Department of Computer Science & Engineering

A Project Report on

LyriQuest : Music Analysis and Recommendation System using Tones and Topic Extraction

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Engineering in Computer Science & Engineering

By

M V S Viswanadh	1MS15CS074
Shravan A R	1MS15CS115
Vaibhav Srivastava	1MS15CS135
Vikas Shankarathota	1MS15CS140

Under the guidance of

Srinidhi H

Assistant Professor, Department of Computer Science and Engineering

M S RAMAIAH INSTITUTE OF TECHNOLOGY
(Autonomous Institute, Affiliated to VTU)
BANGALORE-560054
www.msrit.edu
May 2019

RAMAIAH
Institute of Technology

Department of Computer Science & Engineering

CERTIFICATE

Certified that the project work entitled "LyriQuest Music Analysis and Recommendation System using Tones and Topic Extraction" carried out by M V S Viswanadh-1MS15CS074, Shravan AR-1MS15CS115, Vaibhav Srivastava - 1MS15CS135, Vikas Shankarathota - 1MS15CS140, bonafide students of M.S.Ramaiah Institute of Technology Bengaluru in partial fulfillment for the award of Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belgavi during the year 2018-19. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library.

The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Project Guide **Head of the Department**

SRINIDHI H. **DR.ANITA KANAVALLI**

External Examiners

Name of the Examiners: **Signature with Date**

1.

2.

RAMAIAH
Institute of Technology

Department of Computer Science & Engineering

DECLARATION

We, hereby, declare that the entire work embodied in this project report has been carried out by us at M.S.Ramaiah Institute of Technology, Bengaluru, under the supervision of Srinidhi H**,** Assistant Professor**,** Dept of CSE. This report has not been submitted in part or full for the award of any diploma or degree of this or to any other university.

M V S Viswanadh	Shravan A R
1MS15CS074	1MS15CS115
Vaibhav Srivastava	Vikas Shankarathota
1MS15CS135	1MS15CS140

ACKNOWLEDGEMENT

We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project. We would like to express our profound gratitude to the Management and **Dr. N.V.R Naidu** Principal, M.S.R.I.T, Bengaluru for providing us with the opportunity to explore our potential.

We extend our heartful gratitude to our beloved **Dr. Anita Kanavalli**, HOD, Computer Science and Engineering, for constant support and guidance.

We whole heartedly thank our project guide **Srinidhi H,** for providing us with the confidence and strength to overcome every obstacle at each step of the project and inspiring us to the best of our potential. We also thank him for his constant guidance, direction and insight during the project.

This work would not have been possible without the guidance and help of several individuals who in one way or another contributed their valuable assistance in preparation and completion of this study.

Finally we would like to express sincere gratitude to all the teaching and non-teaching faculty of CSE Department, our beloved parents, seniors and my dear friends for their constant support during the course of work.

M V S Viswanadh Shravan AR

Vaibhav Srivastava Vikas Shankarathota

Abstract

Our project aims at enhancing the experience for sound cataloguing by employing methods to further classify music as well as giving a description about the lyrics. People listen to music in order to feel certain emotions and express joy and sadness. Sometimes people want to listen to sad songs but at other times they want to listen to happy music. This application will help users select music based on their mood.

The software uses various modules to fulfill the operating requirements. It employs the help of song fingerprinting, machine learning, and data science in order to recommend songs, draw graphs, and give analysis on the type of music a person listens to.

The user will be able to find out more information about known and unknown songs rather than just looking at the title. Users will have access to a quick description about the topics and emotions in a song so that they can make a quick judgment on whether to include the track on their playlist. This approach can also help already existing playlist generation machines improve accuracy and create better recommendations.

TABLE OF CONTENTS

List of Tables

Table of Contents

Chapter 1

INTRODUCTION

1.1 General Introduction

Music can affect your mood in many ways. This is because of the rhythm and tone that we hear when we listen to music. When we listen to a rhythm, our heart actually begins to sync with it. A slow heartbeat with a strong diastolic pressure tells our brain that something sad or depressing is occurring. Very fast beating has to do with excitement, while a dreamy rhythm with occasional upbeats can be a sign of love or joy. Tones are just as important as rhythm. A "major key" music piece will usually signal cheerful communication to our brain, while "minor key" pieces are sadder. This all has a very strong effect on our brain, which makes our mind actually feel what's being communicated to us.

When people listen to joyful, happy music, our brains usually produce chemicals such as serotonin and dopamine, which make us feel happy. The same happens when one listens to relaxing, soothing music or to hard, loud, angry music that can also cause a number of different emotional feelings. Music may cause an individual feel joyful, sad, angry, hyped up, relaxed etc. and sometimes you can feel more than one emotion during a song.

As research shows, music not only affects what kind of mood we may be in, but we also seem to have a habit of choosing music based on the moods we are already feeling. So if we are feeling very happy, sorrowful, angry etc. we will generally want to listen to that type of music. However, if you feel like you are falling into picking certain music that makes a negative mood worse, you should be aware of what type of music can help reverse the negative mood you feel instead of encouraging it. Thus, good music recommendation systems are of utmost importance in the current world.

1.2 Problem Statement

Music playlist creating models are already implemented and are used on various applications like Spotify and Apple Music. These models tag the music based on genre and create playlists of songs based on the taste of the user's listening choice. This model is similar to Google Adsense and Netflix's recommendation engine. The software aims are enhancing the tags available for a particular piece of English music by extracting the themes and sentiment associated with it. By considering tones and sentiment associated with the songs, a better recommendation system can be implemented.

1.3 Objectives of the project

LyriQuest is a music recommendation application. A few objectives accomplished in this application would be:

- **Better Recommendation of Songs :** By taking into consideration of the tones and sentiment associated with songs, the application can provide better recommendation to the users
- **Title Extraction** : The engine at finding out the artist,title and genre of the song based on the audio input to the engine, either an MP3 file or a segment of the song being played.
- **Sentiment Analysis:** We carry out sentiment analysis on the lyrics of the song in order to find out the general mood of the singer, such as happy, sad, and analytical.
- **Topic Extraction:** Topic extraction will be done in order to find out the main themes the song is based on like motivational, devotional, break-up song, and romantic.
- **Trend Analysis:** Analyse the sentimental trends in the music released over the past decade.

1.4 Project deliverables

1) **Easy UI Design**: Kivy has been used to build the user interface.
2) **Word Cloud:** This module the most frequent words used in the song which could be used for further analysis.
3) **Authentication:** Provide user login so that users can create their own playlist. Analysis of this playlist gives an idea of the emotions of the user.
4) **Crash-free Services**: Due to the usage of Firebase in the backend for authentication and storage it is 100 % crash free and this ensures smooth flow of the application and ensures crash free situations at any time.
5) **Security**: The database includes personal playlists of the users and this won't be accessible by any third party vendor which ensures security in the project LyriQuest.
6) **Personal Analysis**: Analysis of the personal playlist helps understand the emotional state of the user.
7) **Topic Extraction**: With the extraction of the topics of the song, a user can know what the song speaks of and then decide to add to his personal playlist.
8) **Trends**: Bar graph and line graphs with respect to the trends of the sentiments in music released the past 15 years.

1.5 Current Scope

Rapid development of mobile devices and internet has made possible for us to access different music resources freely. The number of songs available exceeds the listening capacity of single individual. People sometimes feel difficult to choose from millions of songs. Moreover, music service providers need an efficient way to manage songs and help their customers to discover music by giving quality recommendation. Thus, there is a strong need of a good recommendation system. Currently, there are many music streaming services, like Pandora, Spotify, etc. which are working on building high-precision commercial music recommendation systems. These companies generate revenue by helping their customers discover relevant music and charging them for the quality of their recommendation service. Thus, there is a strong thriving market for good music recommendation systems. Presently, the music recommendation systems tag music based on the genre. The application uses the tone and sentiment of the song to create a better recommendation platform.

1.6 Future scope

Ability to play songs from LyriQuest can be added. Currently the application has desktop compatibility which could be extended into mobile devices by developing the Android version of the LyriQuest application. By using the location of the user, the application could suggest songs to the user endemic to that place. The application's recommendation of songs can be extended by increasing the database of the songs used in the application. The API calls to Watson API for sentiment analysis can be replaced by more efficient machine learning model.

Chapter 2

PROJECT ORGANIZATION

2.1 Software Process Models

The project uses the iterative process model, which was found to be most suitable for the task.

Iterative development process model is based on the idea of developing an initial implementation, exposing this to user feedback, and evolving it through several versions until an acceptable system has been developed. The activities of a process are not separated but interleaved with feedback involved across those activities. Each system increment reflects a piece of the functionality that is needed by the customer. Generally, the early increments of the system should include the most important or most urgently required functionality. This means that the customer can evaluate the system at early stage in the development to see if it delivers what's required. If not, then only the current increment has to be changed and, possibly, new functionality defined for later increments. A project control list is created that contains, in order all the tasks that must be performed to obtain the final implementation. This project control list gives an idea of how far along the project is at any given step from the final system.

Each step consists of removing the next task from the list, designing the implementation for the selected task, coding and testing the implementation, performing an analysis of the partial system obtained after this step, and updating the list as a result of the analysis. These three phases are called the design phase, implementation phase, and analysis phase. The process is iterated until the project control list is empty, at which time the final implementation of the system will be available.

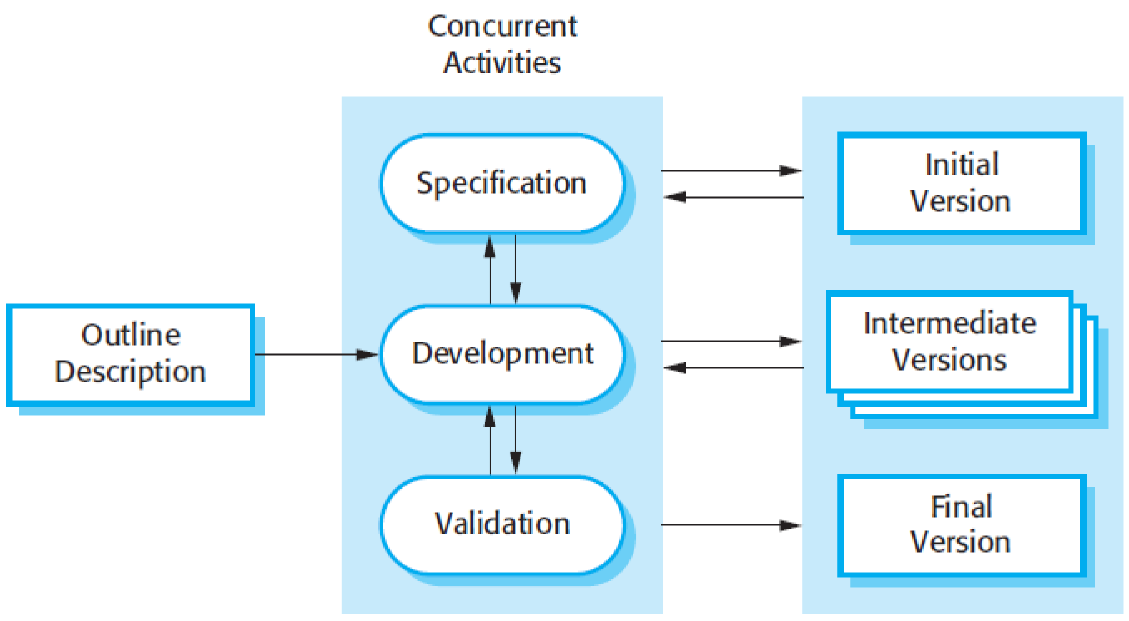


Fig 1. Iterative Development Process Model

2.2 Roles and Responsibilities

Roles and responsibilities of programmers:

- Topic extraction, Word Cloud, and Polarity Assessment: Vikas S
- Web crawling and ACR cloud usage: Vaibhav S
- Frontend Kivy Implementation: Shravan A R
- Firebase authentication and database: Shravan A R and M V S Viswanadh
- User Playlist analysis and Trends analysis: M V S Viswanadh

Chapter 3

LITERATURE SURVEY

3.1 Introduction

The Application aims at creating a solution for tagging music based on genre, theme of the song, and emotions in the lyrics. There are a variety of different approaches that this task may be carried out and we aim to find the most efficient and effective implementation. Music playlist creating models are already implemented and are used on various applications like Spotify and Apple Music. These models tag the music based on genre and create playlists of songs based on the taste of the user's listening choice. This model is similar to Google Ad-sense and Netflix's recommendation engine. The software aims are enhancing the tags available for a particular piece of English music by extracting the themes and sentiment associated with it. The main stakeholders in this engine would be the end user, who can view the theme and mood of a song quickly, as well as other APIs which can use the extra tags we generate in order to create better playlist's suited to the user's listening habits. The software is divided into parts and each part implements a feature. The scope of this project is to implement the features necessary to identify the title of the song, identify the genre of music, grab sentiment from the lyrics, and find the main themes being depicted.

An acoustic fingerprint is a condensed digital summary, a fingerprint, deterministically generated from an audio signal, that can be used to identify an audio sample or quickly locate similar items in an audio database. In the implementation the application must be able to retrieve the title of the input track quickly with a high level of accuracy in order to obtain the lyrics for the song to carry out further analysis. Audio fingerprinting is a process by which music is identified with a small sample of the track to be identified. Audio Fingerprinting technologies have attracted attention since they allow the identification of audio independently of its format and without the need of meta-data or watermark embedding. Other uses of fingerprinting include: integrity verification, watermark support and content-based audio retrieval. The different approaches to fingerprinting have been described with different rationales and terminology: Pattern matching, Multimedia (Music) Information Retrieval or Cryptography (Robust Hashing) [1]. The project shall go about the implementation of song fingerprinting by using the Echoprint service. Echoprint [2] is a music identification system build by The Echo Nest [3], which utilizes acoustic fingerprinting technology for the identification functionality. Think of actual human fingerprints being used for identification. Acoustic fingerprinting uses the same principle, but by means of audio. Echoprint consists of a set of components which can be used to either experiment with and/or set-up an audio identification system/service. An acoustic fingerprint is achieved by creating a condensed digital summary of the audio signal. The system "listens" to the audio signal being played. An algorithm places marks on, for example, different spikes in frequency of the signal and is able to identify the signal by matching these marks to a database on an external server. The signal can include all different forms of audio, songs, melodies, tunes and for example sound effects from advertisements. Echoprint's ability to identify a music track substantially fast and with high accuracy makes it one of the most valuable music identification systems available. Additionally, it can even recognize noisy versions of the original track and even recordings performed by mobile devices with noise "bleed" by environmental factors. [4]

The next phase involves the identification of the genre of music. The software aims to do this by passing the song through a machine learning model which will be able to identify the genre of the song to a very high accuracy. Support vector machines are used to obtain the optimal class boundaries between different genres of music by learning from training data. Experimental results of multi-layer support vector machines illustrate good performance in musical genre classification and are more advantageous than traditional Euclidean distance based method and other statistic learning methods [5]. The implementation discusses relevant features to identify the genre like Beat Spectrum, LPC (linear predictive coding, zero crossing rates, spectrum power and Mel frequency Cepstral coefficients.)

The project intends to approach the sentiment analysis in two methods; one would be using the Watson API for Tone Analysis [6]. This returns 6 tones in speech as well as the degree to which the tone is positive or negative. First, a method for target representation is proposed that better captures the semantic meaning of the opinion target. Second, introduces an attention model that incorporates syntactic information into the attention mechanism [7].

The last and final phase of the project includes the topic extraction. As it is difficult to understand the type of music the model may come across, it is imperative that we have a unsupervised model, which will be able to pick out the main topics in a song regardless of the content. This is an unsupervised topic model that uses soft clustering over distributed representations of words. The distributed word representations are obtained by using a log-linear model and we model the low-dimensional semantic vector space represented by the dense word vectors using Gaussian mixture models (GMMs)[8]. Even though the power of LDA algorithm has been extensively used for leveraging topics, very few studies have been reported for mapping these statistically outputted topics to semantically rich concepts. The proposed framework is an attempt to address this issue by making use of LDA algorithm to generate topics and the project leverages concepts from such topics by using a new statistical weighting scheme and some lightweight linguistic processes [9].

By integrating the four above features the project aims at creating an engine that will help music listeners worldwide quickly make decisions on whether to listen to a song or not based on its content and genre. The engine can also benefit already existing models which try to curate playlists.

Chapter 4

PROJECT MANAGEMENT PLAN

4.1 Schedule of the project

The schedule of the project is as follows:

1. Formulating a problem statement would involve choosing the problem statement from the domain of interest and performing a feasibility studies check.

2. Preparation of a synopsis enclosing details of the problem statement, methodology, hardware and software requirements and the contribution to the society.

3. To track project schedule with actual progress, a Gantt chart is developing outlining the project schedule and activities to be performed.

4. From the feasibility studies, a requirement analysis is performed. A prototype of the system design is prepared and after necessary recommended changes, a final system design is created.

5. The implementation comprises of:

    a) Data Collection: Collecting surveillance video data.

    b) Data Preparation: Consists of the following phases:

        i. Data Cleaning: Removing outliers and noise in the data.

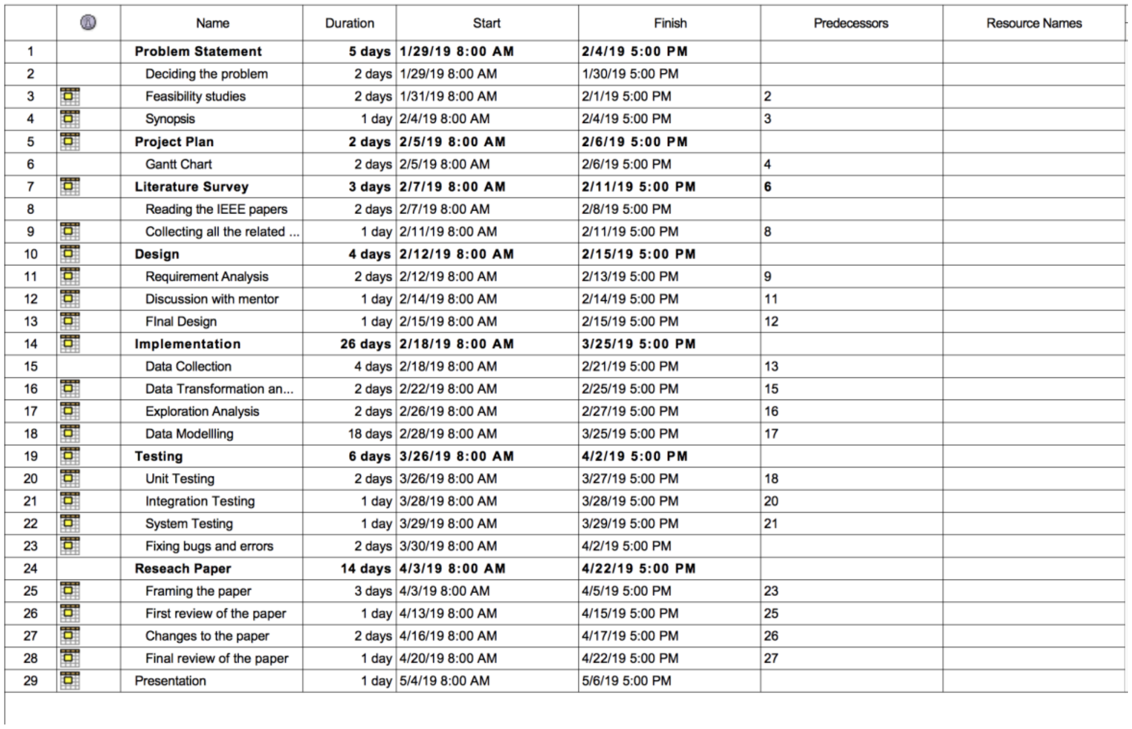
        ii. Data Integration: Enriching the data by combining multiple data sources

        iii. Data Transformation: Transforming data for suitable use in the model.

    c) Data Exploration: Gaining insights from the data to get a deep understanding.

    d) Data modeling: Choosing variables necessary for analysis and model building.

6. Testing would comprise of model diagnosis and fix the necessary bugs and errors in the training and testing.



	ⓘ	Name	Duration	Start	Finish	Predecessors	Resource Names
1		**Problem Statement**	**5 days**	**1/29/19 8:00 AM**	**2/4/19 5:00 PM**		
2		Deciding the problem	2 days	1/29/19 8:00 AM	1/30/19 5:00 PM		
3	📅	Feasibility studies	2 days	1/31/19 8:00 AM	2/1/19 5:00 PM	2	
4	📅	Synopsis	1 day	2/4/19 8:00 AM	2/4/19 5:00 PM	3	
5	📅	**Project Plan**	**2 days**	**2/5/19 8:00 AM**	**2/6/19 5:00 PM**		
6		Gantt Chart	2 days	2/5/19 8:00 AM	2/6/19 5:00 PM	4	
7	📅	**Literature Survey**	**3 days**	**2/7/19 8:00 AM**	**2/11/19 5:00 PM**	**6**	
8		Reading the IEEE papers	2 days	2/7/19 8:00 AM	2/8/19 5:00 PM		
9	📅	Collecting all the related ...	1 day	2/11/19 8:00 AM	2/11/19 5:00 PM	8	
10	📅	**Design**	**4 days**	**2/12/19 8:00 AM**	**2/15/19 5:00 PM**		
11	📅	Requirement Analysis	2 days	2/12/19 8:00 AM	2/13/19 5:00 PM	9	
12	📅	Discussion with mentor	1 day	2/14/19 8:00 AM	2/14/19 5:00 PM	11	
13	📅	FInal Design	1 day	2/15/19 8:00 AM	2/15/19 5:00 PM	12	
14	📅	**Implementation**	**26 days**	**2/18/19 8:00 AM**	**3/25/19 5:00 PM**		
15		Data Collection	4 days	2/18/19 8:00 AM	2/21/19 5:00 PM	13	
16	📅	Data Transformation an...	2 days	2/22/19 8:00 AM	2/25/19 5:00 PM	15	
17	📅	Exploration Analysis	2 days	2/26/19 8:00 AM	2/27/19 5:00 PM	16	
18	📅	Data Modellling	18 days	2/28/19 8:00 AM	3/25/19 5:00 PM	17	
19	📅	**Testing**	**6 days**	**3/26/19 8:00 AM**	**4/2/19 5:00 PM**		
20	📅	Unit Testing	2 days	3/26/19 8:00 AM	3/27/19 5:00 PM	18	
21	📅	Integration Testing	1 day	3/28/19 8:00 AM	3/28/19 5:00 PM	20	
22	📅	System Testing	1 day	3/29/19 8:00 AM	3/29/19 5:00 PM	21	
23	📅	Fixing bugs and errors	2 days	3/30/19 8:00 AM	4/2/19 5:00 PM		
24		**Reseach Paper**	**14 days**	**4/3/19 8:00 AM**	**4/22/19 5:00 PM**		
25	📅	Framing the paper	3 days	4/3/19 8:00 AM	4/5/19 5:00 PM	23	
26	📅	First review of the paper	1 day	4/13/19 8:00 AM	4/15/19 5:00 PM	25	
27	📅	Changes to the paper	2 days	4/16/19 8:00 AM	4/17/19 5:00 PM	26	
28	📅	Final review of the paper	1 day	4/20/19 8:00 AM	4/22/19 5:00 PM	27	
29	📅	Presentation	1 day	5/4/19 8:00 AM	5/6/19 5:00 PM		

Fig 2a. Project Schedule

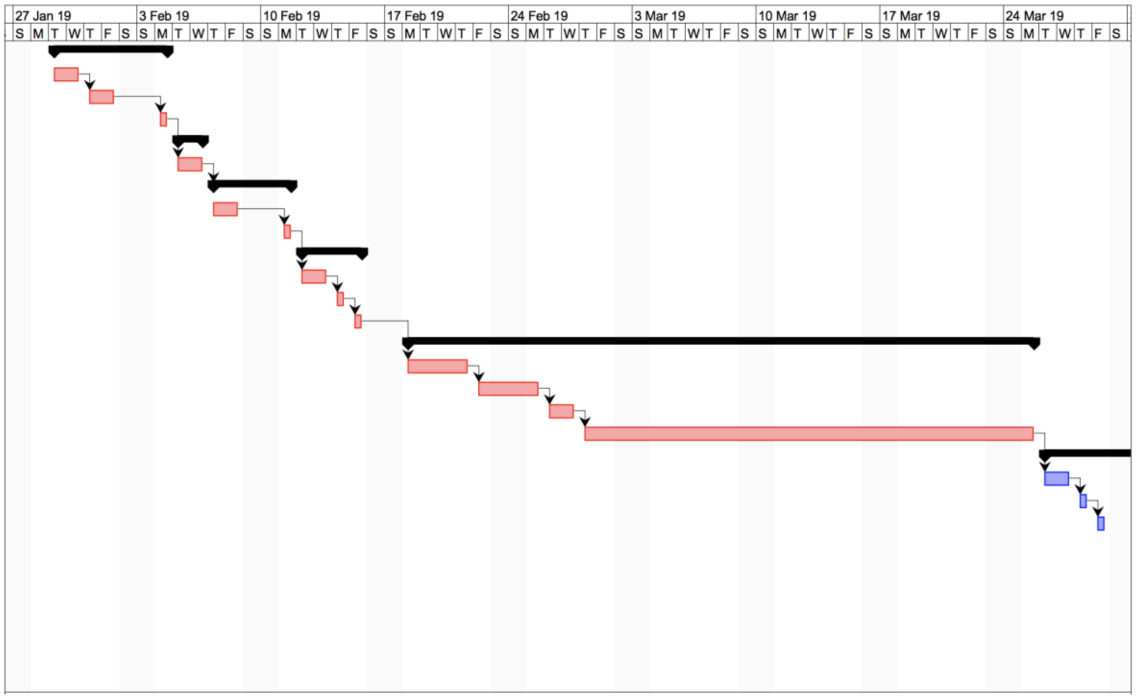


Fig 2b. Gantt Chart

4.2 Risk Management Plan

The following table enlists the risks and their probabilities of occurrence with the mitigation plan proposed to be used:

Risk	**Probability**	**Impact**	**Exposure**	**Mitigation**
Internet connection drops	High	High	High	Use a high quality internet connection
User enters wrong song title	High	High	High	Instruct the user to enter data correctly
Low Power Computer	Medium	Low	Low	Try to use a more powerful computer for faster output
Possibility of no dominant tone	Medium	Low	Low	Recommend songs with no dominant tone

Table 1: Risk Assessment

Chapter 5

SOFTWARE REQUIREMENT SPECIFICATION

5.1 Product Overview

In today's fast-paced technology driven world, the requirements to carry out day to day activities for an ordinary person are sought to be met out almost instantaneously, as time is now the most limited resource. When it comes to an individual's personal interests he or she must think carefully before allotting time in their busy schedules for their own leisure. This leisure activity may include for most people to be able to relax to some favourite tunes or discover new music from the myriad of different artists and various genres. There are a large number of aspects regarding the user experience in song provider applications that can elevated in different ways to enhance the experience of discovering new music, to acquire more details about one's favourite songs and a more a more novel approach to discovering music based on the mood of one to be able to identify the different emotions expressed by the music before having to listen to it completely. These are venues that can be explored using the advanced studies available to us now in the field of AI. Such an application includes song recognition, genre classification, topic extraction and analysis of sentiment or emotion expressed latent within the song. This application will be able to encourage music listeners to discover new music of their own liking and also at the same time help us to understand and study the vast expanse of applications of AI in this particular field. The project mainly aims at enhancing the experience for sound cataloguing by employing methods to further classify music as well as giving a description about the lyrics.

Intended Audience and Reading Suggestions:

The primary audience includes music enthusiasts and music reviewers that are interested in information about the song and understanding the tone or sentiment of a song before listening to it. The mood of a song once determined can be used to create select playlists to cater to the emotional state of an individual and can hence be used by music provider enterprises to create such playlists on their platforms. Researchers interested in observing the AI architecture used may also have a look into the working of the project to see the working of the model used and the reasons for the selected design.

Project Scope:

The project is given a song as an input that gives us the following functionalities: song recognition, obtain genre song belongs to, mood or sentiment expressed by the song, topic extraction, and other useful information. The inputs provided can be audio recordings, mp3 files, or wav files. The language is restricted to English and prefer only songs with lyrical content. Such an application available to a music enthusiast can be used to discover new music and can aid the user in deciding whether to listen to and add to their playlist a particular song based on information generated by the model from the contents of the song.

Music provider enterprises may use the information generation about a song's genre, mood and topics of interest to create a song description to characterize the song. These "mood tags" may be used for creating playlists catering to the emotional state of a user. This approach can also help already existing playlist generation machines improve accuracy and create better recommendations.

The project will produce an integrated architecture that can provide the following deliverables:

- **Song Recognition:** The song that is fed as the input to the application will be able to search a database and match the audio fingerprint to recognize the song. We use an API called ACRCloud service for this task.
- **Tone analysis:** Tone analysis is done with the help of IBM Watson's tone analyser in order to extract the top tones/emotions from the song which is given by the user
- **Polarity Analysis:** Based on the lyrics of the song given by the user, a basic positive negative polarity graph will be generated with the help of NLTK.
- **Word Cloud:** Word cloud will be created on the song that is queried by the user.
- **Topic Extraction:** For thisfeature anunsupervised learning model with LDA is used to obtain a few words description about the main theme of the song. This can be referred to as an aid in choosing songs for listening.

- **Tone Analysis:** The user interface features screens which allow the user to view the trends in the emotions of the top songs over the years.
- **Playlist Management**: Users are able to login to the system securely and add/remove songs from their personal playlist.
- **Song Recommendation:** Based on the song's in a users' playlist the application will be able to recommend similar songs based on the emotions in the songs in the existing playlist.

5.2 External Interface Requirements

5.2.1 User Classification:

- Music Provider Enterprises
- Music Enthusiasts or interested music listeners
- Researchers interested on accuracy of results

Characteristics:

Music Provider Enterprises can use the Audio Tagging Engine as a service for help in creating mood playlists and also the Topic Extraction feature to be able to attach descriptions to songs without the use of human reviewers.

Music enthusiasts can use these features to reduce the amount of time spent in discovering music.

Researchers can use the conclusions drawn from this project implementations in other areas of the field to obtain further knowledge in the domain.

5.2.2 Hardware Interfaces

- Minimum 4 GB of RAM
- Processor of 1.7 GHz or higher capacity
- Storage space of at least 500 MB
- Machine running x86 or x64
- Windows 7 or above, Mac OS 9 or above or Linux (Ubuntu 14.04 or above).

5.2.3 Software Interfaces

- Anaconda
- Jupyter Notebook
- CUDA
- TensorFlow
- Kivy

5.2.4 Communication Interfaces

The application communicates with the user by using the GUI Kivy. The user has different screens to view analysis, manage his playlist, and get analysis on a particular song. Kivy has been used as it is a Python framework and hence backend integration can be carried out in a simpler manner.

5.2.5 Design and Implementation Constraints

This application is only applicable to English songs and the genres that are identified are pre-trained and hence no new genre can be identified. The accuracy of the model depends on the amount of data used for training.

5.2.6 Assumptions and Dependencies

The assumption and dependencies made in the project are:

- The APIs used implement optimal and robust algorithms
- The song has distinctly understandable lyrics.
- The APIs used are under constant maintenance for any updates for optimization.
- Internet accessibility.
- The time taken to complete the project is assumed to be 3 months.
- The workability of the machine learning modules used with respect to the other tools and ease of integration.

5.3 Functional Requirements

As a part of the project, the system must be able to do the following:

- To get the song data and process it.
- To identify the title and artist of a song.
- To understand the mood of the song and specify tags for the mood.
- To extract the various topics in the song.
- To generate a word cloud of the queried song.
- To generate a polarity assessment of the given song
- To aid in enhanced song discovery and playlist creation experience with playlist recommendation.
- Allow the user to view trends of emotions in songs over the years.

Chapter 6

DESIGN

6.1 Introduction

An overview of the design through various diagrammatic representations is introduced in this section. The diagrams give a brief pictorial representation about the architecture of the proposed framework, the Graphical User Interface which provides an abstraction to the end user without revealing the details to the different API calls, crawling engine, LDA etc. Additionally, sequence diagrams and class diagrams shows the association and interaction between the various models.

6.2 Architecture Design

The figure below shows the system architecture. The application is user friendly where all the user has to do is play a song which is the input to the application. There are three features provided in the application. First is to give a song of user choice as input and get the artist and title of the song whose lyrics are obtained by crawling from the web. Final output has major tone of the song given by Watson API and most occurred words in the song via Wordcloud. The second feature is basic analysis of the songs over the course of different years. The third feature is a recommendation engine which takes user playlist as input and suggests songs of similar tone.

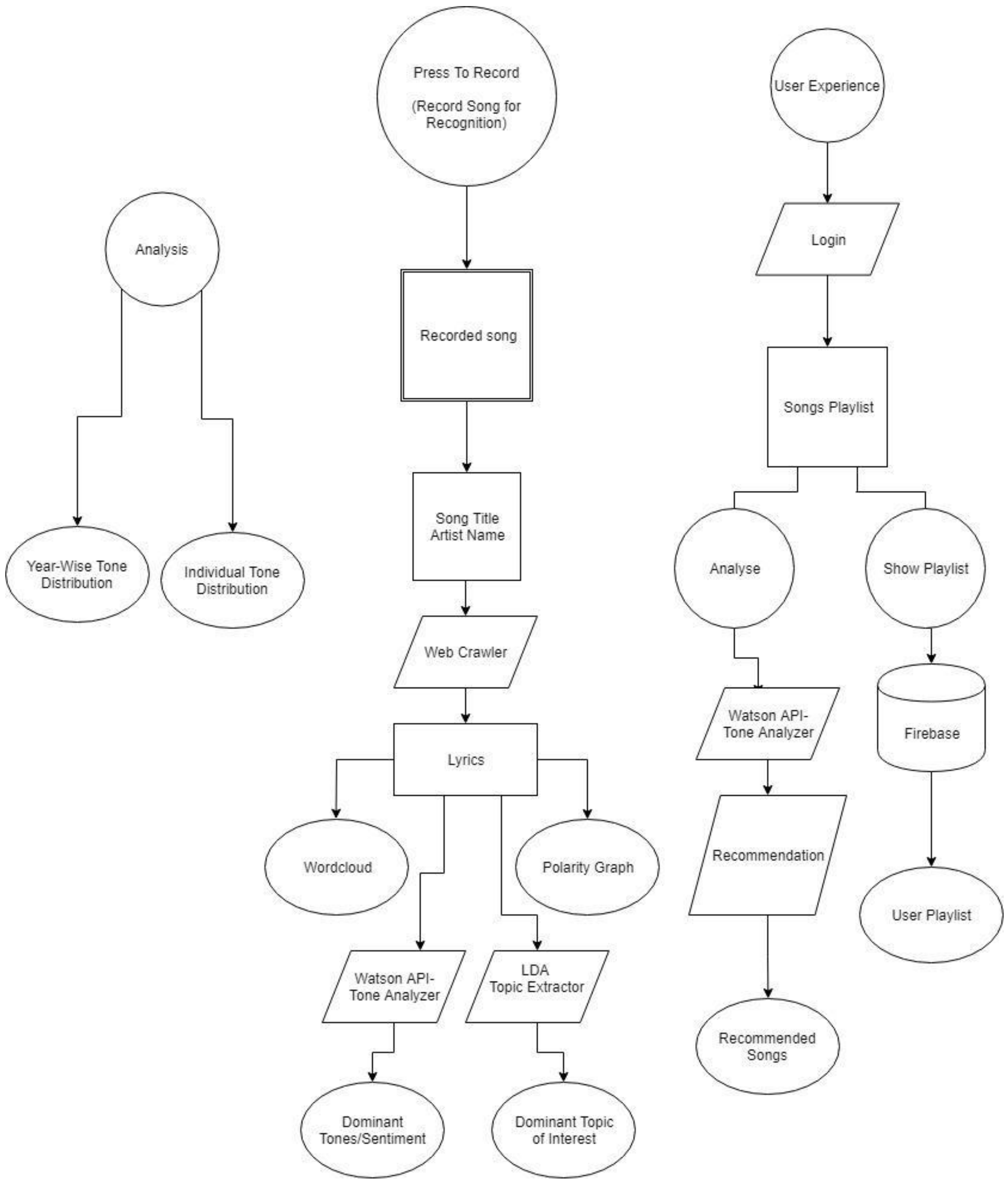


Fig 3. Architecture Design

6.3 Graphical User Interface

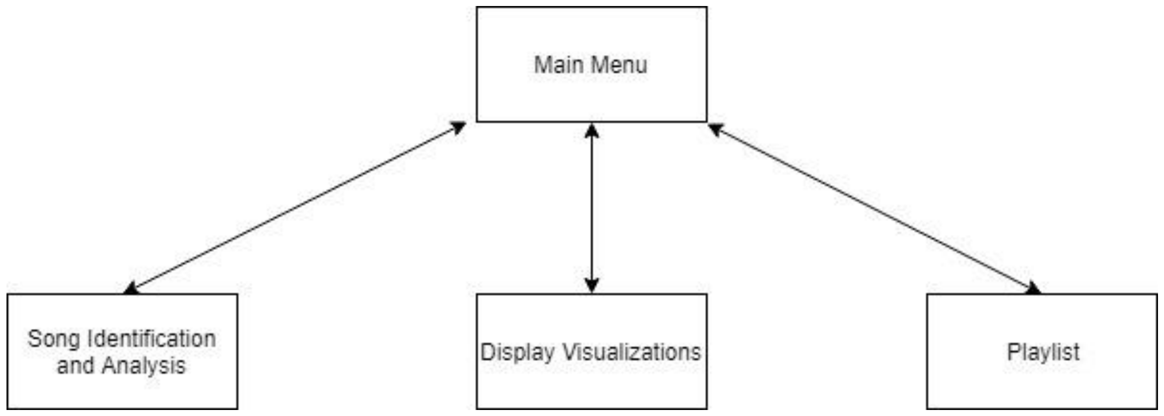


Fig 4. GUI Block Diagram

6.4 Class Diagrams and Classes

The class diagram shows the inheritance, dependencies and relationships which are involved in between the various classes. The individual boxes as shown in Fig 6.4 gives the class name, objects and the operations which are performed in that class.

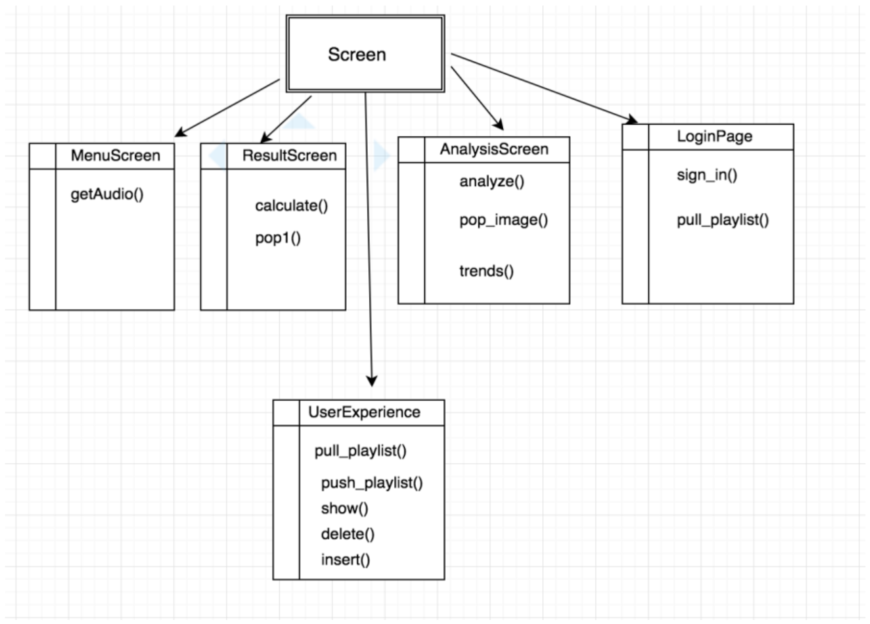


Fig 5. Class Diagram

6.5 Data Flow Diagram

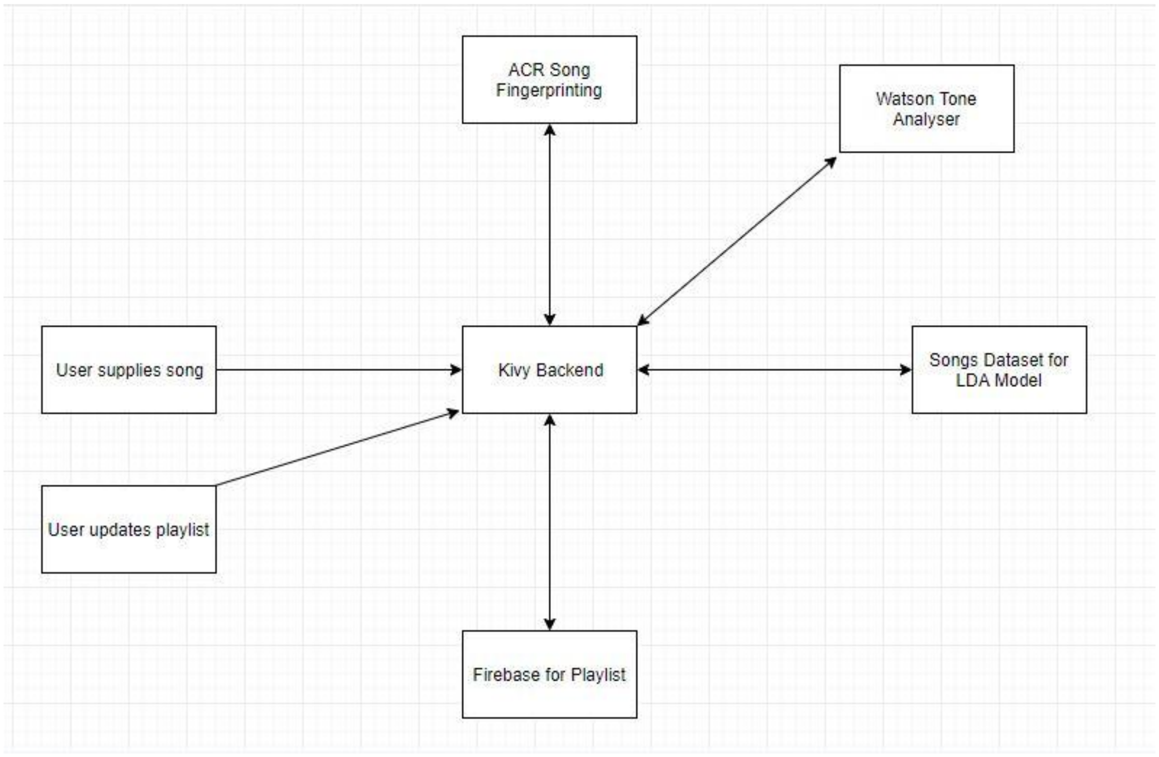


Fig 6. Data Flow Diagram

6.6 Conclusion

The architecture design is used for giving a brief overview of the working of the project, which depict the flow from input to output where user provides a song recording and gets multiple results related to the song. The graphical user interface provides the abstraction layer which hides the implementation of complex APIs and provides a user friendly application.

The sequence diagram shows how the activities and modules are interrelated to each other and communicate in a timely sequence of events. Data flow diagram depicts the input data being injected into the application which goes through several modification to give out final result.

Chapter 7

IMPLEMENTATION

7.1 Tools Introduction

The language used in this project was Python 3.x. The entire project was divided into parts and each of the developers worked in environments of their own comfort including Jupyter Notebook, PyCharm, Terminal, and Anaconda. The final code was integrated into an Anaconda environment and compiled using the Sublime Text editor. Essential libraries used include the following:

a.) Kivy: Used for front end application development.

b.) Pyrebase: Used for connecting to the Firebase cloud database.

c.) SciKit Learn: Used for data pre-processing tools and modelling the data.

d.) Numpy and Pandas: Used for data handling, transformation and calculations.

e.) Matplotlib.Pyplot : Used for plotting results for analysis.

f.) NLTK-Vader: Used for sentiment intensity analysis. NLTK consists of python libraries for natural language processing.

g.) BeautifulSoup: Used for web crawling.

The following APIs were included in the application:

a.) ToneAnalyzerV3 from Watson Cloud APIs: Used for sentiment and tone analysis.

b.) ACRCloud API: Used for song recognition.

7.2 Technology Introduction

Kivy: It is an open source Python library for rapid development of applications that make use of innovative user interfaces, such as multi-touch apps. Kivy runs on Linux, Windows, OS X, Android, iOS, and Raspberry Pi. It can natively use most inputs, protocols and devices including WM_Touch, WM_Pen, Mac OS X Trackpad and Magic

Mouse, Mtdev, Linux Kernel HID, TUIO. A multi-touch mouse simulator is included. Kivy is 100% free to use, under an MIT license (starting from 1.7.2) and LGPL 3 for the previous versions. The toolkit is professionally developed, backed and used. The framework is stable and has a well-documented API, plus a programming guide to help get started. The graphics engine is built over OpenGL ES 2, using a modern and fast graphics pipeline. The toolkit comes with more than 20 widgets, all highly extensible. Many parts are written in C using Cython, and tested with regression tests.

Pyrebase: A simple python wrapper for the Firebase API. Firebase is a cloud database service offered by google gives functionality like analytics, databases, messaging and crash reporting to move quickly and focus on the users.

Jupyter Notebook: The Jupyter Notebook is an open-source web application that allows to create and share documents that contain live code, equations, visualizations and narrative text.

SK Learn: Scikit-learn (formerly scikits.learn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. Scikit-learn is largely written in Python, with some core algorithms written in Cython to achieve performance. Support vector machines are implemented by a Cython wrapper around LIBSVM; logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR.

1. Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python.

2. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

3. The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn. This stack that includes:

- **NumPy**: Base n-dimensional array package
- **SciPy**: Fundamental library for scientific computing
- **Matplotlib**: Comprehensive 2D/3D plotting
- **IPython**: Enhanced interactive console
- **Sympy**: Symbolic mathematics
- **Pandas**: Data structures and analysis

Extensions or modules for SciPy care conventionally named SciKits. As such, the module provides learning algorithms and is named scikit-learn.

The vision for the library is a level of robustness and support required for use in production systems. This means a deep focus on concerns such as easy of use, code quality, collaboration, documentation and performance.

Pandas: *Pandas* is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. *Pandas* is a Python package providing fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis manipulation tool available in any language. It is already well on its way toward this goal.

Pandas is well suited for many different kinds of data:

1. Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet,
2. Ordered and unordered (not necessarily fixed-frequency) time series data,
3. Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels,

4. Any other form of observational / statistical data sets. The data actually need not be labeled at all to be placed into a pandas data structure.

The two primary data structures of pandas, Series (1-dimensional) and DataFrame (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering. For R users, DataFrame provides everything that R's data.frame provides and much more. Pandas is built on top of NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries.

Here are just a few of the things that pandas does well:

1. Easy handling of missing data (represented as NaN) in floating point as well as non-floating point data

2. Size mutability: columns can be inserted and deleted from DataFrame and higher dimensional objects

3. Automatic and explicit data alignment: objects can be explicitly aligned to a set of labels, or the user can simply ignore the labels and let Series, DataFrame, etc. automatically align the data for you in computations

4. Powerful, flexible group by functionality to perform split-apply-combine operations on data sets, for both aggregating and transforming data

5. Make it easy to convert ragged, differently-indexed data in other Python and NumPy data structures into DataFrame objects

6. Intelligent label-based slicing, fancy indexing, and sub setting of large data sets

7. Intuitive merging and joining data sets

8. Flexible reshaping and pivoting of data sets

9. Hierarchical labeling of axes (possible to have multiple labels per tick)

10. Robust IO tools for loading data from flat files (CSV and delimited), Excel files, databases, and saving / loading data from the ultrafast HDF5 format

11. Time series-specific functionality: date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging, etc.

NumPy: NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

·a powerful N-dimensional array object
·sophisticated (broadcasting) functions
·tools for integrating C/C++ and Fortran code
·useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

MatPlotLib: Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged.[3] SciPy makes use of Matplotlib.
Matplotlib was originally written by John D. Hunter, has an active development community,[4] and is distributed under a BSD-style license. Michael Droettboom was

nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012,[5] and further joined by Thomas Caswell.

matplotlib.pyplot is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.
In matplotlib.pyplot various states are preserved across function calls, so that it keeps track of things like the current figure and plotting area, and the plotting functions are directed to the current axes.

NLTK: The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language. It was developed by Steven Bird and Edward Loper in the Department of Computer and Information Science at the University of Pennsylvania.[5] NLTK includes graphical demonstrations and sample data. It is accompanied by a book that explains the underlying concepts behind the language processing tasks supported by the toolkit,[6] plus a cookbook.[7]

NLTK is intended to support research and teaching in NLP or closely related areas, including empirical linguistics, cognitive science, artificial intelligence, information retrieval, and machine learning.[8] NLTK has been used successfully as a teaching tool, as an individual study tool, and as a platform for prototyping and building research systems. There are 32 universities in the US and 25 countries using NLTK in their courses. NLTK supports classification, tokenization, stemming, tagging, parsing, and semantic reasoning functionalities.

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

VADER: VADER (Valence Aware Dictionary and sEntimentReasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. VADER uses a combination of A sentiment lexicon is a list of lexical features (e.g., words) which are generally labelled according to their semantic orientation as either positive or negative.

VADER has been found to be quite successful when dealing with social media texts, NY Times editorials, movie reviews, and product reviews. This is because VADER not only tells about the Positivity and Negativity score but also tells us about how positive or negative a sentiment is.

It is fully open-sourced under the MIT License. The developers of VADER have used Amazon’s Mechanical Turk to get most of their ratings, You can find complete details on their Github Page.

VADER has a lot of advantages over traditional methods of Sentiment Analysis, including:

- It works exceedingly well on social media type text, yet readily generalizes to multiple domains
- It doesn’t require any training data but is constructed from a generalizable, valence-based, human-curated gold standard sentiment lexicon
- It is fast enough to be used online with streaming data, and
- It does not severely suffer from a speed-performance tradeoff.

Beautiful Soup: Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with the parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.

Watson Tone Analyser API: The IBM Watson™ Tone Analyser service uses linguistic analysis to detect emotional and language tones in written text. The service can analyse tone at both the document and sentence levels. By submitting JSON, plain text, or HTML input that contains the written content to the service. The service accepts up to 128 KB of text, which is about 1000 sentences. The service returns JSON results that report the tone of your input. You can use these results to improve the perception and effectiveness of your communications, ensuring that your writing conveys the tone and style that you want for your intended audience.

ACR Cloud: ACRCloud provides a robust song recognition service that is fast and accurate and has the ability to share songs immediately, without needing to search. Automatic content recognition (ACR) is an identification technology to recognise content played on a media device or present in a media file. This enables users quickly obtain detailed information about the content they have just experienced without any text based input or search efforts. ACR can help users deal with multimedia more effective and make applications more intelligent.

7.3 Overall View of Project Implementation

Songs and music play important roles in the human mind to be able to resonate the feelings or emotions latent within the individual. If there was a way to study the kind of lyrics an average music enthusiast listens to, various studies can be conducted to make a number of observations on the details obtained from such a study, some of which the project will address later on. These studies can help us understand the trends of lyric tones or emotions reflected by popular artists in their work and listened to by the masses.

The application developed, LyriQuest, targets all music enthusiasts for usage. The novelty intended is the ability to choose music based on the sentiment or tone set by the lyrics of a particular song. This enables us to study the kind of lyrics used in songs and extract the underlying emotions that are being stimulated within the users who listen to a particular song.

The source code of LyriQuest was developed in Python 3.6 and the front end was coded using Kivy which a python web and application development framework. The application consists of a main menu where 3 choices are given to the users:

1) Tone Analysis

2) Press to Record

3) User Experience

1. Tone Analysis:

Clicking this button transports the user to a window where a series of analytical reports and collected data regarding songs passed through the tone analyzer are presented in an interactive graphical interface. This data shows the scores of the different sentiments or tones of the song lyrics, collected over 20 years. There are 2000 songs that are the top 100 songs over the years from 2000-2019 collected over the 2 decade from a trusted source website. This was done by formatting a list of the songs and their artist names into a text file and passing that through the Tone Analyzer Watson API. The scores thus obtained were plotted in the form of bar graphs and line graphs that can be viewed in this window.

2. Press to Record:

Clicking this button first records a song from the user. The recorded file is sent to the ACR Cloud recognizer which provides the user with the song title and other details of the song. These details are fed to a web crawler implemented using Beautiful Soup libraries. The web crawler gives us the lyrics of the song under focus. The lyrics once obtained are used in 4 different ways:

   1. A Wordcloud of the song lyrics can be observed, this is constructed using certain natural language tools of python libraries.
   2. The dominant tones reflected by the lyrics can be observed by passing them through the Tone Analyzer API.
   3. The dominant topics underlying in the lyrics are brought out using a Latent Dirchlet Allocation model trained on 250,000 songs labeled with their topics.
   4. A polarity graph using the VADER NLTK model was constructed to see how positive or negative the song was oriented.

3. User Experience

Clicking this button allows the user to login to a personal space wherein the user can store songs in a playlist which works with a Firebase backend database. The tone scores for the playlist songs are obtained and analyzed by comparing with the top 100 songs for each of the years in the past two decades. Upon comparison of the Euclidean distance of the score vectors of the playlist songs and normalized scores of each of the songs from the 2000 songs data, we can find the nearest neighbor vectors to the playlist songs and recommend similar scored songs from top songs data. Hence, implementing a recommendation system based on emotional content of the lyrics of the song rather than the music itself. The playlist can be manipulated as per the users will and may or may not add recommended songs to it.

7.4 Explanation of Algorithm and How it is Been Implemented

Main Menu

```
    Choose option between: Tone Analysis, Press to Record or User Experience
        If id==Tone Analysis:
        Options to See Year-wise or Tone-wise data visualization from dataset:
        Enter Year in text-box=x:
            Display bar graph of tonal distribution for given year x
        If options==Tone:
        Display line graph of tonal distribution across last 20 years
```

Song Analysis Module

```
        If id==Press To Record:
            Output.wav=Recorded audio from user
        Re=ACRCloudRecognizer(output.wav) in re.title, re.artist
        Lyrics=BeautifulSoup.find(title,'lyric_box')
        Dominant_tones=ToneAnalyserV3(lyrics)
        Extracted_topics=Lda(lyrics)
        If id==Result:
        Display song title, artist name, dominant tones, topics extracted
        If id==WordCloud:
```

```
        Display wordcloud for retrieved song lyrics
        If id==Polarity Analysis:
        Vader.sentimentanalysis(lyrics)
        Display polarity chart (positivity and negativity in lyrics)
        If id==Back to menu:
        Go back to Main Menu Page
```

Playlist Management

```
        If id==User Experience:
    UserAuth(username,password) to authenticate user
        If id==Show Playlist:
            Display songs from user playlist retrieved from Firebase
        Text_box=song_title
            If option==add song:
                Add(title)
                If option==delete song
        Remove(title),
        Update Playlist in Firebase data
```

Reccomendation

```
            If id==Analyse:
        BeatifulSoup.find(lyrics,song title)
        Tone_score=ToneAnalyserV3(lyrics)
        Min_max_normalization(Tone_score)
                Min_max_normalization(Scores of songs in Database)
        Recommended songs=Nearest_Neighbours(5)
```

```
If id==Press for Result:
Display recommended songs
If id==Exit:
Go back to Main Menu
```

Explanation:

The application consists of a Main Menu with 3 options given to the user: Tone Analysis, Press to Record, and User Experience. The tone analysis option allows the user to a graphical user interface to observe the data collected after passing the 2000 songs dataset with their tonal scores after passing the lyrics to the Tone Analyser API. The interface allows the user to enter the year for which the visual representations for each of the tonal scores are displayed. The tone wise distribution across the years can also be observed. The second option is to record a song and analyse it using different models, as mentioned in the algorithm, providing certain details about the song. The results display include the recognized song title, artist name, lyrics, dominant tones, main topics extracted, lyrics wordcloud, and polarity charts. The third option gives the user ability to login create a playlist and add or remove more songs. The provision of song recommendation is also given where the top 5 songs, close to the tonal scores of the playlist, are displayed. These songs are provided from a dataset of top 100 songs over the last two decades.

7.6 Information about the implemented Modules

The project consists of the following modules implemented:

- **Song Recognition:** The song that is fed as the input to the application will be able to search a database and match the audio fingerprint to recognize the song. We use an API called ACRCloud service for this task.
- **Tone analysis:** Tone analysis is done with the help of IBM Watson’s tone analyser in order to extract the top tones/emotions from the song which is given by the user
- **Polarity Analysis:** Based on the lyrics of the song given by the user, a basic positive negative polarity graph will be generated with the help of NLTK.
- **Word Cloud:** Word cloud will be created on the song that is queried by the user.
- **Topic Extraction:** For thisfeature anunsupervised learning model with LDA is used to obtain a few words description about the main theme of the song. This can be referred to as an aid in choosing songs for listening.
- **Tone Analysis:** The user interface features screens which allow the user to view the trends in the emotions of the top songs over the years.
- **Playlist Management**: Users are able to login to the system securely and add/remove songs from their personal playlist.
- **Song Recommendation:** Based on the song’s in a users’ playlist the application will be able to recommend similar songs based on the emotions in the songs in the existing playlist.

Each module is independent and have been developed to add to the usefulness of the application in their own way. The functionalities combined concurrently work to analyze and provide details about the song that will give further insights to the user about songs being listened to.

7.7 Conclusion

The overall implementation of the project has been made possible by developing each module individually. The algorithm which describes the flow and the working of the project shows how the user can navigate through the application and find various useful and interesting results done from studies and analysis. The different models are used to achieve these results. The reasons for using NLTK VADER for polarity charts include accuracy of the model and efficient working. The utilization of LDA can be justified due the speed at which this model provides results. The nearest neighbour vector recommendation system based on Euclidean distance is the best method to find similar songs one may like using space vectors. Hence, we justify the implementation and tools used to carry out development of this application.

Chapter 8

TESTING

8.1 Introduction

Testing of a deep learning model implies testing the accuracy of the model based on human's intentions and how a human depicts the same object as depicted by a machine. The methods for testing the system involves unit testing which implies the developer who was assigned a particular code makes sure the code is tested in a testing environment before bring the code to deployment or available to the end users.

8.2 Testing tools and Environment

The main testing tool here would be developing a good test dataset consisting of song lyrics from various different genres. While testing, since the accuracy dropped during different genres the model and not deprive the accuracy percentage by a huge margin and result in a fewer false positive predictions by the model. The validation set helped in analyzing and testing how well the code was tight bound when different unit codes was integrated.

8.3 Test Cases

Use Case	Test Case	Description	Procedure	Result
1.Song Identification	Correct ID of the song	On press. user records a song	Song is queried to ACR Cloud	Song is identified and displayed
2. Song Tone Analysis	Are tones identified	Upon recording a song and crawling of lyrics, tones are extracted	Lyrics are crawled with Beautiful Soup and IBM Watson are used to get the tones	Tones are identified and displayed
3. Song Topic Analysis	Are topics identified	Upon recording a song and crawling of lyrics, topics are extracted	Lyrics crawled with Beautiful Soup and LDA is used to get the tones	Topics are identified and displayed

4. Updating Playlist	Adding/Removing songs to playlist	User logs in and adds/deletes song	Records are written to Firebase DB	Playlist is updated and displayed

Table 2. Test Cases

Chapter 9

RESULTS AND PERFORMANCE ANALYSIS

9.1 Results Screenshots

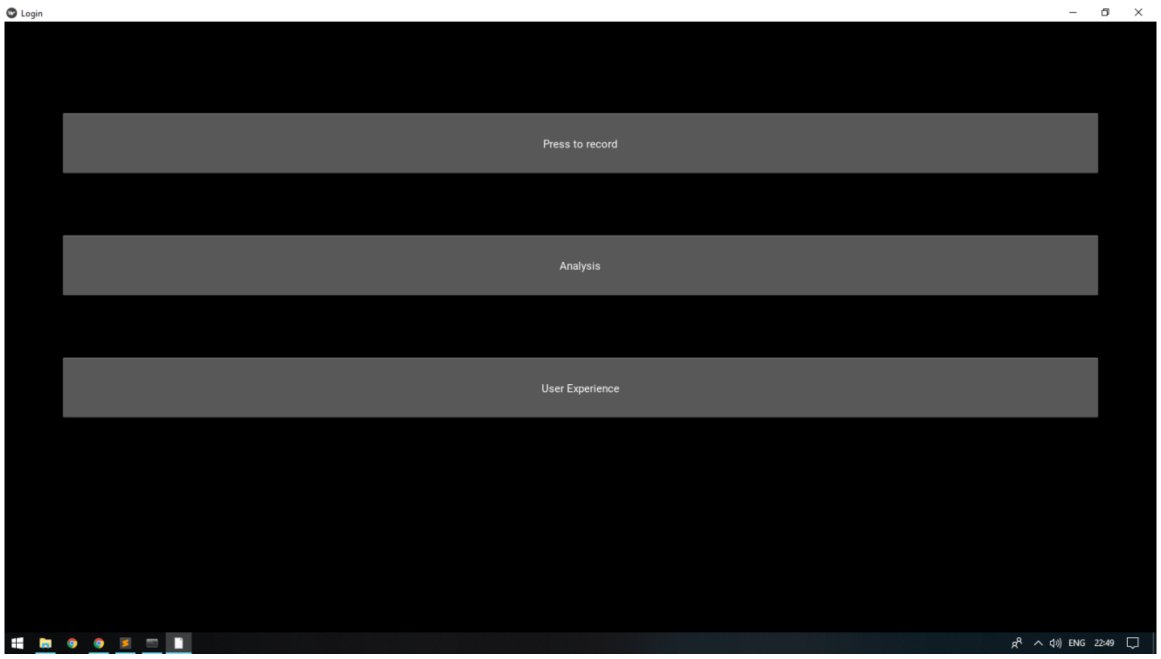
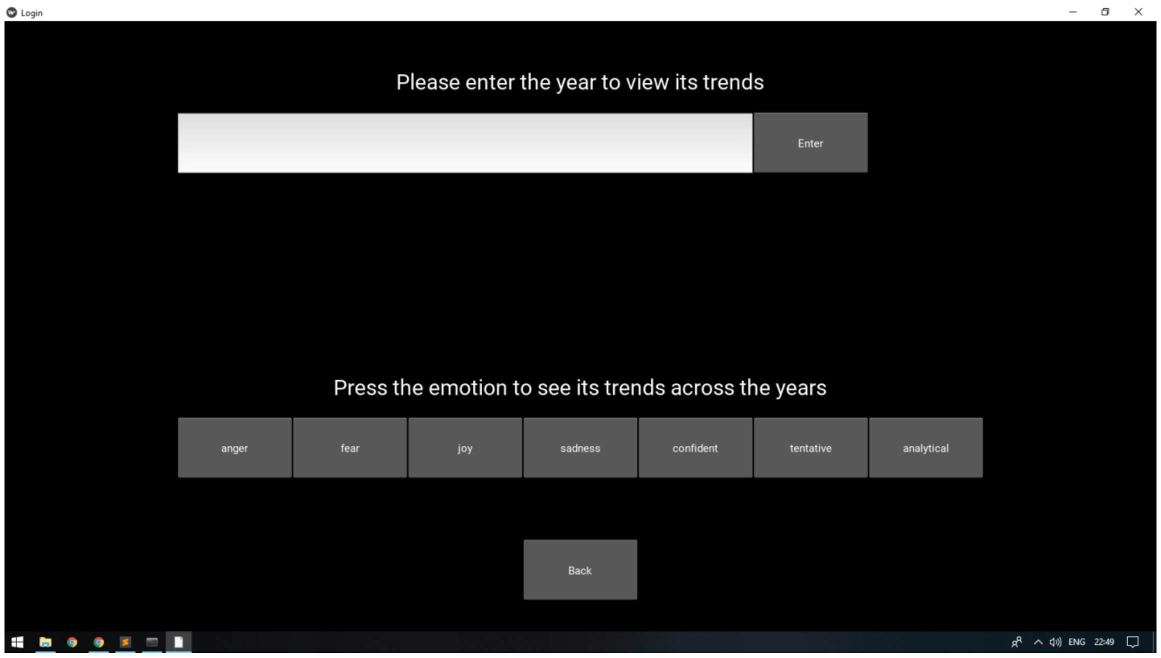


Fig 7.Home Screen – Allows user to select menu option



Please enter the year to view its trends

Press the emotion to see its trends across the years

Fig 8.Analysis Page to view graphs

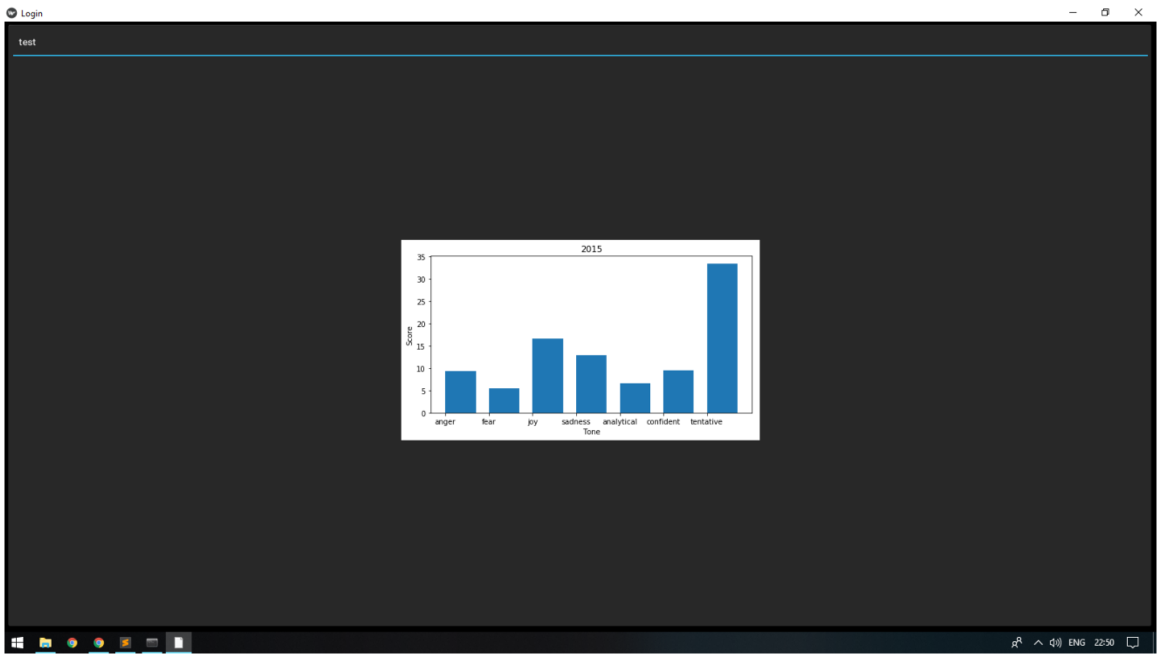


Fig 9.Graph Display Page

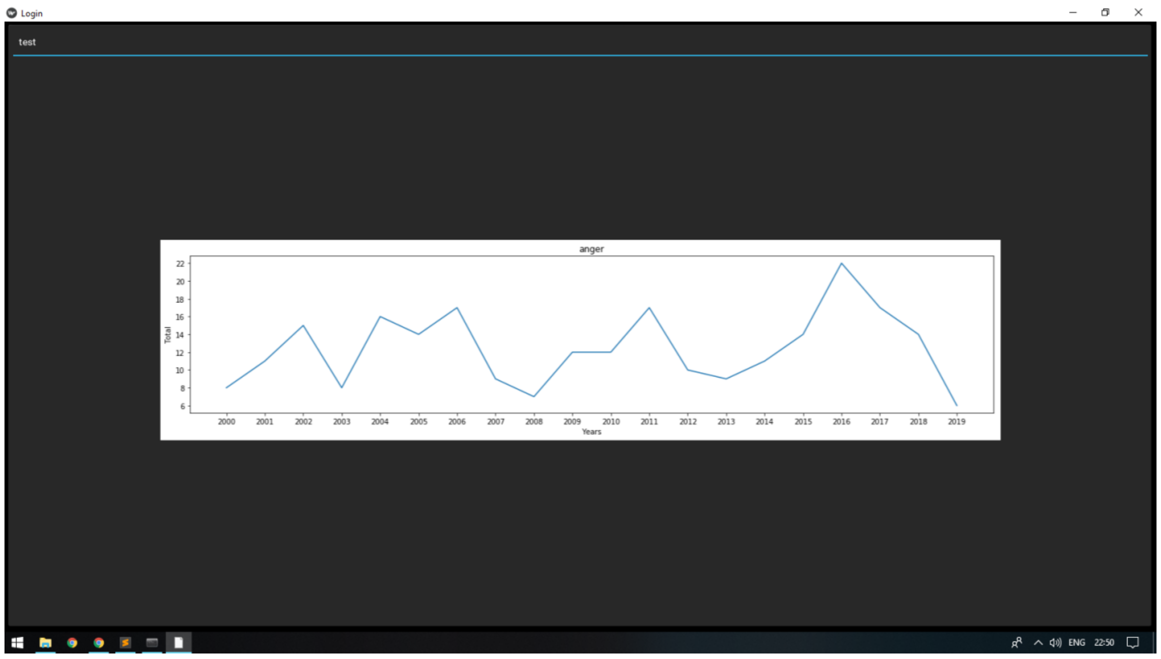


Fig 10 .Graph Display Page Cont.

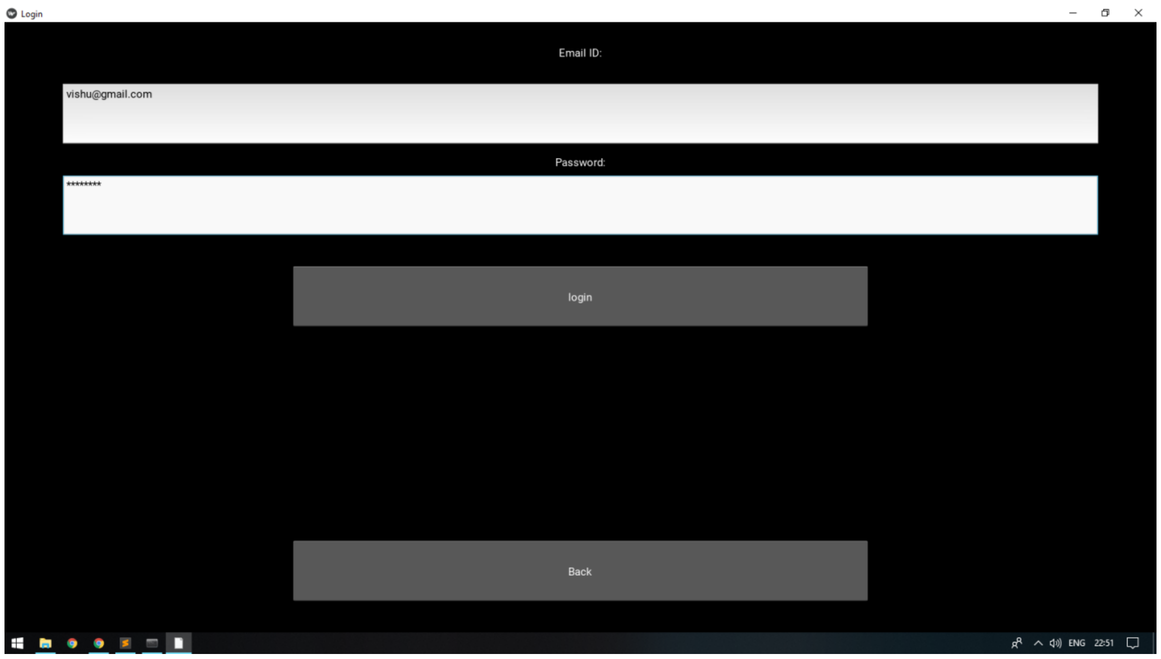


Fig11. User Experience Login Screen

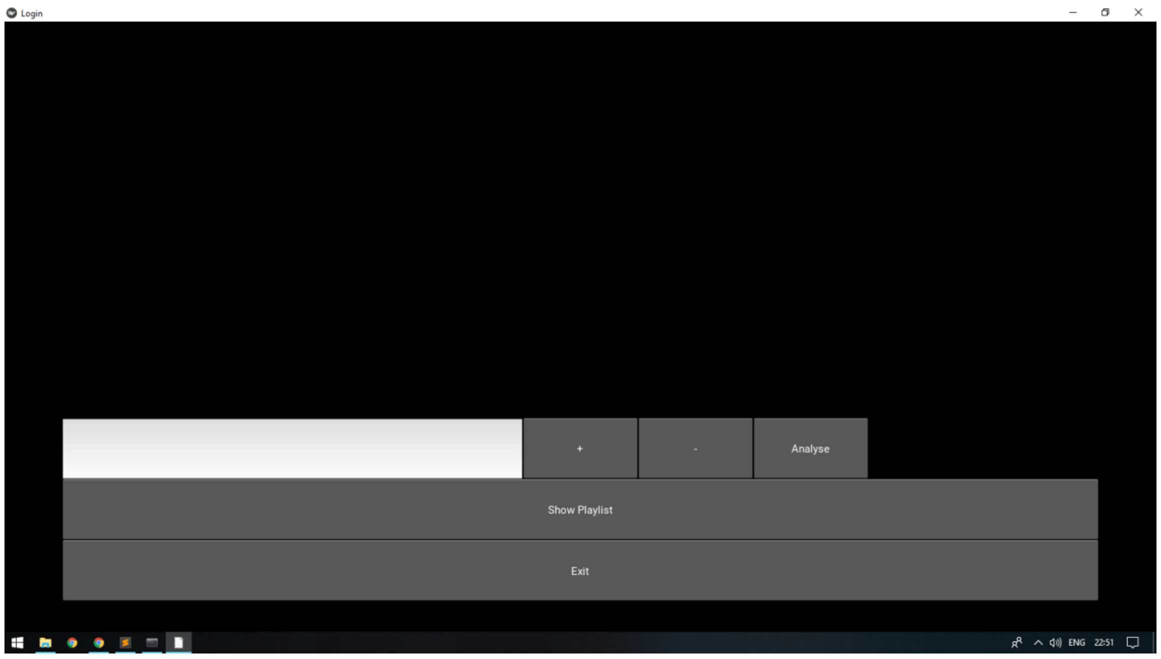


Fig 12.Playlist Menu to add and delete songs

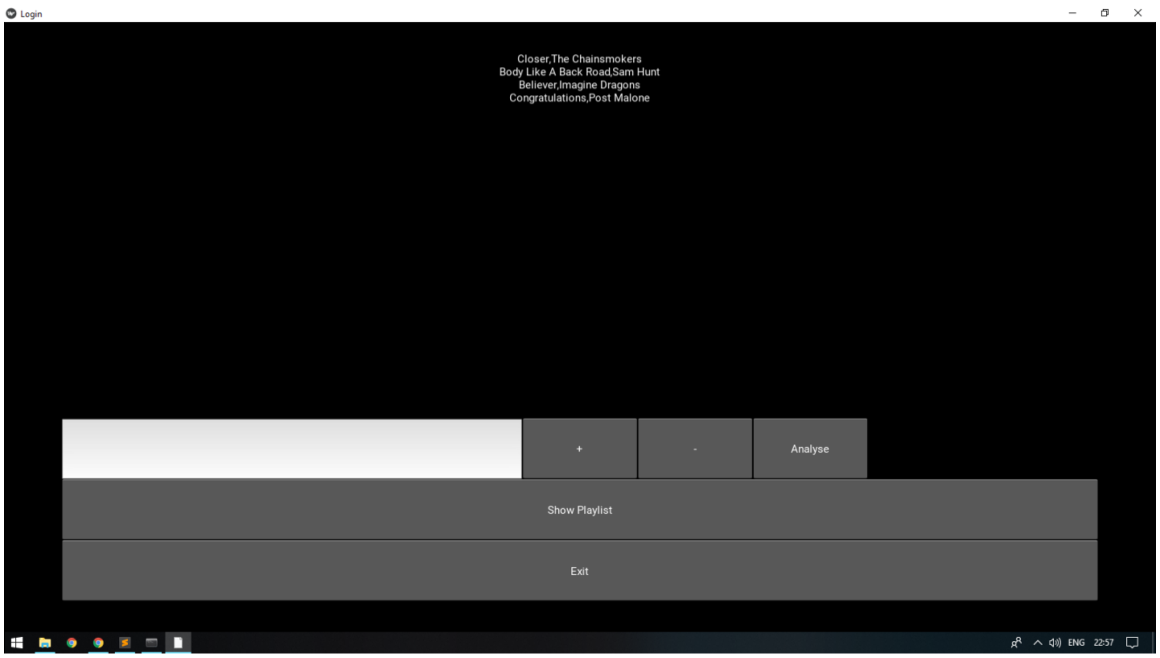

Closer,The Chainsmokers
Body Like A Back Road,Sam Hunt
Believer,Imagine Dragons
Congratulations,Post Malone

Fig 13.View Playlist Option

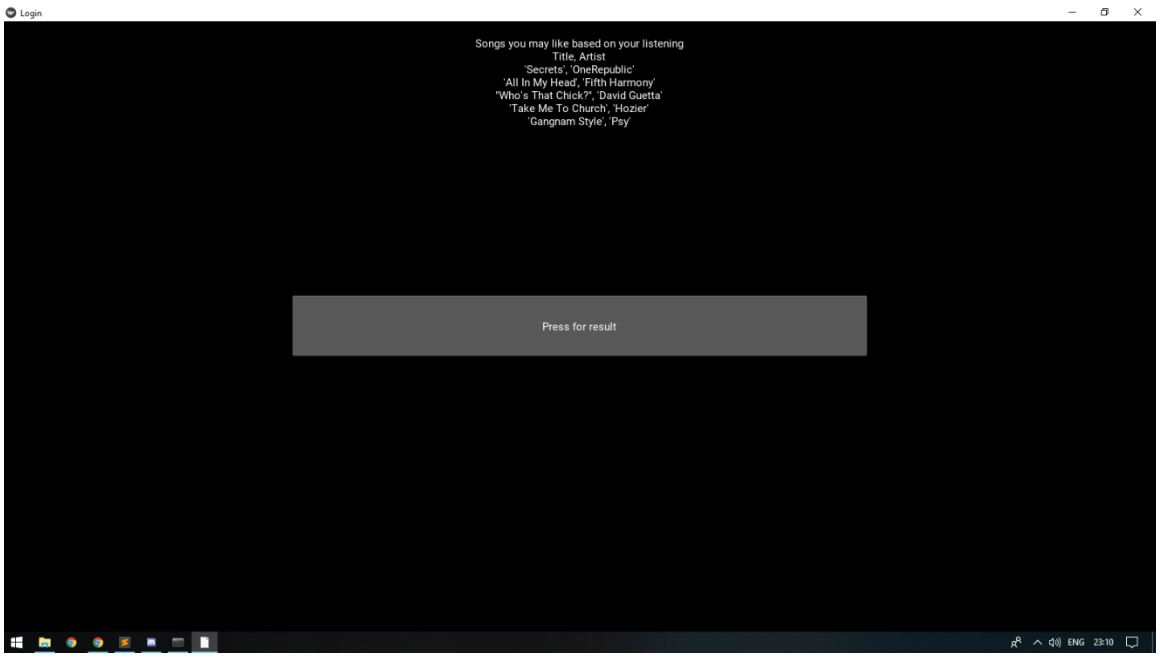
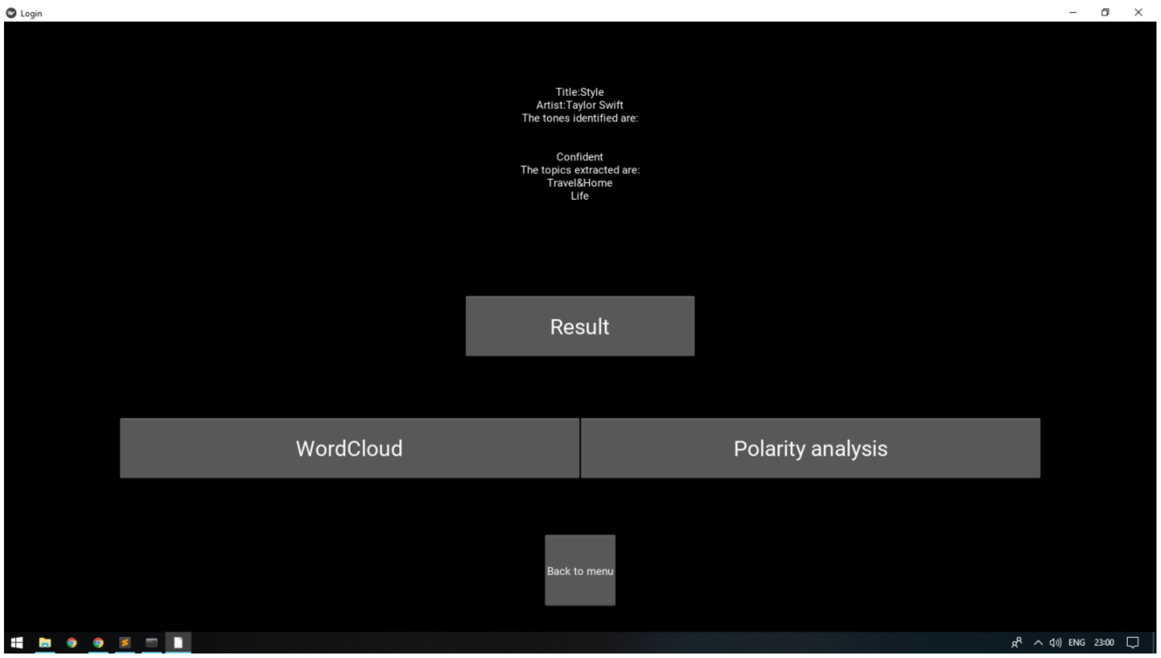

Songs you may like based on your listening
Title, Artist
'Secrets', 'OneRepublic'
'All In My Head', 'Fifth Harmony'
"Who's That Chick?", 'David Guetta'
'Take Me To Church', 'Hozier'
'Gangnam Style', 'Psy'

Fig 14.List of Recommended music



Title:Style
Artist:Taylor Swift
The tones identified are:

Confident
The topics extracted are:
Travel&Home
Life

Fig 15.Song ID Results Screen with Tones and Topics

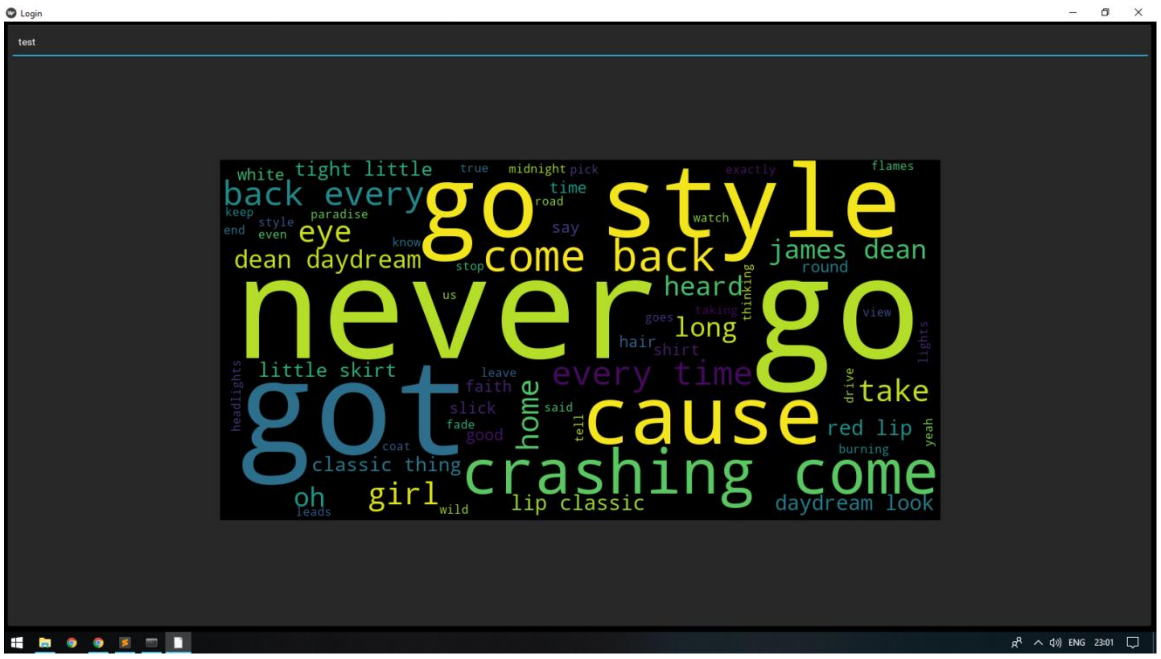


Fig16. WordCloud for recorded song

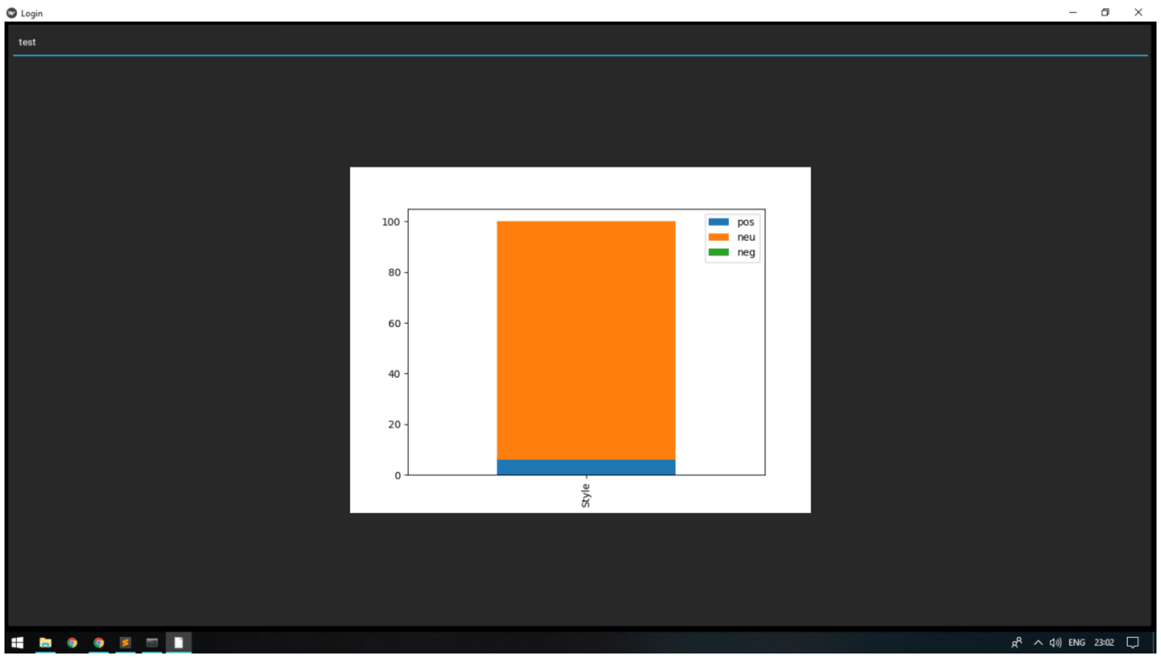


Fig 17.Polarity Analysis for recorded song

Chapter 10

CONCLUSION AND FURTHER SCOPE

From the above analysis we can conclude that we have started a small application which can prove to be beneficial to music lovers and composers in the future. The application is able to identify a piece of music, extract the tones, topics and give some visualizations on the same. This helps give quantification to textual data embedded within the lyrics of a song. These data points can be used in the future to carry out further analysis.

There are a set of improvements and feature additions which can be made to the application which will make it more versatile and novel. A faster, more accurate model can be created for topic extraction and the overall efficiency of the application can be increased. Features such as multi-level recommendation and the ability to search for songs on the internet can be added.

It is easy to forsee production level applications with similar features be developed in the future to be used with music applications such as Spotify and Google Play Music. Music recommendation engines can be greatly enhanced with the inclusion of topic and theme extraction. In the end we do hope our application has a positive social impact and changes the way we filter our music playlist.

Chapter 11

REFERENCES

References:

[1]. A Review of Audio Fingerprinting, PEDRO CANO AND ELOI BATLLE : Music Technology Group, IUA UniversitatPompeuFabra, Ocata, 8 08003 Barcelona, Spain; TON KALKER AND JAAP HAITSMA Philips Research Laboratories Eindhoven, Prof. Holslaan 4, 5656 AA Eindhoven, The Netherlands

[2]. "Echoprint." Open Source Music Identification. Web. 20 Apr. 2015.

[3]. "We Know Music..." The Echo Nest.Web. 12 June 2015.

[4]. Online music recognition: the Echoprintsystem :Cors Brinkman, ManolisFragkiadakis, XanderBos from Leiden University, 2015

[5]. MUSICAL GENRE CLASSIFICATION USING SUPPORT VECTOR MACHINES ChangshengXu, Namunu C. Maddage, Xi Shao, Fang Cao, Qi Tian

[6]. https://console.bluemix.net/apidocs/tone-analyzer

[7]. Effective Attention Modeling for Aspect-Level Sentiment Classification Ruidan He†‡, Wee Sun Lee† ,HweeTou Ng† , and Daniel Dahlmeier‡ †Department of Computer Science, National University of Singapore ‡SAP Innovation Center Singapore

[8]. Unsupervised Topic Modeling for Short Texts Using Distributed Representations of Words Vivek Kumar Rangarajan Sridhar∗ AT&T Labs

[9]. Twelfth International Multi-Conference on Information Processing-2016 (IMCIP-2016) Unsupervised Concept Hierarchy Learning: A Topic Modeling Guided Approach V. S. Anoopa, S. Asharaf and P. Deepak