Semantic Music Discovery Based On Rhythmic Similarity (RhyAn Recommends)

U. Kaveen K. Rodrigo

Date: 2nd May 2018
Department: Computer Science
Key Words: semantic audio, recommendation system

This report is submitted in partial fulfillment of the requirements for the BSc(Hons) Computer Science degree at the University of Westminster.

Declaration

I hereby certify that this project thesis and all it's related material is my own original work done for this degree program.

Student ID

....

Signature Date

Abstract

With the main mode of music listening currently leaning towards streaming and digital media delivery in general, most services provide non-semantic music suggestion/discovery systems to expose users to new listening material. These recommendation systems mainly depend on data mining and metadata. With new music being released by artists everyday, new music may lack the data or have ambiguous metadata thus delaying the recommendation to listeners. Whether the material is new or old, RhyAn Recommends goal is to give recommendations that are aligned with the individual listener's taste.

RhyAn recommends, RhyAn standing for "**Rhy**thm **An**alyzer" counts semantic information extracted from audio signals, in this case rhythmic information. RhyAn is capable of extracting percussive sounds from music which RhyAn recommends uses to give listeners desirable recommendations.

Acknowledgment

This project wouldn't be possible with the encouragement and support of my family and friends, I would like to extend my sincerer gratitude towards them.

I would like to thank my supervisor for advising and giving suggestions during the inception of this project. Also would like to thank other faculty lecturers and members who have helped.

Love goes to my family, who have helped me throughout the years and financially supporting me to study what I love and supporting my decisions.

DEDICATION

To my parents Asanka, Thushari Rodrigo
&
To all who dream

Table of Contents

Illustration Index

Index of Tables

1 Project Background

1.1 Chapter Overview

This chapter will discuss the background of the project and topic to be adressed. Overview of the problem and it's main challenges will be discussed and explored through the authors motivations and scope background for this project. Abridged version of the solution will also be discussed and scoped.

1.2 Project Background

With the mode of music consumption mainly fall into the streaming and online music In the recent years, 37% of the total usage falls to smartphone streaming (International Federation of the Phonographic Industry, 2016)

One of the biggest avenue that has risen is music discovery with the advent of social media based streaming services (Spotify, Apple Music, Deezer, etc.). Most services extract listening patterns through data mining to enhance users recommendation and also employ traditional meta data based filtering to enhance music discovery. Semantic analysis employed in music discovery is fairly low.

With on-demand music streaming services gaining an increase of 39.2% from 2015-2016 (Nielsen, 2016) It's important to provide users with better music discovery methods therefore allowing a hands off experience and allowing users to find music better fit to individual taste and also allow musicians material to be heard; Semantic analysis in music will allow vendors to categorize and suggest music without any in-detail metadata or high volumes of user data.

1.3 Problem Domain

Various online music streaming / purchase outlets have implemented music suggestion systems to help users get recommendations and discover music which mainly is achieved through usage patterns and metadata. With new music rolling out every day some tracks new or obscure may not have enough data or have ambiguous metadata to be recommended to users. These systems base users taste off of listening patterns and metadata but shuns the most important piece of data which is the sonic qualities of the piece of music itself.

Within the audio lies information if harness will allow better music suggestion, which is the very basic idea of semantic music itself. With extracted semantic information such as rhythm, instrumentation, tempo, etc. further enhancements in music suggestion can be archived.

1.4 Motivation

"As an avid music lover and collector a passion inherited from my father, Active listening is an integral part of my life. This love for music has got me digging through crates of records to find new material to listen to. Myself being a computer hobbyist I maintained my own digital collection of music and use different apps and services to manage this collection. One of the key issues that came across is discovering new music, some services either being very predictable and the others being not so accurate. As one of my other hobbies being music making I've gained an understanding in music and the relationship between different genres of music, starting form common instrumentation to specific rhythmic features. These interests lead me to develop RhyAn Recommends in the hope it will solve this issue. As a listener to find new music that touches me and as a musician to be heard. " - **Author**

1.5 Solution

1.5.1 RhyAn Recommends

The proposed solution is **RhyAn** short for ***Rhythm Analyser*** is a music recommendation system that will take leverage of semantics of the music tracks, in this instance rhythmic qualities of music. Extracting rhythmic properties out of music will allow users to be suggested with appropriate music according to the user's taste furthermore allowing the system to classify genre and pace of the piece of music into it's metadata allowing more query possibilities.

1.5.2 Aim

> *"To research, design, implement and evaluate a Semantic Recommendation system for music that analyzes rhythmic patterns in the percussive components of music and compare similarities between patterns to find recommendations"*

To further elaborate on the aim, RhyAn Recommends aims to allow users to discover music that is suited to the users taste by finding tracks that have similar rhythmic tastes without resorting to listening patterns or genre based suggestions. The algorithm is able to classify the current taste of music the listener is interested in as It's not using past metadata to generate an "Overall Taste".

1.5.3 Scope

RhyAn will take a piece of music and extract semantic information, mainly the rhythmic content. This involves the tracking of percussive instruments in a piece of music which will be grouped and stored in a data store. With this data store users can be recommended with recommendations better suited to their tastes.

1.6 Challenges

1.6.1 Rhythmic Analysis

The major challenge is extracting rhythmic information from an audio signal. The process should result in multiple time series that represent the percussive hits of the available percussive sounds, IE – the kick drum, snare drum and hi-hat. Each of the sounds extracted will have a resulting time series that can be used to represent the percussive hits of that specific sound which can be used for the recommendation process.

1.6.2 Recommendation System

As explained in section 1.6.1 the extracted rhythmic contents requires an algorithm to compare the similarity between results of multiple set of processed time series thus allowing the system to give recommendations based on similarity.

1.7 Objectives

Prepare the initiating document with a basic literature review and define the initial scope of the project and draw a basic timeline of how the project will be carried on.

1. Engage in research and prepare the literature interview on the following topics.
    - Researching on current methodologies used to give musical recommendations.
        - Reviewing accuracy and statuses of the said methodologies.
    - Research on the best methods on creating the beat analyzing algorithm.
        - Consider other audio semantic cues that can be used to factor in when find the similarity between two pieces of music.
        - Testing the analyzer accuracy to other solutions.
    - Design an approach for solving the issues based on the Research.
    - On developing an algorithm to see the rhythmic similarity of two processed pieces of music.

- Work on developing an efficient interface to show similarities of the selected music pieces.
- Testing the algorithm's accuracy.
- Work on an intractable and demo-able prototype.

1.8 Solution Overview

1.8.1 Features of the prototype

The basic prototype should be able to process a library of music and then suggest rhythmically similar tracks according to the selected track. The full list of features as follows.

- Ability to manage a local library of music.

  The application should be able to manage a collection of audio files, and use general metadata to categorize.

- Ability track the rhythmic patterns in a library of music.

  The application will be able to take new tracks in the library and process them to extract the rhythmic information.

- To tag rhythmic patterns in the music metadata.

  The extracted rhythmic information will be tagged into the audio metadata / database to be used later for the suggestions part.

- Web interface to play the music library.

  The app will have a web interface to play the music library at ease and have facilities to browse the library.

- Algorithm to suggest rhythmically similar tracks.

  The application will have an algorithm to suggest tracks in the library according to a reference track entered into the algorithm.

- Web interface to show the processed suggestions to the playing track.

  The web interface will guide the user on how to get suggestions and will show the calculated music suggestions from the library.

1.8.2 Rich Picture

Illustrated below is the rich picture diagram of the RhyAn recommends solution and it's different components needed for basic operation.

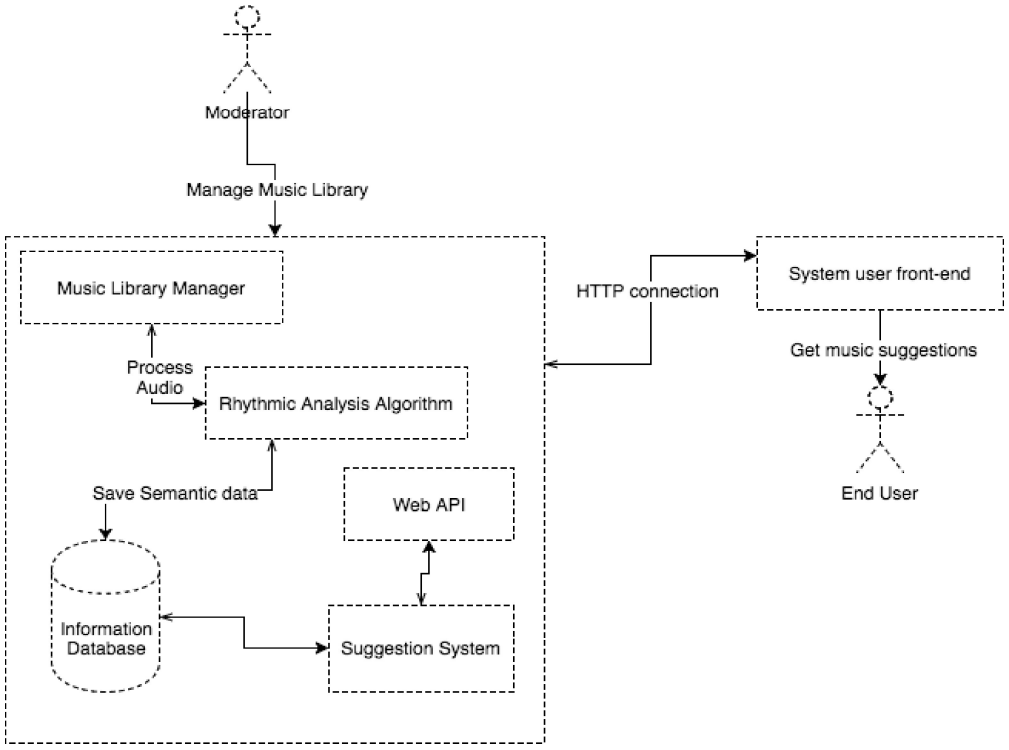


Illustration 1: Rich Picture diagram

1.9 Resource Requirements

Specified below are resource requirements needed for the achievement of this project.

1.9.1 Hardware Requirements

The project will be design and implemented on two main computers, stated below are the optimum requirements.

	Specifications
Personal Computer	Intel i5-6500 x64 based processor
	16GB of physical RAM

1.9.2 Software Requirements

Software requirements listed below account for the implementation of the prototype, test data preparation and applications required for formatting the thesis document.

Program	**Function**
Microsoft Windows 10	Host operating system in which development of the project will take place.
Python 3.6	Programming language used for prototyping.
NodeJS	Used in tandem with yarn to manage all the web related packages.
Visual Studio Code	Code editor for working with both python and JavaScript.
Timidity++	MIDI synthesizer required to prepare test data
Ableton Live 10	Sequencer / DAW used to prepare test data.
LibreOffice	Office suit for working on the thesis document.
StarUML	Draw UML documents

2 Literature Review

2.1 Chapter Overview

As the problem domain and proposed solution was talked about in section 1, this section will dive deep into the current research done in regard of this problem domain and challenges. Literature gathered in this chapter will help aid the design of RhyAn recommend and will give an insight into current progression in the field of semantic music and music recommendation systems in general.

2.2 Review of current music recommendation solutions

2.2.1 Overview

This section will be focusing on various approaches that are used to do music recommendation on overview and research on music recommendation systems showed that there is three main approaches to do recommendation in the music domain.

1. Metadata based.
2. Data mining.
3. Semantic analysis.

2.2.2 Metadata based recommendation

Metadata based recommendation works by leveraging pre-tagged information on the music track, metadata can range from basic information like Artist, Album to other metadata like Genre. With this metadata systems can query for similarities in other track's metadata or used a guided decision tree algorithm to recommend tracks.

2.2.2.1 Metadata Based music recommendation

This approach uses the song metadata like artist, year, genre, etc. to find similar music of the categories. For example, if someone listens to rock msuic the system recommends a song from the same genre. (Van den Oord, Dieleman, & Schrauwen, 2013) Following is an oversimplified example of this approach.

- User A listens to "Donovan" in his music collection and is looking for a recommendation.
- The "Donovan" track is classified(metadata) under Contemporary Folk so is another track in the library under the artist "Bob Dylan".
- User A gets recommended with the track under the same genre meta-tag.

2.2.2.2 Limitations

One of the major limitations in metadata based recommendations is metadata ambiguity, Where some genre's or other attributes maybe similar to another genre but the system will not consider those entries in the system. Unless tediously relationships between genres is defined in system, the user will not be indulged in new genre's with similar qualities of the music the user is used to.

2.2.3 Data mining

This approach uses the collaborative data of user's listening patterns to do music recommendation and to classify with additional meta tags. This method is highly used by different vendors since It yields the best results without any pre-tagged data. Even though with a small user base and minimal data it may not be effective but with large amounts of users it's deemed effective.

2.2.3.1 Collaborative Filtering

With the metadata of the music and other user's listening record, most services use that to give recommendations to other users based on others listening patterns using methods like neighborhood-based or model-based (Ricci, Rokach, & Shapira, 2015) similar interests in music can be found, different users. Following is an over simplified example of this approach in a music service.

- User A listens to "R.E.M" and "The Doors"
- User B listens to The Doors and is looking at the recommendations
- Based on user A's listening patterns "The Doors" is recommended to user B

Most popular music suggestion systems such as last.fm are popular users of this method by using the last.fm scrobbler and to use social media ties (Blogs.cornell.edu, n.d.)

One of the key-points on the success of this methodology is the availability of data for recommendation as a certain percentage of users are needed for each and every new music track being released every day, therefore new material may take time to be recommenced naturally to some users.

2.2.3.2 Limitations

Data mining based systems may eliminate problems in traditional metadata based system, as the recommendations are mined through actual user information thus making the recommendations much more human where new users maybe introduced to new music that they were not previously introduced into as the metadata didn't have any collation.

But the major limitation of these types of systems is the need for data, as the user base should be wide enough for all the users to be interacting with different genres of music. With new music being released each and every day, sometimes users may not be introduced to new material fast enough.

2.2.4 Semantic analysis

Rather than using metadata or user data mining, semantic analysis deals with the most valuable data yet the audio track it self. With DSP[1] systems can extract valuable data like the musical key, rhythm, instrumentation, etc. and using this information the system is able to classify and recommend music to users according to their individual taste.

The common belief of teaching the machine to listen may redeem the problem of this information overload as the computer will be able to listen to the piece of music and analyze It's properties to present it as a suggestion to the user, thus eliminating the problem talked about in section 2.2.3.1.

2.2.4.1 Music Mood Classification

This method factors in multiple features in music and extracts them to be trained in a SVR[2] classifier it considers factors like pitch, tempo, loudness, tonality, etc and the SVR based mood classifier is able to classify the mood of different types of music inputted into the system. (Padial & Goel, n.d.)

This method is popular in personal music programs that gives a mood box that allows the user to select a mood and the system will create a playlist with the mood classifier applied. This implementation can be also seen in popular mobile music managers as it allows quick playlist making using unclassified set of audio signals.

1 Digital Signal Processing
2 Support Vector Regression

2.2.4.2 Feature Kernels

Different set of semantic feature are combined to train a SVM machine learning models with different tags IE – Rock, Pop, Blues. (Barrington, Yazdani, Turnbull, & Lanckriet, 2008)

This approach uses multiple well known semantic descriptors and DSP methods such as MFCC[3]s which can be used as an audio timbre descriptor, Chroma features which when analyzed in short windows in a certain signal can represent the harmonic content of the signal.

These methods trained with a certain data sets that have samples for each genre will allow this algorithm to tag music with a set of descriptors that can be used for giving suggestions.

2.2.5 Why Semantic Analysis?

With new music coming out each and every day, there is a certain problem in data mined methods where there is not enough data for these new tracks as It hasn't been heard by much users, this can be solved if computers were given the capability to listen to music and associate similarities to existing music. Instead of reading metadata with conflicting tags, extracting information from the audio track itself is the way to go.

2.2.6 Limitations

Main limitations may spring up from the different methods being used to for semantic music systems, The semantic descriptors they extract, accuracy and many other variables will be causes for some of the limitations.

A hybrid approach using Metadata, Data Mining and Semantic Analysis can be used to overcome some of the limitations as data from other users can be used to correct the errors in the recommendation.

3 Mel-frequency cepstral coefficients

2.3 Review on current rhythm-tracking methodologies.

2.3.1 Overview

Rhythm in basic terms is arranged sounds moving through time in a pattern. Rhythm is commonly associated with percussive instruments like drums which throughout a piece of music keeps the band and the listener in time / tapping their foot. Watered down basic introduction to tracking rhythm algorithmic way is to extract the on-sets in the audio signal. As shown in Illustration 2, the red lines show the on-set's of this particular audio stem which can be identified by the difference of amplitude exceeding a certain threshold therefore being classified as an on-set.

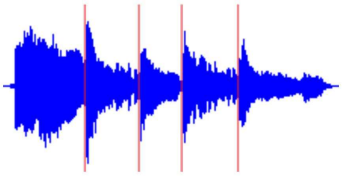


Illustration 2: On set in audio signals

2.3.2 Issue with rhythm-tracking

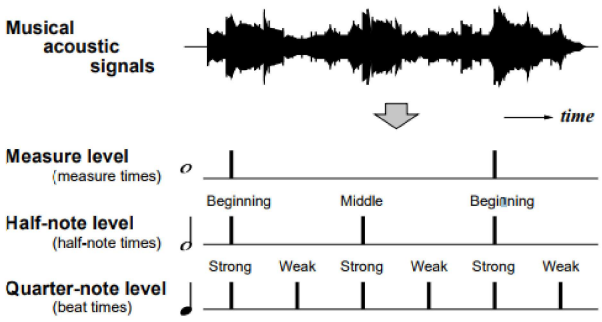


Illustration 3: rhythym-tracking problem (Yoshii, Komatani, Ogata, Okuno, & Goto, n.d.)

In a musical concept the problem is the detection of musical measure level, half notes and quarter notes in a musical context in digital MIDI audio this is not an issue since MIDI audio is the equivalent of digital sheet music, finding MIDI files for each song is not a viable option.

The inputted audio waveform has to be converted into rhythm descriptors which is used to measure level, halftime notes and quarter note beat times. Most beat tracking methodologies work on the concept of onset detection where the algorithm detects the attack of the waveform refer *Illustration 3.*

2.3.3 The Beat Spectrum

Beat spectrum is a rhythm-tracking methodology that doesn't fall when it comes to other methods since beat spectrum can identify the rhythm in a piece of music that doesn't contain any percussive instruments, for example, a piano instrumental. The algorithm works by first characterizing the input audio using the distance-matrix approach which is followed by calculating the difference between each of the characterized frame and then using distance matrix embedding allowing to derive the beat spectrum (Foote & Uchihashi, 2001) which can be used to get the rhythmic pattern of the song..

2.3.4 Beat tracking with hierarchical understanding of beat structure

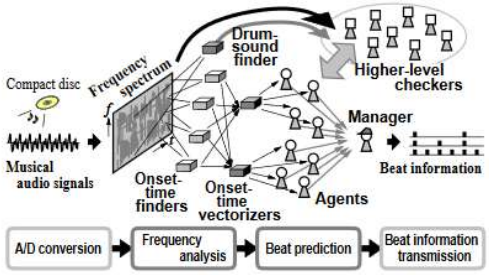


Illustration 4: Overview of Masataka Goto's solution (Goto, 2001)

Compared to the previous beat tracking methodologies discussed in this review, this method understands real life musical beat structure which understands quarter note level, half note level, and the measured level.

This method uses FTT[4] analysis and calculating Onset-time vectors on different frequencies by using a set of filters. For beat detection the algorithm uses onset detection, again to figure out the bass drum and the snare. (Goto, 2001)

This method filters down he audio signal to multiple bands and analyses the On-Set patterns of each and every band and grouping the available sections with the aid of a drum sound classifier which uses basic descriptors of sound IE – hi-hats are noisy, with this classifications it's able to take the on-set profiles and classify the basic drum sounds this is explained well with illustration 4 taken from the author's paper.

4 Fast Fourier transform

2.3.5 Rhythm tracking using drum loop patterns

This method approaches music as same as many other methods by detecting the Onset in the audio waveform then sends the onset data to two methods.

One method will be doing "Meter Structure Estimate" where the profile of the onset is extracted based on the energy significance the second method takes the same onset data filters out the noise and clusters and categorizes the drum sounds to bass and snare drums based on characteristics like the noisy nature of the snare drum. (Bello et al., 2005)

2.4 Review on time series similarity algorithms

2.4.1 Overview

He project hinges on a good time-series comparison algorithm as some of the fundamental and main tasks require measuring similarity between time-series.

Unlike using a distance method like euclidean distance won't satisfy this question as the time factor isn't consistence and is allowed to warp.

2.4.2 Dynamic Time Warping algorithm.

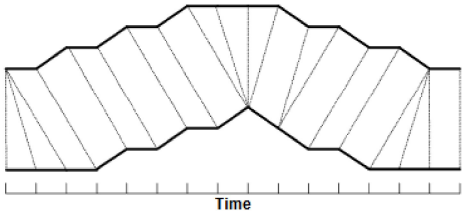


Illustration 5: Dynamic Time Warping between multiple time series (Salvador & Chan, 2007)

Dynamic Time Warping is an algorithm that is used to measure distances between two time series.

This differs from traditional distance algorithm as in a time-series context the distance depends on the time domain where each time frame maybe offsetted by a certain number of seconds but is still a similar distance.

As illustrated in Illustration 5 the two time series distance is measured overthought the time domain is not exact. the algorithm matches each change and measures the distance (Salvador & Chan, 2007)

3 Project Management

3.1 Chapter Overview

This chapter will discuss the project management methods and decisions taken to gather requirements, design and develop this project into life in a certain time frame. Project pan will be discussed regarding allocation of time for the required steps.

3.2 Research Methodology used

3.2.1 Overview on Research Methodologies

Two of the main research methodologies is use is inductive and deductive research, These can be narrowed down using a simple question “is the observation/ theory a result of the research”, if the statement is true it can be classified as inductive research since the observation or theory is a direct result of the research done. Deductive research is the research done to enforce or test a certain theory.

3.2.2 Inductive Research

Inductive research involves the development of a new idea/theory with the help of existing and new research. Where the author will be looking at existing research done in the hope of finding a new methodology or theory to approach a problem.

3.2.3 Deductive Research

Deductive research methodology involves researching a certain topic to prove a certain predefined theory. Where the author will be researching material to further enforce the truthfulness / falsehood of a theory.

3.2.4 Justification on selecting deductive research

For this project the research methodology used is inductive research. An Inductive research method is used in scenarios where the observation or theories are developed as result of the observations.

In this project research will be done on major rhythmic analysis algorithms and recommendation systems to allow developing a new algorithm for the specific aim.

3.3 Design/Development Process

3.3.1 Overview

This section will talk about the execution of this project during the total time span of the project particularly in the deign/ development process of the prototype.

3.3.2 Overview on a couple of methodologies

This section will discuss some of the more popular design and development processes used in the engineering field.

- Waterfall model

  Waterfall model is initially adopted for manufacturing and construction industries where the linear flow of this model is followed and the model doesn't allow changes after one section has been done.

- Iterative and incremental development

  This model was created as a followup to waterfall model, where the waterfall method lacked in re-iterating a step this model allowed re-iteration.

3.3.3 Reasoning for selecting Iterative and incremental development

Unlike the waterfall model, the Iterative development model allowed revisiting previous steps thus allowing the project to be done in portions. Therefore allowing implementation of the project in stages where different features can be designed, developed and tested in one go.

3.4 Data Gathering Methodologies.

3.4.1 Audio gathering

The main data that is required to test this solution is music in .mp3 form. For this purpose a set amount of songs spanning from different genres will be selected, therefore ensuring that the algorithm will work in different types of music.

3.5 Project Plan

3.5.1 Work Breakdown Structure

- Project Initiation
    - Select research domain
    - Decide on project
    - Draft PID
    - Submit PID
    - Submit Ethics Questionnaire
- Literature Review
    - Gather literature for topic
    - Prepare literature review
    - Gathering Requirements
    - Prepare software requirements
    - Submit SRS
- Prototype Implementation
    - Designing the prototype
    - Initial implementation.
    - Preparing the interim progress report
    - Submitting the interim progress report
- Finalizing prototype
    - Finalizing program
- Testing & Evaluation
    - Testing with synthesized audio samples
    - Test each algorithms.
    - Documenting test results.
- Finalizing the project
    - Prepare draft
    - Submission of the draft
    - Prepare final report
    - Submission of the final report

3.5.2 Gannt chart

Gantt Chart available in appendix A.B

4 System Requirement Specification

4.1 Chapter Overview

This chapter will discuss the requirement gathering and specification process of the project with the aid of stakeholder and user role analysis. Further requirements will be gathered by adopting a certain requirement elicitation method,

4.2 Stakeholder analysis

Stakeholder analysis for RhyAn in this section is done in the perspective of this solution being a commercial product.

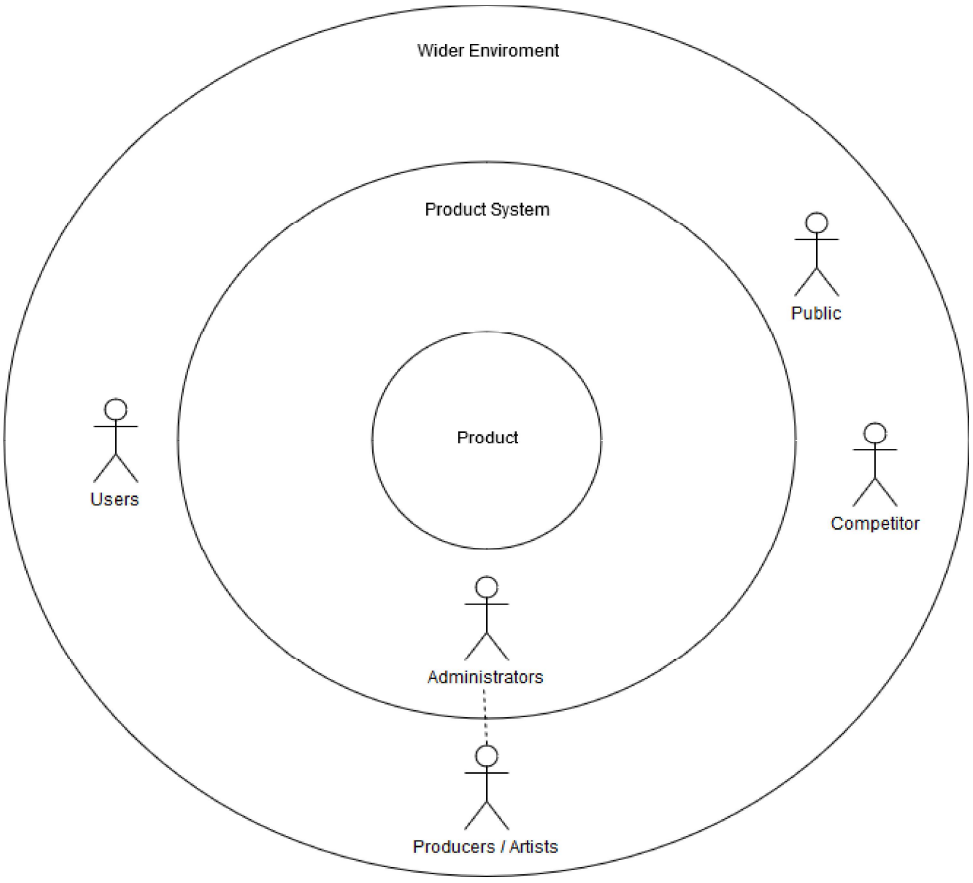


Illustration 6: onion model for stakeholder analysis

4.2.1 Stakeholder descriptions

The following table will explain the stake roles and their viewpoints illustrated in Illustration 6 onion model. The viewpoint will give a brief description appose to the stake role.

Stake role	Stakeholder	View Point
Beneficiary	User	The user is able to find and discover new music based on his/her individual taste.
	Producers/Artists	Producers or Artists are enabling their music to be discovered by potential customers.
Operational Roles	Administrator	Manages the core system and it's media library
	User	Uses the system to discover music that is rhythmically similar.
Regulatory	Public	People interested in the service
Negative Stakeholders	Competitors	Who wants to create a competitive system.

4.3 Requirement Elicitation

4.3.1 Overview on requirement elicitation methodologies

- Questionnaire

  Questioners can be used to gather requirements from a certain research group by using structured questions that question about there current thoughts on the research topic.

- Interviewing

  Interviewing an industry expert can be helpful as with an interview and discussion enables to find out the current state on the research topic.

- Brainstorming

  Brainstorming allows the research student to take in certain variables and use techniques like role playing in different scenarios to gather requirements.

4.3.2 Requirement elicitation methodology

The bulk of the requirements for this projects is gathered through **Brain Storming** with role-playing of the major actors of the system in different scenarios**.**

At a time a major actor of the system is selected and role played to identify the requirements that's required for the actor, the advantage of using this is system is the ability to identify all major functional requirements.

4.3.3 Justification for selected method

Justification for this selected method compared to this other methods is ability to role-play an identify the bare minimum requirements for the system to function compared to other methods where manual prioritization is required to filter out the functional/non-functional requirements.

4.4 Use case model

4.4.1 Use case diagram

The diagram below describes the key actors in the system and the interactions that will take place in the system thus defining the main abstract functions of the system.

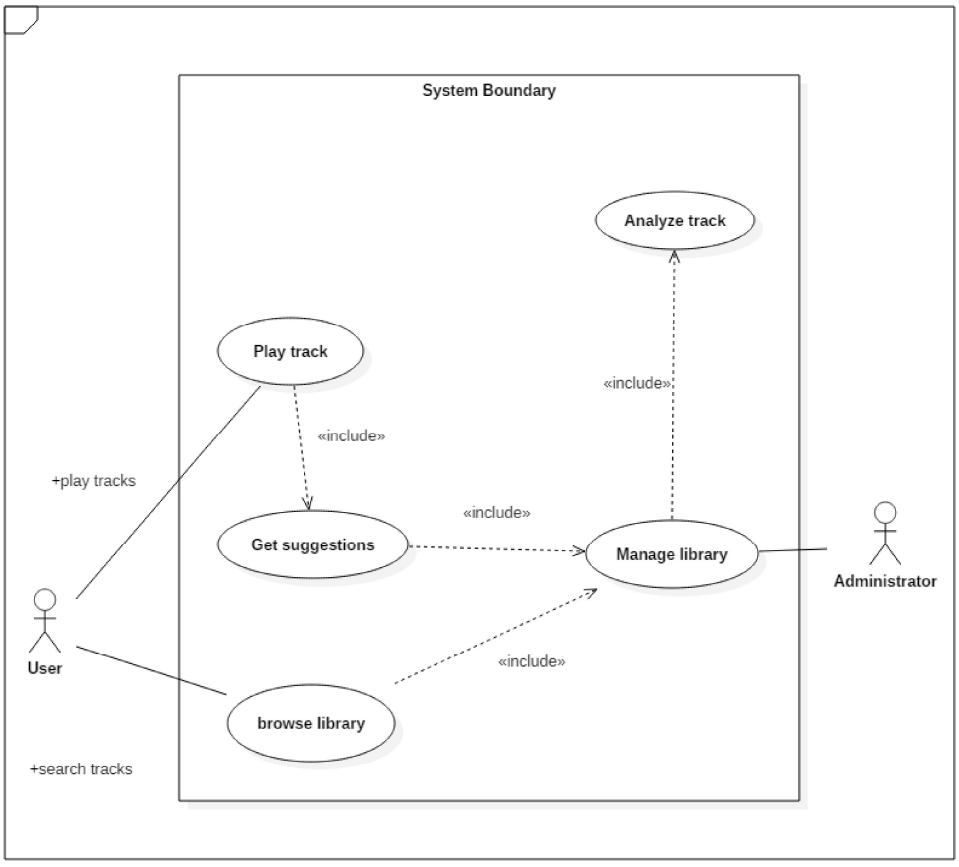


Illustration 7: Use case diagram

4.4.2 Use case description

Use case descriptions can be found in appendix A.A

4.5 Requirement Specification

4.5.1 Section Overview

The requirements defined in this section is the result of the other methods that were used to gather requirements such as the stakeholder analysis and the User case model.

4.5.2 Requirement prioritization index

The prioritization of the functional/non-functional requirements are classified with the terms given below.

1. **Essential** - The requirements needed to complete this project.
2. **Desirable** - Requirements that are not an immediate need for the operation of the project, but is valuable for the project.
3. **Luxury** - Requirements that will add extra value that is not required for the core functions of the system but will be a luxury.

4.5.3 Functional requirements

The table below describes the requirements required for the basic functionality of the system with relation to their priorities.

Id	Priority	Requirement
FR1	Essential	Browse Media User should be able to browse through the collection of audio.
FR2	Essential	Analyze Track Ability to take an audio file and process it's rhythmic qualities.
FR3	Essential	Give Recommendations Ability to give the user music recommendations based on a reference track.
FR4	Essential	Calculate Similarity Between Tracks Ability to take 2 tracks / sections and calculate the similarity
FR4	Essential	Search Tracks Ability to search and filter the music library.
FR5	Essential	Ability To Play Tracks Online Ability to play selected music tracks from the interface.
FR6	Luxury	Graphical Visualizer Graphically Visualize the extracted rhythmic information.

4.5.4 Non-functional requirements

The table below describes the non functional requirements of the system, requirements stated below are non-essential to the implementation of the system but is desirable for It's overall quality of operation and results.

Id	Priority	Requirement
NFR0	Essential	*Accuracy* The algorithms ability to acculturate track the rhythmic patterns of the audio signals.
NFR1	Desirable	*Performance* The algorithms ability to analyze a complete track in the fastest time possible, but since this system does not calculate this in real time it's not necessary.
NFR2	Essential	*Usability* The usability of the system front-end should be easy to use.

5 System Design

5.1 Chapter Overview

In this section of the document, design of the RhyAn system will be talked about with high level abstraction. Design decisions taken in this chapter will be used for the implementation of the RhyAn system.

5.2 Design Goals

Design goals of RhyAn Recommends is listed below. These goals will be taken to count in the design process.

Goal	Description
Modular Structure	Ability to swap different sub components or not use some components as necessary. This can be solved by clearly packaging out the solution and structuring the project.
Performance	Some Algorithms are required to run real time to give suggestions, therefore time factor should be considered.
Scalability	The system should be able to handle and process regardless of the size of the input data.

5.3 High Level Architecture

Following is the high level architecture diagram, which shows the major components of RhyAn Recommends.

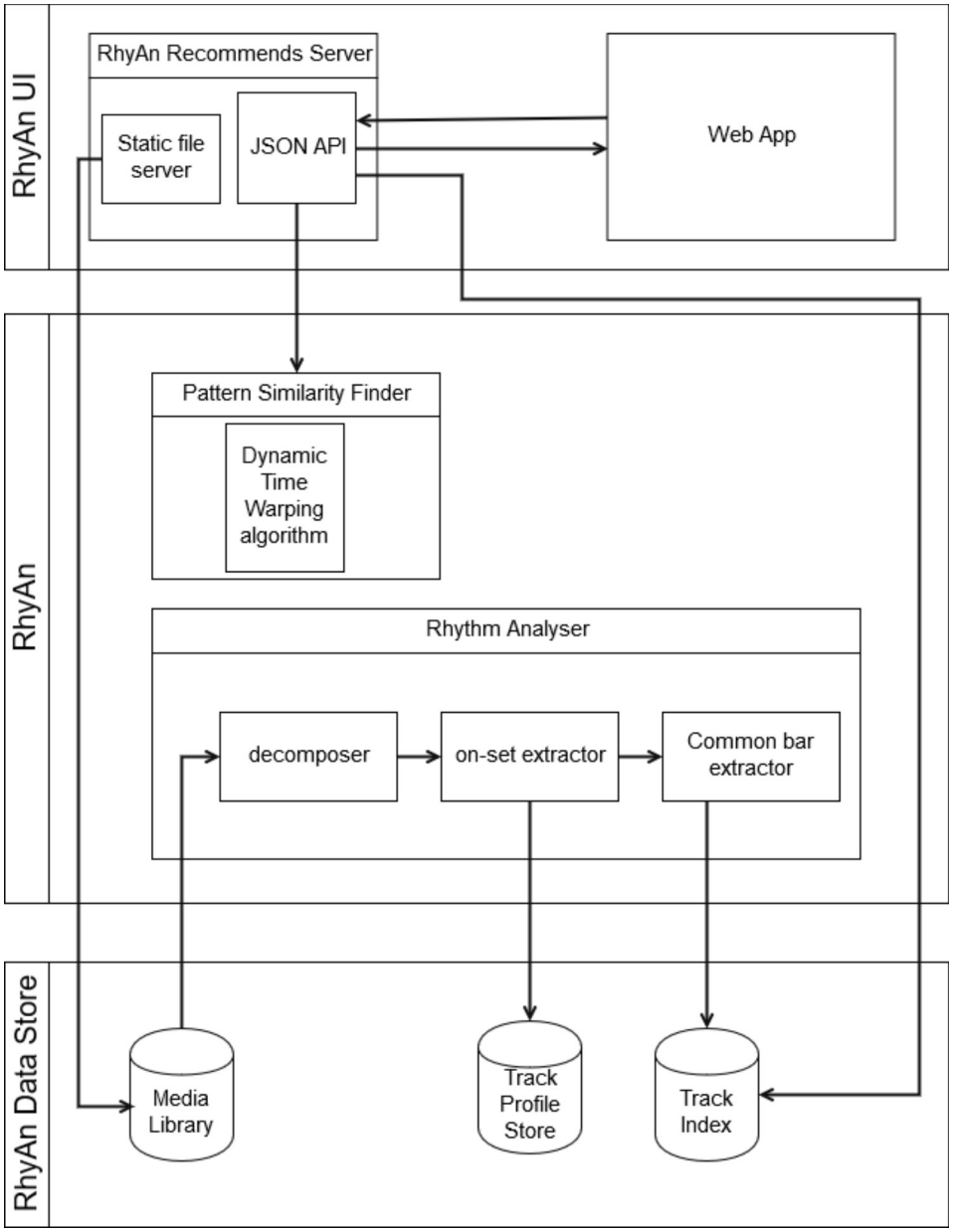


Illustration 8: High Level Architecture

5.4 Design of RhyAn

As the name suggest RhyAn standing for **Rhy**thm **An**alyzer, this component will be the heart of the RhyAn ecosystem as illustrated in illustration 8 this component will be used in-conjunction with the RhyAn UI to give music suggestions.

Step by step the rhythm analyzer will transform an audio signal into an excerpt of a drum pattern. Thus allowing different music to be analyzed for rhythmic similarity.

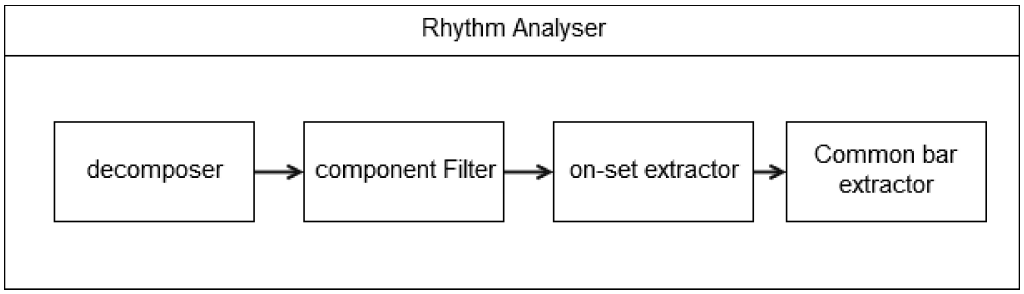


Illustration 9: High Level Architecture of RhyAn

As illustrated about in illustration 9 the system will process the audio signal, a rough idea of the process is as follows.

1. **Decomposer** – Takes input audio signal eliminates melodic components and splits the audio into individual percussion tracks.
2. **Component Filter** – Further refines the decomposed components by filtering out unwanted frequency bands.
3. **On-Set Extractor** – Takes filtered audio signals and converts them to time series which can be mathematically processed.
4. **Common bar extractor** – Identifies the most common drum pattern in the track which will be stored and cross checked to find similarities.

5.4.1 Decomposer

The task of the decomposer is two take an audio signal and extract different drum components of the audio signal. This is accomplished by a two step process to the input audio signal, which first involves running through HPSS algorithm to get rid of the melodic components of the signal and then decomposing the feature matrix with NMF (non-negative matrix factorization) as shown in Illustration 10

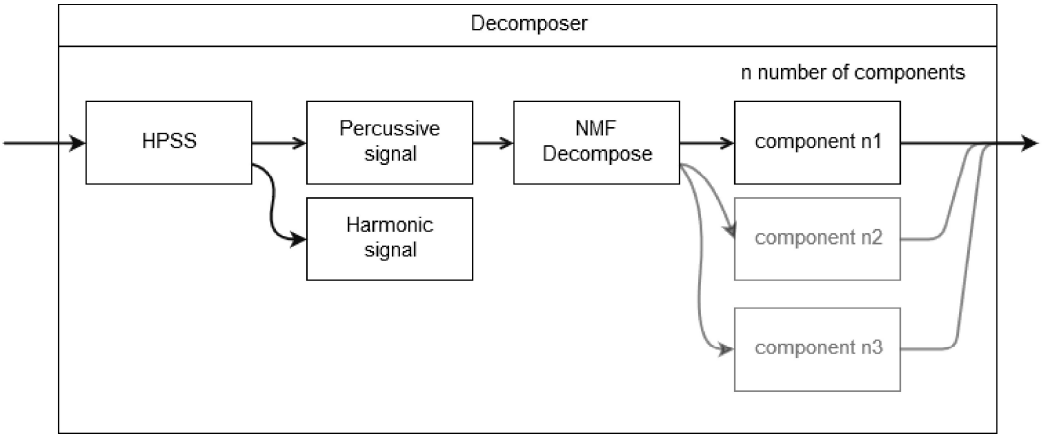


Illustration 10: Decomposer module

5.4.1.1 HPSS

HPSS stands for harmonic percussive source separation which uses median filtering has the ability to split the percussive and harmonic components of the music signal. Using median filtering different frequency bins to eliminate the melodic components of the signal for this application. (Fitzgerald, 2010)

5.4.1.2 NMF

NMF[5] is the final main step that will make the pre HPSS processed signal. Which will decompose the audio signal into a select number of components.

5 non-negative matrix factorization

5.4.2 Component Filter

After the decomposition step there might be bleed audio from the other components, for example some of the kick drum hit sounds might be bled on the snare track, to eliminate this problem. This module will take each component and find it's fundamental frequency which is the most prominent frequency in that component and then perform a band pass filter on that channel as shown in illustration 11.

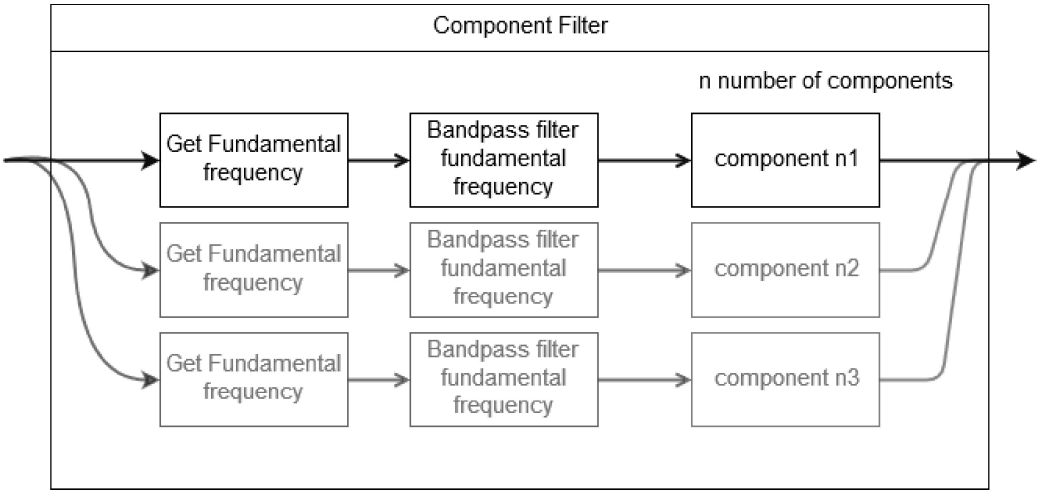


Illustration 11: Component Filter

5.4.3 On-Set Extractor

This module has the basic but complex task of taking the filtered components and using the On-Set energies of the signals to extract the time frames from the signal. Refer illustration 2 for a graphical represent of this process. After the on-set extraction is done further filtering is done to clean up the unwanted on-set's these are mainly done with the help of the amplitudes of the on-set points.

After the clean up process of each individual time series extracted with the help of the On-Set algorithm and then the time-series are merged to remove duplicate signals as shown in illustration 12.

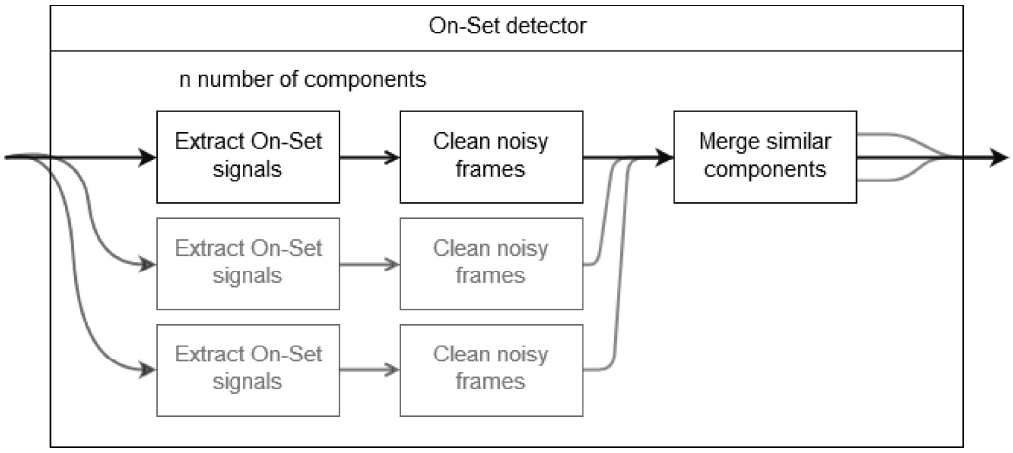


Illustration 12: On-Set Extractor

5.4.4 Common Bar Extractor

After the whole track has been extracted with the On-Set Extractor, It's time to analyze the most common bar, which is the most common section of the piece of the track. The main algorithm that aids this module to function is DTW (Dynamic Time Warping) which I've discussed in the literature review section, In simple terms this function allows to measure similarity between two time series. Which will be used to find the most common section of the track.

5.5 Recommender - Pattern Similarity finder

After each of the common bars are extracted and indexed on the database, RhyAn will again use DTW (refer chapter 2.4.2) one to find the distance between each component and give the average distance.

These time-series will be normalized to a common scale, therefore during the recommendation extra tempo based distance won't affect the score.

5.6 Design Of RhyAn UI

Rhyan UI layer will not only have the interface but the web server that will be running the interface and communicating with the database to give suggestions.

5.6.1 Web API

The web api will have these basic functionalities, that will allow the web interface to search through tracks and get recommendations.

1. List available music
2. Get information on selected track
3. Suggest tracks based on the selected track.

5.6.1.1 Suggestions Criteria

The Web API will contain the suggestion algorithm which upon request will give the closest related tracks to a reference. The following rules will be followed to prepare and filter the suggestions.

- Closest Distance
- BPM should be in the similar ranges
- Sorted by BPM

5.6.2 Web Application

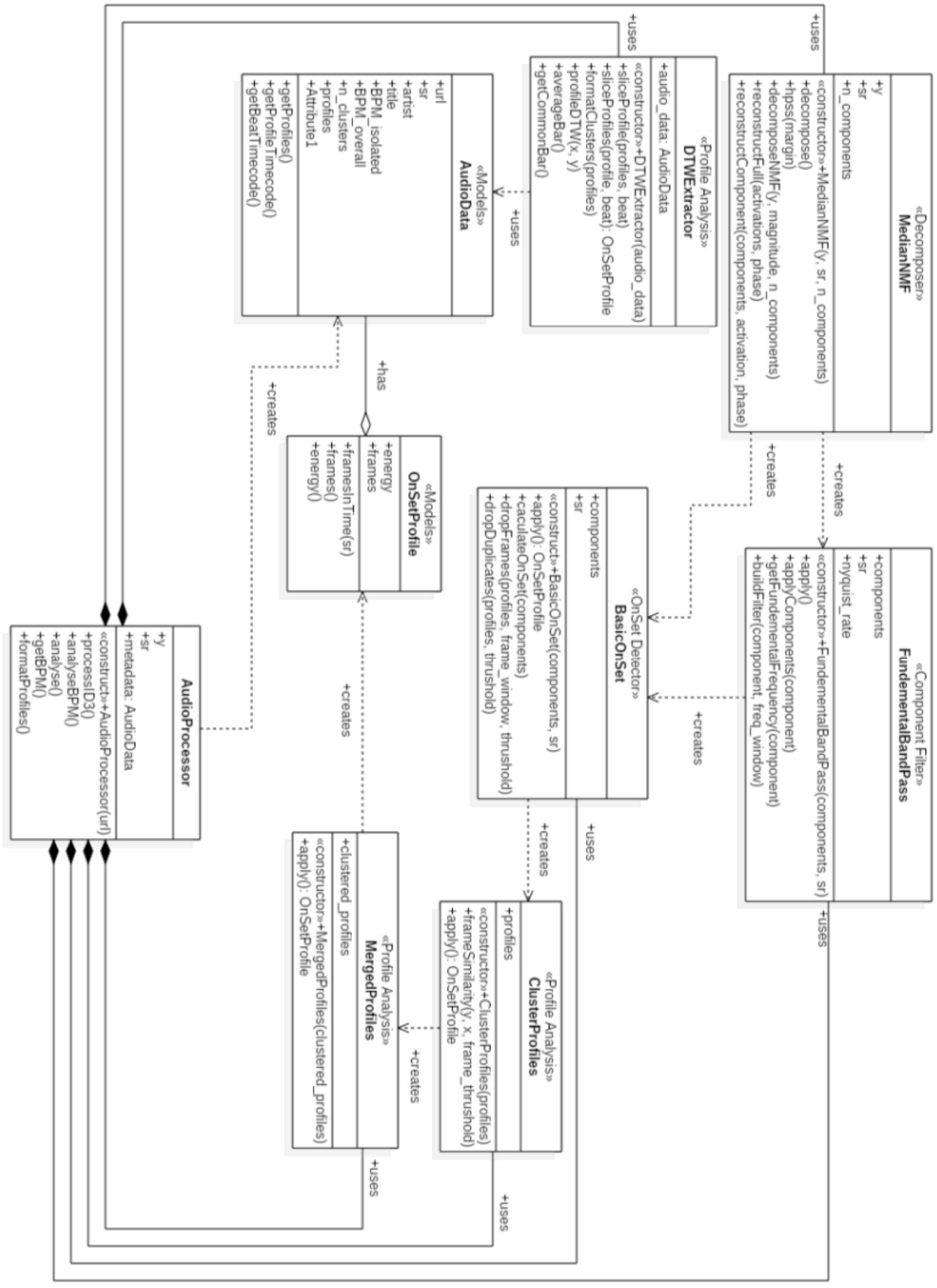
The web application will take leverage of the aforementioned Web API and provide an interface for the user. The application can be accessed through a web browser to browse the media library and find recommendations.

The application will also provide a visualized view of the drum patterns extracted from RhyAn, this is mainly for debugging uses and to provide a view of the functionality of the system.

5.7 Class Diagram

As designed in the higher level architecture and chapter 5.4 regarding the design of RhyAn the following class diagram was designed.

5.7.1 RhyAn class diagram



5.7.2 RhyAn UI class diagram

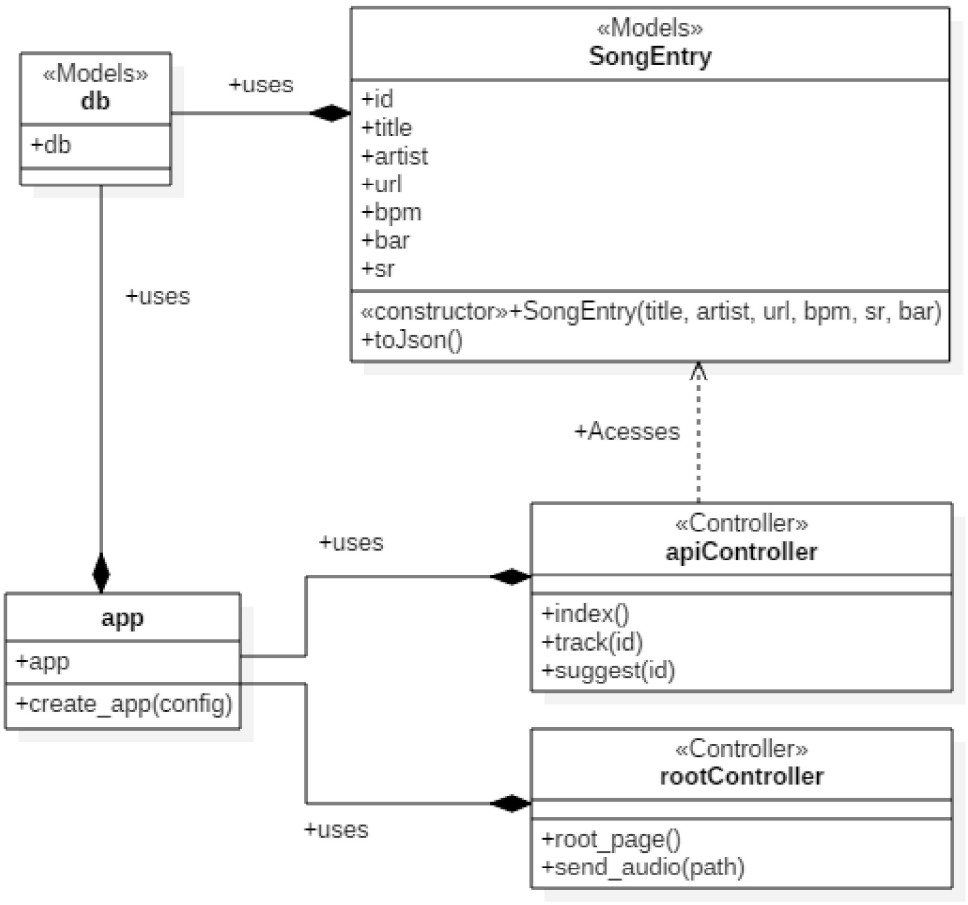


Illustration 13: RhyAn UI class diagram

6 Implementation

6.1 Chapter Overview

With the help of the literature review done in chapter 2 and design done in chapter 5 suitable technologies and methods will be selected and implemented.

6.2 Technological Selection

6.2.1 Programming Languages

RhyAn has multiple components in it's design most serving signal processing tasks but the others deal with running the web API and handling a database with indexing information, plus point for selecting a suitable language would be the ability to use the same language with good library support for the many functions of RhyAn.

6.2.1.1 Available languages

Listed below are a selected few languages that are well known and suited for this task, pros and cons are discussed for the language and for this specific application.

Name	Pros	Cons
Matlab	• Well known in the scientific community for DSP prototyping. • Toolkit is supplied for DSP work.	• Not able to transform implementation into a consumer application easily. • Author is unfamiliar with the language.
C++	• High performance. • Object oriented.	• Complex DSP libraries. • A lot of ground work in necessary.
Python	• Object Oriented • Well documented libraries for DSP and number crunching in general. • Ability to run a web stack at ease with certain libraries. • Author is familiar an is versed in the languages and it's features.	• Performance is not as good in DSP applications compared to C++

6.2.1.2 Selected languages for RhyAN

Listed below are the languages selected for RhyAn based on the table of pros and cons in the previous section. HTML and JavaScript is a granted for the web interface.

Name	Purpose
Python 3.6	The main program will be written in python which will manage the library and will hand the audio processing. Using Python the web API will be implemented which will be used with the web front-end.
HTML / JavaScript	The front-end of the application will be developed with HTML and JS therefore the application will be cross platform and will work across multiple devices.

6.2.1.3 Libraries and Frameworks

Libraries used for RhyAn are used to do signal processing and host the web-process.

Component	Name	Purpose
DSP component	Librosa	Librosa is a DSP[6] library with common DSP functions, the library is also has good documentation online. Librosa also uses Numpy and Scipy under the hood.
	Scipy stack	The scipy stack will be used to do the digital signal processing required for this project.
	eyed3	For extracting metadata from input MP# audio.
Web Component	Flask	Flask is a web micro service platform for python which will be used to expose the web API for the front end.
	Flask-SQLAlchemy	SQLAlchemy is ORM[7] for many database applications which will map objects to the database.
	React.js	React.js will be used as the front-end framework to create a single-page-app that will allow the users to interact with the system.

6 Digital Signal Processing
7 Object Relation Mapping

6.2.2 Choice of database / data store

With the driver agnostic nature of SQLAlchemy, most database solutions can be used with a simple driver installation, therefore testing purposes SQLite is used as the driver is in-built to Python.

As for the processed data from the RhyAn is saved as JSON files to be reused for any other processing.

6.3 RhyAn implementation

RhyAn is the main module of the project which will handle all the DSP related processes. It's implemented in a library like structure which allows it to be easily imported and use.

6.3.1 Decomposer

Decomposer sub-module class has two main functions HPSS and the NMF functions. The class can be called in with the audio signal and the number of components.

When instantiated with an audio signal and sampling rate, the class will be created with which the decompose method can be called to return individual decomposed components.

6.3.1.1 Class usage

After importing RhyAn package and librosa package for loading the audio file , decomposition can be done easily by creating an instance of MedianNMF and passing in the audio file proceeded by calling the decompose function which will return an array of audio signals.

```
import librosa
import RhyAn.decomposer as rhy_decomposer

audio_path = "./sample_audio/busyhead.mp3"
y, sr = librosa.load(audio_path,offset=24, duration=20)

decomposer = rhy_decomposer.MedianNMF(y,sr)
components = decomposer.decompose()
```

This modular behavior will persist throughout the implementation as it allows constant extension and allows to swap different components as needed.

6.3.2 NMF decomposition

NMF decomposition requires an input of an audio signal and number of components in this case the default number is 5 and it returns the activations and components.

```
def decompose(self):
    #filter out precussive parts
    hpss_y = self.hpss()
    #Perform Short-time Fourier transform
    D = librosa.stft(hpss_y)
    # Separate the magnitude and phase
    S, phase = librosa.magphase(D)
    #NMF decompose to components
    components, activations = self.decomposeNMF(hpss_y, S, self.n_components)
    #reconstruct and return
    return [self.reconstructComponent(
        components[:, i], activations[i], phase) for i in range(0,len(activations))]

def decomposeNMF(self, y, magnitude,  n_components):
    # Decompose by nmf
    return librosa.decompose.decompose(magnitude, n_components, sort=True)
```

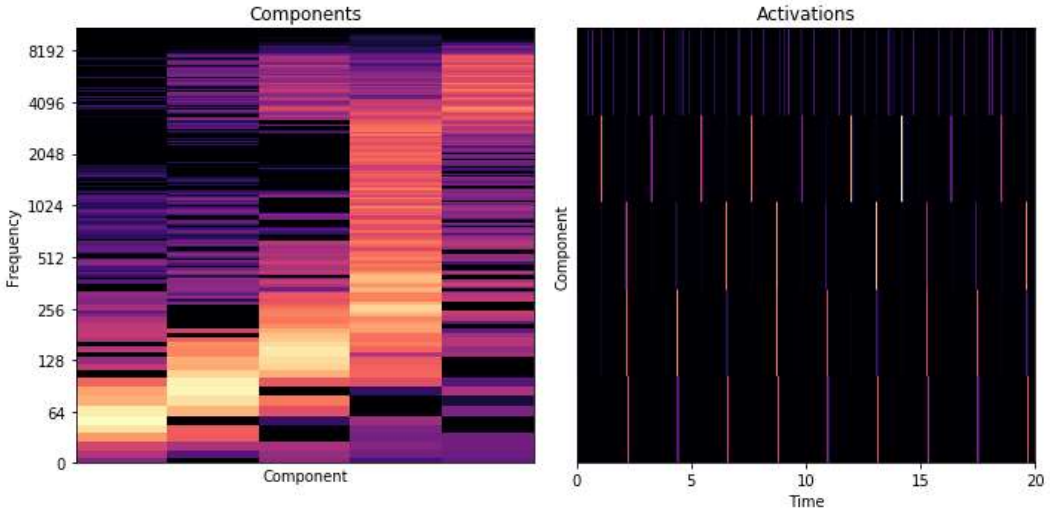


Illustration 14: NMF components and activations

The output of this function visualized in Illustration 14 shows the frequency bins and the activation of the components, then the component is reconstructed.

```
def reconstructComponent(self, components, activation, phase):
    D_k = np.multiply.outer(components, activation)
    y_k = librosa.istft(D_k * phase)
    #filter out noise using Savitzky-Golay filter
    component_filtered = sp.savgol_filter(y_k,11,1)
    return component_filtered
```

6.3.3 Component Filter

This module will take multiple audio signals and analyze it's fundamental frequency and band pass the signal. This is done to reduce bleed between multiple signal.

As averaging multiple FFT bins take much processing power, the algorithm does On-Set analysis and averages the FFT bins of those frames to create IIRfilter of the frequency.

```
def getFundementalFrequency(self, component):
    #Perform on set detection on frames
    frames = librosa.onset.onset_detect(y=component, sr=self.sr)
    #for each beat frame get maximum freq frame
    avgs = list(map(lambda x : np.argmax(librosa.stft(component)[..., x]),frames))
    #get the one with the most value thus fundemental frequency
    return np.max(avgs)

def buildFilter(self, component,freq_window=5):
    fundemental_freq = self.getFundementalFrequency(component)
    #Parameters for the bandpass filter
    lowcut = fundemental_freq - freq_window
    hicut = fundemental_freq + freq_window
    ranges = (
        lowcut / self.nyquist_rate,
        hicut / self.nyquist_rate
    )
    #Make iirfilter filter
    return sp.iirfilter(1, ranges, btype='bandpass', output='sos')
```

The output of this functions will create multiple filters for each component, as shown in illustration 15 which can be applied using sosfilt.

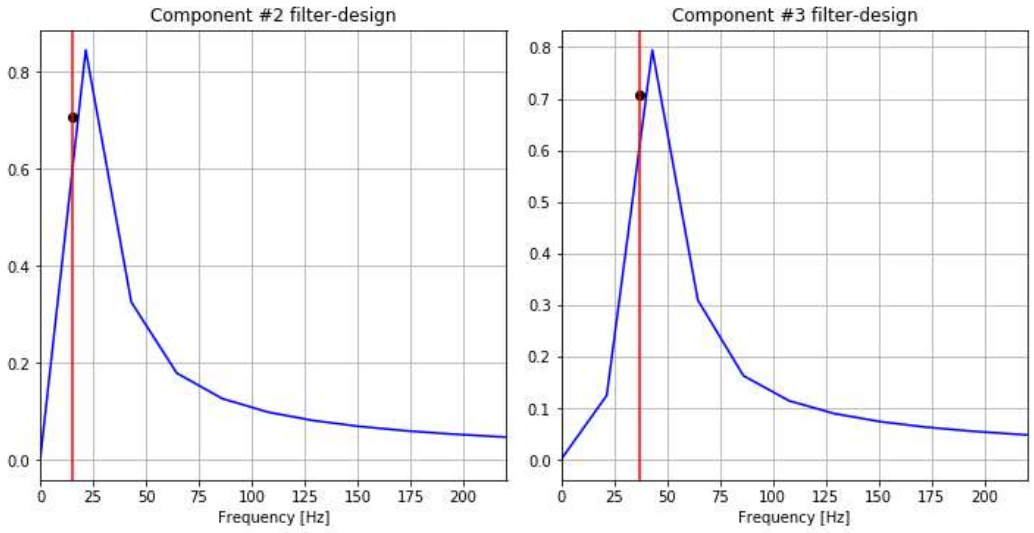


Illustration 15: Band pass filter design with fundemental frequency

6.3.4 On-Set Extractor

After the components are filtered through the previous module, the audio signals are sent to the On-Set Extractor which will convert the audio signals to time series and then will merge the duplicate time series together.

6.3.4.1 On-Set cleanup

The first process of extracting On-Set's will be done using librosa's built in On-Set detector. After that process as the data still has unwanted On-Sets, frames under a certain threshold will be dropped and duplicates frames also will be dropped.

```
def dropFrames(self, profile, frame_window=2, thrushold=0.2):
    #Calculate average energy and drop the other frames
    average = np.max(profile.energy) * thrushold
    filtered = list(filter(
        (lambda x : np.average(
            profile.energy[x-frame_window:x+frame_window]) > average)
    , profile.frames))
    return OnSetProfile(filtered,profile.energy)

def dropDuplicates(self,profile,thrushold=5):
    #frames within certain thrushold window will be dropped
    component = profile.frames
    new_frames = []
    for index,frame in enumerate(component[1:]):
        frames = component[index:index+2]
        if np.diff(frames) > thrushold:
            new_frames.append(frames[1])
    return OnSetProfile(new_frames,profile.energy)
```

As seen in illustration 16 and 17 after clipping frames under an averaged threshold and dropping duplicates the On-Sets are less noisy and more usable.

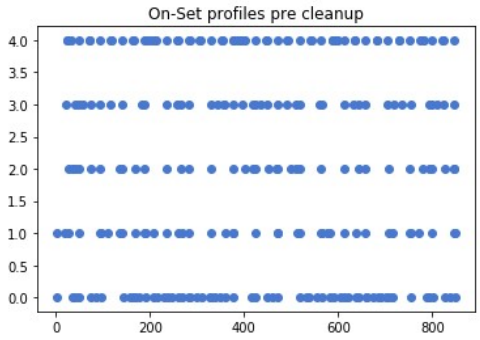


Illustration 16: On-Set profiles pre-cleanup

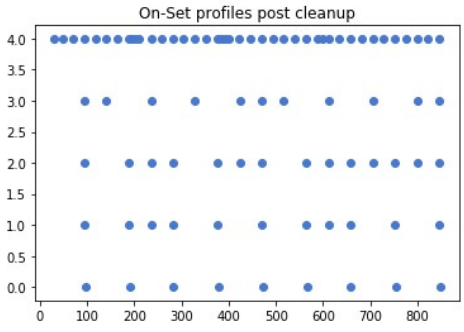


Illustration 17: On-Set profiles post-cleanup

6.3.4.2 Profile Cluster

As seen in illustration 17, there are duplicate components, these time-series needs to be processed for similarity and merged together. This is archived through an algorithm implemented by the author which compares each component for it's similarity and then clusters those time-series to be merged.

```
def frameSimilarity(self, y, x, frame_thrushold=10):
    y_frame_index = 0
    lost_frames = 0
    found_frames = 0
    for x_frame in x:
        old_index = y_frame_index
        found = False
        for index,y_frame in enumerate(y[y_frame_index:]):
            diff = x_frame - y_frame
            found = diff in range(-(frame_thrushold),frame_thrushold)
            if found:
                break
            else:
                y_frame_index = index
        found_frames+= 1 if found else 0
        if not found:
            lost_frames+=1
            y_frame_index = old_index
    return found_frames,lost_frames
```

6.3.4.3 Profile Merge

After the profiles are clustered In the previous step these time-series need to be flattened, this done by taking each of the time-series and there energy's into one time-series and dropping frames under an averaged threshold therefore leaving the most prominent On-Set's.

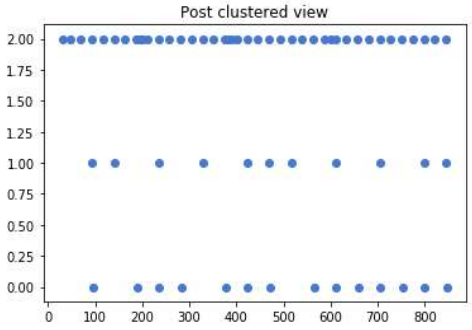


Illustration 18: Post merged Profiles

As shown in Illustration 18 the previously repeated time-series is merged together. Refer illustration 17 for comparison.

6.3.5 Common Bar Extractor

Common bar extraction part is a vital process, as extracted drum pattern from the songs can't be compared directly as songs have different section, intros, outro, chorus, etc. Therefore finding the most common drum pattern section is important.

This is made possible with the beat tracking with On-set profiles which gives us precise points of where the beats of the audio signal is therefore allowing slices to be cross checked with a DTW algorithm to see the similarities between the reference section and the other sections of the song. Basic flow of the algorithm devised is explained below.

1. Slice time-series into 2 bar[8] sections.
2. Analyze each slice bar with other beats of the time-series using DTW.
3. Store the average distance for that section.
4. After all sections are checked, the bar with the least average is the most repeated bar.

Implementation of this flow is listed below.

```
def averageBars(self):
    output = {}

    bar_beats = self.audio_data.beats[::32]
    step_beats = self.audio_data.beats[::16]

    for bar_i,bar in enumerate(bar_beats[0:-1]):
        slice_ref = DTWExtractor.slicer(self.audio_data.getProfiles(),
            [bar,bar_beats[bar_i+1]])
        bar_dist = {}
        for beat_i,beat in enumerate(step_beats):
            cur_i = beat_i
            if cur_i+3 >= len(step_beats): break
            slice_ = DTWExtractor.slicer(self.audio_data.getProfiles(),
                [beat,step_beats[cur_i+2]])
            distance = DTWExtractor.profileDTW(slice_ref, slice_)
            bar_dist[beat,step_beats[cur_i+2]] = distance

        avg = np.average(list(bar_dist.values()))
        output[avg] = (bar,bar_beats[bar_i+1])
    return output

def getCommonBar(self):
    output = self.averageBars()
    key = np.amin(list(output.keys()))
    slice_ = DTWExtractor.slicer(self.audio_data.getProfiles(),output[key])
    return list(map(lambda x : OnSetProfile(x,[]),slice_))
```

8 Set amount of beats

Visualized in Illustration 19 is the beats of the song in the dotted green lines, blue dots show the on-set time-series and the red lines slice shows the most common repeated section of this specific song.

Illustration 19: Most common bar extracted from time-series

Below in Illustration 20 is the slice expanded, as visible the beat based common bar extraction method works well in this case, as the author is familiar with this specific piece of music.

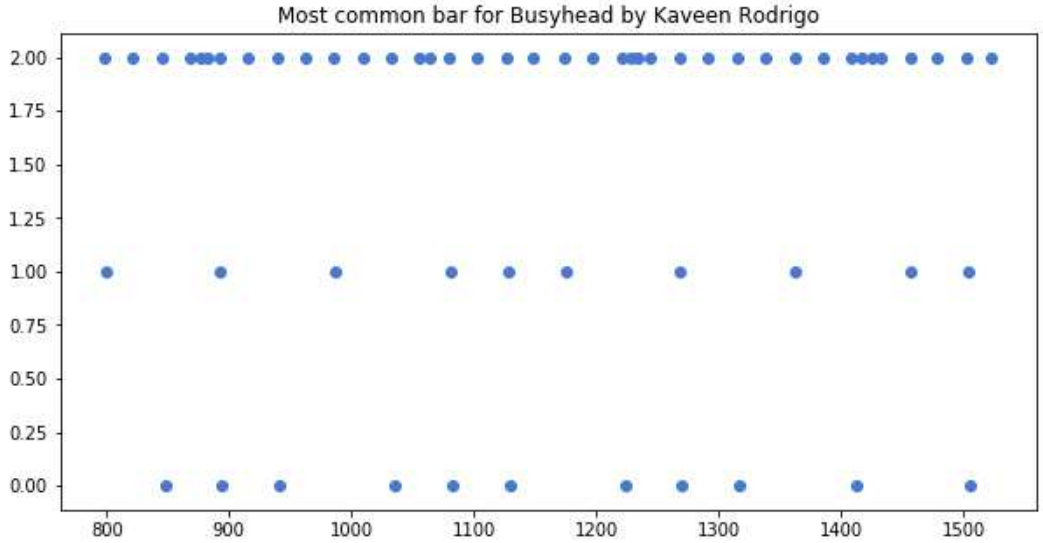


Illustration 20: Sliced common bar

6.3.6 Processing and Storing Set Of Tracks.

They RhyAn process is done prior to the suggestions part therefore, the alogrithm doesn't have to process each and every track upon suggestion. A set of input tracks will be places in an input folder and the process script will import RhyAn library and start processing the tracks and store the metadata of the extracted profiles.

This task is handed over to a class named AudioProcessor which will take the location of the audio file and process it's profile using RhyAn and extract other track information using eyed3, each task of this processing will be explained in this section.

6.3.6.1 Loading input audio tracks for processing and saving the output

The first task is to traverse through the input directory automatically and process each of the audio files using the AudioProcessor which will be explained in the reminding sections of section 6.3.6.

```
if __name__ == "__main__":
    PARENT_DIR =  "data/input_audio"
    OUT_DIR = "data/audio_data"
    for file in os.listdir(PARENT_DIR):
        print ('Procesing {}'.format(file))
        t = AudioProcessor('{}/{}'.format(PARENT_DIR,file))
        outfile = open('{}/{}.json'.format(OUT_DIR,file.split(".")[0]),"w")
        outfile.write(json.dumps(t.metadata, indent=4))
        outfile.close()
```

The output of the AudioProcessor will be saved as a json file in the selected output directory this information will be later used to migrate the database with the common bar data and track information.

6.3.6.2 Using RhyAn to extract the profile information

After the track is loaded, the AudioProcessor will run the analyse function which will import RhyAn module as explained in section 6.3 and will analyses and store the results in the data structure.

```
def analyse(self):
    decomposer = rhy_decomposer.MedianNMF(self.y,self.sr)
    components = decomposer.decompose()
    filtered_components = rhy_filter.FundementalBandPass(components,self.sr).apply()
    profiles = rhy_detect.BasicOnSet(filtered_components,self.sr).apply()
    self.metadata["BPM_isolated"] = self.getBPM(profiles[0].energy)
    clustered_profiles = rhy_analysis.ClusterProfiles(profiles).apply()
    merged_profiles = rhy_analysis.MergeProfiles(clustered_profiles).apply()
    self.metadata["profiles"] = self.formatProfiles(merged_profiles)
```

6.3.6.3 Extracting metadata with eyed3

After the track is processed another set of information to be stored in the data structure is metadata of the track itself which will include information such as track artist an title. This information will be useful for the general identification of the track and in the indexing process.

```
def processID3(self):
    #extract MP3 metadata data
    instance = eyed3.load(self.url)
    self.metadata["artist"] = instance.tag.artist
    self.metadata["title"] = instance.tag.title
```

6.4 RhyAn Web API

This module will allow users to interact with the music library and get suggestions, this interface is done as a prototype to test the algorithm. The users will be able to connect and play and get suggestions from the tracks from the DB.

The web API apart from providing an interface to access information from the data store will also be given the task of hosting and giving access to the audio files for the Web UI and also host the static files of the Web UI.

Since the Web API uses SQLAlchemy, it has the ability to work with different database programs provided a driver is available for the specific database application, for testing purposes the application is currently using SQLite as the data store.

The basic functions of the web API is as follows.

- Serve the web application.
- Provide endpoints for searching the music library.
- Use RhyAn package and data-store to give suggestions.
- Serve the audio files for the web application.

6.4.1 SongEntry index model

The index database will be used to store each of the tracks information regarding it's audio file location and common bar will also be stored in the database for search functionality.

The common bar will be stored as Pickled Type therefore when retrieving the records via the ORM the data will be converted into the specific data structure automatically.

```
from ..db import db

class SongEntry(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(), nullable=False)
    artist = db.Column(db.String())
    url = db.Column(db.String())
    bpm = db.Column(db.Float())
    sr = db.Column(db.Float())
    bar = db.Column(db.PickleType())

    def __init__(self, title, artist, url, bpm, sr, bar):
        self.title = title
        self.artist = artist
        self.url = url
        self.bpm = bpm
        self.bar = bar
        self.sr = sr
```

6.4.2 Suggestions end point

For this prototype a chuck of the index tracks will be recalled and checked against the reference using the DTW function implementation of RhyAn. After processed an array with averages will be returned in which where the average distance is lowest is the rhythmically close to the reference track.

These values in the process are normalized by mapping each and every time-series to a common scale of zero to twohundred.

```
@api.route('/suggest/<id>')
def suggest(id):
    entry = SongEntry.query.get(int(id))

    tracks = (
        SongEntry.query.filter(SongEntry.id != entry.id)
        .filter(SongEntry.bpm >= entry.bpm - 2)
        .order_by(SongEntry.bpm).all()
    )

    avgs = {
        rhy_analysis.DTWExtractor.profileDTW(
            entry.rawProfiles(), track.rawProfiles()) : track for track in tracks
        }

    avgs = {k:v for (k,v) in avgs.items() if k is not -1}

    out = [
        {"score" : avg , "song": avgs[avg].toDisplay()}
        for avg in sorted(avgs.keys()) [0:15]
    ]

    return jsonify(out)
```

6.4.3 Static File Hosting

The Web API will also be given the task of hosting the react application for user interaction and the audio files for playback

```
@root.route('/')
def root_page():
    return root.send_static_file('index.html')

@root.route('/audio/<path:path>')
def send_audio(path):
    return send_from_directory('data/input_audio', path)
```

6.4.4 Migration of the Models and indexing track data

As discussed in section 6.3.6 the processed tracks were stored as a JSON document for indexing and storing the common bar in the SongEntry model. Therefore a script is required that will read each of this JSON documents and parse the information and create an SongEntry model for each of the entries with the common bar information.

6.4.4.1 Loading pre processed JSON documents and storing

In this process the pre processed JSON documents will be imported and processed and stored in the database as an SongEntry entry. The documents wll be loaded one by one into the script and be run through the analysis steps explained in the section 6.4.4.2

```
def addEntries(db):
    PARENT_DIR =  "data/input_audio"
    OUT_DIR = "data/audio_data"
    for file in os.listdir(PARENT_DIR):
        print ('Procesing {}'.format(file))
        infile = open('{}/{}.json'.format(OUT_DIR,file.split(".")[0]),"r").read()
        data = AudioData(json.loads(infile))
        try:
            db.session.add(processEntry(data))
            db.session.commit()
        except:
            print ('Failed On {}'.format(file))
            pass
```

6.4.4.2 processing JSON documents into SongEntry entries

As explained in the previous section the loaded JSON document fields will be mapped into SongEntry fields afterwords the most common bar will also be extracted and stored as a PickledType on the SongEntry model, since RhyAn module is used this can be easily done with one line.

```
def processEntry(data):
    #extract most common bar from input data
    common_slice = rhy_analysis.DTWExtractor(data).getCommonBar()
    #return SongEntry entity
    return SongEntry(
        data.title,
        data.artist,
        data.url.split('/')[1],
        data.BPM_overall,
        data.sr,
        common_slice
    )
```

The returned entities will be inserted to the database as shown in the previous section above.

6.5 Rhyan Web UI

This module will be the front end application for RhyAn and RhyAn Web API, It will allow users to browse through the the media library and get suggestions for the available tracks.

The Web UI will be programmed with React.js and bootstrapped with Facebook's create-react-app, which allows to configure a full fledged JavaScript environment in a matter of minutes.

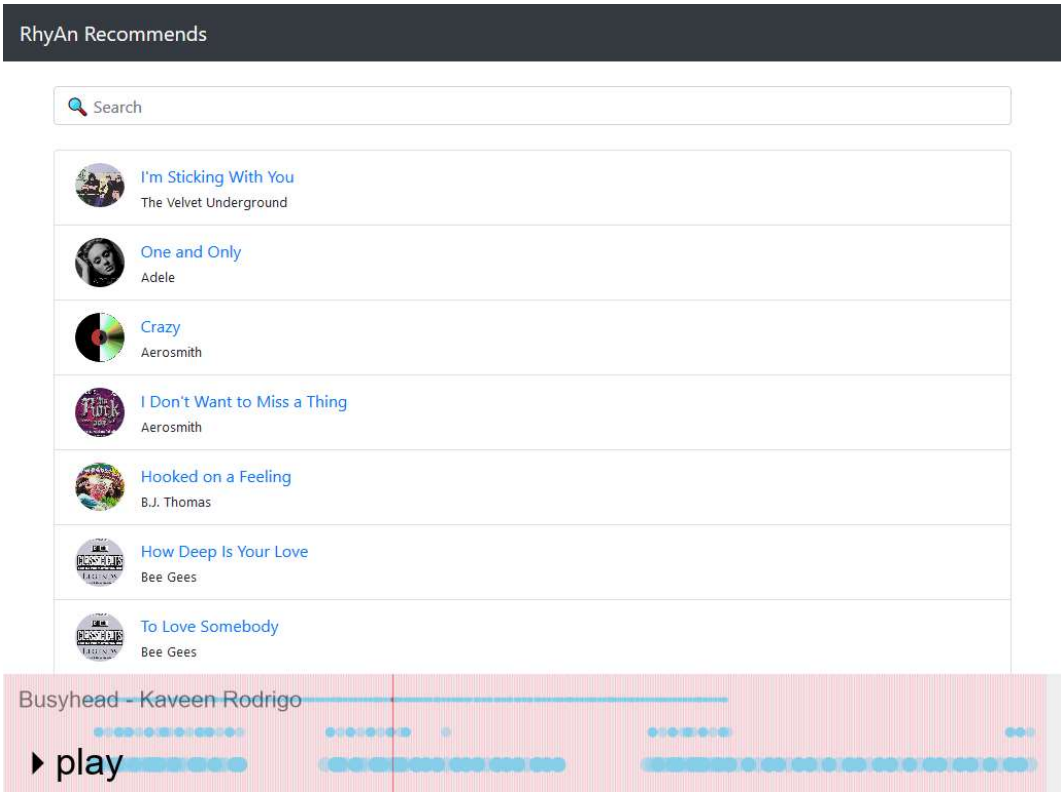


Illustration 21: Screenshot of the UI

6.5.1 Libraries used for the Web UI

The libraries are mainly used to enhance the appearance of the interface, and some are used for communication between the web UI and the API. Below is a table of libraries used and there usage.

Name	Purpose
reactstrap	Reactstrap is front end library with binings to the popular Bootstrap framework, that has base UI components converted to React elements.
react-router-dom	Important part for a SPA[9] this web application will be done in an SPA approach which requires a routing system for routing different screens. This provides an easy solution for routing.
react-audio-player	React Audio Player gives bindings for the HTML5 media element which will allow the media element to be used as a React component with easy to bind props and events.
Konva	Konva is a graphics library that wraps on the HTML canvas element, therefore graphical drawings can be rendered with React components and JSX[10]
Axios	Axios is a network library that will bed used to communicate with the Web API

Basic functionalities that are implemented on this web UI is as follows. Most of this functions are implemented as they are needed for the basic working product, bu the visualization is used mostly as a debugging tool.

1. Browse the media library.
2. Play a certain song.
3. See suggestions for the playing song.
4. Show visual output of RhyAn's processed profiles.

9 Single Page App
10 JavaScript XML

6.5.2 Implementing the graphical visualizer

For demo purposes, A visualizer is needed to display the output of RhyAn for each and every track. For this Konva will be used as a graphics library to render.

The graphic visualizer will be used to visualize the output of RhyAn providing a real time look of all the data extracted while listening to the rack.

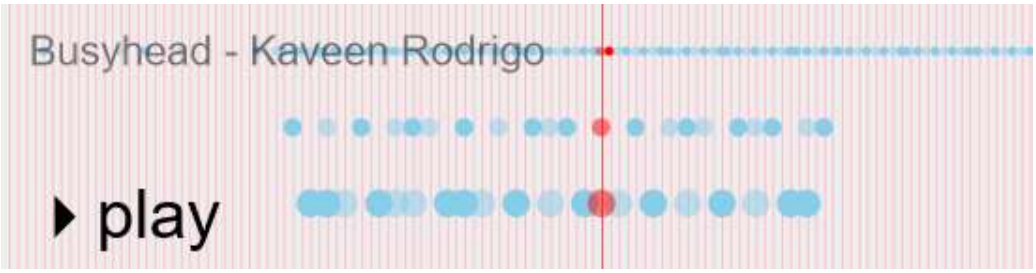


Illustration 22: Screenshot of visualizer

6.5.2.1 Visualizing the beats and hits

Once a track is loaded into the play view, it contains the beat timestamps of the track, on the canvas of the visualizer. After this is done the visualization of the On-Set profiles will be rednered,

the location in the x-y plane of this canvas is calculated by the map function, which maps the time space of the song to the width of the canvas the function is shown below,

```
mapWidth(timecode){
    return map(timecode, 0,this.state.duration,0,this.state.width);
}
```

Next task is visualizing the beats using the aforementioned mapWidth function, this is done by simply iterating through each of the beats and tracks with it's timestamp and

```
renderBeats(){
    return this.state.beats.map((beat) => {
        return <Rect
            x={this.mapWidth(beat)}
            y={0}
            width={1} height={this.state.height} fill={"pink"} />
    })
}
```

rendering the Konva component.

Afterwords rendering of the On-Set patterns can be done the same way as rendering the beats, in this case multiple number of profiles is again spread out using the map function to occupy the whole height of the canvas.

```
renderHits(){
    var all_hits = []
    this.state.profiles.forEach((profile,i) => {
        const new_color = "skyblue"
        var fade_toggle = true
        all_hits = all_hits.concat(profile.map((beat) => {
            const new_y = map(i, 0, this.state.profiles.length,
                10,this.state.height-10) + 15
            const new_r = 5 * ((i+1) * 0.5);
            const hit_color =
                Math.abs(beat - this.state.timeStamp) < 0.5 ? "red" : new_color
            fade_toggle = !fade_toggle;
            return < Circle
                x={this.mapWidth(beat)}
                y={new_y}
                radius={new_r}
                fill={hit_color}
                opacity={fade_toggle ? 0.5 : 1} />
        }));
    });
    return all_hits;
}
```

In the code snippet it is also shown how the current playing time-section is highlighted in red.

6.5.2.2 Adding interface elements

Apart from acting like the visualizer, the canvas also doubles as an interface for the media player therefore it is required to have controls to play/pause and to skip to different sections by clicking the relevant section of the canvas.

```
renderInterface(){
    return (
        <Group>
            <Text
                x={15} y={15}
                fontSize={22}  opacity={0.5}
                text={this.state.title + " - " + this.state.artist}
            />
            <Rect
                x={0} y={0}
                width={this.state.width} height={this.state.height}
                opacity={0} onClick={this.seekTrack}
            />
            <Text
                text={this.state.audioEl.paused ? "\u23f5play" : "\u23f8pause"}
                fontSize="36" x={15} y={this.state.height - (10+36)}
                onClick={this.togglePlay}
            />
        </Group>
    );
}
```

The current playing track is also shown in the visualization as the user can browser through while other tracks are playing.

6.5.3 Recommendations Display

The recommendation display is shown after a track is selected from the main screen (Illustration 21) The album art and track details will be shown on the left side while the track suggestions will be loaded on the right side. The tracks will automatically be played with the new data loaded into the visualizer.

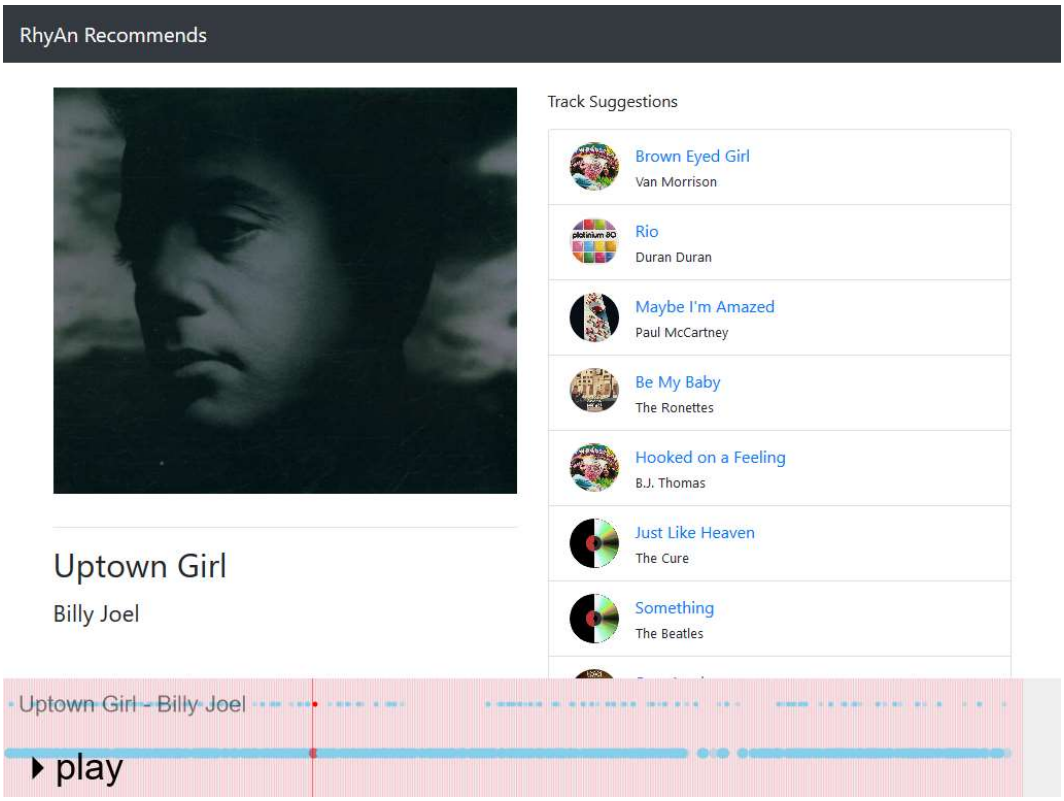


Illustration 23: RhyAn Recommends Track Suggestions

7 Testing

7.1 Overview

There is no out of the box evaluation approaches when it comes to evaluating the final program, there are two main components that needs evaluation, one being the Rhythmic analysis algorithm and the other is the recommendation algorithm.

7.2 Test Data Preparation

For the test data the main and require criteria is that the test track should be synthesized from a MIDI file as the algorithm can be only tested by comparing the synthesized track against the MIDI file which has the proper time-series.

Because of the aforementioned criteria, the tracks were prepared and synthesized by the author. The midi files are downloaded for free from http://www.midiworld.com/ which was later synthesized into audio signals using TiMidity++

400 odd tracks were randomly selected from the aforementioned website spanning genres from pop, rock to country and was synthesized into audio signals, for a complete list of tracks referrer to appendix A.C.

7.3 Testing RhyAn Accuracy

7.3.1 Testing Methodology

The algorithm will be evaluated by selecting a couple of musical works from different genres with a MIDI[11] musical score, the MIDI musical score is a digital equivalent for sheet music. Using the MIDI score and the output of the algorithm a comparison can be used to analyze the accuracy of the algorithm.

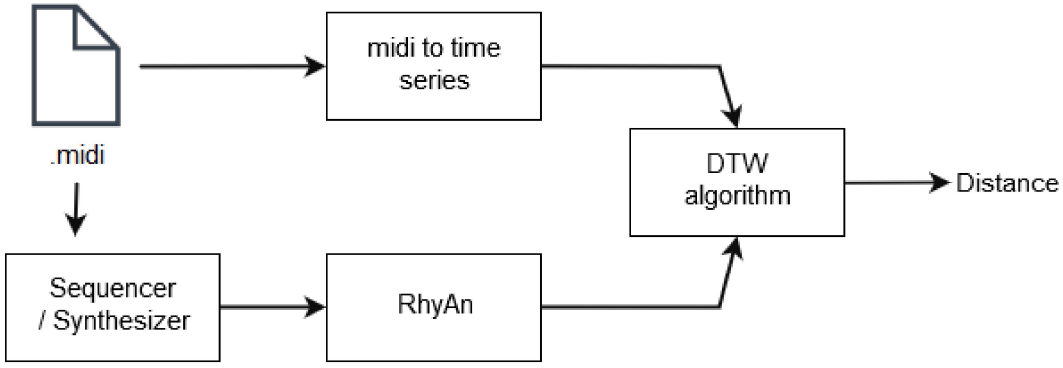


Illustration 24: RhyAn testing method

This task is further explained in Illustration 24, multiple midi documents will be converted to the native time-series profiles of the RhyAn system and will be compared against the RhyAn processed time-series profiles of the synthesized signal.

Generating the synthesized section and converting the midi document into time-series is a tedious task, therefore only a limited amount of tests can be done.

7.3.1.1 Limitations

The major limitation of this testing method is that the synthesizer only has a limited selection acoustic tones and audio configuration therefore a bias to this configuration might be expressed. The general-midi standard does contain multiple kit sounds IE – electronic and acoustic.

11 *Musical Instrument Digital Interface*

7.3.2 Results

Table 1 below shows an excerpt of the results from the total 400 odd tracks processed, important note to take here is the normalized distance is desirable with a low value.

The normalized distances are calculated $normalizedDist(x,y)=\frac{DTW(x,y)}{x^{duration}}$ where the DTW result of x and y is divided by the duration thus the distance is normalized to a common unit. the The profiles parameter shows how many components was extracted from the MIDI file in the original field and how many were extracted from RhyAn in the extracted field. Percent error was calculated with the use of the normalized distance with the following formula D being the distance. Note $expected+1$ as to avoid division by zero.

$$\%error=\frac{|expected-D|}{expected+1}*100\%$$

Title	Duration	Distance		Expected Distance	% Error
		Distance	Normalized		
Blackstreet_-_Take_Me_There	234.5535	517.6122	2.2068	0	2.2068%
Earth_Wind_and_Fire_-_Let's_Groove	342.3869	893.2137	2.6088	0	2.6088%
England_Dan_-_I'd_Really_Love_to_See_You	159.6604	548.2312	3.4337	0	3.4337%
Diana_Ross_-_Touch_Me_in_the_Morning	175.8824	545.6716	3.1025	0	3.1025%
Boyzone_-_No_Matter_What	272.9012	878.9061	3.2206	0	3.2206%
ABBA_-_Name_of_the_Game	222.5371	813.2022	3.6542	0	3.6542%
Boyzone_-_Words	240.2220	1153.2365	4.8007	0	4.8007%
Christopher_Cross_-_Sailing	249.6000	1063.5454	4.2610	0	4.261%
George_Michael_-_Faith	140.0163	525.8939	3.7559	0	3.7559%
Boyzone_-_You_Needed_Me	218.0441	1262.1531	5.7885	0	5.7885%
Bloodhound_Gang_-_The_Bad_Touch	237.8971	615.0245	2.5853	0	2.5853%
Dan_Fogelberg_-_Same_Old_Lang_Syne	262.7918	1136.1244	4.3233	0	4.3233%
Celine_Dion_-_To_Love_You_More	327.7584	1230.3919	3.7540	0	3.754%
Barry_Manilow_-_Mandy	236.5910	1375.5425	5.8140	0	5.814%
Backstreet_Boys_-_Shape_of_My_Heart	231.7322	954.4127	4.1186	0	4.1186%
Chaka_Khan_-_Tell_Me_Something_Good	281.6522	985.2217	3.4980	0	3.498%
George_Benson_-_Turn_Your_Love_Around	236.3820	652.1388	2.7588	0	2.7588%
Backstreet_Boys_-_Show_Me_the_Meaning	231.9151	789.6156	3.4048	0	3.4048%
Enrique_Iglesias_-_Bailamos	222.2759	1478.1632	6.6501	0	6.6501%
ABBA_-_Fernando	255.6604	680.3090	2.6610	0	2.661%
All_Saints_-_Never_Ever	312.4767	1211.6147	3.8775	0	3.8775%
Cardigans_-_Erase_And_Rewind	223.3992	489.1386	2.1895	0	2.1895%
ABBA_-_The_Winner_Takes_It_All	289.9592	1155.3338	3.9845	0	3.9845%
George_Benson_-_Give_Me_the_Night	292.5453	1337.3339	4.5714	0	4.5714%

Table 1: RhyAn tests excerpt

7.3.2.1 Distribution

After binning the data was fitted to a normal distribution taking the Normalized distance as one dimensional array.

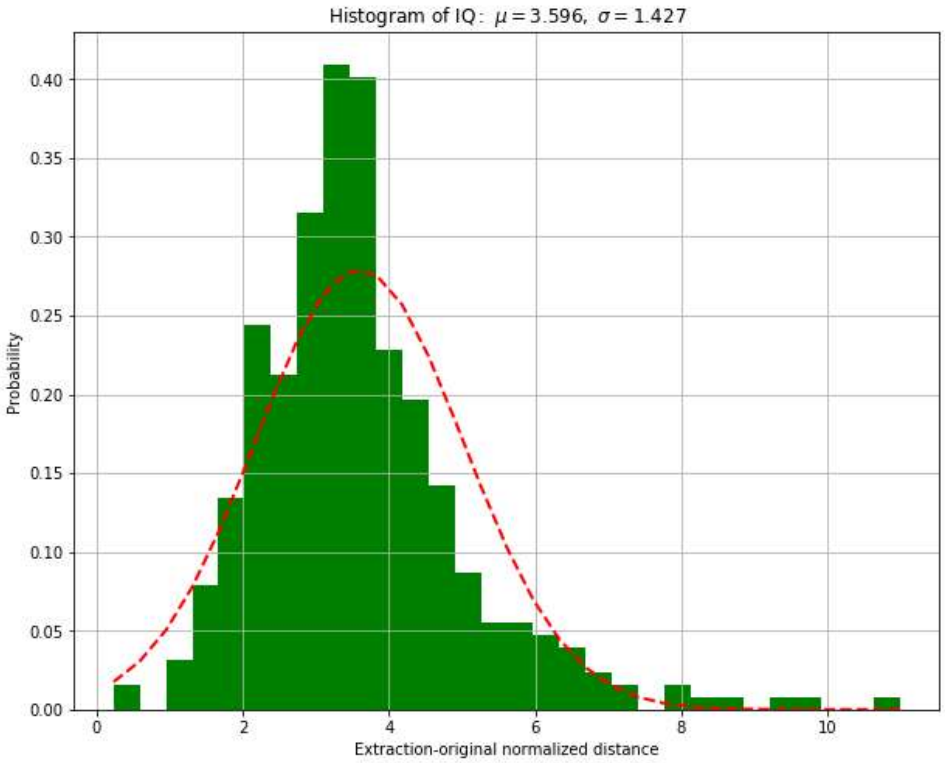


Illustration 25: distribution of RhyAn test results

Illustration 25 shows that the mean is around 3.595, and is close to the target value of zero.

7.3.2.2 RMSE

With the data of table 1 RMSE[12] was calculated with the formula shown below where N is the normalized values list while E being the common target value.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} |N^i - E|}{n}}$$

With a **RMSE of 3.8685%** with 354 entries, % error rate is desirable as it falls under 10%.

12 Root Mean Square Error

7.4 Testing RhyAn Recommends accuracy

7.4.1 Testing Methodology

One of the main factors that needs to be tested is the DTW based recommender's accuracy in different tempos of the same pattern. This is important to be tested as some tracks maybe upbeat but share the same pattern.

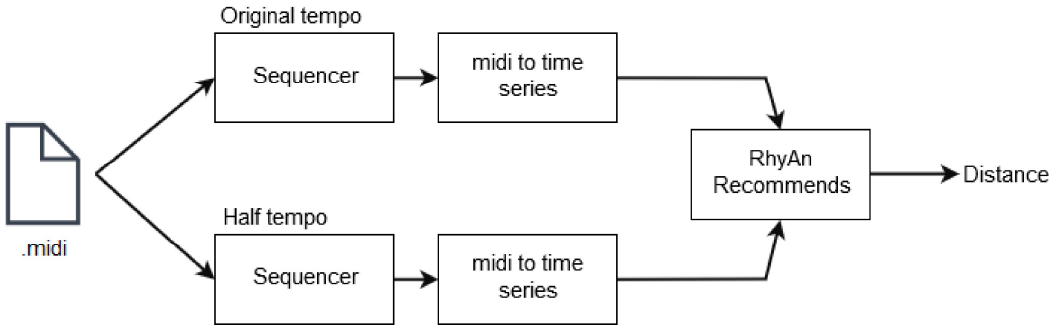


Illustration 26: RhyAn Recommender accuracy

As illustrated in illustration 26, a midi document will be synthesized in two different tempos (speed) which will be fed into the RhyAn recommends, and the most favorable distance should be low.

With this result we are able to determine that RhyAn recommends is tempo agnostic and is capable of comparing patterns with different tempos. RhyAn Recommends core DTW algorithm will return the distance difference between the two inputs which will be normalized by subtracting $x^{duration} - y^{duration}$

7.4.2 Results

Table 2 is an excerpt from the testing results of the aforementioned testing methodology with 47 entries. In this table as with section 7.3.2 lower the distance is better. The normalized values are calculated using $normalizedDist(x,y)=\frac{RhyAnDTW(x,y)}{x^{duration}-y^{duration}}$ which in order was used for calculate the percent error and RMSE.

Title	Duration	Duration Half-time	Distance		Expected Distance	% Error
			Distance	Normalized		
ABBA_-_Gimme_Gimme_Gimme	290.1217	145.0608	19.4044	0.1338	0	0.1338%
ABC_-_Poison_Arrow	202.9363	101.4682	12.9284	0.1274	0	0.1274%
ABC_-_The_Look_of_Love	208.1282	104.0641	13.2471	0.1273	0	0.1273%
Ace_of_Base_-_All_That_She_Wants	205.2628	102.6314	11.7899	0.1149	0	0.1149%
Aha_-_Hunting_High_And_Low	265.9000	132.9500	16.1492	0.1215	0	0.1215%
Aha_-_Take_On_Me	224.8776	112.4388	13.8060	0.1228	0	0.1228%
Aha_-_The_Sun_Always_Shines_On_TV	294.7080	147.3540	20.8939	0.1418	0	0.1418%
Andrew_Lloyd_Webber_-_Phantom_of_the_Opera	255.8600	127.9300	19.5095	0.1525	0	0.1525%
Bananarama_-_Venus	243.7031	121.8516	15.6961	0.1288	0	0.1288%
Barbra_Streisand_-_Woman_In_Love	235.7293	117.8646	14.8563	0.1260	0	0.126%
Barry_Manilow_-_Copacabana	243.2671	121.6335	15.9888	0.1315	0	0.1315%
Belinda_Carlisle_-_Heaven_Is_a_Place_on_Earth	252.8253	126.4126	16.4581	0.1302	0	0.1302%
Billy_Idol_-_White_Wedding	201.8542	100.9271	12.8067	0.1269	0	0.1269%
BJ_Thomas_-_Hooked_on_a_Feeling	161.9513	80.9756	10.4112	0.1286	0	0.1286%
Blur_-_Girls_Boys	288.0000	144.0000	18.3653	0.1275	0	0.1275%
Blur_-_Song_2	118.8872	59.4436	7.2931	0.1227	0	0.1227%
Bobby_Vinton_-_Blue_Velvet	137.9844	68.9922	9.1712	0.1329	0	0.1329%
Cher_-_Believe	247.5767	123.7883	17.4792	0.1412	0	0.1412%

Table 2: RhyAn DTW results excerpt

7.4.2.1 Distribution

After binning the data was fitted to a normal distribution taking the Normalized distance as one dimensional array.

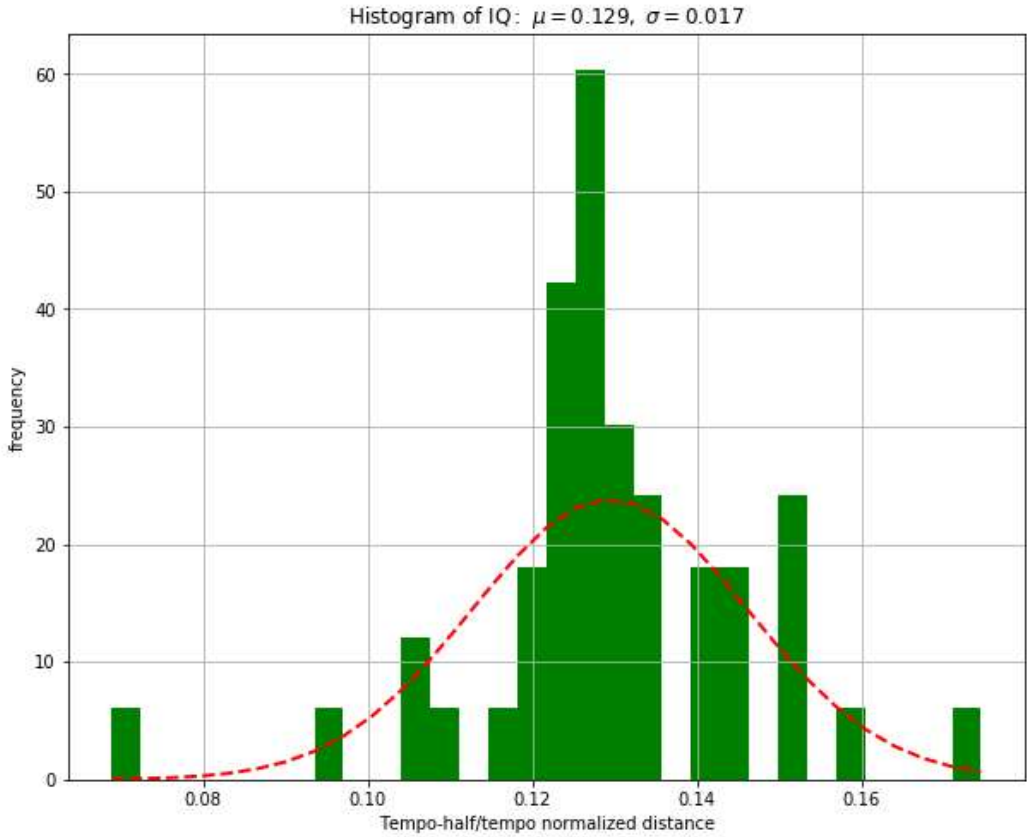


Illustration 27: distribution of RhyAn Recommends

7.4.2.2 RMSE

With a **RMSE of 0.1307%** the results are satisfactory.

7.5 Component Filter Test

7.5.1 Testing Methodology

The filter will be tested with a similar method as used in sections 7.4 and 7.6 as it will take msuic from the existing midi data set and will be tested with RhyAn, in this case the same RhyAn extraction process will be run where in one case it will be tested with the component filter and the latter without the filter.

7.5.2 Results

The results (excerpt) in Table 3 were calculated by checking the difference of the distances with the component filter enabled and disabled, then $x^{normalizedDist} > y^{normalizedDist}$ where x is component filtered and y is component without filtration.

Title	Duration	Distance		Distance Filtered		Difference	improvement
		Distance	Normalized	Distance	Normalized		
Glen_Campbell_-_By_the_Time_I_Get_to_Phoenix	172.1469	917.0833	5.3273	718.1424	4.1717	1.1556	yes
Glen_Campbell_-_Wichita_Lineman	196.8327	816.0338	4.1458	743.5922	3.7778	0.3680	yes
Guns_n_Roses_-_November_Rain	489.6392	2221.4412	4.5369	1892.2226	3.8645	0.6724	yes
Jethro_Tull_-_Aqualung	397.0873	1686.1338	4.2463	1400.3649	3.5266	0.7197	yes
Kate_Bush_-_Wuthering_Heights	222.6678	1019.9133	4.5804	954.1497	4.2851	0.2953	yes
Nirvana_-_Floyd_the_Barber	142.8898	563.8749	3.9462	505.7094	3.5392	0.4071	yes
Nirvana_-_In_Bloom	255.3731	928.5057	3.6359	940.6588	3.6835	-0.0476	no
Nirvana_-_School	137.5608	1043.1202	7.5830	1027.2853	7.4679	0.1151	yes
Nirvana_-_Something_In_The_Way	210.9388	1503.3198	7.1268	1497.0991	7.0973	0.0295	yes
REM_-_Everybody_Hurts	269.2180	1455.1243	5.4050	1031.9475	3.8331	1.5719	yes
REM_-_Losing_My_Religion	279.0139	959.3069	3.4382	574.8949	2.0605	1.3778	yes
REM_-_Man_on_the_Moon	321.7763	1380.6841	4.2908	1179.9868	3.6671	0.6237	yes
REM_-_Shiny_Happy_People	200.4376	645.8398	3.2221	565.6724	2.8222	0.4000	yes
The_Bangles_-_Eternal_Flame	165.1984	523.3473	3.1680	497.0672	3.0089	0.1591	yes
The_Carpenters_-_Close_to_You	228.4931	803.8122	3.5179	726.0149	3.1774	0.3405	yes
The_Corrs_-_Radio	281.9396	815.0371	2.8908	819.1517	2.9054	-0.0146	no
The_Cure_-_Boys_Don't_Cry.mid	165.4073	357.0811	2.1588	285.6591	1.7270	0.4318	yes

Table 3: Component Filter results

7.6 RhyAn Recommends Overall Test Results

This table summarizes the testing done in sections 7.3 and 7.4 which tested the major components of the RhyAn Recommends system.

Feature Tested	Module	Test Data Set Size	RMSE*	Pass Rate	Status
FR2 / 7.3	Main RhyAn Component	354	3.8685%	-	**Passed**
FR3 / 7.4	RhyAn Recommends	47	0.1307%	-	**Passed**
7.5	Component Filter	36	-	66.67%	**Not Satisfactory**

***RMSE under 10% is considered a pass.**

7.6.1 Remarks

7.6.1.1 Component Filter

The results of the component filter was not satisfactory as, In certain cases it worked well, but in come certain types of scenarios it failed or didn't run at all because of the failure of the fundamental frequency finder, as the calculated frequency falls over or under the accepted frequency range.

8 Evaluation

8.1 Chapter Overview

This chapter will evaluate the project progress compared to the project management techniques discussed and the tests done on the project. And will also talk about the overall thesis documentation progress.

8.2 Evaluation Criteria

Evaluation will be done with qualitative, quantitative methods where quantitative evaluation will be carried out with the function tests done in Chapter 7 and qualitative evaluation will be executed with the help of expert reviews and self-review.

8.3 Expert Review

After the project implementation, a couple of experts in the music field and active listeners were contacted to get opinion on the project. The composition of the review group is displayed on the table before, the interviews were conducted orally.

Name	**Description**	**Designation**
Asvajit Boyle	• Independent Musician • Record Label Holder	Founder/A&R at Jambutek Recordings.
Malinthe Samarakoon	• Active Music Listener • Software Engineer • Metallica Mega-fan	Senior Software Engineer at YAMU.

8.3.1 Review Summaries

8.3.1.1 Asvajit Boyle

Asavjjit is an independent musician and is the founder of Jambutek Recordings a record label which is a premier electronic music label in Sri Lanka. He doesn't use any major streaming platforms for his personal listening other than soundcloud and YouTube. New artist discovery generally occur on sites link Beatport and other aforementioned sites and remarked on the abundance of new listening material.

After the abstract of the RhyAn recommends system was introduced and explained Asvjith understood the basic concept and how it operates. He congratulated on the approach on rhythmic similarity and the component filtration process but showed skepticism on the recommendation algorithm as the argument being despite the algorithm being fairly accurate users may hot have the time to listen to just recommendation and with conversation came to a conclusion that a hybrid approach where recommendations spanned throughout a high user base with data mining would be a better approach.

8.3.1.2 Malinthe Samarakoon

Malinthe is an active music listener and a senior software engineer at YAMU where recommendation systems are made for restaurants and other attractions in Sri Lanka. As we progressed through explaining the idea he understood the basic concept of the idea which as limitation by the scope he brought up the issue with how this concept will not work with certain music without drums in their instrumentation. He mainly uses Spotify as his music discovery program is happy with that solution.

As a software engineer I discussed the architecture of the program and it's implementation. He implied the architecture made sense for the RhyAn process, and recommended that the raw extracted data without filtration to be stored therefore with less CPU power if needed the data can be prepossessed with a new approach.

8.3.2 Scope of the project

As established the author's scope of study is not that common for a bachelors thesis so told Malinthe. All the experts remarked on the interesting nature of the scope and the expansion opportunities the algorithm can have if the scope was expanded.

8.3.3 Overall Concept

As an independent musician and record label holder Asvajit appreciates the aim of the project and how independent musicians publishing tracks can benefit from. All experts remarked on how the overall concept of Rhythm tracking in music to give suggestions is a logical sense but questioned one of the main downfalls of the algorithm in some song instrumentation.

8.3.4 Limitations

All evaluators remarked how certain instrumentation in music that does not contain percussive instruments will not benefit from the algorithm, which is given for this approach and justified by using this algorithm with other semantic algorithms to have a hybrid approach.

Asavajit stressed on the sheer amount of new listening material put in to the wild everyday and how accurate the recommendation algorithm maybe the user may not have the time to listen to this material and suggested a hybrid approach where data mining and RhyAn can be used.

8.3.5 Architectural

RhyAn basic architecture was explained orally to all the evaluators, with basic descriptions of each component and summarized in Illustration 9. Evaluators didn't make much remarks on the architecture of the solution as It seemed straight forward.

8.3.6 Conclusion

Summerization of the evaluation suggest that all evaluators were pleased with the research topic and scope. The overall concept and approach was appropriated and remarked of being necessary to archive the aim.

Few limitations were brought up some that was apparent during the inception of the project and some that were not mainly falling to the recommendation algorithm.

8.4 Self-Evaluation

In this section, the author himself will be evaluating the overall project and It's performance and the execution of the project.

8.4.1 Overall Concept

As a music lover and computer hobbyist, this was an utmost pleasure taking this problem as mentioned in Section 1.4 this project was modeled after a problem I had. RhyAn recommends is a true labor of love that Involved me in jumping head first into Signal Processing as I've had minimal prior experience. The performance of RhyAn is satisfactory but there is more further enhancements that can be done to make the performance and accuracy even better and help It's limitations.

8.4.2 Scope Of The Project

The scope involves studying digital signal processing which I had limited prior experience with thus making the learning experience quite interesting. Apart from DSP, basic music knowledge was also required which as an amateur musician I was able to grasp with ease.

8.4.3 Design of the system

The basic architecture of this approach was thought up by brainstorming, then further literature was reviewed on this topic to analyze it further. Libraries and certain algorithms were selected after greater review for the system.

8.4.4 Implementation and Prototype

With my minimum knowledge on DSP the implementation process started slow, but with the backing of the literature review it became much more streamlined. The prototype was developed around the functional requirements and requirements.

Some functionalities like the Component Filter as tested in section 7.5 didn't yield satisfactory results, as there was some design faults and, while this didn't hinder much of the main algorithms results It should have been better designed.

8.5 Status of Functional Requirements

The following table shows the present state of the requirements as decided in section 4.5.

Id	Priority	Requirement	Status
FR1	Essential	Browse Media User should be able to browse through the collection of music through the web interface	**Completed**
FR2	Essential	Analyze Track Ability to take an audio file and process it's rhythmic qualities and store them.	**Completed**
FR3	Essential	Give Recommendations Ability to give the user music recommendations based on a reference track.	**Completed**
FR4	Essential	Calculate Similarity Between Tracks Ability to take 2 tracks / sections and calculate the similarity	**Completed**
FR4	Essential	Search Tracks Ability to search and filter the music library.	**Completed**
FR5	Essential	Ability To Play Tracks Online Ability to play selected music tracks from the interface.	**Completed**
FR6	Luxury	Graphical Visualizer Graphically Visualize the extracted rhythmic information of the song.	**Completed**

8.6 Conclusion

With the evaluation done It's safe to say that the system might be handicapped by scope as there is more further enhancements necessary for the recommendation algorithm. The main RhyAn modules architecture and performance is satisfactory as evaluated by test results and expert review.

9 Conclusion

9.1 Chapter Overview

This chapter will serve as an end note for this thesis on RhyAn recommends, dissecting the progress made on the thesis and discuss of It's strong and weak points to find out the sections that were strong and sections that needs improvement. This chapter will also talk about what further research and development can be done to enhance It's progress and accuracy.

9.2 Core Challenges Overcome

RhyAn recommends had a couple of main challenges in the DSP scope, starting from component extraction to On-set extraction and grouping. With the aid of literature review and design approaches were created to address these problems.

9.2.1 Rhythmic Analysis

RhyAn was the result of the research and design done in this thesis, that is capable of taking an audio signal and having the capability of deconstructing the signals into a set of time-series that discern the different percussive rhythmic components *IE- bass rum, snare, hi-hat, etc.*

RhyAn is capable of demonizing and cleaning up this decomposed signal and using multiple decomposed signals to compare the similarity to re-construct a time-series that is relevant to a certain percussive instruments therefore despite the harmonics of a certain timbre RhyAn tries it's best to associate it with a certain component thus returning the rhythmic components of an audio signal.

9.2.2 Recommendation System

The recommendation system hinges on RhyAn's results and a common section extractor which allows to slice the most common section of a piece of music with the input of component time-series. The advantage of this approach is to ensure that comparison of the most repeated part of the audio signal is taken place thus leaving sections like the intro, refrain etc. from comparison.

For the comparison algorithm RhyAn Recommends leverages DTW, an algorithm capable of comparing distances in two time series, therefore the algorithm is tempo-agnostic thus allowing the comparison of different extracted time-series patterns despite their tempo difference.

9.3 Completion of Aim And Objective

9.3.1 Completion of Aim

> *"To research, design, implement and evaluate a Semantic Recommendation system for music that analyzes rhythmic patterns in the percussive components of music and compare similarities between patterns to find recommendations"*

RhyAn recommends was able to archive this aim in the allocated scope and time space, Qualitative tests were performed to tests It's accuracy in archiving this aim.

9.3.2 Objectives Completed

9.3.2.1 Project Initiation Document

Project initiation document was prepared, to define and scope the project and explain the aims and objectives of the project. The project was better explained in non technical terms to give an understanding of the base idea, The PID consisted of some topics from Chapter 1.

9.3.2.2 Literature Review

Suitable literature was gathered from various sources to research topics based on the scope. Research allowed the development of RhyAn recommends and it's core functions. With appropriate sources literature was gathered on existing efforts on the subject and approaches taken. Other research was done on suitable methods required for the operation of RhyAn.

9.3.2.3 Requirement Specification

System Requirement Specification document was prepared. Requirements were solidified with the help of use case diagrams and stakeholder analysis, thus allowing the requirements of the project to be relevant and in-scope. With these methods the functional and non-functional requirements were defined.

9.3.2.4 Design of the System

The RhyAn and RhyAn recommends prototype was designed with the aid of the requirement specification and the literature review.

9.3.2.5 Implementation

Suitable technologies and approaches were discussed and selected in the implementation process. With mind of the high-level and low-level design decisions discussed in the design chapter withholding the functional requirements required.

9.3.2.6 Testing

Qualitative aspects of the project was tested with a suitable set of data and a test method was devised for testing audio signals and their patterns.

9.3.2.7 Evaluation

The project was taken under review and evaluated in each and every sections by the author to make sure that the objectives and the aim is archived and the results are desirable by performing a self-evaluation.

9.4 Utilization of course module knowledge

The knowledge required for this thesis partly comes from knowledge acquired from modules studied. From object oriented programming to web programming is used to archiving this project's aim.

9.4.1 Object Oriented Programming

Object Oriented Programming modules one and two has had an immense impact on the project and the author's structuring of programs, As with the knowledge acquired from this module has allowed the author to structure, compartmentalize and minimize excessive code reuse.

9.4.2 Advanced Client-side Web Development

Knowledge acquired from the "Advance Client-Side Web Development" module has helped the author to lay grounds in exploring new JavaScript libraries such as React.js

9.5 Use of existing Skill

Existing skill acquired from course material and experience in the field has helped the achievement of the thesis project.

- React Front End Development – Author's experience in ES6 and JavaScript backed by field work done during the internship and personal project.
- Digital Signal Processing – Previous project involving audio have helped the author to have a basic understanding on DSP to help develop and approach for the problem at hand in this thesis.

9.6 Learning Outcomes

The author was exposed to many areas of research and study areas in the course of this project, some directly relating to the research topic some not.

- Knowledge gained in digital signal processing and It's key and core concepts required for signal analysis.
- Author was exposed to Python's vast scientific libraries such as Matplotlib, SciPy, NumPy, etc.

9.7 Limitations

This section will talk about the limitations of the project which were caused by certain constraints set up for the thesis.

9.7.1 Scope Limitations

The scope was limited to rhythmic analysis, but other audio semantic methods may benefit and enhance RhyAn performance and RhyAn recommends accuracy. One such method would be the use of timbre similarity. With the limitation on rhythm tracking, RhyAn recommends solely depends on the rhythm data but will certainly gain more accuracy by incorporating other methods of semantic and other recommendation methods.

9.7.2 Time Constraints

With the set time constraints and allocated time for the implementation, good to have features were dropped in requirement specification as to allocate more time for the RhyAn core features thus leaving less time into the polishing of the user interface and user experience.

With the limited time, less time was allocated to the component filter, which requires more work to get it's accuracy to an acceptable rate.

9.8 Future Enhancements

RhyAn can be further extended to archive better accuracy and better recommendation results with the further listed enhancements below.

9.8.1 Timbre Analysis

As RhyAn is capable of isolating different percussive components, a promising further enhancement is implementing a timbre analysis component. Which will allow to analyze similarities between different percussive components in comparison IE – acoustic drum kit, electric drum kit. Further RhyAn can leverage the use of timbre analysis to reduce in correct on-set detentions thus increasing accuracy.

9.8.2 Time-measure estimation

Another mean of reducing noise and avoiding incorrect On-Set hits is by estimating the musical measures and estimating the time of the next percussive hit. As most of the time how low accuracy ridden RhyAn would be the low frequency percussion is detected without error thus allowing calculating the bpm and by sub-dividing the measures the accuracy of a certain on-set can be enforced therefore increasing accuracy.

9.9 Concluding Remarks

The author firmly believes that semantic approach for music discovery is the next logical step and in result was motivated to undertake this project in an interesting but unfamiliar territory.

Current Iteration of RhyAn Recommends may not be perfect but the author stands by it with passion and believes it's a step in the right direction. With more refining and fine tuning with implementation of further enhancements RhyAn can be integrated with existing music recommendation systems to get every artists vision to interested users as quick as possible.

10 References

Barrington, L., Yazdani, M., Turnbull, D., & Lanckriet, G. R. G. (2008). Combining Feature Kernels for Semantic Music Retrieval. In *ISMIR* (pp. 614–619). Retrieved from https://pdfs.semanticscholar.org/e7e0/f9b6092caab92b315bc8b36a3f8f81fdc3c9.pdf

Bello, J. P., Daudet, L., Abdallah, S., Duxbury, C., Davies, M., & Sandler, M. B. (2005). A Tutorial on Onset Detection in Music Signals. *IEEE Transactions on Speech and Audio Processing*, *13*(5), 1035–1047. https://doi.org/10.1109/TSA.2005.851998

Blogs.cornell.edu. (n.d.). Last.fm – Music Recommendation incorporating social network ties and collaborative filtering. Retrieved from http://blogs.cornell.edu/info2040/2012/09/20/last-fm-music-reccomendation-incorporating-social-network-ties-and-collaborative-filtering/

Fitzgerald, D. (2010). Harmonic/percussive separation using median filtering.

Foote, J., & Uchihashi, S. (2001). THE BEAT SPECTRUM: A NEW APPROACH TO RHYTHM ANALYSIS. *ICME 2001*. Retrieved from http://rotorbrain.com/foote/papers/icme2001.pdf

Goto, M. (2001). An Audio-based Real-time Beat Tracking System for Music With or Without Drum-sounds. *Journal of New Music Research*, *30*(2), 159–171. https://doi.org/10.1076/jnmr.30.2.159.7114

International Federation of the Phonographic Industry. (2016). Global Music Report 2016. Retrieved from http://www.ifpi.org/downloads/GMR2016.pdf

Nielsen. (2016). NIELSEN MUSIC YEAR-END REPORT U.S. 2016. Retrieved from www.nielsen.com/us/en/insights/reports/2017/2016-music-us-year-end-report.html

Padial, J., & Goel, A. (n.d.). Music Mood Classification. CS.

Ricci, F., Rokach, L., & Shapira, B. (2015). *Recommender systems handbook*. Springer.

Salvador, S., & Chan, P. (2007). Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, *11*(5), 561–580. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.432.4253&rep=rep1&type=pdf#page=64

Van den Oord, A., Dieleman, S., & Schrauwen, B. (2013). Deep content-based music recommendation. In *Advances in neural information processing systems* (pp. 2643–2651).

Yoshii, K., Komatani, K., Ogata, T., Okuno, H. G., & Goto, M. (n.d.). An Error Correction Framework Based on Drum Pattern Periodicity for Improving Drum Sound Detection. In *2006 {IEEE} International Conference on Acoustics Speed and Signal Processing Proceedings*. IEEE. https://doi.org/10.1109/icassp.2006.1661256

A Appendices

A.A Use Case Descriptions

A.A.A. Manage Library

Use Case	Manage Library	
Id	001	
Description	Manage files and metadata on the music entries.	
Priority Level	Medium	
Primary Actor	Administrator	
Supporting Actors	-	
Stakeholder and interests	-	
Pre-condition	-	
Triggering event	Administrator accessing management portal	
Main success event	**Actor**	**System**
	-	-
Alternative scenario	-	
Exception points	-	
Inclusions	Analyze Track	
Post conditions	**Successful**	**Unsuccessful**

A.A.B. Analyze Track

Use Case	Analyze Track	
Id	002	
Description	Perform the rhythmic analysis on the track.	
Priority Level	High	
Primary Actor	Administrator	
Supporting Actors	-	
Stakeholder and interests	-	
Pre-condition	Music should be in the library	
Triggering event	New track added to the library.	
Main success event	**Actor**	**System**

	-	Submit results to library.
Alternative scenario	-	
Exception points	-	
Inclusions	-	
Post conditions	**Successful**	**Unsuccessful**
	Submit results to library.	-

A.A.C. Browse library

Use Case	Browse library	
Id	003	
Description	Allow the user to browse through the library using metadata.	
Priority Level	Medium	
Primary Actor	User	
Supporting Actors	-	
Stakeholder and interests	-	
Pre-condition	Music should be in the library	
Triggering event	-	
Main success event	**Actor**	**System**
	-	Submit results to library.
Alternative scenario	-	
Exception points	-	
Inclusions	Manage library, Play track	
Post conditions	**Successful**	**Unsuccessful**
	Play track	-

A.A.D. Play Track

Use Case	Play Track
Id	004
Description	Allow the user to play a selected track.
Priority Level	Medium
Primary Actor	User
Supporting Actors	-
Stakeholder and interests	-

Pre-condition	Track should be available.	
Triggering event	-	
Main success event	**Actor**	**System**
	-	-
Alternative scenario	-	
Exception points	-	
Inclusions	Get Suggestions	
Post conditions	**Successful**	**Unsuccessful**
	Generate recommendations	-

A.A.E. Get Suggestions

Use Case	Get Suggestions	
Id	005	
Description	Generate suggestions using the rhythmic information of the current track	
Priority Level	High	
Primary Actor	User	
Supporting Actors	-	
Stakeholder and interests	-	
Pre-condition	Track should have rhythmic information metadata.	
Triggering event	After a track is done playing	
Main success event	**Actor**	**System**
	Give suggestions	-
Alternative scenario	-	
Exception points	-	
Inclusions	-	
Post conditions	**Successful**	**Unsuccessful**
	Play track	-

A.B Gantt Chart

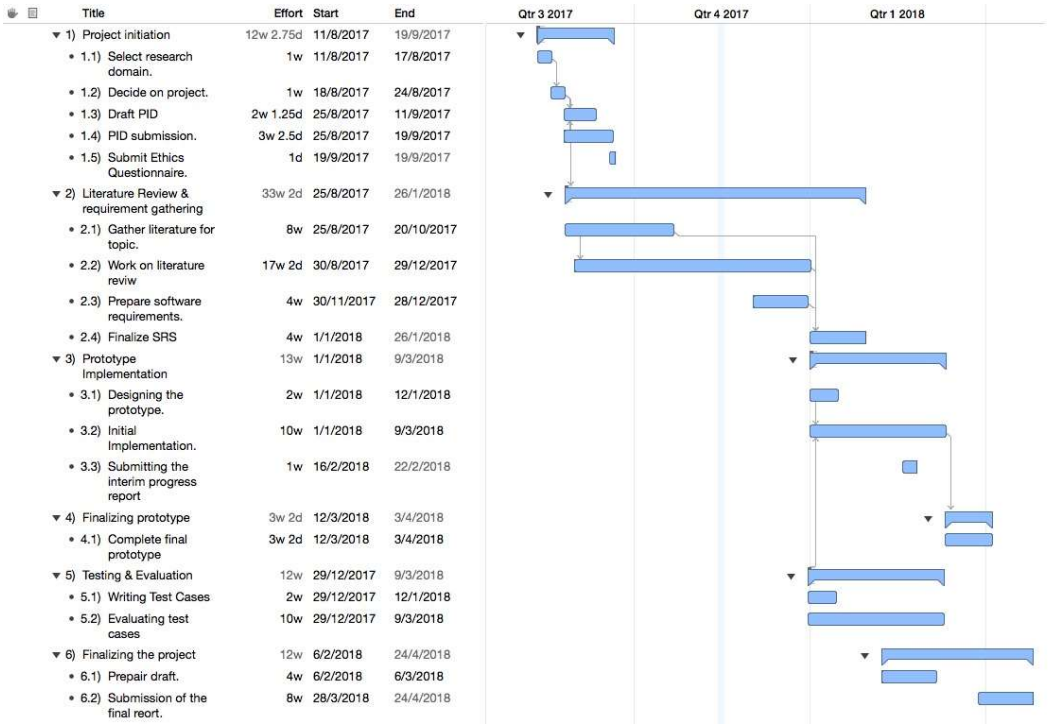


Illustration 28: Gantt chart

A.C Track Selection

Blackstreet_-_Take_Me_There

Bloodhound_Gang_-_The_Bad_Touch

Earth_Wind_and_Fire_-_Let's_Groove

Dan_Fogelberg_-_Same_Old_Lang_Syne

England_Dan_-
_I'd_Really_Love_to_See_You

Celine_Dion_-_To_Love_You_More

Barry_Manilow_-_Mandy

Diana_Ross_-
_Touch_Me_in_the_Morning

Backstreet_Boys_-_Shape_of_My_Heart

Boyzone_-_No_Matter_What

Chaka_Khan_-
_Tell_Me_Something_Good

ABBA_-_Name_of_the_Game

George_Benson_-
_Turn_Your_Love_Around

Boyzone_-_Words

Backstreet_Boys_-
_Show_Me_the_Meaning

Christopher_Cross_-_Sailing

Enrique_Iglesias_-_Bailamos

George_Michael_-_Faith

Boyzone_-_You_Needed_Me

ABBA_-_Fernando

All_Saints_-_Never_Ever

Cardigans_-_Erase_And_Rewind

ABBA_-_The_Winner_Takes_It_All

George_Benson_-_Give_Me_the_Night

Backstreet_Boys_-
_Quit_Playing_Games_With_My_Heart

Backstreet_Boys_-_Back_to_Your_Heart

Eels_-_Last_Stop_This_Town

Connie_Francis_-_Where_the_Boys_Are

Barbra_Streisand_-_The_Way_We_Were

Chumbawamba_-_Tubthumping

Eels_-_Three_Speed

Boyzone_-
_Baby_Can_I_Hold_You_Tonight

Gary_Puckett_-_Young_Girl

Bobby_Vinton_-_My_Special_Angel

Earth_Wind_and_Fire_-_Fantasy

Chris_Isaak_-_Wicked_Game

Curtis_Mayfield_-_Move_On_Up

Backstreet_Boys_-
_As_Long_As_You_Love_Me

Bee_Gees_-
_How_Can_You_Mend_a_Broken_Heart

Chicago_-_25_or_6_to_4

Blur_-_Country_House

Carole_King_-
_Will_You_Still_Love_Me_Tomorrow

George_Benson_-_This_Masquerade

Chaka_Khan_-_I_Feel_for_You

Britney_Spears_-
_What_U_See_I_What_U_Get

Backstreet_Boys_-
_No_One_Else_Comes_Close

Christina_Aguilera_-_I_Turn_To_You

Fastball_-_The_Way

5ive_-_If_Ya_Gettin_Down

Captain_and_Tenille_-
_You_Never_Done_It_Like_That

Bjork_-_Joga

ABBA_-_Knowing_Me_Knowing_You

Cake_-_Frank_Sinatra

Chicago_-_Wishing_You_Were_Here

Bee_Gees_-_You_Should_Be_Dancin'

Christopher_Cross_-
_Ride_Like_the_Wind

Bobby_Vinton_-_Blue_Velvet

Dionne_Warwick_-_Heartbreaker

Everclear_-_Everything_To_Everyone

Bee_Gees_-_Jive_Talkin'

Bette_Midler_-_From_A_Distance

Brandy_-_Almost_Doesn't_Count

Earth_Wind_and_Fire_-
_After_the_Love_Has_Gone

Blood_Sweat_and_Tears_-
_Spinning_Wheel

Earth_Wind_and_Fire_-
_That's_the_Way_of_the_World

Boyzone_-
_I_Love_The_Way_You_Love_Me

5ive_-_You_Got_The_Feelin

Chuck_Mangione_-_Feels_So_Good
En_Vogue_-_Free_Your_Mind
Barenaked_Ladies_-_One_Week
Bjork_-_Human_Behavior
Backstreet_Boys_-
_Let_the_Music_Heal_Your_Soul
Chicago_-_Saturday_in_the_Park
Carole_King_-_Natural_Woman
Desree_-_You_Gotta_Be
Boz_Scaggs_-_We're_All_Alone
Billy_Joel_-_Uptown_Girl
Frankie_Valli_-
_Cant_Take_My_Eyes_Off_Of_Yo
ABBA_-_Waterloo
Backstreet_Boys_-_The_One
Celine_Dion_-
_My_Heart_Will_Go_On_(Techno_Remi
x)
Bee_Gees_-_Too_Much_Heaven
Bonnie_Tyler_-_It's_a_Heartache
Carole_King_-_I_Feel_the_Earth_Move
Barry_Manilow_-_Even_Now
Blur_-_Charmless_Man
Britney_Spears_-_Stronger
Billy_Joel_-_River_of_Dreams
Barenaked_Ladies_-_Shoe_Box
Belinda_Carlisle_-_Mad_About_You
B'Witched_-
_Blame_It_On_The_Weatherman
98_Degrees_-_I_Do_Cherish_You
Carly_Simon_-_Attitude_Dancing
Britney_Spears_-_Soda_Pop
Chicago_-_Color_My_World
Gary_Wright_-_Dream_Weaver
Bee_Gees_-_Stayin_Alive
Backstreet_Boys_-
_We've_Got_It_Goin_On
Diana_Ross_-_Upside_Down
Blackstreet_-_No_Diggity
Captain_and_Tenille_-
_Do_That_To_Me_One_More_Time
George_Michael_-_One_More_Try
Dionne_Warwick_-
_That's_What_Friends_Are_For
Destinys_Child_-_Bugaboo
Brian_McKnight_-_Back_At_One
Bjork_-_Hyper_Ballad
"Backstreet_Boys_-_10 000_Promises"
Bee_Gees_-_How_Deep_Is_Your_Love
Cake_-_Never_There
Cranberries_-_Zombie
98_Degrees_-_The_Hardest_Thing
Gary_Puckett_-_Woman_Woman
Britney_Spears_-
_You_Drive_Me_Crazy_(Remix)
Everclear_-
_I_Will_Buy_You_A_New_Life
Faithless_-_God_Is_A_DJ
Babyface_-_The_Day
Boyzone_-_Going_Gets_Tough
Bee_Gees_-_Love_You_Inside_Out
George_Michael_-_Father_Figure

Geri_Halliwell_-_Look_At_Me
Boz_Scaggs_-_What_Can_I_Say
ABBA_-_Dancing_Queen
Culture_Club_-_Karma_Chameleon
Cyndi_Lauper_-_She_Bop
Cyndi_Lauper_-_Time_After_Time
Captain_and_Tenille_-
_Love_Will_Keep_Us_Together
Barenaked_Ladies_-
_The_Old_Apartment
Chicago_-_Feeling_Stronger_Every_Day
Blaque_-_Bring_It_All_To_Me
Everclear_-_Santa_Monica
Eels_-_Your_Lucky_Day_In_Hell
Blur_-_Song_2
5ive_-_Slam_Dunk_Da_Funk
Bjork_-_Bachelorette
ABBA_-_Money_Money_Money
Dionne_Warwick_-
_I_Say_A_Little_Prayer
Anne_Murray_-_Snowbird
Bill_Withers_-_Lean_on_Me
Billy_Joel_-_Goodnight_Saigon
Bee_Gees_-_Tragedy
En_Vogue_-_Dont_Let_Go
Alan_O_Day_-_Undercover_Angel
B'Witched_-_Jump_Down
Bonnie_Tyler_-
_Holding_Out_for_a_Hero
All_Saints_-_Bootie_Call
Blur_-_Tender
Boyzone_-_Picture_Of_You
Eve_6_-_Inside_Out
Chicago_-_Beginnings
Eagle_Eye_Cherry_-_Save_Tonight
George_Benson_-_On_Broadway
Aaliyah_-_Are_You_That_Somebody
All_Saints_-_Lady_Marmelade
Christina_Aguilera_-_Come_On_Over
Bee_Gees_-_Medley
Ben_Folds_Five_-_Brick
Cardigans_-_My_Favourite_Game
Babyface_-
_Every_Time_I_Close_My_Eyes
Dionne_Warwick_-
_I'll_Never_Love_This_Way_Again
Deep_Blue_Something_-
_Breakfast_At_Tiffanys
Celine_Dion_-_Falling_Into_You
Boyz_2_Men_-_A_Song_For_Mama
Britney_Spears_-_You_Drive_Me_Crazy
Bertie_Higgins_-_Key_Largo
Boz_Scaggs_-_Lido_Shuffle
A-Teens_-_Mama_Mia
Desree_-_Life
Cyndi_Lauper_-
_Girls_Just_Want_to_Have_Fun
Cher_-_Believe
Bjork_-_Venus_As_A_Boy
Boyz_2_Men_-_Ill_Make_Love_To_You
Earth_Wind_and_Fire_-_September
Celine_Dion_-_All_By_Myself

Bloodhound_Gang_-
_The_Bad_Touch_(Eiffel65_Mix)
Blood_Sweat_and_Tears_-
_And_When_I_Die
Enrique_Iglesias_-_Be_With_You
Backstreet_Boys_-_Larger_Than_Life
Donny_Osmond_-_Puppy_Love
Enrique_Iglesias_-_Experiencia_Religiosa
Aqua_-_My_Oh_My
Backstreet_Boys_-_Everybody
ABBA_-_Thank_You_for_the_Music
5ive_-_Keep_On_Moving
Bee_Gees_-_I_Started_A_Joke
Don_McLean_-_Vincent
Cyndi_Lauper_-_True_Colors
Billy_Joel_-_You_May_Be_Right
Celine_Dion_-_Immortality
En_Vogue_-_My_Lovin
Bette_Midler_-_The_Rose
Bill_Withers_-_Just_the_two_of_us
George_Michael_-_Freedom
Billy_Joel_-_My_Life
Billy_Joel_-_Honesty
Billie_Myers_-_Kiss_The_Rain
Dionne_Warwick_-_Walk_On_By
Classics_IV_-_Traces_of_Love
Debbie_Gibson_-_Foolish_Beat
Bill_Withers_-_Aint_No_Sunshine
Boyzone_-_Love_Me_For_A_Reason
Faith_Evans_-_Love_Like_This
Ben_Folds_Five_-
_Steven's_Last_Night_in_Town
Brandy_-_I_Wanna_Be_Down
BJ_Thomas_-_Hooked_on_a_Feeling
Blur_-_Girls_Boys
Britney_Spears_-_Baby_One_More_Time
Ace_of_Base_-
_Always_Have_Always_Will
Five_Man_Electrical_Band_-_Signs
5ive_-_Until_The_Time_Is_Through
Carole_King_-_So_Far_Away
Brian_McKnight_-_Anytime
Celine_Dion_-
_It's_All_Coming_Back_to_Me
Diana_Ross_-_Theme_from_Mahogany
Billy_Joel_-_Only_the_Good_Die_Young
Classics_IV_-_Stormy
Aretha_Franklin_-_Pink_Cadillac
Barbra_Streisand_-_Woman_In_Love
Chaka_Khan_-_Ain't_Nobody
Cyndi_Lauper_-_I_Drove_All_Night
Billy_Joel_-_The_Longest_Time
Boz_Scaggs_-_Low_Down
Destinys_Child_-_Bills_Bills_Bills
Carly_Simon_-_Nobody_Does_it_Better
702_-_Where_My_Girls_At
ABBA_-_SOS
BB_Mak_-_Back_Here
Britney_Spears_-
_Britney_Spears_Medley
Elton_John_-_Candle_in_the_Wind

Dusty_Springfield_-
_Son_of_a_Preacher_Man
Britney_Spears_-
_Don't_Let_Me_Be_the_Last_to_Know
Backstreet_Boys_-_Like_A_Child
Backstreet_Boys_-_I_Want_It_That_Way
Cher_-_Shoop_Shoop_Song
Aretha_Franklin_-_Respect
Enya_-_Bard_Dance
Boyzone_-_All_That_I_Need
Bloodhound_Gang_-_Fire_Water_Burn
Celine_Dion_-_I'm_Your_Angel
All_4_One_-_I_Swear
Everclear_-_Father_Of_Mine
Blood_Sweat_and_Tears_-
_You've_Made_Me_So_Very_Happy
Culture_Club_-
_Do_You_Really_Want_to_Hurt_Me
Dionne_Warwick_-
_Do_You_Know_the_Way_to_San_Jose
Boyzone_-_A_Different_Beat
B'Witched_-_Jesse_Hold_On
George_Benson_-_Breezin'
Chicago_-_Does_Anybody_Really_Know
Dusty_Springfield_-
_You_Dont_Have_To_Say_You_Lo
All_Saints_-_I_Know_Where_It's_At
Gary_Puckett_-_Lady_Willpower
Bee_Gees_-_Massachusetts
Dusty_Springfield_-
_I_Only_Want_To_Be_With_You
Alannah_Myles_-_Black_Velvet
Billy_Joel_-_Tell_Her_About_It
Bjork_-_Oh_So_Quiet
Bee_Gees_-_Words
5ive_-_Dont_Wanna_Let_You_Go
Edwin_Starr_-_War
Ace_of_Base_-_All_That_She_Wants
Dionne_Warwick_-
_I'll_Never_Fall_in_Love_Again
Andrew_Lloyd_Webber_-
_Jesus_Christ_Superstar
Bee_Gees_-_To_Love_Somebody
B'Witched_-_To_You_I_Belong
Classics_IV_-_Spooky
Bonnie_Tyler_-
_Total_Eclipse_of_the_Heart
Christopher_Cross_-_Arthurs_Theme
Fastball_-_Out_Of_My_Head
A-Teens_-_Super_Trouper
Gary_Wright_-_Love_is_Alive
Bruce_Hornsby_-_The_Way_It_Is
Barry_Manilow_-_Copacabana
Billy_Joel_-_Just_the_Way_You_Are
B'Witched_-_Cest_La_Vie
Cranberries_-_Linger
Blessed_Union_Of_Souls_-_I_Believe
Britney_Spears_-_Sometimes
Andrew_Lloyd_Webber_-
_Phantom_of_the_Opera
Carly_Simon_-_No_Secrets
Aretha_Franklin_-_Think

Blur_-_Ecco_Song

Cardigans_-_Lovefool

Ace_of_Base_-_Don't_Turn_Around

Bobby_McFerrin_-_Dont_Worry_Be_Happy

ABBA_-_Take_A_Chance_On_Me

Carly_Simon_-_Thats_the_Way_I've_Always_Heard_It_Should_Be

Arbeau_Thoinot_-_Branle_des_cheveaux

Bee_Gees_-_More_Than_A_Woman

Celine_Dion_-_Tell_Him

Backstreet_Boys_-_I'll_Never_Break_Your_Heart

Billy_Joel_-_Big_Shot

Billy_Joel_-_It's_Still_Rock_and_Roll_to_Me

ABBA_-_Chiquita

Cyndi_Lauper_-_All_Through_the_Night

Frankie_Valli_-_December_1963

Ace_of_Base_-_Beautiful_Life

Everlast_-_What_Its_Like

Destinys_Child_-_Say_My_Name

Carole_King_-_You've_Got_A_Friend

Britney_Spears_-_Lucky

ABBA_-_Alaska

George_Benson_-_Make_Love

En_Vogue_-_Never_Gonna_Get_It

Belinda_Carlisle_-_Circlein_the_Sand

Anne_Murray_-_You_Needed_Me

Eels_-_The_Medication's_Wearing_Off

Culture_Club_-_Church_of_the_Poison_Mind

Aqua_-_Barbie_Girl

Carly_Simon_-_Mockingbird

Bee_Gees_-_Nights_On_Broadway

Celine_Dion_-_Power_of_Love

Cyndi_Lauper_-_Goonies_R_Good_Enough

Another_Level_-_Freak_Me

ABBA_-_Super_Trouper

Aretha_Franklin_-_You_Make_Me_Feel_Like_a_Natural_Woman

Carly_Simon_-_Half_a_Chance

Cherry_Poppin_Daddies_-_Zoot_Suit_Riot

Celine_Dion_-_Because_You_Loved_Me

Blessed_Union_Of_Souls_-_Hey_Leonardo

Aaliyah_-_Try_Again

Britney_Spears_-_Born_to_Make_You_Happy

Earth_Wind_and_Fire_-_Shining_Star

Another_Level_-_I_Want_You_For_Myself

BJ_Thomas_-_Raindrops_Keep_Falling

Belinda_Carlisle_-_Heaven_Is_a_Place_on_Earth

Bobby_Vinton_-_Sealed_With_a_Kiss

Cake_-_The_Distance

Don_McLean_-_American_Pie

Brandy_-_Sittin'_Up_In_My_Room
Debby_Boone_-_You_Light_Up_My_Life
Backstreet_Boys_-_All_I_Have_to_Give
Blood_Sweat_and_Tears_-
_I_Can't_Quit_Her
Boyzone_-_Fathers_And_Sons
Frankie_Valli_-_Grease
Captain_and_Tenille_-
_The_Way_I_Want_To_Touch_You
Bette_Midler_-
_The_Wind_Beneath_My_Wings
Boyz_2_Men_-_End_Of_The_Road
Connie_Francis_-_Stupid_Cupid
Bee_Gees_-_Night_Fever
Aqua_-_Lollipop_Candy_Man
Destinys_Child_-_Jumpin_Jumpin
Brandy_-_Best_Friend
Bee_Gees_-
_I_Just_Want_to_Be_Your_Everything
Christina_Aguilera_-_Genie_In_A_Bottle
Backstreet_Boys_-_Anywhwere_for_You
Billy_Joel_-_New_York_State_of_Mind
Backstreet_Boys_-
_My_Heart_Stays_With_You
Ace_of_Base_-_The_Sign
Babyface_-_Change_the_World
Billy_Joel_-_The_Stranger
ABBA_-_Gimme_Gimme_Gimme
Brandy_-_The_Boy_Is_Mine
Debbie_Gibson_-_Lost_in_your_Eyes
Carole_King_-_It's_Too_Late
Earth_Wind_and_Fire_-_Sing_a_Song
ABBA_-_Gimme_Gimme_Gimme.mid
ABC_-_Poison_Arrow.mid
ABC_-_The_Look_of_Love.mid
Ace_of_Base_-_All_That_She_Wants.mid
Aha_-_Hunting_High_And_Low.mid
Aha_-_Take_On_Me.mid
Aha_-
_The_Sun_Always_Shines_On_TV.mid
Andrew_Lloyd_Webber_-
_Phantom_of_the_Opera.mid
Bananarama_-_Venus.mid
Barbra_Streisand_-_Woman_In_Love.mid
Barry_Manilow_-_Copacabana.mid
Belinda_Carlisle_-
_Heaven_Is_a_Place_on_Earth.mid
Billy_Idol_-_White_Wedding.mid
Billy_Joel_-_Piano_Man.mid
Billy_Joel_-_Uptown_Girl.mid
BJ_Thomas_-_Hooked_on_a_Feeling.mid
Blur_-_Girls_Boys.mid
Blur_-_Song_2.mid
Bobby_Vinton_-_Blue_Velvet.mid
Cher_-_Believe.mid
Corey_Hart_-_Sunglasses_at_Night.mid
Cranberries_-_Zombie.mid
Cyndi_Lauper_-_Time_After_Time.mid
Earth_Wind_and_Fire_-_September.mid
Frank_Sinatra_-
_New_York,_New_York.mid
George_Michael_-_Faith.mid

Glen_Campbell_-
_By_the_Time_I_Get_to_Phoenix.mid
Glen_Campbell_-_Wichita_Lineman.mid
Guns_n_Roses_-_November_Rain.mid
INXS_-_Never_Tear_Us_Apart.mid
Jethro_Tull_-_Aqualung.mid
Kate_Bush_-_Wuthering_Heights.mid
Michael_Jackson_-_Rock_with_You.mid
Nirvana_-_About_A_Girl.mid
Nirvana_-_Floyd_the_Barber.mid
Nirvana_-_In_Bloom.mid
Nirvana_-_School.mid
Nirvana_-_Something_In_The_Way.mid
Nirvana_-_Territorial_Pissings.mid
REM_-_Everybody_Hurts.mid
REM_-_Losing_My_Religion.mid
REM_-_Man_on_the_Moon.mid
REM_-_Shiny_Happy_People.mid
Tears_For_Fears_-
_Sowing_the_Seeds_of_Love.mid
The_Bangles_-_Eternal_Flame.mid
The_Carpenters_-_Close_to_You.mid
The_Carpenters_-
_Top_of_the_World.mid
The_Corrs_-_Radio.mid
The_Cure_-_Boys_Don't_Cry.mid
The_Moody_Blues_-
_Nights_in_White_Satin.mid