Music Cultural Community Management System

Bao Rui
the Chinese University of Hong Kong, Shenzhen
School of Science and Engineering
Student ID: 120090738
ruibao@link.cuhk.edu.cn

Lin Yuyang
The Chinese University of Hong Kong, Shenzhen
School of Science and Engineering
Student ID: 120090377
yuyanglin1@link.cuhk.edu.cn

Jin Derong
The Chinese University of Hong Kong, Shenzhen
School of Data Science
Student ID: 120090562
derongjin@link.cuhk.edu.cn

Tong Zhen
The Chinese University of Hong Kong, Shenzhen
School of Data Science
Student ID: 120090694
zhentong@link.cuhk.edu.cn

1 INTRODUCTION & MOTIVATION

1.1 Background

Emotion is important. Humans are emotional creatures; our emotions influence the decisions we make, the career path we take, the music we enjoy, and the product we buy. Emotions have great power, and we want to make use of it. Music is the best way for us to let out our emotions. But our focus is not on the music itself, we are focusing on the story and culture behind it. Music is essential and common in people's daily life, but there's still a hidden world of fantasy waiting for people to discover. Every heartbreaking rhythm comes with an everlasting story to tell, but no one is willing to speak for them. The market is rich in music applications that helps its user get access to millions of beautiful melodies, but they pay no attention to the culture and story behind them. *GuYongZhe* by Eason Chen is a powerful song that roars the anger and passion for those brave people, but few know that the inspiration comes from its composer, a cancer patient's ten years of fighting cancer on her own. Such stories will deepen and amplify the emotion expressed by the song.

1.2 Motivation

As a company that works closely with record publishers, advertising agencies, and songwriters, we designed this powerful and dedicated music cultural community management system (MCCMS) to solve the problems raised above.

MCCMS contains seven entities, which are users, songs, artists, genres, albums, rank-lists and play-lists. Besides, there are eleven relationships among these entities. We presented our Entity-Relation diagram to demonstrate and and clearly illustrated each relationship in the implementation section and we reduced it in Boycecodd Normal form. All dependencies are lossless and every determinant is a super key. We populate the database with realistic data from qq music and Net-East Music by using crawl technique. We also provided user friendly graphical interface to allow users do simple queries.

MCCMS benefits our clients and users in various ways. For **ordinary users**, MCCMS allows them freely to send out their comments and feelings to share with common-interest fans. Also, the users are able to learn the interesting history and stories behind the songs by searching in the system.

Our clients can better learn the tastes and preference as well as the characteristics of the song listeners from MCCMS.

For **songwriters**, they can view the fan characteristics through the background data, as well as the comments. These information are generated by our data mining algorithm, and can contribute to the composers next creation. For **publishers**, they can issue high quality products and build the brand image using the data mining outcome we provided. For **advertising agencies**, they can choose more suitable songs for advertising by analyzing the emotions behind each song.

2 DATABASE STRUCTURE DESIGN

According to the motivation and users' requirements, we developed the E-R diagram of our database. There are 8 entities and 5 relationships (In **Fig.1** blue for entity and green for relationship).

The core entity is the song is a table for the song data. Each song has a unique song_id as a Primary Key. Two songs may have the same name, so it cannot be a Primary Key. Each song is attached with a genre. We assume that each song must be included in one album when published and cannot be in two lists. The review attribute can access the Review table. The story attribute is a URL link.

The entity called Artist includes musicians and bands in reality. Each song will be created by one artist. Notice we used a trick to deal with temporary collaboration and band. We define a band as an artist. Other musicians can be one band member but remain independent artists. For a temporary collaboration, an artist will also have a non-empty attribute called group_id, which is a Foreign Key to the AritistInGroup relation.

The ArtistInGroup relationship has only two attributes. One is the primary key to identifying itself. Another is the artisit_id to locate the artist associated with the group. Notice, the ArtistInGroup is not a band.

The entity called Album has a unique album_id. We cannot use the album name as the Primary Key because different albums can

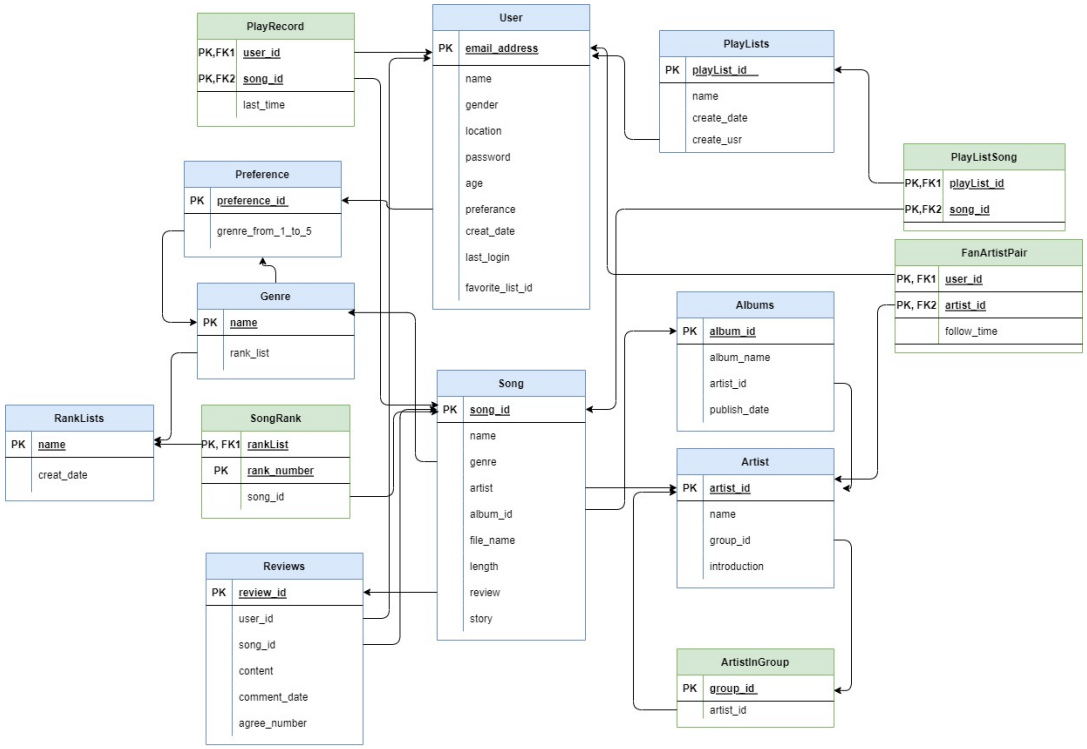


Figure 1: E-R diagram of our database

have the same name. The artist_id attribute has the same design as Song.artist_id.

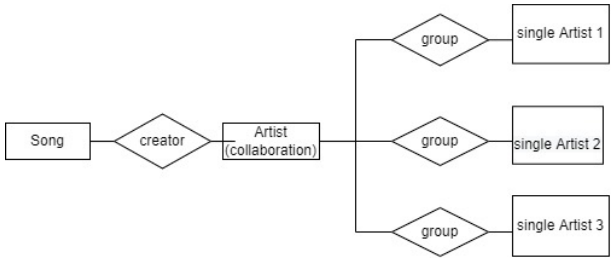


Figure 2: Song, Artist, and ArtistInGroup association

- The User entity uses the users' email for registry as a Primary Key. The Foreign Key favorite_list_id will point to one specific PlayList that collects all the songs users liked. This PlayList will be initialized when the account is built.
- The PlayList entity is a set of songs. It has a Foreign Key create _user_id point to the user who creates it.
- The PlayListSong is a relationship between PlayList and Song.
- The PlayRecord is a relationship between user and song. The last_time attribute can record the play history. If the last_time is too long, the database will delete the data to save memory.
- The Genre entity is a style. Its Primary Key is its name because there cannot be two genres with the same yet different styles. It has a rankList_id point to RankList with its genre.
- The RankList entity contains all the songs with its genre.
- To get the specific song data in a RankList, we can visit the SongRank relationship. It has Primary Keys. With the RankList and rank_number, we can find one specific song_id.
- The Review entity has a review_id as a Primary key. It has user _id to find who create this comment and song_id to find where the comment is. The agree_number attribute is the number of users who like this review.
- The FanAritistPair is the relationship between fan and artist. It has two Foreign Keys as the Primary Key. The follow time record is when the user discovers his/her love artist.

In our E-R model, we use the BCNF form that for all the tables, the functional dependency is based on the primary key of the relation. All the decomposition is lossless.

The first indexing design is the rank list. When we want to find them in one rank list, we want to access all the songs in some block. So it is convenient to store the songs of the same rank list in one bucket. It uses the No.1 song of a specific rank list. So the next time we need to find songs in the rank list, the searching time is deducted from O(N) to O(1). We designed the Fans_Artist_pair data in cluster index because we sometimes want to know the number of fans of an artist and what singers a user like.

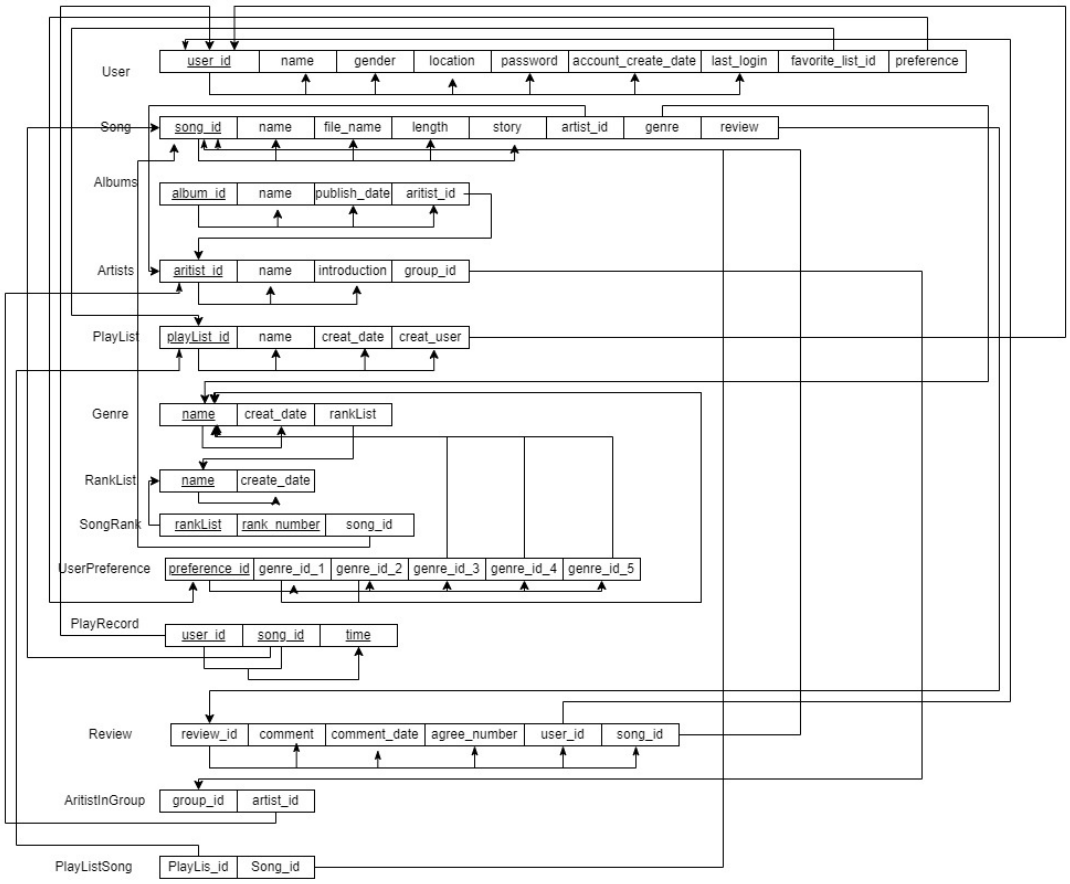


Figure 3: Normalization & Functional Dependency

3 IMPLEMENTATIONS

3.1 Data Preparation

In our database, most of the data is information about users and songs, and the relationships that exist between the two entities.

1. Data collection for music information Information about songs is available in reality in large numbers, and the approach we took was to use a web crawler to crawl the internet for **real** information about songs. QQ Music is one of the most ideal sites for crawling music data for two reasons: 1. the number of music artists on QQ Music is large and complete 2. QQ Music allows access to music/artist pages without logging in, making it less difficult to crawl. In the following paragraphs we will describe the methods used in our data crawling process.

Since the QQ Music database is very large (estimated to be over 1000,000 songs when crawling in practice), not only would it be inefficient to crawl songs randomly (requiring guessing the composition of the url), there is a risk that the distribution of the collected data is too diffuse, e.g. very few songs are from the same singer. So here we use a hierarchical crawling approach: first visit a part of the artists in the api, crawl the songs they have composed, and finally crawl the hot reviews and lyrics of the songs.

During the crawling process, we chose to use python's scrapy framework to write the crawler code (based on this). .

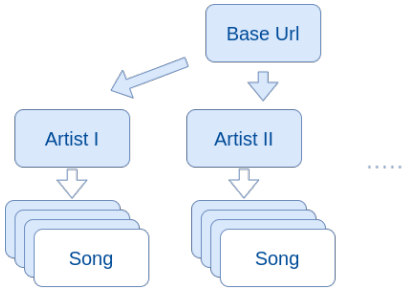


Figure 4: A diagram demostrating the hierarchical collecting process of music information

The crawled data were saved in a .json file. We also did data cleaning such as converting xml characters to utf8 encoding. to normal ones in the lyric.

https://u.y.qq.com/cgi-bin/musicu.fcg?format=json&data=%7B%22req_0%22:%7B%22module%22:%22vkey.GetVkeyServer%22,%22method%22:%22CgiGetVkey%22,%22param%22:%7B%22guid%22:%22358840384%22,%22songmid%20%22:%5B%22%20001X0PDf0W4lBq%20%22%5D,%22songtype%22:%5B0%5D,%22uin%22:%221443481947%22,%22loginflag%22:1,%22platform%22:%2220%22%7D%7D,%22comm%22:%7B%22uin%22:%2218585073516%22,%22format%22:%22json%22,%22ct%22:24,%22cv%22:0%7D%7D

Figure 5: Sample of api-url of QQ music (collect from [2])

Table 1: Convert samples

Original	Converted
\n
	space
(	(
'	\'
"	\”

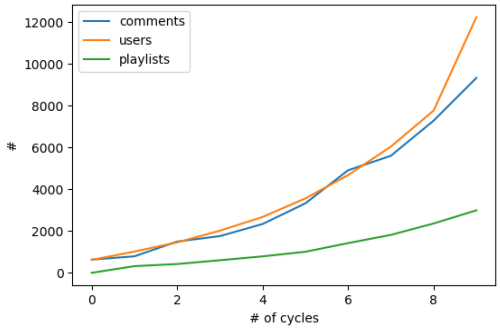


Figure 6: The total number of generated data after # cycles

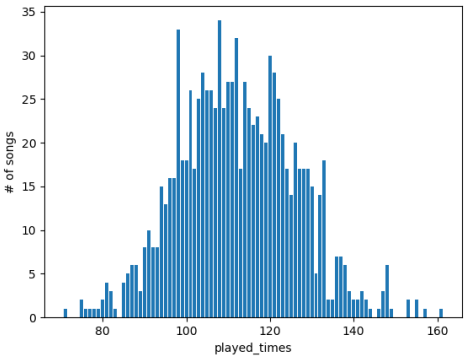


Figure 7: The Distribution of Played Frequency among Songs

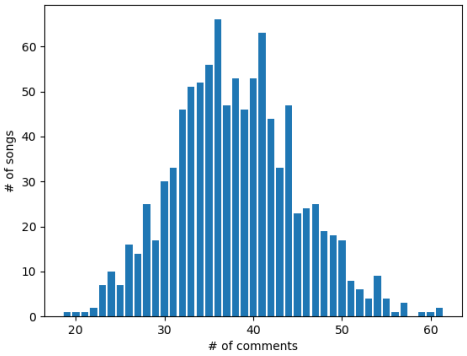


Figure 8: The Distribution of Comments among Songs

By this method, we collected the basic information of more than 410000 songs in total. Since crawling audio involves copyright issues, all the data are without audio information but replaced by music URLs.

Since one of the major goals of our database is to correlate music with user sentiment, we chose to discard most of the data in order to better analyze the music data that contains the most information and to prevent it from being “diluted” by general music information. Finally we kept **2300** music tracks with the highest number of hot reviews for analysis

2. Data generation for user information Due to the privacy clause, it is not feasible to obtain real user information directly through crawler method. So we simulate users to generate data. First, 600 initial users were generated and their information was randomized. Then we performed several rounds of loops, and in each loop, we generated new users’ information based on the existing old users’ profiles, and simulated and updated the users’ behavior such as favoriting music and following artists. Meanwhile, the probability of each user bringing new users to the system in each week decreases with time.

In the end, we generated a total of 12,247 users’ information. Compared with direct random generation of user data, our method makes the user information closer to the distribution in real situations. As shown in the figures below, the number of users, comments, and song lists have increased exponentially, which is more in accordance with the change of database capacity at the beginning of the establishment, and the number of comments and plays of the final songs also roughly fit the bell-shaped curve.

3.2 Database System Construction

With the information of users and songs, we can set up the primary MCCMS. For example, inserting a record is easily accomplished in SQL by:

```
INSERT INTO `Songs` (
    `name`,     `genre`,        `artist`,
    `album`,    `file_name`,    `story`,
    `lyric`,    `time_played`,  `time_played_full`,
) VALUES
( ... ), ( ... ), ..., ( ... );
```

Figure 9: SQL statement of inserting songs to table 'Songs'

Remarkably, it is necessary to specify the correct insertion order when inserting data, in addition to aligning data and labels.

Take 'Songs' as an example, the records of corresponding album and artist should be ensured to exist before a new song is inserted. With such dependencies, we choose a recursive-check-update approach to build the database (if there exists the record of the artist then use it, otherwise create one).

3.3 MCCMS Web App & Graphic User Interface

In this project, we built a simple admin side web app for MCCMS and designed a graphical user interface with multiple functions. The design and construction was done on three levels: front-end, back-end, and database.

Table 2: Details about frameworks/environment

level	framework/environment
server	localhost: 'Ubuntu 20.04 focal'
database	MySQL 8.0.28
back-end	Python 3.7.13 / django 3.2.13
front-end	Vue 2.9.6 / TypeScript 4.6.4 / JavaScript 10.9.0

The choice of these development environments was based on the principle of saving time and effort. At the same time, this project was implemented with separation of front and back ends, making an attempt to improve maintainability and increase the scale further in the future. The Vue template we used is available on github.

The App allows the user to interact with the system in various ways, the logged in administrator can view tables of songs, artists, albums, and users, and has the authority to modify, delete these records. Take viewing songs as an example, the app allows the user to set the max records shown in one pages as well as searching the songs whose name contains a certain string.

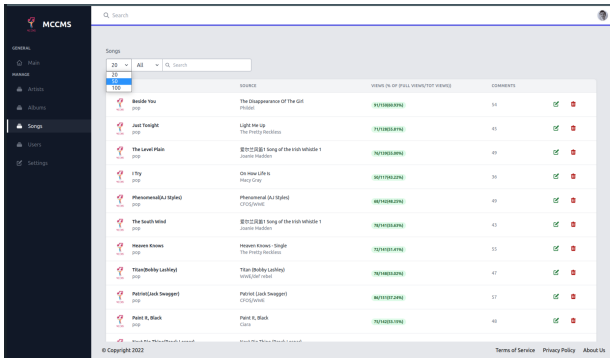


Figure 11: Changing the number of shown records in one page

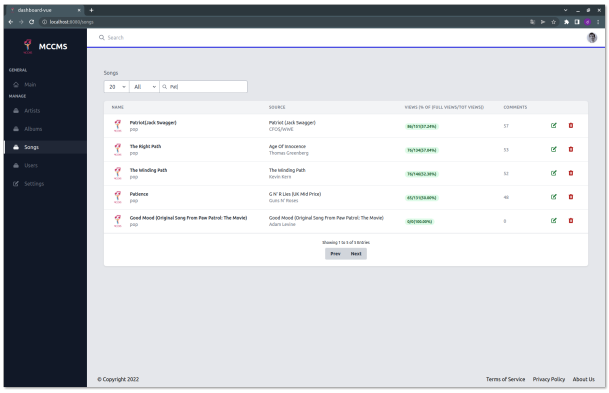


Figure 12: Searching Songs by Name Patterns

4 SQL FUNCTIONAL QUERIES

The "song" entity is the largest entity in this system, both in terms of the number of attributes and the number of relationships with other entities, and the queries on this entity are the most complex and varied. So in this section, we will focus on demonstrating sql queries on the "Songs" entity.

```
CREATE TABLE `Songs` (
    `id`                INT             NOT NULL        AUTO_INCREMENT,
    `name`              VARCHAR(511)    NOT NULL,
    `file_name`         VARCHAR(511)    DEFAULT NULL,
    `story`             TEXT            NOT NULL,
    `time_played`       INT             NOT NULL,
    `time_played_full`  INT             NOT NULL,
    `lyric`             TEXT            NOT NULL,
    `album_id`          INT             NOT NULL,
    `artist_id`         INT             NOT NULL,
    `genre_id`          VARCHAR(31)     DEFAULT NULL,
    PRIMARY KEY (`id`),
    UNIQUE KEY `song_uniq` (`name`,`artist_id`,`album_id`),
    FOREIGN KEY (`album_id`)    REFERENCES  `Albums`    (`id`)
                                ON DELETE CASCADE,
    FOREIGN KEY (`artist_id`)   REFERENCES  `Artists`   (`id`)
                                ON DELETE CASCADE,
    FOREIGN KEY (`genre_id`)    REFERENCES  `Genres`    (`name`)
)AUTO_INCREMENT = 1000000 CHARACTER SET = 'utf8mb4';
```

Figure 13: Create Table

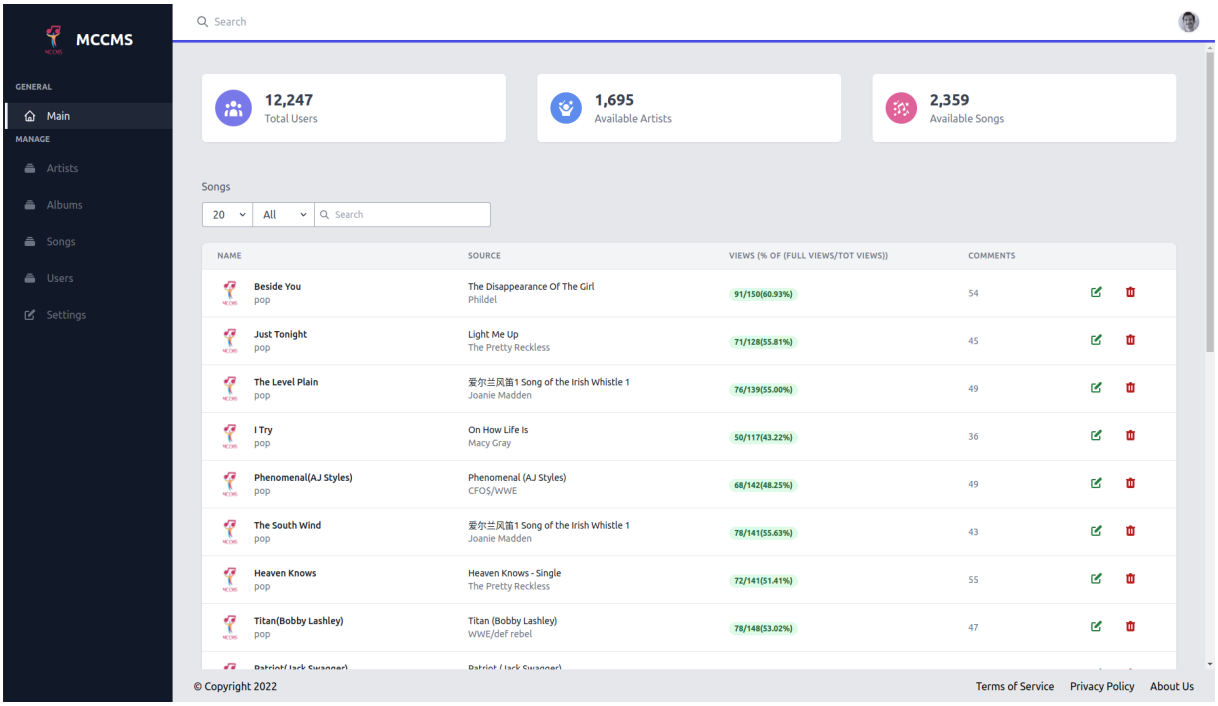


Figure 10: Main Page of MCCMS Web

```
SELECT  `song`.`name`       AS      'Song Name',
        `artist`.`name`     AS      'Artist Name',
        `album`.`name`      AS      'Album Name'
FROM    `Songs`     AS      `song`,
        `Artists`   AS      `artist`,
        `Albums`    As      `album`
WHERE (
    `song`.`album_id`=`album`.`id`
    AND `song`.`artist_id`=`artist`.`id`
    AND `song`.`time_played`>200
)
LIMIT 10;
```

Figure 14: This query shows the first 10 songs whose played times is over 200 with their names, artist, and album information

```
SELECT  `song`.`name`           AS      'Song Name',
        `song`.`time_played`    AS      'View Count',
        `song`.`time_played_full`/`song`.`time_played`
                                AS      'Completion Rate',
        `review`.`#`            AS      '# of Comments'
FROM    `oxygen_songs`  AS  `song`,
        (
            SELECT  `song_id_id`    AS  `song_id`,
                    COUNT(*)        AS  `#`
            FROM        `oxygen_reviews`
            GROUP BY    `song_id_id`
        )               AS  `review`
WHERE   `song`.`time_played` > 20
    AND `review`.`song_id`=`song`.`id`;
```

Figure 15: This query shows name, views statistics, and comment numbers of songs

```
UPDATE `Songs`
SET `time_played` = `time_played` + 1
WHERE (id=12);
```

Figure 16: This query updates a song's total played times.

```
DELETE FROM `Songs` WHERE (`id`=12);
```

Figure 17: Deletion Operation

5 DATA MINING

5.1 Data Mining Motivation & Target

The experiment dataset will be the dataset we collected using crawler methods from the QQ music platform, which means the following experiment and analysis will be carried out on the real-world data. One trade-off of using such data is that real-world data may not be consistent in the case of format. There might be some unexpected contents that our analysis method failed to toggle. Therefore, we will manually select some high-quality data and compatibility for further analysis. Hence that the data is mainly in Chinese and the research we conduct is also in Chinese, so the result will be shown primarily in Chinese. Translation between languages may lose some of the emotional features, so we keep all the data in its original language for better analysis outcomes. However, we will

also provide some English examples result for better understanding.

As stated in the background section, MCCMS aims to provide a database management system for the emotional values behind the music and its story. And we have already accomplished such a goal in the database implantation part by adding story attribute, review attribute etc. Naturally, we want to provide some convenient method for our potential clients to view and understand the emotional value of people when listening to a specific song. For example, if APPLE wants to make an advertisement about the high performance of its devices, they might want to find some background music that will make its listener feel energetic and excited. And this is what our company is good at. We will conduct a sentiment analysis based on natural language processing that can label each song with an emotion tag based on the comments and story behind it. In summary, we will conduct a machine learning model that treats each comment as input and the emotion label as output. Also, the visualization of the result will be present in the form of the word-cloud picture.

5.2 Method

1) Data Preparation:

The data we got from crawling the QQ music is in the form of json file. The overall data processing procedure is as follows. The first task is to get the comment content and its corresponding song name from the original data. This task is accomplished by python, and we get 2147 txt file where the name of the file is the name of the song, and the different comment is separated by a line in the text file.

Figure 18: Data mining model abstraction

2)Training:

Since the data we get does not have a label, the task we are going to solve falls into the category of unsupervised learning. Due to the time limit and the quality of the result, we don't label those data manually, and we will use an open-source pre-trained model provided by Baidu, Sentiment Knowledge Enhanced Pre-training for Sentiment Analysis [3]. This model is pre-trained in common Chinese NLP datasets like ChnSentiCrop, AI-challenge, and SE-ABSA16_PHNS and has excelled over the previous model by 2% in general.

We feed the model with the processed data for prediction. For a specific song, its comment will be analyzed separately, and the result will be taken average accordingly. The result will be in the form of a tuple. The first component will tell you whether this comment is negative or positive, and the second component will show the emotion score this comment got.

Also, for better visualization results, we use ckpe toolkit based on pkuseg [1] to do the word segmentation and find the key phrases of all the comments of a song and then choose the top 5 key phrases together with its emotion label to form a word-cloud picture.

5.3 Result Analysis

(a) Let it be (b) 7 years

(c) Carry on (d) Celestial

(e) Distance (f) Divinity

(g) From praying plans into the fire

Figure 19: Emotion word-cloud figure

We found that the prediction accuracy is not good overall due to the comment data we collected from QQ music. Most data and word fragments are actually not related to emotion. However, they are still taken into account in analysis because that the current model cannot automatically detect and delete those data. Also, our classifier can only give out positive and negative labels. This is because the performance of giving only two categories is better than five or seven.

6 CONCLUSION

In this project, we assume the position of a database development and analysis company to design and implement a database based on a specific user. In the project realization process, we try to stand

in the position of users and customers, manipulate data, and explore the application value of data. We have a deeper understanding of the concepts and practical values in the database.

By applying Vue&TypeScript for front-end implementation, and Python for database construction, front-end and back-end communication, data generation, and analysis, MCCMC successfully allows users to log in and search the website with optimization while maintaining the concurrency of the database. We also provide helpful information for our clients, such as a song's popularity, through data analysis and data mining.

For further improvement, we can label our data with emotion tag and train the pre-trained model based on our specific data to have better performance in data mining part. Besides, we will improve our Web App to support administrator changes to user and other information, as well as implement a user-side Web App that will allow MCCMS to interact directly with real users.

The group members actively discussed and efficiently divided tasks during the teamwork process andxcooperated. In the end, with the joint efforts of everyone, the tasks in our envisaged scenario were completed.

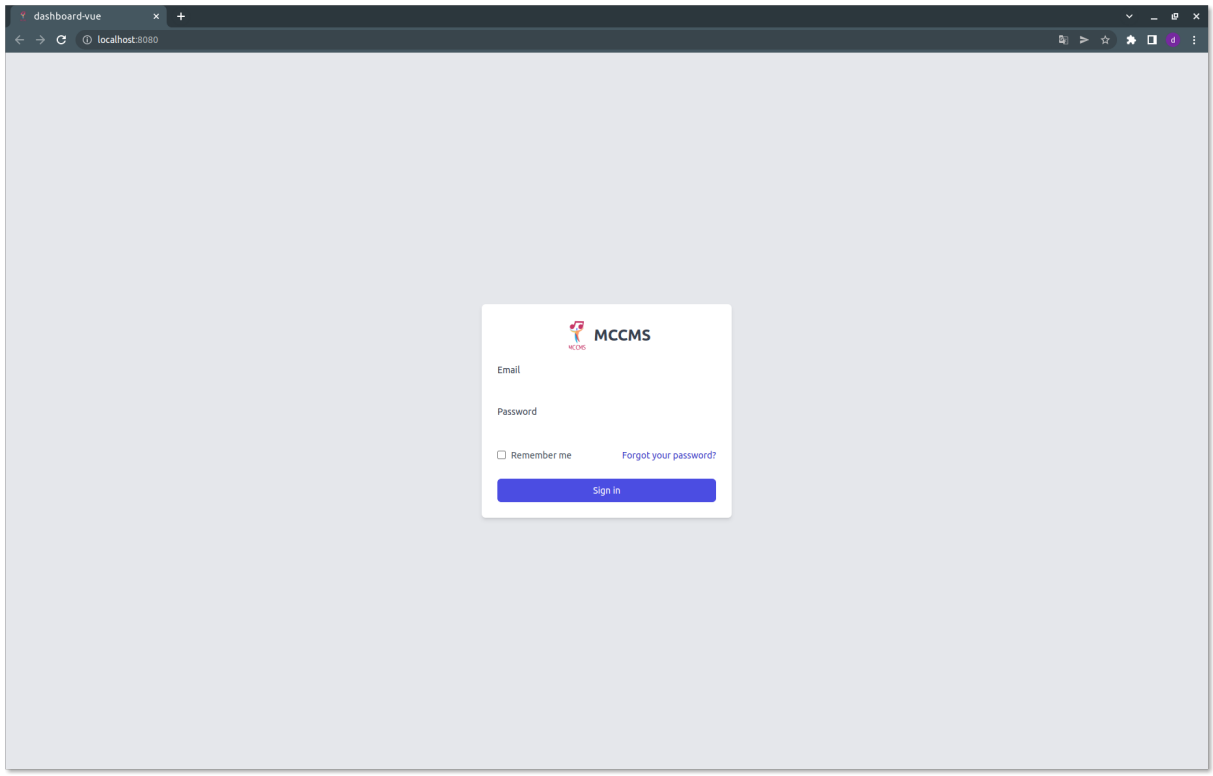
REFERENCES

[1] Ruixuan Luo, Jingjing Xu, Yi Zhang, Xuancheng Ren, and Xu Sun. 2019. PKUSEG: A Toolkit for Multi-Domain Chinese Word Segmentation. https://doi.org/10.48550/ARXIV.1906.11455

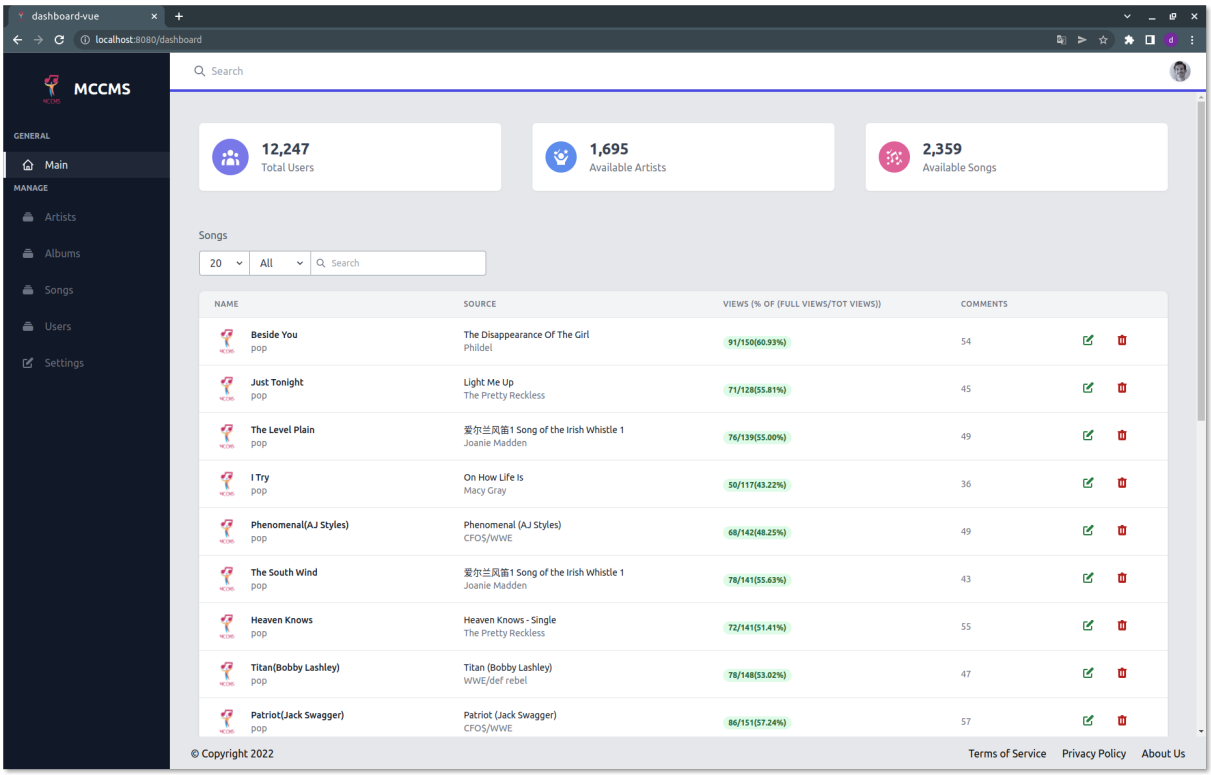
[2] n. d. 2020. Using Python Crawler to Download from QQ Music With Details. https://www.pythonf.cn/read/100623

[3] Hao Tian, Can Gao, Xinyan Xiao, Hao Liu, Bolei He, Hua Wu, Haifeng Wang, and Feng Wu. 2020. SKEP: Sentiment Knowledge Enhanced Pre-training for Sentiment Analysis. https://doi.org/10.48550/ARXIV.2005.05635

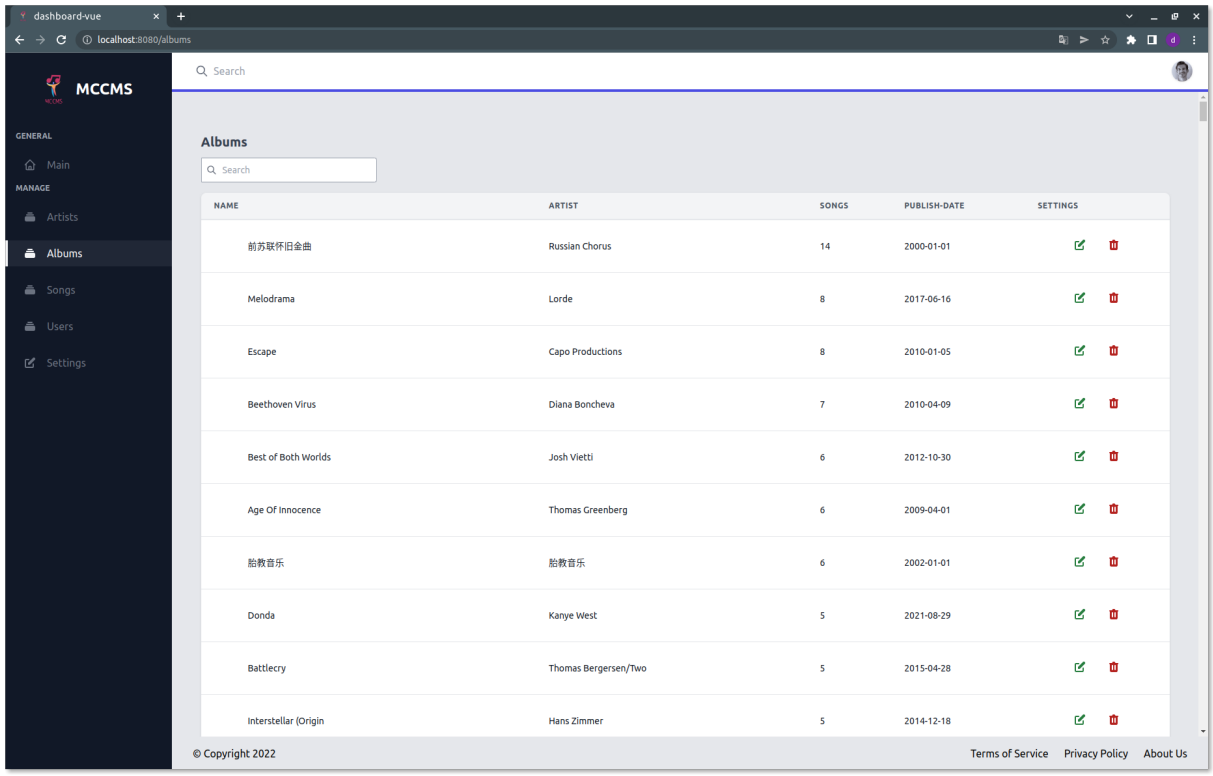
APPENDIX


dashboard-vue
localhost:8080
MCCMS
Email
Password
Remember me
Forgot your password?
Sign in

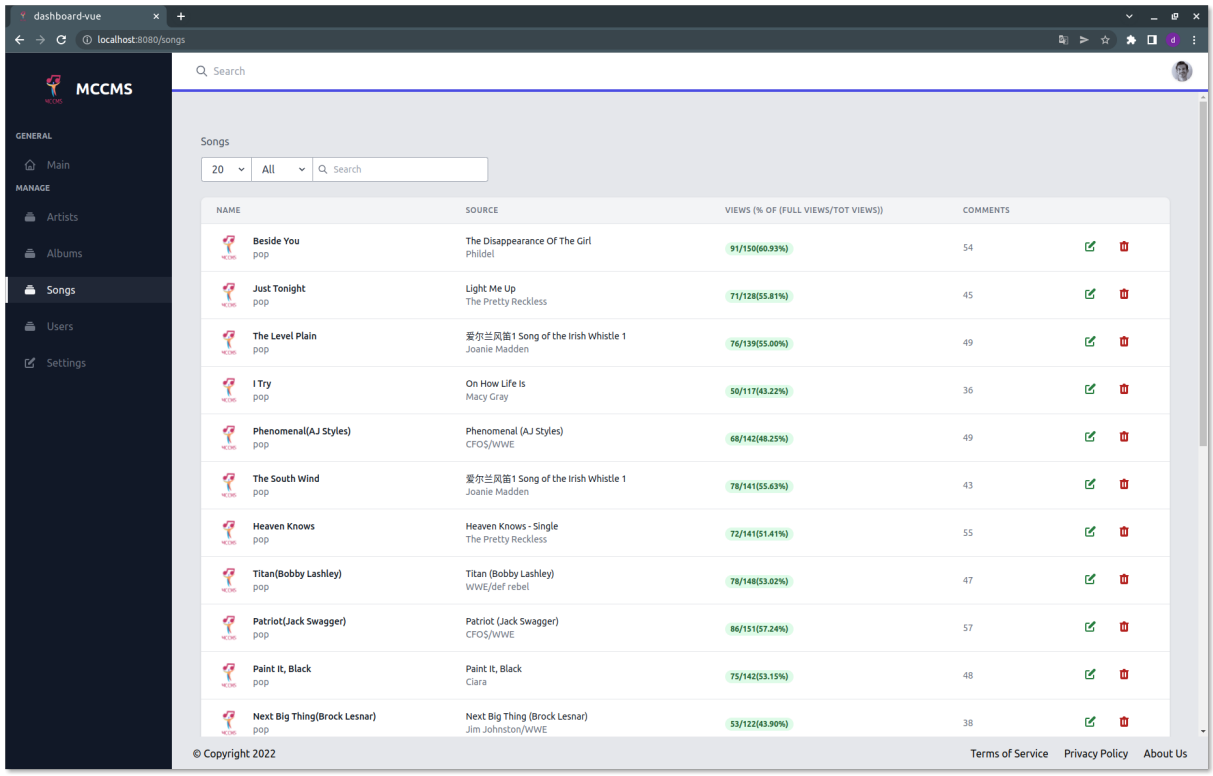
(a)


dashboard-vue
localhost:8080/dashboard
MCCMS
Search
GENERAL
Main
MANAGE
Artists
Albums
Songs
Users
Settings
12,247
Total Users
1,695
Available Artists
2,359
Available Songs
Songs
20
All
Search
NAME
SOURCE
VIEWS (% OF (FULL VIEWS/TOT VIEWS))
COMMENTS
Beside You
pop
The Disappearance Of The Girl
Phildel
91/150(60.93%)
54
Just Tonight
pop
Light Me Up
The Pretty Reckless
71/128(55.81%)
45
The Level Plain
pop
爱尔兰风笛1 Song of the Irish Whistle 1
Joanie Madden
76/139(55.00%)
49
I Try
pop
On How Life Is
Macy Gray
50/117(43.22%)
36
Phenomenal(AJ Styles)
pop
Phenomenal (AJ Styles)
CFOS/WWE
68/142(48.25%)
49
The South Wind
pop
爱尔兰风笛1 Song of the Irish Whistle 1
Joanie Madden
78/141(55.63%)
43
Heaven Knows
pop
Heaven Knows - Single
The Pretty Reckless
72/141(51.41%)
55
Titan(Bobby Lashley)
pop
Titan (Bobby Lashley)
WWE/def rebel
78/148(53.02%)
47
Patriot(Jack Swagger)
pop
Patriot (Jack Swagger)
CFOS/WWE
86/151(57.24%)
57
© Copyright 2022
Terms of Service
Privacy Policy
About Us

(b)



(c)



(d)

Figure 20: Sample Web Pages

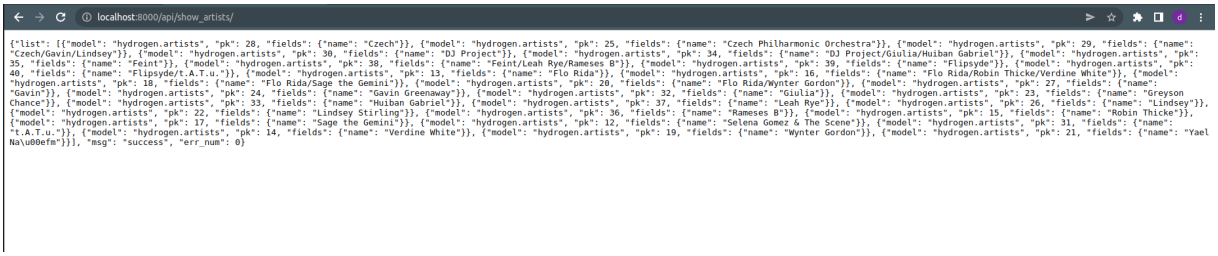


Figure 21: Sample API (testify the front-end connection with the back-end)