CS 6083 Project #2 Report

By:

Yunjie Shi
NetID: ys3251

Shendong Yang
NetID: sy1950

Introduction

In this project, we will design and implement a dataset-backed website for a music streaming service. In general, the website will provide users with information about artists and their songs and albums.

What's more, in the second part of the project, with access to their songs via streaming, Users can generate playlists and make these available to other users, they can like artists, give ratings to songs and follow other users.

Functions

Our system has included the most majority function in Spotify. Such as registering an account; playing a track (using a Spotify web API); searching songs, artists, albums, other users, playlists created by other users using keywords, etc. For every registered user, they can log in to our system by entering the correct username and password. The user could edit their personal information, create or remove playlists, follow or unfollow other users and like or unlike an artist. Once user login to our system, there is a welcome page show the some recommend information depending what the user like. for example, in the welcome page, users could not only find the newest album published, and they will find the most recent songs from artists they like. In addition, users could browse their playing history, which they could found not only what and when they have listened, also they could know where they found this song, i.e. from a playlist, an album, searching by keyword or the welcome page recommendation. For each artist, they will have a homepage include their name, a short description, songs and albums they have published. Same as artists, each album has their own homepage, too, including the album name, and the tracks they contain. THere is also a rating system in this web application. I develop this system based on authentic property, which we only allow each user to rate a song once, but they could change their mind and give a new score.

Design

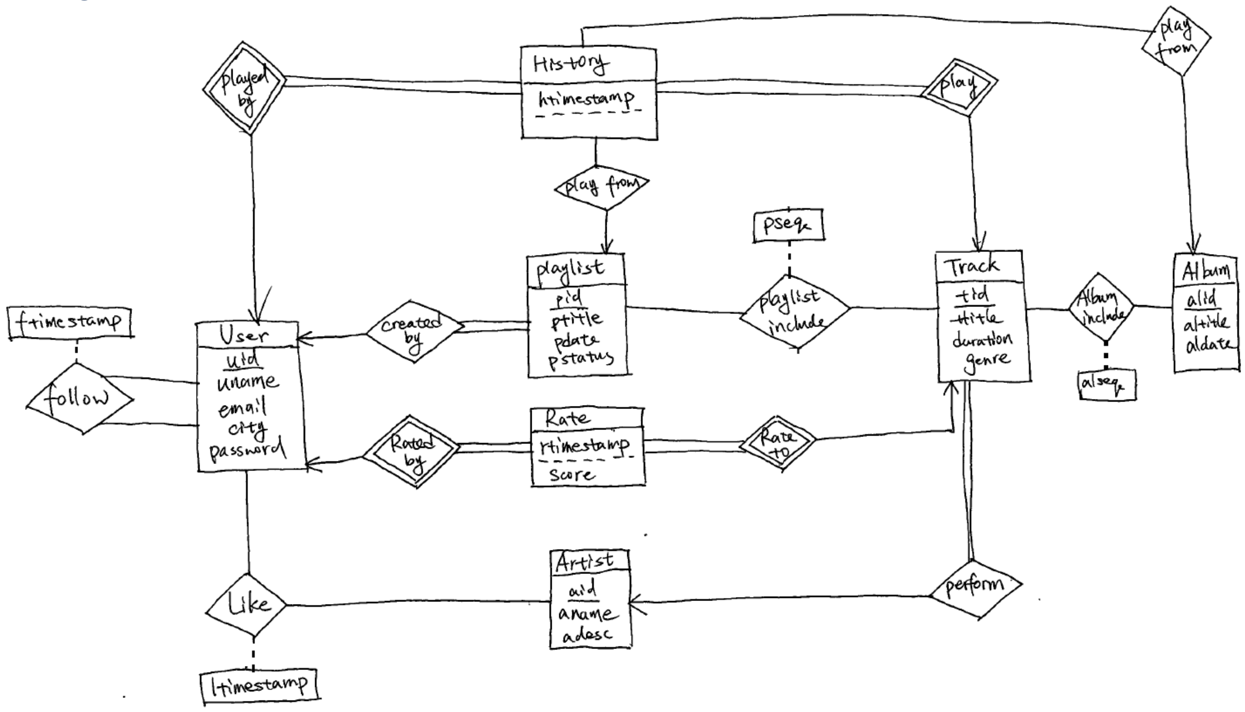
This project can be divided into two parts: the back-end and front-end. In back-end, we used MySQL database and Apache Server. It can guarantee the stability and performance of the project. In front-end, we used HTML/CSS/JS to build the website. What's more, we also used PHP to make them together.

Database

In detail, we use a \`User\` table to store information of users; each user has a unique id. \`Track\` table is used to store information of each track, here each track is shown by an artist and has a unique id. \`Artist\` table is used to store artists' information; each artist has a unique id. \`Album\` and \`Playlist\` are similar. The difference is Playlist stores the creator of each playlist. \`PlaylistInclude\` is the contents of corresponding tables. They are built to avoid repetition.

What's more, \`Favorite\` shows users' favorite artist. \`Rate\` shows users' ratings for tracks. \`Follow\` stores the relationship between users. \`History\` stores the info of users' play history. In \`History\` table, there are two keys called 'pid' and 'alid'. Those keys can be NULL and are used to implement functions in website.

ER diagram



Schemas

There are totally 10 tables.

User: {**<u>uid</u>**, uname, email, city, password}
Track: {**<u>tid</u>**¸ ttitle, duration, aname, alname}
Artist: {**<u>aid</u>**, aname, adesc}
Album: {**<u>alid</u>**, alname, altime}
Playlist: {**<u>pid</u>**, uid, ptitle, pdate, pstatus}
PlaylistInclude: {**<u>pid, tid</u>**}
Favorite: {**<u>uid, aid</u>**, ltimestamp}
Rate: {**<u>uid, tid, rtimestamp</u>**, score}
Follow: {**<u>uid, fuid</u>**, ftimestamp}
History: {**<u>uid, tid, ptimestamp</u>**, pid, alid}

Foreign key constraints

Track (\`aname\`) REFERENCES Artist (\`aname\`)
Playlist (\`uid\`) REFERENCES User (\`uid\`)
PlayListInclude (\`pid\`) REFERENCES PlayList (\`pid\`)
PlayListInclude (\`tid\`) REFERENCES Track (\`tid\`)
Favorite (\`uid\`) REFERENCES User (\`uid\`)
Favorite (\`aid\`) REFERENCES Artist (\`aid\`)
Rate (\`tid\`) REFERENCES Track (\`tid\`)
Rate (\`uid\`) REFERENCES User (\`uid\`)
Follow (\`uid\`) REFERENCES User (\`uid\`)

Follow (`fuidF`) REFERENCES User (`uid`)
History (`pid`) REFERENCES PlayList (`pid`)
History (`alid`) REFERENCES Album (`alid`)
History (`tid`) REFERENCES Track (`tid`)
History (`uid`) REFERENCES User (`uid`)

Table explanation:

User:
This table contains the information about every registered user, including their login user name, their actual name, city they live in, their email address, and the password.

Track:
Each line in this table records the information about one specific track: track id, track title, track duration, who performs this song, and which album contain this track.

Artist:
This table contains the information about each artist, including their id, their name and a short description.

Album:
This table records the information about each album, including their album id, their tittle and the publish time.

Playlist:
This table contains the information about the playlists that users created. each line of table records one playlist's id, owner's id, title, created date, and if it is a private or public playlist.

Playlistinclude:
This table contains the information about the relationship between playlist and the track.

Favorite:
This table records that the relationship between a user and an artist, and when the user like this artist.

Rate:
This table records the information about the rating system which include which song was rated, who, when and what score this song was given.

Follow:
This table has the information about the the relationship between two distinct user, and the time the connection was made.

History:

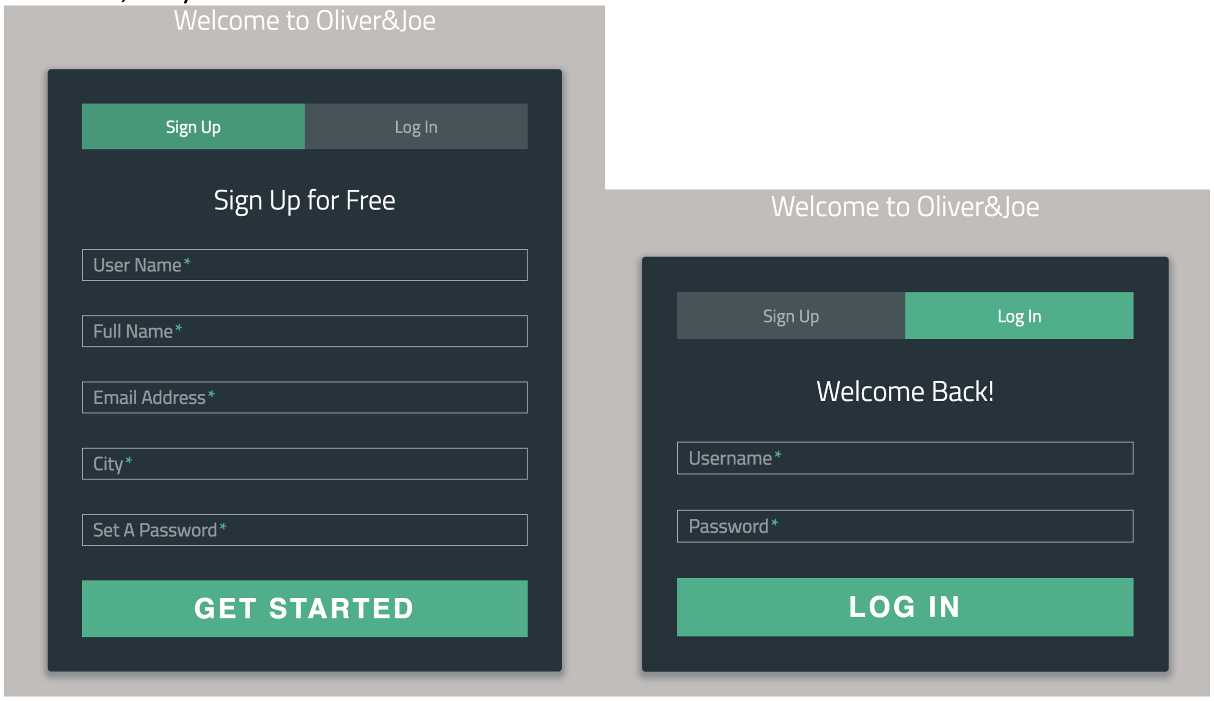
This table contains the play history of current login user include who, when and which song is played and where the song played from.

Websites

In the front end, users can sign up/log in to the "Oliver & Joe Music System", view start page, search what users want etc.

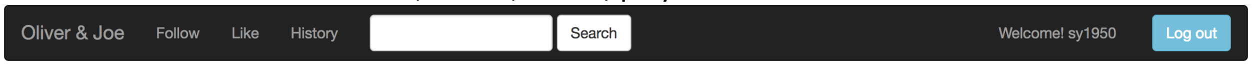
Login/Signup Page

In this page, new user need to sign up firstly. She need to fill out all information needed. If her username is conflicted with database, system will deny the access.
Old users need to log in with username and password. If the Username or password are incorrect, they are asked to refill.



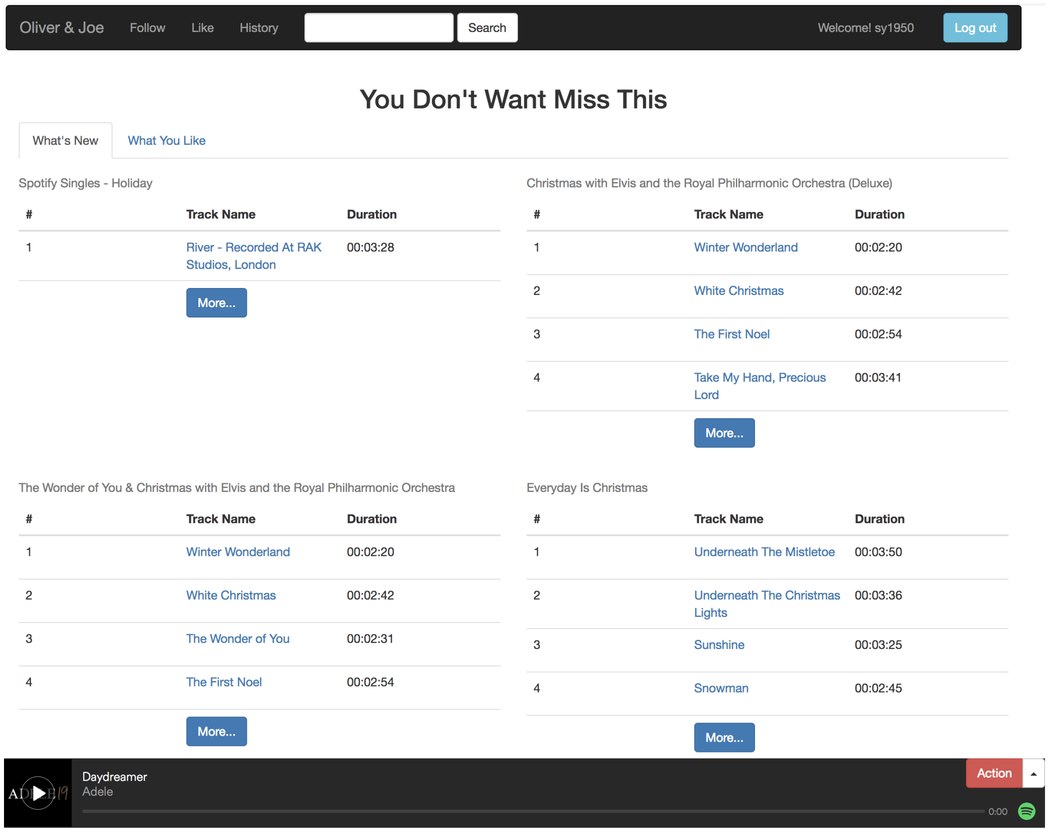
Title

Our system has a title on the top among all website.
Button 'Oliver & Joe' is redirected to the start page.
Button 'Follow' is redirected to website showing all users followed by current user.
Button 'Like' is redirected to website showing user's favorite artists.
Button 'History' is redirected to website showing user's history.
"Welcome!" will lead to profile page
"Search" is used to search users, artists, tracks, playlists and albums.



Start Page

After logging in, users can see the newest album in system and the newest Song from their favorite artists.



You Don't Want Miss This

Spotify Singles - Holiday

#	Track Name	Duration
1	River - Recorded At RAK Studios, London	00:03:28

More...

Christmas with Elvis and the Royal Philharmonic Orchestra (Deluxe)

#	Track Name	Duration
1	Winter Wonderland	00:02:20
2	White Christmas	00:02:42
3	The First Noel	00:02:54
4	Take My Hand, Precious Lord	00:03:41

More...

The Wonder of You & Christmas with Elvis and the Royal Philharmonic Orchestra

#	Track Name	Duration
1	Winter Wonderland	00:02:20
2	White Christmas	00:02:42
3	The Wonder of You	00:02:31
4	The First Noel	00:02:54

More...

Everyday Is Christmas

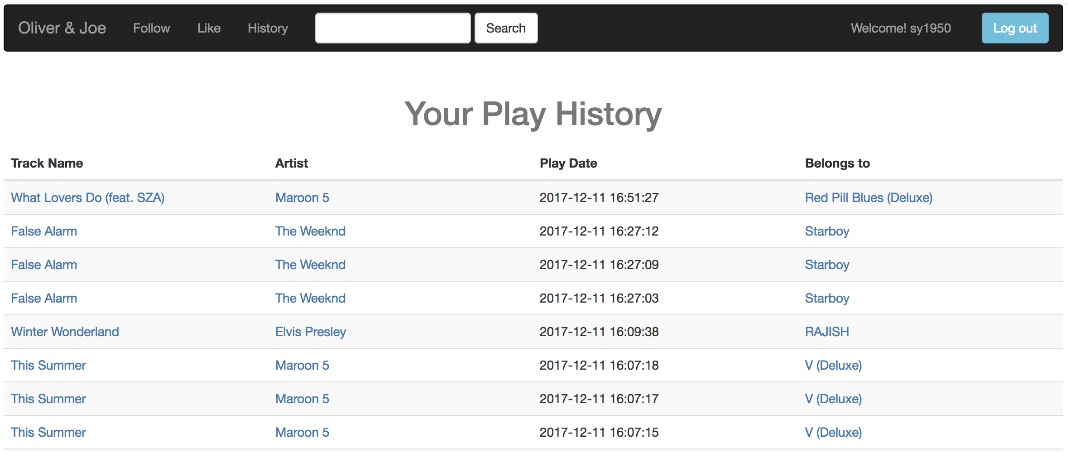
#	Track Name	Duration
1	Underneath The Mistletoe	00:03:50
2	Underneath The Christmas Lights	00:03:36
3	Sunshine	00:03:25
4	Snowman	00:02:45

More...

Daydreamer
Adele

History Page

History page lists all the user's plays history. User can find the where did she listen that track.

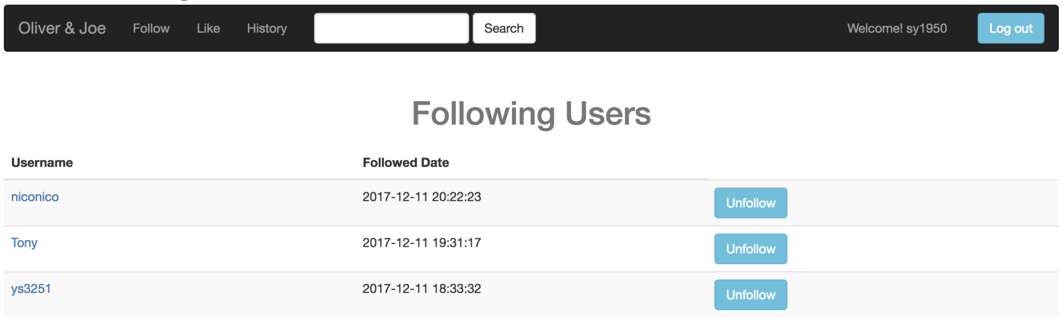


Your Play History

Track Name	Artist	Play Date	Belongs to
What Lovers Do (feat. SZA)	Maroon 5	2017-12-11 16:51:27	Red Pill Blues (Deluxe)
False Alarm	The Weeknd	2017-12-11 16:27:12	Starboy
False Alarm	The Weeknd	2017-12-11 16:27:09	Starboy
False Alarm	The Weeknd	2017-12-11 16:27:03	Starboy
Winter Wonderland	Elvis Presley	2017-12-11 16:09:38	RAJISH
This Summer	Maroon 5	2017-12-11 16:07:18	V (Deluxe)
This Summer	Maroon 5	2017-12-11 16:07:17	V (Deluxe)
This Summer	Maroon 5	2017-12-11 16:07:15	V (Deluxe)

Follow Page

Follow page list other users followed by current one. Clicking username can redirect to other users' profile. Clicking "Unfollow" can unfollow certain user.

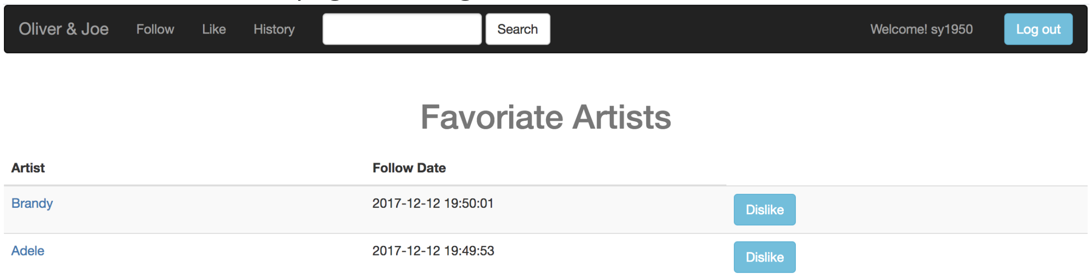


Following Users

Username	Followed Date	
niconico	2017-12-11 20:22:23	Unfollow
Tony	2017-12-11 19:31:17	Unfollow
ys3251	2017-12-11 18:33:32	Unfollow

Favorite Page

Similar with Follow page, favorite page lists all artists liked by user. Clicking artist name can redirect to other artist homepage. Clicking "Dislike" can dislike certain artist.

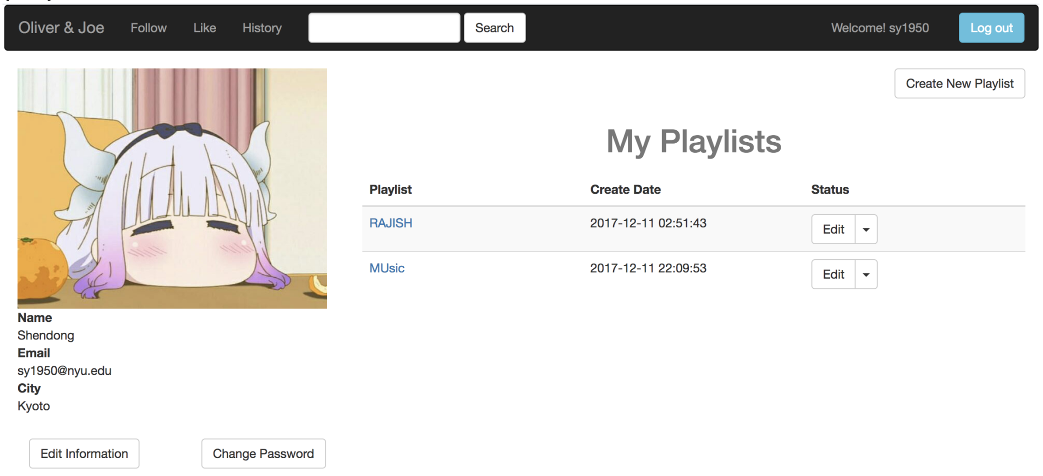


Favoriate Artists

Artist	Follow Date	
Brandy	2017-12-12 19:50:01	Dislike
Adele	2017-12-12 19:49:53	Dislike

User Profile

In user profile, user can edit information like Full name, email and city. She can also create and remove playlists.



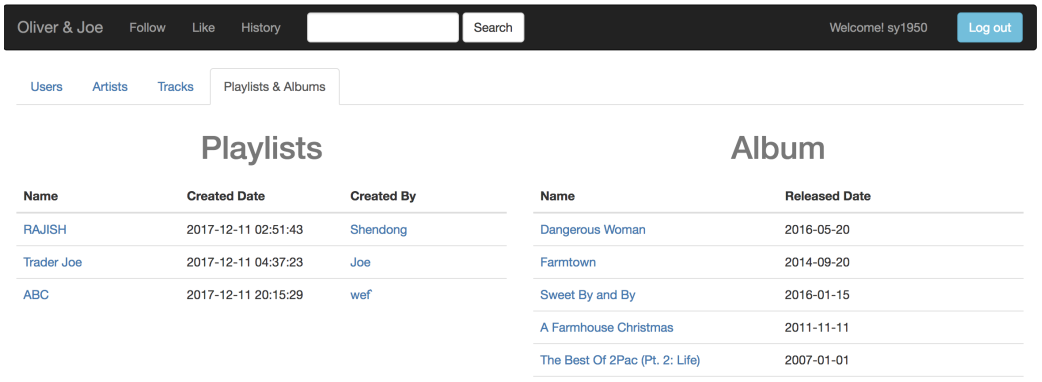
My Playlists

Playlist	Create Date	Status
RAJISH	2017-12-11 02:51:43	Edit
MUsic	2017-12-11 22:09:53	Edit

Name
Shendong
Email
sy1950@nyu.edu
City
Kyoto

Search Page

User can search whatever her want. The search page will display Users, Artists, Tricks and Playlists & Albums.



Playlists

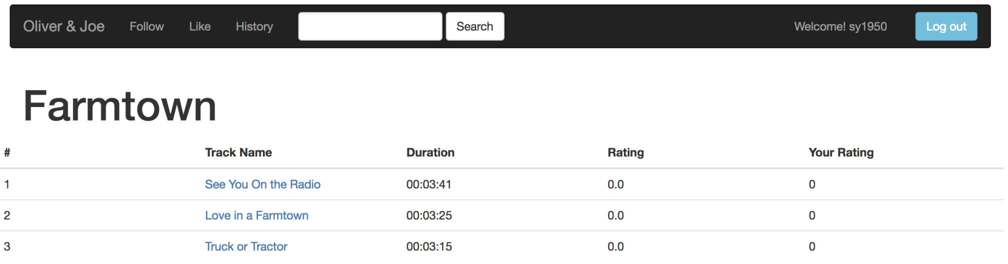
Name	Created Date	Created By
RAJISH	2017-12-11 02:51:43	Shendong
Trader Joe	2017-12-11 04:37:23	Joe
ABC	2017-12-11 20:15:29	wef

Album

Name	Released Date
Dangerous Woman	2016-05-20
Farmtown	2014-09-20
Sweet By and By	2016-01-15
A Farmhouse Christmas	2011-11-11
The Best Of 2Pac (Pt. 2: Life)	2007-01-01

Album Page

Album page shows the content of one album. User can see the overall rate and her rate of each song in this album.

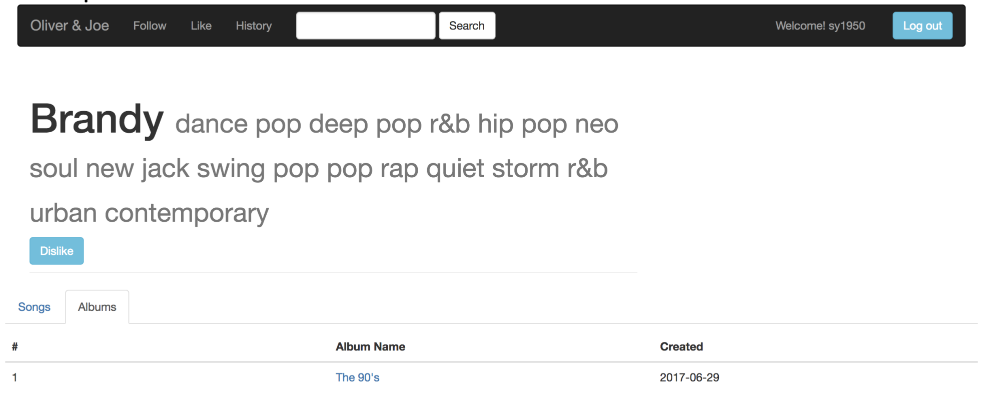


Farmtown

#	Track Name	Duration	Rating	Your Rating
1	See You On the Radio	00:03:41	0.0	0
2	Love in a Farmtown	00:03:25	0.0	0
3	Truck or Tractor	00:03:15	0.0	0

Artist Page

In artist page, user can check the detail information of one artist, including the tracks and albums she took part in.



Brandy dance pop deep pop r&b hip pop neo soul new jack swing pop pop rap quiet storm r&b urban contemporary

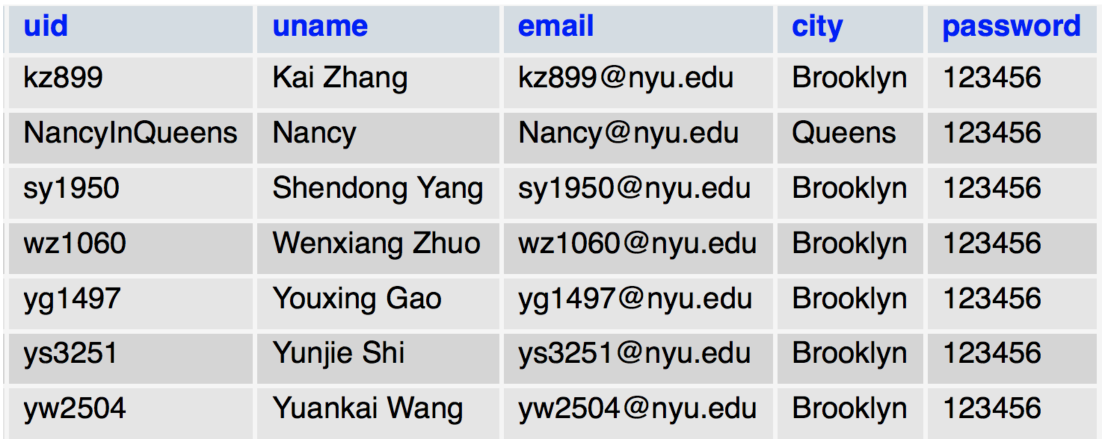
#	Album Name	Created
1	The 90's	2017-06-29

Test

Database Test

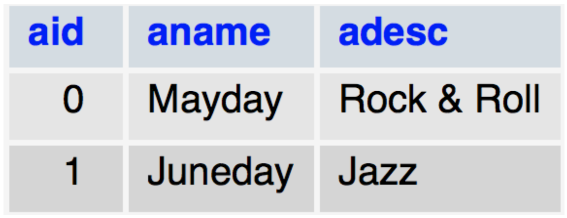
Before implement the website functions, we did some tests of database

-- ----------------------------
-- Records of User
-- ----------------------------



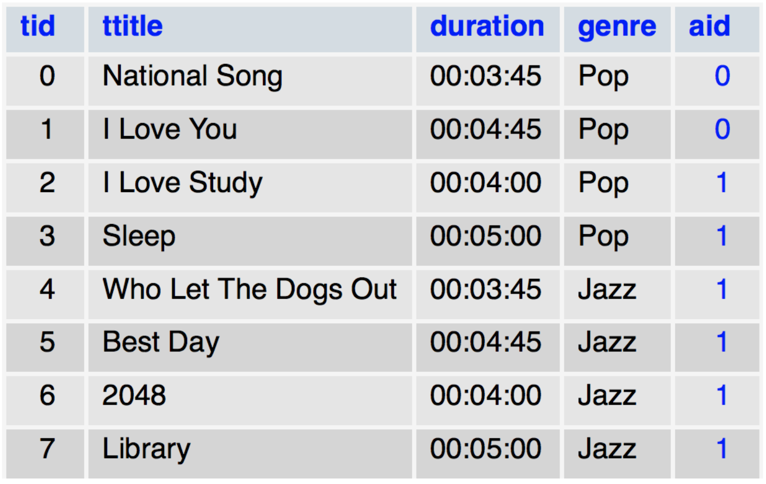
uid	uname	email	city	password
kz899	Kai Zhang	kz899@nyu.edu	Brooklyn	123456
NancyInQueens	Nancy	Nancy@nyu.edu	Queens	123456
sy1950	Shendong Yang	sy1950@nyu.edu	Brooklyn	123456
wz1060	Wenxiang Zhuo	wz1060@nyu.edu	Brooklyn	123456
yg1497	Youxing Gao	yg1497@nyu.edu	Brooklyn	123456
ys3251	Yunjie Shi	ys3251@nyu.edu	Brooklyn	123456
yw2504	Yuankai Wang	yw2504@nyu.edu	Brooklyn	123456

-- ----------------------------
-- Records of Artist
-- ----------------------------



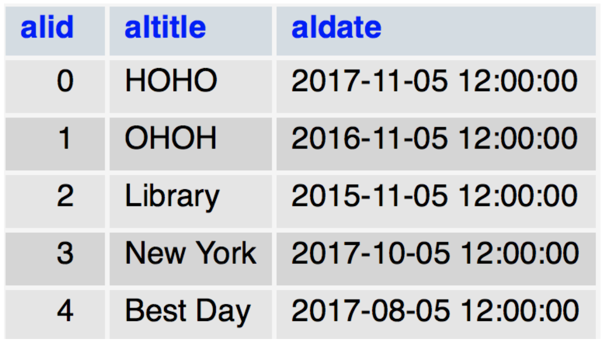
aid	aname	adesc
0	Mayday	Rock & Roll
1	Juneday	Jazz

-- ----------------------------
-- Records of Track
-- ----------------------------



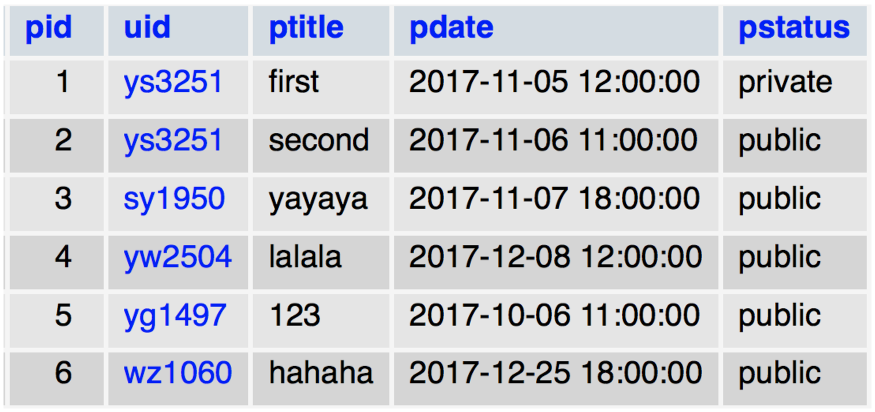
tid	ttitle	duration	genre	aid
0	National Song	00:03:45	Pop	0
1	I Love You	00:04:45	Pop	0
2	I Love Study	00:04:00	Pop	1
3	Sleep	00:05:00	Pop	1
4	Who Let The Dogs Out	00:03:45	Jazz	1
5	Best Day	00:04:45	Jazz	1
6	2048	00:04:00	Jazz	1
7	Library	00:05:00	Jazz	1

```
-- ----------------------------
-- Records of Album
-- ----------------------------
```



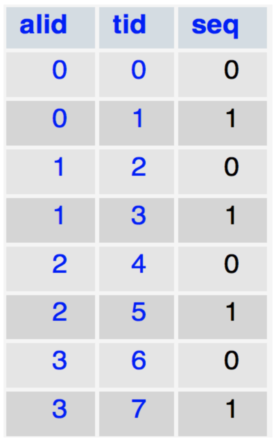
alid	altitle	aldate
0	HOHO	2017-11-05 12:00:00
1	OHOH	2016-11-05 12:00:00
2	Library	2015-11-05 12:00:00
3	New York	2017-10-05 12:00:00
4	Best Day	2017-08-05 12:00:00

```
-- ----------------------------
-- Records of PlayList
-- ----------------------------
```



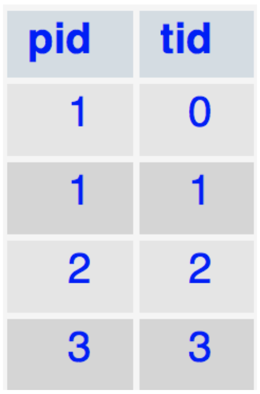
pid	uid	ptitle	pdate	pstatus
1	ys3251	first	2017-11-05 12:00:00	private
2	ys3251	second	2017-11-06 11:00:00	public
3	sy1950	yayaya	2017-11-07 18:00:00	public
4	yw2504	lalala	2017-12-08 12:00:00	public
5	yg1497	123	2017-10-06 11:00:00	public
6	wz1060	hahaha	2017-12-25 18:00:00	public

```
-- ----------------------------
-- Records of AlbumInclude
-- ----------------------------
```



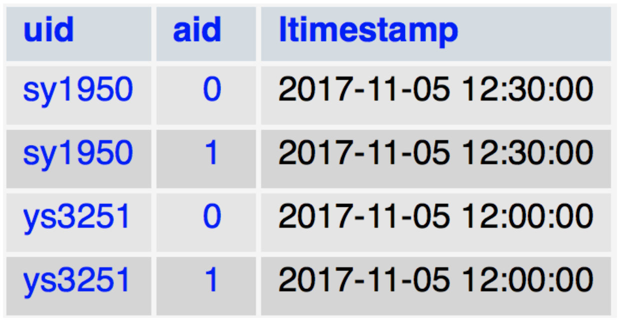
alid	tid	seq
0	0	0
0	1	1
1	2	0
1	3	1
2	4	0
2	5	1
3	6	0
3	7	1

```
-- ----------------------------
-- Records of PlayListInclude
-- ----------------------------
```



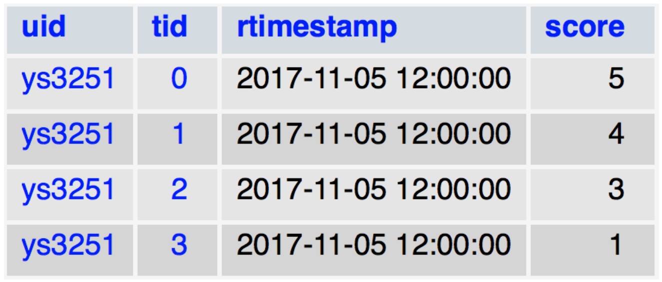
pid	tid
1	0
1	1
2	2
3	3

```
-- ----------------------------
-- Records of Favorite
-- ----------------------------
```



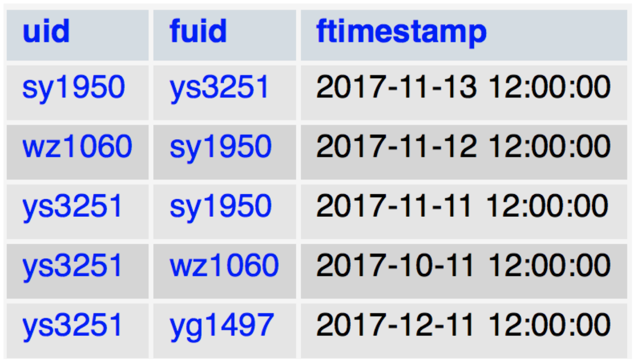
uid	aid	ltimestamp
sy1950	0	2017-11-05 12:30:00
sy1950	1	2017-11-05 12:30:00
ys3251	0	2017-11-05 12:00:00
ys3251	1	2017-11-05 12:00:00

```
-- ----------------------------
-- Records of Rate
-- ----------------------------
```



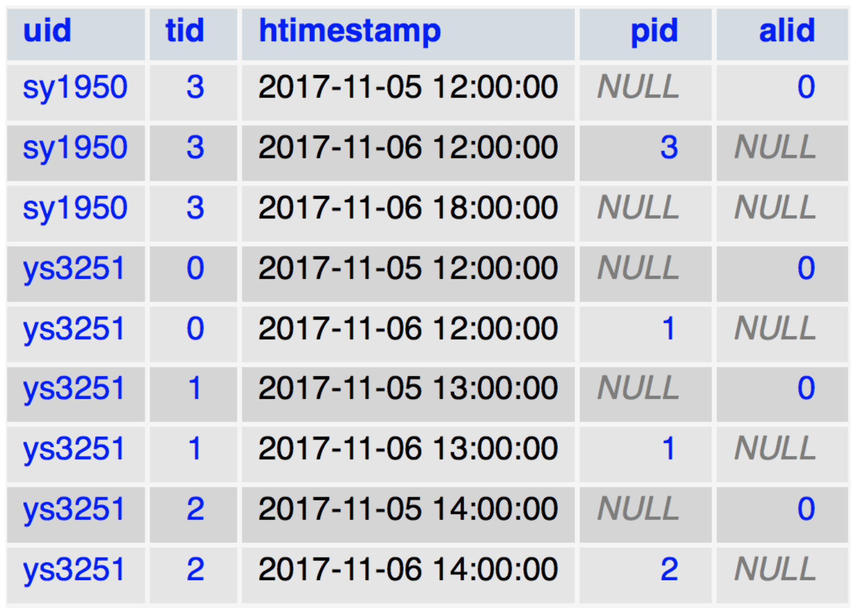
uid	tid	rtimestamp	score
ys3251	0	2017-11-05 12:00:00	5
ys3251	1	2017-11-05 12:00:00	4
ys3251	2	2017-11-05 12:00:00	3
ys3251	3	2017-11-05 12:00:00	1

```
-- ----------------------------
-- Records of Follow
-- ----------------------------
```



uid	fuid	ftimestamp
sy1950	ys3251	2017-11-13 12:00:00
wz1060	sy1950	2017-11-12 12:00:00
ys3251	sy1950	2017-11-11 12:00:00
ys3251	wz1060	2017-10-11 12:00:00
ys3251	yg1497	2017-12-11 12:00:00

```
-- ----------------------------
-- Records of History
-- ----------------------------
```



uid	tid	htimestamp	pid	alid
sy1950	3	2017-11-05 12:00:00	NULL	0
sy1950	3	2017-11-06 12:00:00	3	NULL
sy1950	3	2017-11-06 18:00:00	NULL	NULL
ys3251	0	2017-11-05 12:00:00	NULL	0
ys3251	0	2017-11-06 12:00:00	1	NULL
ys3251	1	2017-11-05 13:00:00	NULL	0
ys3251	1	2017-11-06 13:00:00	1	NULL
ys3251	2	2017-11-05 14:00:00	NULL	0
ys3251	2	2017-11-06 14:00:00	2	NULL

Query & Results

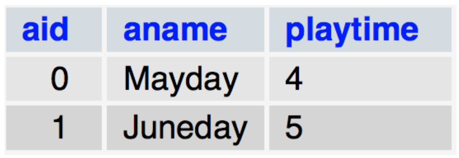
(1) Create a record for a new user account, with a name, a login name, and a password.

```
INSERT INTO `User` VALUES ('NancyInQueens', 'Nancy', 'Nancy@nyu.edu', 'Queens', '123456');
```

(2) For each artist, list their ID, name, and how many times their tracks have been played by users.

```
select aid, aname, playtime
from artist natural join
        (select aid, sum(count) as playtime
   from track natural join
                 (SELECT tid, count(*) as count
                 From history
                 group by tid) as temp1
        group by aid) as temp2
```

Result:

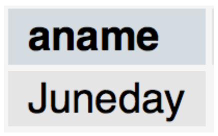


aid	aname	playtime
0	Mayday	4
1	Juneday	5

(3) List all artists that are mainly playing Jazz, meaning that at least half of their tracks are of genre Jazz.

```
select aname
from artist as a
where (select count(*)
       from track
       where genre = 'Jazz' and track.aid = a.aid) * 2 >= (select count(*)
              from track
              where track.aid = a.aid);
```

Result:



aname
Juneday

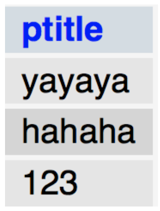
(4) Insert a new rating given by a user for a track.

```
INSERT INTO rate VALUES ('NancyInQueens', 0, '2017-11-05 12:00:00', 5);
```

(5) For a particular user, say "NancyInQueens", list all playlists that were made by users that she follows.

```
select ptitle
from playlist, (select fuid
       from follow
       where uid = 'ys3251') as temp
where fuid = uid
```

Result:

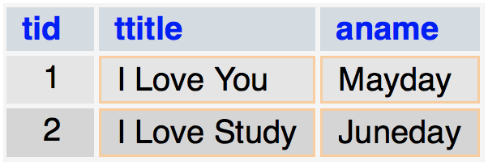


ptitle
yayaya
hahaha
123

(6) List all songs where the track title or artist title matches some set of keywords (if possible, use ``contains'', or otherwise ``like'', for this query).

```
select tid, ttitle, aname
from track natural join artist
```

where track.ttitle like '%Love%' or artist.aname like '%Love%'
Result:

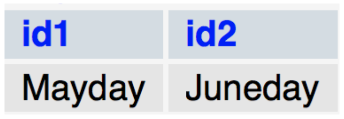


tid	ttitle	aname
1	I Love You	Mayday
2	I Love Study	Juneday

(7) Find pairs of related artists, where two artists are related if they have many fans in common. (Define this appropriately.)

```
create view temp as
        (select * from favorite natural join artist);

select temp.id1, temp.id2
from (select f1.aname as id1, f2.aname as id2, count(*) as count
        from temp as f1, temp as f2
        where f1.aid < f2.aid and f1.uid = f2.uid) as temp
where temp.count >= 2;

drop view if exists temp;
```

Result:



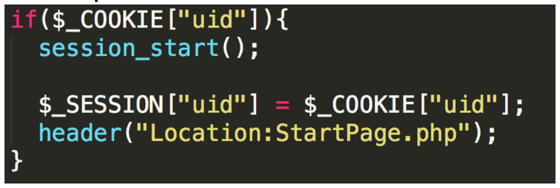
id1	id2
Mayday	Juneday

Note: For test purpose, we regard two common fans as related. Users can setup a larger number if necessary.

Highlights

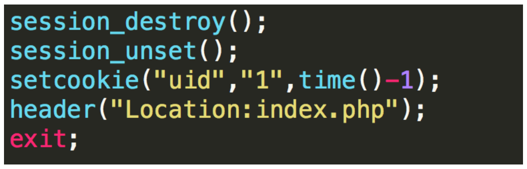
Cookie

we used cookies in our codes. After log into the system, they can log in again in five minutes without password.
In login page, if the system detect cookie in browser, user don't need to log in again. If not, user need to login with username and password while a 5-minute survival cookie is set.



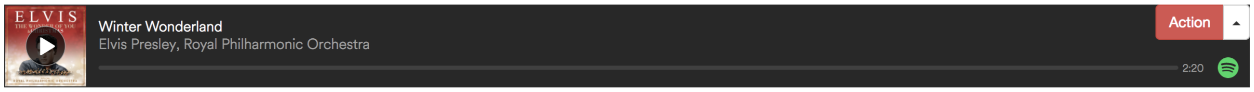
```
if($_COOKIE["uid"]){
  session_start();

  $_SESSION["uid"] = $_COOKIE["uid"];
  header("Location:StartPage.php");
}
```

In logout page, system will destroy session and cookie.



```
session_destroy();
session_unset();
setcookie("uid","1",time()-1);
header("Location:index.php");
exit;
```

Play music demo



We found this single track & compact widget from the spotify developer website(https://developer.spotify.com/technologies/widgets/examples/). Every time we click on a track link, we will pass the corresponding track id to the session which will be read by the widget and play the song.
The button in top-right corner is an draw-down menu that user could choose two operation: "Add song to my playlist" and "Rate this song":

Add song to my playlist:
After selecting this option, a modal will pop out and ask user to choose an existing playlist, and then the current playing song will be saved in the desired playlist. In this system, user is not allowed to add a track into one playlist twice, if user try to do that, a warning will pop out and tell the user the track already exists.

Rate this song:
After selecting this option, a modal will pop out and ask user to choose 1-5 point for selecting track. If user has already rated, then the option will change to "re-Rate this song", the the new rating score will overwrite the original score. The reason to have "re-rate" function is in order to avoid user "Click Farm" and make the rating system more authentic.

Conclusion

In conclusion, we designed and implemented a music web application. In back-end, we created MySQL database to store the information of user, artist, album and tracks. We also create tables for the relationships, including play history, like and follow functions. Our database is keeping with First and Third Normal Form. We created sample data and tested the asking queries. The functions of our design work well.
In front-end, users can generate playlists and make these available to other users; like artists, give ratings to songs and follow other users; search what they want and play music demo via Spotify API.

Appendix:

SQL Query for Each Function

Sign up:
```
INSERT INTO `User` (`uid`, `uname`, `email`, `city`, `password`) VALUES ( '".$uname."',
'".$name."', '".$email."', '".$city."', '".$pwd."')
```

Log in:
```
SELECT *
FROM User
where uid =  \"$uid\" and password = ".$pwd." limit 1
```

Search:

User:
```
SELECT fuid, uname, city, uid
FROM (SELECT uid as fuid, uname, city
        FROM User
        WHERE (uid LIKE \"%$key%\" or uname LIKE \"%$key%\") and uid <> \"$uid\") as A
        NATURAL LEFT JOIN (SELECT  fuid, uid
                                  FROM Follow
                                  WHERE uid = \"$uid\") as B
```

Artist:
```
SELECT aid, aname, adesc
FROM Artist
WHERE aname LIKE \"%$key%\" or adesc LIKE \"%$key%\"
```

Album:
```
SELECT *
FROM album
WHERE alnameLIKE \"%$key%\"
```

Playlist:
```
SELECT *
FROM playlist NATURAL LEFT JOIN user
WHERE ptitle LIKE \"%$key%\" and pstatus = 1
```

Welcome page:

Find newest albums:
```
SELECT *
From album
ORDER BY `altime` DESC;
```

Find newest songs:
```
SELECT aname, alname, tname, tduration, tid, alid
```

```
FROM favorite NATURAL JOIN artist NATURAL JOIN track NATURAL JOIN album
WHERE uid = '$uid' and TIMESTAMPDIFF(DAY,now(), altime) <= 35
ORDER BY `altime` DESC;
```

User profile:

retrieve user profile:

```
SELECT*
FROM User
WHERE uid = '".$otheruid."'
```

retrieve playlist:

(view own playlist)

```
SELECT pid, ptitle, pdate, pstatus
FROM Playlist
WHERE uid = '$uid’
```

(view other playlist)

```
SELECT pid, ptitle, pdate, pstatus
FROM Playlist
WHERE uid = '$otheruid' and pstatus = 1
```

edit personal info

```
UPDATE user
SET uname = '$uname',
    city = '$city',
    email = '$email'
WHERE uid = '$uid'
```

create playlist:

```
INSERT INTO `PlayList` (uid, ptitle, pdate, pstatus) VALUES ('$uid', '$ptitle', '$date', '$pstatus')
```

remove playlist:

```
DELETE FROM PlaylistInclude WHERE pid = \"pid\"
DELETE FROM Playlist WHERE pid = \"$pid\"
```

Follow:

```
INSERT INTO `Follow` VALUES ('$uid', '$fuid', '$time')
```

Unfollow:

```
DELETE FROM `Follow`
WHERE uid = \"$uid\" and fuid = \"$fuid\"
```

Like:

```
INSERT INTO `Favorite` VALUES ('$uid', '$aid', '$time')
```

Unlike:
DELETE FROM Favorite
WHERE uid = '$uid' and aid = '$aid'

Rate:
Get own rate:
SELECT *
FROM rate
WHERE uid = '$uid'

Get average score:
Album:
SELECT distinct track.tid, track.tname, track.tduration, round(ifnull(avg(score),0),1) as avgscore
FROM Track NATURAL JOIN Album LEFT OUTER JOIN rate on track.tid = rate.tid
WHERE alname = '$alname'
GROUP BY track.tid;

Playlist:
SELECT distinct track.tid, track.tname, track.tduration, round(ifnull(avg(score),0),1) as avgscore
FROM Track NATURAL JOIN playlist NATURAL JOIN playlistinclude LEFT OUTER JOIN rate on track.tid = rate.tid
WHERE alname = '$alname'
GROUP BY track.tid;

Rate score:
INSERT INTO rate VALUES ('$uid', '$tid', now(),'$score')

re-Rate score:
Delete FROM rate WHERE uid = '$uid' and tid = '$tid'
INSERT INTO rate VALUES ('$uid', '$tid', now(),'$score')

Artist Page:
Artist info:
SELECT * FROM Artist WHERE aid = '$aid'

Tracks by artist:
SELECT *
FROM artist NATURAL JOIN album NATURAL JOIN Track
WHERE aid = '$aid'
GROUP BY alname

Album by artist:
SELECT * FROM Album WHERE alid = '$alid'

Album Page:
Album info:
SELECT * FROM Album WHERE alid = '$alid'
SELECT distinct track.tid, track.tname, track.tduration, ROUND(IFNULL(avg(score),0),1) as avgscore
FROM Track NATURAL JOIN Album LEFT OUTER JOIN rate on track.tid = rate.tid
WHERE alname = '$alname'
GROUP BY track.tid;

History:
Add to history:
From Album:
INSERT INTO history VALUES ('$uid', '$tid', now(), NULL, '$alid')

From Playlist:
INSERT INTO history VALUES ('$uid', '$tid', now(), '$pid', NULL)

others:
INSERT INTO history VALUES ('$uid', '$tid', now(), NULL, NULL)

Show history:
SELECT history.tid, htimestamp, tname, artist.aid, artist.aname, history.alid, history.pid
FROM history join track on history.tid = track.tid join artist on artist.aname = track.aname
WHERE history.uid = '$uid'
ORDER BY htimestamp desc;

from album:
SELECT alname
FROM album
WHERE alid = '".$row["alid"]."

from playlist:
SELECT ptitle
FROM playlist
]WHERE pid = '".$row["pid"]."'

others:
NULL