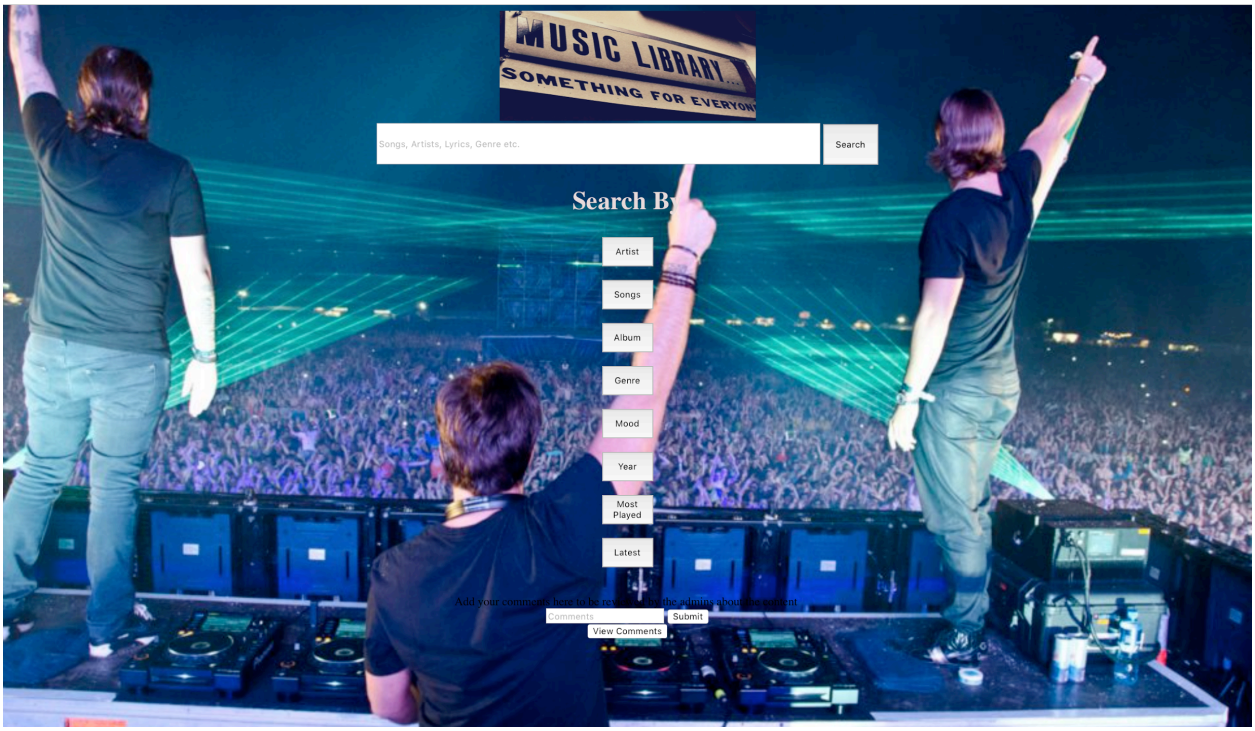
Music Library Management



Abstract:
The purpose of this project is to serve the user's expectations regarding the music library management. In current scenario users from different background expect to sort the music library as per name of singer, genre, frequency of playing, rating, popularity, date of release in market, latest release, albums etc. The project has the feature of user commenting on the content of the website, which will be later reviewed by the administrator. In this project, we are going to implement a library sorting algorithm which will help user to do the library management in an efficient way. The techniques implemented will help the user to get the desired music of his interest.

Approach:
The objective of developing Music Library Management is to create a database which could centrally handle the details and information about the music as mentioned and this data would be easily accessible to the user as per his needs or the kinds of sorting the user wants.

In this system, there will be two kinds of users who can access the database – one will be administrator who can perform all kinds of operations like insert, update, alter, delete etc. on the database and is responsible for the maintenance of the database. And other users can only access the database who can only view the database but cannot perform any operations as administrator.

In this project, we will have the tables for artists, genre, frequency of playing, rating, popularity, date of release in market, latest release, albums etc. and we will connect these tables to each other to get the complete set of data per the user requirements of sorting.

Design:
The User Interface Design module is made for providing the User Interface to the user for interaction with the application. The user will be able to use the application without having any previous knowledge about the application. This module provides the user to access to the database for retrieval of the songs, and search the database according to his interests. And the database we used for the back end is MySQL.

The tables that are used in our application are as follows:

1. Artist

Table No. 1

Column	Type	Null	Description
id *(Primary)*	int(200)	No	Artist id (Primary key)
Artist_name	varchar(200)	No	Name of the artist

The Artist table contains the records of artists. It contains fields such as Artist Id (id) which is the primary key and Artist name.

2. Album

Table No. 2

Column	Type	Null	Description
Album_id *(Primary)*	int(200)	No	Album id (Primary key)
album_name	varchar(200)	No	Name of the Album

3. Genre

Table No. 3

Column	Type	Null	Description
genre_id *(Primary)*	int(200)	No	Genre id (Primary key)
genre_name	varchar(200)	No	Name of the Genre

4. Mood

Table No. 4

Column	Type	Null	Description
track_no	int(200)	No	Track number (foreign Key)
mood_description	varchar(200)	No	Mood Description

5. Tracks

Table No. 3

Column	Type	Null	Description
Id	Int(200)	No	Id of the artist (Foreign Key)
Track_no	Int(200)	No	Song number (Primary Key)
Track_name	Varchar(255)	No	Name of the song
Lyrics	Text	No	Lyrics of the song
Album_id	Int(200)	No	Album id(Foreign Key)
Year	Year	No	Year of the song released
Link	Varchar(200)	No	Name the .mp3 extension in the server
Genre_id	Int(250)	No	Id of genre(Foreign Key)
Count	Int(250)	No	Keeps the track of number of songs played

6. Comments

Table No. 6

Column	Type	Null	Description
Id	Int(250)	No	Id of the comment(Primary Key)
Comments	Varchar(255)	No	Comments of the users

ER Diagram:

The Entity Relationship diagram for our application which is used to represent the attributes, entities and their relationships is shown below.

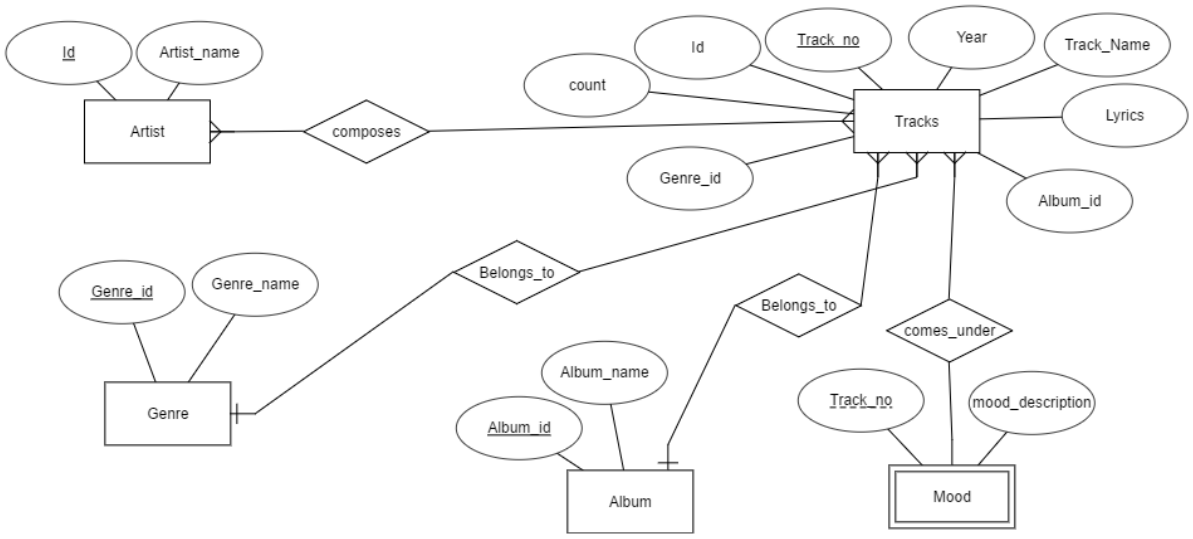


Figure No. 1. ER Diagram

- ➔ Each Artist has an id and an Artist_Name
- ➔ Each Tracks has id, Track_No, Year, Track_Name, Lyrics, Album_Id, Genre_Id and count.
- ➔ Each Album has Album_Id and Album_Name.
- ➔ Each Genre has Genre_Id and Genre_Name.
- ➔ Each Mood has Track_No, mood_description.
- ➔ Each Album can have many tracks. So there is one to many relationship between Album and Tracks.
- ➔ A single Artist have many tracks. Two or more Artists can compose a single track. So, it is a many relationships.
- ➔ A Track Belongs_To one and only one Genre.
- ➔ Mood is a weak entity. It depends on Tracks. A Track comes_under a particular Mood.

Use Case Diagram:

Use-case diagram shows a set of use-cases and actors and their relationships. These diagrams illustrate the static use case view of system and are important in organizing and modeling the behavior of a system. The interaction of the application with the outside world can be represented using use-case diagram.

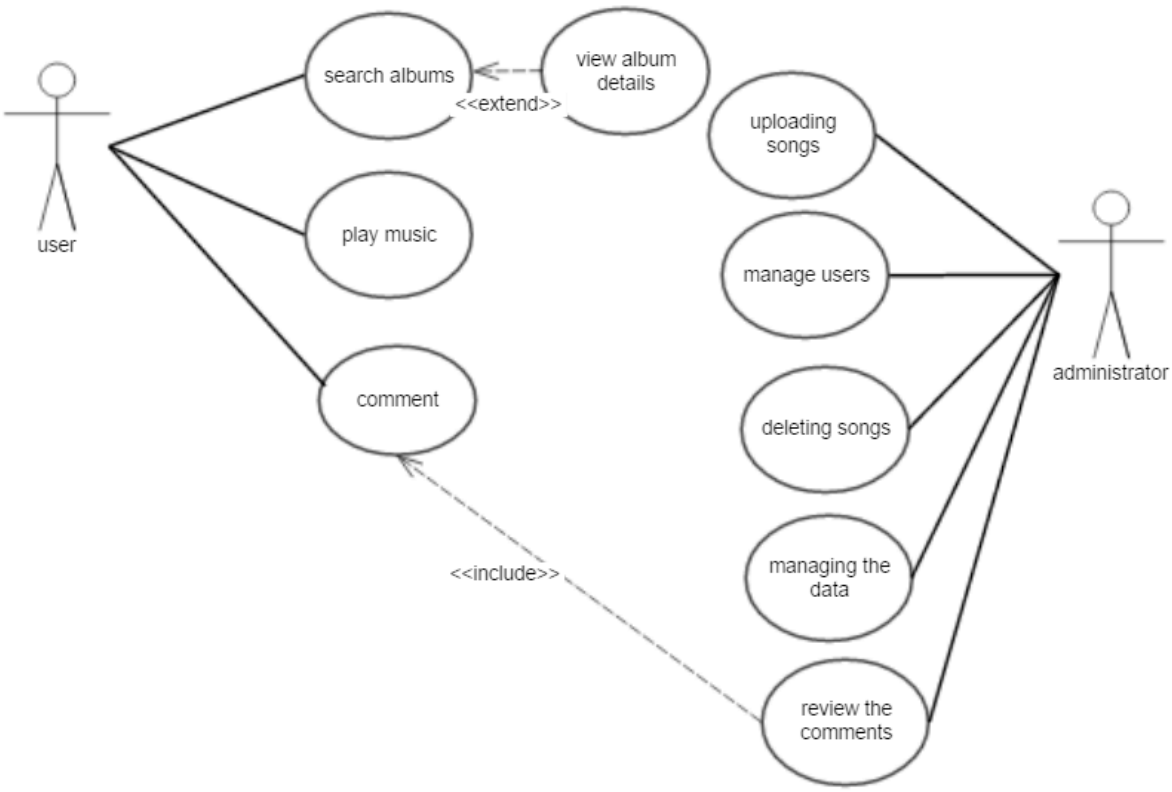


Figure No. 2. Use Case Diagram

Implementation:

The web server we used for implementing our project is MAMP server which is an open source and proprietary commercial software which is used to run dynamic websites on Apple Macintosh computers. The database server which is used to store all the data regarding music libraries is MySQL. The application is written in PHP (Hypertext Preprocessor) which is a scripting language used for web development and can be embedded into HTML. And HTML and CSS are used for styling the user interface.

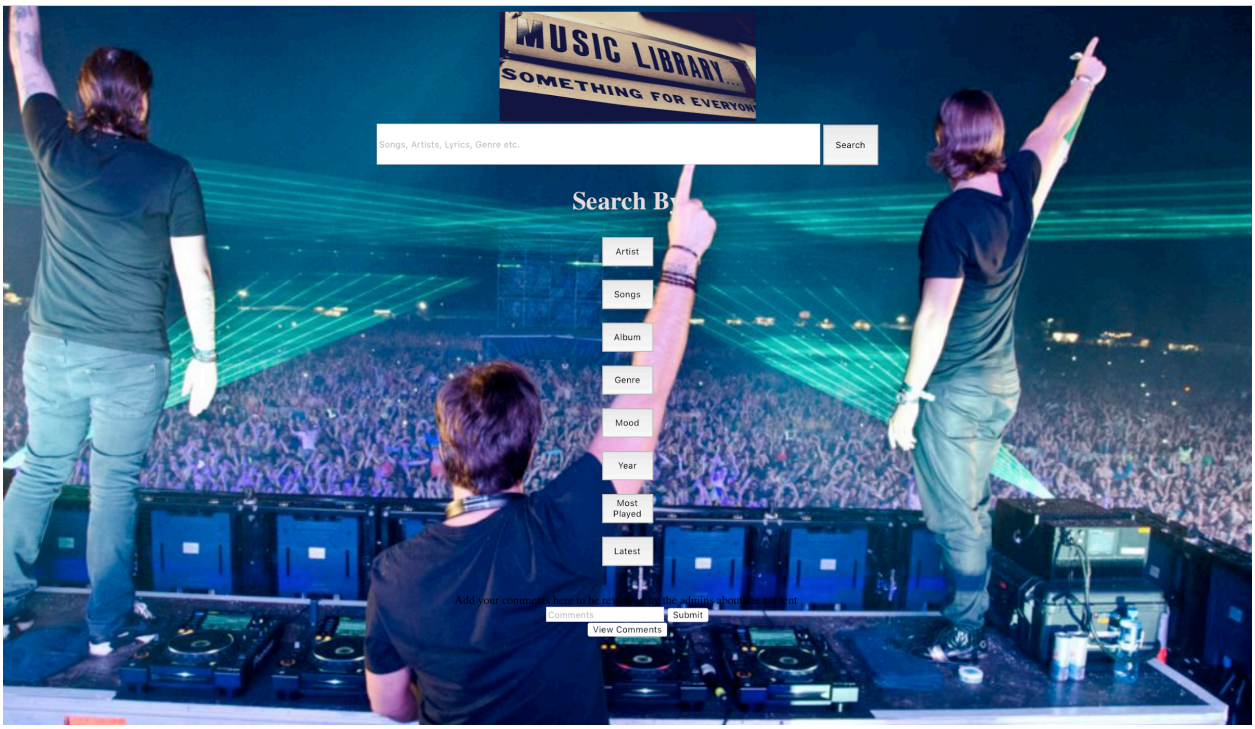
The system works by first loading the application's dynamic front end. This is done by querying the server for a list of tracks in the database, list of artists and the list of playlists, while the user interacts with the site, the display information is kept up-to-date, such as the recently played songs will be displayed with the number of times the song has been played and the current albums can be displayed with the search filter of latest released albums option. The user will be

able to search songs, filter them according to his interest and can play the songs and also he will able to comment on the songs which can be reviewed by the administrator.

Results:

We have developed an application which runs on a browser, using PHP with an interactive web interface, which allows the user to access and query the database using MySQL. It lets the user search for an entity such as song, artist, etc. based on setting filters by checking buttons. All this data will be collected from various sources from the internet.

This is the home page of the project. More details on how to deploy the project and get it running can be found in the README file.



Future work:

A more interactive web app can be developed using AJAX (for dynamic loading of content), OAuth (for authentication), Django(for handling large datasets), Javascript and Jquery (for more attractive interface). As the hits to the web app increases, in order to have more songs on the website, we can give upload permissions to the users who have created an account. In order to make it easy for the users to find the songs of their interests, we can give ratings on different entities such as popularity of song, album, artist etc. We can also let users make a playlist of songs which can later be reviewed by other users. We would like to create a dynamic feed on what songs are being uploaded into the website at that moment. To improve the search performance, we would like to implement clustering algorithms.

References:

[1]. Music Downloaded from: MusicPleer http://musicpleer.cc/

[2]. MAMP Documentation: https://www.mamp.info/en/documentation/

[3]. PHP Documentation: https://php.net/docs.php

[4]. HTML, CSS Documentation and Reference:
W3Schools: http://www.w3schools.com/html/html_intro.asp
HeadFirst:"http://artsites.ucsc.edu/sdaniel/170a_2014/Head_First_HTML_CSS_XHTML.pdf"

[5]. Mozilla Foundation CSS: https://developer.mozilla.org/en-US/docs/Web/CSS

[6]. MySQL Reference: https://gist.github.com/sathishs/2841425