Table of Contents

CONTENTS	PAGE NO
ACKNOWLEDGEMENT	I
ABSTRACT	II
1. INTRODUCTION	1
1.1 Overview of Database Management System	1
1.2 Problem Statement	1
1.3 Objectives	1
1.4 Description of the dataset	2
2. SYSTEM REQUIREMENTS	3
2.1 Software Requirements	3
2.1.1 Front End	3
2.1.2 Back End	3
2.2 Hardware Requirements	3
3. SYSTEM DESIGN	4
3.1 E-R Diagram	4
3.2 Schema Diagram	5
3.3 Overview of GUI	6
3.4 Normalization	8
4. IMPLEMENTATION	12
4.1 Table Creation	12
4.2 Description of Table	13
4.3 Populated Tables	15
4.4 SQL Triggers and Stored Procedure	17
4.5 Database connectivity	18
5. RESULTS	21
CONCLUSION	24
FUTURE ENHANCEMENT	25

List of Figures

Figure No	Figure Name	Page No
3.1	Entity-Relationship diagram	4
3.2	Schema diagram	5
3.3	Submit Button	6
3.4	Navigation Bar	7
3.5	Dropdown Menu	7
4.1	Login table description	13
4.2	Songs table description	14
4.3	Singers table description	14
4.4	Movies table description	14
4.5	Trending table description	15
4.6	Playlist table description	15
4.7	Login table populated	15
4.8	Singers table populated	15
4.9	Songs table populated	16
4.10	Movies table populated	16
4.11	Playlist table populated	16
4.12	Trending table populated	17
4.13	Trigger on Movie table	17
4.14	Stored Procedure on Login table	20
5.1	Admin Homepage	21
5.2	User Homepage	21
5.3	Login page for Admin and User	22
5.4	Signup page for Admin and User	22
5.5	Insertion	23
5.6	Updating and Deletion	23

List of Tables

Table no	Table name	Page no
3.1	Song Table	8
3.2	Song Table in 1NF	8
3.3	Song Table in 2NF	9
3.4	Song Table in 3NF	10

CHAPTER 1

INTRODUCTION

1.1 Overview of Database Management Systems

A Database is a collection of related data organized in a way that data can be easily accessed, managed and updated. Any piece of information can be a data. Database is actually a place where related piece of information is stored and various operations can be performed on it. A database management system (DBMS) is software that allows creation, definition and manipulation of database. DBMS is actually a tool used to perform any kind of operation on data in database. DBMS also provides protection and security to database. It maintains data consistency in case of multiple users. Here are some examples of popular DBMS: MySQL, Oracle, Sybase, Microsoft Access and IBM DB2 etc.

A DBMS makes it possible for end users to create, read, update and delete data in a database. The DBMS essentially serves as an interface between the database and the end users or application programs, ensuring the data is completely organized and remains easily accessible. Ex: A real estate database stores information of different types of properties and the preferences of the owner regarding the tenant.

The information stored in the database can be accessed by registered or valid users upon login. The users can also update, retrieve or insert data into the database.

1.2 Problem Statement

Everyday hundreds of songs of different genre and language by various artists are being released across the world, distributing these songs in physical form like CD's is financially not feasible to artists. This application provides an easy way for singers to release music and an easy way to listen to music.

1.3 Objectives

- To create a user-friendly interface as the front end of the database
- To provide wide variety of songs to listen
- To provide an easy method for singers to distribute
- To provide lag free listening experience
- To curate playlists for the users based on their listening habits

1.4 Description of the Dataset

- An entity called **LogIN** is created with the **EmailID** being the Main Key Attribute also known as the Primary key. The other Attributes belonging to the Entity are **First Name, Last Name, DOB, Country, Gender**, **Password and UserType.**
- The Entity **SONGS** is created with the **Song_ID** being the Primary key. The other attributes are **Song_Name, Language** and **Genre.** The attributes **Singer_ID** and **Movie_ID** references from **SINGERS** and **MOVIES** respectively.The other Attributes belonging to this Entity are **Director**, **Actor**, **Actress**, **Lyricist**, **Producer** and **Distributer**.
- The Entity **SINGERS** consists of the **Singer_ID** acting as the primary key, **Singer_Name, DOB, Gender** and **Country**.
- The Entity **MOVIES** consists of **Movie_ID** as the primary key. It also involves other attributes such as **Director, Actor, Actress and Language.**
- The Entity **TRENDING** consists of **Sl_No** and **Song_ID.**
- The Entity **PLAYLISTS** consists of **Playlist_ID** as the primary key. It also contains **Song_ID.**

CHAPTER 2

SYSTEM REQUIREMENTS

2.1 Software Requirements

The Software Requirement deal with defining software resource requirements and prerequisites that needs to be installed on a computer to provide optimal functioning of an application.

2.1.1 Front End

- User interface : HTML/CSS/JavaScript
- Operating System : Microsoft Windows 7 or above
- Web Browser : Chrome, Internet Explorer

2.1.1 Back End

- Programming language : Java / J2EE,JavaScript
- Database : My SQL
- Application Server : XAMPP server(v3.2.2) for Apache server (localhost)
- PHP (v7.2.10) for server side scripting
- Sublime3 (Source Code Editor)

2.2 Hardware Requirements

The software should run on any sort of desktop or laptop environment, regardless of the operating system. Essential input/output devices are keyboards, mouse, and printers; nothing else is required but can be recommended if desired.

- Processor : Pentium IV and above
- Hard Disk: 100 GB
- RAM : 2 GB or above
- Display Resolution : 1366 x 768 (1920 x 1080 Recommended)
- Other standard physical devices like keyboard, mouse etc

CHAPTER 3

SYSTEM DESIGN

System designs the process of defining the architecture, modules, interfaces and data for a system to satisfy specific requirements. System design could be seen as the application of system theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering.

3.1 Entity – Relationship Diagram

An Entity Relationship model, also called as Entity – Relationship (ER) Diagram, is a graphical representation of entities and their relationship to each other, typically used in computing in regard to the organization of data within databases or information systems.

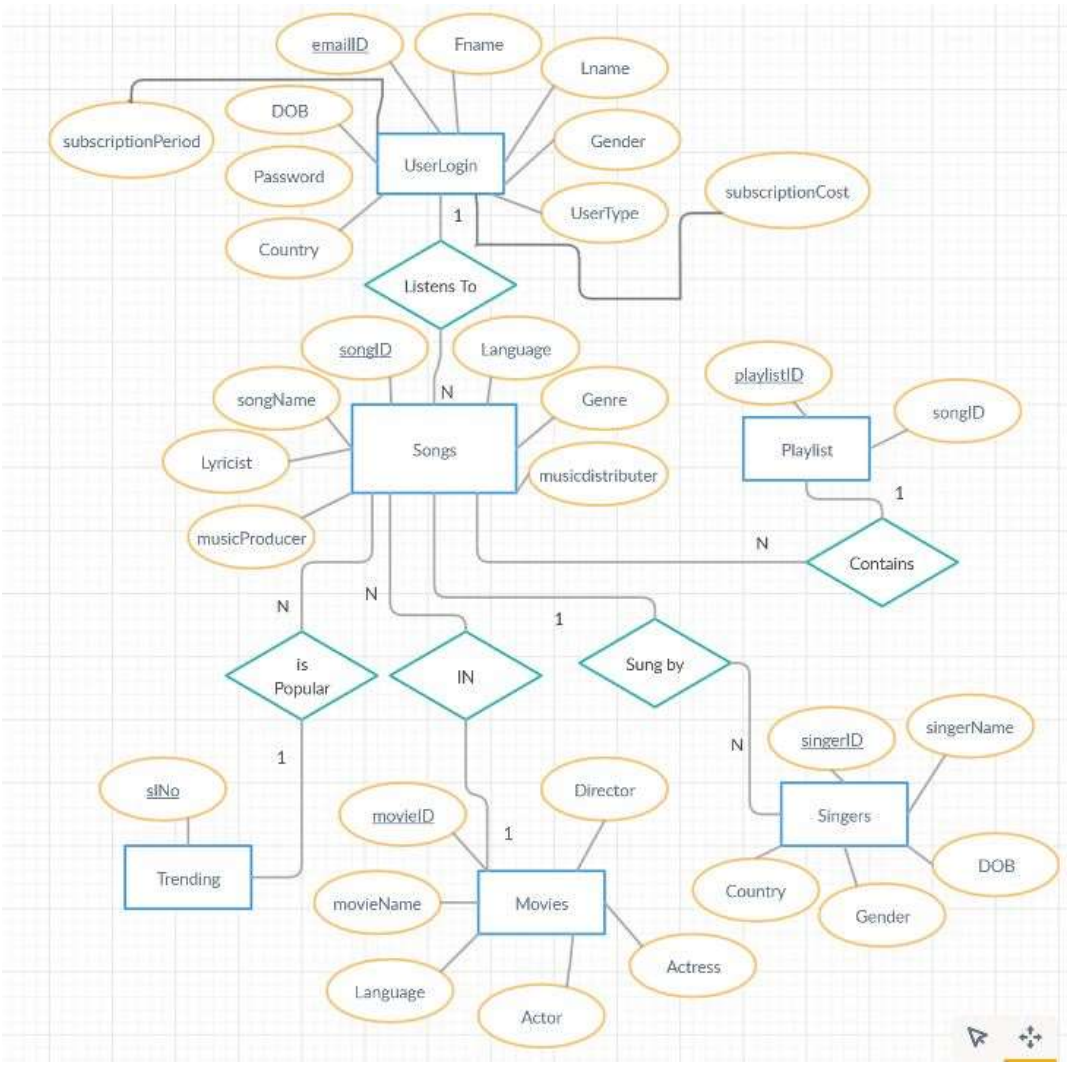

Fig 3.1 ER Diagram

3.2 Schema Database Relationship Diagram

A Database schema is the skeleton structure that represents the logical view of the entire database. It formulates all the constraints that are to be applied on the data. A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams.

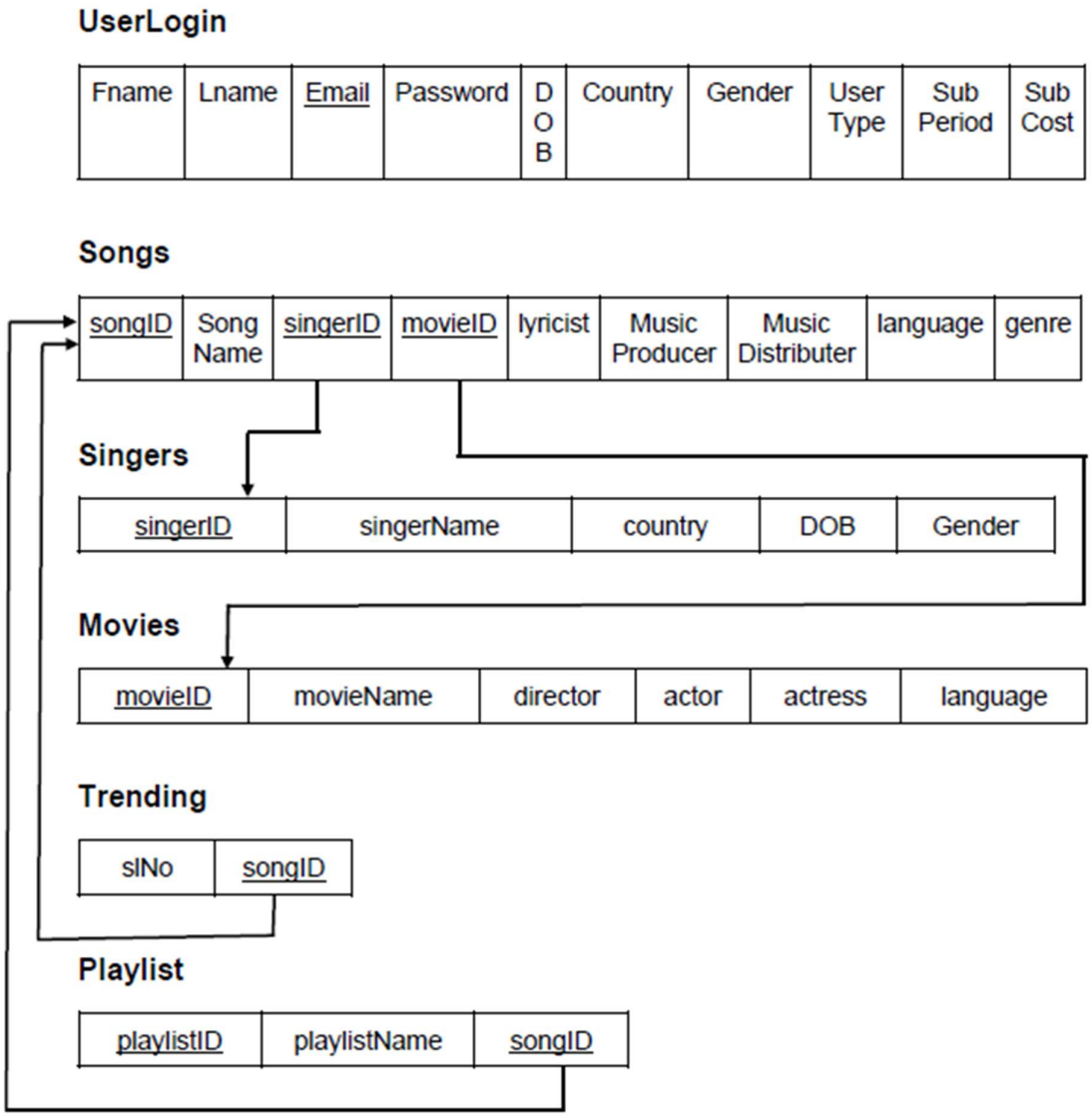

Fig. 3.2 Schema Database Relationship Diagram

3.3 Overview of GUI

The **graphical user interface** (**GUI**), is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, instead of text-based user interfaces, typed command labels or text navigation.

The actions in a GUI are usually performed through direct manipulation of the graphical elements. Beyond computers, GUI's are used in many handheld mobile devices such as MP3 players, portable media players, gaming devices, smartphones and smaller household, office and industrial controls. The term GUI tends not to be applied to other lower-display resolution types of interfaces, such as video games, or not including flat screens able to describe generic information, in the tradition of the computer science research at the Xerox Palo Alto Research Centre.

The following buttons have been used in my project:

1. Button

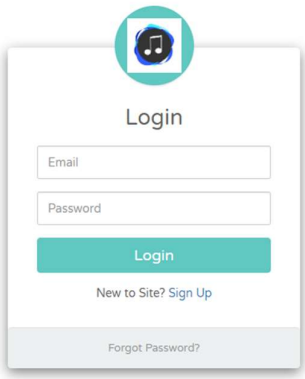

Figure 3.3 Button

2. Navigation Bar

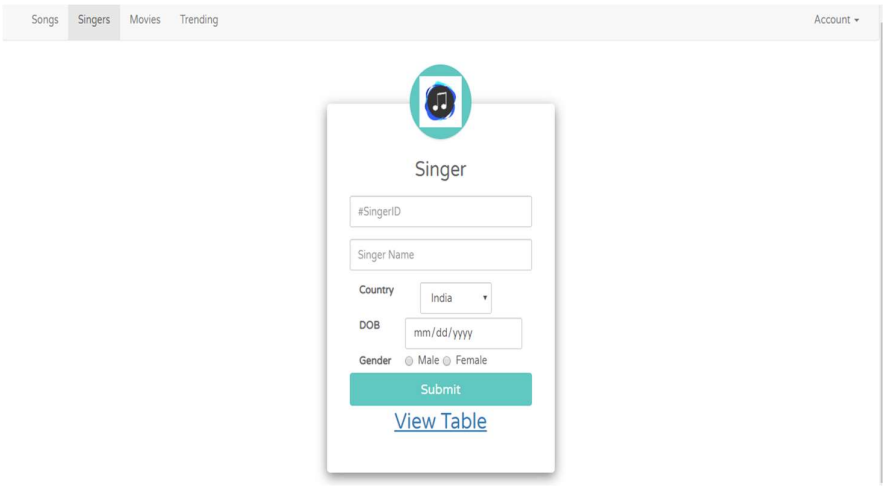

Figure 3.4 Navigation Bar

3.Dropdown

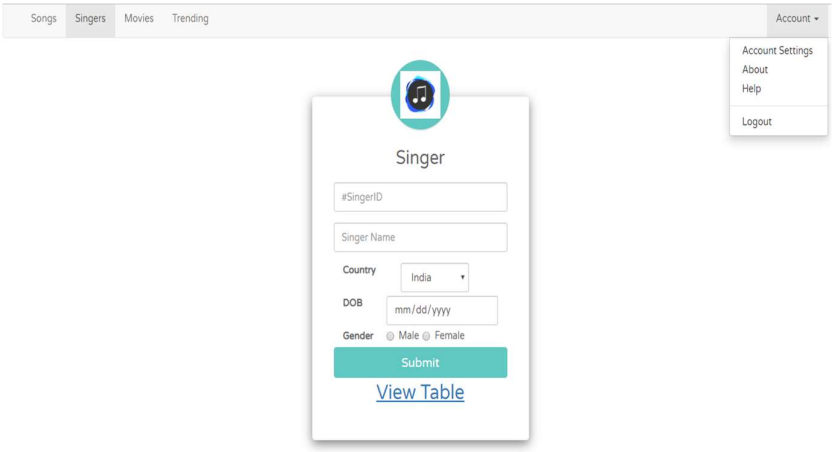

Figure 3.5 Dropdown

3.4 Normalization

Normalization is the process of analyzing the given relation schema based on their functional dependencies and primary key to achieve desirable properties of minimizing redundancy and minimizing insert, delete, update anomaly. The normalization process takes a relation schema through a series of tests to certify whether it satisfies a certain normal form. The normal form of a relation refers to the highest normal form condition that it meets, and hence the degree to which it has been normalized.

There are two goals of the normalization process: eliminating redundant data (for example, storing the same data in more than one table) and ensuring data dependencies make sense (only storing related data in a table). Both of these are worthy goals as they reduce the amount of space a database consumes and ensure that data is logically stored.

Assume a music database having an entity song. Without any normalization, all the information stored in one table is as shown below:

songID	songName	singerID	movieID	lyricicst
1	Aal Izz Well	501	1001	Sonu Nigam, Pritam
2	Salaam Rocky Bhai	503,504	1003	Naresh Iyer
3	Dhoom Machale Dhoom	505	1002	Sonu Nigam, Naresh Iyer

Table 3.1 Song Table

3.4.1 First Normal Form(1NF)

This is the simplest stage of normalization and involves making sure that each individual field within the table can hold only one piece of data and no repeating groups, there should not be multi-valued attributes and the relations should be simple and atomic.

1NF Rules:

- Each table cell should contain a single value
- Each record needs to be unique.

The above table 3.1 in 1NF:.

songID	songName	singerID	movieID	lyricist
1	Aal Izz Well	501	1001	Sonu Nigam
1	Aal Izz Well	501	1001	Pritam

3	Dhoom Machale Dhoom	505	1002	Sonu Nigam
3	Dhoom Machale Dhoom	505	1002	Naresh Iyer
2	Salaam Rocky Bhai	503	1003	Naresh Iyer
2	Salaam Rocky Bhai	504	1003	Naresh Iyer

Table 3.2 Song Table in 1NF

In the above table 3.1 lyricist column in not unique as it has two entries in the same row and after normalization each row has only one entry I lyricist column.

3.4.2 Second Normal Form(2NF)

This normalization is used when the primary key is dependent upon more than one field, if another field within that table is only dependent on a part of the key unnecessary redundancies could result. In the second normal form a non-key attribute of an entity must depend on the entire primary key. In second normal form every non-prime attribute should be functionally dependent on prime key attribute. That is, if X → A holds, then there should not be any proper subset of Y on X, for which Y→A also holds true.

2NF Rules:

- The Table should be in 1NF.
- The Primary key of the table should compose of exactly one column.

songID	**songName**	**singerID**	**movieID**
1	Aal Izz Well	501	1001
1	Aal Izz Well	501	1001
3	Dhoom Machale Dhoom	505	1002
3	Dhoom Machale Dhoom	505	1002
2	Salaam Rocky Bhai	503	1003
2	Salaam Rocky Bhai	504	1003

songID	lyricist
1	Sonu Nigam
1	Pritam
2	Naresh Iyer
3	Sonu Nigam
3	Naresh Iyer

Table 3.3 Song table data in 2NF

3.4.3 Third Normal Form(3NF)

The table must contain no transitive determinants, meaning that all fields must be determined directly by the primary key. If this is not the case, then the transitive determinant field should be transferred to another table with the field that determines it, reducing redundancy further. For a relation to be in the Third Normal Form, it must be in Second Normal Form and the following must be satisfy – No non-prime attribute is transitively dependent on prime key attribute. For any non- trivial functional dependency, X → A, then either- X is a super key or, A is a prime attribute.

3NF Rules:

- The Table should be in 2NF
- There should not be any functional dependency

Third normal form avoids this by breaking this into separate tables:

songID	songName	singerID
1	Aal Izz Well	501
1	Aal Izz Well	501
3	Dhoom Machale Dhoom	505
3	Dhoom Machale Dhoom	505
2	Salaam Rocky Bhai	503
2	Salaam Rocky Bhai	504

songID	lyricist
1	Sonu Nigam
1	Pritam

2	Naresh Iyer
3	Sonu Nigam
3	Naresh Iyer

songID	**songName**	**movieID**
1	Aal Izz Well	1001
2	Dhoom Machale Dhoom	1002
3	Salaam Rocky Bhai	1003

Table 3.4 Song table in 3NF

CHAPTER 4

IMPLEMENTATIONS

4.1 Table Creation

LOGIN:

```
CREATE TABLE LOGIN(EMAILID VARCHAR2(20) PRIMARY KEY,
                   FNAME  VARCHAR2(20),
                   LNAME  VARCHAR2(20),
                   GENDER  VARCHAR2(20),
                   USERTYPE  VARCHAR2(20),
                   COUNTRY  VARCHAR2(20),
                   DOB  DATE,
                   PASSWORD VARCHAR2(20));
```

SONG:

```
CREATE TABLE SONG(SONGID NUMBER(10) PRIMARY KEY,
SONGNAME VARCHAR2(200),
                  LYRICIST VARCHAR2(20),
                  GENRE VARCHAR2(20),
                  MUSICPRODUCER VARCHAR2(20),
                  MUSICDISTRIBUTER VARCHAR2(20),
                  LANGUAGE VARCHAR2(20),
MOVIEID REFERENCES MOVIE(MOVIEID) ON DELETE CASCADE,
SINGERID REFERENCES SINGER(SINGERID) ON DELETE CASCADE);
```

SINGER:

```
CREATE TABLE SINGER(SINGERID NUMBER(10) PRIMARY KEY,
                    SINGERNAME VARCHAR(20),
                    GENDER CHAR(1), DOB DATE,
```

```
                                DOB DATE,
                                COUNTRY VARCHAR(20));
```

MOVIE:

```
CREATE TABLE MOVIE(MOVIEID NUMBER(10) PRIMARY KEY,
                                MOVIENAME VARCHAR2(20),
                                DIRECTOR VARCHAR2(15),
                                ACTOR VARCHAR2(15),
                                ACTRESS VARCHAR2(15),
                                LANGUAGE VARCHAR2(15));
```

TRENDING:

```
CREATE TABLE TRENDING(SLNO NUMBER(10) PRIMARY KEY,
SONGID REFERENCES SONG(SONGID) ON DELETE CASCADE);
```

PLAYLIST:

```
CREATE TABLE PLAYLIST(PLAYLISTID NUMBER(10),
                                PLAYLISTNAME VARCHAR2(20),
SONGID REFERENCES SONG(SONGID)ON DELETE CASCADE);
```

4.2 Description of Tables

LOGIN

DESC LOGIN;

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	Fname	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
2	Lname	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
3	email	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
4	password	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
5	DOB	date			No	None			Change Drop More
6	country	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
7	gender	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
8	userType	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More

Figure 4.1 LOGIN TABLE DESCRIPTION

SONG

DESC SONG;

Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
songID	int(30)			No	None			Change Drop More
songName	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
singerID	int(30)			No	None			Change Drop More
movieID	int(30)			No	None			Change Drop More
lyricist	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
musicProducer	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
musicDistributer	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
language	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
genre	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More

Figure 4.2 SONG TABLE DESCRIPTION

SINGER

DESC SINGER;

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	singerName	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
2	singerID	int(30)			No	None			Change Drop More
3	country	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
4	DOB	date			No	None			Change Drop More
5	Gender	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More

Figure 4.3 SINGER TABLE DESCRIPTION

MOVIE

DESC MOVIE;

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	movieName	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
2	movieID	int(30)			No	None			Change Drop More
3	director	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
4	actor	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
5	actress	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
6	language	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More

Figure 4.4 MOVIE TABLE DESCRIPTION

TRENDING

DESC TRENDING;

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	slno	int(30)			No	None			Change Drop More
2	songID	int(30)			No	None			Change Drop More

Figure 4.5 TRENDING TABLE DESCRIPTION

PLAYLIST

DESC PLAYLIST;

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	playlistName	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
2	playlistID	int(30)			No	None			Change Drop More
3	songID	int(30)			No	None			Change Drop More

Figure 4.6 PLAYLIST TABLE DESCRIPTION

4.3 Populated Tables

LOGIN

SELECT * FROM LOGIN;

Fname	Lname	email	password	DOB	country	gender	userType
Prashanth	Raju	admin@gmail.com	admin123	2019-11-03	India	Male	admin
Ravi	Mk	ravi@gmail.com	ravi123	1999-11-03	India	Male	user
Prasad	RC	prasad@gmail.com	prasad123	0000-00-00	2019-11-20	Male	user
preetham	kumar	p@gmail.com	kumar	2019-11-20	India	Male	admin

Figure 4.7 LOGIN TABLE POPULATED

SINGER

SELECT * FROM SINGER;

singerName	singerID	country	DOB	Gender
Ram	501	India	1985-05-08	Male
Anil	502	USA	1995-06-12	Male
Priya	503	UK	1992-02-20	Female
Murali	504	India	1970-05-12	Male
Shriya	505	UK	1994-05-25	Female

Figure 4.8 SINGER TABLE POPULATED

SONG

SELECT * FROM SONG;

songID	songName	singerID	movieID	lyricist	musicProducer	musicDistributer	language	genre
101	Malang	501	1001	Pritam	Aditya Chopra	YRF Music	Kannada	Pop
102	Kamli	502	1001	Pritam	Aditya Chopra	YRF Music	Kannada	Pop
103	Tu Hi Junoon	504	1001	Pritam	Aditya Chopra	YRF Music	Kannada	Classical
104	Dhoom Machale Dhoom	503	1001	Pritam	Aditya Chopra	YRF Music	Kannada	Jazz
105	Bande Hain Hum Uske	505	1001	Pritam	Aditya Chopra	YRF Music	Kannada	Classical
106	Aal Izz Well	502	1002	Sonu Nigam	Shantanu Moitra	T-Series	Hindi	EDM
107	Zoobi Doobi	501	1002	Sonu Nigam	Shantanu Moitra	T-Series	Hindi	Electronic
108	Behti Hawa Sa Tha Woh	504	1002	Naresh Iyer	Shantanu Moitra	T-Series	Hindi	Classical
109	Give Me Some Sunshine	504	1002	Naresh Iyer	Shantanu Moitra	T-Series	Hindi	Hip Hop
110	Jaane Nahin Denge Tujhe	502	1002	Naresh Iyer	Shantanu Moitra	T-Series	Hindi	Pop
111	Salaam Rocky Bhai	505	1003	Dr. V. Nagendra Prasad	Ravi Basrur	Lahari Music	Kannada	Jazz
112	Garbadhi	501	1003	Kinnal Raj	Ravi Basrur	Lahari Music	Kannada	Rock and Roll
113	Sidila Barava	501	1003	Ravi Basrur	Ravi Basrur	Lahari Music	Kannada	Electronic
114	Dheera Dheera	502	1003	Jayagopal	Ravi Basrur	Lahari Music	Kannada	Classical

Figure 4.9 SONG TABLE POPULATED

MOVIE

SELECT * FROM MOVIE;

movieName	movieID	director	actor	actress	language
3Idiots	1001	Raj	Aamir	Kareena	Kannada
Dhoom3	1002	Khan	Hrithik	Katrina	Hindi
KGF	1003	Neel	Yash	Radhika	Kannada

Figure 4.10 MOVIE TABLE POPULATED

TRENDING

SELECT * FROM TRENDING;

sIno	songID
3	101
4	102
1	103
9	104
7	105
2	108
8	109
6	110
10	111
5	114

Figure 4.11 TRENDING TABLE POPULATED

PLAYLIST

SELECT * FROM PLAYLIST;

playlistName	playlistID	songID
Favourite	82	103
Favourite	82	108
Favourite	82	114
Favourite	82	109
Old	90	113
Old	90	105
Favourite	82	110

Figure 4.12 PLAYLIST TABLE POPULATED

4.4 SQL Triggers and Stored Procedure

4.4.1 Triggers

Triggers are stored programs, which are automatically executed or fired when some event occur. Triggers are, in fact, written to be executed in response to any of the following events:

- A Database manipulation (DML) statement (DELETE, INSERT, or UPDATE)
- A Database definition(DDL) statement (CREATE, ALTER, or DROP)
- A Database operation(SERVERERROR,LOGON,LOGOFF,STARTUP,SHUTDOWN)

Triggers can be defined on the table, view, schema, or database with which the event is associated. The trigger used in this application is used to update the language of a song in Song when language of a movie is changed

The Trigger is:

CREATE TRIGGER LANGCHANGE AFTER

UPDATE ON MOVIES FOR EACH ROW

UPDATE SONGS SET LANGUAGE=(SELECT LANGUAGE

FROM MOVIE M WHERE M.MOVIEID=SONG.MOVIEID);

movieName	movieID	director	actor	actress	language
3Idiots	1001	Raj	Aamir	Kareena	Kannada
Dhoom3	1002	Khan	Hrithik	Katrina	Hindi
KGF	1003	Neel	Yash	Radhika	Kannada

songID	songName	singerID	movieID	lyricist	musicProducer	musicDistributer	language	genre
101	Malang	501	1001	Pritam	Aditya Chopra	YRF Music	Kannada	Pop
102	Kamli	502	1001	Pritam	Aditya Chopra	YRF Music	Kannada	Pop
103	Tu Hi Junoon	504	1001	Pritam	Aditya Chopra	YRF Music	Kannada	Classical
104	Dhoom Machale Dhoom	503	1001	Pritam	Aditya Chopra	YRF Music	Kannada	Jazz
105	Bande Hain Hum Uske	505	1001	Pritam	Aditya Chopra	YRF Music	Kannada	Classical
106	Aal Izz Well	502	1002	Sonu Nigam	Shantanu Moitra	T-Series	Hindi	EDM

Figure 4.13 TRIGGER ON MOVIE TABLE

4.4.2 Stored Procedure

A stored procedure is a prepared SQL code that can be saved and can be reused over and over again. So if a query has to be written over and over again, instead of having to write that query each time, it can be saved as a stored procedure and can be executed just by calling the procedure. In addition, parameters can also be passed to the stored procedure. So depending on the need, the stored procedure can act accordingly.

Stored procedures are useful in the following circumstances:

- If a database program is needed by several applications, it can be stored at the server and invoked by any of the application programs. This reduces duplication of effort and improves software modularity.
- Executing a program at the server can reduce data transfer and communication cost between the client and server in certain situations.
- These procedures can enhance the modelling power provided by views by allowing, more complex types of derived data to be made available to the database users via the stored procedures. Additionally, they can be used to check for complex constraints that are beyond the specification power of assertions and triggers.

The Stored procedure used in this application is to calculate the subscription cost by accepting 2 parameters. When calling this Stored Procedure, 2 parameters needs to be passed with a call.

Stored Procedure is:

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `subcost`(IN `period` INT(3),IN ‘eid’ VARCHAR2(10))
BEGIN
DECLARE per  int;
DECLARE cost  int;
```

```
OPEN subper;
FETCH FROM period INTO per;
FETCH FROM eid INTO id;
IF per==1
        BEGIN
        cost=100
        END
IF per==3
        BEGIN
        cost=250
        END
IF per==6
        BEGIN
        cost=500
        END
IF per==12
        BEGIN
        cost=1000
        END
UPDATE login
SET subscriptionCost = cost
WHERE email =id;
CLOSE subper;
END$$
DELIMITER ;
```

Show Procedure:

Fname	Lname	email	password	DOB	country	gender	userType	subscriptionPeriod	subscriptionCost
Prashanth	Raju	admin@gmail.com	admin123	1999-09-12	India	Male	admin	NULL	NULL
Prasad	Swamy	prasad@gmail.com	prasad123	2019-11-24	USA	Male	user	1	100
Raveesh	M K	raveesh@gmail.com	raveesh123	2000-06-19	UK	Male	user	6	500
Preetham	M S	preetham@gmail.com	preetham123	1999-06-29	India	Male	user	12	1000

Figure 4.14 STORED PROCEDURE ON MOVIE TABLE

4.5 Database Connectivity

The front end can easily be connected to the back end/database (i.e., MySql) by adding a few instructions in PHP. The following instructions are to be added.

```
<?php
//initializing variables
// function OpenCon() {
$servername = "localhost";
$username = "root";
$password = "";
$db  ="musicdb";  //connect
to server
$conn = mysqli_connect ($servername , $username , $password,$db) or die("unable to connect to host");  //return $conn;
 //}
?>
```

CHAPTER 5

RESULTS

1. Admin Homepage

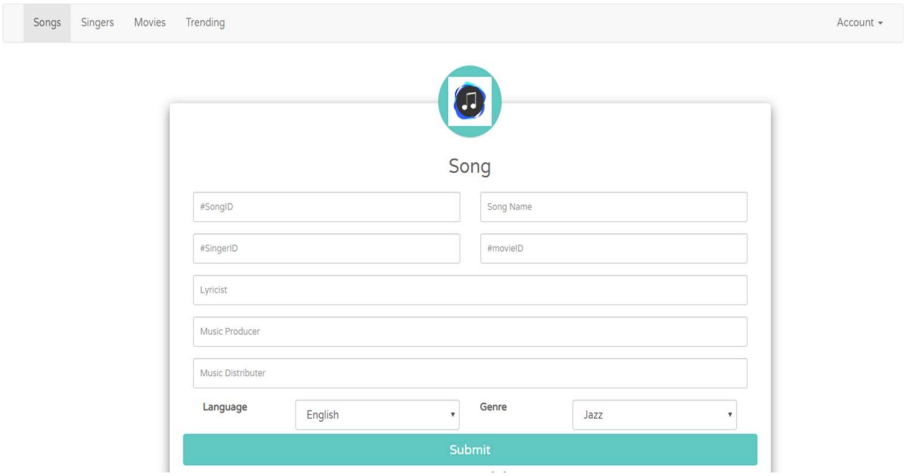

Figure 5.1 ADMIN HOMEPAGE

After admin logins he would be redirected to admin homepage where he could enter data.

2. User Homepage

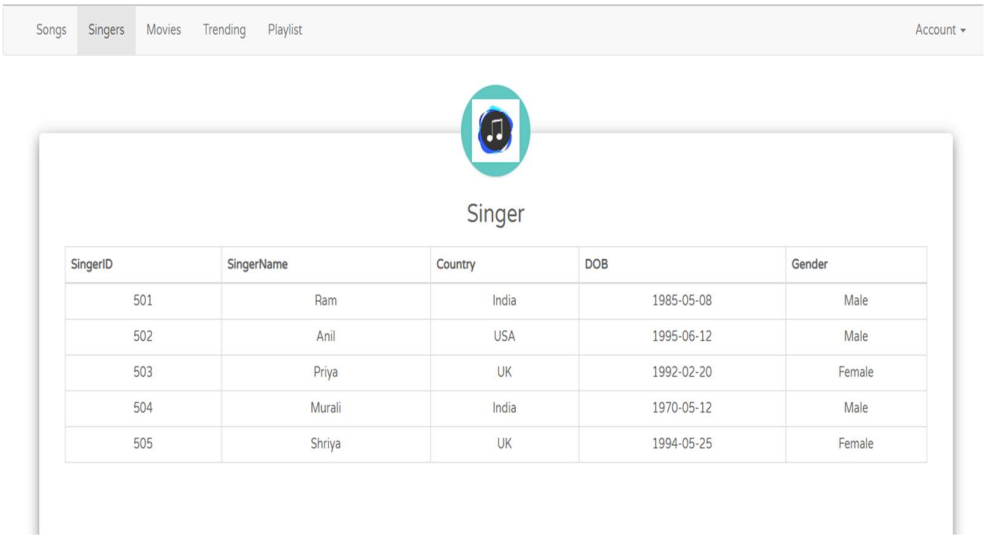

Figure 5.2 USER HOMEPAGE

After user logins he would be redirected to user homepage where he could see the list of songs.

.

3. Admin and User Login

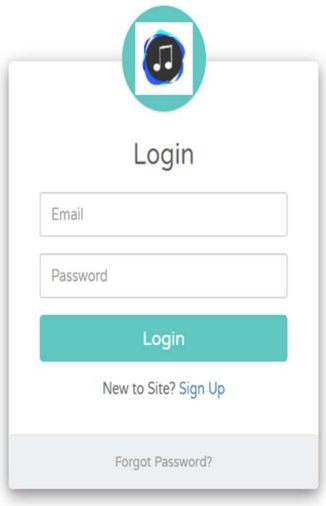

Figure 5.3 LOGIN PAGE FOR ADMIN AND USER

Existing user can login or can register as a new user.

4. Admin and User Signup

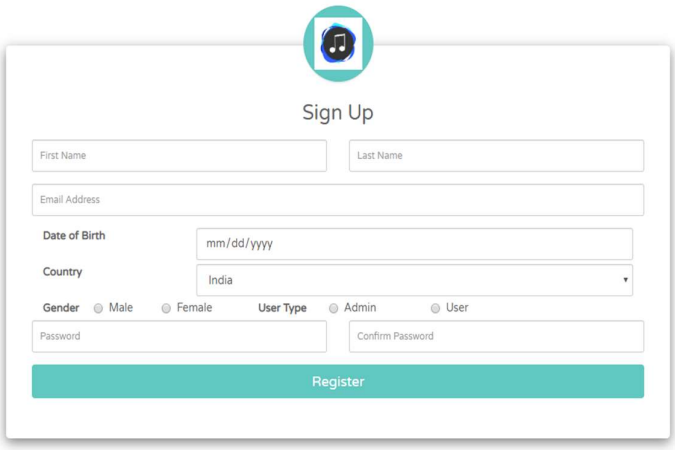

Figure 5.4 SIGNUP PAGE FOR ADMIN AND USER

New users can create a new account by entering their details.

5. Insertion

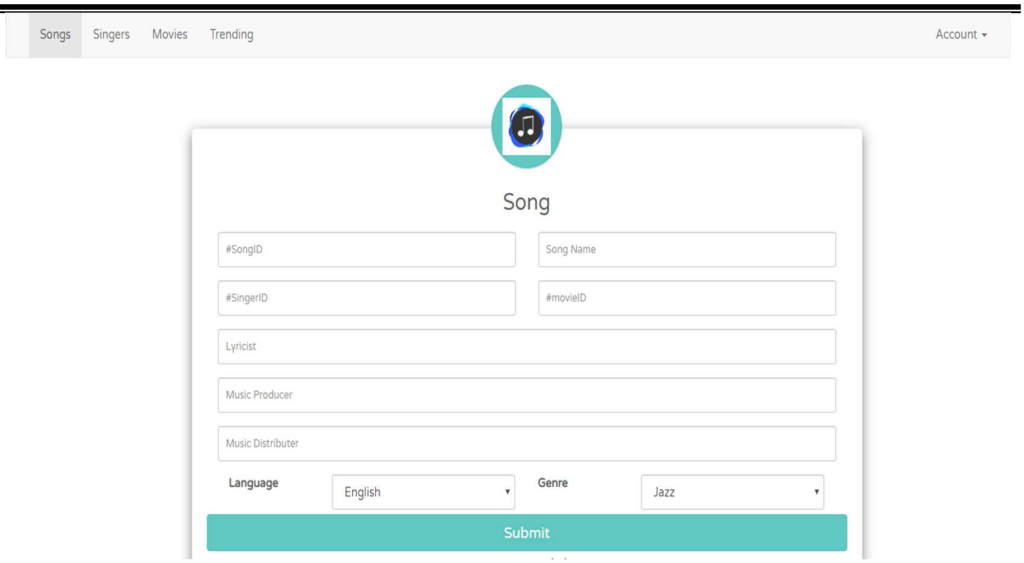

Figure 5.5 INSERTION

Admin can add new songs by entering its respective details.

6. Updating and Deleting Rows

Song

SongID	SongName	SingerID	MovieID	Lyricist	MusicProducer	MusicDistributor	Language	Genre		
101	Malang	501	1001	Pritam	YRF Music	Aditya Chopra	Kannada	Pop	Update	Delete
102	Kamli	502	1001	Pritam	YRF Music	Aditya Chopra	Kannada	Pop	Update	Delete
103	Tu Hi Junoon	504	1001	Pritam	YRF Music	Aditya Chopra	Kannada	Classical	Update	Delete
104	Dhoom Macha	503	1001	Pritam	YRF Music	Aditya Chopra	Kannada	Jazz	Update	Delete
105	Bande Hain Hu	505	1001	Pritam	YRF Music	Aditya Chopra	Kannada	Classical	Update	Delete
106	Aal Izz Well	502	1002	Sonu Nigam	T-Series	Shantanu Moit	Hindi	EDM	Update	Delete
107	Zoobi Doobi	501	1002	Sonu Nigam	T-Series	Shantanu Moit	Hindi	Electronic	Update	Delete
108	Behti Hawa Sa	504	1002	Naresh Iyer	T-Series	Shantanu Moit	Hindi	Classical	Update	Delete
109	Give Me Some	504	1002	Naresh Iyer	T-Series	Shantanu Moit	Hindi	Hip Hop	Update	Delete
110	Jaane Nahin D	502	1002	Naresh Iyer	T-Series	Shantanu Moit	Hindi	Pop	Update	Delete

Figure 5.6 UPDATING AND DELETION

Admin can update and delete entries.

CONCLUSION

Music Catalogue Management System is an application which allows easy distribution of songs. Users can listen to any songs in a very user-friendly interface. The application also provides users with additional song info like singer name, movie name, lyricist etc. User can also create numerous playlists of their favorite songs. This software also provides a subscription model for users who like to take complete advantage of the system like access to premium content, ad-free listening experience, high quality audio etc. This software reduces the amount of manual data entry and gives greater efficiency. The application can also recommend music based on the listening behavior of the user. The User Interface of it is very friendly and can be easily used by anyone. At the end, this software can perform all the tasks accurately and can do the work for which it is made.

FUTURE ENHANCEMENT

Music Catalogue Management system can be further improved by providing users with curated playlists using data analytics and machine learning. Musicians and singers can take advantage of data analytics and machine learning by getting to know the type of music their fans love or the type of music the users are listening more.

Most of the music streaming sites nowadays provide 320kbps quality music to users but with 5G technology grower faster an even more higher quality of music i.e. 1440kbps can be provided. Using modern compression algorithms high quality audio can streamed with lower data consumption.

Social Media sites like Instagram ,Twitter ,Facebook can be integrated into the application so users can share their listening activity to their friends and family. Singers can take advantage of these features to promote their music on social media.