



音乐推荐系统

文件状态： [] 草稿 [✓] 正式发布 [] 正在修改	文件标识：	音乐推荐系统项目文档
	当前版本：	1.0
	作 者：	2020111136, 刘祖帆
	完成日期：	2023. 6. 26

上海财经大学信息管理与工程学院

目录

音乐推荐系统	1
1. 产品介绍	3
1.1 产品定义	3
1.2 产品开发背景	3
1.3 产品主要功能和特色	3
1.4 产品范围	4
2. 用户界面设计	4
2.1 前端界面逻辑框架	4
2.2 登录/注册页	4
2.3 首页	4
2.4 歌曲详情页	5
2.5 个性化音乐推荐	6
2.6 数据分析	7
3. 系统软硬件平台	7
3.1 系统开发平台	7
4. 项目架构	8
5. 算法设计	8
5.1 基于用户相似度的协同过滤推荐 (UserCF)	8
5.2 基于矩阵分解的协同过滤推荐 (FM)	8
6. 数据库设计	9
7. Django 项目解析	10
7.1 项目结构	10
7.2 urls 解析	12
7.3 Views 解析	13
7.4 Templates 解析	14
7.5 Models 解析	15
8. 关键技术/功能	15
8.1 权限管理和登录验证	15
8.2 ajax 异步更新	15
8.3 ECharts 数据可视化分析	16
8.4 推荐算法	16
8.5 Paginator 分页	16
9. 项目自评	16
9.1 优点	16
9.2 缺点	16

1. 产品介绍

本产品是一个为音乐爱好者提供音乐相关功能的平台，致力于为广大音乐爱好者、音乐传播者（如歌手）提供一个音乐分享、交流的平台。本平台的核心功能包括不同角色的权限管理、音乐推荐、数据分析、用户评价与讨论等。

1.1 产品定义

该音乐推荐系统是一个实用型网络应用，致力于为用户提供个性化推荐、数据分析以及互动体验。产品通过收集用户对音乐的历史评价信息，为用户不同用户进行个性化推荐，同时提供检索、分享、讨论、评价等多种功能，使用户能够得到良好的使用体验。

1.2 产品开发背景

随着互联网的发展，大数据的到来，传统的音乐行业受到了很大的冲击，原有的音乐数字化给人们的生活带来了极大的便利。随着数字音乐的兴起，各大音乐平台层出不穷，人们在音乐平台上收听音乐的时，常常因为歌曲信息繁杂，而不能找到自己想听的音乐。为了解决这个问题，音乐领域引入了推荐系统。

目前所有主流的音乐平台在实现了原有功能的基础上都增加了推荐功能的实现，如何设计出一款满足用户和平台的推荐系统具有重大意义。对于用户来说，优秀的推荐系统可以通过分析用户历史行为数据挖掘出用户潜在偏好从而为用户进行推荐，提高用户体验感；对于平台来说，优秀的推荐系统可以帮助平台挽留更多的用户，避免用户的流失造成经济损失。

基于以上，本文基于协同过滤推荐算法，设计了一个音乐推荐系统，希望为不同音乐爱好者提供个性化推荐。

1.3 产品主要功能和特色

产品功能列表：

- (1) 用户的注册和登陆；
- (2) 音乐榜单和当日热点推荐；
- (3) 针对特定用户的个性化推荐；
- (4) 对歌曲受欢迎程度等的数据分析与可视化；
- (5) 用户关于音乐评分与评价的区域；
- (6) 互联网音乐资源整合。

产品特色：

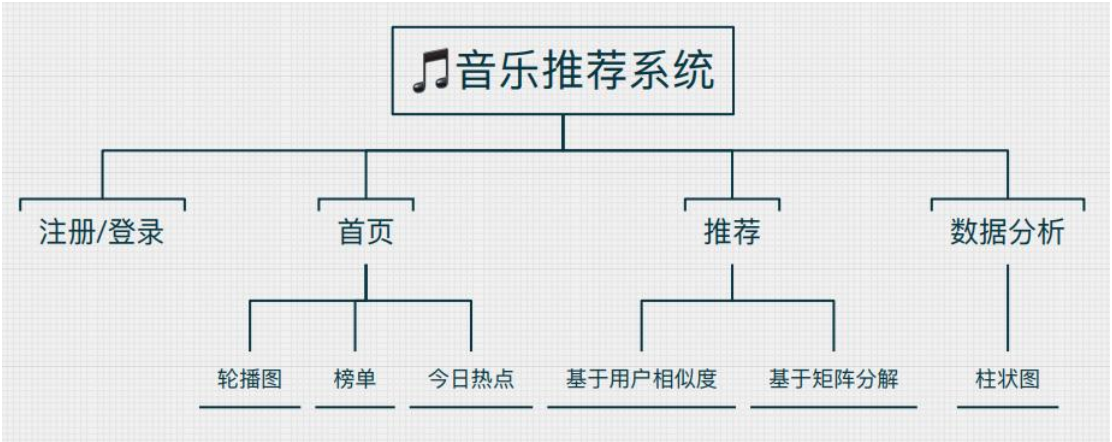
- (1) 从互联网整理音乐相关信息，并根据项目特点进行创新性利用；
- (2) 给音乐爱好者提供交流、评价的平台；
- (3) 给音乐爱好者提供个性化推荐；
- (4) 给歌手提供关于歌曲受欢迎程度等的数据分析功能。

1.4 产品范围

产品适合音乐爱好者、歌手、唱片公司以及想要了解音乐文化的所有人。

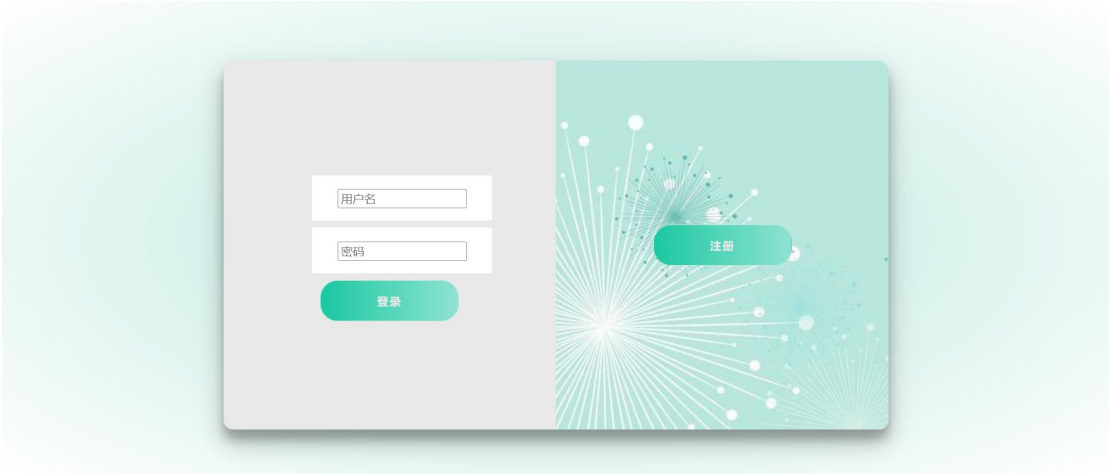
2. 用户界面设计

2.1 前端界面逻辑框架



用户界面主要分为首页、推荐、数据分析以及注册/登录四大板块，界面友好，简洁清晰，自适应布局。

2.2 登录/注册页



2.3 首页

音乐推荐系统

首页推荐登录

搜索

Q

祝你平安

毛不易

电影《保你平安》主题曲

影视

祝你平安

榜单

红绝

Enchanted

Love is a storm

Lucid Dream

Red

夜游

拥抱

云烟成雨

作罢

如果可以

今日热点

红绝

发行时间: Oct. 19, 2020

Enchanted

发行时间: Oct. 25, 2010

Love is a storm

发行时间: Sept. 21, 2022

个性化音乐推荐系统

左边是近段时间的宣传图轮播。点击“榜单”和“今日热点”可跳转至歌曲详情页。

2.4 歌曲详情页

清梦! 生如夏花般绚丽! June 25, 2023

yydsJune 26, 2023

清梦! 生如夏花般绚丽! June 25, 2023

xrh好美June 25, 2023

夏花因此曲封神June 25, 2023

127.0.0.1:8000

评分修改成功

☐ 不允许 127.0.0.1:8000 再次向您提示

确定

清梦! 生如夏花般绚丽! June 25, 2023

yydsJune 26, 2023

清梦! 生如夏花般绚丽! June 25, 2023

xrh好美June 25, 2023

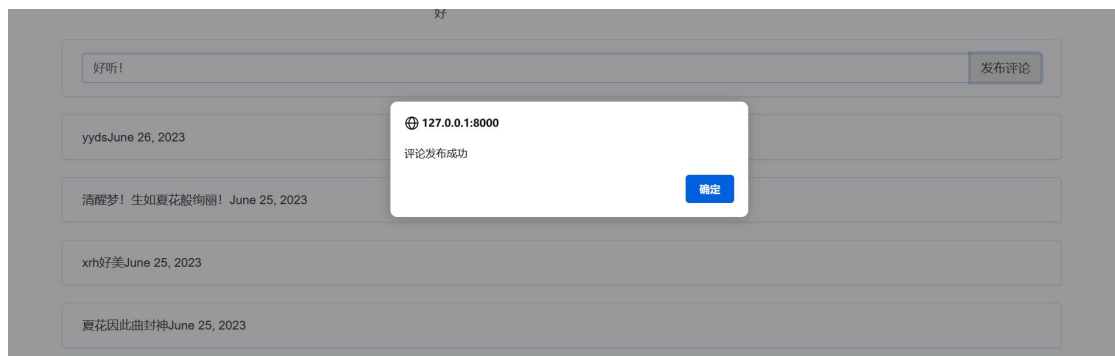
夏花因此曲封神June 25, 2023

127.0.0.1:8000

评分修改成功

☐ 不允许 127.0.0.1:8000 再次向您提示

确定



歌曲详情页，用户可以查看歌曲详细信息、播放歌曲，以及对歌曲评分、发布评论等。

2.5 个性化音乐推荐

该页面基于用户的历史评分数据，针对用户未评分过的歌曲，为不同用户提供基于用户相似度的协同过滤、基于矩阵分解的协同过滤两种算法的个性化推荐结果，并做分页（每页6个项目）展示。以两个用户为例：

针对用户（用户名：贝叶斯定理，密码：musicRec_normal）的个性化推荐结果：

音乐推荐系统

首页推荐退出登录

基于用户相似度的协同过滤推荐

红绝

Enchanted

Love is a storm

Red

拥抱

云烟成雨

上一页12下一页

基于矩阵分解的协同过滤推荐

红绝

Enchanted

Love is a storm

Red

拥抱

云烟成雨

上一页12下一页

针对用户（用户名：liberty，密码：musicRec_normal）的个性化推荐结果：

音乐推荐系统

首页推荐退出登录

基于用户相似度的协同过滤推荐

Enchanted

Love is a storm

Lucid Dream


Red

拥抱

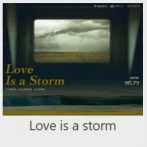
作罢

上一页12下一页

基于矩阵分解的协同过滤推荐



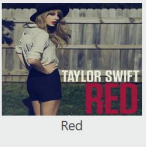
Enchanted



Love is a storm



Lucid Dream



Red

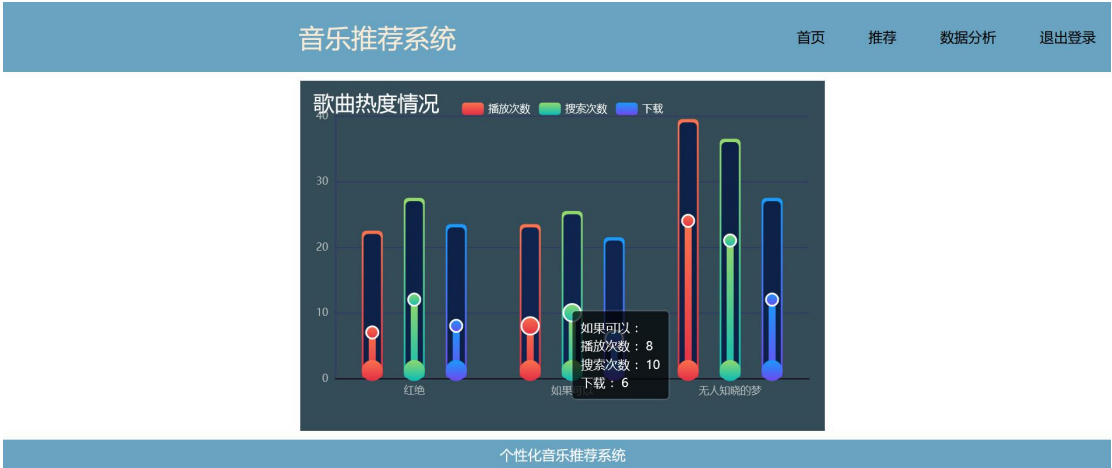


拥抱



作罢

2.6 数据分析



以用户（用户名：胡夏，密码：musicRec_singer）为例。以歌手(Group为singer)身份登录时，才能看到“数据分析”页面。该图横坐标为该歌手曾发布的歌曲，纵坐标为对应歌曲的播放次数、搜索次数、下载次数。

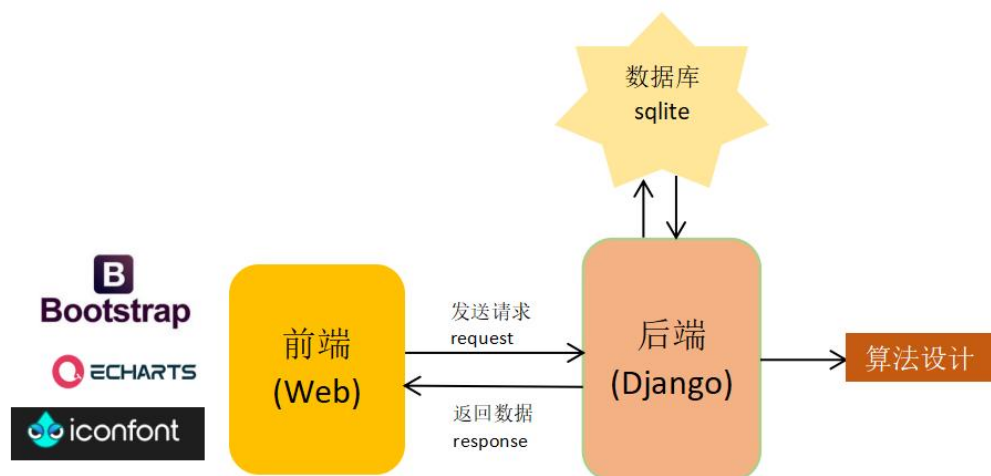
3. 系统软硬件平台

3.1 系统开发平台

表格 1 系统开发平台

项目名称	环境名称
系统环境	Windows 10 操作系统
开发软件	Visual Studio Code
硬件环境	CPU: Intel (R) Core (TM) i5-1135G7, 内存: 8GB, 硬盘: 256G
数据库系统	Sqlite

4. 项目架构



本项目采用Django框架进行开发。Django遵循 MTV 的设计模式。MTV是Model、Template、View三个单词的简写，分别代表模型、模版、视图。此外，通过 URL 分发器，将一个个 URL 的页面请求分发给不同的 View 处理，View 再调用相应的 Model 和 Template。

用户发送请求(request)，这个请求会去访问视图函数，如果不涉及到数据调用，那么这个时候视图函数直接返回一个模板也就是一个网页给用户；如果涉及到数据调用，那么视图函数调用模型，模型去数据库查找数据，然后逐级返回。视图函数把返回的数据填充到模板中空格，最后返回网页给用户。

5. 算法设计

5.1 基于用户相似度的协同过滤推荐(UserCF)

(1) 计算用户之间的相似度

$$w_{uv} = \frac{|N(u) \cap N(v)|}{\sqrt{|N(u)||N(v)|}}$$

其中 $N(u)$ 表示用户 u 评价过的物品集合，其中 $N(v)$ 表示用户 v 评价过的物品集合。

(2) 计算用户对新物品的评分预测

$$R_{up} = \frac{\sum_{v \in S} (w_{uv} * R_{vp})}{\sum_{v \in S} w_{uv}}$$

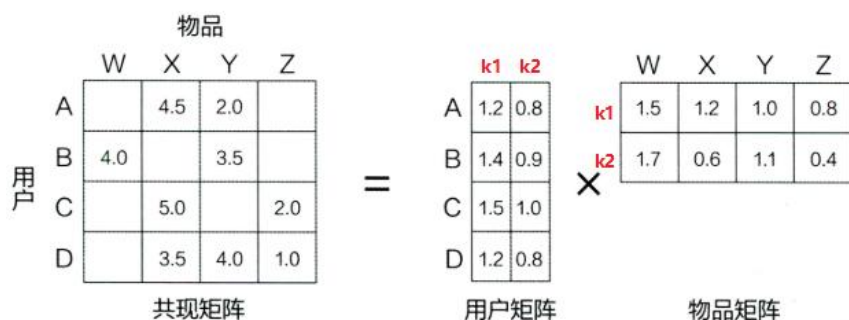
其中 S 表示与用户 u 最相似的 $Top-n$ 好友集合。

(3) 对用户进行物品推荐

5.2 基于矩阵分解的协同过滤推荐(FM)

(1) 得到用户和物品的隐向量

在矩阵分解的算法框架下，可以通过分解协同过滤的共现矩阵（评分矩阵）来得到用户和物品的隐向量，原理如下：



矩阵分解算法将 $m * n$ 维的共享矩阵 R ，分解成 $m * k$ 维的用户矩阵 U 和 $k * n$ 维的物品矩阵 V 相乘的形式。其中， m 是用户数量， n 是物品数量， k 是隐向量维度，也就是隐含特征个数，需要模型自己去学习。

(2) 计算用户对新物品的评分预测

将用户和物品的隐向量做内积得到预测评分。

(3) 对用户进行物品推荐

6. 数据库设计

表 1: 数据库基本信息

表名	说明
歌曲分类表	歌曲类别信息
歌曲信息表	歌曲详细信息
歌曲动态表	歌曲播放次数等动态信息
歌曲评论表	用户对歌曲的评论信息
歌曲评分表	用户对歌曲的评分信息
Users (Django 自带)	用户基本信息
Groups (Django 自带)	用户组别信息

表 2: 歌曲分类表

字段名称	字段含义	数据类型	约束
label_id	序号	AutoField	主键，自增
label_name	分类标签	CharField	非空，最大长度 10

表 3: 歌曲信息表

字段名称	字段含义	数据类型	约束
song_id	序号	AutoField	主键，自增
song_name	歌名	CharField	非空，最大长度 50
song_time	时长	CharField	非空，最大长度 10
song_album	专辑	CharField	非空，最大长度 50
song_languages	语种	CharField	非空，最大长度 20
song_release	发行时间	DateField	非空，最大长度 20
song_img	歌曲图片	CharField	非空，最大长度 100

song_lyrics	歌词	CharField	非空，最大长度 500
song_file	歌曲文件	CharField	非空，最大长度 100
label	类型	ForeignKey	非空，关联歌曲分类表
singer	歌手	ManyToManyField	非空，关联 User 表

表 4: 歌曲动态表

字段名称	字段含义	数据类型	约束
dynamic_id	序号	AutoField	主键，自增
dynamic_plays	播放次数	IntegerField	非空
dynamic_search	搜索次数	IntegerField	非空
dynamic_down	下载次数	IntegerField	非空
song	歌曲	ForeignKey	非空，关联歌曲表

表 5: 歌曲评论表

字段名称	字段含义	数据类型	约束
comment_id	序号	AutoField	主键，自增
comment_text	内容	CharField	非空，最大长度 500
comment_date	日期	DateField	非空，最大长度 50
comment_user	用户	ForeignKey	非空，关联 User 表
song	歌曲	ForeignKey	非空，关联歌曲表

表 6: 歌曲评分表

字段名称	字段含义	数据类型	约束
rating_id	序号	AutoField	主键，自增
rating_number	内容	CharField	非空
rating_date	日期	DateField	非空，最大长度 50
rating_user	用户	ForeignKey	非空，关联 User 表
song	歌曲	ForeignKey	非空，关联歌曲表

7. Django 项目解析

7.1 项目结构

```

MUSICRECOMMENDATION
├── musicRecommendation #project
│   ├── __pycache__
│   ├── __init__.py
│   ├── asgi.py
│   ├── wsgi.py
│   └── settings.py

```

```
|   ├── urls.py
|   └── views.py
└── play #app
    ├── __pycache__
    ├── migrations
    ├── __init__.py
    ├── admin.py
    ├── apps.py
    ├── models.py
    ├── urls.py
    └── views.py
└── user #app
    ├── __pycache__
    ├── migrations
    ├── __init__.py
    ├── admin.py
    ├── apps.py
    ├── models.py
    ├── urls.py
    └── views.py
└── recommend #app
    ├── __pycache__
    ├── migrations
    ├── __init__.py
    ├── admin.py
    ├── apps.py
    ├── models.py
    ├── urls.py
    └── views.py
└── static
    ├── css
    ├── js
    ├── image
    ├── songFile
    ├── songImg
    └── songLyric
└── templates
    ├── base.html
    ├── index.html
    ├── play
    │   ├── detail.html
    │   └── recommend.html
    ├── user
    │   └── login.html
```

```
| | └─ analysis.html
| └─ db.sqlite3
└─ manage.py
```

该 project 命名为 musicRecommendation, 共包含 3 个 app, 分别为 play、user、recommend, play 主要与歌曲相关, user 主要与用户相关, recommend 主要与推荐相关。

static 存储项目所需的静态文件, templates 存储项目所需的模板文件。

7.2 urls 解析

Project: musicRecommendation 的 url.py

```
musicRecommendation > urls.py > ...
1  from django.contrib import admin
2  from django.urls import path, include
3  from . import views
4
5  urlpatterns = [
6      path("admin/", admin.site.urls),
7
8      path("", views.index_view, name="index"),
9
10     path("play/", include("play.urls")),
11     path("user/", include("user.urls")),
12     path("recommend/", include("recommend.urls")),
13 ]
14
```

App:play 的 url.py

```
play > urls.py > ...
1  from django.urls import path
2
3  from . import views
4  app_name='play'
5
6  urlpatterns = [
7      path('detail/', views.detail_view, name='detail'),
8      path('rating/', views.rating, name='rating'),
9      path('comment/', views.comment, name='comment'),
10 ]
```

App:user 的 url.py

```
user > urls.py > ...
1  from django.urls import path
2
3  from . import views
4  app_name='user'
5
6  urlpatterns = [
7      path('login/', views.LoginView.as_view(), name="login"),
8      path('logout/', views.to_logout, name="logout"),
9
10     path('analysis/', views.analysis, name='analysis')
11 ]
```

App:recommend 的 url.py

```

recommend > urls.py > ...
1  from django.urls import path
2
3  from . import views
4  app_name='recommend'
5
6  urlpatterns = [
7      path('recommendation/', views.recommendation_view,name='recommendation'),
8  ]

```

7.3 Views 解析

Project: musicRecommendation 的 views.py

```

musicRecommendation > views.py > ...
1  from django.http import HttpResponse
2  from django.shortcuts import render
3  from play.models import *
4
5
6  def index_view(request):
7
8      monthlyTopSongList=Song.objects.all()[:10]
9      dailyTopSongList=Song.objects.all()[:3]
10
11     return render(request,'index.html',locals()) #返回当前位置的全部局部变量

```

App:play 的 views.py

```

play > views.py > ...
1  from django.shortcuts import render, redirect
2  from django.http import JsonResponse
3  from .models import *
4  from django.contrib.auth.models import User
5  from django.contrib.auth.decorators import login_required
6  from datetime import date
7
8  # Create your views here.
9
10 @login_required(login_url='user:login')
11 > def detail_view(request): ...
26
27 > def rating(request): ...
47
48 > def comment(request): ...

```

App:user 的 views.py

```

user > views.py > ...
1  from django.shortcuts import render,redirect
2  ##表单
3  from django import forms
4  ##CBV基于类的视图
5  from django.views import View
6  ##django自带的User和Group模型
7  from django.contrib.auth.models import User,Group
8  ##登录验证
9  from django.contrib.auth import authenticate,login,logout
10 from django.contrib.auth.decorators import login_required
11 from play.models import *
12
13 ##登录表单
14 > class LoginForm(forms.Form): ...
25
26 > class LoginView(View): ...
55
56 > def to_logout(request): ...
59
60 @login_required(login_url='user:login')
61 > def analysis(request): ...

```

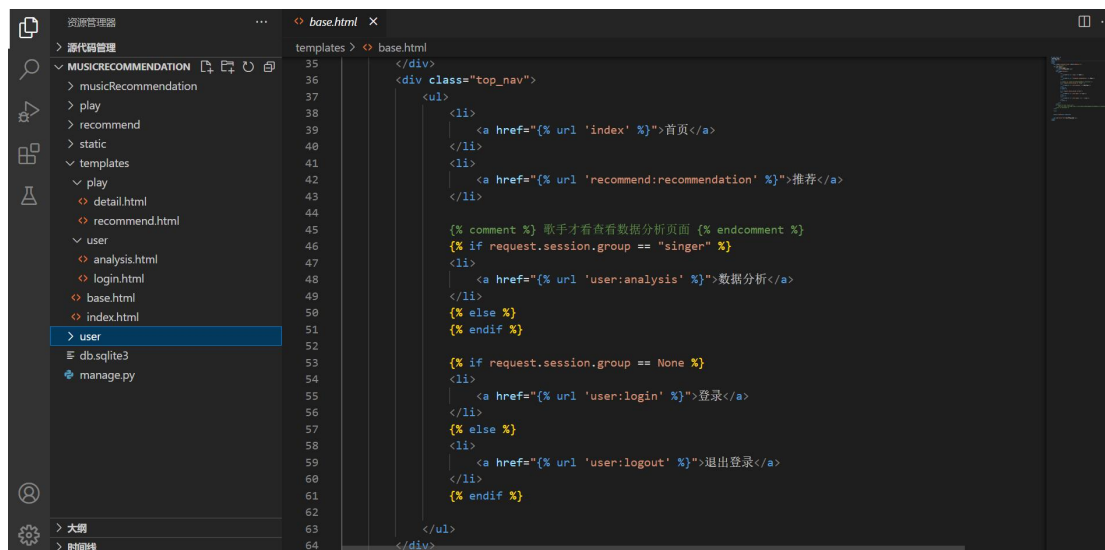
App:recommend 的 views.py

```

recommend > views.py > ...
1  from django.shortcuts import render
2  from django.contrib.auth.decorators import login_required
3  #数据分析
4  import numpy as np
5  import pandas as pd
6  import math
7  import random
8  #分页
9  from django.core.paginator import Paginator , PageNotAnInteger,EmptyPage
10
11 from play.models import *
12
13 # Create your views here.
14 @login_required(login_url='user:login')
15 > def recommendation_view(request): ...
79
80 #基于用户相似度的协同过滤推荐
81 > def userCF(users,unrated_items,current_user): ...
121
122 #基于矩阵分解的协同过滤推荐
123
124 > class BiasSVD(): ...
175
176
177 > def FM_BiasSVD(users,unrated_items,current_user): ...
192

```

7.4 Templates 解析



7.5 Models 解析

```
play > models.py > ...
1  from django.db import models
2  from django.contrib.auth.models import User
3
4  # Create your models here.
5  # 歌曲分类表
6  > class Label(models.Model): ...
16
17
18  # 歌曲信息表
19  > class Song(models.Model): ...
38
39
40  # 歌曲动态表
41  > class Dynamic(models.Model): ...
51
52
53  # 歌曲评论表
54  > class Comment(models.Model): ...
64
65
66  # 歌曲评分表
67  > class Rating(models.Model): ...
77
78
```

8. 关键技术/功能

8.1 权限管理和登录验证

采用 Django 的用户认证（Auth）组件，用于判断当前的用户是否合法，登录成功跳转至首页。

项目中首页可在未登录状态下进行访问，若需访问首页之外其它页面，需要先成功登录。身份为 singer 时可以查看“数据分析”页面；身份为 normal 时则不可以。

8.2 ajax 异步更新

本项目采用 ajax (Asynchronous JavaScript and XML) 进行网络请求，可以在不重载整

个网页的情况下，通过后台加载数据，并在网页上进行显示。本项目的评分和评论功能使用 ajax 技术。

8.3 ECharts 数据可视化分析

ECharts 是一个使用 JavaScript 实现的开源可视化库，涵盖各行业图表，满足各种需求。本项目中以 singer 身份登录时，可以查看自己的歌曲的受欢迎程度柱状图，其中横坐标为该歌手曾发布的歌曲，纵坐标为对应歌曲的播放次数、搜索次数、下载次数。

8.4 推荐算法

本项目导入 numpy、pandas、math 等，对用户的历史评分数据，使用基于用户相似度的协同过滤推荐与基于矩阵分解的协同过滤两种算法为用户生成个性化推荐列表。

8.5 Paginator 分页

使用 Paginator 组件对推荐结果进行分页展示。

9. 项目自评

9.1 优点

本项目基本综合了老师课堂上讲到过的内容和所有的案例，包括 bootstrap 前端组件库、jquery、数据库增删查改、ECharts 图表、ajax 异步请求、Paginator 分页，以及利用表单 Form 组件和用户认证 Auth 组件实现了多角色（singer 和 normal）用户的分权限管理。

除此之外，本项目还增加了算法设计的部分。利用用户对音乐的历史评分数据，分别利用基于用户相似度的协同过滤推荐、基于矩阵分解的协同过滤推荐进行个性化的音乐推荐。

9.2 缺点

由于时间有限，本项目的一些功能还未实现，如注册功能等，在接下来的时间里可以进一步修改并完善。

目前推荐系统算法比较简单，数据量较小，为用户进行推荐时，需要遍历数据库的所有历史评分数据。未来需要进一步完善推荐算法，如更大的数据量，使用更准确、更高效的算法。