

Project #2

Wei Wang - ww1306, Yijun Chen - yc3166

Part 1. Introduction

Purpose

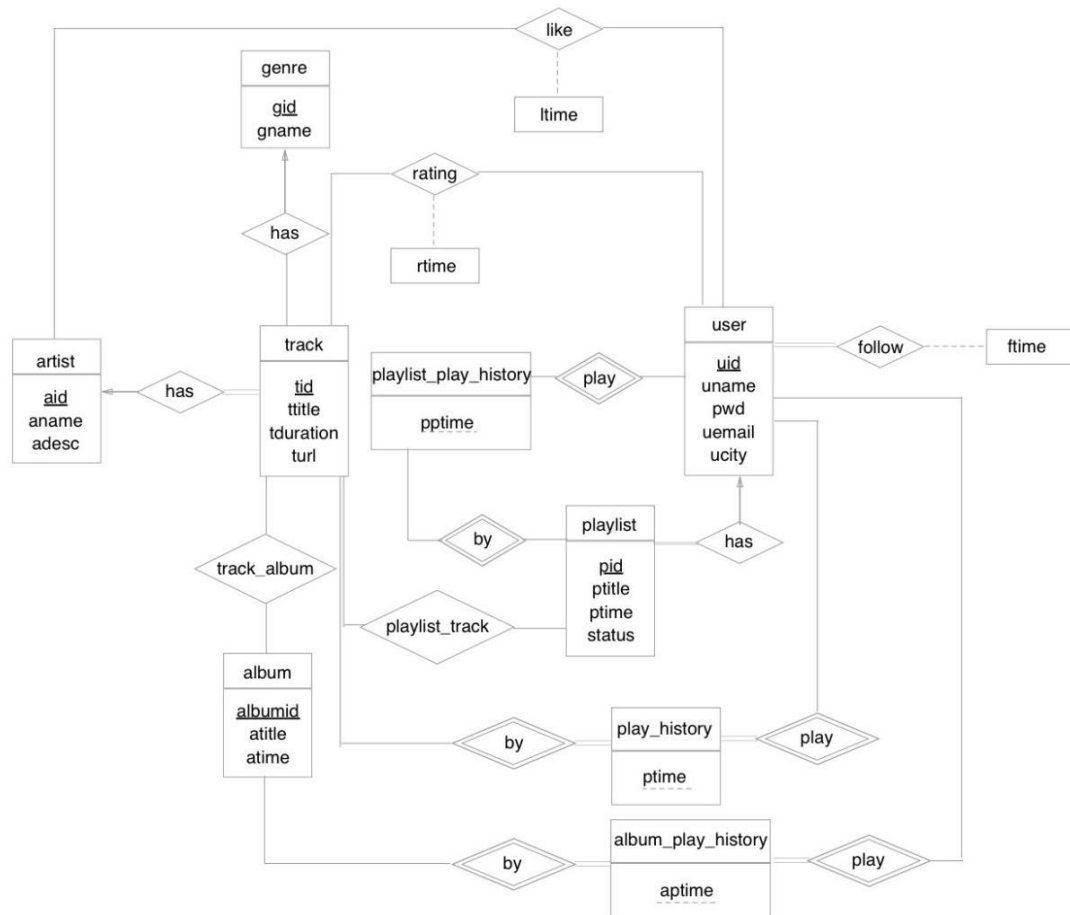
The purpose of this document is to illustrate how the 'wemusic' website is designed which mainly includes two parts: the design of database and the front end web pages design. The website is a music player platform implemented with API of Spotify, where users can sign up, create a profile, log in, play tracks, make playlists, like an artist, rate songs, follow other users, browse and search for musics, artists and playlists.

Technology

In this project, we use MySQL as the relational database system. We apply Apache as the Server. The front end and back end will be implemented by PHP, Javascript, HTML/CSS. We apply Bootstrap framework and jQuery for the front end development.

Part 2. Database Design

ER diagram of the website



The assumptions are taken into consideration:

1. Generate playlist (two types: private or public).
2. User has own profile (name, email, city) (create, edited)
3. Artist have many tracks (trackId, ArtistId, title, duration, genre)
4. Artist (artistId, artistname, description) (single, duo, band, orchestra)
5. Album (albumId, title, date) (contains different artists)
6. Playlist (userid, date) (private or public) similar to album but it is created by users.
7. Users may also like an artist, rate a song (from 1 to 5 stars) and follow a user (need timestamp)

8.Recent Play (whenever a user plays a song, the database should keep track of this information if user decide to store this information in order to find out which playlist or album can get a lot of plays).

9.Playlist can only be modified by the owner, playlist cannot be followed.

10.When a track is been playing, user can decide whether store it in a particular playlist.

Relational Schema

1. artist(aid, aname, adesc)

Artist table is to store the artist's information, including their id, name, simple description.

2. track(tid, aid, ttitle, tduration, gid, turl)

Track table includes basic information of a track, containing id, aid(identify whose song), title, duration, gid(the genre of the track), turl (store the url of the track, because the back end needs to know where it can get the track and play it)

Foreign key track(aid) references artist(aid)

Foreign key track(gid) references genre(gid)

3. album(albumid, atitle, atime)

Album table contains the id of the album, title, and time it released.

4. genre(gid, gname)

Genre table contains the id of each genre and its name.

5. user(uid, unname, pwd, uemail, ucity)

User table includes user's id, name, password, email and city.

6. playlist(pid, ptitle, uid, ptime, status)

Playlist table includes the id, title of the playlist. Uid identifies who crates it. Ptime is the time of creation. Status has two values: 1 means it is private while 0 means public.

Foreign key playlist(uid) references user(uid)

7. track_album(tid, albumid)

Track_album is a relation table shows the track belongs to which album.

Foreign key track_album(tid) references track(tid)

Foreign key track_album(albumid) references album(albumid)

8. playlist_track(pid, tid)

Playlist_track is a relation table shows the tracks of each playlist.

Foreign key playlist_album(pid) references playlist(pid)

Foreign key playlist_album(tid) references track(tid)

9. rating(uid, tid, star, rtime)

Rating table records the information of rating(who at what time rated which track with how many stars (from 1-5))

Foreign key rating(uid) references user(uid)

Foreign key rating(tid) references track(tid)

10. like(uid, aid, ltime)

Like table records the information of who liked which artist at what time.

Foreign key like(uid) references user(uid)

Foreign key like(aid) references artist(aid)

11. follow(uid, uid, ftime)

Follow table records the information of who followed whom at what time.

Foreign key follow(uid) references user(uid) (both uid)

12. play_history(uid, tid, ptime)

Play_history table records the information of who listened which track (does not contain in any playlists or albums)at what time.

Foreign key play_history(uid) references user(uid)

Foreign key play_history(tid) references track(tid)

13. playlist_play_history(uid, pid, pptime)

Foreign key playlist_play_history(uid) references user(uid)

Foreign key playlist_play_history(pid) references playlist(pid)

14. album_play_history(uid, albumid, aptime)

Foreign key album_play_history(uid) references user(uid)

Foreign key album_play_history(albumid) references album(albumid)

Description of Table

1. Description of Artist Table

Column	Type	Description
aid	varchar(255)	id of artist (unique)
aname	varchar(255)	name of artist
adesc	varchar(255)	description of artist

2. Description of Track Table

Column	Type	Description
tid	varchar(255)	id of track (unique)
aid	varchar(255)	foreign key refers to Artist(aid)
ttitle	varchar(255)	title of track
tduration	varchar(6)	duration of a track
gid	int(11)	foreign key refers to Genre(gid)
turl	varchar(255)	url of a track

3. Description of Album Table

Column	Type	Description
albumid	varchar(255)	id of album (unique)
atitle	varchar(255)	title of album
adate	date	album released time

4. Description of Genre Table

Column	Type	Description
gid	int(11)	id of genre (unique)
gname	varchar(255)	name of genre

5. Description of User Table

Column	Type	Description
uid	int(11)	ID of user (unique)
uname	varchar(255)	real name of user
loginname	varchar(255)	login name of user (unique)
pwd	varchar(255)	password

uemail	varchar(255)	email of user
ucity	varchar(255)	city of user

6. Description of Playlist Table

Column	Type	Description
pid	int(11)	id of playlist (unique)
ptitle	varchar(255)	name of playlist
uid	int(11)	foreign key refers to User(uid)
ptime	datetime	create time of playlist
status	int(11)	0 for public, 1 for private

7. Description of Track_album Table

Column	Type	Description
tid	varchar(255)	foreign key refers to Track(tid)
albumid	varchar(255)	foreign key refers to Album(albumid)

8. Description of Playlist_track Table

Column	Type	Description
pid	int(11)	foreign key refers to Playlist(pid)
tid	varchar(255)	foreign key refers to Track(tid)

9. Description of Rating Table

Column	Type	Description
uid	int(11)	foreign key refers to User(uid)
tid	varchar(255)	foreign key refers to Track(tid)
star	int(11)	score for each track(star from 1 to 5)
rtime	datetime	time of the rate action

10. Description of Like Table

Column	Type	Description
uid	int(11)	foreign key refers to User(uid)
aid	varchar(255)	foreign key refers to Artist(aid)
ltime	datetime	time of the like action

11. Description of Follow Table

Column	Type	Description
uid1	int(11)	foreign key refers to User(uid), follower's uid
uid2	int(11)	foreign key refers to User(uid), following's uid
ftime	datetime	time of the follow action

12. Description of Play_history Table

Column	Type	Description
uid	int(11)	foreign key refers to User(uid)
tid	varchar(20)	foreign key refers to Track(tid)
playtime	datetime(6)	time of play action

13. Description of Playlist_play_history Table

Column	Type	Description
uid	int(11)	foreign key refers to User(uid)
pid	int(11)	foreign key refers to Play(pid)
pptime	datetime(6)	time of play action

14. Description of Album_play_history Table

Column	Type	Description
uid	int(11)	foreign key refers to User(uid)
albumid	varchar(255)	foreign key refers to Album(albumid)
aptime	datetime(6)	time of play action

Basic Functions

The UI of the website is like Spotify.

The basic functions of the websites:

1. Sign in / Sign out / Sign up (store login information in session and create cookie)
2. Play tracks (tracks have been already added into database before using Spotify API)
3. Record the play history of every user, including playing tracks, albums and playlist history.
4. User can create play lists and add tracks to play lists. Each play list has two types: private(open to the owner) and public (open to all users)
5. User could press 'like' button to like artists and then they could see the artists they like.
6. User can follow other users.
7. In the Artist section, user could see the tracks of that artist.
8. In the Album section, user could see all the tracks contained.
9. User could search artists, tracks, play lists (only public play lists) and albums.
10. At user's home page, users could see the new releases of the artists they liked and the new play lists of users they followed

Part 3. System Architecture

The main programming language we use to design the wemusic project is PHP.

Here is the architecture of our system.

1. Front End

Basic UI design like Spotify, containing a sidebar to record the Artists and Users you follow, a footer that is a Audio Player accomplished by H5 tag <audio>. Simply use jQuery to manipulate the audio by `audio.play()` and `audio.pause()`. And a content area to show the main pages of the web player (display the Artists, Users, Albums, Play History and Home page).

2. Back End

Several Controllers could deal with the data sent by Front-End (using Ajax or Form) and send to Front - End or do SQL queries to store data to the database.

3. Database Management System

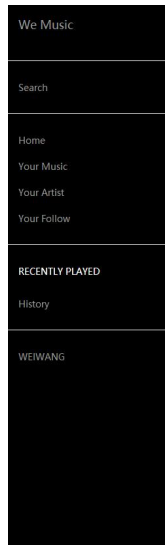
Use MySQL to store the data. The tables have been shown above.

Front End Design:

Frameworks: Bootstrap Library: jQuery

We have 3 different parts for every front end pages.

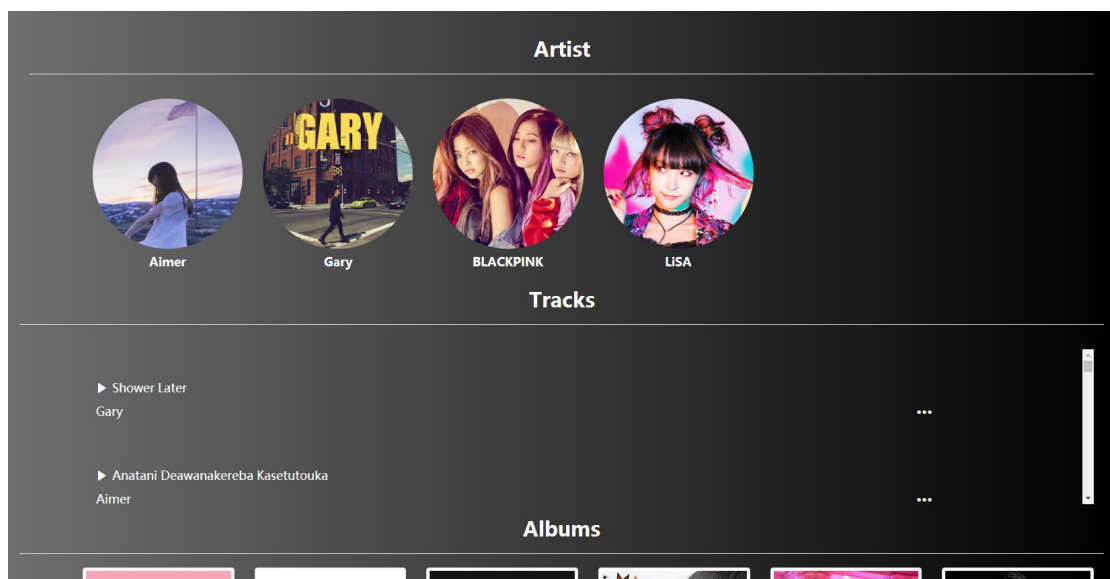
1. Sidebar:



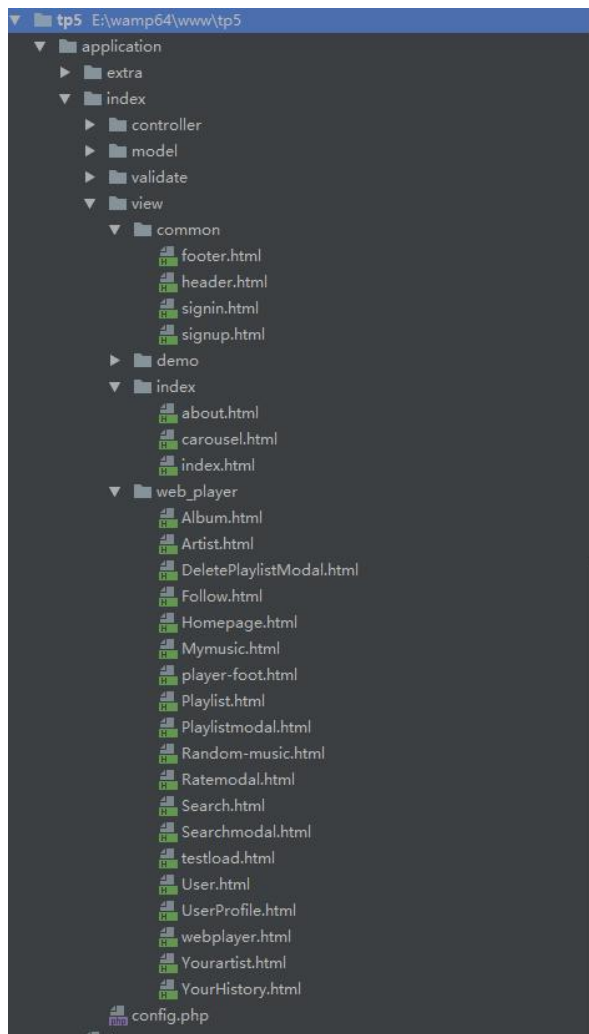
2. Footer: A H5 Audio Player; Reading the url provided by Spotify API.



3. Content Area: Display several pages



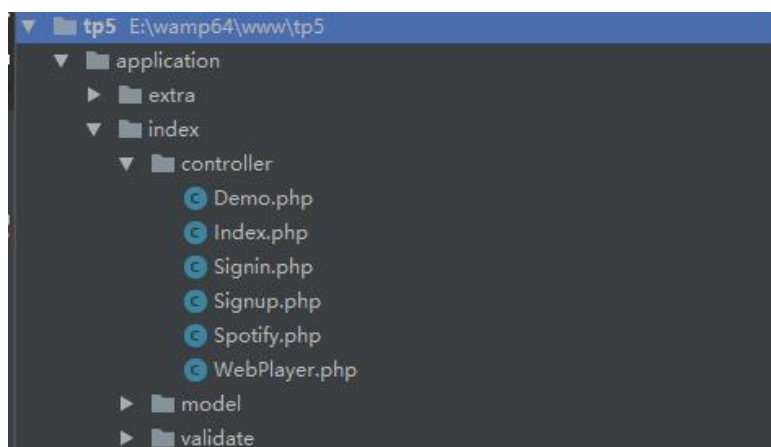
Front End Files Structure:



Back End Design

Frameworks: thinkphp5 - A simple MVC framework

Back End files structure:



Part 4. Security & Concurrency & Using Spotify API

Protection of SQL injection.

We have two ways to avoid SQL injection. One of them is to use PDO. Instead of string connection, PDO uses prepared statements quires which are generally sufficient to prevent injection.

```
$query = "SELECT * FROM `user` WHERE loginname = :loginname  
AND pwd = :pwd;";  
  
$result = $pdo->prepare($query);  
  
$result->execute(array(':loginname'=>$loginname,':pwd'=>$pwd1));  
  
$result = $result->fetchAll(PDO::FETCH_NUM);
```

And the second way is to used build-in function to do the queries:

```
db('user')->insert($data)
```

Password Encryption:

We apply md5 for encryption

```
$pwd1 = md5($pwd);
```

Concurrency is implemented by using Session. We keep different sessions record on different user (using different browser).

```
if($result){  
    session_start();  
    $_SESSION['uid']=$result[0]['uid'];  
    $_SESSION['uname']=$result[0]['uname'];  
}
```

About the API of Spotify:

```

public function GetToken(){

    $client_id = 'd0b25d3ab57b452b99fac5418744cca5';

    $client_secret = '1bcfbd475c044d729c99ea705a41971f';

    $ch = curl_init();

    curl_setopt($ch, CURLOPT_URL,
'https://accounts.spotify.com/api/token');

    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

    curl_setopt($ch, CURLOPT_POST, 1);

    curl_setopt($ch, CURLOPT_POSTFIELDS,
'grant_type=client_credentials');

    curl_setopt($ch, CURLOPT_HTTPHEADER,
array('Authorization: Basic
'.base64_encode($client_id.':'.$client_secret)));

    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE);

    curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, FALSE);

    $result=curl_exec($ch);

    curl_close($ch);

    return json_decode($result);

}

```

Firstly, we need to get access token using the client_id and client_secret provided by Spotify for every user.

And then we could use the APIs with the access token as the authorization. We use

APIs to get tracks' urls, albums' images and artists' images.

To get the data by applying

```
$ curl -X GET "https://api.spotify.com/v1/albums/OsNOF9WDwhWunNAHPD3Baj"  
  
-H "Authorization: Bearer {your access token}"
```

We could implement curl requests by curl in PHP.

```
public function GetTrackAudio(){  
    $track_id = $_GET['track_id'];  
    $my_token = $_GET['my_token'];  
    $ch = curl_init();  
    curl_setopt($ch, CURLOPT_URL,  
    'https://api.spotify.com/v1/tracks/'.$track_id);  
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);  
    curl_setopt($ch, CURLOPT_HTTPHEADER,  
    array('Authorization: Bearer '.$my_token));  
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE);  
    curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, FALSE);  
    $result=curl_exec($ch);  
    curl_close($ch);  
    return json_decode($result);  
}
```

Part 5. Web Pages Design

Extra Details that should be considered:

We mainly used Ajax to submit the form instead of form tag, considering that if we use original way to submit the form, the whole page will be refreshed, so the audio tag can not be play all time and has to be paused when we submit the form, which is not practical for a music streaming service. By applying Ajax, we could reduce the stress for server, which make us web application become more faster.

1. Welcome Page

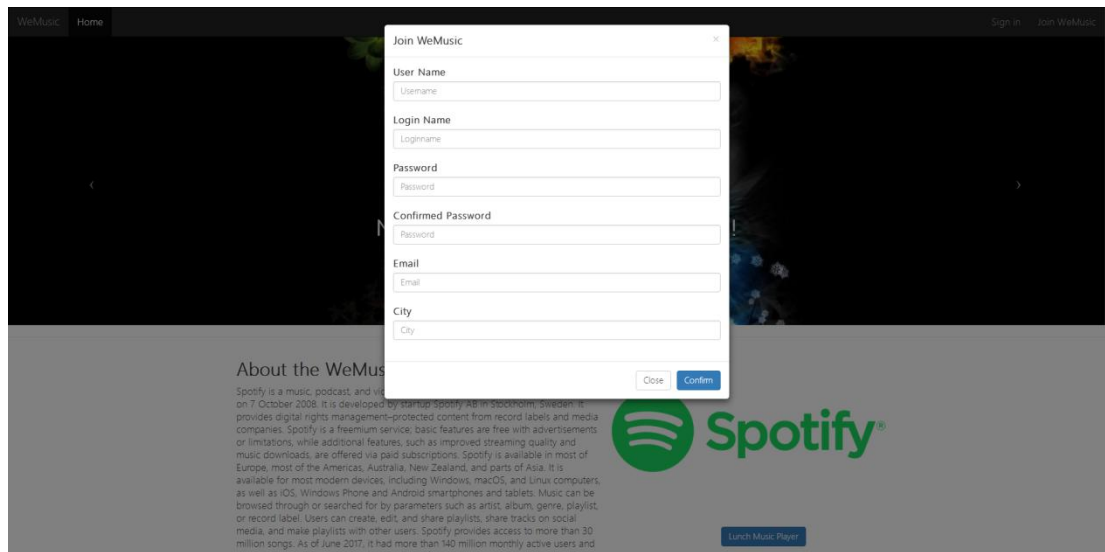
When you start browsing this website, you has two options, login and sign up.

Your activities will be logged until you log out.



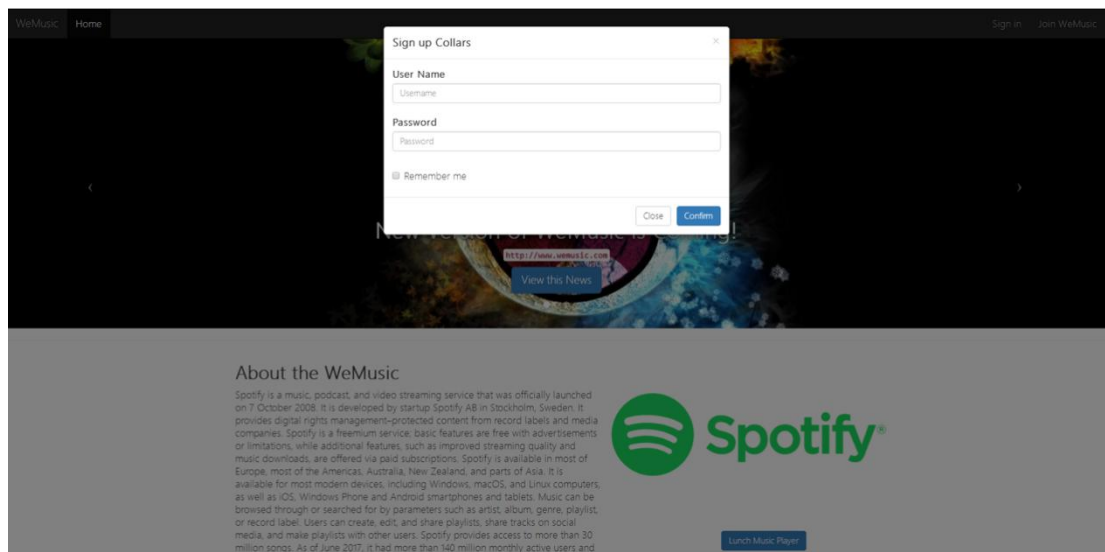
2. Sign Up Page

If you are new to this website, then you may need an account. Click on “Sign Up” button, then we will lead you to a register page. Note that your username should be unique and we’ll check if this username is valid. You should fill out all the information in this page. Also, you can edit the information of their login account and city after logging in.



3. Login Page

If you already have an account, then you can just input your username and password to login to the home page. If you click the 'Remember me' checkbox, it will create cookie for login information.

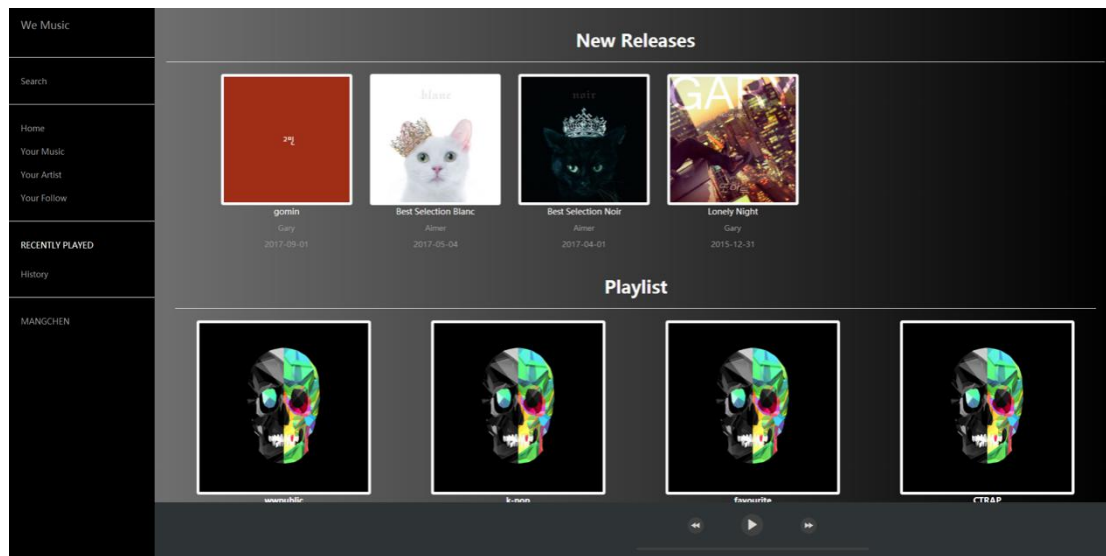


4. Main Page

After logging in the wemusic website, you will see the main page of our music system. Here is everything you can do in this page, which includes three parts.

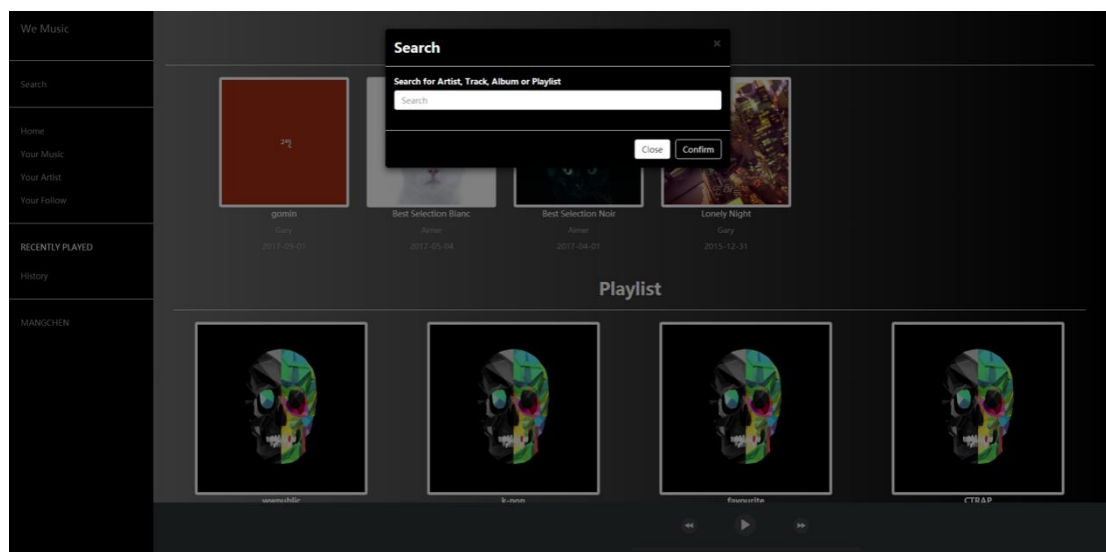
- (1) Search function. You can search using keywords that are matched against artist names, genres, song titles, etc.
- (2) List of music, artist and follow.
- (3) Recently played. This part records the recently played track, playlist and album.

You can easily find the recent records here.

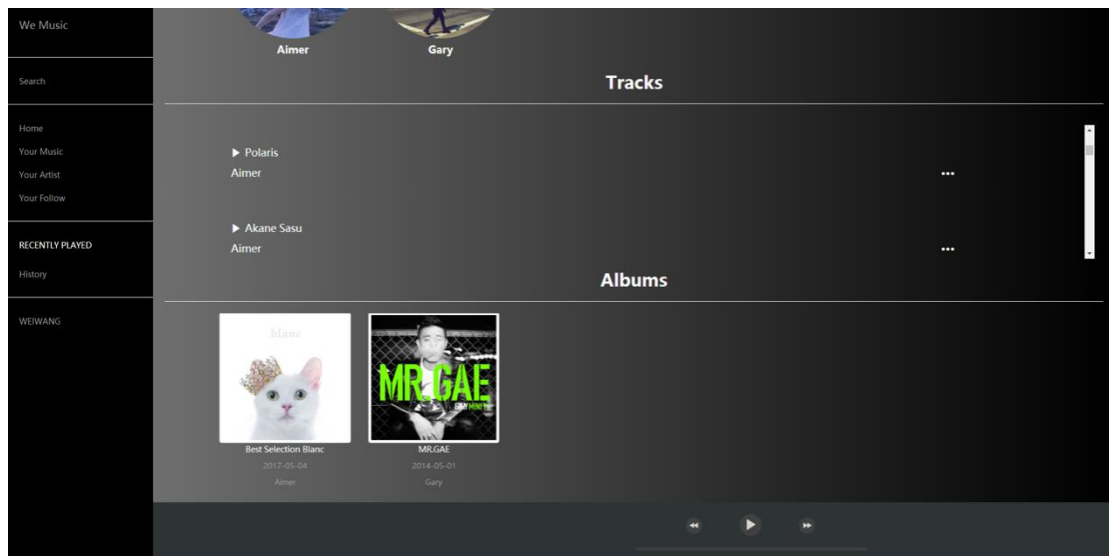


5. Search Tracks Page

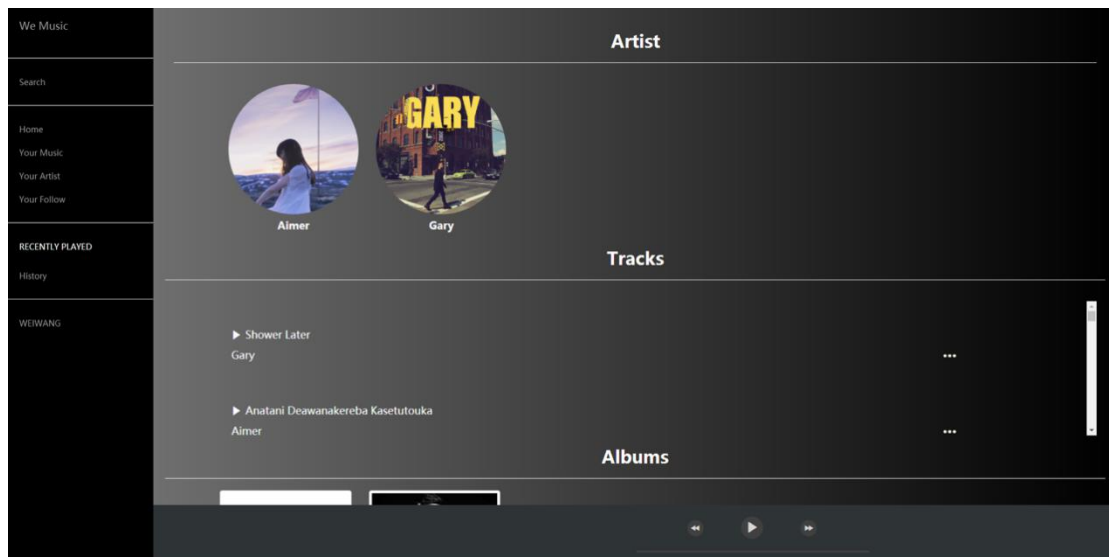
You can search tracks, albums, artists using the search function. We will output all the information about your search keywords of the tracks, albums and artists in the following page.



6. Search Albums Page

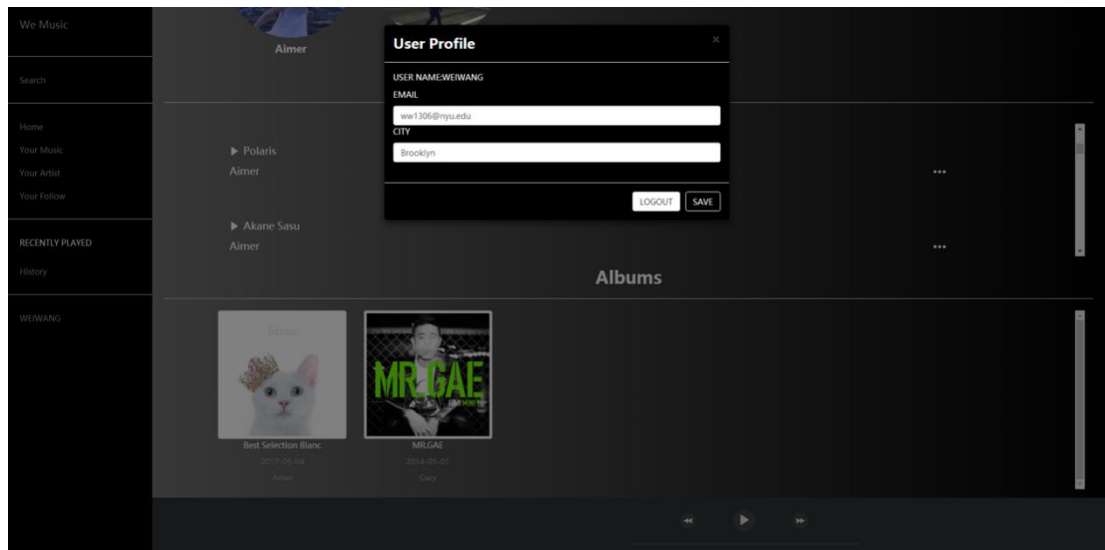


7. Search Artists Page



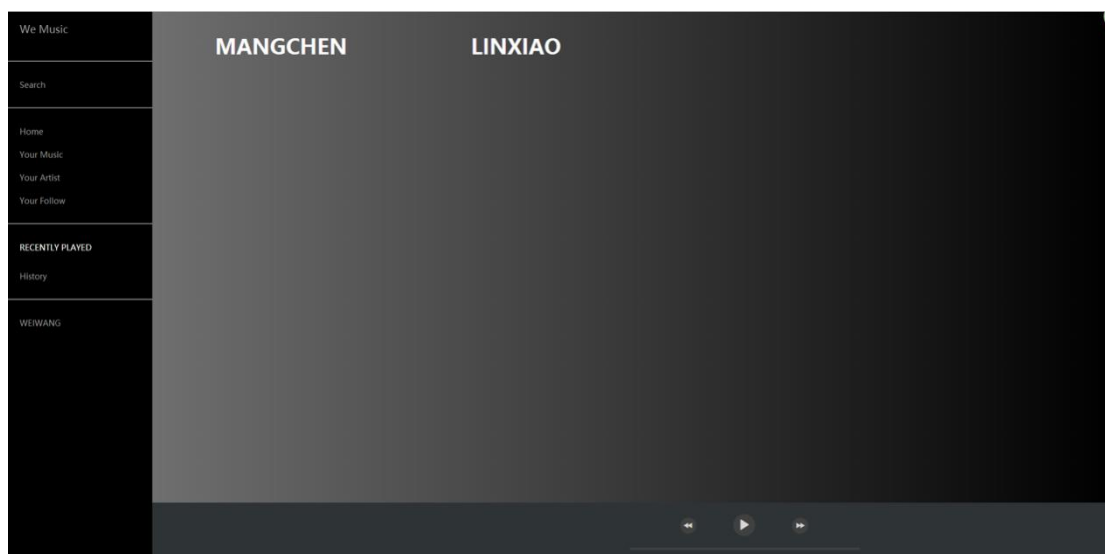
8. Profile Page

you can see your login email and city here. Also, you can edit your basic information here and click the “save” button to save your information.



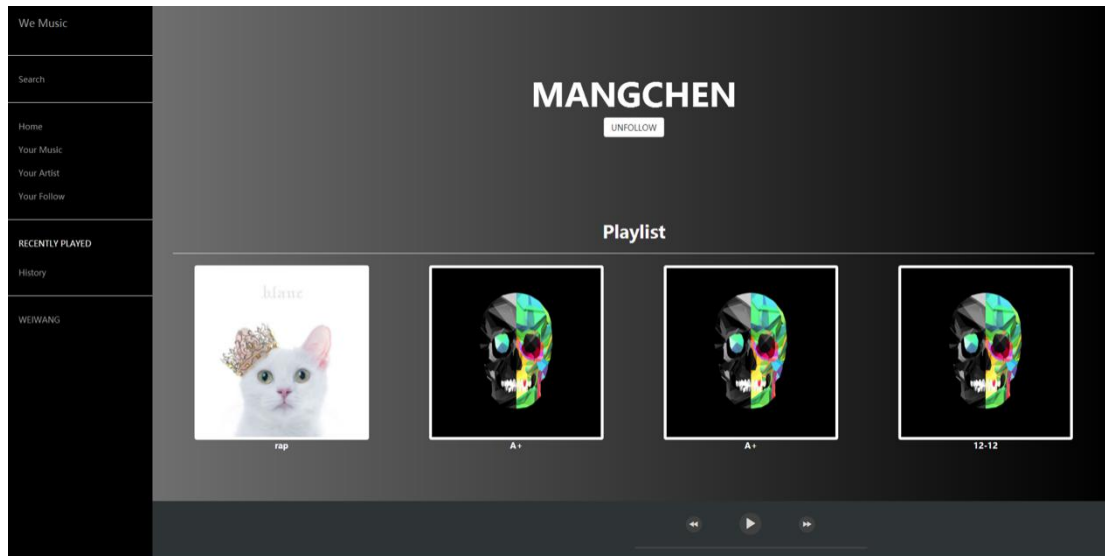
9. Following List Page

You can see all your following here. You can see their public playlists by clicking the name of each followings.



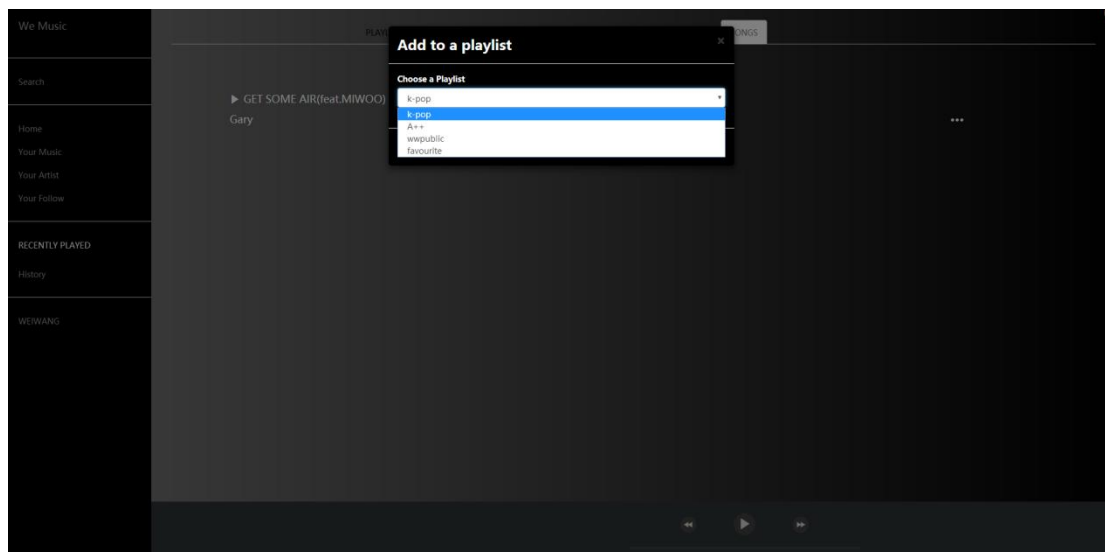
10. Following's Playlist Page

You can see the following's public playlist here. You can play any of the tracks, albums in the playlist.



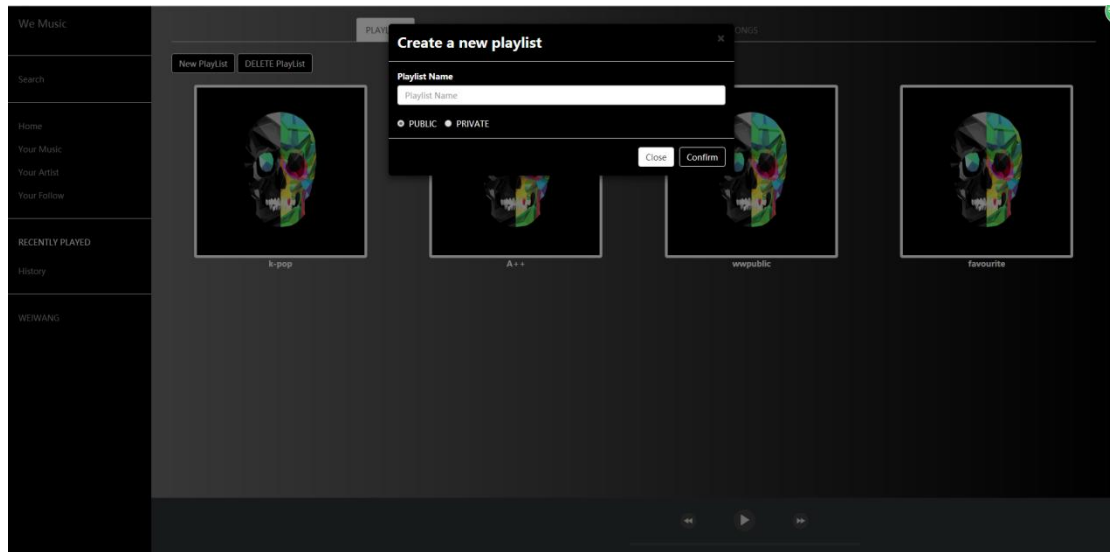
11. Add to Playlist Page

If you want to add a track to your playlist, just click the choose options button right of the track to add. You can choose any of your existed playlist below.



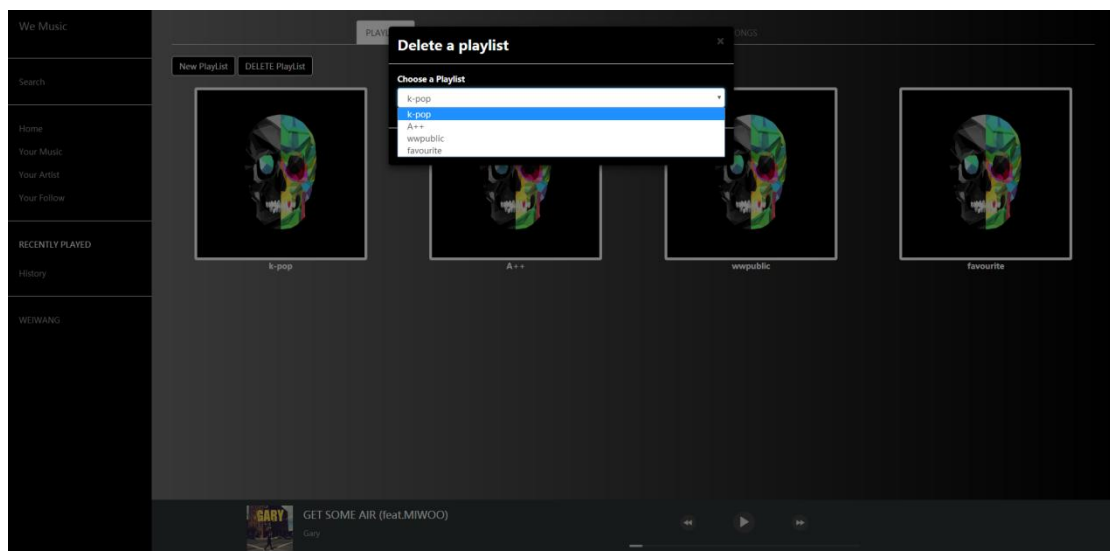
12. Create new Playlist Page

You can create a new playlist. When create playlist, you should input a playlist name. Note that it is fine if playlist name is not unique. You can also set it public which can be seen by anyone, or private which can only be seen by yourself.



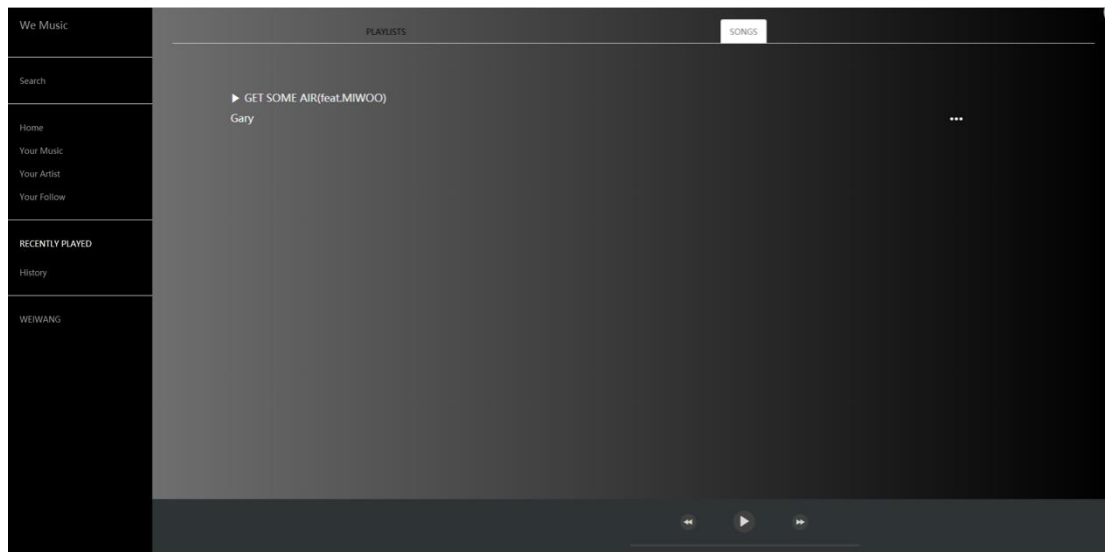
13. Delete a Playlist Page

You can delete your playlist here.



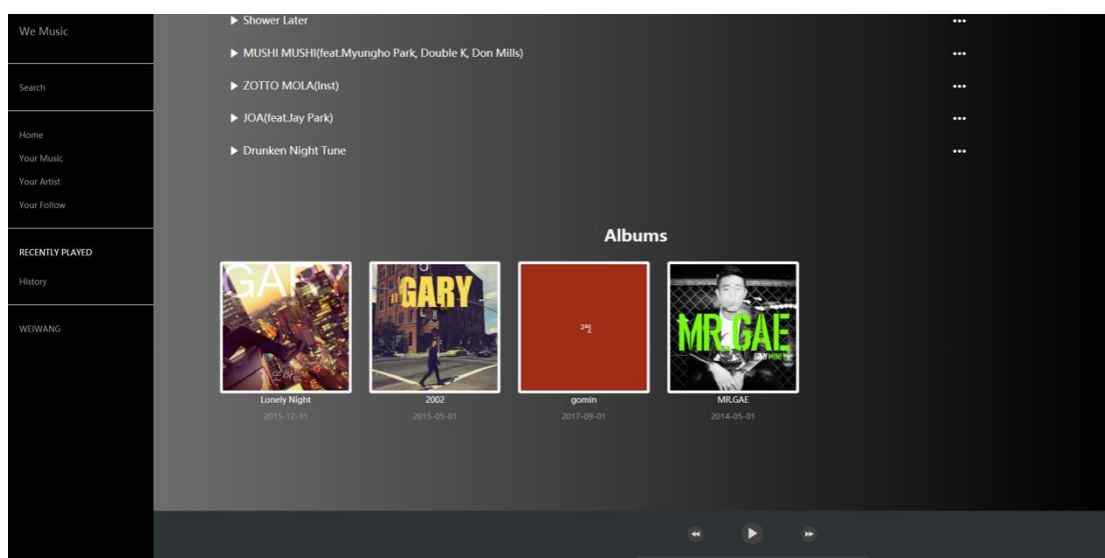
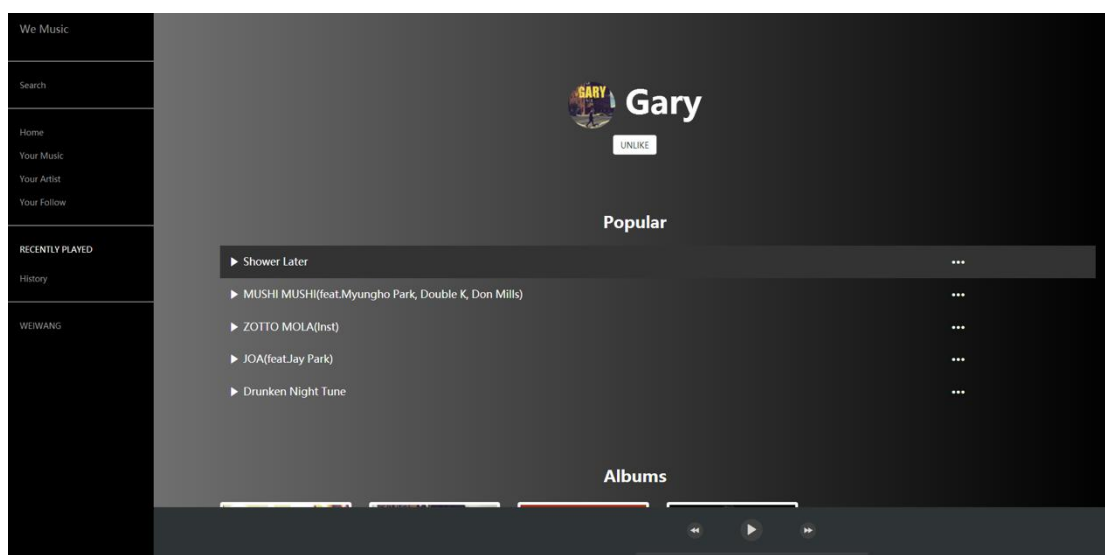
14. Songs Page

You can click on a playlist and check the songs in the playlist. You can remove the songs from the playlist also.



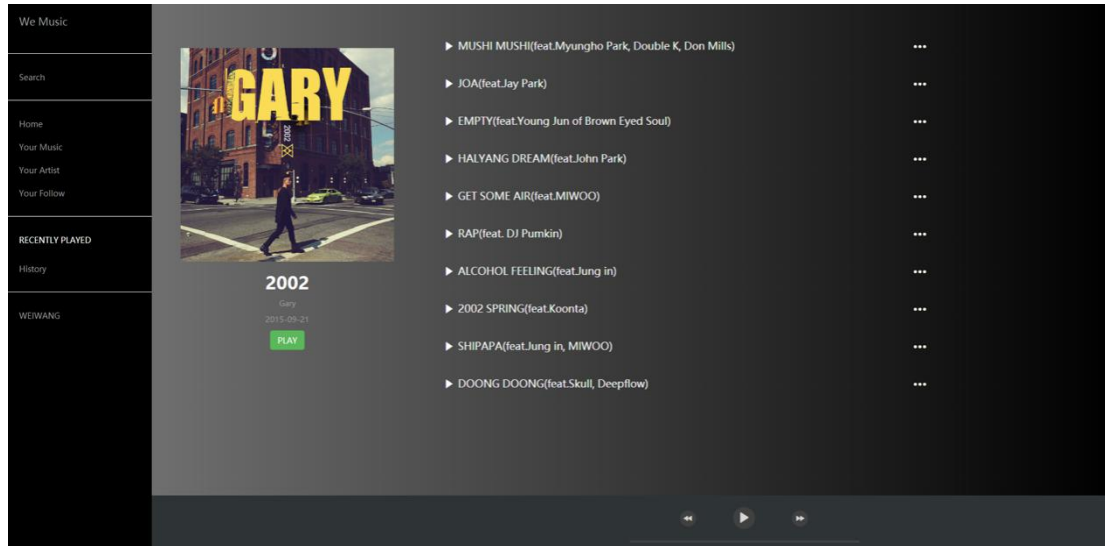
15. Artist Page

You can visit an artist and see all of his/her tracks and albums.



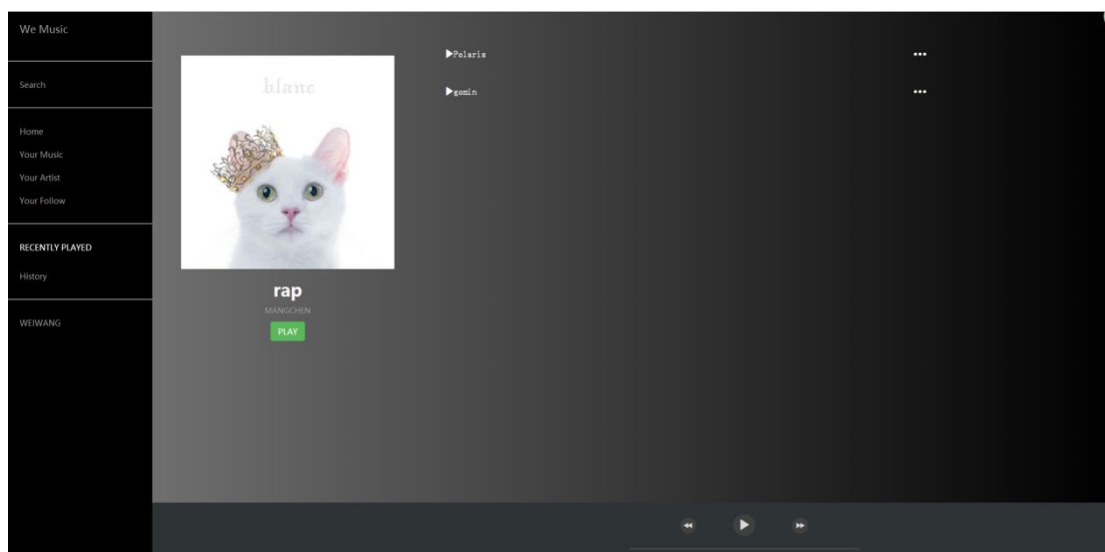
16. Track Page

You can play a track by clicking the name of the songs. You can play tracks wherever pages you are. When you press the pause/play button, the song will pause or play.



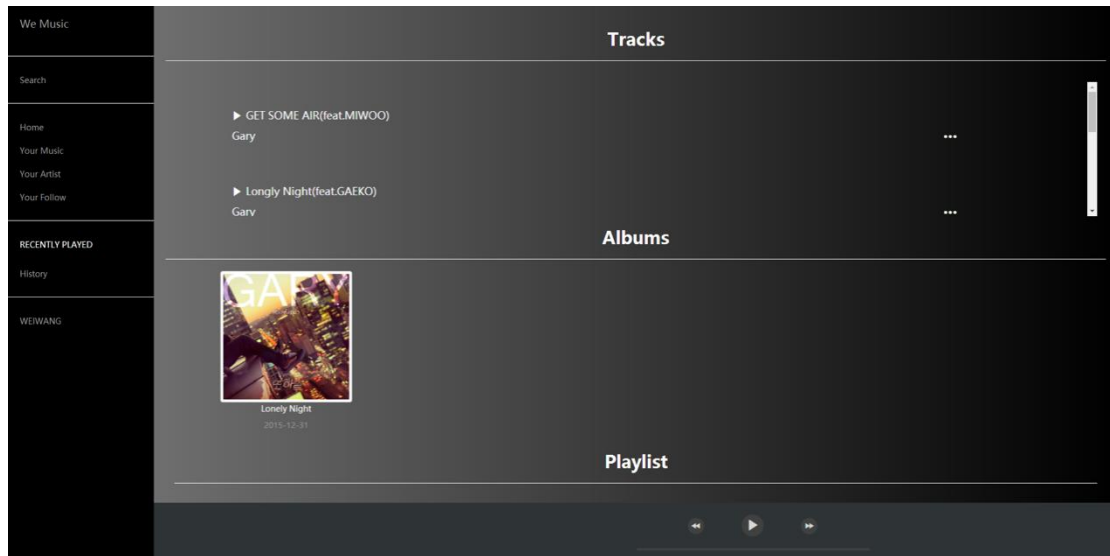
17. Album Page

You can check an album and all of the songs in the album.

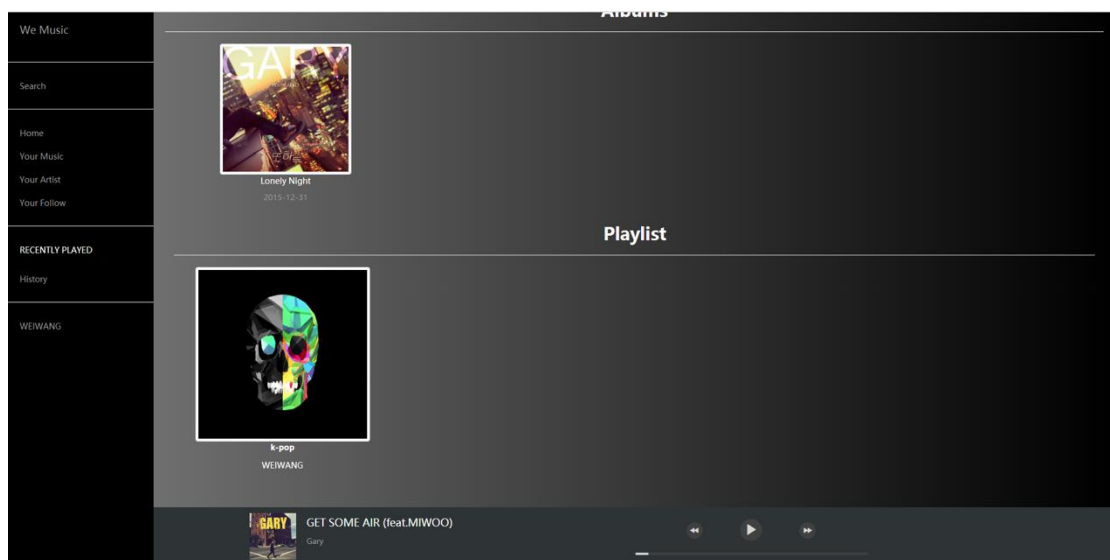


18. Track History Page

This section shows your recently played tracks, playlist and albums. Every time you play a track the system will log your behavior and store the data into the database. This mechanize provides the data for analyzing your behavior. You can see your recently tracks, albums, playlists easily.




19. Playlist History Page



20. Rating Page

You can rate a song by clicking the options function button in the right of the track.

You can rate from 1 star to 5 stars.



Best Selection Noir

Aimer
2017

▶ Ninelle

▶ Us

▶ Last Stardust

▶ Nemuri No Mori

▶ Insane Dream

▶ Brave Shine

▶ Hollow World

▶ StarRingChild

▶ Dareka Umiwo

▶ Re I Am

▶ S-ave

▶ Stars in the Rain

...

...

...

...

...

...

...

...

...

...

...

...

Rate

Remove From Your Music

Add to PlayList

We Music

Search

Home

Your Music

Your Artist

Your Follow

RECENTLY PLAYED

History

WEI WANG

PLAYLIST

SONGS

▶ GET SOME AIR(feat.MIWO)

Gary


...

Rate

Choose the score

1 Star 2 Star 3 Star 4 Star 5 Star

Close Confirm



GET SOME AIR (feat.MIWO)

Gary

⏮

▶

⏭

Progress bar