

1. Design of Database

(1) Relational Schema

User (username, password, uname, uemail, ucity)

Artist (artistID, arname, ardesc)

Album (albumID, alttitle, aldate)

Track (trackID, ttitle, duration, arname, albumID)

Playlist (playID, ptitle, pdate, visible, username)

Playlist_Track (playID, trackID, lsequence)

Like_Artist (username, artistID, ltimestamp)

Rate_Song (username, trackID, score, rtimestamp)

Follow_User (follow, followed, ftimestamp)

Play_Track (username, trackID, ptimestamp)

(2) Keys and Foreign key constraints

All the primary keys are underlined above.

Track.arname is a foreign key referencing to Artist.arname

Track.albumID is a foreign key referencing to Album.albumID

Playlist.username is a foreign key referencing to User.username

Playlist_Track.playID is a foreign key referencing to Playlist.playID

Playlist_Track.trackID is a foreign key referencing to Track.trackID

Like_Artist.username is a foreign key referencing to User.username

Like_Artist.artistID is a foreign key referencing to Artist.artistID

Rate_Song.username is a foreign key referencing to User.username

Rate_Song.trackID is a foreign key referencing to Track.trackID

Follow_User.follow is a foreign key referencing to User.username

Follow_User.followed is a foreign key referencing to User.username

Play_Track.username is a foreign key referencing to User.username

Play_Track.trackID is a foreign key referencing to Track.trackID

(3) Explanation of the schema

User, Artist, Track, Album and Playlist are exactly as what is described in the requirement. The visible in Playlist can be 'Public' or 'Private' to decide if it's available to other users. We use Playlist_Track to store the tracks in playlists, the attribute lsequence is the integer to indicate the track order in the lists since they are ordered. Like_Artist holds data when a user likes an artist, when they don't like the artist, we can simply delete that record, so the timestamp is not included in PK. Follow_User and Rate_Song are the same, timestamp is not in PK because users cannot follow the same person or rate the same song before the record is deleted. But since a user can play a track as much as they want, we have to include the timestamp in PK.

(4) Create the schema using MySQL

```
CREATE DATABASE MusicOnline;
USE MusicOnline;
```

```
CREATE TABLE `User` (
  `username` VARCHAR(20) NOT NULL,
  `password` VARCHAR(45) NOT NULL,
  `uname` VARCHAR(45) NOT NULL,
  `uemail` VARCHAR(45),
  `ucity` VARCHAR(20),
  PRIMARY KEY (`username`),
  UNIQUE INDEX `uemail_Index` (`uemail` ASC));
```

```
CREATE TABLE `Artist` (
  `artistID` VARCHAR(45) NOT NULL ,
  `arname` VARCHAR(200) NOT NULL,
  `ardesc` TEXT,
  PRIMARY KEY (`artistID`),
  UNIQUE INDEX `arname_Index` (`arname` ASC)
);
```

```
CREATE TABLE `Album` (
  `albumID` VARCHAR(45) NOT NULL,
  `alttitle` TEXT,
  `aldate` VARCHAR(45),
  PRIMARY KEY (`albumID`));
```

```
CREATE TABLE `Track` (
  `trackID` VARCHAR(45) NOT NULL,
  `ttitle` TEXT,
  `duration` INT NOT NULL,
  `arname` VARCHAR(200) ,
  `albumID` VARCHAR(45),
  PRIMARY KEY (`trackID`),
  FOREIGN KEY (`arname`) REFERENCES `Artist`(`arname`) ON DELETE NO
ACTION ON UPDATE CASCADE,
  FOREIGN KEY (`albumID`) REFERENCES `Album`(`albumID`) ON DELETE NO
ACTION ON UPDATE CASCADE
);
```

```
CREATE TABLE `Playlist` (
  `playID` INT NOT NULL AUTO_INCREMENT,
  `ptitle` VARCHAR(20) NOT NULL,
```

```
    `pdate` DATE NOT NULL,  
    `visible` VARCHAR(20) NOT NULL,  
    `username` VARCHAR(20) NOT NULL,  
    PRIMARY KEY (`playID`),  
    INDEX `username_Index` (`username` ASC),  
    CONSTRAINT `username`  
        FOREIGN KEY (`username`) REFERENCES `User` (`username`) ON DELETE  
        CASCADE ON UPDATE CASCADE);
```

```
CREATE TABLE `Playlist_Track` (  
    `playID` INT NOT NULL,  
    `trackID` VARCHAR(45) NOT NULL,  
    `lsequence` INT,  
    PRIMARY KEY (`playID`, `trackID`),  
    FOREIGN KEY (`playID`) REFERENCES `Playlist` (`playID`) ON DELETE  
    CASCADE ON UPDATE CASCADE,  
    FOREIGN KEY (`trackID`) REFERENCES `Track` (`trackID`) ON DELETE  
    CASCADE ON UPDATE CASCADE);
```

```
CREATE TABLE `Like_Artist` (  
    `username` VARCHAR(20) NOT NULL,  
    `artistID` VARCHAR(45) NOT NULL,  
    `ltimestamp` DATETIME NOT NULL,  
    PRIMARY KEY (`username`, `artistID`),  
    INDEX `artistID_Index` (`artistID` ASC),  
    FOREIGN KEY (`username`) REFERENCES `User` (`username`) ON DELETE  
    CASCADE ON UPDATE CASCADE,  
    FOREIGN KEY (`artistID`) REFERENCES `Artist` (`artistID`) ON DELETE  
    CASCADE ON UPDATE CASCADE);
```

```
CREATE TABLE `Rate_Song` (  
    `username` VARCHAR(20) NOT NULL,  
    `trackID` VARCHAR(45) NOT NULL,  
    `score` INT NOT NULL,  
    `rtimestamp` DATETIME NOT NULL,  
    PRIMARY KEY (`username`, `trackID`),  
    INDEX `trackID_Index` (`trackID` ASC),  
    FOREIGN KEY (`username`) REFERENCES `User` (`username`) ON DELETE  
    CASCADE ON UPDATE CASCADE,  
    FOREIGN KEY (`trackID`) REFERENCES `Track` (`trackID`) ON DELETE  
    CASCADE ON UPDATE CASCADE);
```

```
CREATE TABLE `Follow_User` (  
    `follow` VARCHAR(20) NOT NULL,  
    `followed` VARCHAR(20) NOT NULL,
```

```

`ftimestamp` DATETIME NOT NULL,
PRIMARY KEY (`follow`, `followed`),
INDEX `follow_Index` (`follow` ASC),
FOREIGN KEY (`follow`) REFERENCES `User` (`username`) ON DELETE
CASCADE ON UPDATE CASCADE,
FOREIGN KEY (`followed`) REFERENCES `User` (`username`) ON DELETE
CASCADE ON UPDATE CASCADE);

```

```

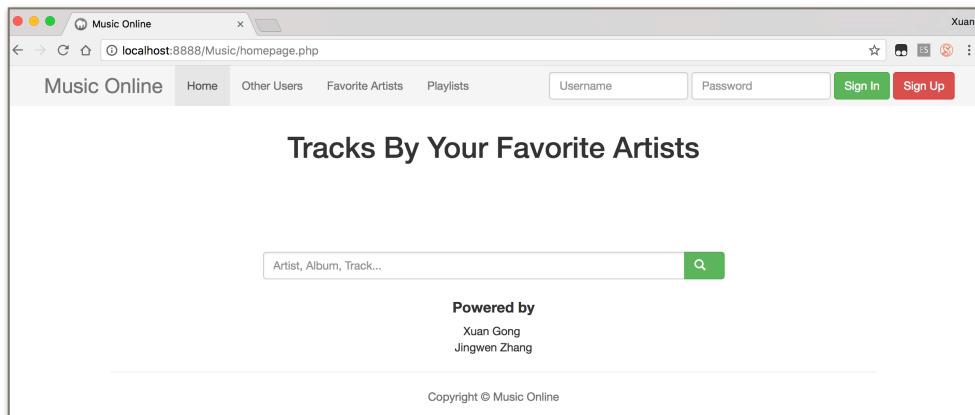
CREATE TABLE `Play_Track` (
`username` VARCHAR(20) NOT NULL,
`trackID` VARCHAR(45) NOT NULL,
`ptimestamp` DATETIME NOT NULL,
PRIMARY KEY (`username`, `trackID`, `ptimestamp`),
INDEX `trackID_Index` (`trackID` ASC),
FOREIGN KEY (`username`) REFERENCES `User` (`username`) ON DELETE
CASCADE ON UPDATE CASCADE,
FOREIGN KEY (`trackID`) REFERENCES `Track` (`trackID`) ON DELETE
CASCADE ON UPDATE CASCADE);

```

2. User Guide

(1) Sign Up and Sign In

The homepage is as follows:



Users can sign up and sign in using the buttons in the up-right corner. If the username or the email has been already registered, the system will tell the user and the input data will not be passed to database, also we verify the validation of user's input such as email to make sure the database is maintained properly. If the username or the password is incorrect, the system will prevent user from login:

Sign Up

Username

This username has been registered

Password

Confirm Password

Name

Email

City

Already Registered? [Click Here to Sign In](#)

Sign In

Username

Username

This username has not been registered

Password

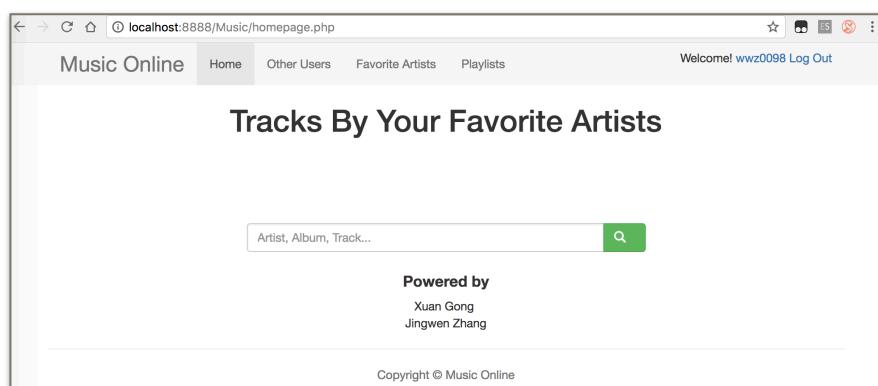
Password

Wrong Password

Sign In

(2) Edit personal information and log out

After the user signs in, the sign-in area will change, system will show username, and that is the place to check out the user's information. Click 'Save' to make a change, and click 'Cancel' to close the page. If you click 'Log Out', the sign-in area will change back:



Personal Information

Edit

Username	wwz0098
Password	*****
Confirm Password	*****
Name	Xuan Gong
Email	xg709@nyu.edu
City	Brooklyn

[Save](#) [Cancel](#)

(3) Follow other users

Click the second tab on the top, so that you can see other users in a list. And you can click their name to enter their information page, there you can follow a user:

User Lists		
Username	Name	Email
Jerry1993	Jerry Huang	jh@gmail.com
jz1030	Jingwen	jz1030@nyu.edu
Thomas1994	Thomas M	tm@gmail.com

Powered by
Xuan Gong
Jingwen Zhang

Copyright © Music Online

User Information

Username	Name	Email
jz1030	Jingwen	jz1030@nyu.edu

[Click to Follow This User](#)

User Information

Username	Name	Email
jz1030	Jingwen	jz1030@nyu.edu

[You Followed This User!!](#)

(4) Search artists and tracks

On the homepage, you can input keyword to search artists and tracks, so the system will present the user two lists, artist list and track list respectively. The user can click the name of the artist to enter an artist info page or click the name of the track to enter a track info page. On an artist info page, you can like the artist (and their songs may be recommended on your homepage). On the track info page, you can rate this track or add this track to one of your playlists, if you have rated this track, the system will show you the score:

Artist Lists	
Name	Genre
Bruno Mars	dance pop pop rap post-teen pop
Thirty Seconds To Mars	alternative metal modern rock nu metal pop punk post-grunge
Hollywood Porn Stars	belgian indie french rock
Blue Scholars	abstract hip hop alternative hip hop pop rap underground pop rap underground rap
Zara Larsson	dance pop pop post-teen pop swedish pop tropical house
Britney Spears	dance pop pop pop christmas pop rap post-teen pop r&b urban contemporary
Cobra Starship	dance pop emo modern rock pop punk pop rap post-teen pop
Jamie Lynn Spears	post-teen pop
Jefferson Starship	album rock art rock blues-rock british blues bubblegum pop classic rock country rock folk rock hard rock mellow gold progressive rock rock singer-songwriter soft rock southern rock symphonic rock
Tears For Fears	album rock art rock dance rock mellow gold new romantic new wave new wave pop permanent wave rock soft rock synthpop

Artist Information	
Name	Genre
Bruno Mars	dance pop pop rap post-teen pop
Click to Like This Artist	

Track Information			
Title	Duration	Artist	Album
This Is War	05:27.260	Thirty Seconds To Mars	This Is War
You can rate this track:			
<input type="text" value="1"/>			
Submit			
You can add this track to your playlist:			
<input type="text"/>			
Submit			

(5) Create your own playlists and play music online

On the homepage, click the last tap on the top, you can see your playlists and the playlists from those whom you follow. Click on the name of the playlist, you can enter the page of that playlist and play the song in that playlist. You can also create a new playlist, input a name and choose the visibility, if the visibility is 'Private', the playlist can only be seen by yourself.

The image consists of three vertically stacked screenshots of a music application interface.

Screenshot 1: Create Your Playlist

This screenshot shows a modal window titled "Create Your Playlist". It contains fields for "Title" (with a placeholder "Title") and "Visibility" (set to "Private"). Below the fields are "Create" and "Cancel" buttons. The background of the main window shows a "Powered by" section with names: Xuan Gong, Jingwen Zhang.

Screenshot 2: Your Playlists

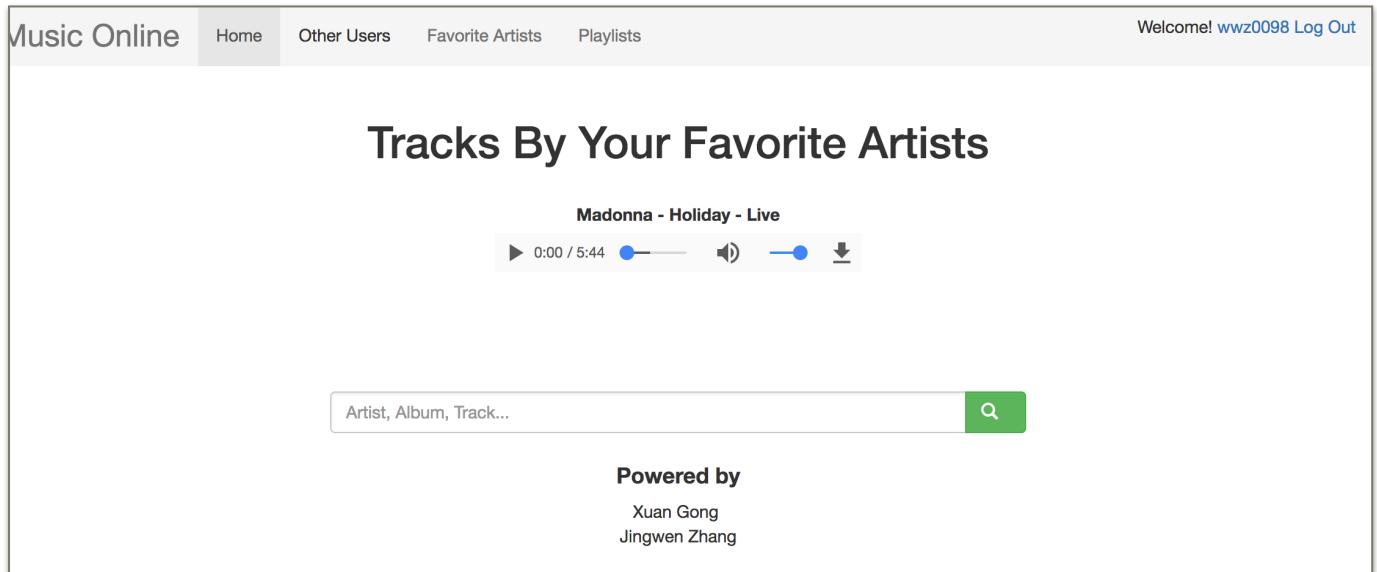
This screenshot shows a list of "Your Playlists". A table with three columns: "Title", "Visible", and "Date Created". One entry is listed: "Pop" (Visible: public, Date Created: 2017-12-12). Below this section is a heading "Others' Playlists" and a table for "Others' Playlists" with one entry: "Music" (Creator: jz1030, Date Created: 2017-12-12).

Screenshot 3: Pop

This screenshot shows the details for the "Pop" playlist. It lists three songs: "Holiday - Live" by Madonna, "Force Of Nature" by Pearl Jam, and "2 Become 1 (Georgie Porgie" by Spice Girls. Each song entry includes its title, duration, artist, album, track progress (0:00 / 5:44, 0:00 / 4:04, 0:00 / 4:01), volume control, and download icon.

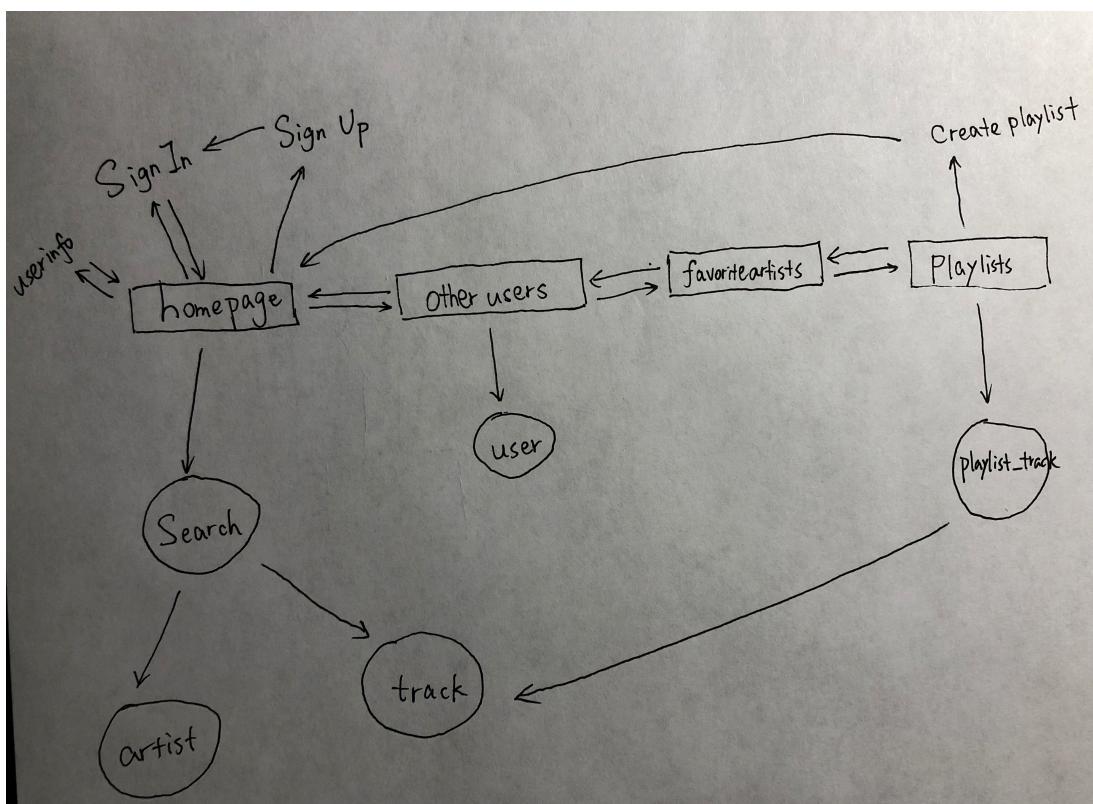
(6) Recommendation on the homepage

The system will keep the record if a user plays a track online. And the homepage should be like this if you play a track performed by an artist you like:



3. Code Structure

The relationship between the pages is as follows, and all these .php files can be found in the server root: (Two .php files are not in the graph, connection.php is used to connect database and logout.php is used to destroy the session so the user can log out)



The music which is used for playing online is in the music file, since we don't use Spotify API, so the source has to be in the server (in music file), and there are some .php files in the server file. We use AJAX in this project, so these server files are used to deal with the request from the front end. Just as the name of these files, we post a request to the server when the user likes an artist, so a new record need to be inserted into database, the code in likeArtist.php deals with it. And the code in followUser.php, playTrack.php etc., all of them are used to insert a new record into the database.