Music Management System - MusePlay

4^{TH} SEMESTER DBMS PROJECT

Course code: UE17CS252

To:

Professor Vinay Joshi

By:

Anirudh Maiya	PES1201700170
Madhav Mahesh Kashyap	PES1201700227
Sangam Kedilaya	PES1201701139

Section 4H

Department of Computer Science

PES University, Bangalore

4/26/2019

MusePlay

DBMS Project

ANIRUDH MAIYA

MADHAV MAHESH KASHYAP

SANGAM KEDILIYA

INTRODUCTION

Music is a vital part of daily living, as Albert Einstein stated, “life without playing music would be inconceivable”. There is no one living in this earth who does not listen to any kind of music.

Music has evolved from the days of cassettes and Vinyl players. Nowadays, with the rise of online communities and social media websites like YouTube and Spotify. These sites make it easier for aspiring singers and amateur and professional bands to distribute videos of their songs, connect with other musicians, and gain audience interest.

Our goal is to provide a free and open platform that connect the music Consumers with the Artists directly without much interference from Publishing companies.

TARGET AUDIENCE

Our aim is to include a platform that includes all the components of the music industry: Consumers, Artists and Record Publishing Companies. The best experience is seen only when these 3 parts are included.

- Music Consumers
  - Already well acquainted with their Music taste
  - Who want to get introduced to new Music
- Artists
  - Amateur bands
  - Professional bands
  - Individual artists
- Record Labels

MINIWORLD

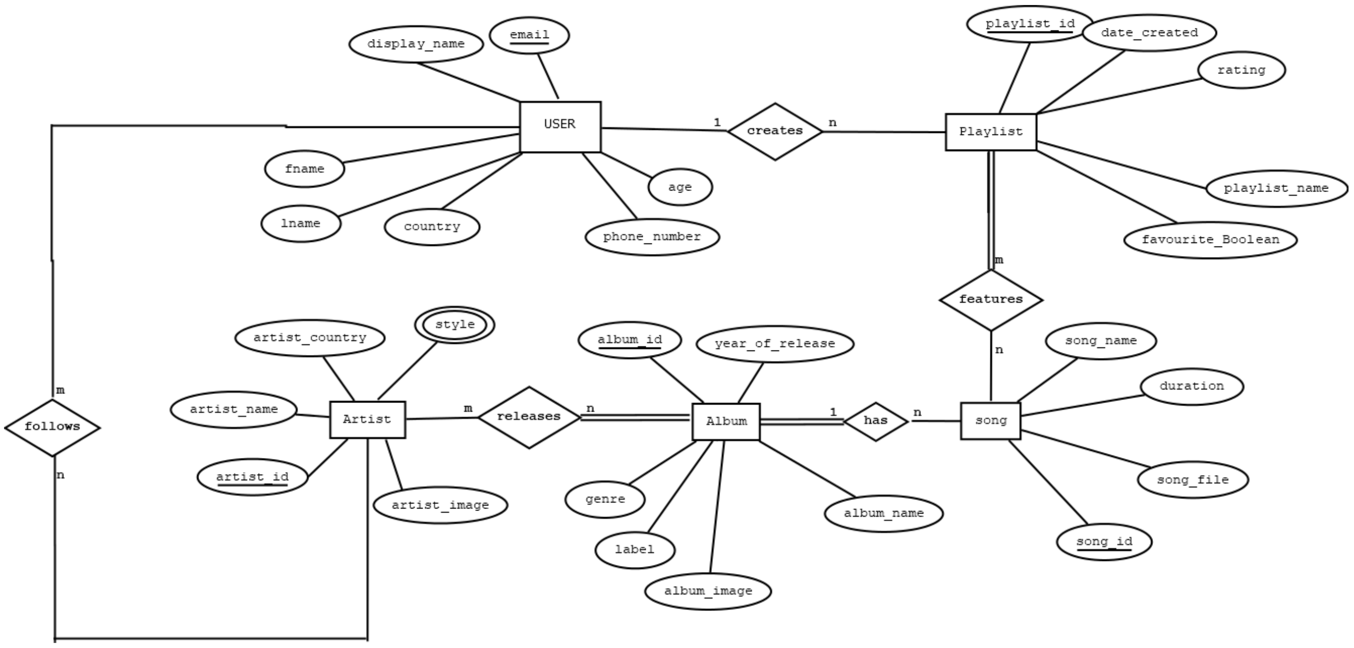
- The miniworld consists of Users, Playlists , Songs , Album and Artists.
- Each contributing a significant part to the MusePlay app.
- User can create Playlist and can also play the songs present in the Playlist.
- The playlist features the songs.
- Artist releases the albums and user follows the artist.
- The user has some user details on which the songs are recommended to him.
- Artist has a specific style and Album has a specific genre.
- The user is identified by his email id and has a unique username.
- The details of the user are stored and is used to recommend new songs to him.

- Based on users’ artist following habit new artists are recommended to him based on his following artists.
- User can be up to date on what his part of the world is currently hearing and outside his country also.

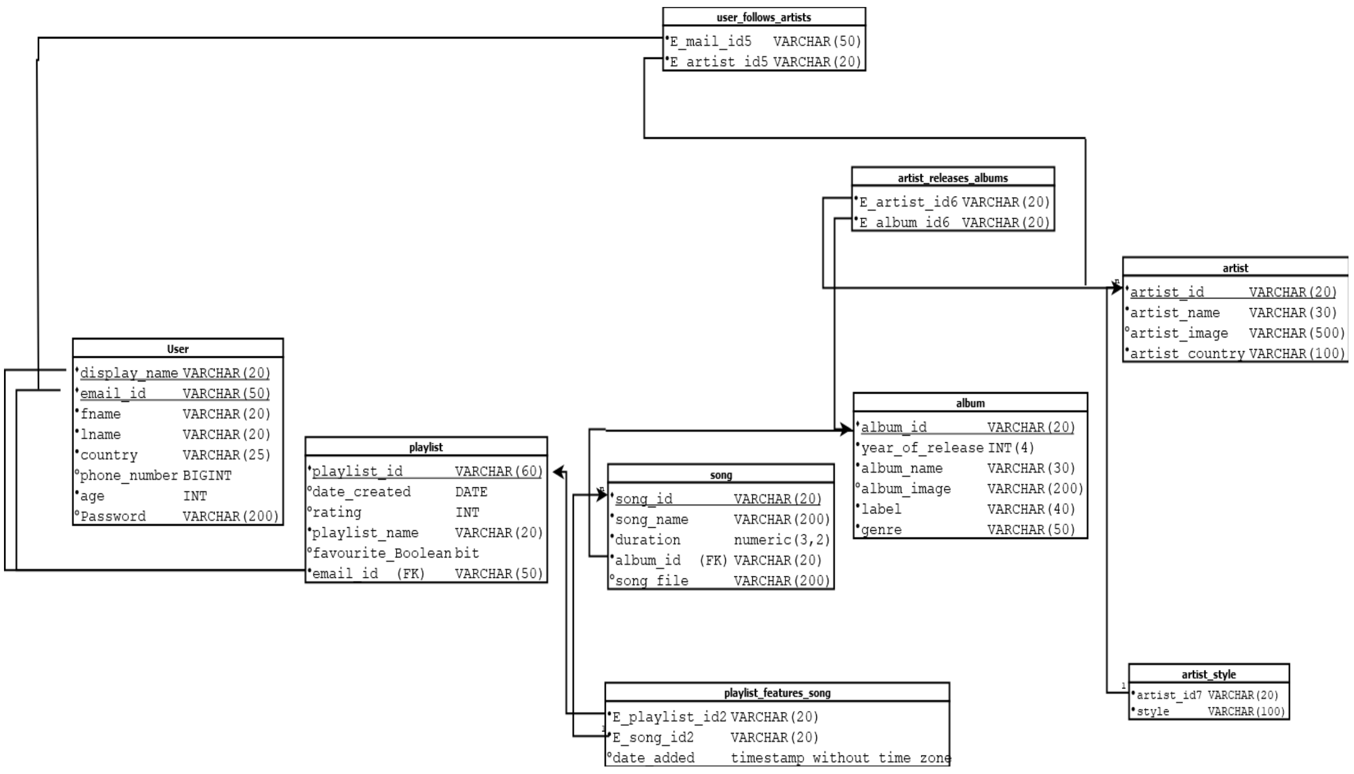
REQUIREMENTS

1. User should be able to login to his account.
2. If User has not created a previous account, then they should be able to Sign Up for a new account.
3. User should be able to create Playlists with custom names.
4. User should be able to browse through the Songs in their Playlist.
5. He should be able to add songs to the Playlist of his choice
6. Delete songs from their Playlist.
7. Delete the complete Playlist.
8. Playlist should be displayed sorted by Time Added.
9. User should be able to make a playlist his “Favourite” or remove from “Favourites”.
10. Search bar for Songs.
11. Search bar for Artists.
12. User should be able to view an Artist’s Biography and details.
13. User should be able to view the Album’s details.
14. User should be able to follow Artists.
15. Top 20 local songs
16. Top 20 global
17. User must get recommendation based on his love towards some genre.
18. Recommends artists based on the artists he follows.
19. User should be able to view the top artists on the website.

ER DIAGRAM



SCHEMA



user_follows_artists	
•E_mail_id5	VARCHAR(50)
•E_artist_id5	VARCHAR(20)

artist_releases_albums	
•E_artist_id6	VARCHAR(20)
•E_album_id6	VARCHAR(20)

artist	
•artist_id	VARCHAR(20)
•artist_name	VARCHAR(30)
°artist_image	VARCHAR(500)
•artist_country	VARCHAR(100)

User	
•display_name	VARCHAR(20)
•email_id	VARCHAR(50)
•fname	VARCHAR(20)
•lname	VARCHAR(20)
•country	VARCHAR(25)
°phone_number	BIGINT
•age	INT
°Password	VARCHAR(200)

album	
•album_id	VARCHAR(20)
•year_of_release	INT(4)
•album_name	VARCHAR(30)
°album_image	VARCHAR(200)
•label	VARCHAR(40)
•genre	VARCHAR(50)

playlist	
•playlist_id	VARCHAR(60)
°date_created	DATE
°rating	INT
•playlist_name	VARCHAR(20)
°favourite_Boolean	bit
•email_id (FK)	VARCHAR(50)

song	
•song_id	VARCHAR(20)
•song_name	VARCHAR(200)
•duration	numeric(3,2)
•album_id (FK)	VARCHAR(20)
°song_file	VARCHAR(200)

playlist_features_song	
•E_playlist_id2	VARCHAR(20)
•E_song_id2	VARCHAR(20)
°date_added	timestamp without time zone

artist_style	
•artist_id7	VARCHAR(20)
•style	VARCHAR(100)

CREATE SCRIPTS

```
CREATE DATABASE spotify;

\c spotify

CREATE TABLE userg

 (        fname VARCHAR(20) NOT NULL ,

          lname VARCHAR(20)NOT NULL,

          display_name VARCHAR(20) NOT NULL,

          email_id VARCHAR(50) NOT NULL,

          country VARCHAR(25),

          phone_number BIGINT,

          age INT NOT NULL,

          password VARCHAR(200) NOT NULL,



PRIMARY KEY (email_id),

UNIQUE(display_name)    );


CREATE TABLE playlist

 (        playlist_id VARCHAR(60) NOT NULL,

          date_created DATE,

          rating INT,

          playlist_name VARCHAR(20) NOT NULL,

          favourite_Boolean bit,

          email_idfk VARCHAR(50) NOT NULL,
        PRIMARY KEY (playlist_id),
        FOREIGN KEY(email_idfk) REFERENCES userg(email_id)
);

CREATE TABLE album
(       album_id VARCHAR(20) NOT NULL,
        year_of_release INT NOT NULL,
        album_name VARCHAR(80) NOT NULL,
        album_image varchar(200),
        label VARCHAR(40) NOT NULL,
        genre VARCHAR(50) NOT NULL,
        PRIMARY KEY (album_id)
);

CREATE TABLE artist
(       artist_id VARCHAR(20) NOT NULL,
        artist_name VARCHAR(30) NOT NULL,
        artist_image VARCHAR(500),
        artist_country VARCHAR(100) NOT NULL,
        PRIMARY KEY (artist_id)
);

CREATE TABLE artist_releases_albums
(
        E_artist_id6 VARCHAR(20) NOT NULL,
        E_album_id6 VARCHAR(20) NOT NULL,
        FOREIGN KEY (E_artist_id6) REFERENCES artist(artist_id),
        FOREIGN KEY (E_album_id6) REFERENCES album(album_id)
);

CREATE TABLE song
(       song_id VARCHAR(20) NOT NULL,
        song_name VARCHAR(200) NOT NULL,
        duration DECIMAL(3,2) NOT NULL,
        label VARCHAR(80) NOT NULL,
        album_idfk VARCHAR(20) NOT NULL,
        song_file VARCHAR(200) NOT NULL,
        PRIMARY KEY (song_id),
        FOREIGN KEY(album_idfk) REFERENCES album(album_id)
);

CREATE TABLE artist_style
(       artist_id7 VARCHAR(20) NOT NULL,
        style VARCHAR(100) NOT NULL,
        FOREIGN KEY (artist_id7) REFERENCES artist(artist_id)
);

CREATE TABLE playlist_features_song
(
        E_playlist_id2 VARCHAR(20) NOT NULL,
        E_song_id2 VARCHAR(20) NOT NULL,
        date_added timestamp without time zone,
          FOREIGN KEY(E_playlist_id2) REFERENCES playlist(playlist_id),

          FOREIGN KEY(E_song_id2) REFERENCES song(song_id)

);


CREATE TABLE user_follows_artist

(         E_mail_id5 VARCHAR(50) ,

          E_artist_id5 VARCHAR(20),

          FOREIGN KEY (E_mail_id5) REFERENCES userg(email_id),

          FOREIGN KEY (E_artist_id5) REFERENCES artist(artist_id)

);
```

SQL QUERIES

For searching song and the artist:

```
" Select song_name,song_id,album_idfk,artist_name,artist_id from
song,album,artist,artist_releases_albums where  song.album_idfk = album.album_id
and E_artist_id6 = artist_id and E_album_id6  = album_id and song_name like
'%$searchq%' "
```

For searching only the artist:

```
“Select artist_name,artist_image from artist where artist_name like '%$searchq%' “
```

For displaying artist details in an artist’s page:

```
"SELECT artist_name,artist_image,artist_country,album_nameFROM
artist,album,artist_releases_albums WHERE E_album_id6  = album_id and
E_artist_id6 = artist_id and artist_id = 'cla1';"
```

For displaying number of followers' artist has:

```
"SELECT artist_name,COUNT(artist_name) as Most_FollowedFROM
userg,artist,user_follows_artist WHERE userg.email_id =
user_follows_artist.E_mail_id5 and artist.artist_id = user_follows_artist.E_artist_id5
and artist_id= 'cla1'

GROUP BY (artist_name);"
```

For displaying the style of the artist:

```
SELECT style FROM artist,artist_style WHERE artist.artist_id = artist_style.artist_id7
and artist_id = 'cla1';";
```

For displaying number of albums artist has:

```
SELECT artist_name,COUNT(album_name) FROM   artist, album,
artist_releases_albums WHERE E_album_id6  = album_id and E_artist_id6 =
artist_id and artist_id = 'cla1'

GROUP BY (artist_name);"
```

For displaying all the albums the artist has:

```
"SELECT album_name,album_image,album_id FROM album, artist,
artist_releases_albums WHERE E_album_id6  = album_id and E_artist_id6 =
artist_id and artist_id = 'cla1';"
```

For displaying all the songs present in the album:

```
"SELECT song_name,song_file,duration,song_id FROM song, album,
artist_releases_albums where album_id = '$album_id' and song.album_idfk =
album.album_id and E_album_id6  = '$album_id'";
```

For displaying all the songs present in the users' playist:

```
"SELECT song_name, artist_name, song_file, artist_id, album_id, date_added,
song_id FROM song, artist, playlist_features_song, playlist, userg, album,
artist_releases_albums WHERE

song.song_id = playlist_features_song.E_song_id2 and

playlist.playlist_id = playlist_features_song.E_playlist_id2 and

song.album_idfk = album.album_id and

artist_releases_albums.E_album_id6 = album.album_id and

 E_artist_id6 = artist_id and

email_id = '$_SESSION[email_id]' and

userg.email_id = playlist.email_idfk and

playlist_id = '$playlist_id'

ORDER BY (date_added) ASC")
```

For displaying the bit value Favorite and rating :

```
" SELECT playlist_name,date_created,rating,favourite_boolean from playlist where
playlist_id = '$playlist_id' "
```

For deleting the song from playlist:

```
“ DELETE FROM playlist_features_song where e_playlist_id2 = '$playlist_id' and
e_song_id2 = '$song_id' and date_added = '$date_added' "
```

For deleting complete playlist:

```
“ DELETE FROM playlist where playlist_id = '$playlist_id' “
```

For displaying the top 10 most followed artists:

```
"SELECT artist_name, artist_image, COUNT(artist_name) as Most_Followed

FROM userg, artist,user_follows_artist WHERE

userg.email_id = user_follows_artist.E_mail_id5 and

artist.artist_id = user_follows_artist.E_artist_id5

GROUP BY (artist_name,artist_image)

ORDER BY COUNT(*) DESC

LIMIT 10"
```

MORE COMPLEX SQL QUERIES:

For displaying the top 20 songs heard from users from the present users' country :

ALGORITHM:

1)SELECTING PRESENT USERS' COUNTRY

2)SELECTING ALL THE SONGS FROM THE PLAYLIST OF ALL THE OTHER USERS' FROM THE SAME COUNTRY

```
" Select song_name, COUNT(song_name), song_file, song_id, album_id, artist_name,
artist_id

from song,album, playlist, playlist_features_song, userg, artist_releases_albums, artist

where userg.country = '$row[4]' and email_idfk = email_id and song.song_id =
playlist_features_song.E_song_id2 and

 playlist.playlist_id = playlist_features_song.E_playlist_id2 and

song.album_idfk = album.album_id and

artist_releases_albums.E_album_id6 = album.album_id and
artist_releases_albums.E_artist_id6 = artist.artist_id

GROUP BY (song_name,song_file,song_id,album_id,artist_name,artist_id)

ORDER BY COUNT(*) DESC"
```

For displaying the top 20 songs heard from users all over the world:

```
“Select song_name, COUNT(song_name), song_file, song_id, album_id, artist_name, artist_id

from song,album,playlist,playlist_features_song,artist_releases_albums,artist

where song.song_id = playlist_features_song.E_song_id2 and

playlist.playlist_id = playlist_features_song.E_playlist_id2 and

song.album_idfk = album.album_id and

artist_releases_albums.E_album_id6 = album.album_id and

E_artist_id6 = artist_id

GROUP BY (song_name,song_file,song_id,album_id,artist_name,artist_id)

ORDER BY COUNT(*) DESC

LIMIT 20"
```

For recommending songs to user which he hasn’t heard and is similar to songs in his playlist:

ALGORITHM:

1) SELECTING THE TOP 5 GENRE FROM USERS PLAYLIST AND CREATING A DYNAMIC TABLE CALLED top_ten.
2) CREATING DYNAMIC TABLES CALLED pop0, pop1, pop2, pop3, pop4 for each of the 5 GENRES’.
3) TAKING THE DIFFERENCE OF SONGS IN USERS’ PLAYLIST AND SONGS PRESENT IN pop0, pop1, pop2, pop3, pop4 and adding them back to these 5 tables.
4) ADDING ALL OF THE TABLES IN STEP 4 TO ONE DYNAMIC TABLE CALLED pop OF THE SAME TYPE.
5) DROPPING THE DYNAMIC TABLES IN STEP 3.
6) SELECTING THE SONGS RANDOMLY FROM DYNAMIC TABLE POP (30).

```
“SELECT genre, COUNT(genre) INTO top_ten

FROM song,artist, playlist_features_song, playlist, userg, album,
artist_releases_albums

WHERE song.song_id = playlist_features_song.E_song_id2 and

playlist.playlist_id = playlist_features_song.E_playlist_id2 and

song.album_idfk = album.album_id and

artist_releases_albums.E_album_id6 = album.album_id and

E_artist_id6 = artist_id and

email_id = '$_SESSION[email_id]' and

userg.email_id = playlist.email_idfk

group by (genre)

ORDER BY COUNT(*) DESC

LIMIT 5"


"SELECT song_name, artist_name, song_id, album_id, genre, song_file, artist_id
INTO $string

FROM song,artist,album,artist_releases_albums

where song.album_idfk = album.album_id and

E_artist_id6 = artist_id and

E_album_id6  = album_id and

album.genre = '$row6[0]'

EXCEPT

SELECT song_name,artist_name,song_id,album_id,genre,song_file,artist_id

FROM song,artist,playlist_features_song,playlist,userg,album,artist_releases_albums

WHERE userg.email_id = '$_SESSION[email_id]' and

song.song_id = playlist_features_song.E_song_id2 and

playlist.playlist_id = playlist_features_song.E_playlist_id2 and

song.album_idfk = album.album_id and
```

```
artist_releases_albums.E_album_id6 = album.album_id and
E_artist_id6 = artist_id and userg.email_id = playlist.email_idfk and
album.genre = '$row6[0]'
order by (artist_name)"
```

For recommending artists similar to what artists the user follows:

ALGORITHM:

1) CREATING A TABLE CALLED user_doesnt_follow TO GET ALL THE ARTISTS THE CURRENT USER DOESN'T FOLLOW.
2) GETTING ALL THE STYLE OF THE ARTISTS THE USER FOLLOWS AND INSERTING THEM INTO DYNAMIC TABLE xyz.
3) SO NOW WE CAN SELECT THE ARTISTS,STYLE FROM user_doesnt_follow AND SELECT STYLE FROM xyz AND GET THE SIMILAR ARTISTS (DISTINCT).
4) DROPPING BOTH THE TABLES.

```
"select style, artist_id7 into user_doesnt_follow
from artist_style
except
select style,artist_id7
from user_follows_artist,artist_style
where e_artist_id5 = artist_id7 and e_mail_id5 = '$email_id' "
```

```
"select style into xyz
from user_follows_artist,artist_style
where e_artist_id5 = artist_id7 and e_mail_id5 = '$email_id'
order BY RANDOM()"
```

```
“ select distinct artist_id7
from user_doesnt_follow,xyz
where user_doesnt_follow.style = xyz.style “
```

WE NEED PHP CODE TO EXPLAIN THE DYNAMIC DISPLAY OF BUTTON WHEN THE USER FOLLOWS ARTIST:

```
$query = pg_query("SELECT E_artist_id5
                        FROM user_follows_artist
                        WHERE E_mail_id5='$email_id' AND E_artist_id5='$artist_id' ");
 if(!(pg_num_rows($query)>=1)){
pg_query("INSERT INTO user_follows_artist(E_mail_id5, E_artist_id5)
               VALUES ('$email_id', '$artist_id') ");
}
//end of following


$query = pg_query("SELECT E_artist_id5
                           FROM user_follows_artist
                           WHERE E_mail_id5='$email_id' AND
                            E_artist_id5='$artist_id' ");
if(pg_num_rows($query)>=1 ){
                        pg_query("DELETE FROM user_follows_artist
                                      WHERE E_mail_id5='$email_id' AND
                                      E_artist_id5='$artist_id' ");
}
//end of unfollowing
//similar way is used for like and unliking of playlist.
```

TEST CASES

- 2 users cannot have the same email ID, since email ID is the primary key.
  This rejects the user registration and prevents multiple fake users.
- User cannot add songs to an Unknown playlist. In case this happens, an error massage is displayed.
- If an unknown song or artist is searched, then no rows are displayed to prevent accidental addition to playlist.

CONCLUSION

In conclusion, MusePlay is a very flexible and robust platform. It allows for easy and convenient Music searching and listening. The UI is ergonomic and simple and allows for a pleasant viewing experience.It can be used by all types of users: from the experienced Music lovers, to the novice listeners, from beginner artists to professional artists to mega record labels.

The project satisfies all the requirements and is a great Music listening and discovering application.