TRIBHUVAN UNIVERSITY

INSTITUE OF ENGINEERING

CENTRAL CAMPUS, PULCHOWK

FINAL PROJECT REPORT

ON

Music Classification Based on Genre and Mood

Part-A

Submitted By:

Ayush Shakya	[43155]
Bijay Gurung	[43162]
Mahendra Singh Thapa	[43169]
Mehang Rai	[43174]

Submitted To:

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

LALITPUR, NEPAL

(april 28, 2016)

COPYRIGHTS

ACKNOWLEDGMENT

First and foremost, we would like to express our sincere gratitude to Dr. Basanta Joshi, our supervisor for this major project. Without his invaluable guidance and suggestions, it would have been a difficult journey for us. His useful suggestions for this whole work and cooperative behavior are sincerely acknowledged.

We would like to thank the Department of Electronics and Computer Engineering for adding this major project as part of final year curriculum and hence giving us this opportunity to undertake the project. The great need of research, time and sheer coding has allowed us to harness our skills, experience and knowledge.

We are also grateful to Manoj Ghimire and Prof.Dr.Sashidhar Ram Joshi and our all respective subject teachers of for their constant support and guidance.

Last but not the least, we would like to thank our friends for their numerous assistance throughout the project development duration.

ABSTRACT

This progress report details the work completed and remaining work to be done for the final year project titled Music Classification System Based on Genre and Mood as of the end of the mid-term session. The project is part for the curriculum for the subject Major Project under the course of final year of B.E. in Computer Engineering.

The overall aim of the project is to develop a system capable of classifying music based on Genre and mood with the availability of large number of digital media and the disorder introduced being the primary motivation.

The methodology used is that of a modular system consisting of two main stages. The first stage involves the preprocessing of the raw audio data resulting in the extraction of a number of features: Intensity, MFCC, rhythm, pitch. Each feature extractor reduces the information content in the raw data to a vector in a small number of dimensions.

In the second stage, the set of feature vectors are classified(indexed) into certain clusters by the use of certain algorithms: K-means and Artificial Neural Networks.

The work completed includes feature extraction of Intensity, MFCC and rhythmic features and the implementation of the K-means Clustering Algorithm. Currently, the accuracy of the system (46 per cent) is pretty bad but it is sure to be improved further in the coming days.

The work to be done includes mood based classification, implementation of the ANN and extraction of the Pitch feature. Apart from those tasks, we will also be optimizing the parts already completed and also analyzing and validating the individual components and the overall results.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 Background

Due to the rapid development in the music industry, there has been an increasing amount of work in the area of automatic genre classification of music in audio format. One of the prime area focused on by MIR is the Automatic Classification of such music based on signal analysis. Such systems can be used as a way to evaluate features describing music content as well as a way to structure large collections of music.

The presence of huge amount of data in music databases can be a perfectly applicable area. Music is complex in nature not only due to its origination but also due to the evolution of music in the technological era. So it requires specialized representations, abstraction and processing techniques for effective analysis, evaluation and classification that are fundamentally different form those used for other mediums and tasks.

1.2 Motivation

The presence of numerous genre is a source of confusion and more often than not people are overwhelmed with the sheer vastness of music available. As such, the primary motivation is to make it easier for people to classify music (based on genre and/or mood) so that they can find songs suited to their own tastes.

It can also lay the foundation for figuring out ways to represent similarity between two musical pieces and in the making of a good recommendation system.

1.3 Problem Statement

Though the project is focused on music classification based genre and mood, we focused only on consolidating the genre part for this mid term milestone. The mood based classification has been left for the next semester. Although the number of genres present worldwide is huge, we have restricted ourselves to five major genres which are: Rock, Pop, Classical, Jazz and Hiphop.

The choice of these genres is based on their being sufficiently distinguishable from each other. Choosing some genre thats very unique and abnormal might have made them more distinguishable and easier to classify but it would have been harder to find quality data/works for those genre. So, we chose these genre with availability of musical pieces in mind too.

1.4 Objectives

i.) To study and implement different preprocessing steps involved in extracting features from audio data.

ii.) To implement suitable classification algorithm for various features of the song

iii.) To cross validate the result and analyze the efficiency of the algorithms used.

1.5 Significance and Scope

1.5.1 Significance

Music classification based on genre is the current topic on hype in the musical sector. Music Information Retrieval(MIR) has put on a huge interest in this sector. The availability of huge music databases can be well organized by use of such classification system.

1.5.2 Scope of work

i) The project will work on classifying music based on genre and mood. More specifcally, the classifcation will be done on western music only as the data is more easily available and lots of works have been done in the past for it. Also, only five genres will be used for genre classifcation:

- Rock
- Pop
- Classical
- Jazz
- Hiphop

ii) The mood based classifcation will use the Thayer model, a two dimensional model based on Energy and Stress:

- High Energy, High Stress = Anxious/Frantic

- High Energy, Low Stress = Exuberance
- Low Energy, High Stress = Depression
- Low Energy, Low Stress = Contentment

iii) Also, it is entirely possible for a song or a piece of music to fall into multiple genre or moods. The characteristics that defne the genre and the mood may change within the song itself with one part showing seeming to belong to one class while other parts may seem to belong to an entirely diferent class. The project will not cover such issues. In other words, multiple-tagging will not be done.

iv) The classification will work only on mp3 files. No other fle format will be supported in the initial release/version.

2 LITERATURE REVIEW

2.1 Human Audio Perception

The human ear is an exceedingly complex organ. To make matters even more difficult, the information from two ears is combined in a perplexing neural network, the human brain. [1]

Figure 1 illustrates the major structures and processes that comprise the human ear. The outer ear is composed of two parts, the visible flap of skinand cartilage attached to the side of the head, and the ear canal, a tubeabout 0.5 cm in diameter extending about three cm into the head. These structures direct environmental sounds to the sensitive middle and inner ear organs located safely inside of the skull bones. Stretched across the end of the ear canal is a thin sheet of tissue called the tympanic membrane or eardrum. Sound waves striking the tympanic membrane cause it to vibrate. The middle ear is a set of small bones that transfer this vibration to the cochlea (inner ear) where it is converted to neural impulses. The cochlea is a liquid filled tube roughly two mm in diameter and three cm in length.

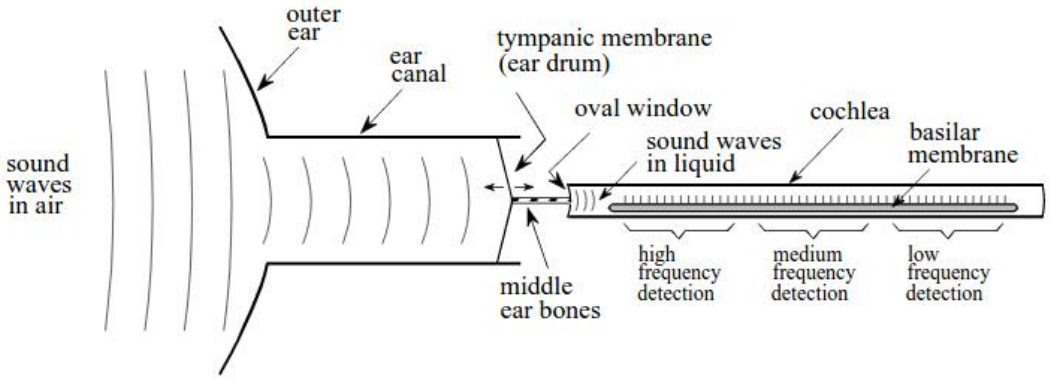

Figure 1: Functional Diagram of Human Ear

Music can be defined as organised sound comprising the following structural elements: pitch, timbre, key, harmony, loudness (or amplitude), rhythm, meter, and tempo. Processing these elements involves almost every region of the brain and nearly every neural subsystem.

Sound does not exist outside of the brain; it is simply air molecules moving. Sound is produced by vibrating air molecules connecting with the eardrum at varying frequencies (pitch) and velocities (amplitude). The process starts with the brains primary auditory cortex receiving a signal from the eardrum/inner ear which immediately activates our primitive brain, the cerebellum. The cerebellum is the oldest part of the brain in evolutionarily terms

and plays an important part in motor control. It contributes to coordination, precision, and accurate timing of movements. The ear and the primitive brain are known collectively as the low-level processing units. They perform the main feature extraction which allows the brain to start analysing the sounds, breaking down the sensory stimulus into pitch, timbre, spatial location, amplitude, reverberant environment, tone durations, and onset times of different notes.

This data is conducted through neurons in the brain; cells specialized in transmitting information, and the basic building blocks of the nervous system. The output of these neurons connects to the high-level processing units located in the frontal lobe of the brain. It is important to note that this process is not linear. The different regions of the brain constantly update each other with new information.

2.2 Audio Processing

General Audio signal processing is an engineering field that focuses on the computational methods for intentionally altering sounds, methods that are used in many musical applications.

Particularly speaking, music signal processing may appear to be the junior relation of the large and mature field of speech signal processing, not least because many techniques and representations originally developed for speech have been applied to music, often with good results. However, music signals possess specific acoustic and structural characteristics that distinguish them from spoken language or other nonmusical signals. [2]

In music the most important qualities of sound are: pitch, duration, loudness, and timbre. Duration and loudness are unidimensional, while pitch and timbre are complex and multidimensional. [3]

- **Loudness** - Intensity of a tone is the physical correlate that underlies the perception of loudness. Loudness variations play an important role in music, but are less important than pitch variations.

- **Duration** - A composer or performer can alter the pace of a piece so that its apparent (virtual) time is slower or faster than clock time.

- **Timbre** - Timbre is the subjective code of the sound source or of its meaning. According to the American Standards Association, ”Timbre is that attribute of auditory

senstation of which a listener can judge that two steady-state tones having the same pitch and loudness are dissimilar."

- **Pitch** - Pitch is related to the frequency of a pure tone and to the fundamental frequency of a complex tone. In its musical sense, pitch has a range of about 20 to 5000 Hz. Some five to Zseven harmonics of a complex tone can be heard out individually by paying close attention. There is a dominance region for pitch perception, roughly from 500 to 2000 or 3000 Hz. Harmonics falling in the dominance region are most influential with regard to pitch.

These types of low dimensional features extracted from the acoustical signals are more popular than higher dimensional representations such as Spectrograms for Classification purposes. [4]

2.3 Classification

A variety of methods have been used for music classification. Some of the popular ones are SVM, K Nearest Neighbours and variants of Neural Networks.

The results are also widely different. In [5] 61 per cent accuracy has been achieved using a Multilayer Perceptron based approach while in [6], the authors have managed 95 per cent (for Back Propagation Neural Network) and 83 per cent (for SVM).

In [7], the authors have achieved 71 per cent accuracy using an additional rejection and verification stage.

In [8], simpler and more naive approaches (k-NN and k-Means); and more sophisticated neural networks and SVMs have been compared. The author found the latter gave better performance.

However, lots of unique methods – either completely novel or a variation of a standard method – have been put into use too. In [9], the authors propose a method that uses Chord labeling (ones and zeros) in conjunction with a k-windowSubsequenceMatching algorithm used to find subsequence in music sequence and a Decision tree for the actual genre classification.

It is also noted that high-level and contextual concepts can be as important as low-level content descriptors. [10]

2.4 Factors affecting accuracy

Some of the factors that affect the accuracy are:

(i) **Multi-tagging:** A song can belong to multiple genre. So it is sure to consist of features characterizing multiple genre. This might creating a problem for any classification technique applied as it is sure to create ambiguity.

(ii) **Noise:** Many of the songs may not be recorded in the studio. Some may be recorded during live music while some in concert. We are sure to find noise in the latter cases which may tamper the original signal of music hence giving a deviated feature vector. This is sure to affect the accuracy of classification system.

(iii) **Similar feature in different genre:** Some of the feature of different genres may somehow be similar in some aspects. For example: intensity of metal and rock are high, beat is also high in both, and so on.

3 METHODOLOGY

The literature review has exposed us to a variety of ways (some simple, some complex) used in the problem domain. In fact, no two papers had remotely similar methodologies; they varied in at least one of feature extraction, classification algorithm used or the Genre themselves. The features selected are mostly based on the results obtained in [11] where FFT, MFCC, Pitch and Beat were investigated and a combination of all those features were found to give the best results. Although SVM, K-NN, Variants of Neural Networks, were found to be the most commonly used algorithms. However, as we had decided to try out one unsupervised method and one supervised method, we were mostly interested in how K-means and ANN were used.

Also we were focusing on K-means for this mid-term, so we looked at the two papers [8], [12] that had made use of it. We also went over lots of other papers [14], [15], [16] related to the algorithm itself (not directly related to Music Classification). How the literature affected our methodology will be discussed in the sections pertaining to the methodologies used.

3.1 System Block Diagram

The main components of our system are:

- **Audio Sampling -** It makes use of MP3 decoder and OGG decoder to sample the audio.
- **Feature Extraction -** The relevant features are extracted in this stage.
- **Feature Integration -** The extracted features are integrated into a single JSON file.
- **Learning Algorithm -** The learning algorithm uses the feature file to create a model for classification.
- **Validation -** The last stage involves validating the results obtained.

The block diagram is as follows:

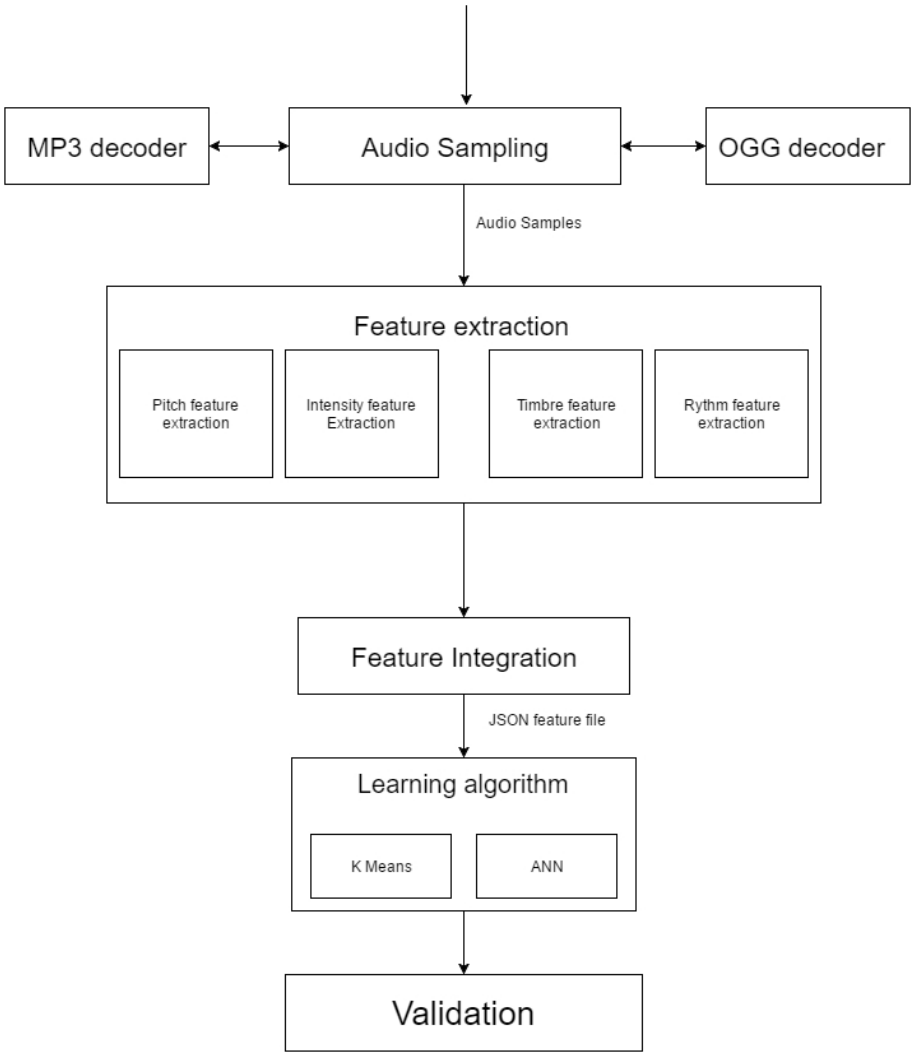

Figure 2: System Block Diagram

3.2 Data Collection

As we have chosen five genres for our classification ourselves, we decided to first collect data based on our own view. For that we will be using our own music collection and collect remaining from our friend.Then according to the result we will be opting for the standard database of music provided by sites like marsyas,etc.

3.3 Feature Extraction

3.3.1 Overview

Feature extraction is the process of computing a compact numerical representation that can be used to characterize a segment of audio. It is observed that the feature selection always

plays an important role in building a good pattern classifier. Once the features are extracted standard machine learning techniques which are independent of the specific application area can be used.

After considering many commonly used features, we decided on using the intensity, rhythm, timbral and pitch as the key features for our initial classification task.

For this mid-term, we have completed the extraction of the first three of the four features: intensity, rhythm and timbre (as MFCC).

3.3.2 Overview of JAVA sampled package and AudioSystem class

Thejavax.sound.sampledpackage is fundamentally concerned with audio transport in other words, the Java Sound API focuses on playback and capture. The central task that the Java Sound API addresses is how to move bytes of formatted audio data into and out of the system.

To make use of the Java Sound API, at least three things are necessary: formatted audio data, a mixer, and a line. The following provides an overview of these concepts.

a. **Data Formats -** A data format provides information on how to interpret a series of bytes of ”raw” sampled audio data, such as samples that have already been read from a sound file, or samples that have been captured from the microphone input. It has various use cases, for example, how many bits constitute one sample (the representation of the shortest instant of sound), and the sound’s sample rate (how fast the samples are supposed to follow one another). In the Java Sound API, a data format is represented by anAudioFormatobject, which includes the following attributes:

   - Encoding technique, usually pulse code modulation (PCM)
   - Number of channels (one for mono, two for stereo, etc.)
   - Sample rate (number of samples per second, per channel)
   - Number of bits per sample (per channel)
   - Frame rate
   - Frame size in bytes
   - Byte order (big-endian or little-endian)

   PCM is one kind of encoding of the sound waveform. The Java Sound API includes two PCM encodings that use linear quantization of amplitude, and signed or unsigned inte-

ger values. Linear quantization means that the number stored in each sample is directly proportional (except for any distortion) to the original sound pressure at that instantand similarly proportional to the displacement of a loudspeaker or eardrum that is vibrating with the sound at that instant.

A frame contains the data for all channels at a particular time. For PCM-encoded data, the frame is simply the set of simultaneous samples in all channels, for a given instant in time, without any additional information. In this case, the frame rate is equal to the sample rate, and the frame size in bytes is the number of channels multiplied by the sample size in bits, divided by the number of bits in a byte. For other kinds of encodings, a frame might contain additional information besides the samples, and the frame rate might be completely different from the sample rate. For the MP3 (MPEG-1 Audio Layer 3) encoding, which is not explicitly mentioned in the current version of the Java Sound API, but which could be supported by an implementation of the Java Sound API or by a third-party service provider. In MP3, each frame contains a bundle of compressed data for a series of samples, not just one sample per channel. Because each frame encapsulates a whole series of samples, the frame rate is slower than the sample rate. The frame also contains a header. Despite the header, the frame size in bytes is less than the size in bytes of the equivalent number of PCM frames. (After all, the purpose of MP3 is to be more compact than PCM data.) For such an encoding, the sample rate and sample size refer to the PCM data that the encoded sound will eventually be converted into before being delivered to a digital-to-analog converter (DAC).

b. **File Formats -** A file format specifies the structure of a sound file, including not only the format of the raw audio data in the file, but also other information that can be stored in the file.

- In the Java Sound API, a file format is represented by anAudioFileFormatobject, which contains:
- The file type (WAVE, AIFF, etc.)
- The file's length in bytes
- The length, in frames, of the audio data contained in the file

- An AudioFormat object that specifies the data format of the audio data contained in the file

c. **Mixer -** Many application programming interfaces (APIs) for sound make use of the notion of an audiodevice. A device is often a software interface to a physical input/output device. For example, a sound-input device might represent the input capabilities of a sound card, including a microphone input, a line-level analog input, and perhaps a digital audio input. In the Java Sound API, devices are represented byMixerobjects. The purpose of a mixer is to handle one or more streams of audio input and one or more streams of audio output. In the typical case, it actually mixes together multiple incoming streams into one outgoing stream. AMixer object can represent the sound-mixing capabilities of a physical device such as a sound card, which might need to mix the sound coming in to the computer from various inputs, or the sound coming from application programs and going to outputs.

d. **Line -** A line is an element of the digital audio ”pipeline” that is, a path for moving audio into or out of the system. Usually the line is a path into or out of a mixer (although technically the mixer itself is also a kind of line).

Accessing Audio System Resources

The Java Sound API takes a flexible approach to system configuration. Different sorts of audio devices (mixers) can be installed on a computer. The API makes few assumptions about what devices have been installed and what their capabilities are. Instead, it provides ways for the system to report about the available audio components, and ways for the program to access them.

The AudioSystem Class

TheAudioSystemclass acts as a clearinghouse for audio components, including built-in services and separately installed services from third-party providers.AudioSystemserves as an application program’s entry point for accessing these installed sampled-audio resources. AudioSystemcan be queried to learn what sorts of resources have been installed, and then obtain access to them. For example, an application program might start out by asking theAudioSystemclass whether there is a mixer that has a certain configuration, such as one of the input or output configurations illustrated earlier in the discussion of lines. From the mixer, the program would then obtain data lines, and so on.

Here are some of the resources an application program can obtain from theAudioSystem:

- **Mixers -** A system typically has multiple mixers installed. There is usually at least one for audio input and one for audio output. There might also be mixers that don't have I/O ports but instead accept audio from an application program and deliver the mixed audio back to the program. TheAudioSystemclass provides a list of all of the installed mixers.

- **Lines -** Even though every line is associated with a mixer, an application program can get a line directly from theAudioSystem, without dealing explicitly with mixers.

- **Format conversions -** An application program can use format conversions to translate audio data from one format to another.

- **Files and streams -** TheAudioSystemclass provides methods for translating between audio files and audio streams. It can also report the file format of a sound file and can write files in different formats.

3.3.3 Audio Sampling

Although Audio sampling is not part of the actual feature extraction process, it does form the basis for further processing. It involves the reduction of a continuous audio signal to a discrete audio signal. To sample the audio signal we have used the java sound api for .wav file and mp3spi library for .mp3 files.

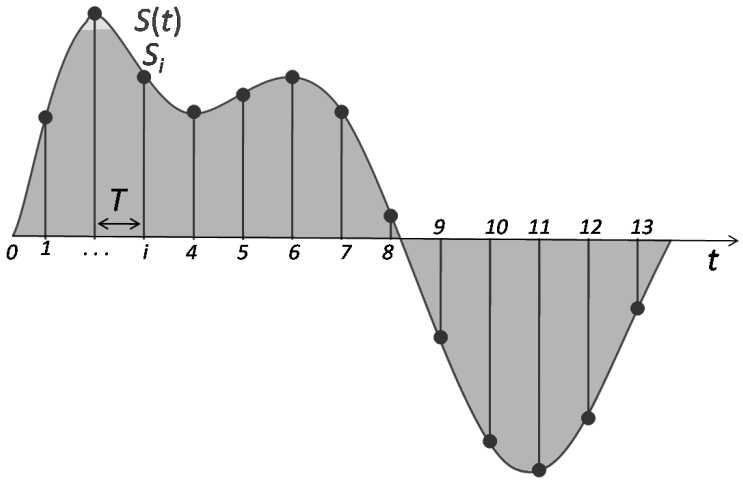

Figure 3: Sampling of an audio signal

3.3.4 Timbral Features

The main point to understand about speech is that the sounds generated by a human are filtered by the shape of the vocal tract including tongue, teeth etc. This shape determines what sound comes out. If we can determine the shape accurately, this should give us an accurate representation of the phoneme being produced. The shape of the vocal tract manifests itself in the envelope of the short time power spectrum, and the job of MFCCs is to accurately represent this envelope.

Many existing researchers have shown that mel-frequency cepstral coefficients (MFCCs), so called spectral shapes and spectral contrast are the best features for analyzing the music. The mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency. Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFCC. They are derived from a type of cepstral representation of the audio clip (a nonlinear spectrum-of-a-spectrum).

MFCC Algorithm

We start with a speech signal, which we sample at 44100Hz.

1) Frame the signal into 20-40 ms frames. 25ms is standard. This means the frame length for a 16kHz signal is 0.025*16000 = 400 samples. Frame step is usually something like 10ms (160 samples), which allows some overlap to the frames. The first 400 sample frame starts at sample zero, the next 400 sample frame starts at sample 160 etc. until the end of the speech file is reached. If the speech file does not divide into an even number of frames, pad it with zeros so that it does.

   The next steps are applied to every single frame, one set of 12 MFCC coefficients is extracted for each frame. A short aside on notation: we call our time domain signal S(n). Once it is framed we have where $S_i(n)$ ranges over 1-400 (if our frames are 400 samples) and i ranges over the number of frames. When we calculate the complex DFT, we get $S_i(k)$ where the i denotes the frame number corresponding to the time-domain frame and $P_i(s)$ then the power spectrum of frame i.

2) To take the Discrete Fourier Transform of the frame, perform the following:

$$S_i(k) = \sum_{n=1}^{N} (S_i(n)h(n)\mathrm{e}^{2\pi kn/n}) \qquad for 1 < k < K \tag{1}$$

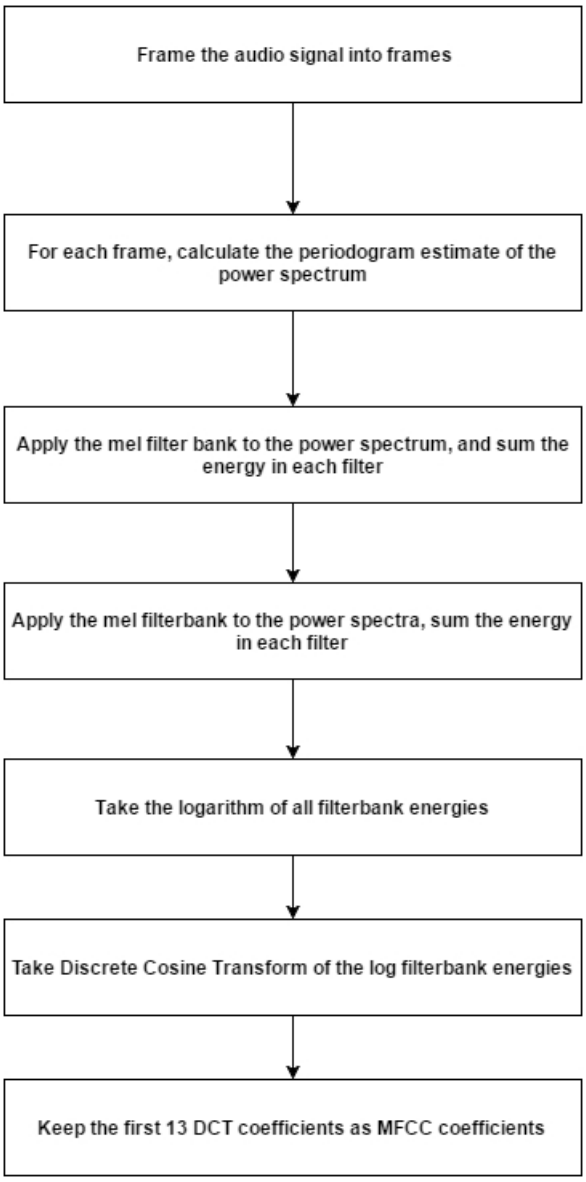

Figure 4: MFCC component

where h(n) is an N sample long analysis window (e.g. hamming window), and K is the length of the DFT. The periodogram-based power spectral estimate for the speech is $S_i(n)$ is given by:

$$P_i(k) = \frac{1}{N}|S_i(k)^2| \tag{2}$$

This is called the Periodogram estimate of the power spectrum. We take the absolute value of the complex fourier transform, and square the result. We would generally perform a 512 point FFT and keep only the first 257 coefficents.

3) Compute the Mel-spaced filterbank. This is a set of 20-40 (26 is standard) triangular

filters that we apply to the periodogram power spectral estimate from step two. Our filterbank comes in the form of 26 vectors of length 257 (assuming the FFT settings from step two. Each vector is mostly zeros, but is non-zero for a certain section of the spectrum. To calculate filterbank energies we multiply each filterbank with the power spectrum, then add up the coefficents. Once this is performed we are left with 26 numbers that give us an indication of how much energy was in each filterbank. For a detailed explanation of how to calculate the filterbanks see below. Here is a plot to hopefully clear things up:

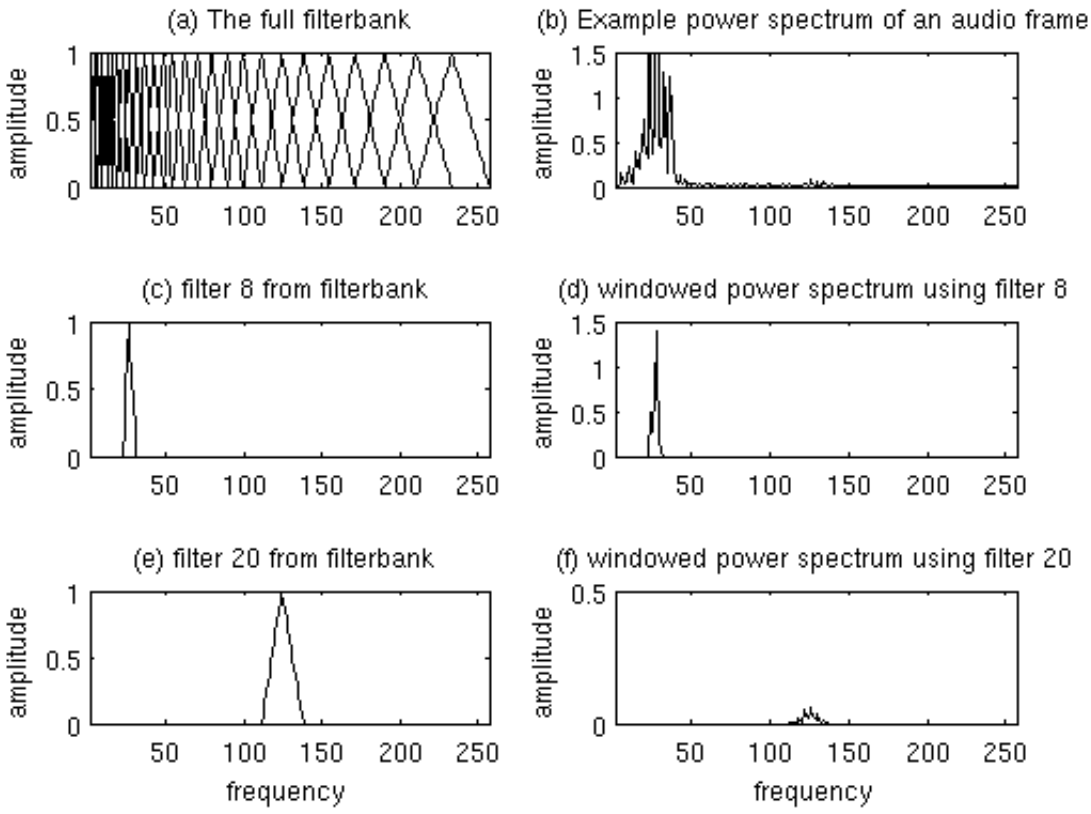

Figure 5: MFCC Calculation spectrum

4) Take the log of each of the 26 energies from step three. This leaves us with 26 log filterbank energies.

5) Take the Discrete Cosine Transform (DCT) of the 26 log filterbank energies to give 26 cepstral coefficents. For ASR, only the lower 12-13 of the 26 coefficients are kept.

The resulting features (12 numbers for each frame) are called Mel Frequency Cepstral Coefficients.

3.3.5 Rhythmic Features

Rhythmic features characterize the movement of music signals over time and contain information such as the regularity of the rhythm, beat tempo, and time signature. The feature set for representing rhythm structure is usually extracted from the beat histogram. Tzanetakis [13] used a beat histogram built from the autocorrelation function of a signal to extract rhythmic features.

Simulating a physical phenomena which obeys to known mathematical equations is, with a number of approximations, always feasible. But what about more abstract concepts, such as feelings, which do not follow any laws? The simplest things we can feel are often the hardest things to capture in a program. Beat detection follows this rule : feeling the beat of a song comes naturally to humans or animals. Indeed it is only a feeling one gets when listening to a melody, a feeling which will make you dance in rhythm or hit a table with your hands on the melody beats.

The human listening system determines the rhythm of music by detecting a pseudo periodical succession of beats. The signal which is intercepted by the ear contains a certain energy, this energy is converted into an electrical signal which the brain interprets. Obviously, The more energy the sound transports, the louder the sound will seem. But a sound will be heard as a beat only if his energy is largely superior to the sound's energy history, that is to say if the brain detects a brutal variation in sound energy. Therefore if the ear intercepts a monotonous sound with sometimes big energy peaks it will detect beats, however, if you play a continuous loud sound you will not perceive any beats. Thus, the beats are big variations of sound energy.

To detect the beats we have used the Frequency selected sound energy algorithm.
Every 1024 samples:

1. Compute the FFT on the 1024 new samples taken in (a_n) and (b_n). The FFT inputs a complex numeric signal. We will say (a_n) is the real part of the signal and (b_n) the imagenary part. Thus the FFT will be made on the 1024 complex values of:

$$a_n + i(b_n) \tag{3}$$

2. From the FFT we obtain 1024 complex numbers. We compute the square of their module and store it into a new 1024 buffer. This buffer (B) contains the 1024 frequency

amplitudes of our signal.

3. Divide the buffer into 32 subbands, compute the energy on each of these subbands and store it at (E_s). Thus (E_s) will be 32 sized and $E_s[i]$ will contain the energy of subband 'i':

$$E_s[i] = \frac{32}{1024} * \sum_{k=i*32}^{(i+1)*32} B[k] \tag{4}$$

4. Now, to each subband 'i' corresponds an energy history buffer called (E_i). This buffer contains the last 43 energy computations for the 'i' subband. We compute the average energy $< E_i >$ for the 'i' subband simply by using:

$$E_i[0] = \frac{1}{43} \times \sum_{k=0}^{42} E_i[k] \tag{5}$$

5. Shift the sound energy history buffers (E_i) of 1 index to the right. We make room for the new energy value of the subband 'i' and flush the oldest.

6. Pile in the new energy value of subband 'i' : $E_s[i]$ at $E_i[0]$.

$$E_i[0] = E_s[i] \tag{6}$$

7. For each subband 'i' if $E_s[i] > (C * E_i)$ we have a beat.

3.3.6 Intensity Features

Sound intensity also known as acoustic intensity is defined as the sound power per unit area. The usual context is the noise measurement of sound intensity in the air at a listeners location11 as a sound energy quantity. Intensity is approximated by the signals root mean square (RMS). The true RMS value of the input signal is calculated over a running average window of one cycle of the specified fundamental frequency. Procedure taken:

1. Input the WAV file and extract its metadata such as frame size, the number of channels etc.

2. Calculate the sample count for the file using number of channels, frame size and sample size in bits as parameters.

3. Initialize a byte buffer to store the samples read from the AudioInputStream for the wav file.

4. Depending on the audio byte format(little or big endian), convert the byte buffer into a float buffer.

5. Calculate the rms of the audio signal using basic RMS formula.

6. Finally convert the rms value to decibel scale.

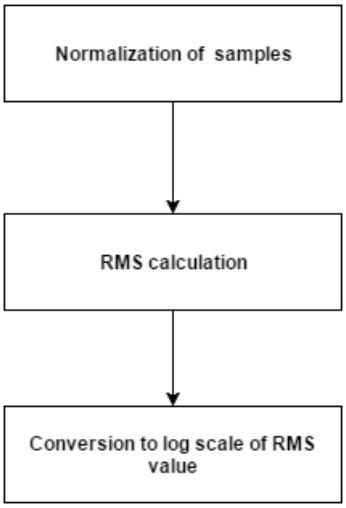

Figure 6: Flowchart for intensity calculation

3.3.7 Feature Outputs

For each song we obtain the following information:

1. The Song Name (String) obtained from the filename.

2. The Intensity (Double)

3. The MFCC (List of 40 Doubles)

4. The Rhythm (List of 32 Doubles)

An example set of data is given below (some values removed for clarity):

”songName”:”Pop-22.mp3”,

”intensity”:

58.510281774287414,

”mfcc”:

[20.7403817933,-.1448360237883,-.10059620606765,....,0.619933641865,2.75522298062],

”rhythm”:

[0.0,2.0,6.0,9.0,4.0,14.0,11.0,4.0,2.0,0,12.0,17.0,4.0,9.0,6.0,4.0,0.0]

Intensity is usually in the 50 - 80 range, MFCC values are around 1.0 except the first value, while rhythm normally lies in the 0.0 to 20.0 region.

3.4 K-means Clustering

3.4.1 Overview

As mentioned before, at this point in the project, we have completed implementing the K-means algorithm. We decided to implement it ourselves for two reasons:

a.) It is a pretty straightforward and simple algorithm.

b.) We wanted to have full control over all aspects related to the implementation: the initialization method, distance metric, etc.

The implementation however is basic in the sense that no modification has been done on the algorithm to better suit the problem domain.

Some finer points of the implementation are discussed below after quickly going over the algorithm itself.

3.4.2 Algorithm

Let X = $x_1, x_2, ..., x_n$ be the set of data points and V = $v_1, v_2, ..., v_c$ be the set of centers.

1.) Randomly select ’c’ cluster centers.

2.) Calculate the distance between each data point and cluster centers.

3.) Assign the data point to the cluster center whose distance from the cluster center is minimum of all the cluster centers.

4.) Recalculate the new cluster center:

$$v_i = \frac{1}{c_i} \sum_{j=1}^{c_i} x_i \tag{7}$$

5.) Recalculate the distance between each data point and new obtained cluster centers.

6.) If no data point was reassigned then stop, otherwise repeat from step 3).

3.4.3 Initialization method

Commonly used initialization methods are Forgy and Random Partition. The Forgy method randomly chooses k observations from the data set and uses these as the initial means. The Random Partition method first randomly assigns a cluster to each observation and then proceeds to the update step, thus computing the initial mean to be the centroid of the clusters randomly assigned points. The Forgy method tends to spread the initial means out, while Random Partition places all of them close to the center of the data set. According to [6], the Random Partition method is generally preferable for algorithms such as the k-harmonic means and fuzzy k-means. For expectation maximization and standard k-means algorithms, the Forgy method of initialization is preferable.

And with these facts in mind, we went for the random initialization method. This method has been used in [2] too although they have also added the constraint that the centroids be separated by at least a threshold KL-divergence distance. As the choice of initial centroids have a drastic effect on the cluster formed, we have also considered other methods of initialization such as the breakup method which uses the actual data points and the scrambled midpoints method which uses synthetic data points as suggested in [4]. However, these techniques are to be implemented in the next phase.

3.4.4 Distance Metric

Apart from the initialization method, K-means is also highly sensitive to the distance metric used. K-Means is implicitly based on pairwise Euclidean distances between points, because the sum of squared deviations from centroid (that it tries to minimize) is equal to the sum of pairwise squared Euclidean distances divided by the number of points [5].

As such, we decided to use Euclidean distance with weights for the three different features added to give us a way to control the metric. Thus, the distance between two songs S1 and S2 is given by:

$$Distance(d) = \sqrt[2]{w_I(I_1 - I_2)^2 + w_M \sum_i (M_{1i} - M_{2i})^2 + w_R \sum_i (R_{1j} - R_{2j})^2} \qquad (8)$$

Where: w_I , w_M and w_R are the weights for Intensity, MFCC and Rhythm respectively.

3.4.5 Reading the Data Points

The feature extractor stores the data as JSON which we read into the classifier using Gson a Java serialization library that can convert Java objects into JSON and back.

The choice of JSON was based on the fact that it's easily human readable and there are lots of libraries in a variety of languages to read it if needed.

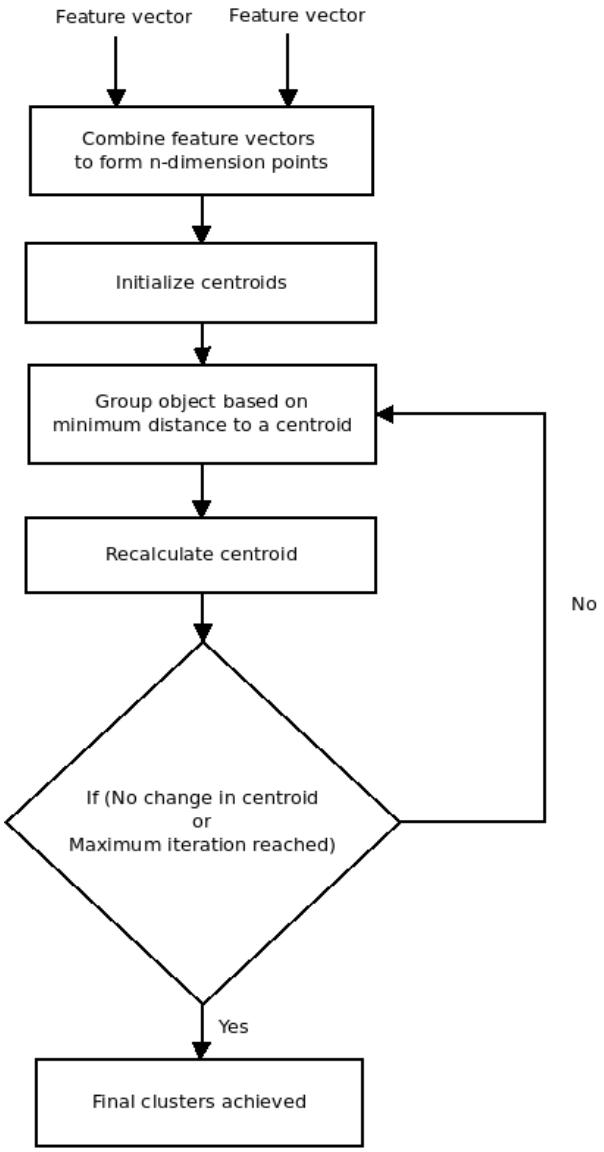

Figure 7: K-means clustering

4 IMPLEMENTATION

=

5 DEVELOPMENT METHODS

6 PROBLEM FACED AND SOLUTION

problem faced are

7 RESULTS AND VALIDATION

7.1 Results Overview

We obtain five clusters as the overall output of the system.

```
ID:2
Label:Hiphop
Centroid:Pop-22.mp3
Pop-22.mp3
Hiphop-37.mp3
Hiphop-12.mp3
Hiphop-2.mp3
Hiphop-3.mp3
Hiphop-18.mp3
Pop-1.mp3
Hiphop-13.mp3
Jazz-4.mp3
```

Figure 8: Result Overview

The cluster is identified by its ID. We also obtain the initial centroid used by the cluster which is useful in giving us an indication as to how the clusters were formed and how much impact the initial choice had. Furthermore, each of the clusters are labeled to be one of the five Genre and this is also part of the output.
The process of labeling the cluster is discussed in the next section.

7.2 Validation Process

We were uniquely placed in terms of how the validation needed to be carried out. As it was not supervised learning, there was no notion of training and test dataset distinction. However, it also wasn't one of those (normal) cases where we do clustering in a completely unsupervised manner hoping to discovery new knowledge. Instead, we had true labels of the song genre which could/should be used to validate the results obtained. And so, we needed to perform external evaluation of the clusters as opposed to internal evaluation (Dunn Index, Silhouette coefficient).

Eventually, we settled on using an accuracy measure obtained from the confusion matrix for the five genre.

The diagonal values of the confusion matrix are the number of correctly assigned songs for each genre. And so, the accuracy is the ratio of the sum of all diagonal elements to the

Table 1: Validation Process

	Rock	Classical	Pop	Jazz	Hiphop
Rock	#	-	-	-	-
Classical	-	#	-	-	-
Pop	-	-	#	-	-
Jazz	-	-	-	#	-
Hiphop	-	-	-	-	#

total number of songs.

$$Accuracy(A) = \frac{\sum^{i=j} M_{ij}}{\sum_{i,j \in G} M_{ij}} \tag{9}$$

The next job was to determine a way to label each of the clusters as without labeling them we cannot obtain the confusion matrix! To label them we could have simply labeled the cluster according to which genre holds the majority in the cluster but we found it to be problematic as there might be no clear majority and one genre might hold a majority in multiple clusters.

Instead, we decided to use a better but computationally expensive approach: try out all possible combinations of the labels and choose the combination with the best accuracy. Here, we create 120 (5!) possible confusion matrices, calculate the accuracy for each and keep the one with the best result.

7.3 Results and Discussion

We used a data-set of 230 songs with the following distribution:
Pop (40), Rock (50), Classical (50), Jazz (50), Hiphop (40).

The result obtained varies slightly for each run due to the randomness of the initialization process but perhaps due to the small size of the data-set they usually converge to the same clusters/accuracy.

With all weights (w_I , w_M and w_R) set at 1.0; in other words without letting them affect the clustering, we obtain an accuracy of around 46 per cent.

One of the result showing the confusion matrix is given below. This result converged

in 16 iterations. The number of iterations vary but mostly convergence is reached in 10-20 iterations.

[Assigned\Actual]	Classical	Pop	Hiphop
Classical	17	1	0
Pop	1	19	8
Hiphop	0	10	30
Rock	10	7	2
Jazz	22	3	0

Figure 9: Cross-Validation

Each column represents the actual genre of the data-set while each row represents one particular cluster.

As seen from the figure, Hiphop has the best classification with a 75 per cent accuracy. The others don't fare so well with Classical (34 per cent) being the most badly assigned genre as 22 classical songs has been classified as Jazz.

```
Distance:Classical-12.mp3 to Jazz-11.mp3 => 59.80492225997541
Distance:Pop-19.mp3 to Jazz-4.mp3 => 320.9455169324791
Distance:Pop-18.mp3 to Pop-17.mp3 => 102.03630292981451
Distance:Jazz-20.mp3 to Classical-24.mp3 => 208.47384068656174
Distance:Hiphop-13.mp3 to Pop-13.mp3 => 261.22291368028874
```

Figure 10: Inter-cluster Distance

Initial observations seem to point towards our distance metric being at fault for assuming Classical to be close to Jazz. Or, it could also be that for the features we have chosen, these two genre are too similar. This problem is also encountered in [8]

The system also cannot distinguish Rock, Pop and Jazz from each other with Jazz being classified as Rock being the second biggest problem after Classical being classified as Jazz.

Overall, as it currently stands, the system is only able to cluster Hiphop songs with satisfactory accuracy.

8 FUTURE SYSTEM ENHANCEMENT

8.1 Feature Extraction

Most of the job in this part has been completed. Some of the remaining tasks are:

1. Extracting the Pitch Feature. It will be the foremost job for this section of the project in the coming days.
2. Optimizing the extraction process and making it faster and less memory extensive.
3. Initial tinkering with the weights have shown that Intensity is highly influential in the clustering process and that the other two features aren't contributing much to the results. This will be investigated.
4. The role and affect of the various constants and assumptions used during feature extraction will be analyzed.
5. Code Refactoring and Documentation.

8.2 K-means Clustering

The basic implementation has been completed. Other work in this area will be to extend it:

1. Trying out different initialization methods discussed in [4] and studying their effects.
2. Trying out different distance metrics and studying their effects.
3. Investigating why the accuracy is poor for genre other than Hiphop.
4. Using measures for cluster quality such as Entropy, Purity, etc.
5. Code Refactoring and Documentation

8.3 Artificial Neural Network and Mood Based Classification

Nothing has been done in this part of the project. So, it will take up the majority of the efforts in the coming days:

1. Research
2. Code Implementation

3. Training set collection

4. Validation methodologies determination

8.4 Application Development

The project aims to develop an application with following features:

1. Ability to list out all mp3 fles on the accessible File-system.

2. Basic media player abilities: play, pause, stop, previous, next.

3. Create smart playlists:
   - Playlists with similar songs
   - Playlists that focus on variety
   - Playlists that interpolate/add songs between any two given songs

Developing this application will also take up a lot of the time and effort.

8.5 Gantt Chart

The gantt chart for the remaining task is as:

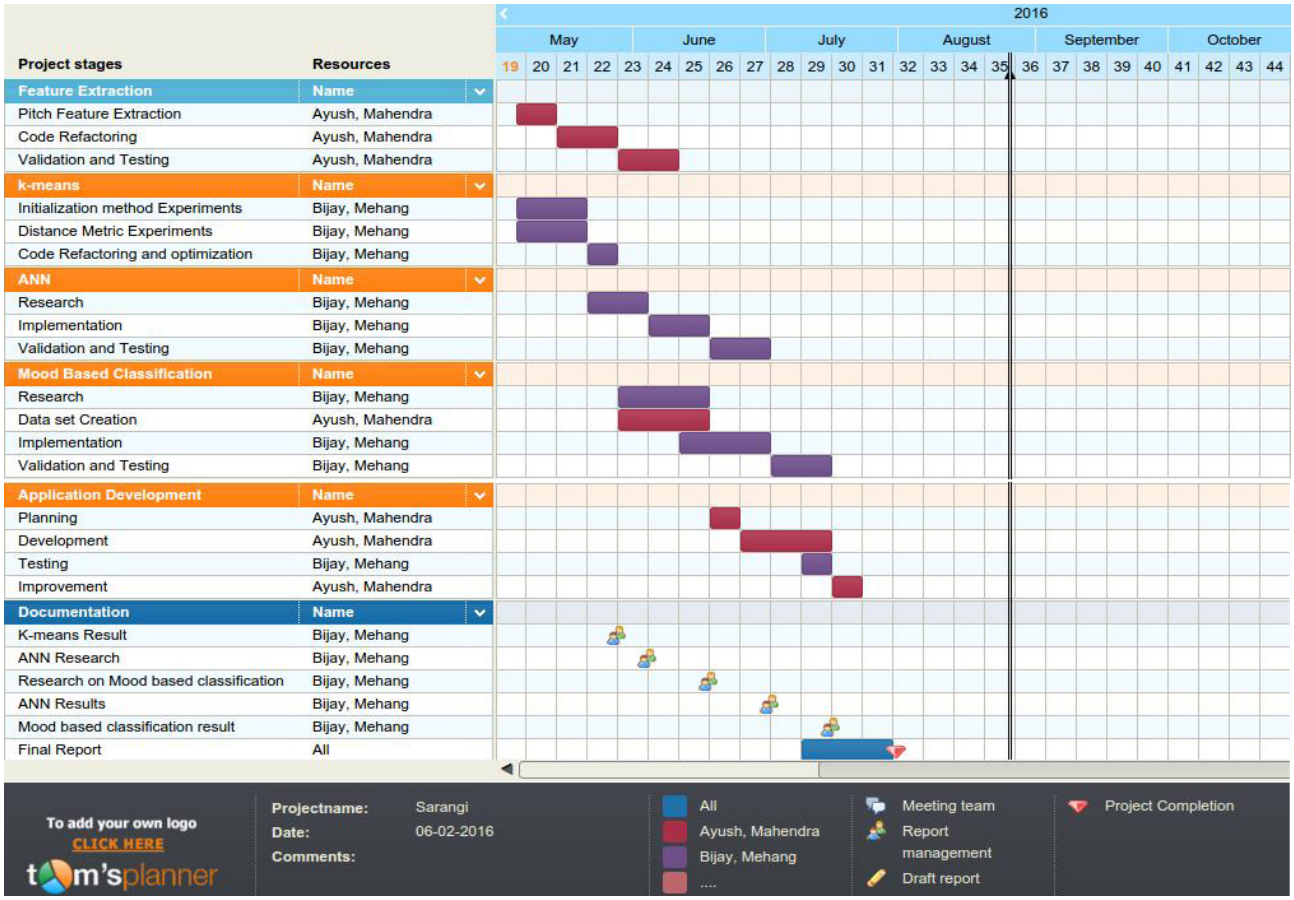

Figure 11: Gantt chart for the remaining task

9 CONCLUSION

Among the three deliverables of the project - Genre based classification system, Mood based classification system and Playlist Generating Application - substantial work has been done only on the first job. However, as the feature extraction is close to completion, we expect the latter two jobs to take up less time. So, overall we can say that we are only slightly behind schedule if not on time. Regardless, we remain on course to meet the project objective.

The current accuracy of the system is very poor indeed but through analysis and further development, it is sure to be improved by the end of the project.

We have only briefly discussed the work to be done in the next session. A more detailed methodology and schedule will be created as soon as possible.

REFERENCES

[1] Smith, Steven W. *The scientists and engineers guide to digital signal processing.* California Technical Publishing, 2013. Print.

[2] Muller, Mathias, et al. *Signal processing for music analysis.*, Selected Topics in Signal Processing, IEEE Journal of 5.6 (2011): 1088-1110.

[3] Dooling, Robert J and Stewart H Hulse. *The Comparative Psychology Of Audition.* Hillsdale, N.J.: L. Erlbaum Associates, 1989. Print.

[4] Prasad, Bhanu, and SR Mahadeva Prasanna, eds. *Speech, audio, image and biomedical signal processing using neural networks.* Vol. 83. Springer, 2007.

[5] Neumayer, Robert. *Musical genre classification.* (2004).

[6] Kour, Gursimran, and Neha Mehan. *Music Genre Classification using MFCC, SVM and BPNN.* International Journal of Computer Applications 112.6 (2015).

[7] Koerich, Alessandro L. *Improving the Reliability of Music Genre Classification using Rejection and Verification.* ISMIR. 2013.

[8] Haggblade, Michael, Yang Hong and Kenny Kao. Music genere classification. *Department of Computer Science*, Stanford University, 2011.

[9] Nasridinov, Aziz, and Young-Ho Park. *A Study on Music Genre Recognition and Classification Techniques.* International Journal of Multimedia and Ubiquitous Engineering 9.4 (2014): 31-42.

[10] Anglade, Amlie, et al. *Improving music genre classification using automatically induced harmony rules.* Journal of New Music Research 39.4 (2010): 349-361.

[11] Li, Tao, and George Tzanetakis. *Factors in automatic musical genre classification of audio signals.* Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on.. IEEE, 2003.

[12] Tzanetakis, George, and Perry Cook, Musical genre classification *Journal of Personality*, 60(2):225251, 1992.

[13] Tzanetakis, George, and Perry Cook, Musical genre classification based on audio signals *IEEE Transactions on Speech and Audio Processing*, 10.5 (2002)

[14] Apon, Amy, et al. *Inital Starting Point Analysis for K-Means Clustering: A Case Study.* (2006).

[15] Jain, Anil K. *Data clustering: 50 years beyond K-means.* Pattern recognition letters 31.8 (2010): 651-666.

[16] Hamerly, Greg, and Charles Elkan. *Alternatives to the k-means algorithm that find better clusterings.* In Proceedings of the eleventh international conference on Information and knowledge management, pp. 600-607. ACM, 2002.

APPENDIX A

appendix to write