



# SoundRule: Final Document

## *Project 2*

CS-GY 6083 Principles of Database Systems ,under Prof : Torsten Suel

By : Ujjawal Gupta  
Ajay Shete

---

## **INDEX**

1. Introduction .....	2
2. Entity Relationship Diagram .....	3
3. Design Assumptions .....	4
4. Converting ER diagram into relational database model/Schema .....	4
5. UI description .....	6
6. Code Explanation .....	13
7. Future consideration .....	18

# **SoundRule**

Let The Music Play!

## **Introduction**

SoundRule is an online music streaming website for users to listen to their favorite artist and their favorite music for free. Every user has to register, using a unique username, email id, city the user lives in and a password. The user may create a playlist, like an artist, rate a song or follow other users. Once the user registers it can login and change his details.

The website will provide a list of tracks along with their genre and artist description. Same track can be in multiple locations i.e. a solo song, Album or a user Playlist. So when the user will search a particular song or artist the text will be searched through track and artist tables. The user can mark its playlist public or private. To implement this we have created a view for public playlists. So every time a user creates a new playlist script will check whether the playlist type is private or public, if its private it will be added to playlist database only, and if its public the entry will be added to both playlist database and public\_playlists view.

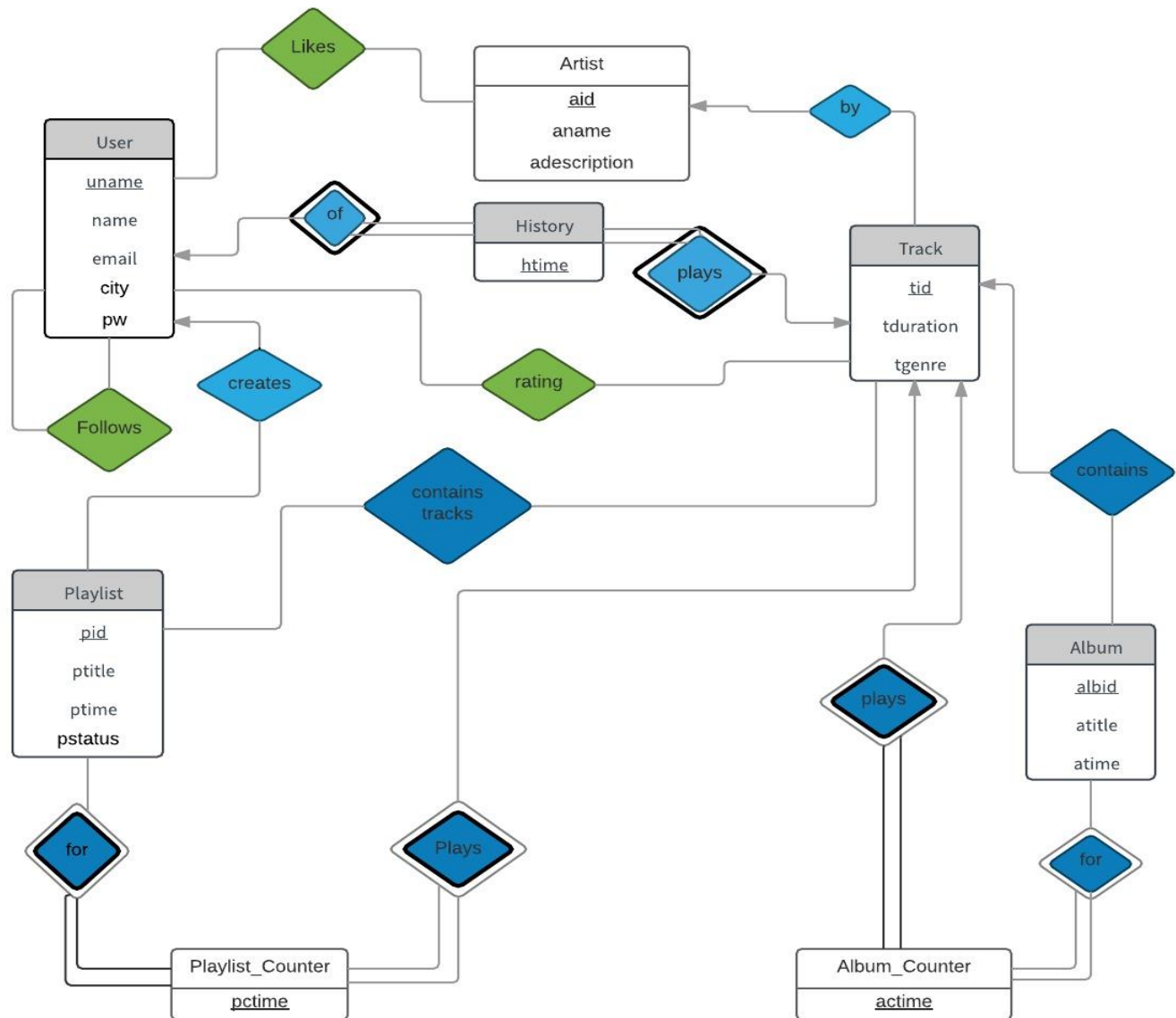
A history of number of times a song is played by user is maintained at the backend. To identify the popularity if a song, an album or a playlist three tables have been made history, playlist counter and album counter. So, when the user searches a song or an artist, it will be presented with the search results from tracks, Album or Playlist. Whichever option the user chooses to play the data entry will be made accordingly in that table. For example, the user choses to play the song from a playlist, an entry will be made in two tables i.e. playlist counter and history. So we assume that a playlist is played if any one song of the playlist is played. And depending on that we can calculate the popularity of any album or playlist.

The user can like an artist or rate a particular track on the scale of 1 to 5. So, if we have to check the popularity if a particular song or an artist we can use the Like and Rating tables in our schema.

The other feature that this website offers is that a user can follow other users, depending upon the interests and the popularity of the user's playlists. We have not implemented a method to follow a playlist, you can access any user's playlist of it is publicly available, but you cannot like/rate that playlist. To know which user is following which other user, or in other words, to know which users are followed by which other users, we create a Follow table. This table is a self-loop to User table. It stores the uname of the user that follows the other user and the uname of the other user. The attributes are renamed for the table as user, followedby.

The website is designed using MySQL as database server. Further, backend scripting is done in PHP and Front end will be designed using HTML5, CSS3 and JavaScript. WAMP is the local host server application that will be used for the website.

## Entity Relationship Diagram



## **Design Assumptions**

1. Users will have to register by providing username, password, name, email and city. They will be able to create their playlists separately.
2. The Artist table stores the detail of all artists. If there are multiple artists performing on a track their collaboration will be stored as a unique aid.
3. A user can follow other users, but they cannot follow any specific playlist. Additionally, the User can also like an artist.
4. A separate table has been designed to keep tab of all the tracks that are added in the playlist and verify when the user played the track from that playlist.
5. We have implemented a similar structure for the Album & Album\_Counter.
6. The popularity of a playlist can be determined by calculating the number of times a song from that playlist was played. For this purpose, we have implemented a Playlist\_Counter table.
7. Similarly, we can also find out the popularity of the Albums by verifying the Album\_Counter table.
8. A history of the tracks played by the user is maintained in the history table. So if a track played is also available in an album or a playlist the entries will be made both to history table as well as respective counter table.
9. The user can choose to keep his Playlist private or public, and we will allow other users to access the songs by using the Playlist's view table.

## **Converting ER diagram into relational database model/Schema**

- User (**uname**, name, email, ucity, pw)
- Artist (**aid**, aname, adesc)
- Track (**tid**, ttitle, tdur, genre, aid, abid)  
Foreign key (aid) references Artist (aid)
- History (**uname, tid, htime**)  
Foreign key (uname) references User (uname)  
Foreign key (tid) references Track (tid)
- Rating (**uname, tid, rtime**, score)  
Foreign key (uname) references User (uname)  
Foreign key (tid) references Track (tid)
- Album (**abid**, abtitle, abtime)
- Album Counter (**abid, tid, actime**)  
Foreign key (abid) references Album (abid)

Foreign key (tid) references Track (tid)

- Playlist (**pid**, ptitle, pdate, ptype, uname)  
Foreign key (uname) references User (uname)
- Playlist Tracks (**pid, tid**)  
Foreign key (pid) references Playlist (pid)  
Foreign key (tid) references Track (tid)
- Playlist Counter (**pid, tid, pctime**)  
Foreign key (pid) references Playlist (pid)  
Foreign key (tid) references Track (tid)
- Follow (**user, followedby**)  
Foreign key (uname) references User (uname)  
Foreign key (followedby) references User (uname)

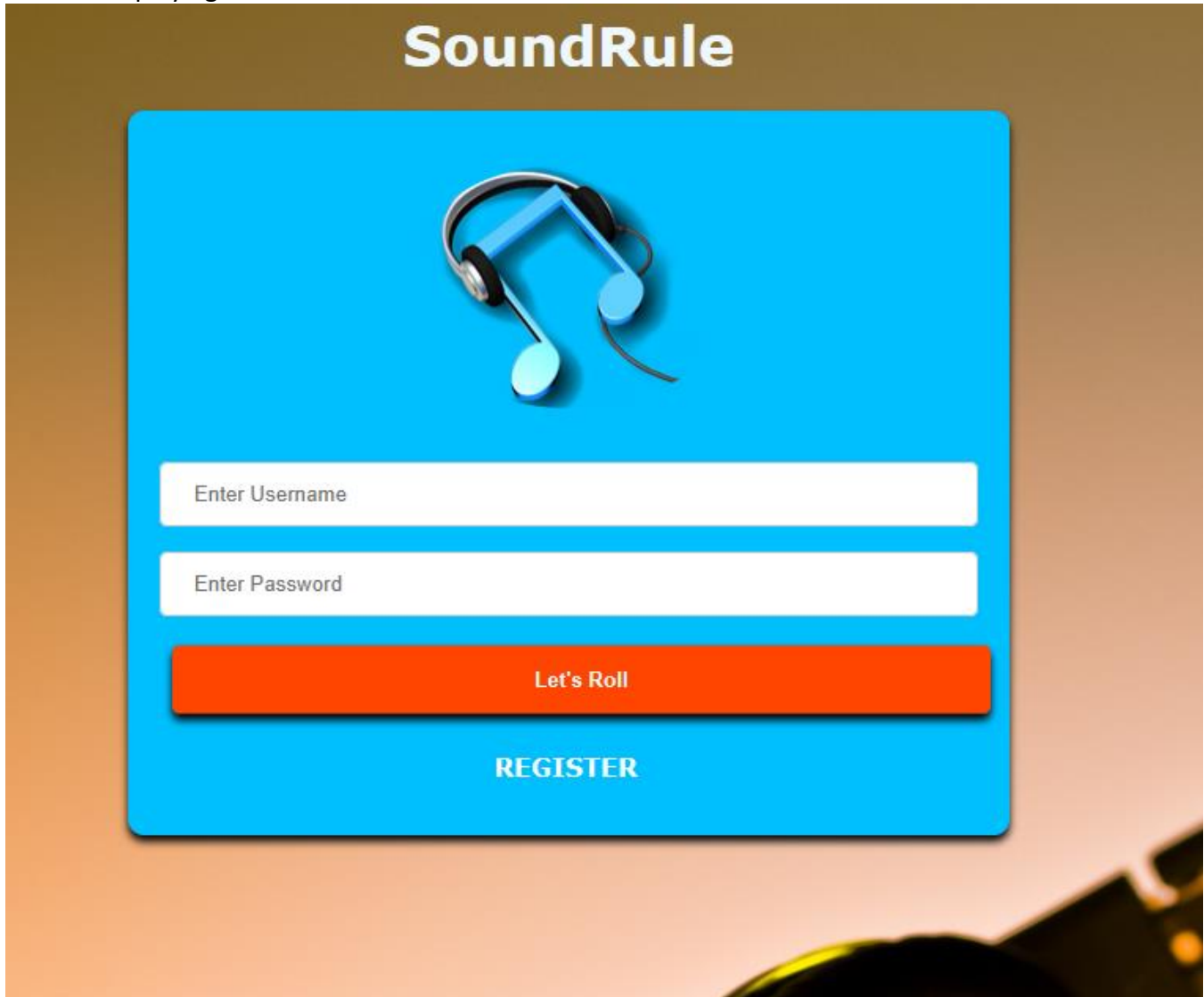
Also, to distinguish between public and private playlists we have created **a view** for public playlists using:

```
create View public_playlists(pid, ptitle, pdate, uname) as
(select pid, ptitle , pdate, uname
from playlist
where ptype = "public")
```

## UI description

### Login Page

We require user to login first before entering the website, this model would have been helpful, even when the Music streaming service was paid, So user needs to first Authenticate and then he/she can enter start playing Music. So the user will first see this screen.

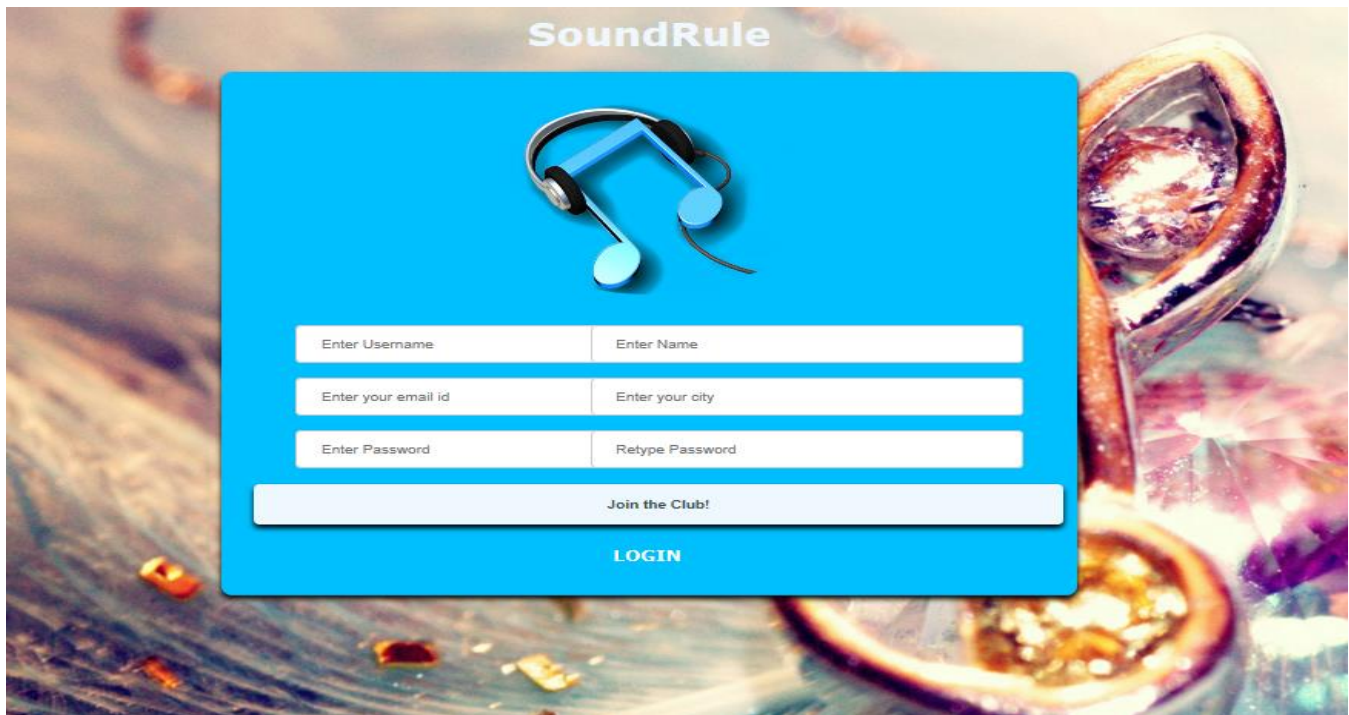
The image shows a login page for 'SoundRule'. The page has a brown background. At the top, the word 'SoundRule' is written in a large, white, sans-serif font. Below the title is a blue rectangular box. Inside this box, at the top, is an illustration of a pair of headphones with a blue musical note integrated into the headband. Below the illustration are two white input fields: the first is labeled 'Enter Username' and the second is labeled 'Enter Password'. Below these fields is a large orange button with the text 'Let's Roll' in white. At the bottom of the blue box, the word 'REGISTER' is written in white, all-caps, sans-serif font.

### Register

If the user is not registered then user has to first register, by entering its name, city email address a unique uname and a password.

After Registration user will be requested to login again.

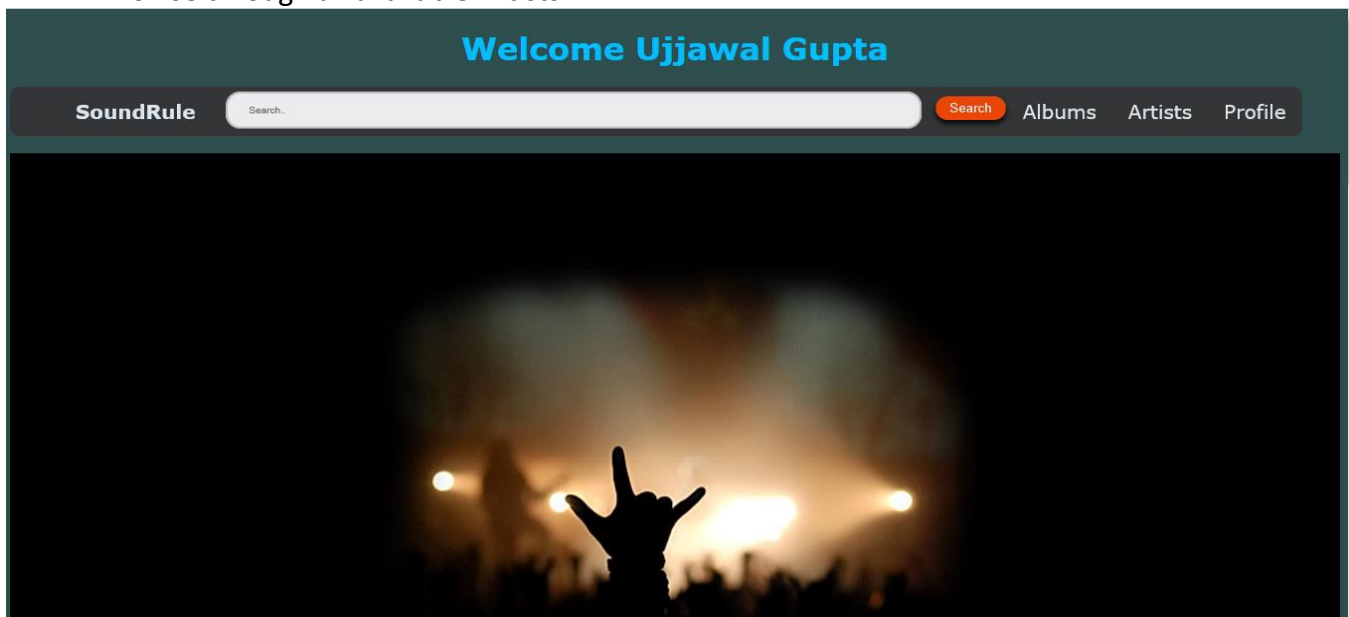




## Main Page

After Login in user will be greeted with a welcome message and he/she will receive following options on the main page.

1. Search their favorite songs, artists and albums.
2. Browse through all available Albums
3. Open their profile page
4. Browse through all available Artists.



As we scroll down on the page we also see tile representation of popular artists and Albums.



## Album Page

This page shows all the available albums, since the data provided did not include genre the albums are not categorized.

SoundRule			Search..	Search	Albums	Artists	Profile
Album		Release Date					
Collision Course		2004-10-12		Explore			
Prism		2010-05-11		Explore			
Recovery		2011-04-26		Explore			
Rose petals		1996-02-11		Explore			

## Explore Album Page

As soon as we click on Explore option on any album on the Album page we see all the tracks associated with this album. And the following page opens up.

SoundRule			Search..	Search	Albums	Artists	Profile
Track	Duration	Album					
Numb-Encore	03:45:00	Linkin Park and JayZ		Add to Playlist	Play	Rate	
Numb	03:25:00	Linkin Park		Add to Playlist	Play	Rate	
99 problems	04:20:00	JayZ		Add to Playlist	Play	Rate	
Roar	03:55:00	Linkin Park and JayZ		Add to Playlist	Play	Rate	

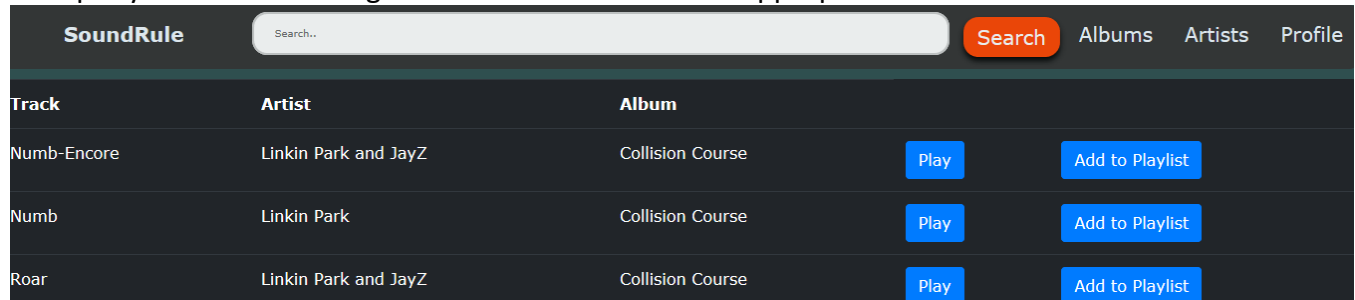
## Artists Page

The Artists page is similar to albums page, just that it enlists all the Artists in the database, you can like the artist, or you can check out all the tracks associated to the artist.

SoundRule			Search..	Search	Albums	Artists	Profile
Artist		Description					
Linkin Park		Nu metal, Rock, Pop		Unlike	Tracks		
Pink Floyd		Jazz, Rock, Pop		Unlike	Tracks		
Linkin Park and JayZ		Nu metal, Rock, Pop, Rap		Like	Tracks		
Eminem		Rap, Rock		Like	Tracks		
Katy Perry		Folk, Rock, Pop		Like	Tracks		
JayZ		Rap, Rock		Like	Tracks		
Justin Timerlake		Pop Artists		Like	Tracks		

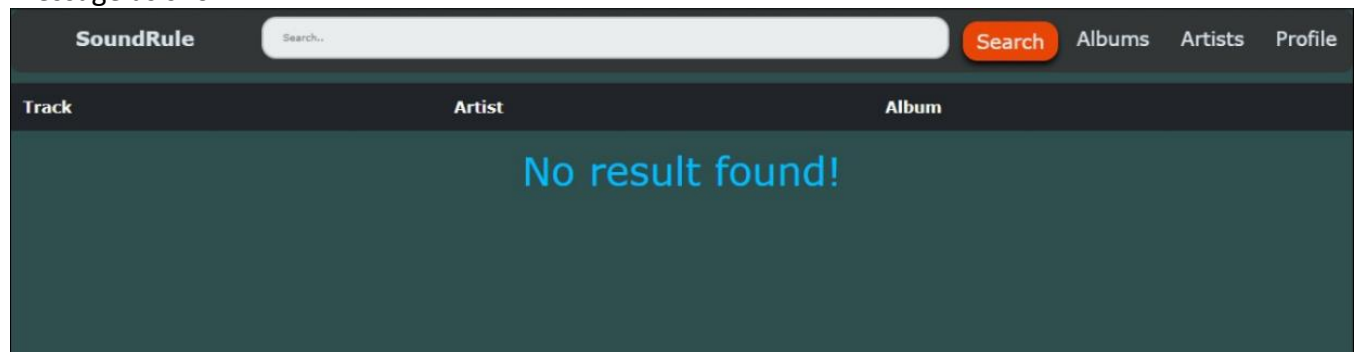
## Search Results

The user can search through their favorite artist, track or album and based on the keyword enter the SQL query will search through the database the find the appropriate results.



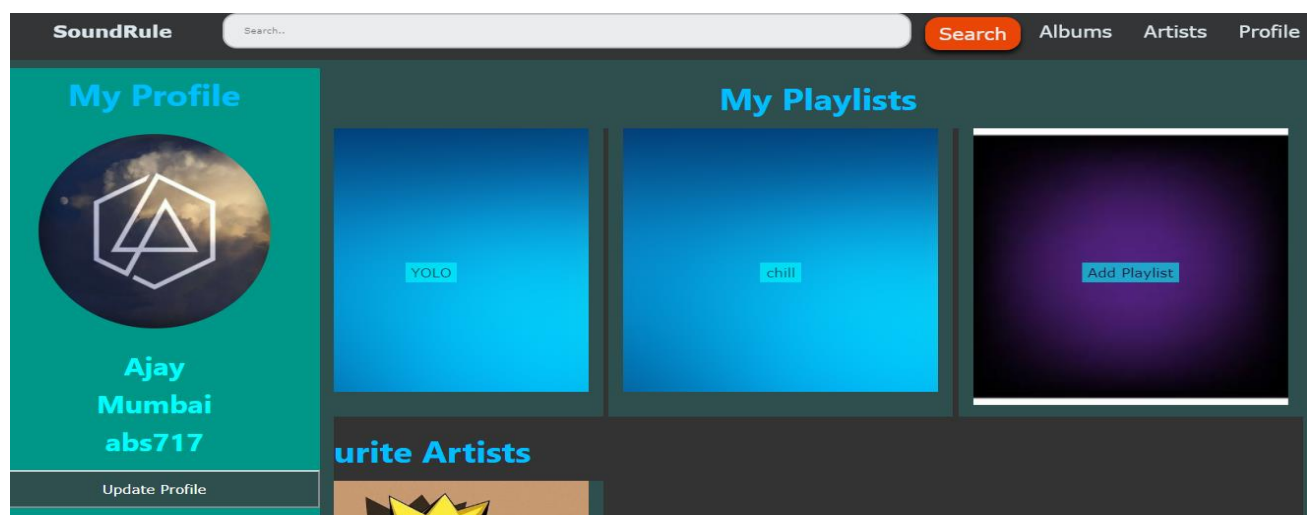
Track	Artist	Album		
Numb-Encore	Linkin Park and JayZ	Collision Course	<a href="#">Play</a>	<a href="#">Add to Playlist</a>
Numb	Linkin Park	Collision Course	<a href="#">Play</a>	<a href="#">Add to Playlist</a>
Roar	Linkin Park and JayZ	Collision Course	<a href="#">Play</a>	<a href="#">Add to Playlist</a>

Here as we can see, user can either play that track or it can add to his/her playlist.  
In case user tries to search something which is not available in the database, it will show an error message as shown.



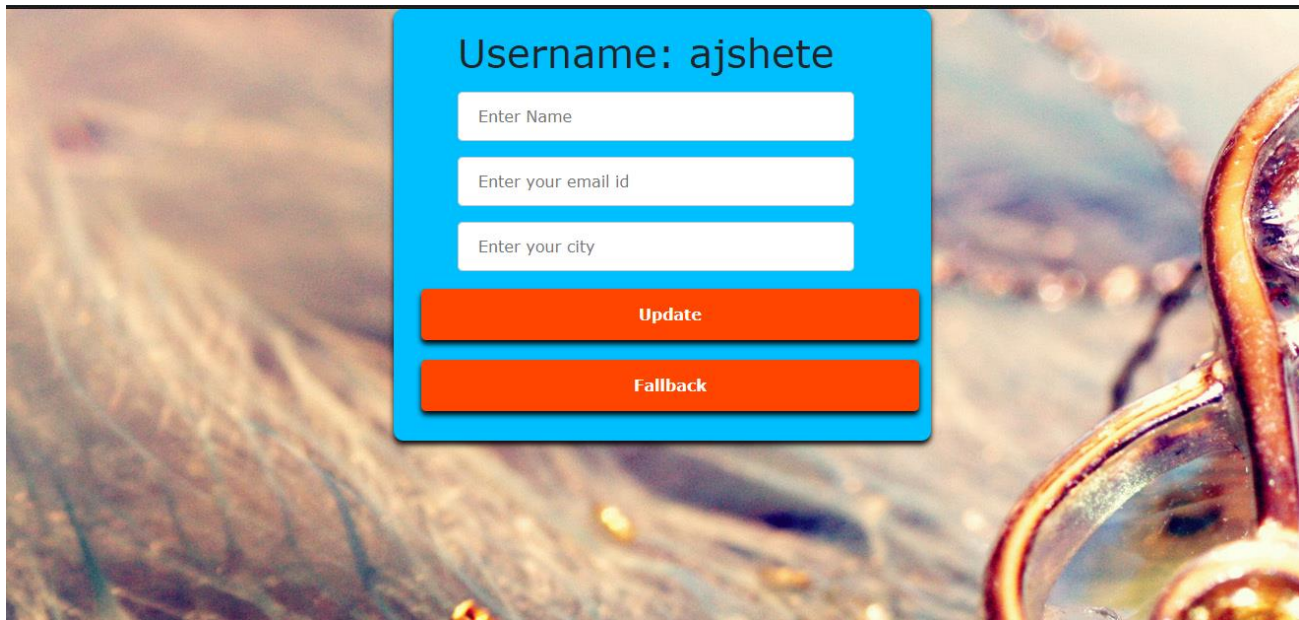
## My profile page

My profile page offers user to checkout his details; edit them and his favorite artists and the people he/she follows.



## Update Profile

User can update the details of his profile like name, city and email address by clicking on update profile page.

A screenshot of a web application showing a profile update form. The form is a blue rectangle with rounded corners, centered over a background image of a string of pearls and a gold chain. At the top of the form, it says 'Username: ajshete'. Below this are three white input fields with placeholder text: 'Enter Name', 'Enter your email id', and 'Enter your city'. At the bottom of the form are two orange buttons: 'Update' and 'Fallback'.

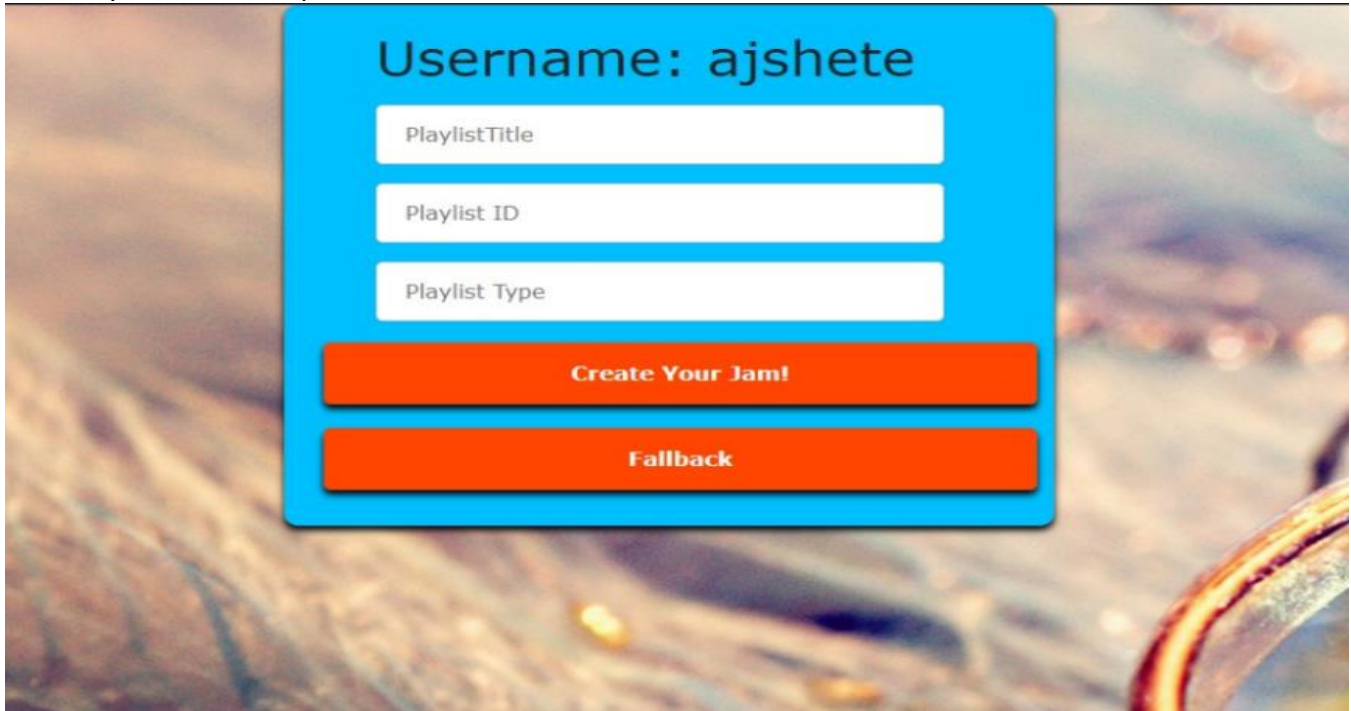
## History

All the tracks that have been played by the user are shown by clicking on the check history page at the bottom of the profile page.

SoundRule			
Search..		Search	Albums Artists Profile
Track	Artist	Album	Time
Highway to Hell	AC/DC	Death flower	2017-12-15 13:53:56
Turning Tables (Live Acoustic)	Adele	21	2017-12-14 22:51:22
Turning Tables (Live Acoustic)	Adele	21	2017-12-14 22:50:38
Turning Tables (Live Acoustic)	Adele	21	2017-12-14 22:49:13
Numb-Encore	Linkin Park and JayZ	Collision Course	2017-12-14 20:18:43
Numb-Encore	Linkin Park and JayZ	Collision Course	2017-12-14 20:18:41
Numb-Encore	Linkin Park and JayZ	Collision Course	2017-12-14 20:18:13
Teenage Dream	Katy Perry	Prism	2017-12-14 12:25:48
Teenage Dream	Katy Perry	Prism	2017-12-14 12:06:25
Teenage Dream	Katy Perry	Prism	2017-12-14 12:01:19

## Create Playlist

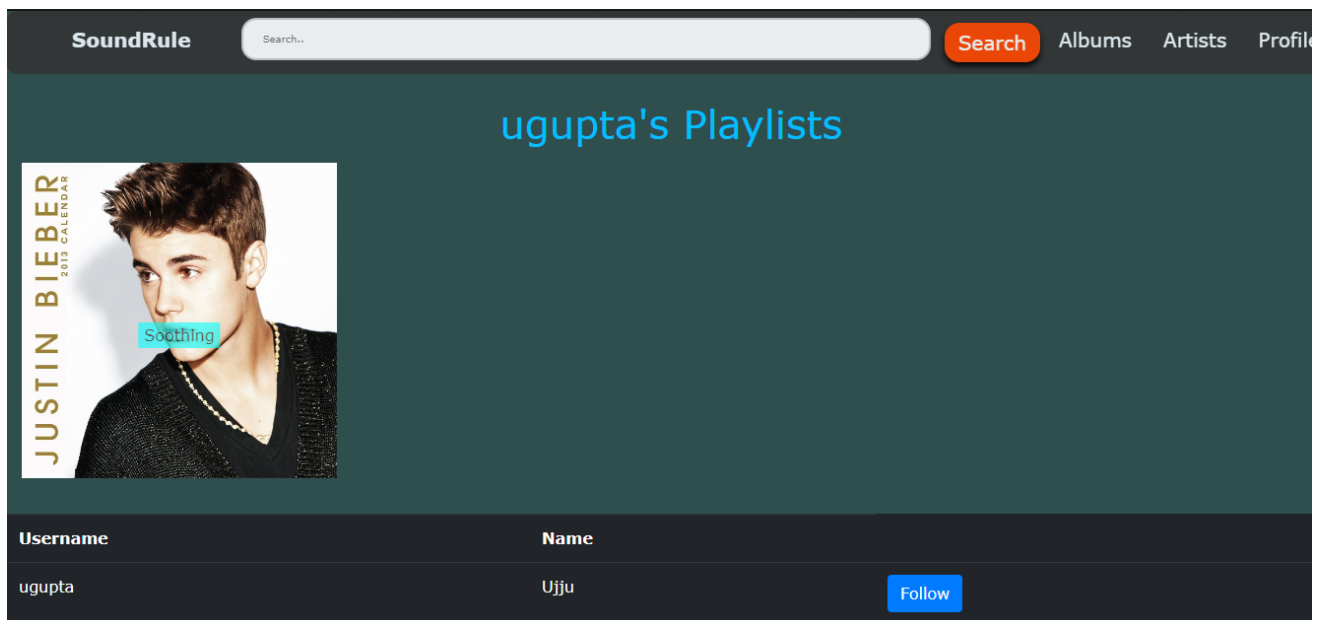
User can create his own playlist and then add tracks that he wants. Further, he/she can keep it public for everyone to see or private.



The screenshot shows a 'Create Playlist' form with a blue background. At the top, it displays 'Username: ajshete'. Below this are three input fields: 'PlaylistTitle', 'Playlist ID', and 'Playlist Type'. At the bottom of the form are two orange buttons: 'Create Your Jam!' and 'Fallback'.

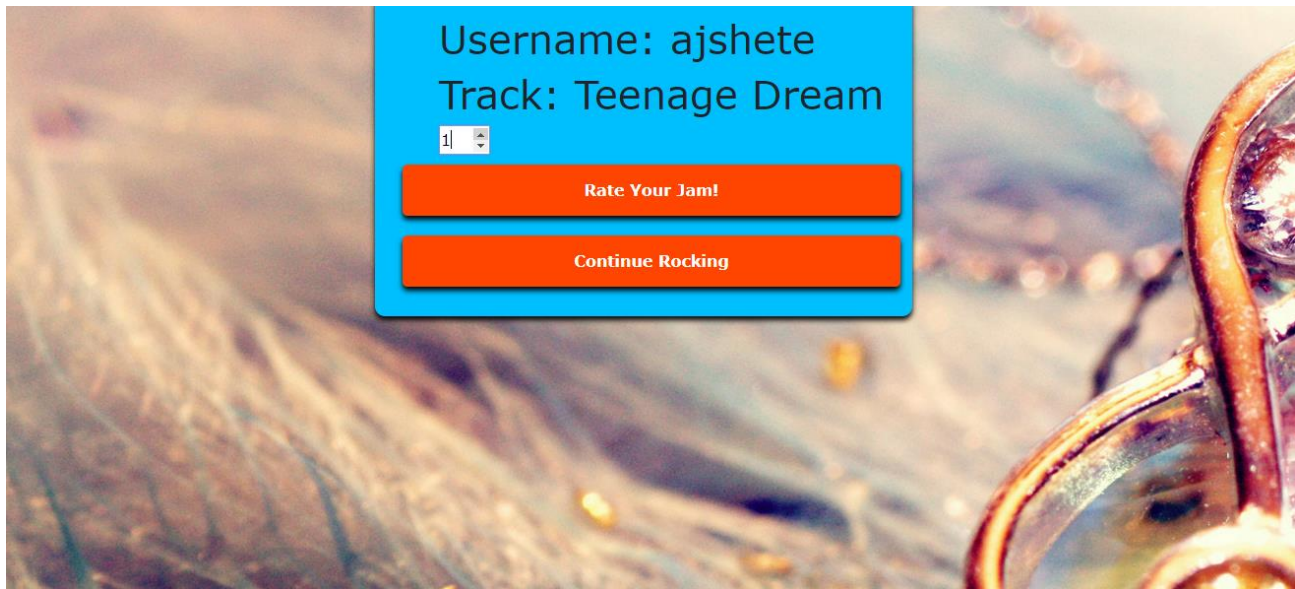
## Follow User

One user can follow other users.



## Rating

User can rate the track from 1 to 5



## Wamp

We used wamp server as the local host, so wamp controls the MySQL queries and backend php

A screenshot of the phpMyAdmin web interface. The left sidebar shows a tree view of the database structure, including 'dbproj', 'album counter', 'albums', 'artists', 'follow', 'history', 'likes', 'playlist', 'playlist counter', 'playlist track', 'rating', 'tracks', and 'user'. The main area displays the 'user' table with 8 rows. The table has columns: 'uname', 'pw', 'name', 'email', and 'ucity'. The rows contain user data, including 'ugupta', 'ajshete', 'justinb', 'shrutigarg', 'Jhonsmith', 'BradleyCooper', 'NancyInQueens', and 'vivek12'. The interface includes various navigation and action buttons at the top and bottom.

## Code Explanation:

### Home Page

```
<?php
require_once('../dbProj_connect.php');

//include 'ps3_checkavailability.php';
if(isset($_POST['login'])){

//    echo '<form action = "http://localhost/ps3/bookings.php" method="post">';

$username = $_POST['uname'];
$password = $_POST['pw'];

$query1 = "select uname
          from user
          where uname = '". $username."' and pw = '". $password.'";
$response1 = @mysqli_query($dbc, $query1);

while($row = mysqli_fetch_array($response1)){
    $user_name = $row['uname'];
}

$query2 = "select aid, aname, adesc
          from artists
          where aid like '10_";
$response2 = @mysqli_query($dbc, $query2);

$query3 = "select abid, abtitle, abdate
          from albums";
$response3 = @mysqli_query($dbc, $query3);

$query4 = "select uname as user, name
          from user
          where uname != '". $username.'";
$response4 = @mysqli_query($dbc, $query4);

$query5 = "select name
          from user
          where uname = '". $username.'";
$response5 = @mysqli_query($dbc, $query5);
while($row = mysqli_fetch_array($response5)){
    $name = $row['name'];
}

echo '<h1>Welcome ' . $name. '</h1>';

if($user_name == $username){
echo '

```

This code, displays the username, and fetches all the artists and albums from the database.



## Tracks

```
if($response1){
    while($row = mysqli_fetch_array($response1)){

        echo '<tr>';
        $tname = $row['tname'];
        $tdur = $row['tdur'];
        $abtitle = $row['abtitle'];
        $tid = $row['tid'];
        $abid = $row['abid'];

        echo '<td align="left">' .
            $tname .
            '</td>
            <td align="left">' .
            $tdur .
            '</td>
            <td align="left">' .
            $abtitle.
            '</td>
            <td align = "left">
                <form action = "add.php" method = "post">
                    <input type="submit" name="add" value="Add to Playlist"
class="btn btn-primary">
                    <input type = "hidden" name = "tid" value = "'. $tid.'">
                    <input type = "hidden" name = "uname" value = "'. $uname.'">
                </form>
            </td>
            <td align = "left">
                <form action = "tracks.php" method = "post">
                    <input type="submit" name="playArtHist" value="Play"
class="btn btn-primary">
                    <input type = "hidden" name = "tid" value = "'. $tid.'">
                    <input type = "hidden" name = "aname" value = "'. $aname.'">
                    <input type = "hidden" name = "uname" value = "'. $uname.'">
                </form>
            </td>
            <td align = "left">' .
                '<form action = "rate.php" method = "post">
                    <input type="submit" name="rateArt" value="Rate" class="btn
btn-primary">
                    <input type = "hidden" name = "tid" value = "'. $tid.'">
                    <input type = "hidden" name = "aname" value = "'. $aname.'">
                    <input type = "hidden" name = "uname" value = "'. $uname.'">
                </form>
            </td>';
        echo '</tr>';
    }
    echo '</table>';
}
echo mysqli_error($dbc);
```

This code fetches all the tracks from the database and arranges them in rows.

## Profile Page

```
$uname = $_POST['uname'];

$query5 = "select name, ucity, email
          from user
          where uname = '". $uname . "'";
$response5 = @mysqli_query($dbc, $query5);
$row = mysqli_fetch_array($response5);
$name = $row['name'];
$ucity = $row['ucity'];
$email = $row['email'];

$query1 = "select pid, ptitle, pdate, ptype
          from playlist
          where uname like '". $uname . "%'";
$response1 = @mysqli_query($dbc, $query1);

echo '
```

This query displays all the user information, like Name, email Id and password, user can further edit it if he wants.

## Playlists

```
if($response1){
    while($row = mysqli_fetch_array($response1)){

        echo '<tr>';
        $pid = $row['pid'];
        $ptitle = $row['ptitle'];
        $pdate = $row['pdate'];
        $ptype = $row['ptype'];

        echo '
                <div class="container">
                
                <div class = inner-but>
                <form action = "playlistTrack.php" method = "post">
                <button class = "btn-in-option" name = "playlistTrack" type
= "submit">'. $ptitle . '</button>
                <input type = "hidden" name = "ptitle" value =
"'. $ptitle . "'>
                <input type = "hidden" name = "uname" value = "'. $uname . "'>
                <input type = "hidden" name = "pid" value = "'. $pid . "'>
                </form>
                </div>
            </div>';
    }
}
```

On the profile page user can access his past playlists, to display that above mentioned code is used.

## Rate Track

```
$uname = $_POST['uname'];
$abid = $_POST['abid'];
$tid = $_POST['tid'];
$score = $_POST['score'];
echo $score;

$query1 = "select tname from tracks where tid = '". $tid . "'";
$response1 = @mysqli_query($dbc, $query1);
while($row = mysqli_fetch_array($response1)){
    $tname = $row['tname'];
}

$query2 = "INSERT INTO rating (uname, tid, score, rtime)
VALUES('". $uname . "', '". $tid . "', '". $score . "', now())
ON DUPLICATE KEY UPDATE
score = '". $score . "'";
$response2 = @mysqli_query($dbc, $query2);

echo '

```

User can rate a track with a rating of 1 to 5, this code accepts that value and put an entry in the database.

## History

```
$uname = $_POST['uname'];

$query1 = "select t.tname, t.aname, ab.abtitle, h.htime
from history as h join tracks as t on h.tid = t.tid
join albums as ab on t.abid = ab.abid
where h.uname = '". $uname . "'";
$response1 = @mysqli_query($dbc, $query1);

echo '

```

Whenever user plays any song an entry is made in the history database, this displays all the tuples.

## Follow User

```
if($response1){
    while($row = mysqli_fetch_array($response1)){

        echo '<tr>';
        $user = $row['user'];
        $name = $row['name'];

        echo '<td align="left">' .
            $user.
            '</td>
            <td align="left">' .
            $name .
            '</td>
            <td align = "left">
            <form action = "users.php" method = "post">

```

```

        <input type="submit" name="follow" value = "Follow" class="btn
btn-primary">
        <input type = "hidden" name = "user" value = "'. $user.'">
        <input type = "hidden" name = "uname" value = "'. $uname.'">
    </form>
</td>';
echo '</tr>';

```

A user can follow or Unfollow a particular user, this code sends an entry in the database if the user wants to follow or deletes one if he wants to unfollow.

### Update User Profile

```

$uname = $_POST['uname'];
$name = $_POST['name'];
$ucity = $_POST['ucity'];
$email = $_POST['email'];

$query1 = "update user
set name = '". $name."',
email = '". $email."',
ucity = '". $ucity."'
where uname = '". $uname."'";
$response1 = @mysqli_query($dbc, $query1);

echo '

<div class="center">
<div class="container">
    <form action="edit.php" method="post">
        <ul>
            <li>
                <h1>Username: '. $uname.'</h1>
            </li>
            <li>

```

If the user wants to update the profile, he can update the values like name, city and email id. The corresponding value will be updated in the database.

### Search Track

```

$uname = $_POST['uname'];
$key = $_POST['key'];

$query1 = "SELECT t.tname, t.aname, ab.abtitle
from tracks t join albums ab on t.abid = ab.abid
where t.tname LIKE '%". $key."%' or ab.abtitle LIKE '%". $key."%' or
t.aname LIKE '%". $key."%'";
$response1 = @mysqli_query($dbc, $query1);

echo '

```

The user can search his/her favorite, artists or tracks, the keyword is searched against all the tuples in respective tables, and then results are shown.

### Like an Artist

```
$uname = $_POST['uname'];
    $aid = $_POST['aid'];
    echo $aid;

$query3 = "insert into likes (uname, aid, ltime)
          values ('".$uname."', ".$aid.", now());";
$response3 = @mysqli_query($dbc, $query3);

$query1 = "select a.aid, a.aname, a.adesc
          from likes l join artists a on l.aid = a.aid
          where l.uname = '".$uname."'";
$response1 = @mysqli_query($dbc, $query1);

$query2 = "select a.aid, a.aname, a.adesc
          from artists a
          where a.aid not in (select a.aid
                             from likes l join artists a on l.aid = a.aid
                             where uname = '".$uname."' );";
$response2 = @mysqli_query($dbc, $query2);

echo '

```

user can also like an artist as we can see when the user clicks on like button , an insert query is generated and the entry is made in the database.

### Future consideration

1. We understand that inserting and deleting and inserting again like in the case of Follows if a User wishes to follow and unfollow and refollow using Insert, Delete and Insert isn't a good practice and as a revision shall make it try including a status variable to set status to work around this problem
2. Consecutively, we will also implement the same method for a user liking an artist.
3. We can create a trigger to notify the user every time an artist of his liking uploads a new song or album.
4. We will try to calculate the popularity of the song based on the ratings it has received, in addition to the play\_count.