Software Design Specification

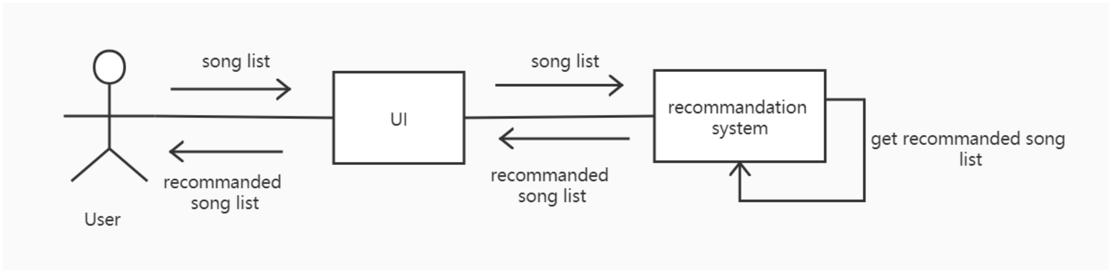
1.Introduction

The goal of this project is to develop a practical and easy-to-use music recommendation system, which can help users quickly and accurately find songs related to their favorite songs. According to the design requirements of the recommended system, this document provides an overall framework and design direction, and also defines some requirements of the system for users to confirm the function and performance of the system. The purpose of writing a document is to explain the design considerations of each program in each level of the music recommendation system, and to make the solution concrete, and explain how to implement the system.

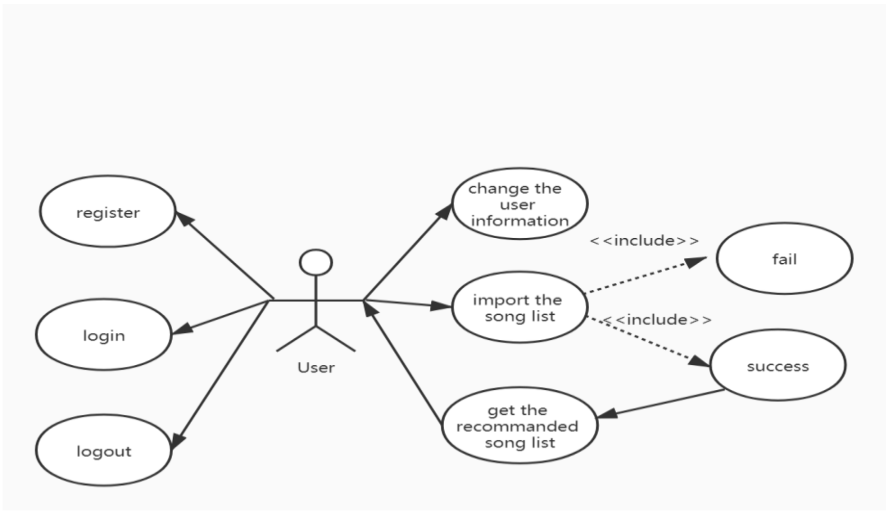
2. The UI design

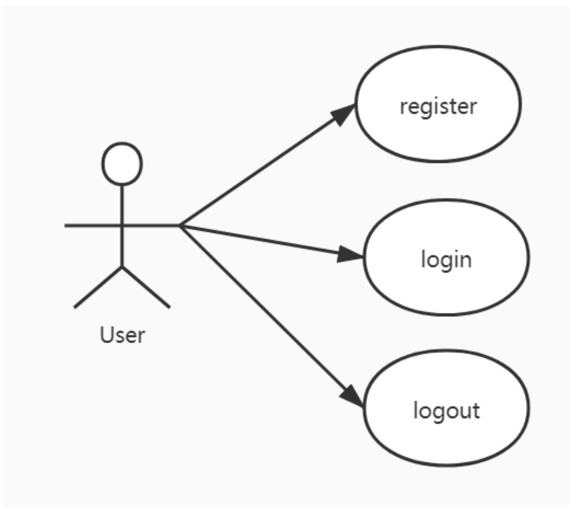
2.1 introduction

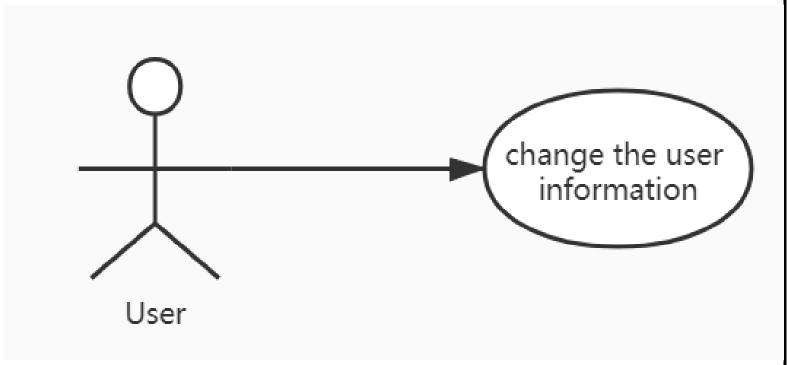
User interface is the most important part of a recommendation system. It is a platform for users to interact with the system. Users can register, log in and log out through it, import the song list and display the recommended song list. The user interface accepts the user's imported song list, sends it to the recommendation system, accepts and displays the recommended song list from the recommendation system. In addition to basic functions, the interface should be beautiful and increase user experience.



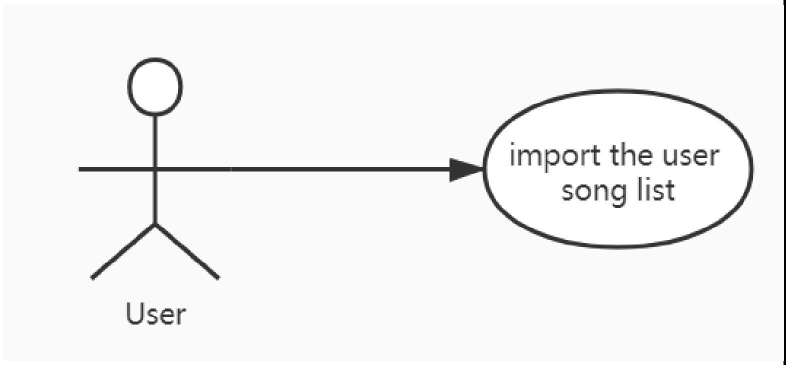
2.2 UML Use case diagram

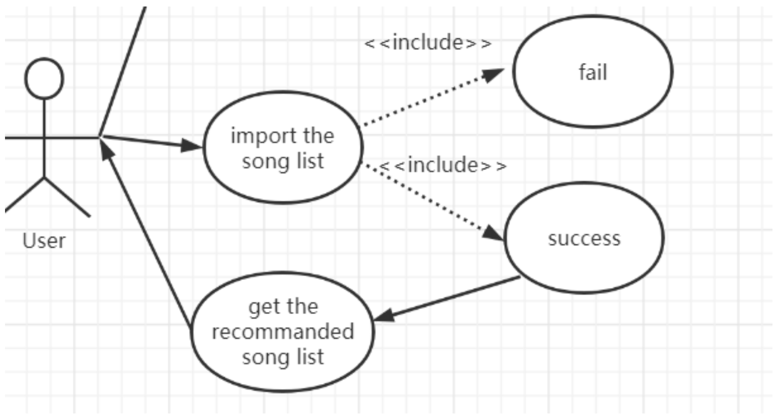


Use Case	Case1: login ,logout,register
goal	User can login ,logout and register the system form the UI
detail	Users can click the registration, login and logout buttons on the user interface to perform these operations. The premise of login and logout is to register an account.
actor	User
diagram	

Use Case	Case2: Users change account information
goal	The user can change the account information. For example, they want to change the password
detail	Users click the settings button on the user interface to change their personal information
actor	User
diagram	

Use Case	Case3: user importing song list data
goal	Users can import their own collection of songs and listen to songs

detail	The user can click the phase import record button on the user interface to import data
actor	User
diagram	

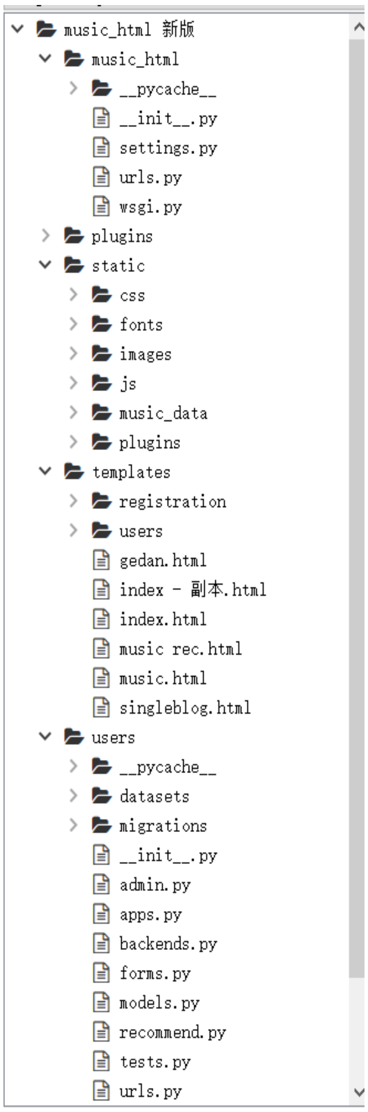
Use Case	Case4: users get recommended song list
goal	Users can get their own song list data according to their own preferences
detail	After sending a query request, users can see their recommended songs on the user interface. The precondition is that they must import their own relevant data, such as collecting songs and listening records
actor	User
diagram	 <<include>> fail import the song list <<include>> success User get the recommanded song list

Views 中的函数：

	Description	Input	Expected output

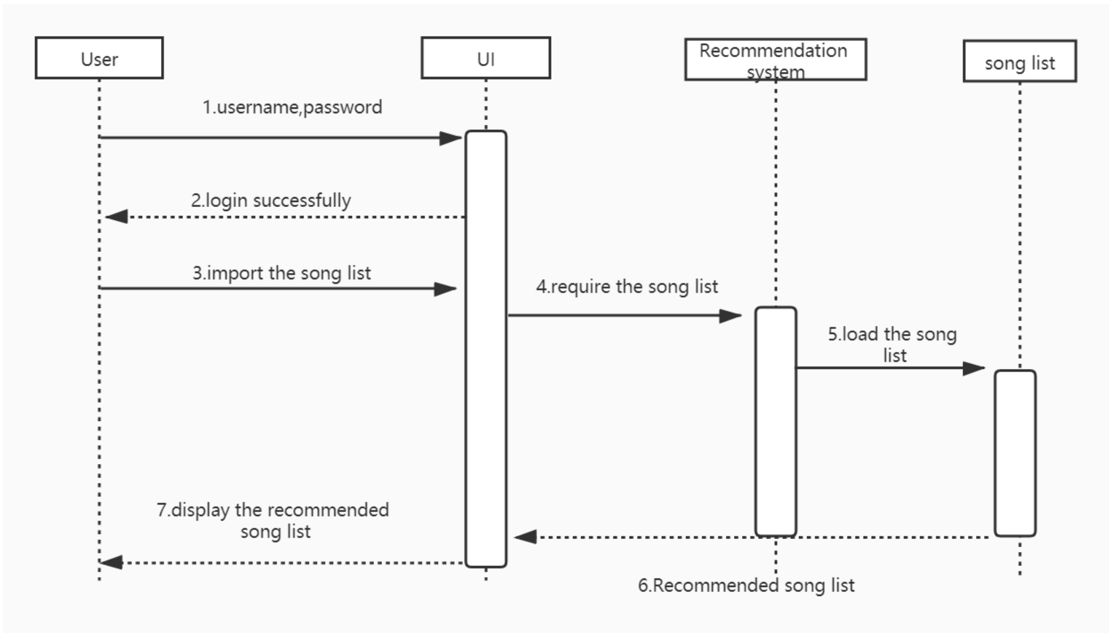
1. register(request)	Register in the system	Username, password , email , (repeat) password	{'success': True, 'error': ''}
2. index(request)	To get into the index html	Username, password	{'success': True, 'error': ''}
3. recommend(request)	To recommend a music list	A list of music	A list of music

The frame of Django

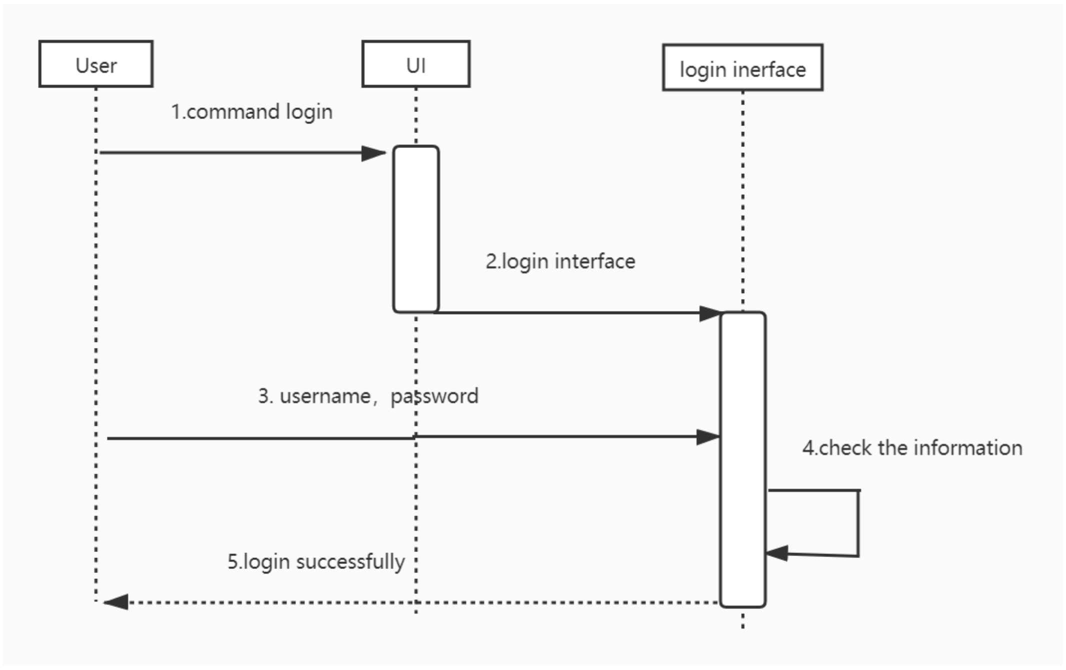


```
music_html 新版
    music_html
        __pycache__
        __init__.py
        settings.py
        urls.py
        wsgi.py
    plugins
    static
        css
        fonts
        images
        js
        music_data
        plugins
    templates
        registration
        users
        gedan.html
        index - 副本.html
        index.html
        music rec.html
        music.html
        singleblog.html
    users
        __pycache__
        datasets
        migrations
        __init__.py
        admin.py
        apps.py
        backends.py
        forms.py
        models.py
        recommend.py
        tests.py
        urls.py
```

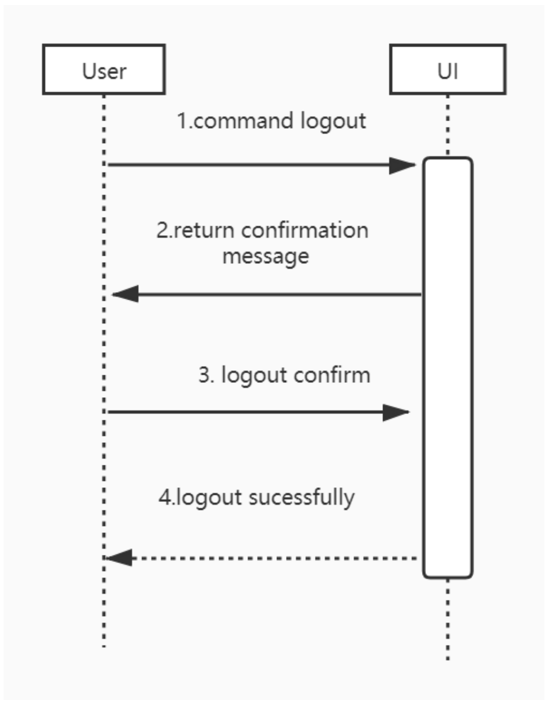
2.3 UML sequence diagram



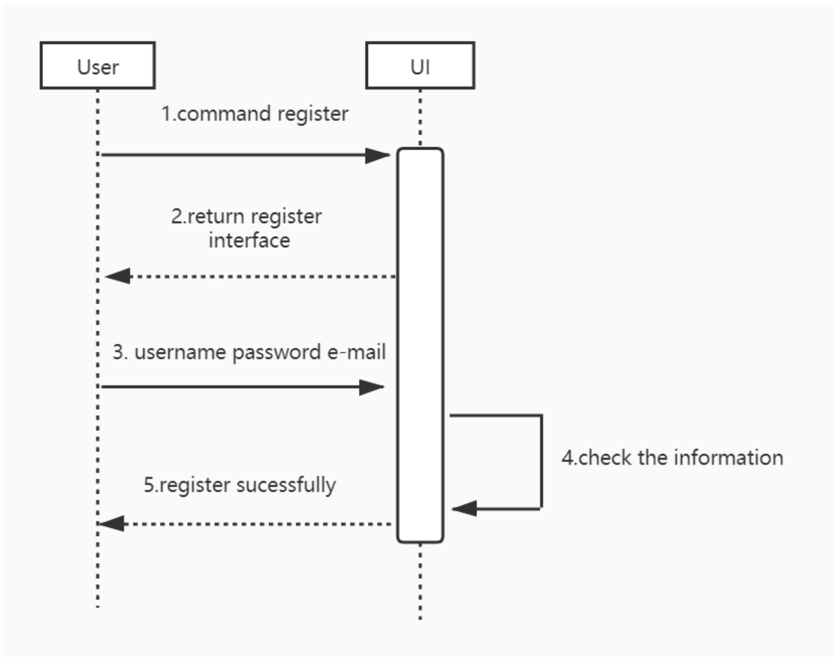
1.Login



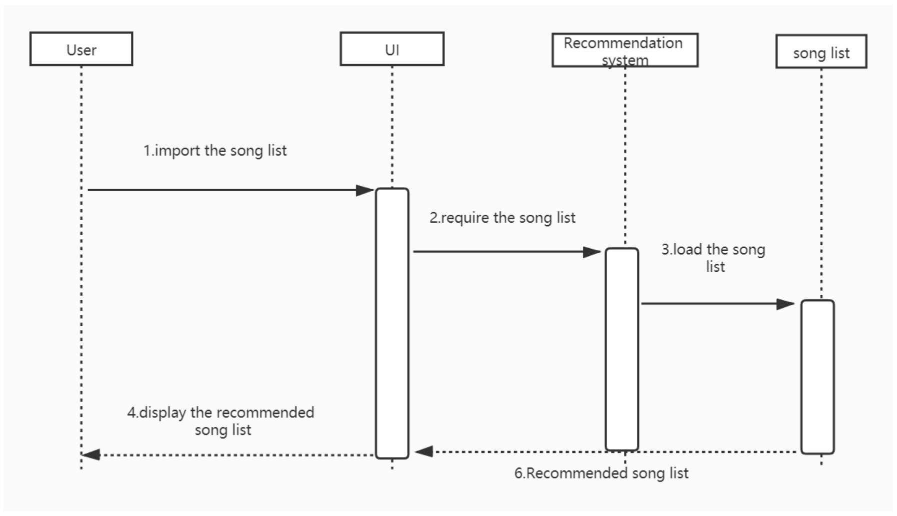
2.Logout



3.Register

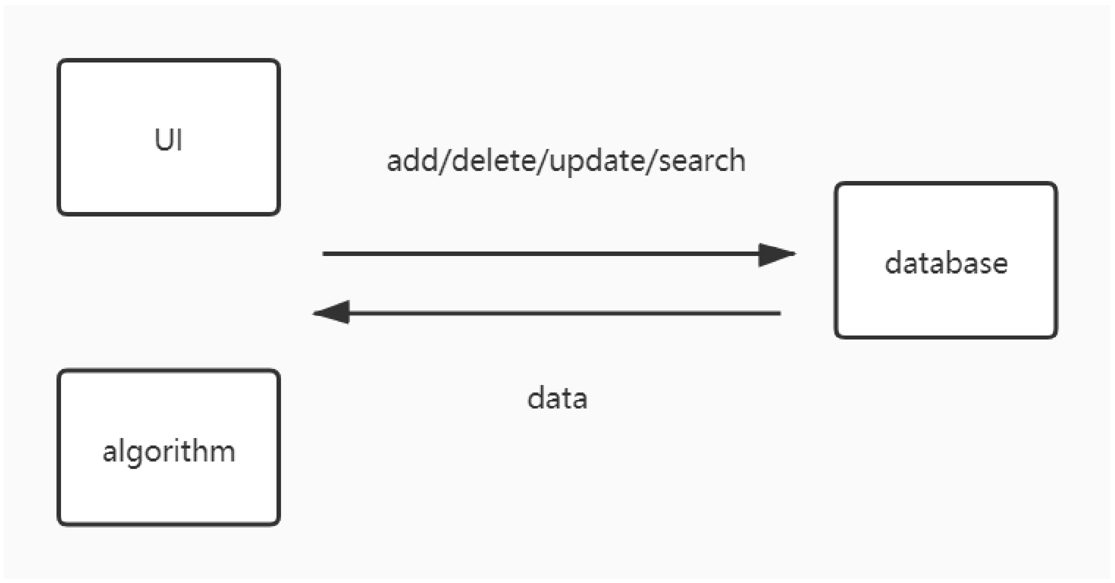


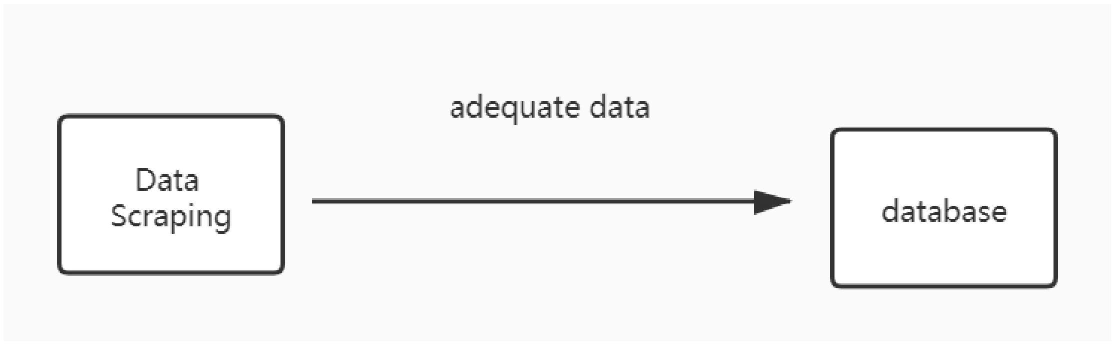
4.import the song list and display the song list



3.The database design

Database is an important part of a recommendation system. It needs to save the user information from the interface and a large number of song lists from the data crawling part. When the recommendation algorithm needs to recommend songs, the database needs to transfer the data to the algorithm. For other modules, they can add, update, delete and search for the data if necessary.





3.1 The tables in the database

allsong	time	author	album	songlist	
songlist	listname	author	viewct		
user	userid	username			
usersong	songname	author	viewct	username	

There are four tables designed for the database,they are allsong ,songlist,user and usersong.

3.2 physical design

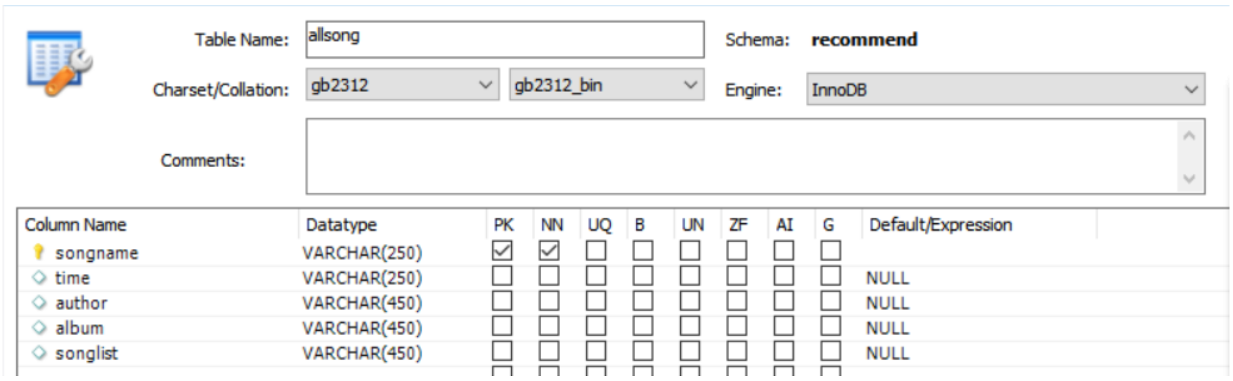


Table Name: allsong
Schema: recommend
Charset/Collation: gb2312 gb2312_bin
Engine: InnoDB
Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
songname	VARCHAR(250)	✓	✓							
time	VARCHAR(250)									NULL
author	VARCHAR(450)									NULL
album	VARCHAR(450)									NULL
songlist	VARCHAR(450)									NULL

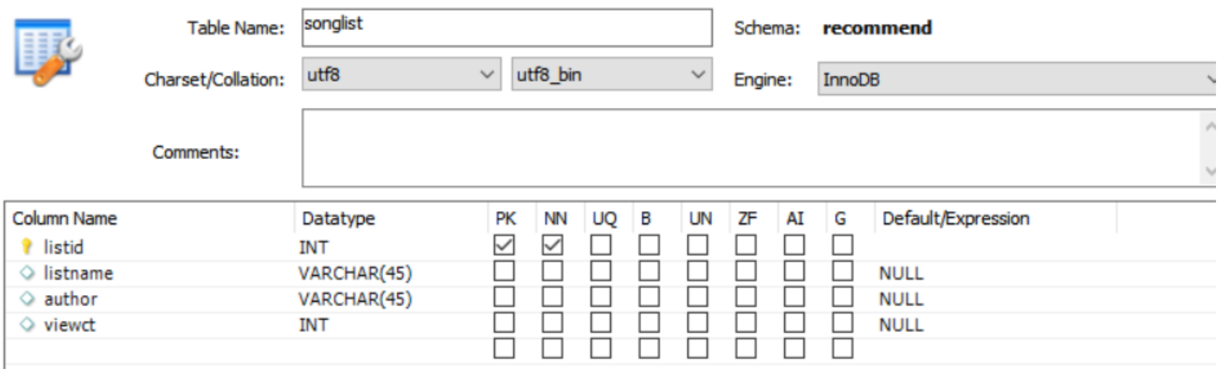


Table Name: songlist
Schema: recommend
Charset/Collation: utf8 utf8_bin
Engine: InnoDB
Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
listid	INT	✓	✓							
listname	VARCHAR(45)									NULL
author	VARCHAR(45)									NULL
viewct	INT									NULL

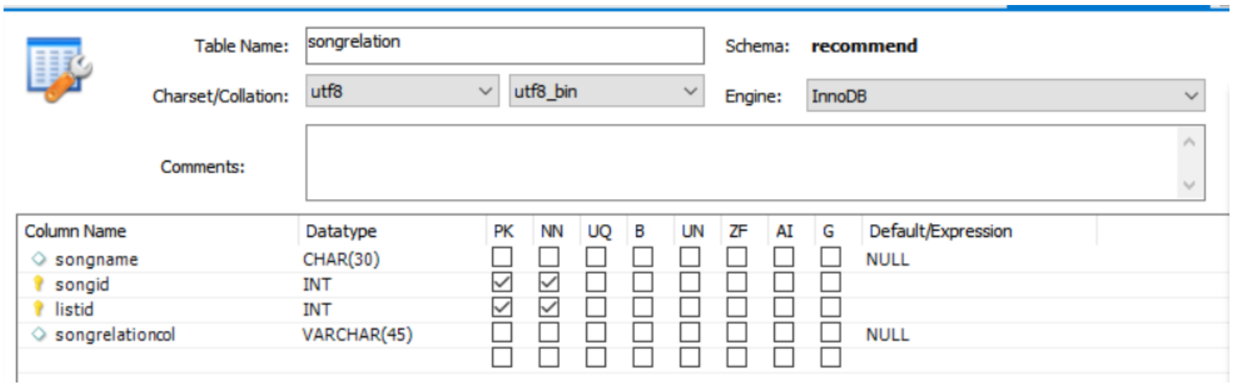


Table Name: songrelation   Schema: **recommend**

Charset/Collation: utf8   utf8_bin   Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
songname	CHAR(30)									NULL
songid	INT	✓	✓							
listid	INT	✓	✓							
songrelationcol	VARCHAR(45)									NULL

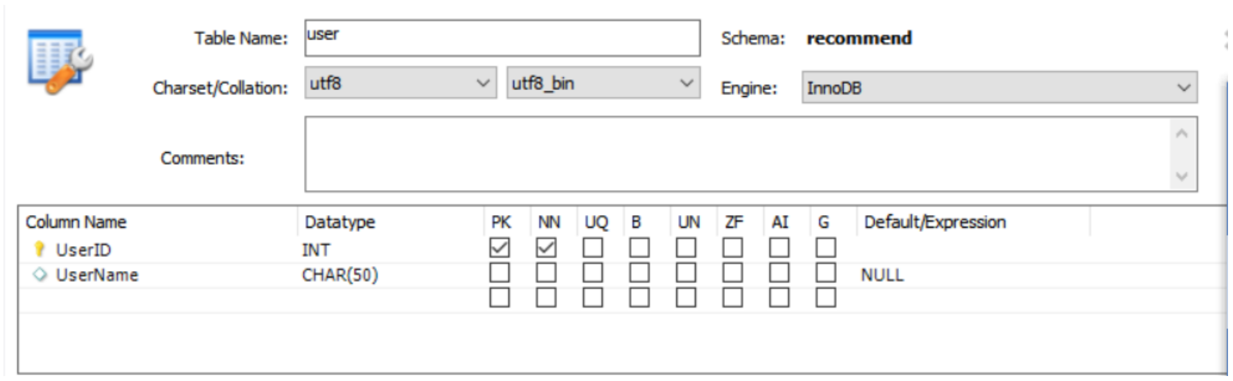


Table Name: user   Schema: **recommend**

Charset/Collation: utf8   utf8_bin   Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
UserID	INT	✓	✓							
UserName	CHAR(50)									NULL

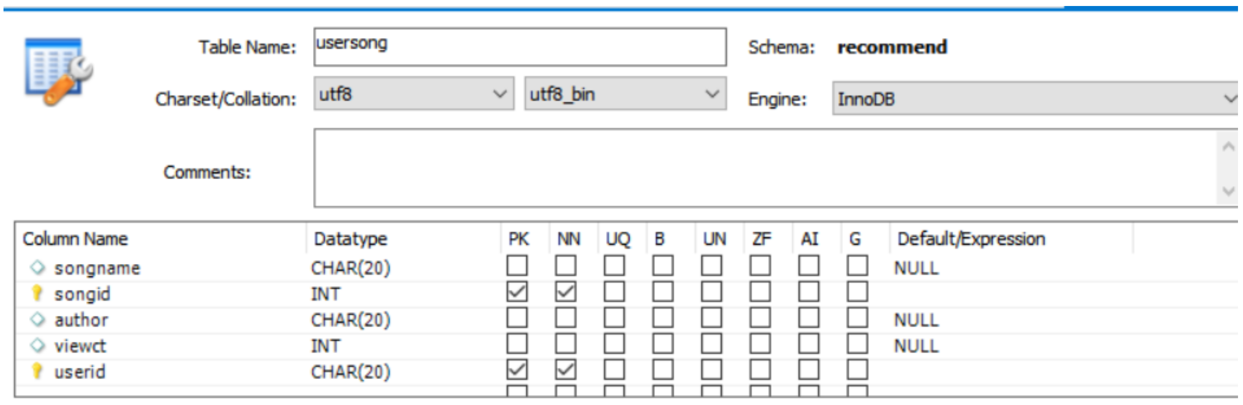
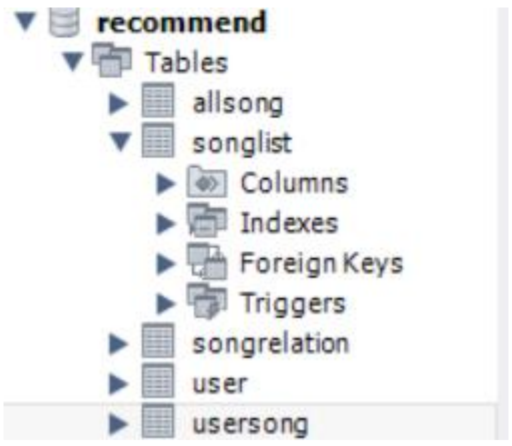


Table Name: usersong   Schema: **recommend**

Charset/Collation: utf8   utf8_bin   Engine: InnoDB

Comments:

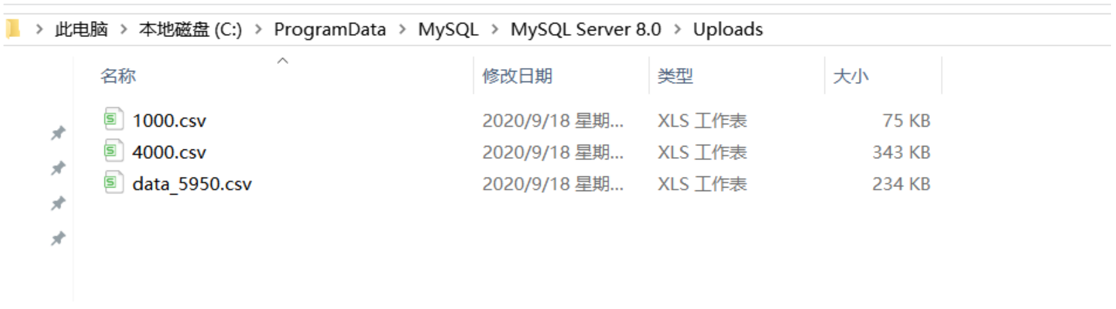
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
songname	CHAR(20)									NULL
songid	INT	✓	✓							
author	CHAR(20)									NULL
viewct	INT									NULL
userid	CHAR(20)	✓	✓							



```
▼ recommend
  ▼ Tables
    ▶ allsong
    ▼ songlist
      ▶ Columns
      ▶ Indexes
      ▶ Foreign Keys
      ▶ Triggers
    ▶ songrelation
    ▶ user
    ▶ usersong
```

3.3 Database environment description

We use mysql 8.0 database, using mysql8.0 will command the crawler CSV files in the database how, spiders crawl the file location in C: / ProgramData/MySQL/MySQL Server 8.0 / Uploads / 4000.



此电脑 > 本地磁盘 (C:) > ProgramData > MySQL > MySQL Server 8.0 > Uploads

名称	修改日期	类型	大小
1000.csv	2020/9/18 星期...	XLS 工作表	75 KB
4000.csv	2020/9/18 星期...	XLS 工作表	343 KB
data_5950.csv	2020/9/18 星期...	XLS 工作表	234 KB

4.The Algorithm

The algorithm is the core part of a recommendation system. It needs to analyze the data of the user's imported song list and the song list of the database to get a recommended song list or song.

	Description	Input	Expected output
1.lst_dist(lst1, lst2)	calculate distance between 2 music lists	lst_dist(lst2music['伤感翻唱版集合'], lst2music['又是一个睡不着的夜晚'])	5.0
2.music_dist(music1, music2)	calculate the distance between 2 musics	music_dist('星河清梦', '繚 星')	5.0
3.most_similar_lst(lst)	get the most similar music list	most_similar_lst(lst2music['刷 题 看书 学习 工作 冥想'])	({'夏日喝汽水lofi', '我去宇宙偷星星，放在夜里等你??', '浮生若梦', '银河赴约 -(网易云音乐助力高考自制曲目)', '你能來保護我的世界嘛', '等', '风的小径', '夜里失联', '这个深海是你的眼眸', '奇怪吗'}, 5.0)

4.most_similar_music(music)	get the most similar music	most_similar_music('无人之岛（翻自 任然）')	('好想好想（翻自 群星）', 5.0)
5.recommend(lst)	recommend according to music list	recommend({'呼吸', '无人之岛（翻自 任然）', '水星记', '像鱼', '大鱼 - (动画电影《大鱼海棠》印象曲)', '千千阙歌(Live)', '心はいつもあなたのそばに Piano'})	{'永不失联的爱', '多远都要在一起', '你的样子', '想い', '是想你的声音啊 - (你快听滴答滴)', '古诗中国', '心领神会', '你还好吗', '不爱我', '玻璃之情', '或是一首歌', '那个她', '秋海棠', '看见你的声音 - (电视剧《想见你》插曲)', '好想好想（翻自 群星）'}

5.Data Scaping

The purpose of data crawling is to provide a large number of data to the database, only with a large number of data recommendation algorithm can be realized.

adequate data
Data Scraping
database

5.1 Data acquisition method

Because the information of songs is nested, it is no longer suitable to use XPath to get data after getting the source code. In this system, selenium and chromdriver are used to obtain data. This is because the requests module is a module that does not completely simulate the browser behavior. It can only crawl to the HTML document information of the web page, and cannot parse and execute CSS and JavaScript code. Therefore, we need to make human judgment. The essence of selenium module is to drive the browser, fully simulate the browser's operation, such as jump, input, click, drop-down, etc., to get the results of web page rendering, and can support a variety of browsers; because selenium parses and executes CSS and JavaScript, its performance is relatively low compared with requests.

1. Selenium installation

    pip install selenium

2. Chromdriver installation

    Download chromdriver.exe , move to the scripts directory in the python installation path.

    Note: the version of chromedriver should correspond to the version of chrome.

3. Selenium selector

    The steps to simulate the browser are as follows:

    Request ---> display page ---> search tag --->click the tag, so the key of selenium is how to find the tag in the page, and then trigger the tag event.

    (1)Positioning by tag ID attribute:

        browser.find_element(By.ID,'').send_keys("")

        browser.find_element_by_id('').send_keys('')

    (2) Positioning by tag name attribute:

        browser.find_element_by_name("").send_keys("")

        browser.find_element(By.NAME,'').send_keys("")

    (3) Positioning by tag name

        browser.find_element_by_tag_name("").send_keys("")

        browser.find_element(By.TAG_NAME, '').send_keys('')

    (4) Positioning through CSS search

        browser.find_element(By.CSS_SELECTOR, '').send_keys('')

        browser.find_element(By.CSS_SELECTOR, '').send_keys(' ')

4. Wait for the element to be loaded

    Selenium only simulates the behavior of the browser. However, it takes time for the browser to parse the page (execute CSS, JS). Some elements may take some time to load. In order to ensure that the elements can be found, we must wait.

    There are two ways to wait:

        Explicit wait: specifies to wait for a tag to finish loading

        Implicit wait: wait for all tags to load

5.2 Data content

After discussion in this group, we decided to get the information we need from Netease cloud music. In order to implement the recommendation system, the following information is important:

Song title: as a result of recommendation to users

Songwriter: used to match users' favorite musicians

Duration: show song details

Song list: recommended for users

5.3 Results

Some data are as follows:

1	歌名	时间	歌手	专辑名字	歌单名称
2	Fashion Bl	4:37	RHYME SC	Fashion Bl	【日语】听这些就可以走路带风
3	Comme D	3:01	Rina Saway	SAWAYAM	【日语】听这些就可以走路带风
4	Transcend	3:34	Ovall	Ovall Rewc	【日语】听这些就可以走路带风
5	御伽の街	3:23	DAOKO	御伽の街	【日语】听这些就可以走路带风
6	MAIGO	3:52	SIRUP/Joe	CIY	【日语】听这些就可以走路带风
7	Lost (Fresh	3:03	End of the	Lost (Fresh	【日语】听这些就可以走路带风
8	RUNAWAY	3:45	Nao Kawar	RUNAWAY	【日语】听这些就可以走路带风
9	In Your Arr	3:07	Aiobahn/R	In Your Arr	【日语】听这些就可以走路带风
10	Hurly Burly	5:12	Perfume	Spending	【日语】听这些就可以走路带风
11	呼吸	4:57	蔡健雅	Tanya 蔡健	你的声音连同气息 穿过秋天漫长的电话线
12	你的样子	5:48	刘莱斯	你的样子	你的声音连同气息 穿过秋天漫长的电话线
13	是想你的声	3:54	傲七爷	是想你的声	你的声音连同气息 穿过秋天漫长的电话线
14	永不失联的	4:19	周兴哲	如果雨之后	你的声音连同气息 穿过秋天漫长的电话线
15	你还好吗	4:34	吴大文	你还好吗	你的声音连同气息 穿过秋天漫长的电话线
16	看见你的声	4:15	陈零九	看见你的声	你的声音连同气息 穿过秋天漫长的电话线
17	心领神会	4:16	莫文蔚	我们在中场	你的声音连同气息 穿过秋天漫长的电话线
18	或是一首歌	4:34	田馥甄	或是一首歌	你的声音连同气息 穿过秋天漫长的电话线
19	多远都要在	3:37	G.E.M.邓紫	新的心跳	你的声音连同气息 穿过秋天漫长的电话线
20	秋海棠	3:44	澈澈limpic	秋海棠	你的声音连同气息 穿过秋天漫长的电话线
21	Dance Like	3:02	Iggy Azale	Dance Like	街头扮酷指南 | 小心别被节奏带跑偏
22	imma	2:03	bbno$/Ler	imma	街头扮酷指南 | 小心别被节奏带跑偏
23	Baggin'	3:17	Marshmell	Baggin'	街头扮酷指南 | 小心别被节奏带跑偏
24	Endorphin	3:25	tobi lou	Endorphin	街头扮酷指南 | 小心别被节奏带跑偏
25	LOCKED U	3:23	6ix9ine/Ak	TattleTales	街头扮酷指南 | 小心别被节奏带跑偏
26	Lucky Mist	2:55	Vincent/Ali	Lucky Mist	街头扮酷指南 | 小心别被节奏带跑偏
27	Lie to Me	2:54	BLOWFEVE	Lie to Me	街头扮酷指南 | 小心别被节奏带跑偏
28	Ring	2:54	T.I./Young	Ring (feat.	街头扮酷指南 | 小心别被节奏带跑偏
29	Kobe	2:51	Dame D.O	Kobe (feat.	街头扮酷指南 | 小心别被节奏带跑偏
30	99 Problen	2:17	Hugo	Old Tyme	街头扮酷指南 | 小心别被节奏带跑偏