

Social connection with Friends and Music Recommendation

Submitted in partial fulfillment of the requirements
of the degree of
Bachelor of Computer Engineering

B. E. Computer Engineering

By

Vishakha Mistry	Roll No. 31	PID: 182072
Siddhant Mishra	Roll No. 32	PID: 192261
Hitanshu Panchal	Roll No. 34	PID: 192263
Crystal Fargose	Roll No. 73	PID: 182034

Guide:

Ms. Priya Karunakaran

Assistant Professor



Department of Computer Engineering
St. Francis Institute of Technology
(Engineering College)

University of Mumbai
2021-2022

CERTIFICATE

This is to certify that the project entitled **“Social connection with Friends and Music Recommendation”** is a bonafide work of **“Vishakha Mistry(31), Siddhant Mishra(32), Hitanshu Panchal(34), Crystal Fargose(73)”** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of B.E. in Computer Engineering

(Ms. Priya Karunakaran)
Guide

(Dr. Kavita Sonawane)
Head of Department

(Dr. Sincy George)
Principal

Project Report Approval for B.E.

This project report entitled (*Social connection with Friends and Music Recommendation*) by (*Vishakha Mistry, Siddhant Mishra, Hitanshu Panchal, Crystal Fargose*) is approved for the degree of *B.E. in Computer Engineering*.

Examiners

1. _____

2. _____

Date: 04-05-2022

Place: Mumbai

Declaration

We declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Vishakha Mistry (31)

Siddhant Mishra (32)

Hitanshu Panchal (34)

Crystal Fargose (73)

Date: 04-05-2022

Abstract

Music is the art of composing sounds in time to produce composition through elements of melody, harmony, rhythm, and timbre. And it's probably not difficult to understand how certain music may mirror people's own thoughts and feelings. Lyrics or melodies can potentially communicate the message that anyone wants to convey. Songs can capture an emotional state or personal situation in the best way possible. Music is a mechanism for humans to connect with one another. Making a connection with people in new places or just looking for a change in scenery, meeting new people can be as hard as it is fun. We have found that people with the same music taste make up the best connection. So why not bring together people who like the same music, to share experiences which are currently missing out on. So, this application connects with people nearby who have the most in common with. The matchmaking algorithm of the application system filters out people based on the music playlist created and based on the genre and the artist of the music. All people have to do is share the playlist of songs and start making connections with new people and hang out on the social network provided by the application.

Contents

Chapter		Contents	Page No.
1		INTRODUCTION	1
	1.1	Description	1
	1.2	Problem Formulation	2
	1.3	Motivation	2
	1.4	Proposed Solution	3
	1.5	Scope	3
2		LITERATURE REVIEW	4-8
3		SYSTEM ANALYSIS	9-12
	3.1	Functional Requirements	9
	3.2	Non-Functional Requirements	9
	3.3	Specific Requirements	9
	3.4	Use-Case Diagrams and description	10
4		ANALYSIS MODELING	13-14
	4.1	Sequence Diagram	13
	4.2	Timeline Chart	14
5		DESIGN	17-23
	5.1	Architectural Design	17
	5.2	Database Design	19
6		IMPLEMENTATION	24-44
	6.1	Dataset	24
	6.2	Spotify Music Trends Analysis	24

	6.3	Friends Suggestion	27
	6.4	Music Recommendation	29
	6.4.1	Preprocessing and Feature Extraction	29
	6.4.2	Classification Algorithms	32
	6.4.3	Code	33
	6.4.4	Result Analysis	38
	6.5	Output	38
7		CONCLUSION	45

References

Acknowledgments

List of Figures

Fig. No	Figure Caption	Page No.
3.4	Use Case Diagram	10
4.1	Sequence Diagram of System	13
4.2	Timeline Chart	14 - 16
5.1	System Flow Diagram	17
5.2	Friend Suggestion System Block Diagram	18
5.3	Music Recommendation System Block Diagram	19
5.4	Authentication Methods	20
5.5	Registered Users on Google Firebase Auth	21
5.6	Cloud Firestore Database	22
5.7	Firebase Storage	23
6.1	Music Trend over the years	24
6.2	Clustering of Top 10 Music Genres	25

6.3	Song Cluster Based on Song Features	26
6.4	Content Based Music Recommendation	26
6.5	Genre Similarity Score	28
6.6	Principle Components Analysis on MFCC Coefficients	30
6.7	Principle Components Analysis on Chroma Vectors	31
6.8	Spectral Centroid for a sample audio file	32
6.9	Gaussian Naive Bayesian Accuracy Plot	38
6.10	Logistic Regression Accuracy Plot	38
6.11	Login Screen	39
6.12	Sign Up Screen	39
6.13	User Profile	40
6.14	User's Posts Screen	40
6.15	User's Profile Interaction	40

6.16	Home Screen	40
6.17	Music Player with Remote Spotify	41
6.18	Upload Post	41
6.19	Upload Posts - 2	42
6.20	Editing Post's Caption & Location	42
6.21	User's Chats	43
6.22	Chat Screen	43
6.23	Search Users to Interact	43
6.24	Friends Suggestions Screen	43
6.25	Friends Profile	44
6.26	User's Activity Feed Screen	44

Chapter 1

Introduction

Social media has emerged as part of our day by day lives as it's a way to maintain contact with friends. Most of the social networking sites like Facebook, Instagram, snapchat recommend friends on the basis of friend of friend or who are being followed. The actual situation is whether or not they may have common interests. People with similar music taste are found to make better connections as it becomes a tool for communication and gives a common ground. Also, Nowadays people have started to use music streaming platforms to listen to music but because of the sheer volume of choices, it becomes hard for the users to find the songs they like. It becomes a task to Adapt to the user's current context and recommend music. So we propose a social media application which recommends friends as well as music on the basis of musical interests.

1.1 Description

Facebook, Snapchat, Instagram & Twitter are some of the most popular social networking sites. These social networks come from people who engage or interact with each other or people who are suggested by social network recommendation algorithms. A social network can be represented as a social graph. In social networks, a user may periodically communicate with his friends by commenting or liking their posts etc. And as a graph the edges are weighted and in a user's social network the edge's weight represents how close the friend is related to its other friend, the edge is bidirectional to represent the mutual friendship. The weight of the edge is computed based on the number of interactions between two users. Based on an interaction ranking algorithm mainly for capturing the email interactions among users, we present a new interaction algorithm for suggesting friends based on factors such as music interests by summing the user's spotify music playlist, their location and the number of times they liked music-related posts. Each of these factors is also associated with a corresponding weight capturing the importance of such factors in the generation of the preference matrix between users. Based on the weighted sum, we compute the similarity of the target users' tags and potential recommendation. We then rank and select the highest rated users as recommended friends.

In the field of music recommendation, adapting recommendations to the user's current context is critical as, throughout the day, users listen to different music in numerous different contexts and situations. In order to meet people's personalized demand for music, some music

recommendation systems have been developed, such as LastFm and Pandora abroad, radio station, and Xiaomi music map in China, which show the interface of radio station and Xiami music. These music recommendation systems first establish their own music library, then analyse the characteristics of songs and users' listening habits, and then make recommendations for users. Previous research has shown that information about the context of a user (e.g., time, location, occasion, or emotional state) is vital for providing suitable personalized music recommendations as people listen to different music during different activities. Analysis of the acoustic features (e.g., tempo or danceability) of the tracks contained in individual playlists along with contextual characteristics may be used as a leverage for music recommendation. For a more personalised music recommendation, we present music recommendations based on the user history as well as its current context. The system gathers each user's listening history with the help of the user's spotify profile and generates listening patterns so that the users are able to enjoy the preferable music without checking all the music information such as title, singer and genre.

1.2 Problem Formulation

With rapid development of technology and software in the current era of big data, high volumes of a wide variety of valuable data with different veracity can be easily generated which can be collected at a high velocity [5]. Much useful information and knowledge are embedded in this data. Social media and social networks are rich sources of big data. They are referred to as rich sources as they have become a necessity in daily lives of people and a way for them to keep in touch with friends and share events and incidents. Finding compatible people to be friends on social media can be a challenge since the real concern would be common interests between people. A solution to this is to develop a social media application which recommends friends to the user with similar music taste. Also, it recommends music based on the content of that music. Using music as a primary factor to cultivate social connections can prove to be a great experience for the users.

1.3 Motivation

Everyone has their personal networks of friends regardless whether or not they met on-line or in person. The statistics of global internet usage suggests an average of 135 minutes is spent by

internet users per day on social media applications and is increasing every year. The basic goal of any social media application is to connect people and build relationships among brands, companies and individual users. A social media application with key features makes for a better user experience. With several features for social interaction an additional feature where a user can play songs based on it's song history which will allow a better experience as it is not realistic to choose a wanted song among all these songs because of immeasurable time consumption of the work.

1.4 Proposed Solution

The proposed system is divided into different functionalities working together to provide the result. The goal of the Friend Recommendation algorithm is to recommend friends based on common music interests. To do that, we need to value the common music interest the most by assigning a higher importance factor of common music listened. Music-related posts and likes is believed to be a great source for indicating common music interests and interactions as this would contribute more to the interaction score calculation than general comments and likes. And even the music playlist of the user will be used and its features would be extracted. Comparing different genres and unique differences in sound using audio features of songs and genres. The features extracted would then be used for recommending friends as well as music. And several other social network features for the system. REST API to fetch music playlist of users from Spotify after authentication [17]. Friends and music recommendations based on the content(features) of the music using clustering techniques. Location based filter which helps to find nearby connections. Social media application on cloud storage.

1.5 Scope of the Project

A social media application is designed for any individual or a company who seeks a larger audience and interacts with them. A system that uses a Spotify playlist and list of friends to suggest friends & a music recommendation system integrated with a social media application provides a personalized experience for the user. The system also helps suggest nearby friends to the user. It helps the user find compatible friends which makes them better connections. Along with the key features of any social media application, it additionally provides the user an experience of enjoying music with friends.

Chapter 2

Review of Literature

With the development of the Internet, social networks are also developing rapidly. It is difficult for users to find the friends they are interested in among a large number of users. In a social network, there are a large number of nodes with many social features, as well as massive interactions and relations among users [9]. More specifically, a social network could be viewed as a heterogeneous network no less than a traditional complex network, where fellowship/friendship and interactions (like review and retweet) could imply the underlying information of relevance [9]. The role of each relationship varies along with the user's preferences. Users in a social network sharing common or similar features and interests have high similarities or relevance in several aspects. The existing methods like analogous Pagerank algorithms are relatively time consuming, and some influence computation did not embody true influences of the user [9]. With regard to social relations and interactions, trust that is extracted from them is widely leveraged to recommend friends [9].

2.1. Friend Suggestions

2.1.1. Social Graph and Bipartite Graph

A social graph with weighted edges, which express the relationship strength between a user and his implicit groups. The core criteria are to compute the edge weights with frequency, recency, and direction. In order to satisfy these criteria, an interaction ranking algorithm—which sums the number of friends' comments and page likes between a user and another particular user that weighs each interaction between friends as a function of its recency—was developed [5]. The aforementioned interaction ranking algorithm computes the interaction score based on the time of each interaction activity to the current time, and it will weigh more if this operation is newer. In contrast, our proposed algorithm computes the interaction score based on the implicit social graph [5].

2.1.2. K-means clustering

K-means clustering to classify objects based on distinct features into a K number of groups. They first determined the number of groups K, decided the centroid of each cluster, and computed the

interaction of the rest samples to the centroid. Computation of interaction is repeated until convergence is achieved. The K-means approach converges quickly and is highly scalable. However, as the clustering result is based on the K centroids (instead of medoids), there may be no actual data that match the centroids [11]. K-means cluster to compare a given user's friends to each other by clustering its data into 4 other clusters. As a result, it formed an 8-dimensional feature set, which reports a cluster based on how close it is to the central user. Moreover, the clustering is set to a higher-dimensional space instead of reducing data to a lower-dimensional space and reporting positions based on how far from the origin [12]. A genetic-based weighted K-means clustering algorithm, in which the weighted mean is computed based on the cluster centroid. The genetic process introduces a new function for competent record selection based on data-sets. [14].

2.1.3. *Logistic Regression*

A logistic regression to examine if factors such as demographics and socioeconomic status can predict a user's behavior in a social network. The three indicators that are related to this behavior include demographic variables (x_0A), access variables (y_0B), and communication variables (z_0C). The final logistic regression model is $\ln(p) = \alpha + x_0A + (y_0B) + (z_0C) + \epsilon$. However, it is not easy to find the optional weight for negative samples [15]. A probabilistic relational model (PRM) for learning probabilistic models from relational data, which uses a relational database rather than user-item matrices. The input relational schema has three classes which include items, searching details, and the relation between them. The relational schema can be expanded by adding more entities related to main entities. It can predict the probability of the user visiting the property for the given search criteria [16].

Traditional friend recommendation algorithms are mostly based on common friends or similar interests. However, when the user's friend relationship or user's interest is sparse, the recommendation result is unsatisfactory. Social applications are usually classified into three categories: one is based on acquaintances, such as ICQ, and the second is based on strangers, such as MySpace. The third category includes both acquaintances and strangers, such as Twitter [6]. Recommendation algorithms have different emphases in different types of social applications [6].

User as the key object in social networks has many features, i.e., demographic features included in user profile. User's features are very significant, which could be leveraged to help recommend

friends to a target user. User's features are generally combined with other information like user's interests and common neighbors. Some authors took gender, hometown, religion, educational status etc. into account and learned the weights of features by Genetic Algorithm. The experiment conducted on a synthetic dataset showed the effectiveness of results in the case of considering both user's features and interactions. A FR system using the user's total features which is based on the Law of Total Probability, in which the probabilities of features are obtained by the statistical results according to the information of the user's friends and friends' friends. Compared to other methods including Common-Neighbors, Adamic/Adar and Jaccard coefficient, the authors found that the method based on user's features performs is better on total, especially when the number of user's friends is less than 100. The FR problem can be viewed as link prediction in social networks in essence. FR systems can be based on several basis like recommendation based on user features, network structure and trust in social networks.

The friend recommendation algorithm detects the implicit groups of friend candidates—namely, recommend a user's friends or friends to the user based on music interests and the number of interactions between users. The input needed for the algorithm is two users which are the source user which is the user that is focused on getting data from, and the target user which is what is tried to recommend the source's friends to. The algorithm will return a list of all the friend candidates in descending order of a score. This score indicates how compatible the candidate is. In other words, how good the candidate will be as a friend of the target user. The Interaction Score calculates the weight of edges by summing the products of importance factors and value of each attribute while the Interaction Rank takes time into consideration and evaluates the time values with an exponentially decaying scale [5]. The FriendRecommend function takes two inputs where the source is the source user who is the center of our partial graph and target is the target user to whom we are recommending friends to. The friends of the source user are attached to the source user in the input. The function returns a list of all the source's friends and their score received. The list is ordered in the descending order of the source's friends' score [5].

2.2. *Music Recommendation*

A multi-context aware user model and track recommender system that jointly exploit information about the current situation and musical preferences of users. system clusters users based on their situational context features and similarly, clusters music tracks based on their content features. Relying on Factorization Machines for the computation of recommendations, the proposed multi

context-aware user model successfully leverages interaction effects between user listening histories, situational, and track content information, substantially outperforming a set of baseline recommender systems. Its user model combines situational and acoustic context information, using Factorization Machines [1].

2.2.1. Convolutional Recurrent Neural Networks

Uses convolutional recurrent neural networks (CRNN) for feature extraction and similarity distance to look for similarities between features. Recommendations are based on perceptual resemblance of what was previously heard by the user. CRNNs that consider both the frequency features and time sequence patterns have overall better performance. It indicates the effectiveness of its hybrid structure to extract the music features [2]. Personalized music recommendation is a challenging task in the field of music information retrieval (MIR). A personalized music recommendation algorithm recommends new music to a user that is similar to the previously listened by the user. In order to recommend music to a user, the first step is to classify the music according to the user's history [2]. Traditional classifiers such as the support vector machine and linear regression classify the music by extracting the mel-frequency cepstral coefficients (MFCC) from the audio signal of the music. As the structural complexity of the music is more, the efficiency of traditional classifiers reduces in classifying the music from different genres.

In order to predict personalized sequential patterns, the long-term user preferences and the short-term variations in preference need to be jointly considered for accounting for both personalization and sequential transitions. The most widely-studied approach for modeling user preferences is the collaborative filtering (CF) method, and the content based methods [3]. Neighborhood methods recommend items that are appreciated by users who share the same preferences. Another category of CF models employ machine learning models, such as Bayesian models and matrix factorization, to perform the recommendation task.

2.2.2. Markov Chain Model

The content based methods recommend items that are similar to those that a user preferred in the past. The similarity of items is calculated using the features associated with the compared items. Markov chains have enjoyed a wide popularity in modeling dynamic stochastic transition, because of their powerful strength at uncovering sequential patterns [4]. First-order Markov

models do not perform well in predicting a user's next-step behavior since these models only consider the last item. High-order Markov models are introduced to capture the longer history behaviors of users, which are able to improve the prediction accuracy [4]. Despite the success achieved by Factorized Personalized Markov Chains (FPMC), the transition matrix of the Markov model is assumed to be the same over all users, which may not be true in certain real-world applications [4]. In order to classify the music, firstly we should represent the audio signals in the form of the MFCCs. These MFCCs denote the structural and hierarchical features of an audio signal. In order to obtain the MFCCs, we sampled the audio signal of the music into a set of small audio clips of 4 seconds [4]. Later we convert these audio signals into a log-compressed mel-spectrogram with 128 components to generate the MFCCs. Based on these MFCCs, CNN classifies the music. The confidence score (CS) value is calculated with respect to the fitness of music predictions for each user and each genre available in the testing data. The CNN approach classifies music based on the corresponding audio signals of the music. The CF algorithm uses the classified music data and the log file to provide music recommendations to users [4].

Chapter 3

System Analysis

3.1 Functional Requirements:

- The system will allow the user to listen to music.
- The system will recommend songs to the user based on past listenings.
- The system will allow the user to add people as friends.
- The user will be able to chat and share music with friends.
- The system will suggest friends of similar music taste.

3.2 Non-Functional Requirements:

These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to another. They are also called non-behavioral requirements.

- The processing of each request should be done within 10 seconds.
- The system should provide better accuracy.
- User friendly.

3.3 Specific Requirements

Mobile Phone

- The mobile phone should be an Android powered phone.
- The mobile phone should have at least Android API 21.
- The mobile phone should have an Internet Connection.

For Development

- Microsoft Windows 10 (32- or 64-bit).
- 4GB RAM minimum.

- 2GB of available disk space minimum.
- i5 Processor.

3.4 Use Case Diagram & Specifications

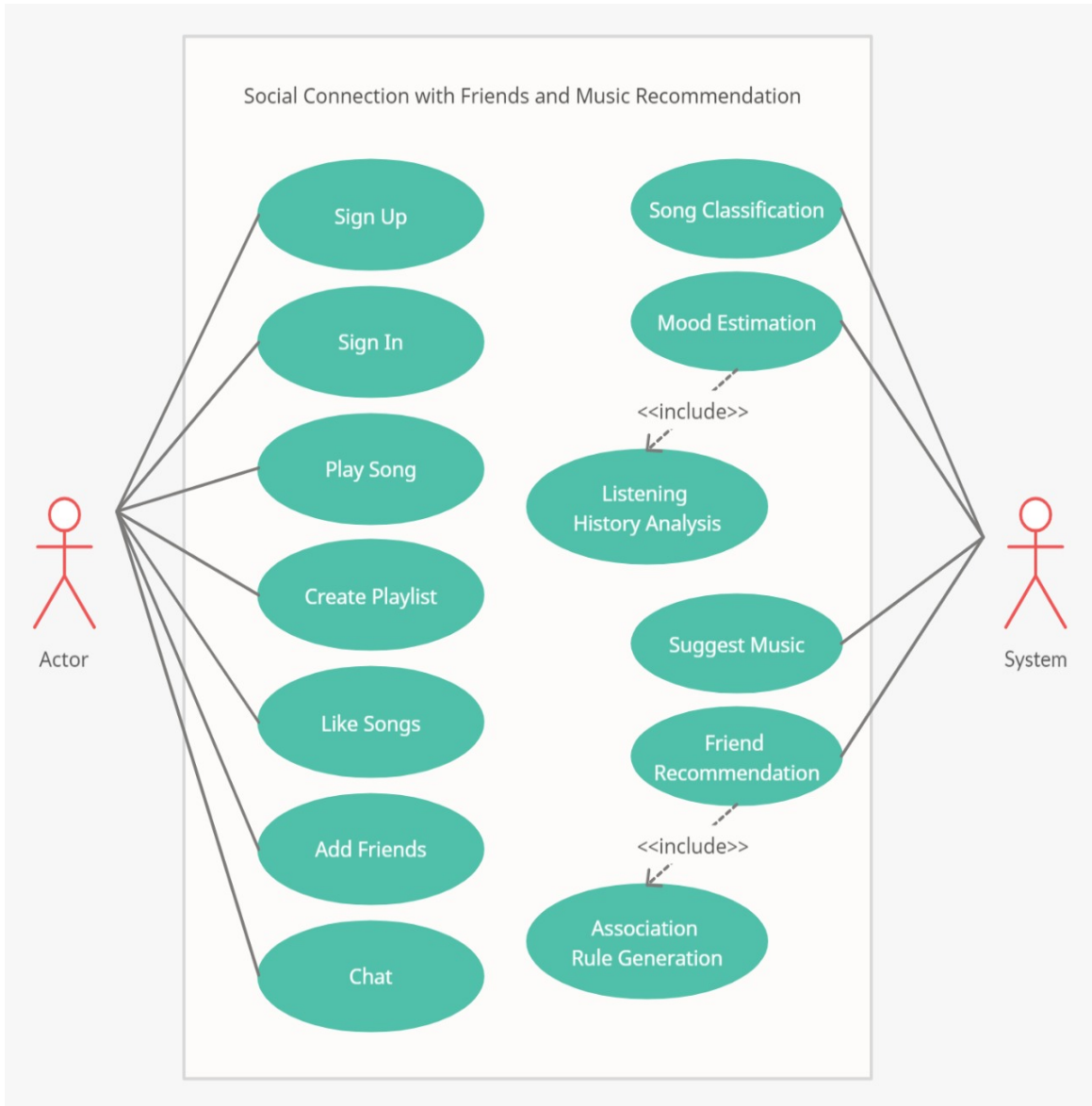


Fig. 3.4 Use Case Diagram

Use Case 1: Sign Up

Actor: User

Description: In this activity the user has to sign up to avail the functionality of our system.

Pre-Condition: The frontend has loaded.

Post-Condition: The user has registered.

Basic Course Of Action:

1. User registers
2. Enter all the details
3. Registered Successfully

Use Case 2: Sign In

Actor: User

Description: In this activity the user has to sign in to avail the functionality of our system.

Pre-Condition: The frontend has loaded.

Post-Condition: The user has logged in.

Basic Course Of Action:

1. User logs in
2. Enter all the details
3. Logged In

Use Case 3: Play Song

Actor: User

Description: In this activity the user can play songs from their playlist.

Pre-Condition: The user is logged in.

Post-Condition: Playing the song.

Basic Course Of Action:

1. Access music playlist

Use Case 4: Create Playlist

Actor: User

Description: In this activity the user can create a music playlist.

Pre-Condition: The user is logged in.

Post-Condition: Creating song playlist.

Basic Course Of Action:

1. Access songs
2. Create music playlist

Use Case 5: Add Friends

Actor: User

Description: In this activity the user can search and add friends.

Pre-Condition: The user is logged in.

Post-Condition: List of recommended users as friends.

Basic Course Of Action:

1. Access the list of recommended friends
2. Add users as friends

Use Case 6: Chat

Actor: User

Description: In this activity the user can have conversation with friends.

Pre-Condition: The user is logged in.

Post-Condition: Conversation with a selected friend.

Basic Course Of Action:

1. Access the list of added friends
2. Select a specific friend
3. Have a conversation with friend

Chapter 4

Analysis Modeling

4.1 Sequence Diagram

Fig. 4.1 represents the sequence diagram of Social Connection with Friends and Music Recommendation. The system fetches the user's music playlist using the Spotify API. Based on the user's listening history, playlist and music related posts like the system suggests friends to the user and also based on listening history recommends songs. Then all the calculated data is then passed to the cloud firestore for storage. The user then can hang out on social media and access various features. The system requires registration to use the platform.

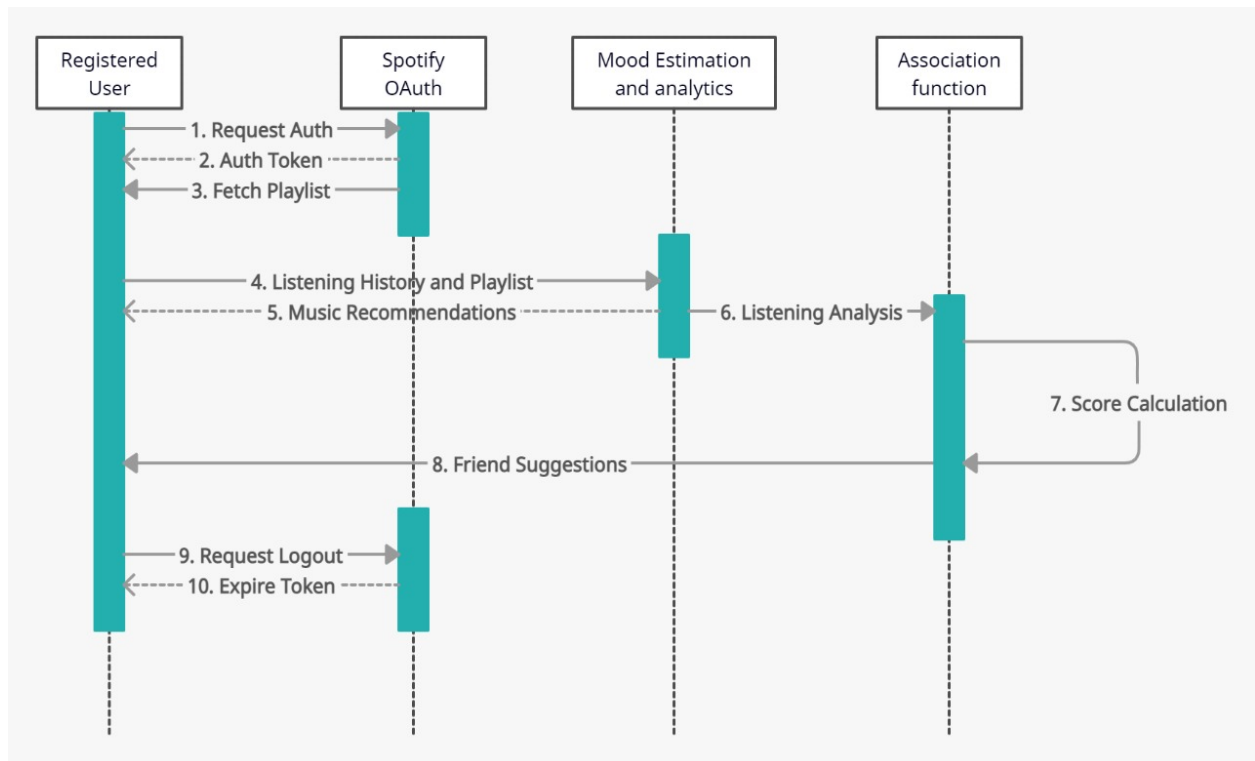
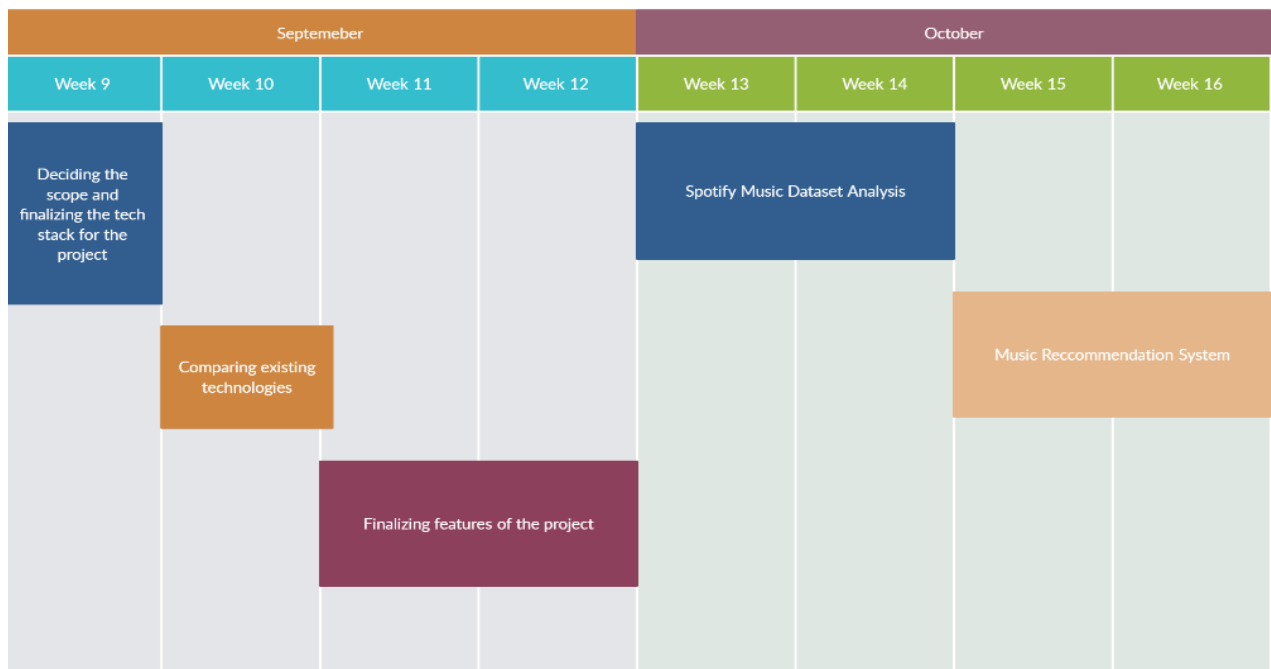
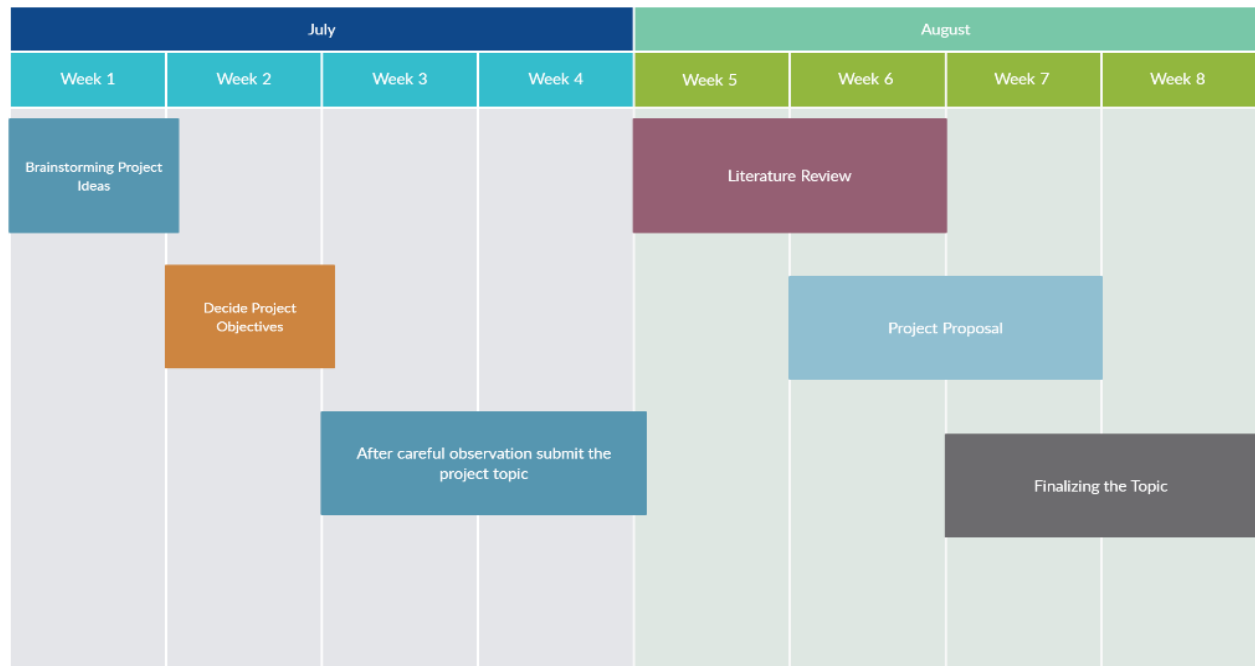
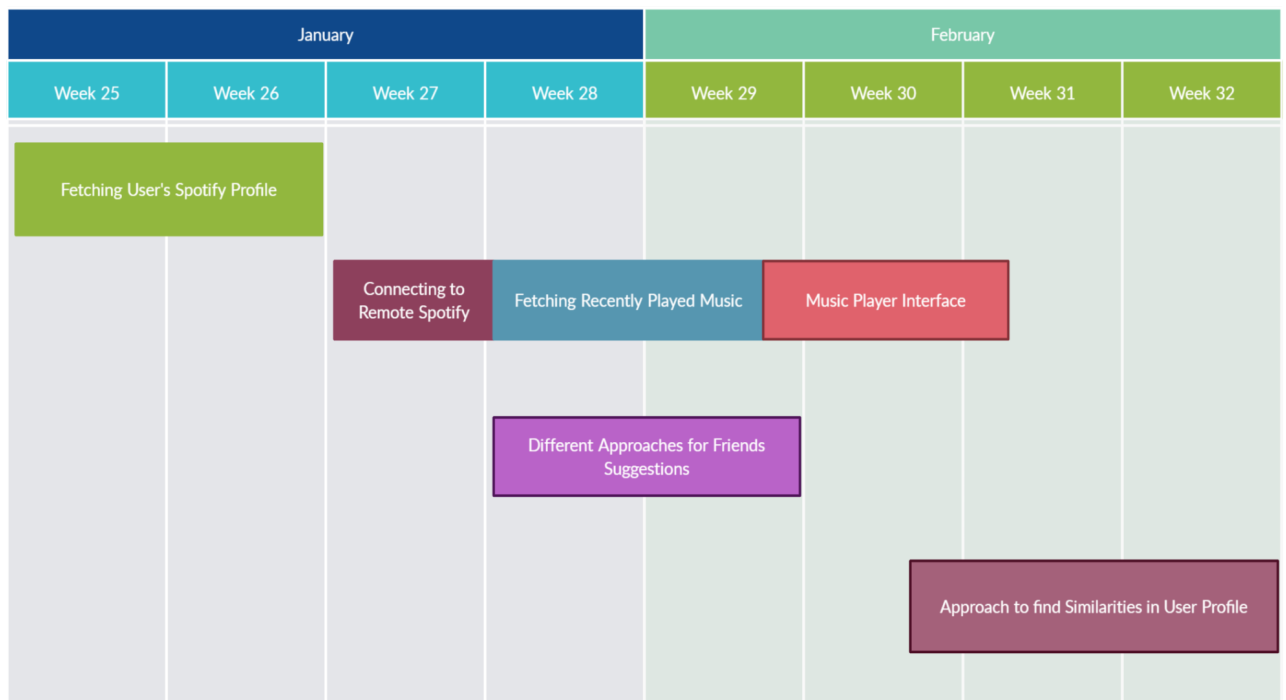
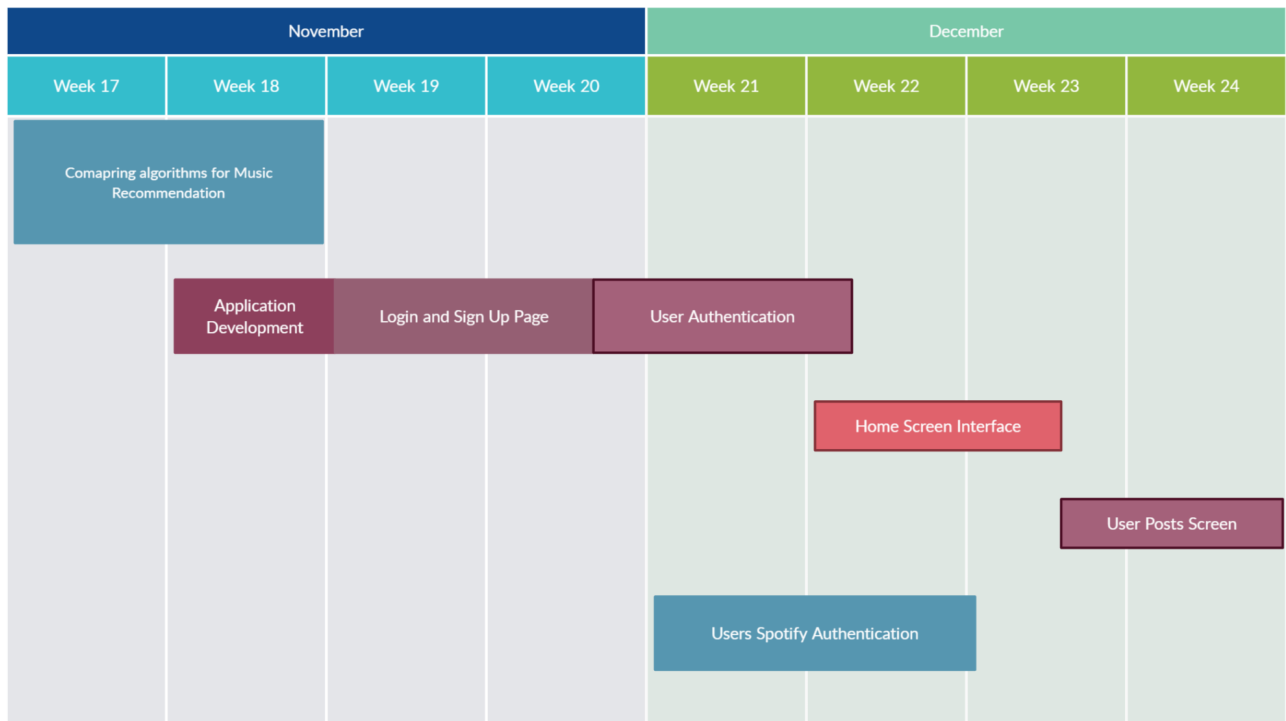


Fig. 4.1 Sequence Diagram

4.2 Timeline Diagram





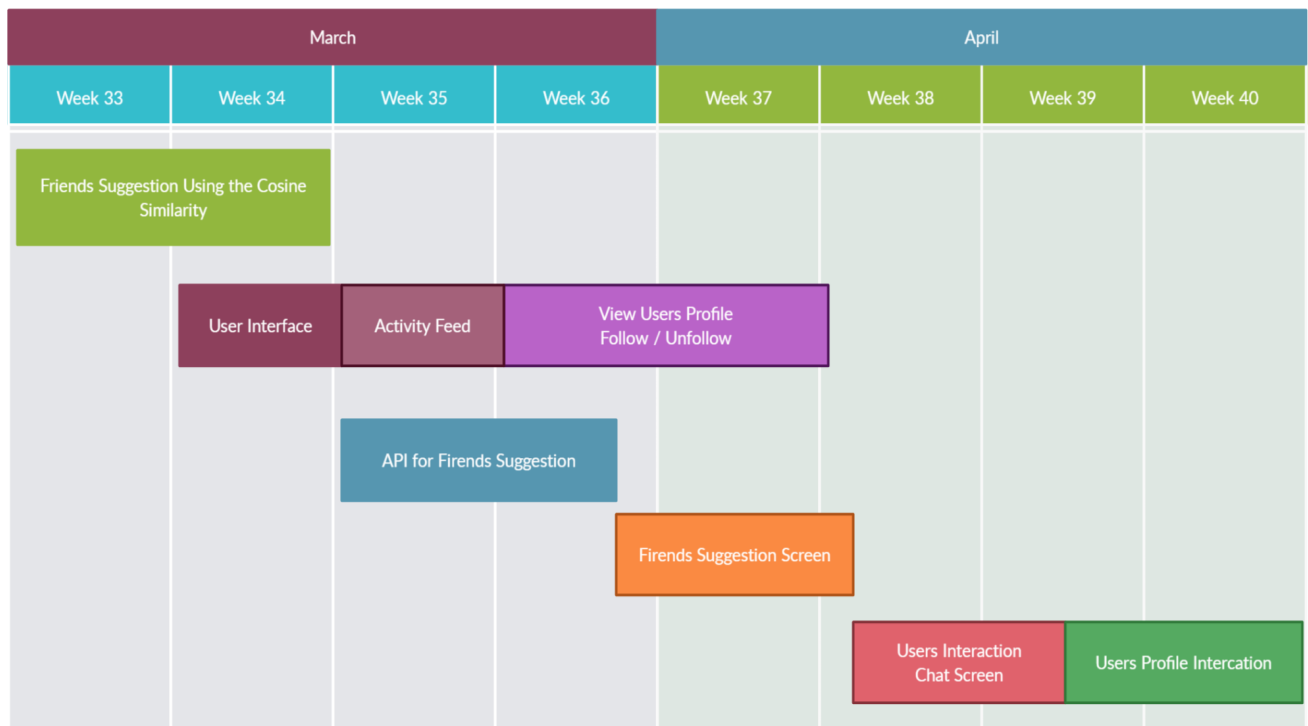


Fig. 4.2 Timeline Chart

Chapter 5

Design

5.1 Architectural Design

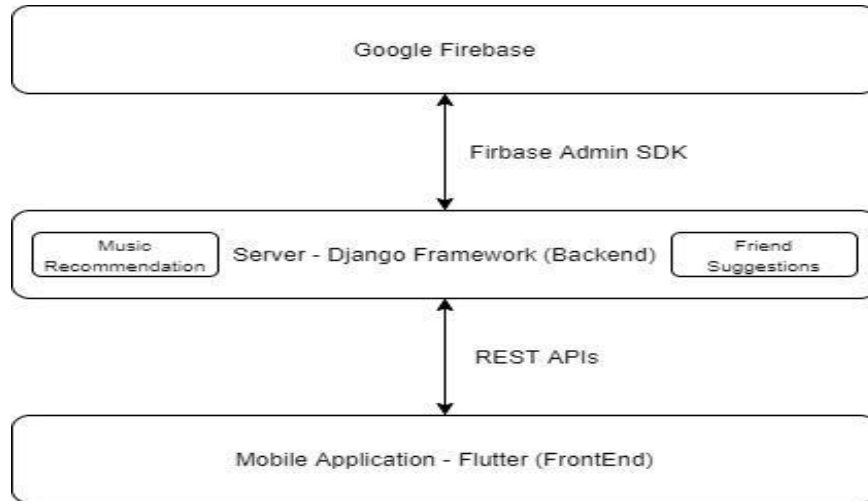


Fig. 5.1 System Flow Diagram

Fig. 5.1 depicts the basic system flow for our application where the frontend of the application is developed using Google's framework called Flutter. Google Cloud Firestore is used as a database for real time data access. And using the Django Framework to develop the backend of the system.

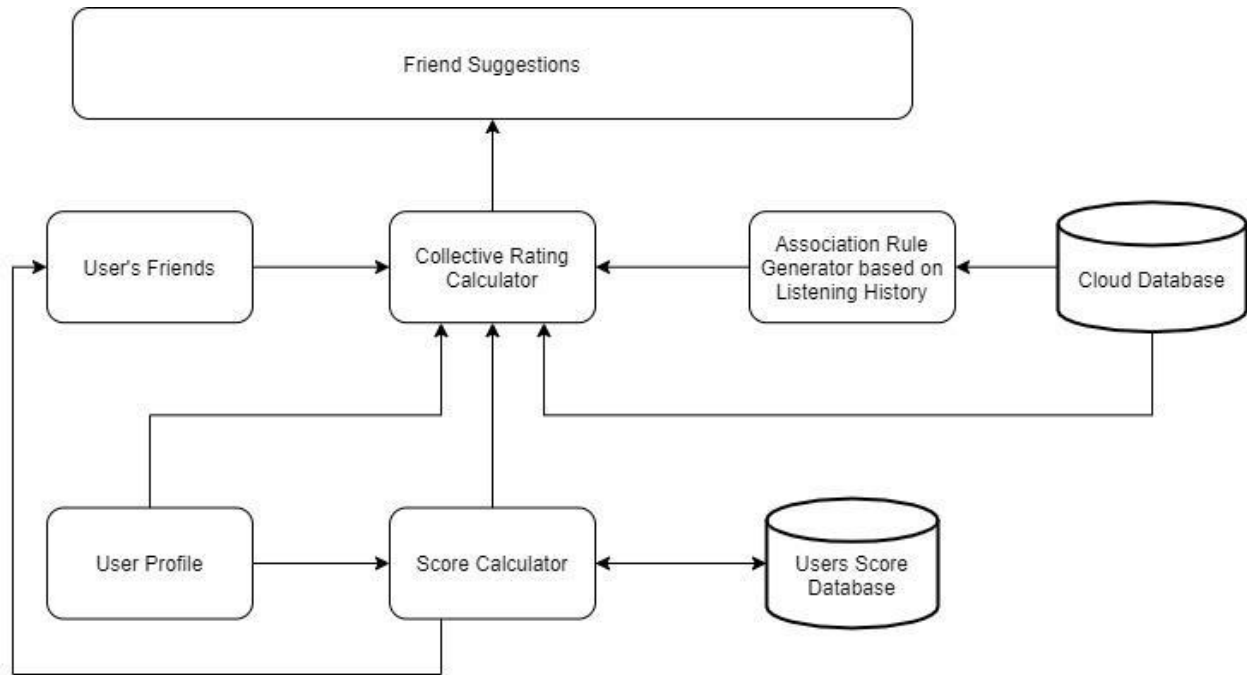


Fig. 5.2 Friend Suggestion System Block Diagram

Fig. 5.2 is a system block diagram for friend suggestion. Recommend a user's friends or friends to the user based on music interests and the number of interactions between users. The input needed for the algorithm is two users. The source user which is the user that we are focusing on getting data from. The target user, which is what we are trying to recommend to the source's friends too. The algorithm will return a list of all the friend candidates in descending order of a score. This score indicates how compatible the candidate is. Interaction score will be calculated where the important factors are music playlist features, music-related posts liked & location of the user. A social graph with weighted edges, which express the relationship strength between a user and his implicit groups. The core criteria are to compute the edge weights.

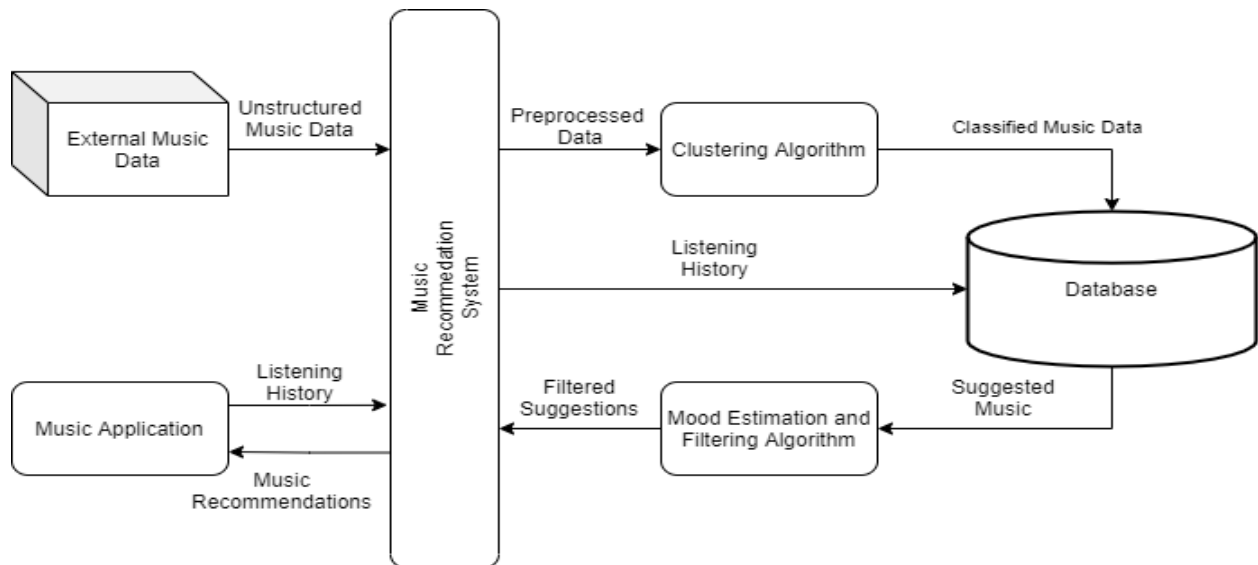


Fig. 5.3 Music Recommendation System Block Diagram

Fig. 5.3 represents a music recommendation system block diagram. The recommendation system takes the preprocessed data and the user's song listening history and based on the features of the listening history the system estimates the data point and recommends songs which are nearby to the song listening history data point.

5.2 Database Design

Google Cloud

Google Cloud consists of a set of physical assets, such as computers and hard disk drives, and virtual resources, such as virtual machines (VMs), that are contained in Google's data centers around the globe. Each data center location is in a region. Regions are available in Asia, Australia, Europe, North America, and South America. Each region is a collection of zones, which are isolated from each other within the region. Each zone is identified by a name that combines a letter identifier with the name of the region. For example, zone a in the East Asia region is named asia-east1-a.

This distribution of resources provides several benefits, including redundancy in case of failure and reduced latency by locating resources closer to clients. This distribution also introduces some rules about how resources can be used together.

Google Firebase

Google Firebase is a Google-backed application development software that enables developers to develop iOS, Android and Web apps. Firebase provides tools for tracking analytics, reporting and fixing app crashes, creating marketing and product experiment.

For real-time access to users' data, Google Firebase is used. The users are authenticated from the Google Firebase Authentication which has several user authentication options like email & password, phone number, etc. Google Firebase Authentication helps users to register and authenticate themselves in the application.

Most apps need to know the identity of a user. Knowing a user's identity allows an app to securely save user data in the cloud and provide the same personalized experience across all of the user's devices.

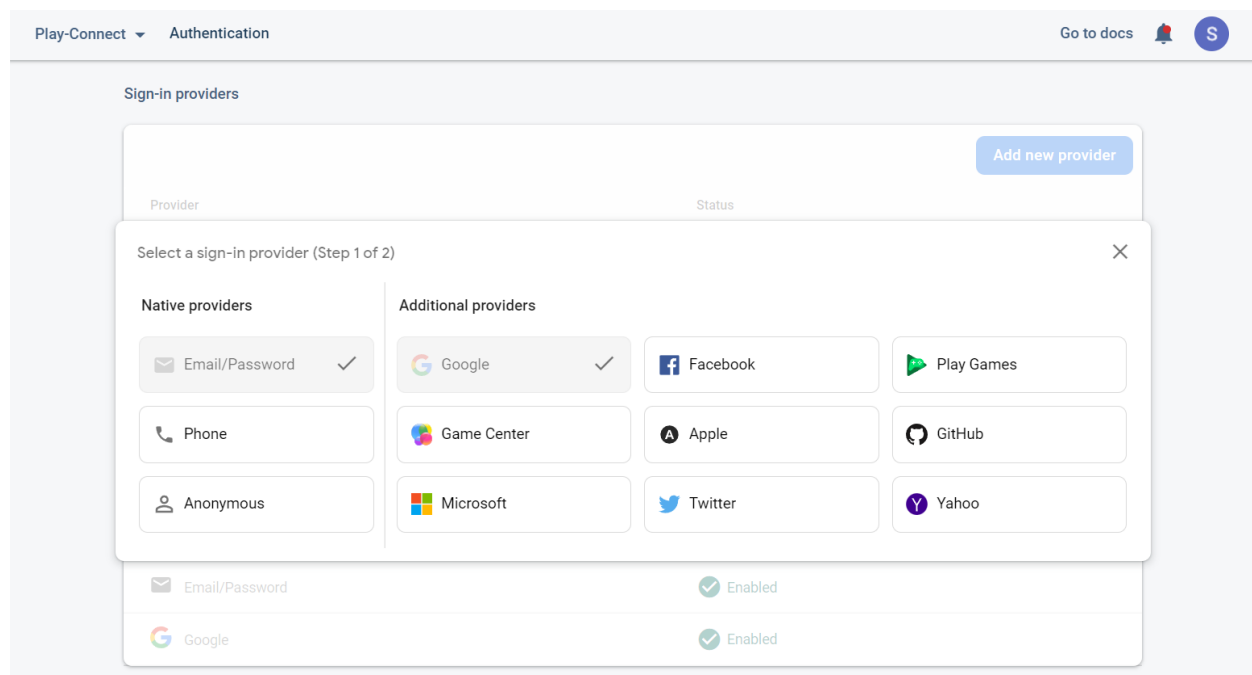
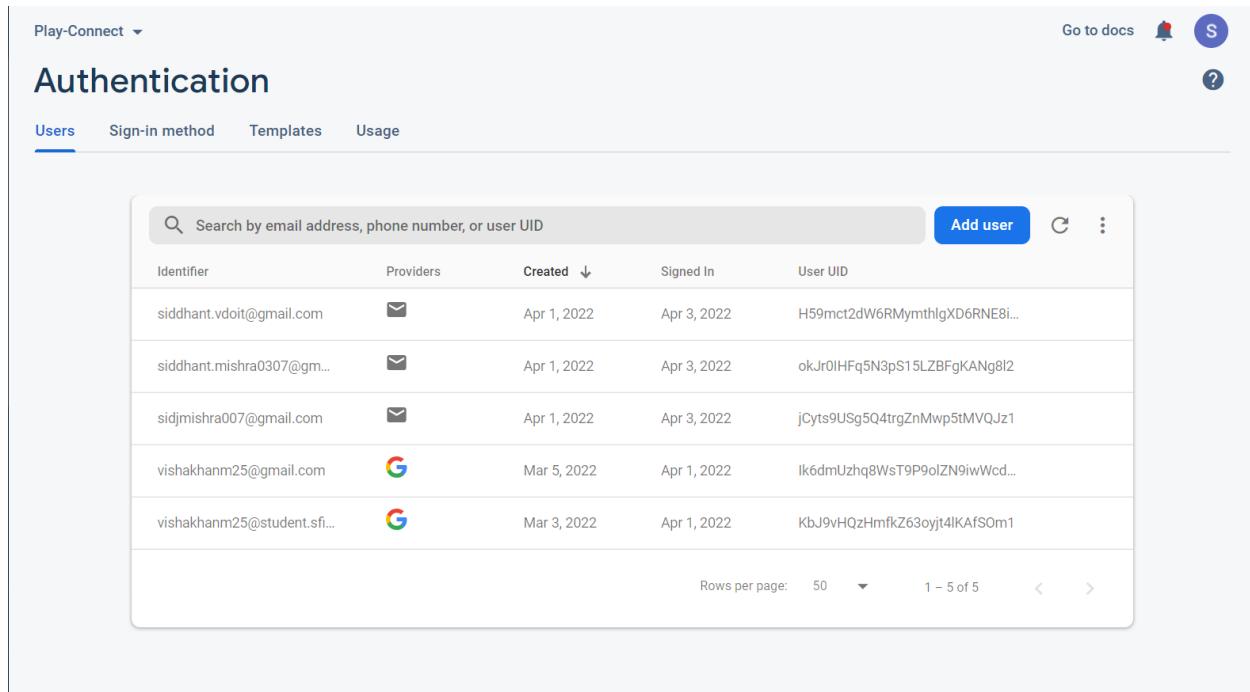


Fig. 5.4 Authentication Methods

Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to your app. It supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and more (fig 5.4).

As shown in fig 5.5, once the user is registered a list of registered users is displayed in the dashboard of Google Firebase, which helps us to easily identify the registered one while authenticating on the application.



Identifier	Providers	Created ↓	Signed In	User UID
siddhant.vdoit@gmail.com	📧	Apr 1, 2022	Apr 3, 2022	H59mct2dW6RMymthlgXD6RNE8i...
siddhant.mishra0307@gm...	📧	Apr 1, 2022	Apr 3, 2022	okJr0IHfQ5N3pS15LZBFgKANG8l2
sidjmishra007@gmail.com	📧	Apr 1, 2022	Apr 3, 2022	jCyts9USg5Q4trgZnMwp5tMVQJz1
vishakhanm25@gmail.com	🌐	Mar 5, 2022	Apr 1, 2022	lk6dmUzhq8WsT9P9olZn9iwWcd...
vishakhanm25@student.sfi...	🌐	Mar 3, 2022	Apr 1, 2022	KbJ9vHQzHmfkZ63oyjt4lKAfSOM1

Fig. 5.5 Registered Users On Google Firebase Auth

Cloud Firestore is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud. Like Firebase Realtime Database, it keeps your data in sync across client apps through realtime listeners and offers offline support for mobile and web so you can build responsive apps that work regardless of network latency or Internet connectivity. Cloud Firestore also offers seamless integration with other Firebase and Google Cloud products, including Cloud Functions.

Cloud Firestore promotes the user data storage in key-value pair manner. Fig 5.6 shows how the user data is stored in Firestore and in order to access the store data the associated keys are used.

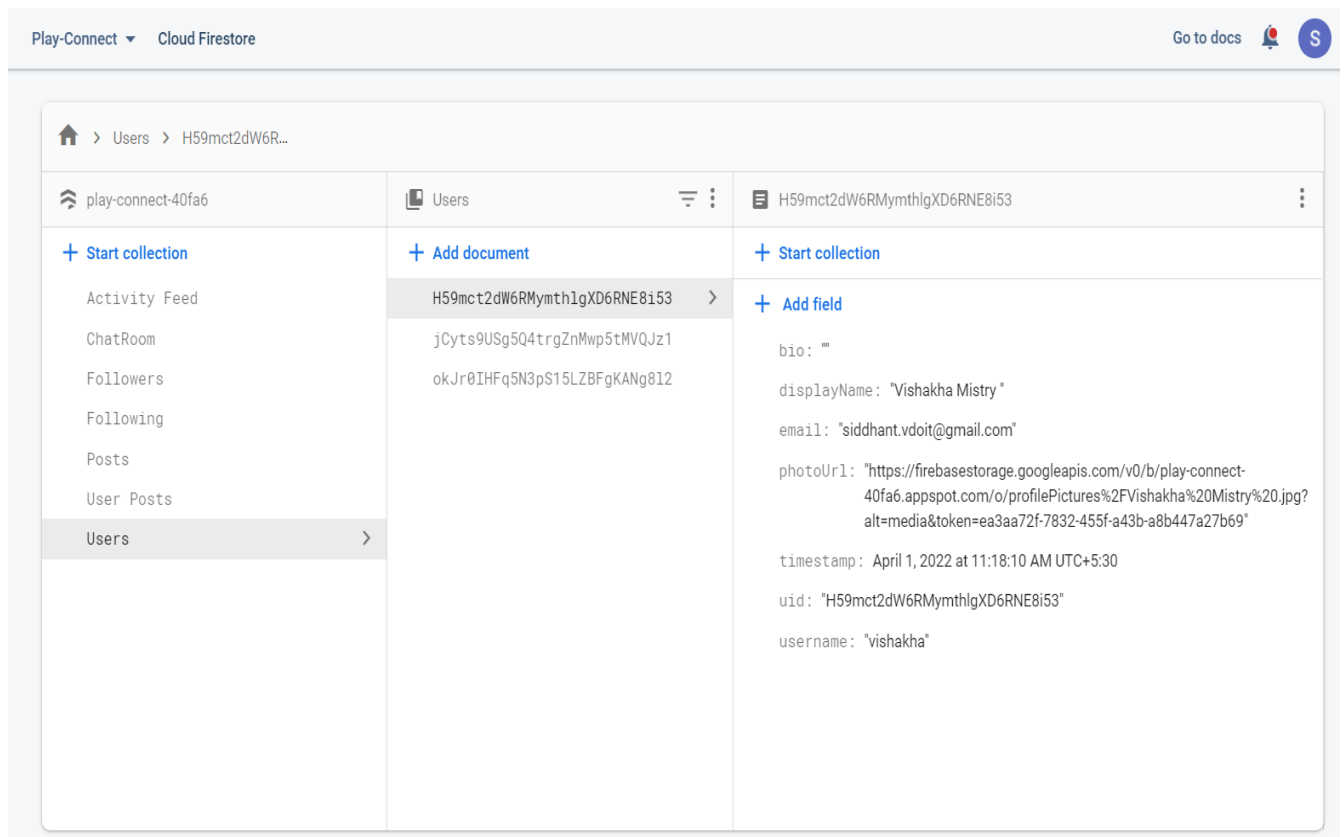


Fig. 5.6 Cloud Firestore Database

Cloud Storage for Firebase is built for app developers who need to store and serve user-generated content, such as photos or videos. Cloud Storage for Firebase is a powerful, simple, and cost-effective object storage service built for Google scale. The Firebase SDKs for Cloud

Storage add Google security to file uploads and downloads for your Firebase apps, regardless of network quality.

Fig 5.7, shows how a media file is stored in Firebase Storage. Once the media is uploaded into the storage it returns a download url which helps us to access the media stored. The download url is stored with the associated post data in Firestore which is then used to display contents in the application.

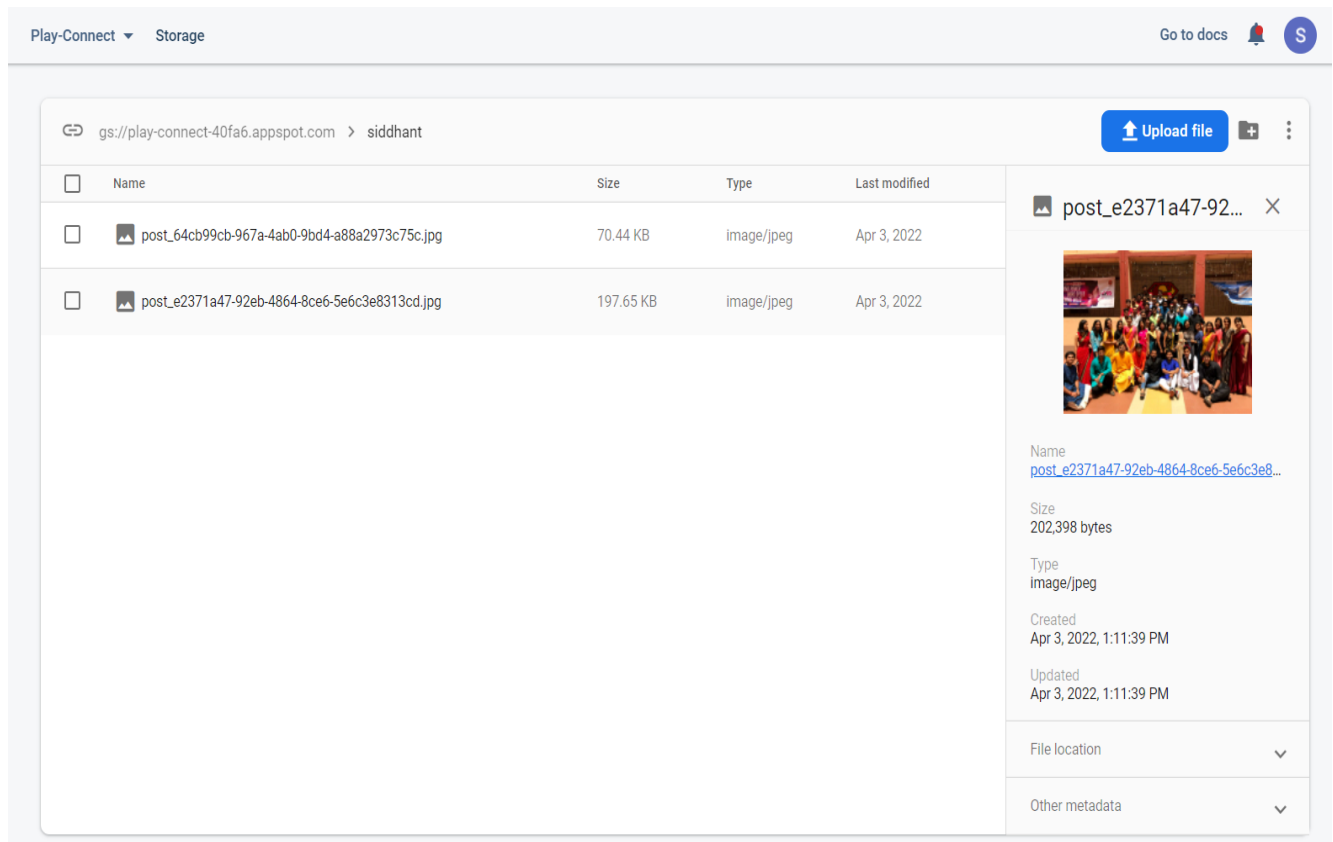


Fig. 5.7 Firesbase Storage

Chapter 6

Implementation

6.1 Dataset

The Spotify Song Dataset is used to train the model to classify the songs in the dataset into 20 clusters and the genres in the dataset into 10 clusters.

6.2 Spotify Music Trends Analysis

We require a dataset holding (i) listening histories of users, and (iii) acoustic characteristics of the songs. Hence, we propose to use the Spotify API. Spotify has a web API that can be used to retrieve audio features and metadata about songs such as the song's popularity, tempo, loudness, key, and the year in which it was released [17]. These high-level features are well established in the MIR community and are widely used as a compact form for describing songs for modeling audio characteristics of tracks in the field of music information retrieval. Using this data to build a recommendation system, it will recommend friends and songs.

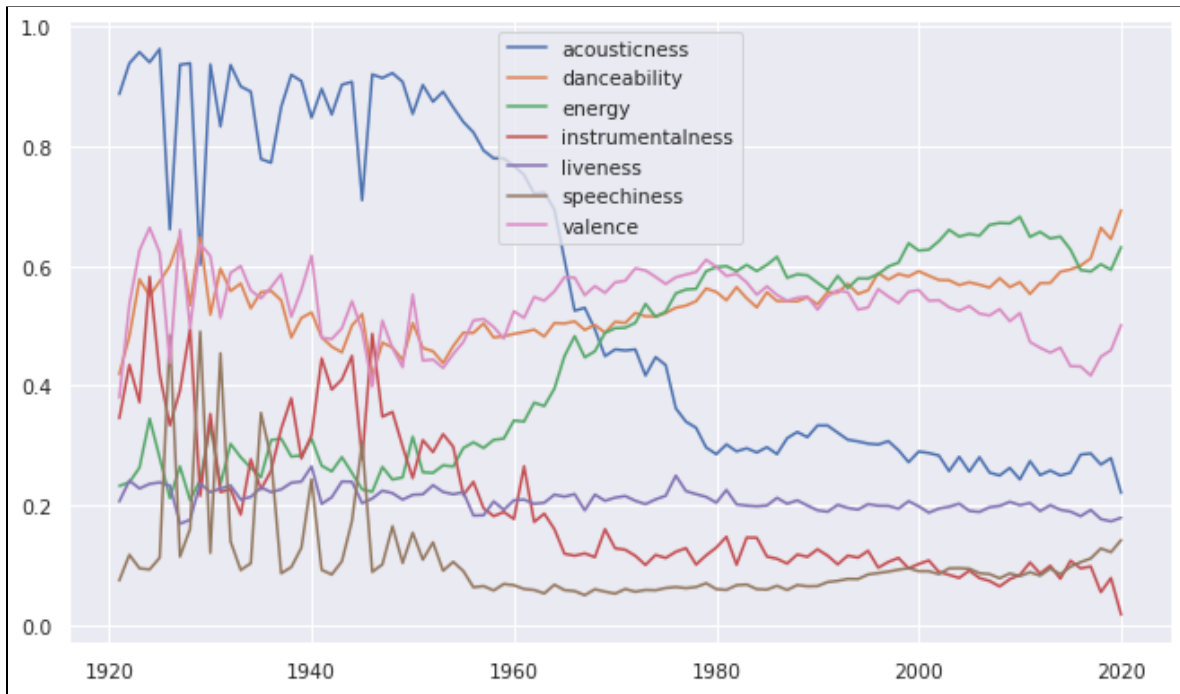


Fig. 6.1 Music Trend over the years

Using the data grouped by year, we can understand how the overall sound of music has changed from 1921 to 2020. We can see that music has transitioned from the more acoustic and instrumental sound. The music of the 2000s sounds very different due to the advent of computers and advanced audio engineering technology (Fig. 6.1).

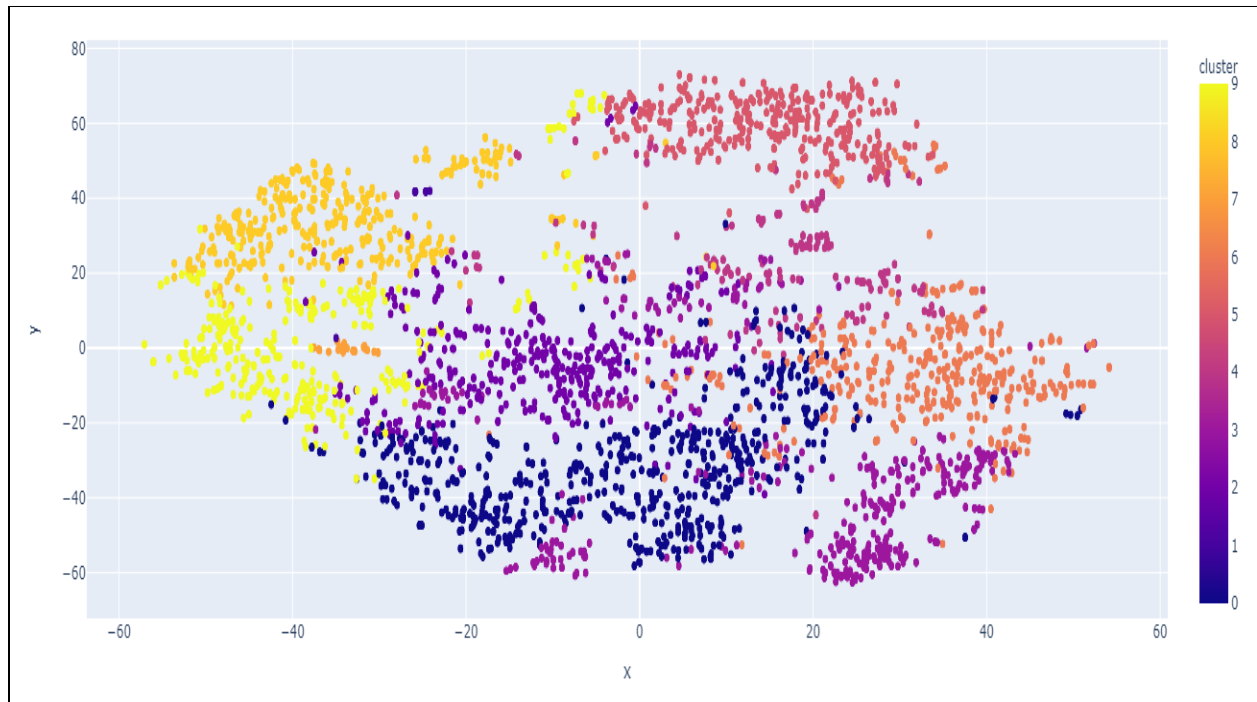


Fig. 6.2 Clustering of Top 10 Music Genres

Clustering algorithm to divide the over 2,900 genres in this dataset into ten clusters based on the numerical audio features of each genre. Similar genres tend to have data points that are located close to each other. Similar genres will sound similar and will come from similar time periods. Built a recommendation system by taking the data points of the songs a user has listened to and recommending songs corresponding to nearby data points (Fig. 6.2 & 6.3).

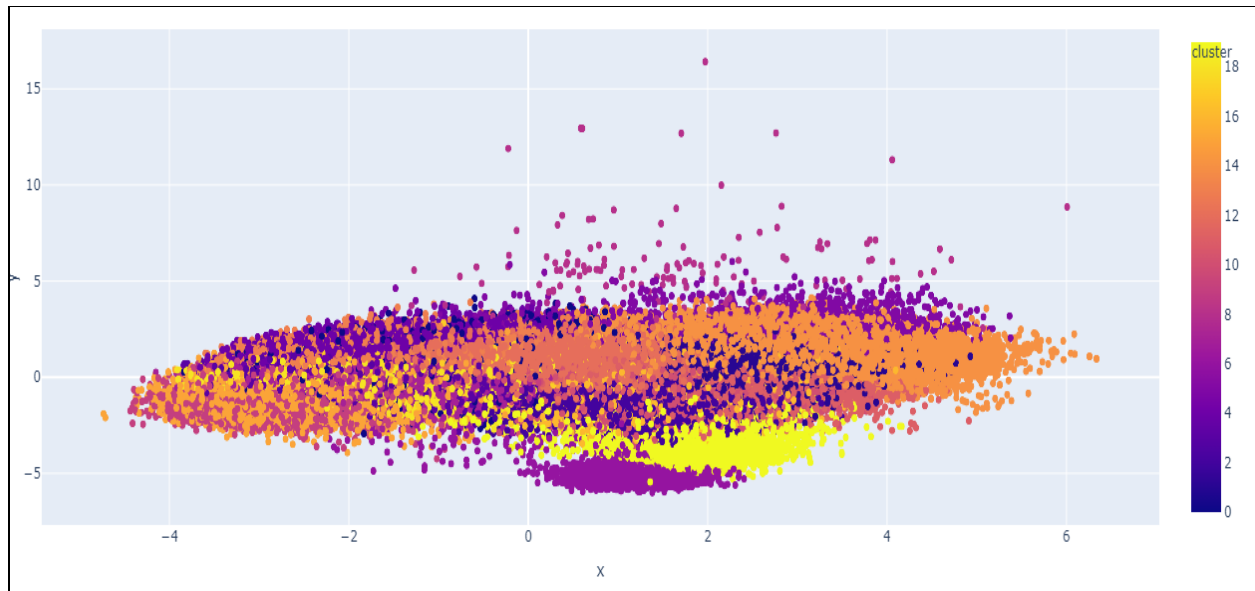


Fig. 6.3 Song Clusters Based on Song Features

```
[{'artists': "['Machine Gun Kelly']",
  'name': 'bloody valentine',
  'year': 2020},
 {'artists': "['girl in red']", 'name': 'bad idea!', 'year': 2019},
 {'artists': "['The Chainsmokers', '5 Seconds of Summer']",
  'name': 'Who Do You Love',
  'year': 2019},
 {'artists': "['Ariana Grande']", 'name': 'God is a woman', 'year': 2018},
 {'artists': "['Marshmello', 'Khalid']", 'name': 'Silence', 'year': 2017},
 {'artists': "['JVLA']",
  'name': 'Such a Whore (Stellar Remix)',
  'year': 2020},
 {'artists': "['Sam Smith', 'Demi Lovato']",
  'name': 'I'm Ready (with Demi Lovato)',
  'year': 2020},
 {'artists': "['Juice WRLD']", 'name': 'Legends', 'year': 2018},
 {'artists': "['Juice WRLD', 'Marshmello']",
  'name': 'Come & Go (with Marshmello)',
  'year': 2020},
 {'artists': "['Ariana Grande']", 'name': 'obvious', 'year': 2020}]
```

Fig. 6.4 Content Based Music Recommendation

Fig. 6.4 represents music recommendation based on contents. The music recommendation system takes the input as user listening history and using the Spotify API system extracts different song features and, based on the similar data points, recommends the songs which are nearby the song data point.

6.3 Friends Suggestions

In this section, we present friend recommendations using the user's location, musical profile, top artists and their genres. The rapid expansion of user data and geographic location data in the location-based social networking applications, it is becoming increasingly difficult for users to quickly find the information they need. For suggestions of friends, a score is generated for each pair of users based on the similarities between the top artists that they listen to along with their genres. The user's geographic location is then used to stream-down the search for suggesting friends. Once a new user has friends, similarity between their friends and the user is also considered and the friends with more similarity are suggested.

Inputs: source user , target user

Outputs: list of friend suggestions

Step 1: Get the base Score between source and target

Step 2: for each friend of source repeat Step 3 and Step 4

Step 3: if base Score == 0 then add friend_id and friend_sc = 0 in result

Step 4: if base Score != 0 then

Calculate score between target and friend

add friend_id and friend_sc = (friend_sc - baseScore) in result

Step 5: Return result

Calculating similarities between numerical vectors is not difficult, but when it comes to strings, the trick is to convert strings to numerical vectors first, discard everything irrelevant. Some of the methods to find similarity between strings include

1. Euclidean Distance
2. Cosine similarity

Euclidean distance between two points in the Euclidean space is defined as the length of the line segment between two points whereas Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. It is thus a judgment of orientation and not magnitude.

$$d(x, y) = \sqrt{\sum_{i=1}^n (y_i - x_i)^2}$$

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

So, if two genres “pop” and “dance pop” are considered which are musically similar, the euclidean distance between them is more as compared to cosine distance. Hence, we take cosine similarity between the users and suggest them friends.

```
def similarity(genres1, genres2):
    g1=[]
    g2=[]
    genres = []
    for genre in genres1:
        g1.append(genre.replace(" ", ""))

    for genre in genres2:
        g2.append(genre.replace(" ", ""))

    genres.append(' '.join(g1))
    genres.append(' '.join(g2))
    # print(genres)
    vectorizer = CountVectorizer().fit_transform(genres)
    vectors = vectorizer.toarray()
    csim = cosine_similarity(vectors)
    return csim[0][1]
```

```
genres1 = ["dance pop",
            "dutch edm",
            "edm",
            "electro house",
            "pop",
            "pop dance",
            "progressive house",
            "tropical house"]
genres2 = [
            "edm",
            "electro house",
            "pop",
            "tropical house"];

print(similarity(genres1, genres2))

0.7071067811865475
```

Fig. 6.5 Genre Similarity Score

Fig. 6.7 shows the similarity between the genres from two users. The similarity score would be more for more similar genres. Similarly, the similarity score for artists are calculated. The scores are averaged to get one single score for each pair of users and friends are suggested to source user having a score closer to his own score, i.e. difference between their similarities closer to 0.

6.4 Music Recommendation

Free Music Archive (FMA), an open and easily accessible dataset suitable for evaluating several tasks in MIR, a field concerned with browsing, searching, and organizing large music collections. The FMA provides 917 GiB and 343 days of Creative Commons-licensed audio from 106,574 tracks from 16,341 artists and 14,854 albums, arranged in a hierarchical taxonomy of 161 genres. It provides full-length and high-quality audio, pre-computed features, together with track- and user-level metadata, tags, and free-form text such as biographies. For our project we used the fma_small dataset containing 8,000 tracks of 30s, 8 balanced genres (7.2 GiB).

6.4.1 Preprocessing and Feature Extraction:

The first step is to create the audio features which will be used for training our model. We consider the following five features in our case:

- MFCC (Mel Frequency Cepstral Coefficients): MFCC features represent phonemes (distinct units of sound) as the shape of the vocal tract (which is responsible for sound generation) is manifest in them.

$$\text{Mel}(f) = 2595 \log \left(1 + \frac{f}{700} \right)$$

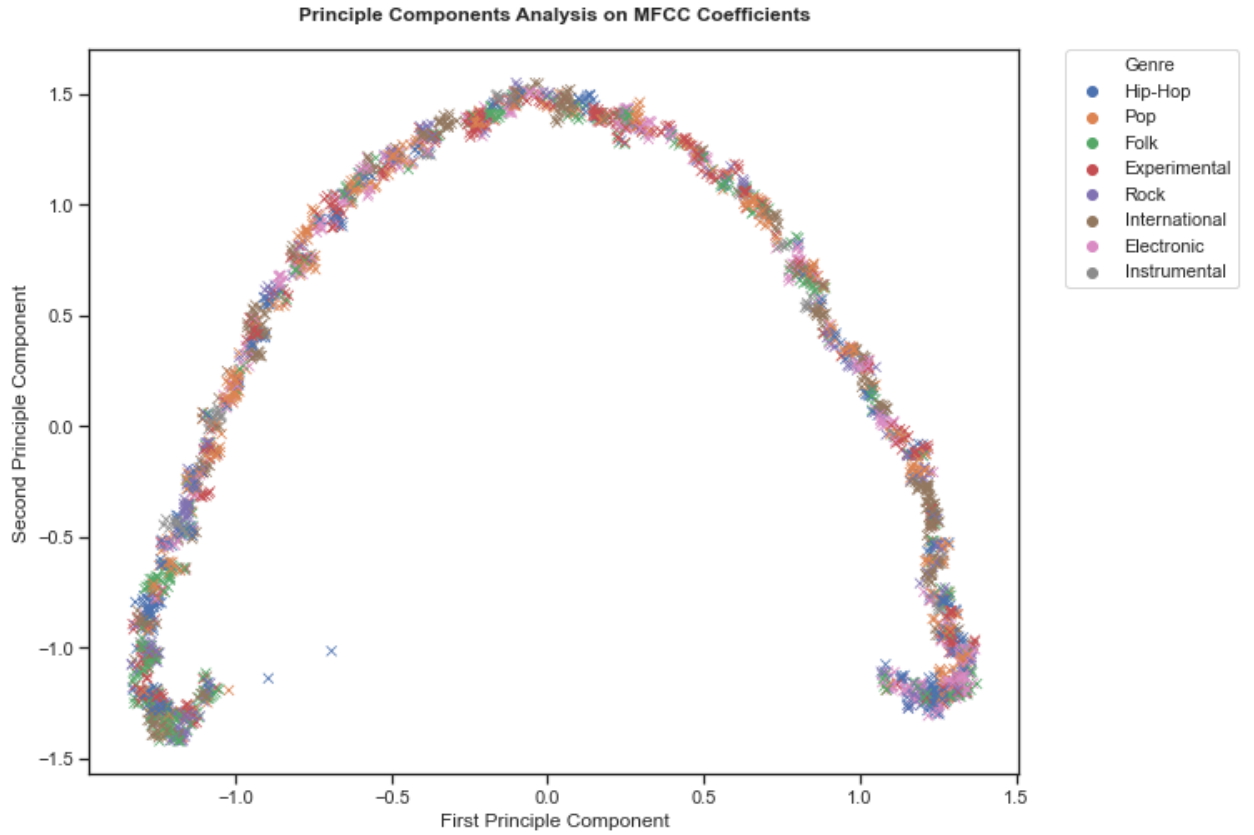


Fig. 6.6 Principle Components Analysis on MFCC Coefficients

- ZCR (Zero Crossing Rate): The zero-crossing rate (ZCR) is the rate at which a signal changes from positive to zero to negative or from negative to zero to positive. Its value has been widely used in both speech recognition and music information retrieval, being a key feature to classify percussive sounds.

$$Z(i) = \frac{1}{2W_L} \sum_{n=1}^{W_L} | \text{sgn}[x_i(n)] - \text{sgn}[x_i(n-1)] |,$$

where $\text{sgn}(\cdot)$ is the sign function, i.e.

$$\text{sgn}[x_i(n)] = \begin{cases} 1, & x_i(n) \geq 0, \\ -1, & x_i(n) < 0. \end{cases}$$

- Chroma STFT (Short Term Fourier Transform): The Chroma value of an audio basically represents the intensity of the twelve distinctive pitch classes that are used to study music. They can be employed in the differentiation of the pitch class profiles between audio signals.

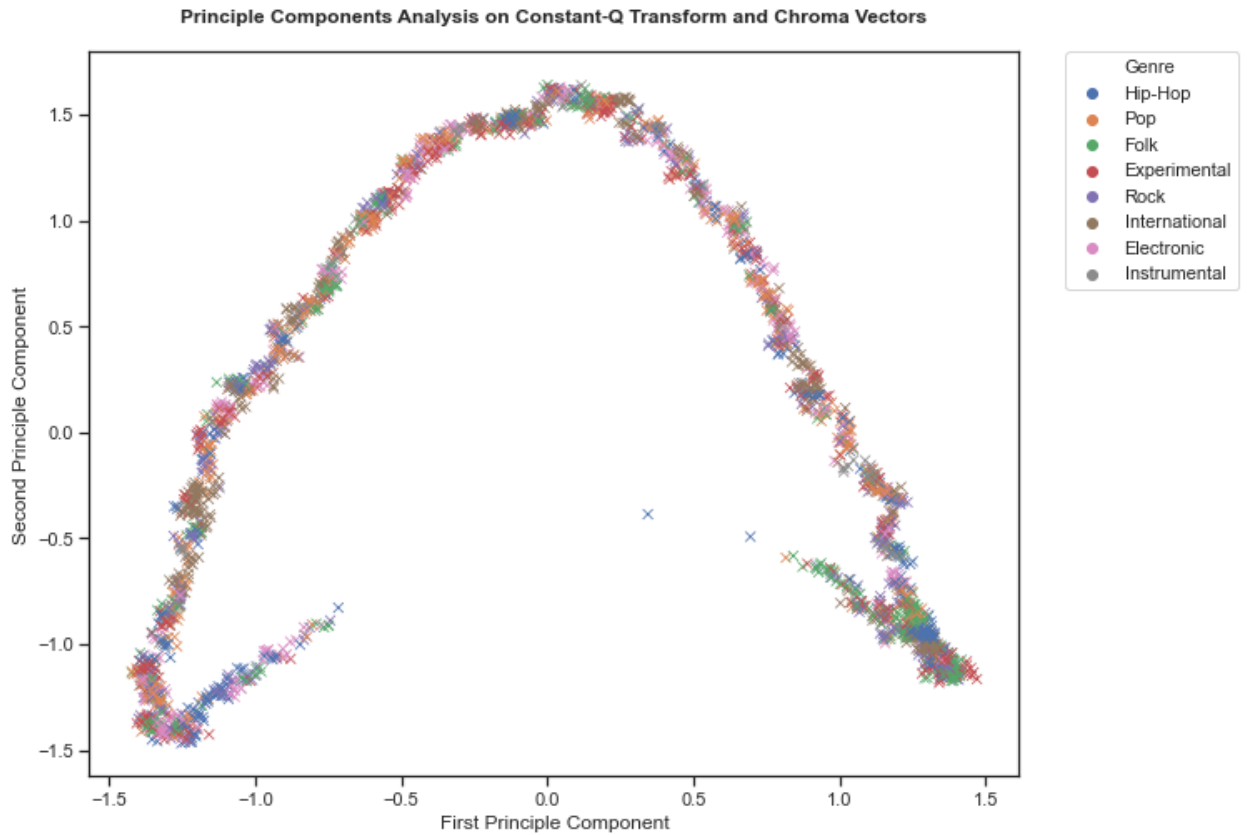


Fig. 6.7 Principle Components Analysis on Chroma Vectors

- Spectral Centroid: The spectral centroid is a measure used in digital signal processing to characterize a spectrum. It indicates where the center of mass of the spectrum is located. Perceptually, it has a robust connection with the impression of brightness of a sound. It is calculated as the weighted mean of the frequencies present in the signal, determined using a Fourier transform, with their magnitudes as the weights:

$$\text{Centroid} = \frac{\sum_{n=0}^{N-1} f(n)x(n)}{\sum_{n=0}^{N-1} x(n)}$$

where $x(n)$ represents the weighted frequency value, or magnitude, of bin number n , and $f(n)$ represents the center frequency of that bin.

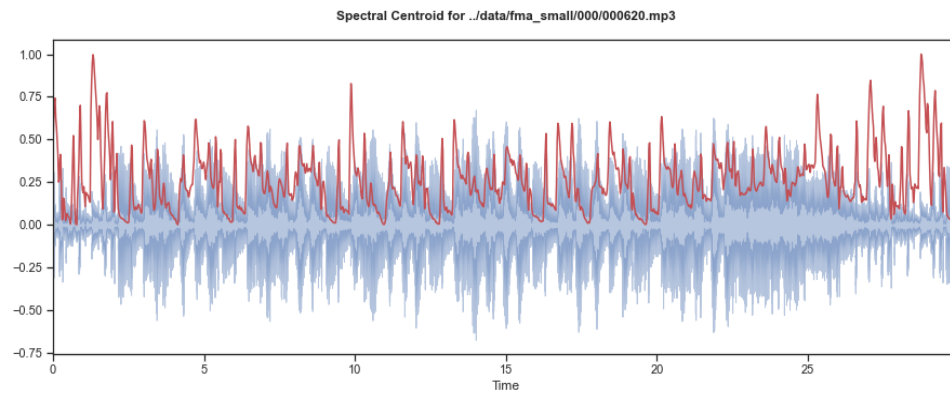


Fig. 6.8 Spectral Centroid for a sample audio file

- **Spectral Rolloff:** Spectral rolloff is the frequency below which a specified percentage of the total spectral energy, e.g. 85%, lies. It can be defined as the action of a specific type of filter which is designed to roll off the frequencies outside to a specific range. The reason we call it roll-off is because it is a gradual procedure. There can be two kinds of filters: hi-pass and low pass and both can roll off the frequency from a signal going outside of their range.

6.4.2 Classification Algorithms:

Logistic regression:

Logistic regression is basically a supervised classification algorithm. In a classification problem, the target variable (or output), y , can take only discrete values for a given set of features (or inputs), X . Just like Linear regression assumes that the data follows a linear function, Logistic regression models the data using the sigmoid function.

In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1). The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc. Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets. Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification.

$$S(x) = \frac{1}{1 + e^{-x}}$$

Gaussian Naive Bayesian:

When working with continuous data, an assumption often taken is that the continuous values associated with each class are distributed according to a normal (or Gaussian) distribution. The likelihood of the features is assumed to be-

$$P(x_i | y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Sometimes assume variance:

- is independent of Y (i.e., σ_i),
- or independent of X_i (i.e., σ_k)
- or both (i.e., σ)

Gaussian Naive Bayes supports continuous valued features and models each as conforming to a Gaussian (normal) distribution.

An approach to create a simple model is to assume that the data is described by a Gaussian distribution with no co-variance (independent dimensions) between dimensions. This model can be fit by simply finding the mean and standard deviation of the points within each label, which is all that is needed to define such a distribution.

6.4.3 Code:

```
import pickle
import numpy as np
import csv
from sklearn.preprocessing import LabelEncoder
from collections import defaultdict

from sklearn.naive_bayes import GaussianNB
```

```

from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import StandardScaler
# change the file locatio here
train_path = "../data/pickle/train/"
test_path = "../data/pickle/test/"
#load train data
# need reshap
chroma = pickle.load( open( train_path + "chroma_stft.p", "rb" ) )
mfcc = pickle.load( open( train_path + "mfcc.p", "rb" ) )
#not need
spectral_centroid = pickle.load( open( train_path + "spectral_centroid.p", "rb" ) )
spectral_rolloff = pickle.load( open( train_path + "spectral_rolloff.p", "rb" ) )
zcr = pickle.load( open( train_path + "zcr.p", "rb" ) )

#load test data
# need reshap
chroma_test = pickle.load( open( test_path + "chroma_stft_test.p", "rb" ) )
mfcc_test = pickle.load( open( test_path + "mfcc_test.p", "rb" ) )
#not need
spectral_centroid_test = pickle.load( open( test_path + "spectral_centroid_test.p", "rb" ) )
spectral_rolloff_test = pickle.load( open( test_path + "spectral_rolloff_test.p", "rb" ) )
zcr_test = pickle.load( open( test_path + "zcr_test.p", "rb" ) )

# find data point used in this small data set
list_key=list(zcr.keys())
list_key_test=list(zcr_test.keys())

### define function
def preprocess_data(list_key, feature):
    cleaned_feature_dict = defaultdict(np.array)
    cleaned_feature=[]
    for path in list_key:
        feature_result = dict(feature)[path]
        __, n_frames = feature_result.shape
        if n_frames < 2500: ## Remove all shorter mp3 files from analysis
            continue
        feature_result_fixed = feature_result[:, :2582] ## Resizes all np.arrays to 2582 frames

```

```

    scale = StandardScaler()
    scaled_feature_result = scale.fit_transform(feature_result_fixed)
    feature_result_long = scaled_feature_result.reshape(1,-1)
    cleaned_feature_dict[path] = feature_result_long
    cleaned_feature.append(feature_result_long)
    new_list_key = dict(cleaned_feature_dict).keys()
# return new data path and dictionary of its corresponding value
    return list(new_list_key), cleaned_feature_dict, cleaned_feature

def nonmatrix_data(list_key, feature):
    cleaned_feature=[]
    for path in list_key:
        feature_result = dict(feature)[path]
        feature_result_fixed = np.array(feature_result[:, :2582]).reshape(-1,1) ## Resizes all
np.arrays to 2582 frames
        scale = StandardScaler()
        scaled_feature_result = scale.fit_transform(feature_result_fixed)
        cleaned_feature.append(scaled_feature_result.reshape(1,-1))
# return new data path and dictionary of its corresponding value
    return cleaned_feature

def combinefeature(datalen, arr1, arr2, arr3, arr4, arr5):
    overallfeature=[]
    for x in range(datalen):
        a,b,c,d,e = arr1[x],arr2[x],arr3[x],arr4[x],arr5[x]
        overallfeature.append(np.concatenate((a,b,c,d,e), axis=1))
    return overallfeature

def combinefeature1(datalen, arr1, arr2, arr3, arr4, arr5):
    overallfeature=[]
    for x in range(datalen):
        a,b,c,d,e = arr1[x],arr2[x],arr3[x],arr4[x],arr5[x]
        a= np.array(arr1[x]).reshape(-1,1)[:10000].reshape(1,-1)
        b=np.array(arr2[x]).reshape(-1,1)[:10000].reshape(1,-1)
        c=np.array(arr3[x]).reshape(-1,1)[:2582].reshape(1,-1)
        d=np.array(arr4[x]).reshape(-1,1)[:2582].reshape(1,-1)
        e=np.array(arr5[x]).reshape(-1,1)[:2582].reshape(1,-1)
        overallfeature.append(np.concatenate((a,b,e), axis=1))
    return overallfeature

```

```

# %% data preprocessing
list_key_, chromafeature = preprocess_data(list_key, chroma)
list_key_, mfccfeature = preprocess_data(list_key, mfcc)
sc_feature = nonmatrix_data(list_key, spectral_centroid)
sr_feature = nonmatrix_data(list_key, spectral_rolloff)
zcr_feature = nonmatrix_data(list_key, zcr)
ttl_feature =
combinefeature(len(list_key), chromafeature, mfccfeature, sc_feature, sr_feature, zcr_feature)

list_key_test_, chromafeature_test = preprocess_data(list_key_test, chroma_test)
list_key_test_, mfccfeature_test = preprocess_data(list_key_test, mfcc_test)
sc_feature_test = nonmatrix_data(list_key_test, spectral_centroid_test)
sr_feature_test = nonmatrix_data(list_key_test, spectral_rolloff_test)
zcr_feature_test = nonmatrix_data(list_key_test, zcr_test)
ttl_feature_test =
combinefeature(len(list_key_test), chromafeature_test, mfccfeature_test, sc_feature_test, sr_feature_test, zcr_feature_test)

train_genere_id = [float(list_key[z].split('/')[1][-4]) for z in range(len(list_key))]
test_genere_id = [float(list_key_test[z].split('/')[1][-4]) for z in range(len(list_key_test))]
total_genere_id = []
total_genere_id.extend(train_genere_id)
total_genere_id.extend(test_genere_id)

# %% data preprocessing 2
TRACKS_PATH = '../data/fma_metadata/tracks.csv'
tracks = []
trackid_t = []
heads2 = []
i = 1
rawdata = []
with open(TRACKS_PATH, encoding="utf8") as csvfile:
    readCSV1 = csv.reader(csvfile, delimiter=',')
    for row in readCSV1:
        if i < 4:
            readhead = row
            heads2.append(readhead)
        else:
            readentry = [x for x in row]#[float(x) for x in row]
            rawdata.append(readentry)

```

```

        trackid_t.append(float(readentry[0]))
        tracks.append(readentry[40])
        i=i+1
index=np.searchsorted(trackid_t,total_genere_id)
tracks_new=[tracks[index[small]] for small in range(len(index))]
# label each catagory with numbers
le = LabelEncoder()
genre_id = le.fit_transform(tracks_new)
classes=list(le.classes_)
# fit the classes to this small data set
y_train, y_test = genre_id[:len(train_genere_id)],genre_id[len(train_genere_id):-1]
y_test=np.append(y_test,genre_id[-1])

x_train = np.array(ttl_feature).reshape(6384,-1)
x_test = np.array(ttl_feature_test).reshape(1596,-1)

# Logistic regression
clf = LogisticRegression(solver='sag', max_iter=1000, verbose=1)
clf.fit(x_train, y_train)
filename = 'sag27_model.p'
pickle.dump(clf, open(filename, 'wb'))
y_pred = np.array(clf.predict(x_test),dtype=np.int)
y_expected = np.array(y_test,dtype=np.int)
print("%.2f"%( ( np.sum(y_pred==y_expected)/y_expected.shape[0])*100 ) )

# Gaussian Naive Bayesian
clf = GaussianNB(var_smoothing=0.00000000011)
clf.fit(x_train, y_train)
filename = 'gaussian_model.p'
pickle.dump(clf, open(filename, 'wb'))
y_pred = np.array(clf.predict(x_test),dtype=np.int)
y_expected = np.array(y_test,dtype=np.int)
print("%.2f"%( ( np.sum(y_pred==y_expected)/y_expected.shape[0])*100 ) )

```

6.4.4 Result Analysis:

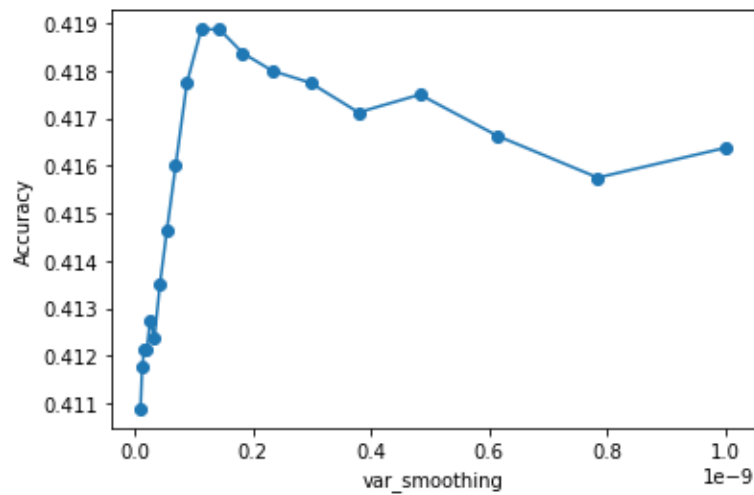


Fig. 6.9 Gaussian Naive Bayesian Accuracy Plot

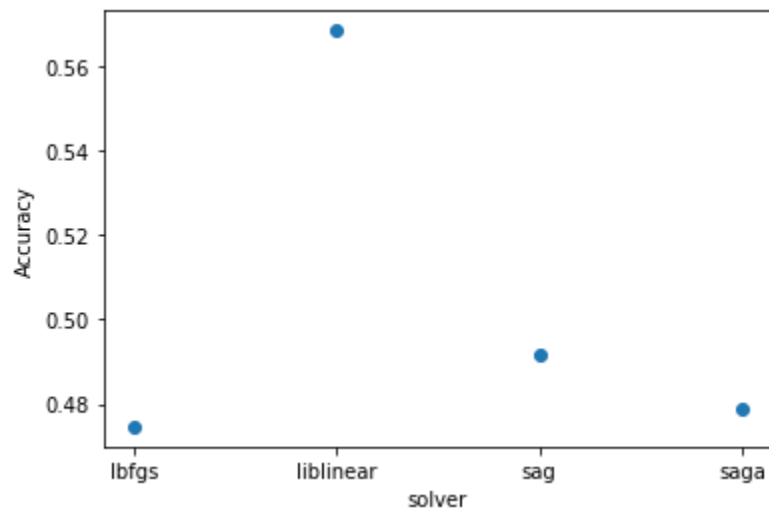


Fig. 6.10 Logistic Regression Accuracy Plot

6.5 Output

As proposed, a social media application with features of user interaction, friend suggestion, and music recommendation. The below figures show the output for the application. Where fig 6.11 and 6.12 shows the login screen and the sign-up screen where the user can authenticate itself and also register if he/she is a new user.

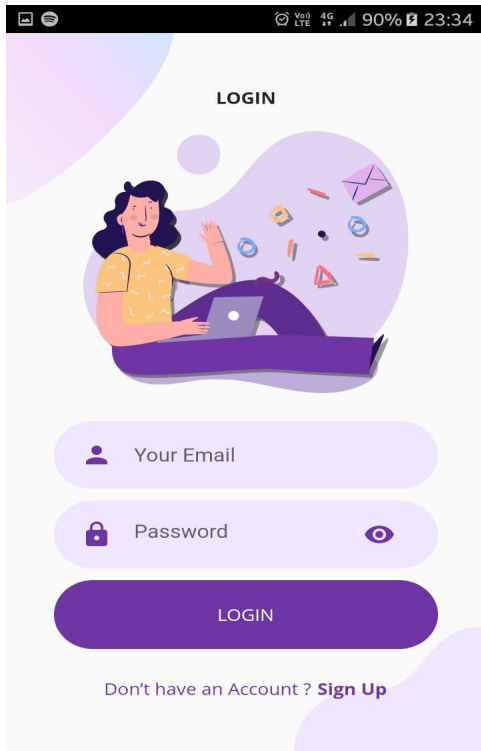


Fig. 6.11 Login Screen

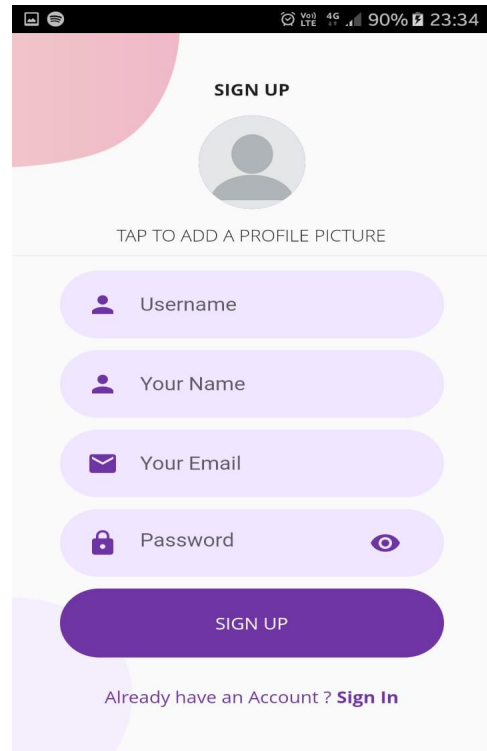


Fig. 6.12 Sign Up Screen

As for the users to manage and create the profile the figs 6.13, 6.14 & 6.15 show how users can edit, and interact with their own profile by editing the name and bio of the user. Getting the list of images posted and also delete the posts if wanted to.

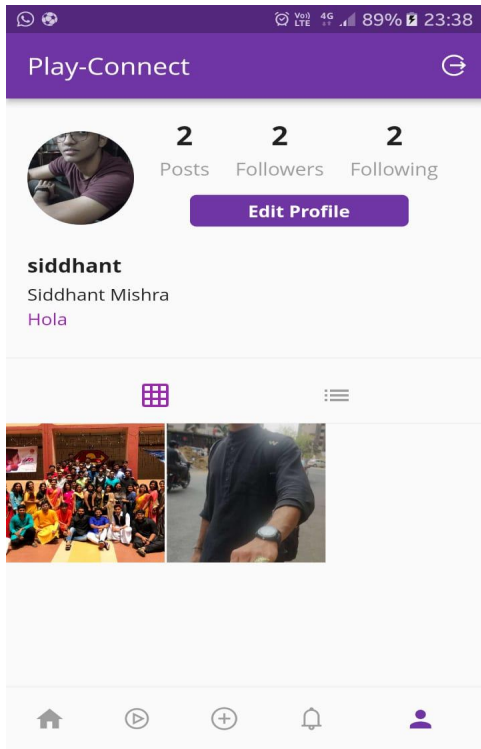


Fig. 6.13 User Profile

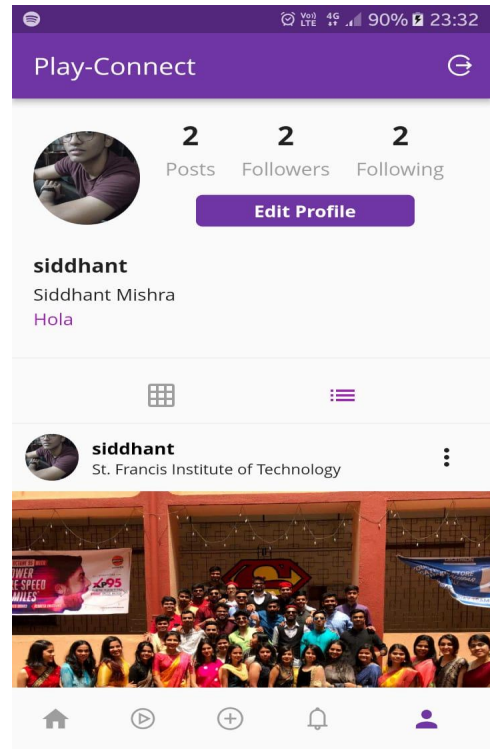


Fig. 6.14 User's Posts Screen

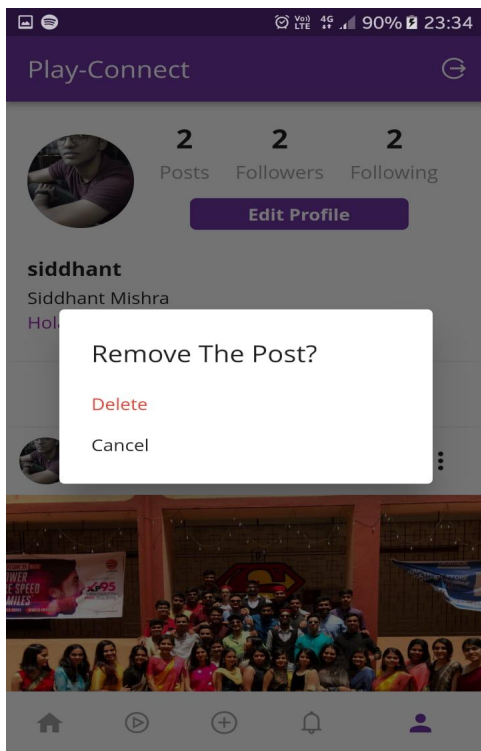


Fig. 6.15 User's Profile Interaction

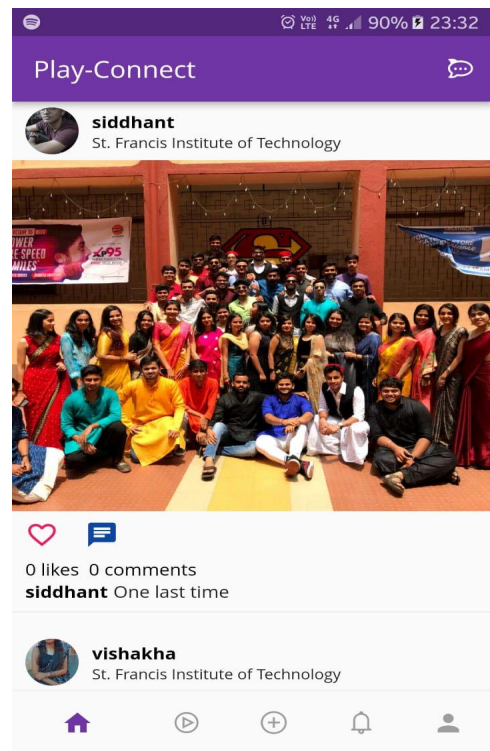


Fig. 6.16 Home Screen

Fig. 6.16 shows the application home page where the list of images posted by users are displayed and with the double-tap as well as using the like button user can like the post and also add comments to the posts. Fig. 6.17, is the music player which is connected to the remote Spotify account and displays the recently played music. The music player is interactable and the user can play songs, and also can change the track to play the next or the previous songs based on the music recommendations.

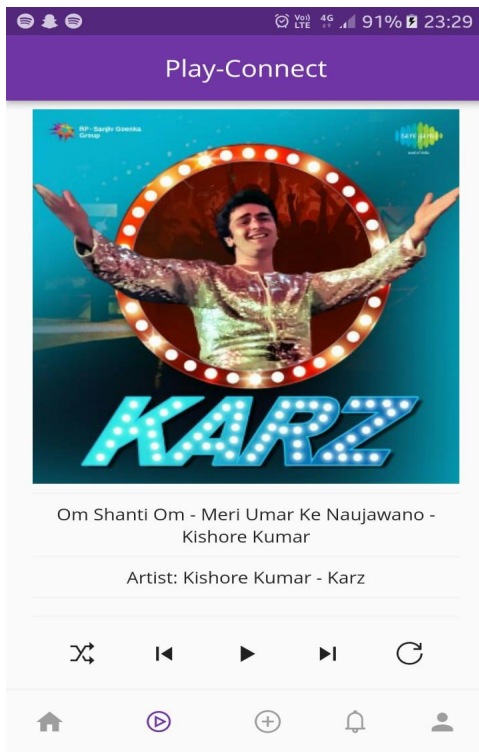


Fig. 6.17 Music Player with Remote Spotify

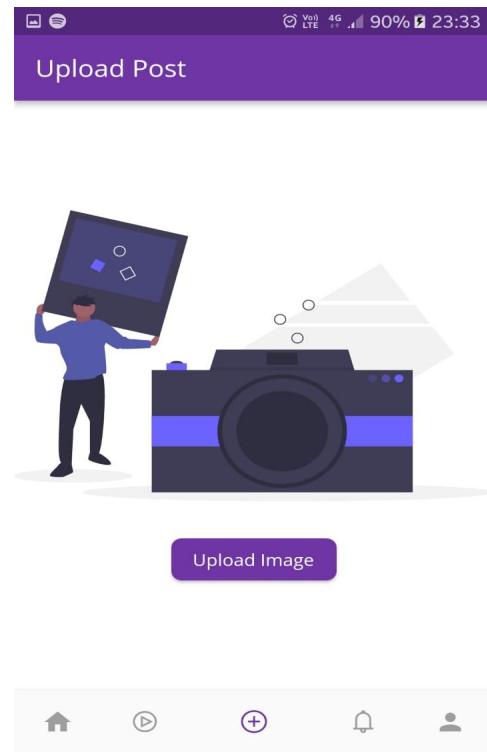


Fig. 6.18 Upload Post

Fig. 6.18 is the screen from where images can be posted and it has the option to upload images from the gallery or to click images using the phone's camera as shown in fig 6.19. After clicking a picture using the camera or by selecting from the gallery user can add a caption as well as the location of the picture clicked and post the image as shown in fig 6.20.

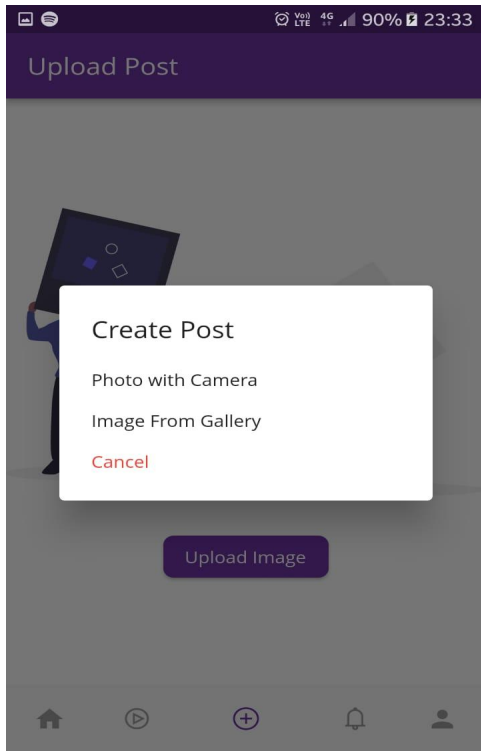


Fig. 6.19 Upload Posts - 2

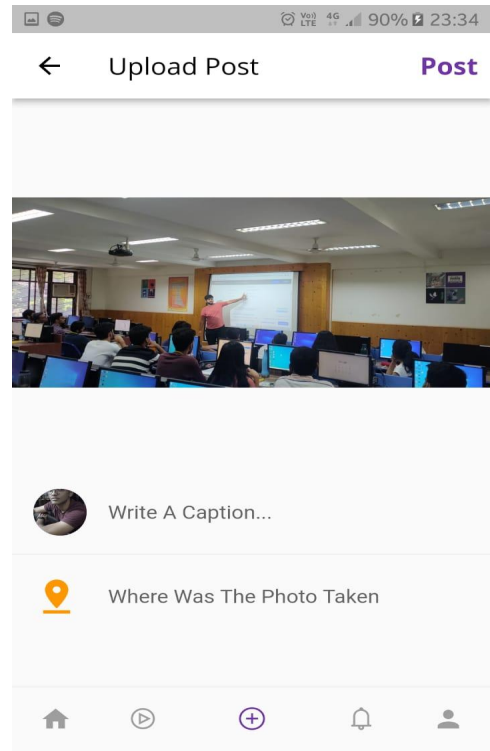


Fig. 6.20 Editing Post's Caption & Location

Interacting with users' posts and sending messages to friends are some of the prominent features of a social media application. Fig. 6.21 shows us the list of friends, a chat list of the friends the user has interacted with or sent messages to. Users can select any of the chat and can send messages as shown in fig 6.22. The user can also search for a user using the user's username shown in fig 6.23.

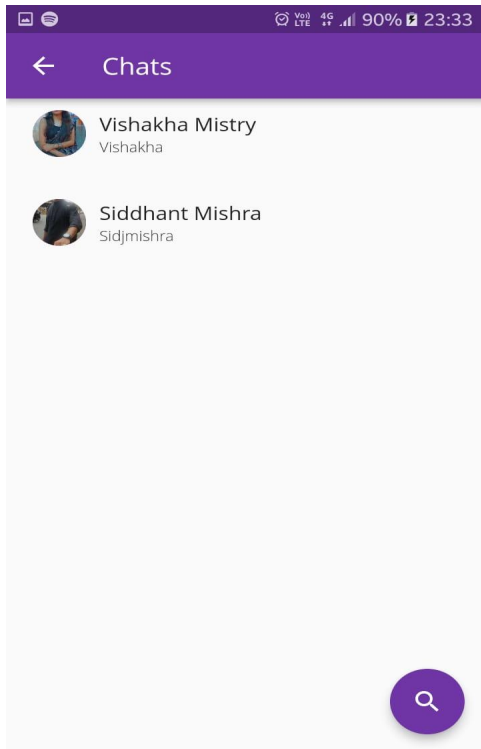


Fig. 6.21 User's Chats

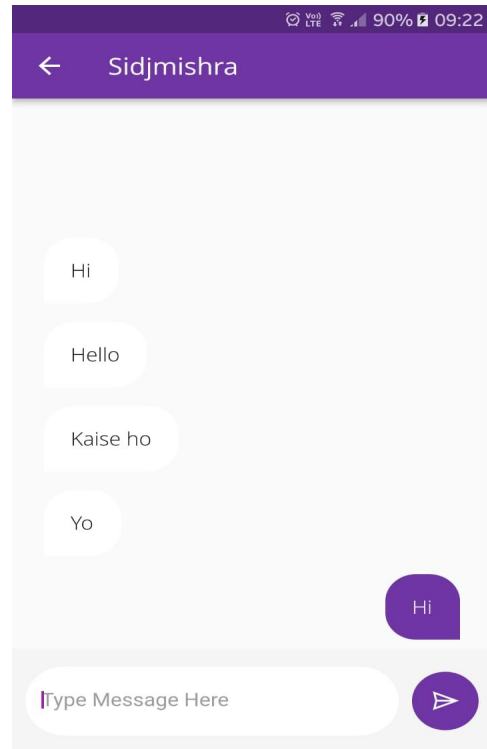


Fig. 6.22 Chat Screen

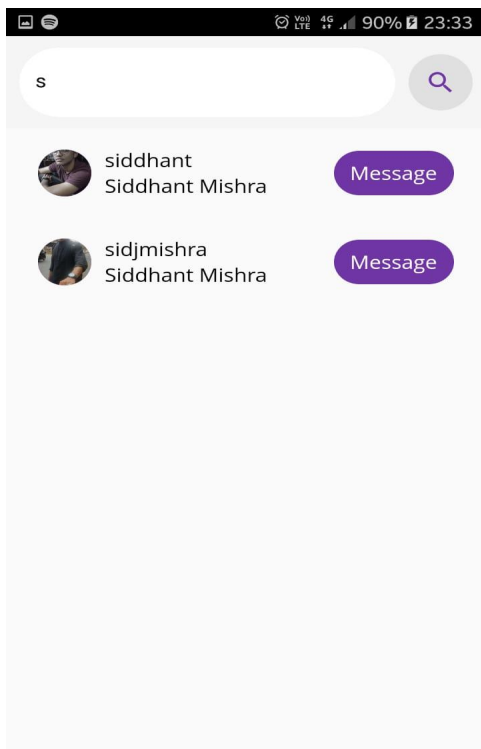


Fig. 6.23 Search Users to Interact

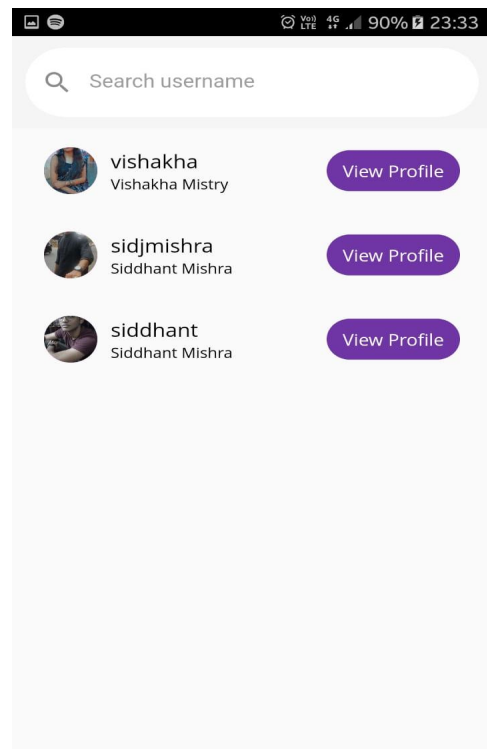


Fig. 6.24 Friends Suggestions Screen

Friends Suggestion is an important part where the user gets to know who all have a similarity to his/her likeness. Friends suggestion is a list of users who have similarities with one's own profile. Fig. 6.24 shows the list of friends suggested based on the similarity of the user's own profile.

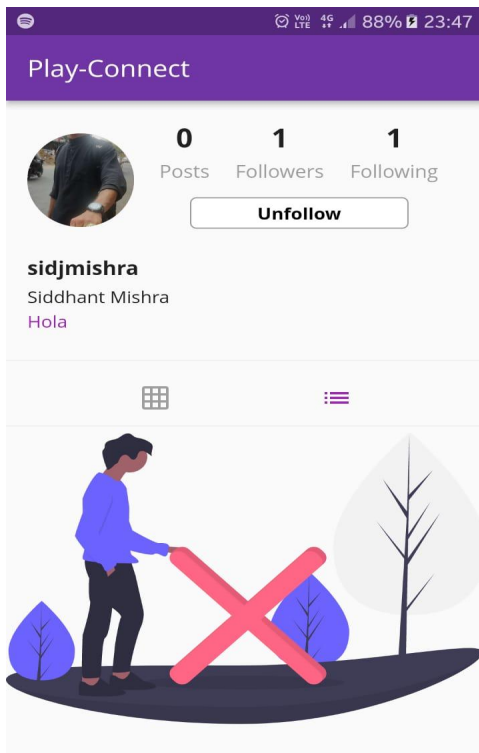


Fig. 6.25 Friends Profile

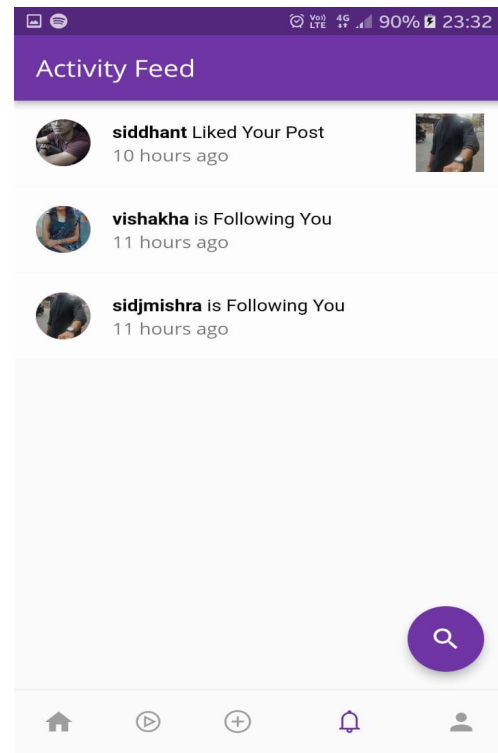


Fig. 6.26 User's Activity Feed Screen

People can also search for a user and view the user's profile and users can decide whether to follow or unfollow a user or friend respectively (fig 6.24 & 6.25). Fig. 6.26 shows the user activity feed where shows who liked their posts, commented on their posts, and started to follow him/her.

Finally, the social media application is the collection of all the features which are primarily required in today's world by the people. People can easily register themselves or authenticate using the authentication feature and once authenticated can access all the features of the social media application.

Chapter 7

Conclusion

Social media helps people establish better relationships with their family and friends. Finding compatible people to be friends on social media is often a challenge as many of the people recommended to the user by social media are people that are already friends with them or are followed. However, when users are trying to find friends, the important concern is whether or not they need common interests or hobbies with each other and whether or not they often interact with one another. People with similar music tastes make better connections so we bring together people who like the same music, to share experiences which are currently missing out on. Our goal is to provide the user with a personalised experience for music recommendation which revolves around his musical interests along with suggesting friends that enjoy the similar taste in music. Using music as the basis for common interests, we proposed an application for friends and music recommendation revolving around music interests and current context of the music.

References

- [1]. Pichl, Martin & Zangerle, Eva. (2021). User models for multi-context-aware music recommendation. *Multimedia Tools and Applications*. 80. 1-23. 10.1007/s11042-020-09890-7.
- [2]. Adiyansjah, Alexander A S Gunawan, Derwin Suhartono, Music Recommender System Based on Genre using Convolutional Recurrent Neural Networks, *Procedia Computer Science*, Volume 157, 2019, Pages 99-109, ISSN 1877-0509, <https://doi.org/10.1016/j.procs.2019.08.146>.
- [3]. S. Chang, A. Abdul, J. Chen and H. Liao, "A personalized music recommendation system using convolutional neural networks approach," 2018 IEEE International Conference on Applied System Invention (ICASI), 2018, pp. 47-49, doi: 10.1109/ICASI.2018.8394293.
- [4]. K. Zhang, Z. Zhang, K. Bian, J. Xu and J. Gao, "A Personalized Next-Song Recommendation System Using Community Detection and Markov Model," 2017 IEEE Second International Conference on Data Science in Cyberspace (DSC), 2017, pp. 118-123, doi: 10.1109/DSC.2017.14.
- [5]. C. Fan, H. Hao, C. K. Leung, L. Y. Sun and J. Tran, "Social Network Mining for Recommendation of Friends Based on Music Interests," 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), 2018, pp. 833-840, doi: 10.1109/ASONAM.2018.8508262.
- [6]. Cheng, Shulin & Zhang, Bofeng & Zou, Guobing & Huang, Mingqing & Zhang, Zhu. (2019). Friend recommendation in social networks based on multi-source information fusion. *International Journal of Machine Learning and Cybernetics*. 10. 10.1007/s13042-017-0778-1.
- [7]. J. Liu, L. Fu, X. Wang, F. Tang and G. Chen, "Joint Recommendations in Multilayer Mobile Social Networks," in *IEEE Transactions on Mobile Computing*, vol. 19, no. 10, pp. 2358-2373, 1 Oct. 2020, doi: 10.1109/TMC.2019.2923665.
- [8]. A. A. Rezaee and N. Abravan, "A hybrid friend-based recommendation system using the combination of Meta-heuristic Invasive weed and genetic algorithms," 2020 10th International Conference on Computer and Knowledge Engineering (ICCKE), 2020, pp. 665-669, doi: 10.1109/ICCKE50421.2020.9303619.
- [9]. Y. Yin and X. Feng, "Friend Recommendation Algorithm Based on Interest and Cognition Combined with Feedback Mechanism," 2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), 2019, pp. 1025-1030, doi: 10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00199.
- [10]. Kunhui Lin, Yating Chen, Xiang Li, Qingfeng Wu and Zhentuan Xu, "Friend recommendation algorithm based on location-based social networks," 2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2016, pp. 233-236, doi: 10.1109/ICSESS.2016.7883056.
- [11]. Q. Shen, H. Zhou, S. Li and Z. Pei, "Friend Recommendation Algorithm Based on Interest Classification with Time Decay," 2017 International Conference on Network and Information Systems for Computers (ICNISC), 2017, pp. 117-121, doi: 10.1109/ICNISC.2017.00033.
- [12]. E. Daniyalzade & T. Lipus, Facebook Friend Suggestion, CS 229 (2007) project report, Stanford University.

- [13]. X. Yang, Y. Wang, D. Wu, & A. Ma, "K-means based clustering on mobile usage for social network analysis purpose," in IMS 2010, pp. 223-228.
- [14]. V. Soundarya, U. Kanimozhi, & D. Manjula, "Recommendation system for criminal behavioral analysis on social network using genetic weighted k-means clustering," JCP 12(3), 2017, pp. 212-220.
- [15]. J. Ahn, "Digital divides and social network sites: which students participate in social media?" JECR 45(2), 2011, pp. 147-163.
- [16]. R. Chulyadyo & P. Leray, "A personalized recommender system from probabilistic relational model and users' preferences," in KES 2014, pp. 1063-1072.
- [17]. "Spotify Web API". <https://developer.spotify.com/documentation/web-api/reference/#/>. (Accessed Jul. 8, 2021).

Acknowledgements

It would be highly unfair on our part to claim credit for the whole project. Hence with due respect, we express our sincere gratitude to our esteemed Principal Dr. Sincy George for this teaming experience and for giving us an opportunity to work on this project.

We sincerely want to thank our project guide Ms. Priya Karunakaran for her valuable guidance and advice she has given us when faced difficulties while working on the project. We are also grateful for her constructive criticism and for helping us develop this idea to pursue the project.

We would also like to thank our Head of the Department, Dr. Kavita Sonawane who gave us this wonderful opportunity. We are fortunate enough to get encouragement and support from all the teaching staff of the computer engineering department. We would also like to extend our sincere regards to all the non-teaching staff of the computer department for their timely support.