The static graph view helped in maybe actually figuring out what was truly in the collection. Although this could also only be due to the fact that they were sitting down with me to actually look at and discuss their collection

Chapter 7

Project Planning Retrospective (1056 words)

This section will explore in detail the planning and development of the project itself, throughout the stages: research, designing, development, evaluation and the final write-up.

I will go over what were the significant issues with the approach I took. I will then talk about how I would this project were I to start it over again.

Since this project had a very limited timeframe, ensuring that progress remained on schedule was critical.

7.1 What Went Well

One of the main things that significantly helped with planning the project and managing all the things I needed to do was building a hierarchical mindmap in Miro. Miro is a diagramming tool that allows for building any type of diagram. The reason Miro's mindmap was useful was that it allowed for children nodes to be toggled, meaning that they were visually hidden, but still accessible.

This meant that I could decompose the project into its significant core parts. Each section could then be decomposed into its significant parts. This significantly reduced the mental load and allowed for me to better breakdown each aspect of the project as much as I needed to.

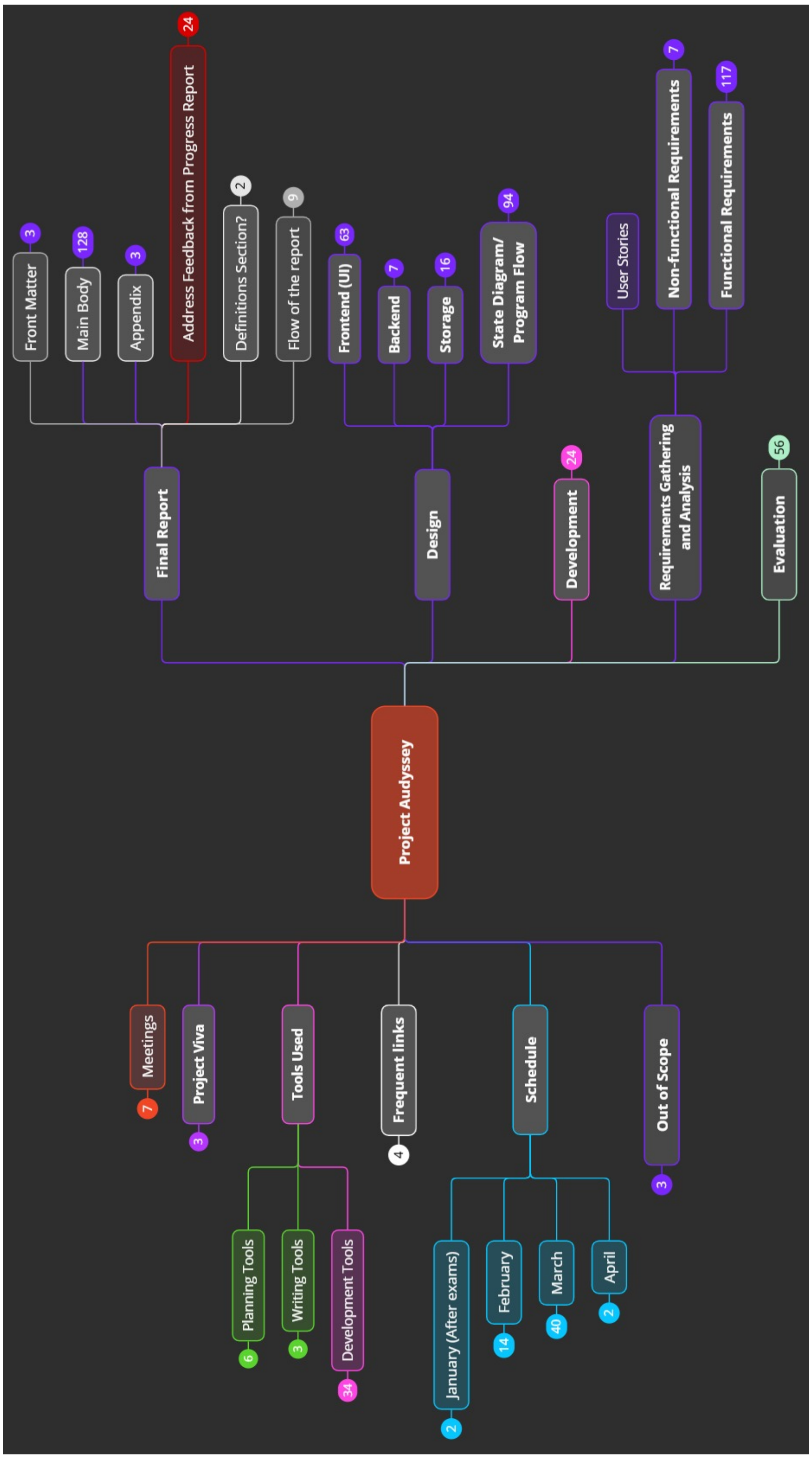

Figure 7.1: The Miro Mindmap Tree only showing nodes to a depth of 1 or 2

7.2 Pitfalls

There were many issues that occurred, some within my control and some due to external sources:

7.2.1 Acquiring Song Attributes for all Songs in a User's Library

[2014] Spotify buys EchoNest - now EchoNest's API is locked behind a Spotify Premium Account (this wasn't an issue due to already having a premium account, but this is a paywall), however this was also a sort of blessing in disguise as it would've meant that I only need to work with one API.

[27 Nov 2024] Spotify announces they are deprecating several endpoints for applications made after the 27th November - these endpoints included both `Get Track's Audio Features` and `Get Track's Audio Analysis` which were key for the project

These audio features (or attributes) were necessary as they would provide the numerical basis for mapping the songs and helping control the audio journeys. To ensure that this didn't derail the project, after much searching an alternative was found, namely the SoundCharts API. This API had an endpoint, that given a track's Spotify ID, would provide the attribute data that Spotify had deprecated access to. However, this data was less accurate (only to 2 decimal places) and was also behind a paywall. For one month, the cost of access for 500,000 API calls at a 30% academic discount amounted to 125 USD. This was within the budget of the project. Initially, the plan was to purchase one month once development had finished. As such the API access would be used only when it was fully needed.

[Late March 2025] Due to significant issues with purchasing the API access using the University's system, another alternative method to gaining the attributes had to be found. This solution was found in Exportify, a web tool built by Pavel Komarov. This tool accesses the Spotify API, including the newly deprecated endpoints, to allow for exporting a spotify user's library to a `.csv` file.

This tool was made before the deprecation announcement and is free to use, making it a very suitable replacement. Futhermore, the attribute values are to the original precision as provided by Spotify. The tool also allowed for exporting of individual playlists, making it easier for my software application to know how one's full collection was composed by the playlists and the catch-all liked songs.

Unfortunately, due to the sudden nature of the Spotify deprecation announcement, an alternative had to be chosen very quickly meaning that Exportify wasn't found until very, very late in development. This meant that the API setup portion of the project took longer than it theoretically could've, as the workflow using Exportify's `.csv` files is much simpler (as shown in the figure NUMBER below). The ideal method would've been to continue searching for alternatives after finding SoundCharts allowing for Exportify to be found sooner and be integrated into the application state flow from the start, however due to the time constraint in needing to submit a project brief, this was not done.

7.2.2 Unfinished Code in the Development Stage

7.2.3 Losing sight of the Goal Only learning and understanding the true goal of the project near the end.

This project whilst in a limited timeframe, was still quite long, hvaing taken the better part of 7 months. During this process the true goal/research questions were only fully understood quite late into the project sadly.

The true/original niche/gap in the research was investigating better ways to create listening journeys, mainly using the graph visualisations as a foundation for controlling and viewing them. This is because, with the advent of the digital streaming era, there has not been any research (none that I could find at least) for creating better tools for users. As discussed in the Evaluation, there is a desire for these tools.

However, during development, as the static graph views were being created, I realised that these were more specifically useful in a data analysis perspective. Unfortunately, data analysis on a user's music library, especially using the EchoNest (now Spotify's) attributes, is something that has been investigated extensively, as mentioned in the background chapter.

In hindsight developing the staic graph views should've been allocated to be done after the dynamic views so that the listening journeys could be accomplished as fast as possible. Static cartesian graphs were intially planned first as I did not realise at the time that they were significantly less useful for interacting with listening journeys than the dynamic view.

One participant in the evaluatory study also mentioned that they liked there being an absolute order to their songs/collection that they could return to. This absolute order synergises better with the dynamic graph, as the static graph produced different distributions for each combination of attributes on the axes.

7.3 Initial vs Final vs Ideal Project Plan

Below are three project plans:

- **Initial Plan** → this is what was planned initially near the start of the project
- **Actual Progress** → this is the actual progress of the project
- **Ideal Plan** → upon retrospection, this is how the project should be approached if to be done again

7.3.1 Initial Plan

7.3.2 Actual Progress

Key Points: - reduce the amount of features - should've only done the dynamic graph as other people have done the static cartesian graphs before - the evaluation itself took ages - either of the below: - do an initial survey of people before design and development -¿ get a better sense of requirements - had a review meeting with the review team on the application - was supposed to do this but hadn't finished enough code in time and they both left the country early

7.3.3 Ideal Plan

Chapter 8

Conclusions

It works

8.1 Future Work

Bibliography

[1] A. N. Hagen, “The playlist experience: Personal playlists in music streaming services,” *Popular Music and Society*, vol. 38, pp. 625 – 645, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:193242204

Appendix A: Photos

This is an appendix

Appendix B: Code Listings

This is an appendix