大连海事大学 2021-2022（2）学期　　序号：22

移动应用开发课程实践报告

项目名称：手机音乐 APP

项目组名称：Stella

项目组成员：林晨、杨灏哲、贾征强、徐达

姓名		查卷了任务	成绩
林晨	2220191537	选爸题，制定开发计划，需求分析，系统设计，UI 设计与实现，服务器搭建，后端部署，负责前后端的交互，歌曲录入、检索部分功能编码，统筹项目进度，视频制作。	
杨灏哲	2220191651	需求分析，负责 Git 的建立和维护，数据库设计，概要设计，navigation 部分实现，音乐播放部分功能编码，报告撰写。	
贾征强	2220192615	需求分析，运行测试，数据流程图制作，Spring Boot 搭建，运行测试，社交部分功能编码，报告撰写。	
徐达	2220192614	需求分析，数据字典编写，汇总数据，业务流程图，注册登录功能编码，报告撰写。	

项目组成员完成工作量：

姓名	学号	主要完成工作	占总工作量比例
林晨	2220191537	选题，制定开发计划，需求分析，系统设计，UI 设计与实现，服务器搭建，后端部署，负责前后端的交互，歌曲录入、检索部分功能编码，统筹项目进度，视频制作。	30
杨灏哲	2220191651	需求分析，负责 Git 的建立和维护，数据库设计，概要设计，navigation 部分实现，音乐播放部分功能编码，报告撰写。	25
贾征强	2220192615	需求分析，运行测试，数据流程图制作，Spring Boot 搭建，运行测试，社交部分功能编码，报告撰写。	25
徐达	2220192614	需求分析，数据字典编写，汇总数据，业务流程图，注册登录功能编码，报告撰写。	20

目录

第一章 系统分析

1.1 需求描述

1.1.1 系统开发背景

音乐的魅力在生活中是极其大的，不同的国家、不同语言的人，可以从音乐中体会到相同的情感，可以加强人与人之间的联系，我们也可以从音乐中了解他国，因为音乐是人类共同的食粮，它也可以可以让身体放轻松，缓解压力。因此为使人们能够随时听到音乐，我们需要开发一款音乐系统，为用户提供了便利，使得人们可以通过手机等设备在任何时候欣赏到音乐。

1.1.2 可行性分析

- 技术可行性：提供快速的搜索匹配，同时给用户提供交流讨论的平台
- 经济可行性：开发成本相对较低，同时符合当代社会简单快捷的特点
- 运营可行性：在小的音乐圈试点，然后慢慢扩大使用者范围
- 社会因素可行性分析：方便实用的音乐播放器便于被人们接收。

1.1.3 需求描述

- 登录、注册功能：通过此功能对用户身分经行识别和用户基本信息的存储。
- 歌曲录入功能：歌曲管理人员承担的是对平台上所有上传歌曲的管理工作，对库中的音乐进行审核、统计等管理
- 歌曲检索功能：基于文本检索通过输入歌曲名、歌手名或者歌词来检索歌曲，通过对音乐库中的音乐进行特征标记完成，每首音乐都有歌名、歌手和歌词信息。同时还可以根据歌词片段搜索，也可以根据不同地区，风格，歌手来搜索
- 歌曲播放功能：进行音乐的播放、暂停、调整播放顺序等基本的播放器功能
- 评论：用户可以对歌曲发表自己的见解
- 歌单：用户可以创建歌单对歌曲进行管理

1.2 业务流程图

业务过程是管理各类资源的各种相关活动和决策的组合。管理人员，通过管理这些资源，支持管理目标。定义业务过程是 BSP 方法的核心，业务过程应独立于组织机构，要从企业的全部管理工作中，分析归纳出相应的业务过程。

绘制业务流程图，具体业务流程图如下。

用户业务图

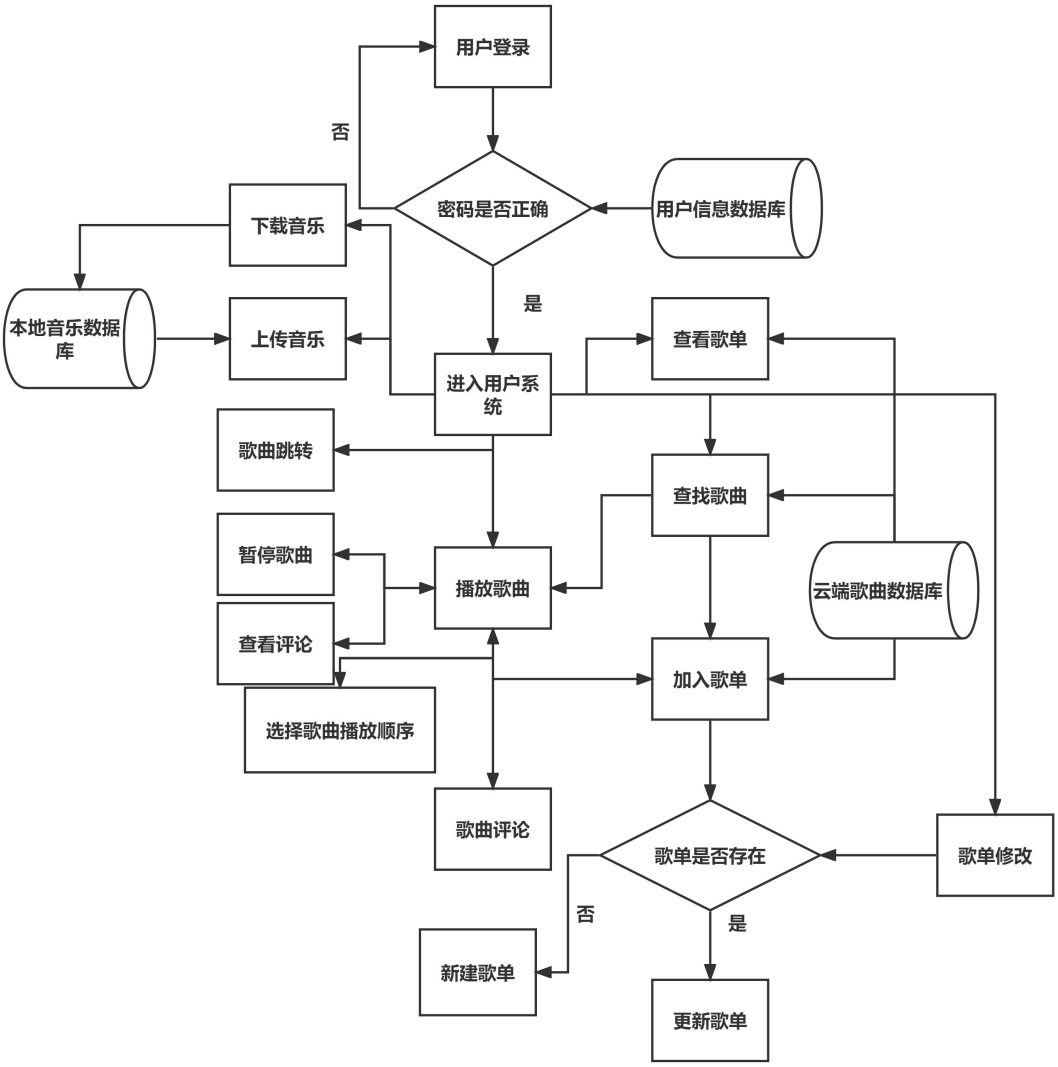

图 1.1 用户业务流程图

管理员业务图

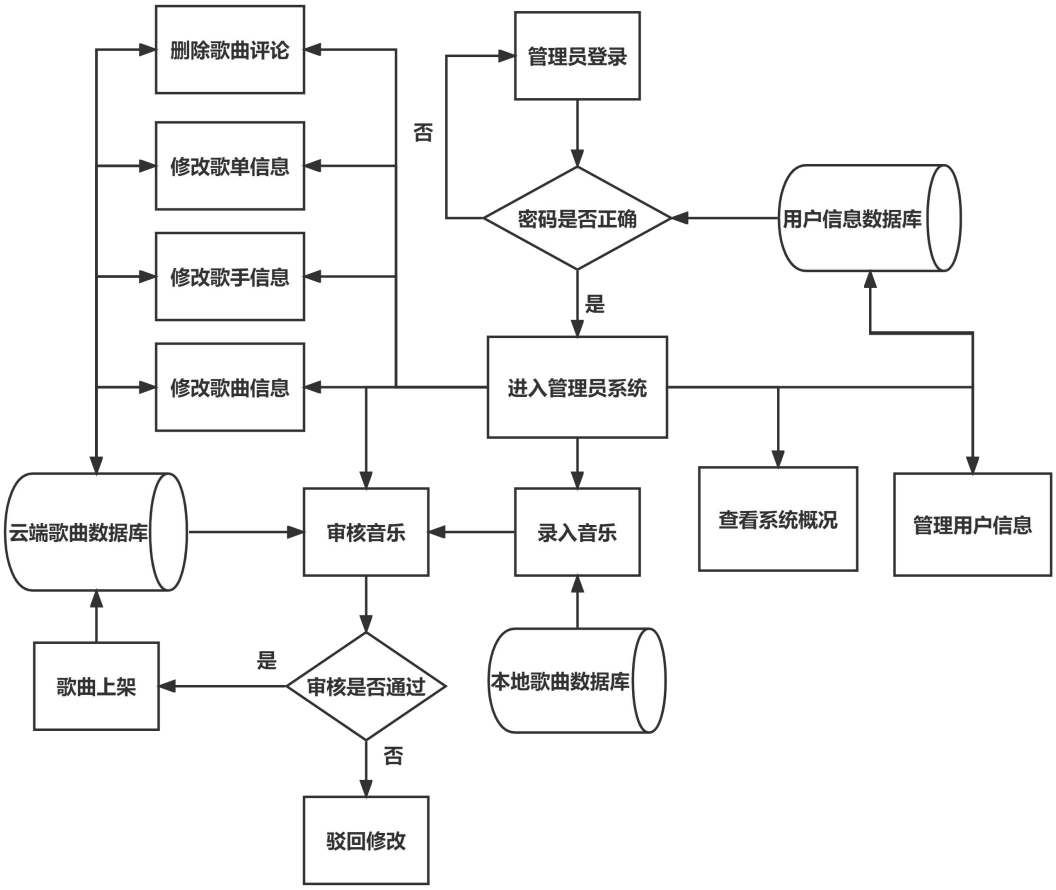

图 1.2 管理员业务流程图

1.3 数据流程图

数据流程图（Data Flow Diagram，DFD/Data Flow Chart），简称数据流图，是一种描述系统数据流程的主要工具，它用一组符号来描述整个系统中信息的全貌，综合地反映出信息在系统中的流动、处理和存储情况。

数据流程图有两个特征:

（1）抽象性

数据流程图把具体的组织机构、工作场所、物质流都去掉，只剩下信息和数据存储、流动、使用以及加工情况

（2）概括性

指数据流程图把系统对各种业务的处理过程联系起来考虑，形成一个总体。

我们通过对系统需求的分析和业务流程的分析，将整个系统划分为用户信息管理子系统、歌曲管理子系统、歌单管理子系统、歌曲评论管理子系统、管理员子系统。以下是我们的数据流图：

顶层数据流图：

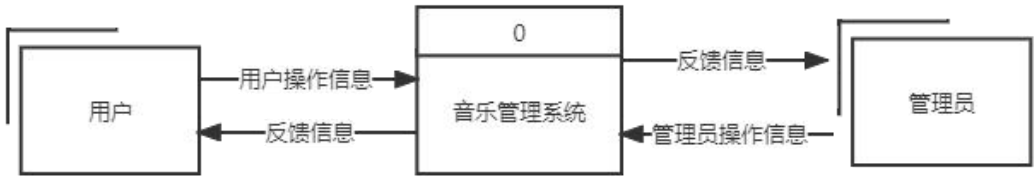

图 1.3 顶层数据流图

用户和管理员通过向音乐管理系统输入操作信息，系统经过处理后，形成相应的反馈信息，在图形界面展示出来。

第一层数据流图：

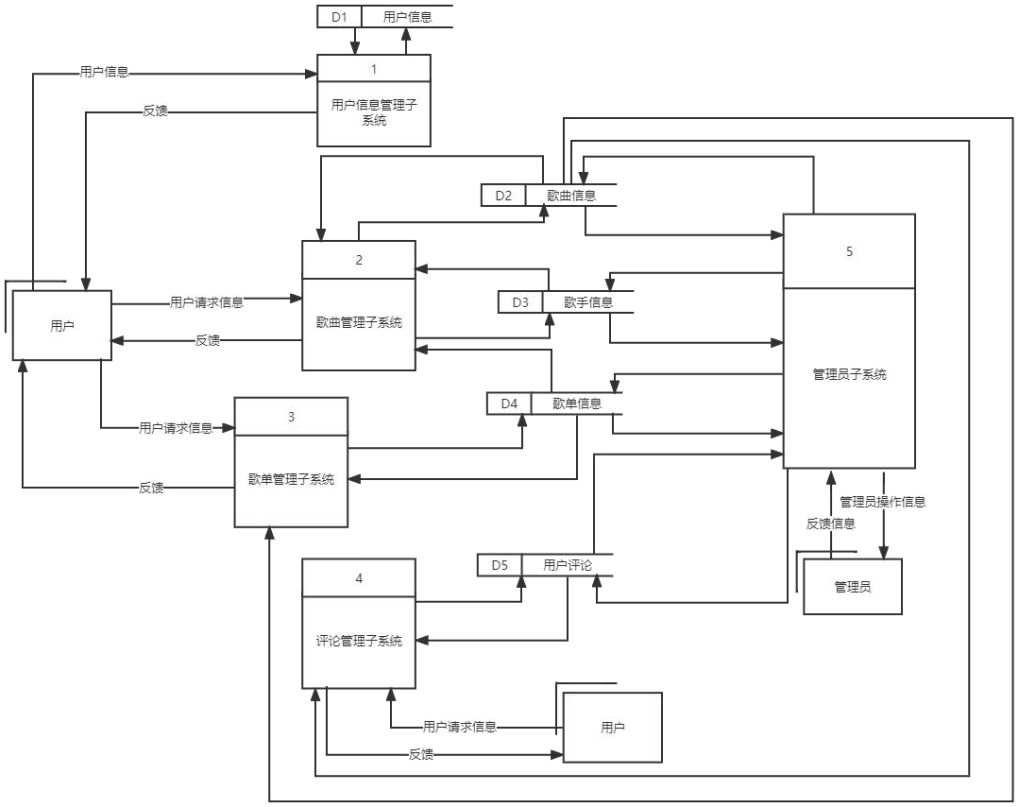

图 1.4 第一层数据流图

用户将用户信息输入到用户信息管理子系统，子系统通过与用户信息表交互，对用户信息实现增删改查，并生成反馈信息，在用户界面显示。

用户将歌曲管理请求信息（包括歌曲查看、查询、上传、下载）输入到歌曲管理子系统中，子系统通过调用相应的模块，进行歌曲信息的管理，并生成反馈信息，在用户界面显示。

用户将歌单管理请求信息（包括歌单创建、删除、修改、分享、查看）输入到歌单管理子系统中，子系统通过调用相应的模块，进行歌曲信息的管理，并生成反馈信息，在用户界面显示。

用户将评论管理请求信息（包括评论创建、查看、修改、删除）输入到歌曲评论管理子系统中，子系统通过调用相应的模块，进行评论信息的管理，并生成反馈信息，在用户界面显示。

管理员将管理操作信息（包括管理歌曲、评论、歌单、用户、歌手信息和查看平台概况请求）输入到管理员子系统中，子系统通过调用相应的模块，进行整个平台的管理，并生成反馈信息，在用户界面显示。

第二层数据流图：

用户信息管理子系统

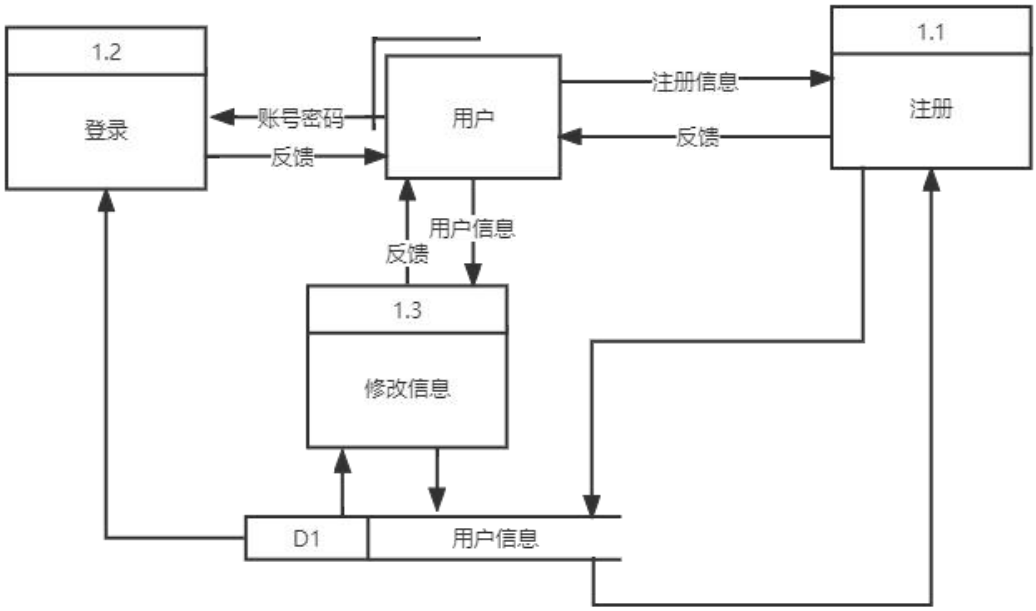

图 1.5 第二层数据流图——用户信息管理子系统

用户将账号密码输入系统，经过登录模块处理，与用户信息表中的信息相比对，最后将反馈信息返回给用户。

用户将注册输入系统，经过注册模块处理，与用户信息表中的信息相比对，最后将反馈信息返回给用户。

用户将修改后的用户信息输入到修改信息模块，该模块将信息与数据库中的信息相比对，修改数据库中的用户信息，并将反馈信息返回给用户。

歌曲管理子系统

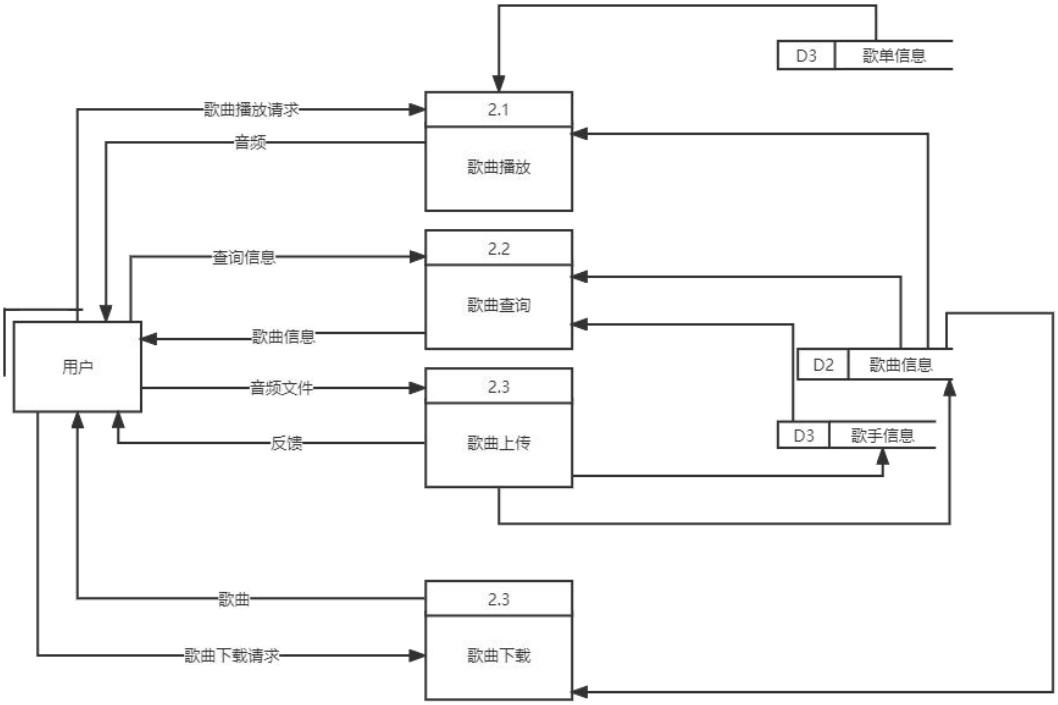

图 1.6 第二层数据流图——歌曲管理子系统

用户将歌曲播放请求输入到歌曲播放模块，该模块根据用户播放请求中的歌曲 ID 或歌单 ID，将信息与数据库中的信息相比对，提取出数据库中的歌曲信息，并将歌曲的音频返回给用户。

用户将歌曲查询信息输入到歌曲查询模块，该模块根据用户查询请求中的搜索关键词，在歌曲信息数据表中搜索，搜索出符合条件的歌曲信息，并将歌曲信息返回给用户。

用户将音频文件输入到歌曲上传模块，该模块根据用户上传的音频信息，将歌曲信息存储到数据库中，并将反馈信息返回给用户。

用户将歌曲下载请求信息输入到歌曲下载模块，该模块根据用户的歌曲下载信息，与数据库中的歌曲信息相比对，并将歌曲文件返回给用户。

歌单管理子系统

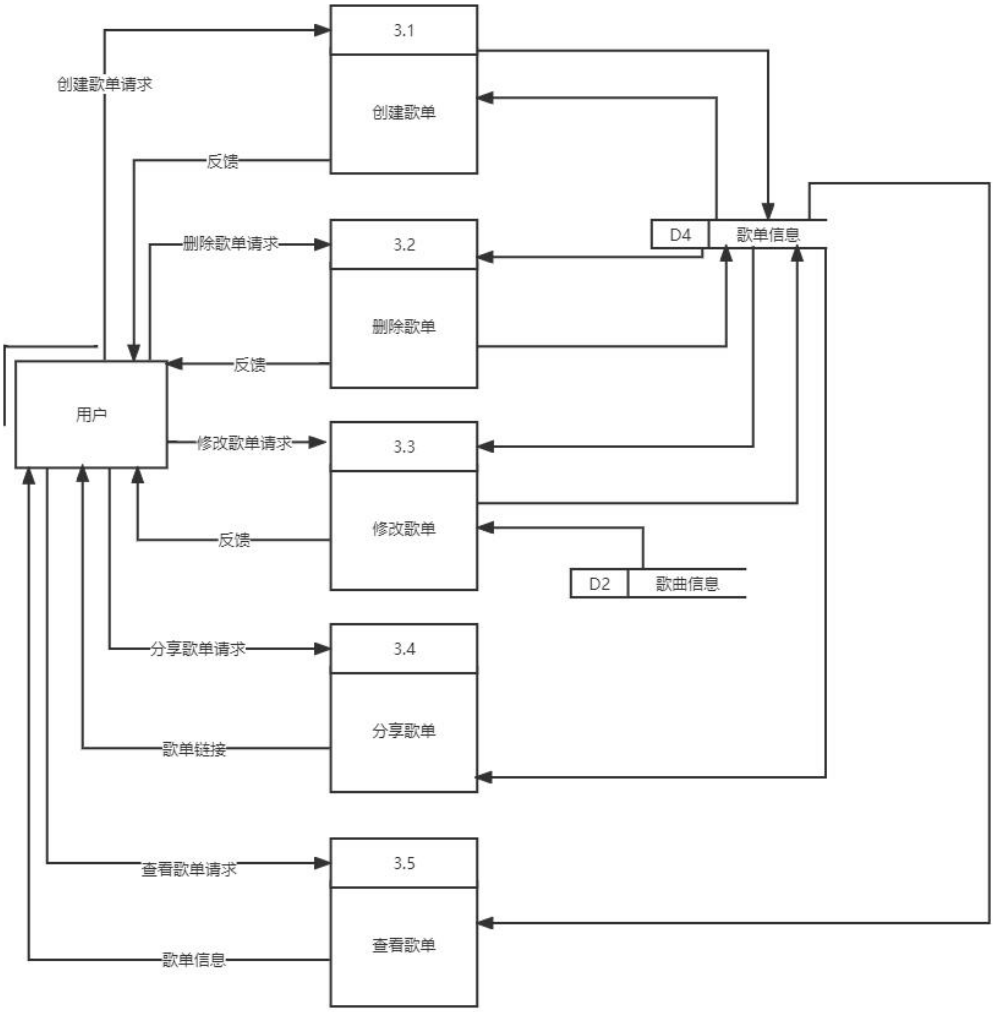

图 1.7 第二层数据流图——歌单管理子系统

用户将创建歌单、删除歌单请求输入到对应模块，对应模块调用代码，在歌单信息的数据库中实现各类操作，并将操作结果反馈返回给用户。

用户将修改歌单的请求输入到修改歌单模块，对应模块调用代码，根据请求中的修改信息，在歌单信息的数据库中执行修改操作，并将操作结果反馈返回给用户。

用户将分享歌单的请求输入到分享歌单模块，对应模块调用代码，根据请求中的歌单 ID，生成歌单链接，并返回给用户。

用户将查看歌单请求输入到查看歌单模块，模块访问并获得歌单及歌单中的歌曲信息，并将歌单信息返回给用户。

歌曲评论管理子系统：

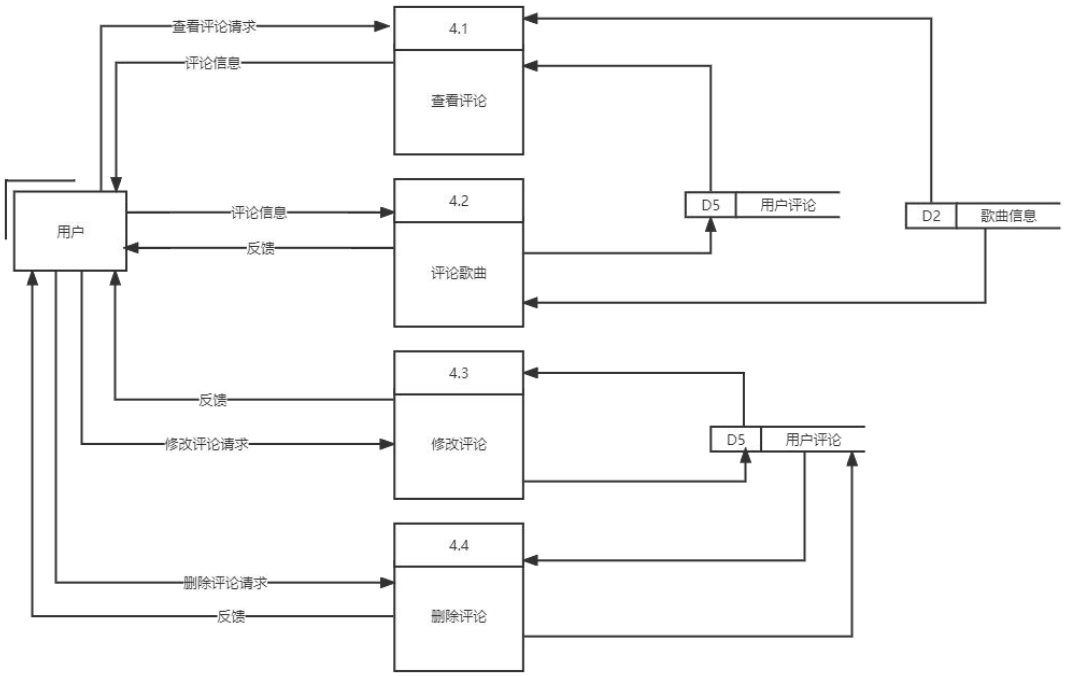

图 1.8 第二层数据流图——歌曲评论管理子系统

用户将查看评论请求输入到查看评论模块，模块访问用户评论数据库并获得该歌曲的用户评论信息，并将评论信息返回给用户。

用户将评论信息输入到评论歌曲模块，该模块根据歌曲信息和评论信息，将评论存入用户评论表中，并生成反馈信息，返回给用户。

用户将修改评论请求输入到修改评论模块，该模块根据修改评论请求中的评论 ID，修改用户评论表中的对应记录，并生成反馈信息，返回给用户。

用户将删除评论请求信息输入到删除评论模块，该模块根据评论 ID，将评论从用户评论表中删除，并生成反馈信息，返回给用户。

管理员子系统：

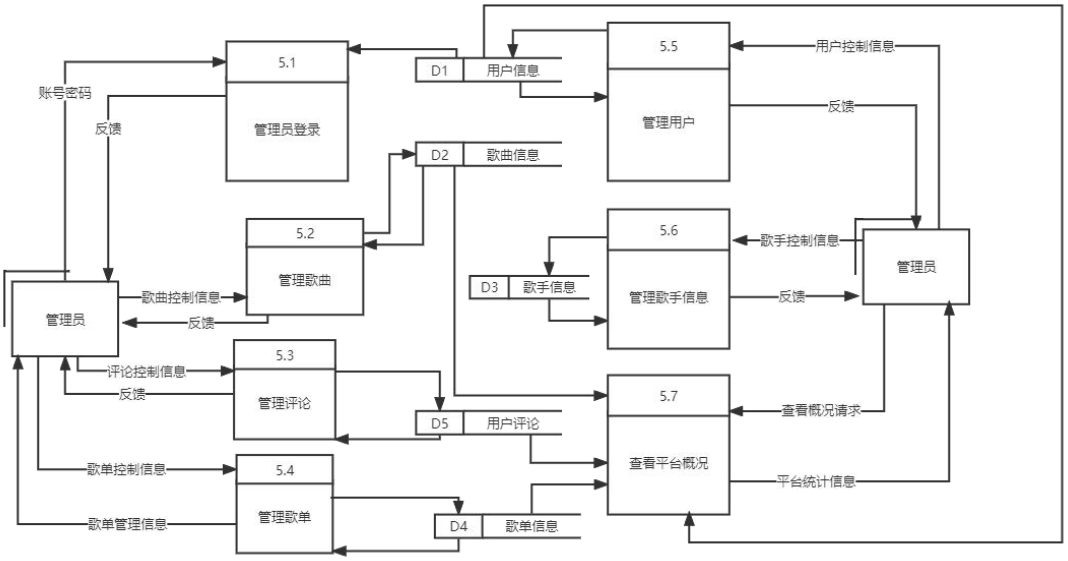

图 1.9 第二层数据流图——管理员子系统

管理员将账号密码输入到管理员登录模块，模块访问用户信息数据库并与表象进行比对，并将反馈信息返回给管理员。

管理员将歌曲控制信息输入到管理歌曲模块，模块根据歌曲控制信息操纵数据库中的歌曲信息并将反馈信息返回给管理员。

管理员将评论控制信息输入到管理评论模块，模块根据评论控制信息操纵数据库中的评论信息并将反馈信息返回给管理员。

管理员将歌单控制信息输入到管理歌单模块，模块根据歌单控制信息操纵数据库中的歌单信息并将反馈信息返回给管理员。

管理员将用户控制信息输入到管理用户模块，模块根据用户控制信息操纵数据库中的用户信息并将反馈信息返回给管理员。

管理员将歌手控制信息输入到管理歌手模块，模块根据歌手控制信息操纵数据库中的歌手信息并将反馈信息返回给管理员。

管理员将查看概况请求输入到查看平台概况模块，模块综合统计用户信息、歌曲信息、用户评论、歌单信息，输出平台统计信息，返回给管理员。

1.4 功能分析图

功能分析图是帮助我们表达设计思路的最有效方式，它是一个连贯的思考过程，是帮助我们放大设计中闪光点的最有利的工具，是系统组织内部各个有机构成要素相互作用的联系方式或形式，以求有效、合理地把组织部分组织起来，为实现共同目标而协同工作。由于组织结构在系统中的基础地位和关键作用，系统所有战略意义上的变革，都必须首先在组织结构上开始。

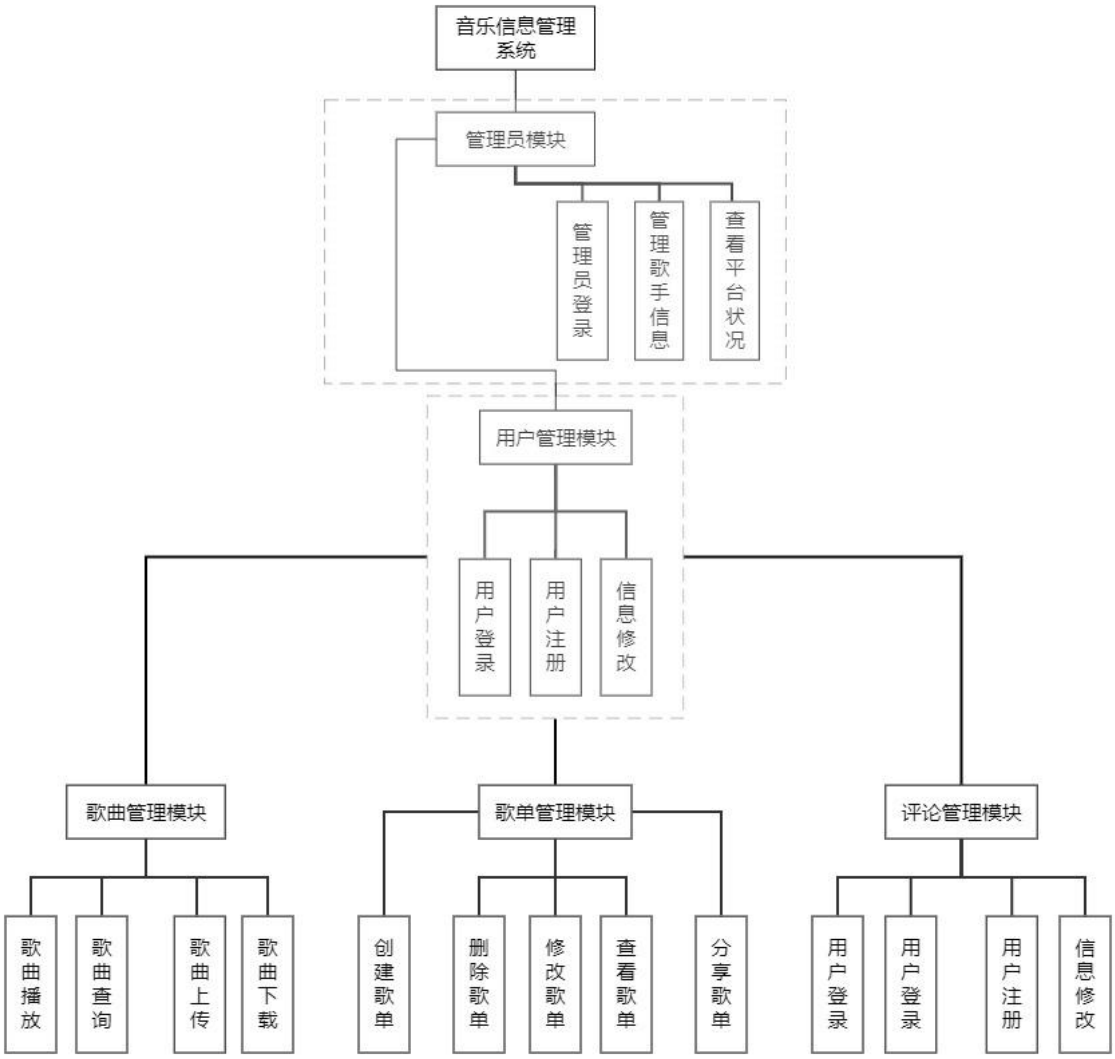

图 1.10 功能分析图

系统功能分析如上，这样的系统组织结构安排，一方面使用户体验感强，使用的功能更加明确，并且让管理员更好的管理用户和歌曲；另一方面精简了系统规模，避免了庞大的系统框架和组织结构基础，为整个系统的实际分析与设计打下基础。

第二章 系统设计

2.1 APP 总体设计

APP 的前端通过 activity 和 fragment 相结合来设计，我们为此设计了 个 Activity 和 个 Fragment。这是我们的整体结构图：

2.1.1 Fragment

Fragment 字面上的意思是碎片。如同它字面意思，Fragment 是 Activity 的碎片，它能将 Activity 分为多个模块，解决了每次给定的一个时间点在屏幕上只能显示单一的活动的问题，更具有灵活性。每个 Fragment 都有自己的布局，事件和完整的生命周期。

我们通过 Navigation 来实现 Fragment 之间的切换和导航，以实现底部菜单栏。

Navigation. 字面意思是导航,但是除了做 APP 引导页面以外.也可以使用在 App 主页分 tab 的情况.. 甚至可以一个功能模块就一个 activity 大部分页面 UI 都使用 fragment 来实现,而 navigation 就成了管理 fragment 至关重要的架构。

我们在 MainActivity 中分出 3 个 Fragment：，分别实现
XX 功能。并通过 Navigation 控制 Fragment 之间的数据交换。

于此同时，为了在 Fragment 之间共享数据，我们使用 ViewModel 实现 Fragment 之间的数据共享。因为 ViewModel 能够将数据从 Activity 中剥离出来。只要 Activity 不被销毁，ViewModel 会一直存在，并且独立于 Activity 的配置变化，即旋转屏幕导致的 Activity 重建，不会影响到 ViewModel。

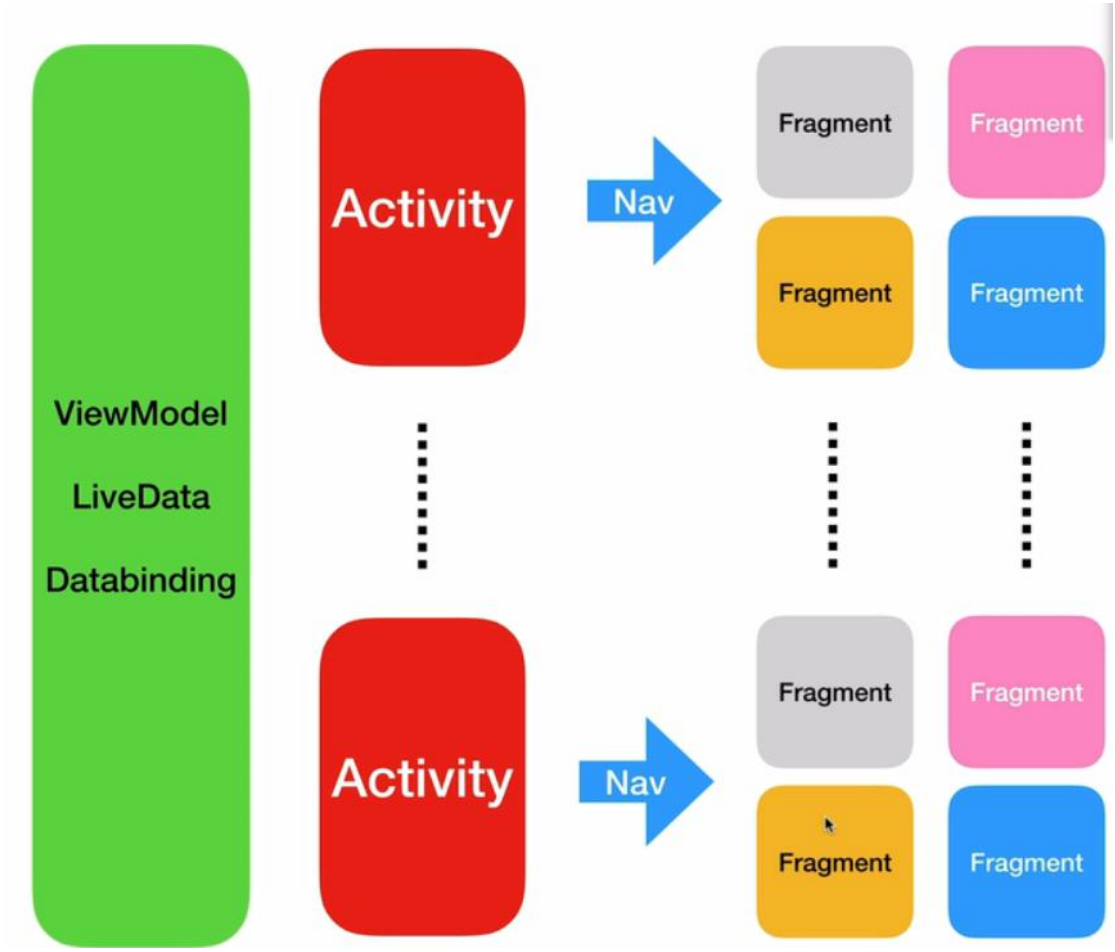

图 2.1　Activity 与 Fragment 关系图

playFragment

歌曲播放界面。实现歌曲播放功能，包括歌曲的播放暂停、切换下一首下一首、更改播放列表的播放顺序（单曲循环、列表循环、随机播放）、进度条等功能。实现本地歌曲搜索功能，自动扫描本地时长大于 30 秒音乐，将其加入到播放列表。点击播放列表中对应的音乐进行播放。用户选择要听的歌曲，客户端先通过歌曲 ID 判断歌曲是否在本地缓存中，若不在本地缓存中则发送请求给服务器，从服务器下载歌曲及歌曲信息，存储到本地，并播放。如果在本地缓存中则直接播放。

userFragment

个人信息界面。实现个人信息管理的功能，实现修改个人信息、切换用户、上传音乐等功能。用户需要修改个人信息时，在修改页面填写表单，输入修改后

的信息，然后封装成 JSON 发送给服务器，服务器将封装后的信息提取出来，修改数据库中的对应项，并返回修改成功消息给客户端，客户端输出修改成功消息和修改后的个人信息。

MainFragment

App 首页。实现歌曲搜索、轮播条、个人上传歌曲信息等功能。用户通过搜索框输入要搜索的歌曲，客户端将用户的搜索词发送给服务器，服务器通过 SQL 语句模糊查询，查看是否存在该歌曲，如果不存在则输出错误信息，存在就输出歌曲信息。

2.1.2 Activity

根据需要实现的功能，我们设计了　个 activity，它们之间的关系如下图所示：

LoginActivity、RegisterActivity

用户通过选择注册或登录按钮，选择进入注册或登录界面，在界面中输入账号密码和个人信息，提交之后，将信息封装成 JSON 格式，通过 http 发送给远端服务器。服务器判断请求是注册还是登录。

如果是注册，先使用 SELECT 语句在数据库中查找对应的用户名，如果用户名已注册则输出错误信息，返回给客户端；如果用户名未注册，则将用户信息存入数据库中，客户端输出注册成功消息。

如果是登录，先使用 SELECT 语句在数据库中查找对应的用户名，如果用户名未注册则输出未注册错误信息，返回给客户端；如果密码错误，客户端输出密码错误消息。

UploadActivity

用户进入上传界面，输入要上传的歌曲文件以及歌曲信息，点击上传，将信

息打包上传至服务器，同时在数据库中更新歌曲信息，并输出操作成功的信息。

DownloadActivity

用户选择一首歌曲发起下载请求，客户端将歌曲 ID 及下载请求发送给服务器，服务器接收到请求后，根据歌曲 ID 在数据库中搜索歌曲的存储路径，将歌曲发送给客户端，客户端接收到歌曲后将歌曲存储在本地。

SearchActivity

用户通过搜索框输入要搜索的歌曲，客户端将用户的搜索词发送给服务器，服务器通过 SQL 语句模糊查询，查看是否存在该歌曲，如果不存在则输出错误信息，存在就输出歌曲信息。

UserActivity

用户需要修改个人信息时，在修改页面填写表单，输入修改后的信息，然后封装成 JSON 发送给服务器，服务器将封装后的信息提取出来，修改数据库中的对应项，并返回修改成功消息给客户端，客户端输出修改成功消息和修改后的个人信息。

AdminActivity

管理员进入管理用户页面，客户端发送管理用户请求到服务器，服务器将用户信息发送到客户端。管理员通过选择用户，能对用户执行删除，修改等操作，客户端接收到操作请求后，将用户 ID 和修改后信息发送给服务器，服务器调用对应 SQL 语句处理之后将结果返回给客户端，客户端输出反馈信息。管理员通过点击查看平台概况按钮，客户端发送查看平台概况请求到服务器，服务器通过 SQL 语句统计出平台的各项数据并通过界面输出。

2.1.3 实体类定义

定义 4 个实体类：Music 类、MusicList 类、User 类、Single 类。

Music 类的属性包括歌曲 id、歌曲名、歌手名、类别、路径、上传用户 id、时长。MusicList 类的属性包括歌单 id、歌曲地址、创建者 id、图片 url。User 类的属性包括用户 id、用户名、密码、性别、年龄、手机号、歌曲兴趣。Single 类的属性包括歌手 id、歌手名。

2.2 SpringBoot 后端设计

我们使用安卓前端+ Spring Boot 框架后端来构造整个项目。前端负责展示用户界面及处理部分事件，以及向后端发送请求并接受后端传输的数据。后端负责提供接口来处理前端的请求，并将数据加工后返回给前端。

我们使用自己搭建的服务器作为后端，服务器系统为 Ubuntu 20.04.1。通过拨打运营商电话，向运营商申请 IPv4 地址，使我们的服务器能够通过公网访问，但需保证服务器的拨号连接不能中断，否则 IP 地址会发生改变。

我们在服务器上部署 Spring Boot 框架后台，并根据前端的不同请求设计不同接口，前端通过 HTTP 协议，将请求信息发送到服务器的对应接口，服务器提取请求信息，并通过加工，使用回调方法返回给前端，前端根据返回的数据进行界面的控制。

我们前后端之间的数据交换采用 JSON 格式。JSON（JavaScript Object Notation）是一种轻量级的数据交换格式。易于人阅读和编写，可以在多种语言之间进行数据交换。同时也易于机器解析和生成。

2.3 数据库设计

云端数据库使用 SQL Server，部署在云服务器上，通过 1443 端口号连接。通过数据库的概要设计，识别系统中的实体和实体的属性，分析实体与实体之间的联系，建立一个 E-R 模型来描述概念设计的结果。

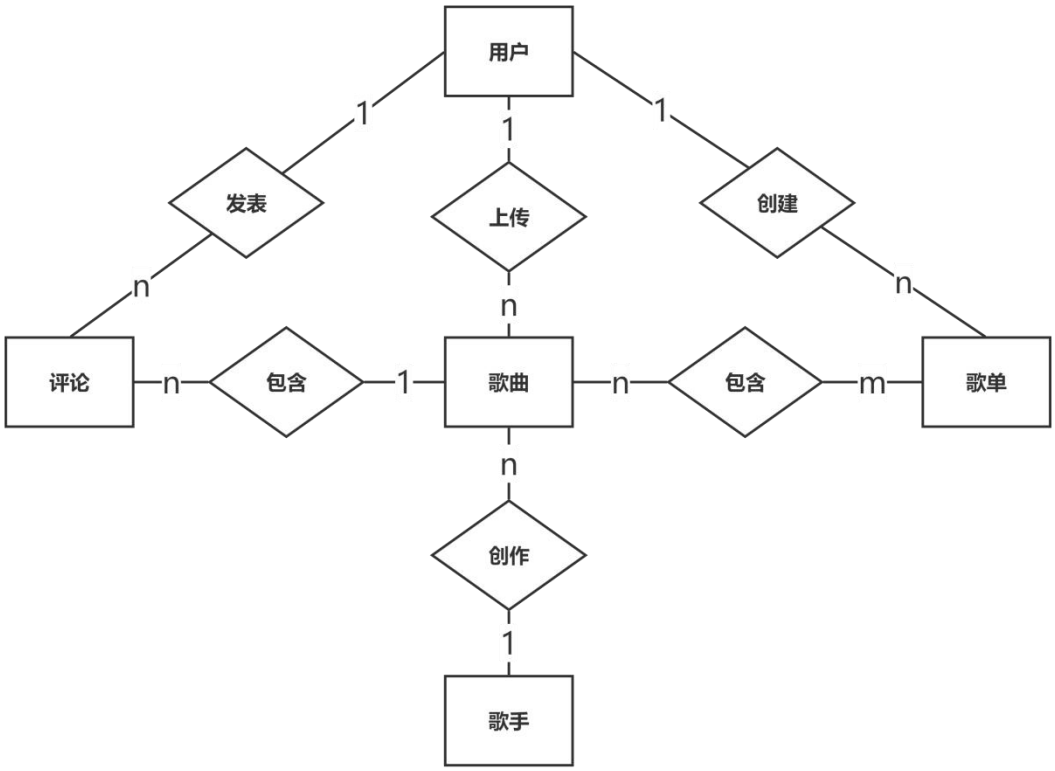

图 2.2　数据库 E-R 图

将概念模型转化为逻辑结构，并对其进行优化，使其满足数据库 3NF 的要求，最终的物理存储表如下：

表 2.1 物理存储表

	字段	数据类型	约束条件	复合主键	主键/外键	说明
用户表（user）	u_id	int	auto_increment		主键	用户 id
	u_username	varchar(15)				用户名
	u_password	varchar(20)				密码
	u_sex	varchar(20)	male or famale			性别
	u_region	varchar(20)				地区
	u_age	int				年龄
	u_phone	char(11)	不能重复,not null			手机号

	u_type	int	1 或 0（1 表示管理员，0 表示用户）			类型
歌曲表（music）	m_id	int	auto_increment		主键	歌曲 id
	m_name	varchar(30)				歌曲名
	m_url	varchar(50)				歌曲文件 url
	m_singer	int			外键	歌手 id
	m_type	int	1 为待审核，0 为正常			歌曲状态
	m_userid	int			外键	用户 id
歌单表（musiclist）	ml_id	int	auto_increment		主键	歌单 id
	ml_name	varchar(20)				歌单名
	ml_userid	int			外键	用户 id
歌曲歌单关系表（music_musiclist）	mml_musicid	int		复合主键	外键	歌曲 id
	mml_listid	int			外键	歌单 id
评论表(comment)	c_id	int	auto_increment		主键	评论 id
	c_content	varchar(200)				评论内容
	c_userid	int			外键	用户 id
歌曲评论关系表（music_comment）	mc_musicid	int		复合主键	外键	歌曲 id
	mc_commentid	int			外键	评论 id
歌手表（singer）	s_id	int	auto_increment		主键	歌手 id
	s_name	varchar(				歌手名

		30)				
	s_region	varchar(30)				歌手地区
	s_intro	varchar(200)				歌手介绍

生成的数据库关系图如下：

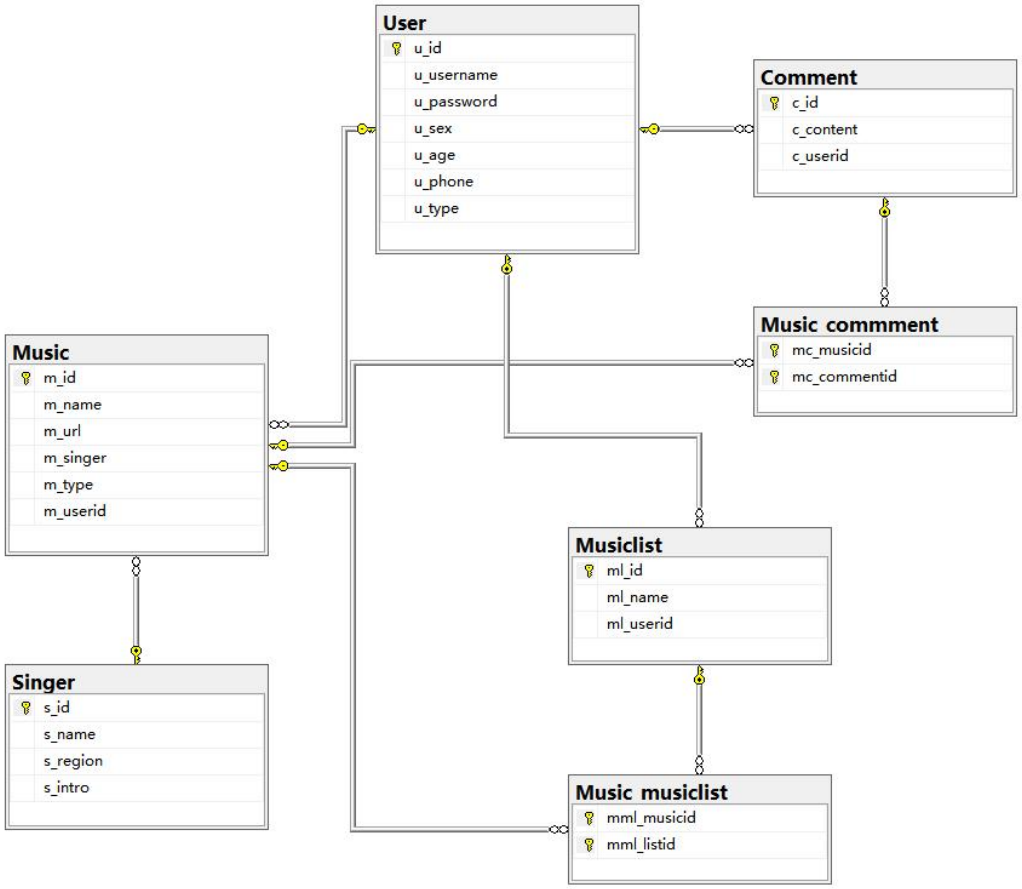

图 2.3　数据库关系图

2.4 UI 设计

图 2.4　注册登录界面

图 2.5　管理员界面

图 2.6　修改信息界面

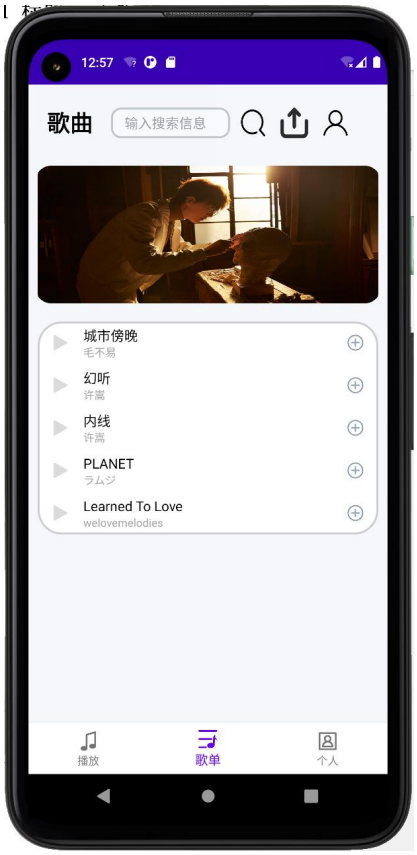

图 2.7　主界面

图 2.8　上传界面

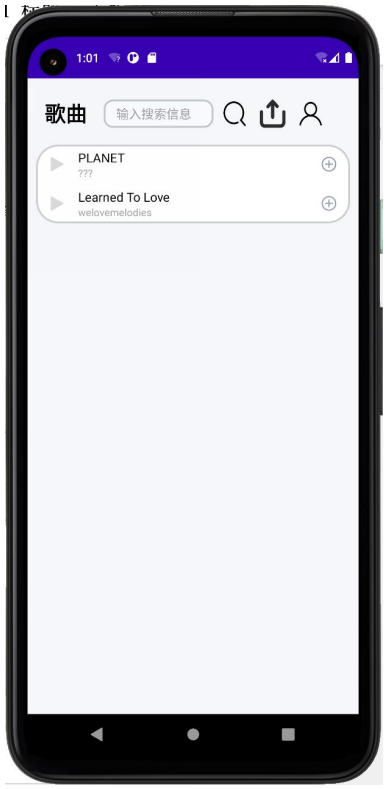

图 2.9 搜索界面

图 2.10 播放界面

2.5 GIT 版本控制

我们使用 Git 来进行版本的控制和同步，安装好 Git 后，在 Android Studio 中配置好路径，即可直接在 Android Studio 中进行 push、pull 等操作。使用 Git 大大提高了我们的开发效率。我们项目的 github 网址为：https://github.com/yhz1651/MusicApp

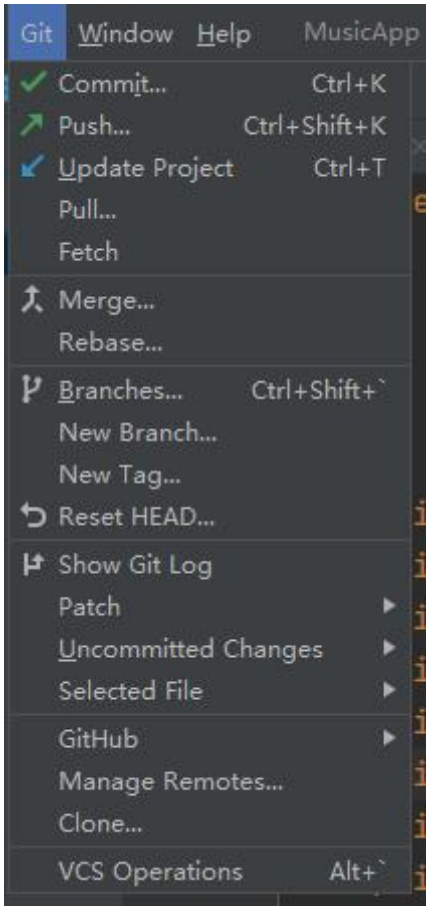

图 2. 11 Git 功能图

第三章 系统实现

3.1 登录注册功能

通过 LoginActivity 和 RegisterActivity 来实现登录注册功能。用户通过选择注册或登录的 Button，选择进入注册或登录界面，在 EditTextView 中输入账号密码和个人信息，提交之后，将信息封装成 JSON 格式，通过 http 发送给远端服务器。服务器判断请求是注册还是登录。

如果是注册，先使用 SELECT 语句在数据库中查找对应的用户名，如果用户名已注册则输出错误信息，返回给客户端；如果用户名未注册，则将用户信息存入数据库中，客户端输出注册成功消息。

如果是登录，先使用 SELECT 语句在数据库中查找对应的用户名，如果用户名未注册则输出未注册错误信息，返回给客户端；如果密码错误，客户端输出密码错误消息。

LoginActivity 的主要代码

```
        UserService uService=new UserService(LoginActivity.this);
    String flag=uService.login(name, pass);

    if(!flag.equals("false")){//如果不是返回 false 即登录成功
        Log.i("TAG","登录成功");
        UserApplication application1 = (UserApplication)
LoginActivity.getInstance().getApplication();//将用户名存入 Application
        application1.setValue(flag);
        Toast.makeText(LoginActivity.this, "成功登录",
Toast.LENGTH_SHORT).show();
        Intent intent;
        if(flag.equals("U00000000001"))
        {intent = new Intent(LoginActivity.this, AdminActivity.class);
        startActivity(intent);}//管理员账号进入管理页面
        else{
        intent = new Intent(LoginActivity.this,MainActivity.class);
        startActivity(intent);}
    }else{
        Log.i("TAG","登录失败");
```

```
            Toast.makeText(LoginActivity.this, "登录失败",
Toast.LENGTH_SHORT).show();
        }
}
```

RegisterActivity 的主要代码

```
    public void on2click(View view) {
            Button button = findViewById(R.id.xingquButton);
            if(button != null){
                AlertDialog.Builder aBuilder = new
AlertDialog.Builder(RegisterActivity.this);
                aBuilder.setTitle("歌曲兴趣（可多选）");

                // 创建多选对话框的内容
                aBuilder.setMultiChoiceItems(strList, blList, new
DialogInterface.OnMultiChoiceClickListener() {
                    // 取消或勾选对话框内的单选框
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i, boolean
b) {
                        blList[i] = b;
                    }
                });

                // 加入多选对话框下面的确认按钮,打印勾选的项名称
                aBuilder.setPositiveButton("确认", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {
                        // 获取勾选到的文本并打印
//                          String txt = "";
                        for(short index=0;index<blList.length;++index){
                            if(blList[index]){
                                txt += (strList[index]+" ");
                            }
```

```
                        }
                        EditText et =
(EditText)findViewById(R.id.xingquRegister);//获取 edittext 组件
                        et.setText(txt);
                        // 打印
                        Toast.makeText(RegisterActivity.this, txt,
Toast.LENGTH_SHORT).show();

                        // 取消内部已勾选
                        cancelBool();
                    }
                });

                // 点击对话框外,进行取消对话框事件
                aBuilder.setOnCancelListener(new DialogInterface.OnCancelListener()
{
                    @Override
                    public void onCancel(DialogInterface dialogInterface) {
                        // 取消内部已勾选
                        cancelBool();
                    }
                });

                // 显示对话框
                aBuilder.show();
            }
        }
        //
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_register);
            findViews();
            register.setOnClickListener(new View.OnClickListener() {
                public void onClick(View v) {
```

```
                    String name=username.getText().toString().trim();
                    String pass=password.getText().toString().trim();
                    String
sexstr=((RadioButton)RegisterActivity.this.findViewById(sex.getCheckedRadioButto
nId())).getText().toString();
                    int ag= Integer.parseInt(age.getText().toString().trim());
                    String ph=phone.getText().toString().trim();
                    String xingqustr=txt.trim();
                    Log.i("TAG",name+"_"+pass+"_"+xingqustr+"_"+sexstr);
                    UserService uService=new UserService(RegisterActivity.this);
                    User user=new User();
                    user.setUsername(name);
                    user.setPassword(pass);
                    user.setSex(sexstr);
                    user.setAge(ag);
                    user.setPhone(ph);
                    user.setXingqu(xingqustr);
                    uService.register(user);
                    Toast.makeText(RegisterActivity.this, "注册成功",
Toast.LENGTH_LONG).show();
                    Intent intent = new
Intent(RegisterActivity.this,LoginActivity.class);
                    startActivity(intent);
                }
            });
        back.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                Intent intent=new Intent(RegisterActivity.this,LoginActivity.class);
                startActivity(intent);
            }
        });
    }
```

后台服务器代码

开放 request 接口，接收前端的 SQL 请求，处理数据库，返回值。

```
    @ResponseBody
@RequestMapping("/register")
public String rejister(@RequestParam(value = "u_username")String u_sername
            ,@RequestParam(value = "u_password")String password
            ,@RequestParam(value = "u_sex")String u_sex
            ,@RequestParam(value = "u_age")String u_age
            ,@RequestParam(value = "u_phone")String u_phone
            ,@RequestParam(value = "u_hobby")String u_hobby) throws
SQLException {
    DATA_BASE db = new DATA_BASE();
    String st= db.select("SELECT *" + " FROM UserList "+ "WHERE
u_username='"+u_sername+"'for json auto");//是否注册
    if (st!=null){
        System.out.println("false");
        return "false";//没有注册
    }
    String num = db.select("select TOP 1 u_id FROM UserList ORDER BY u_id
DESC");
    DecimalFormat decimalFormat = new DecimalFormat("00000000000");
    String m2= decimalFormat .format(Integer.parseInt(num.substring(1))+1);
    String new_id =    "U"+m2;
    String insert_sql="INSERT INTO UserList
VALUES('"+new_id+"','"+u_sername+"','"+password+"','"+u_sex+"','"+u_age+"
','"+u_phone+"',0,'"+u_hobby+"');";
    db.exec(insert_sql);
    System.out.println(new_id);
    return new_id;
}
@ResponseBody
@RequestMapping("/login")
public String login(@RequestParam(value = "u_username")String u_sername
            ,@RequestParam(value = "u_password")String password
        ) throws SQLException {
    DATA_BASE db = new DATA_BASE();
```

```
        String st= db.select("SELECT u_id" + " FROM UserList "+ "WHERE
u_username='"+u_sername+"' AND u_password ='"+password+"'");//是否登录
        if (st==null){
            System.out.println("false");
            return "false";//登录失败
        }
        System.out.println(st);
        return st;
}
```

3.2 歌曲播放功能

通过 playFragment 来实现登录注册功能。首先利用 MediaScanner 类扫描本地的音乐文件，筛选出时长大于 30 秒的加入播放列表 MusicList。使用 mediaplayer 的 play 和 pause 函数实现音乐的播放暂停。播放列表具有三种播放模式，单曲循环、列表循环、随机播放，通过设置一个标志位，每次对 3 取余，0 为列表循环，1 为单曲循环，2 为随机播放。上一首下一首切换时，首先判断是三种播放模式中的哪一种，根据不用的模式使用不同的策略。使用 seekbar 实现进度条功能，使用 getDuration 和 getCurrentPosition 方法实现当前时间和完整时长。播放列表使用 recyclerView 实现，重写 adapater 类的 setOnItemClickListener 的方法，实现点击播放列表中的音乐进行播放。

playFragment 的主要代码

```
        //切换歌曲播放模式
btn_music_mode.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            playmode=(++playmode) % 3;
            if(playmode==0){// 0 为列表循环

btn_music_mode.setBackgroundResource(R.drawable.music_mode_list);
            }else if(playmode==1){// 1 为单曲循环
```

```
btn_music_mode.setBackgroundResource(R.drawable.music_mode_single);
            }else{// 2 为随机播放

btn_music_mode.setBackgroundResource(R.drawable.music_mode_random);
            }
        }
});

//暂停，播放
btn_music_play.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if(mediaplayer==null) return;
            if(isplay==true){
                isplay=false;
                mediaplayer.pause();
                btn_music_play.setBackgroundResource(R.drawable.music_pause);
            }else{
                isplay=true;
                mediaplayer.start();
                btn_music_play.setBackgroundResource(R.drawable.music_play);
            }
        }
});

//上一首下一首
btn_music_pre.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if(mediaplayer==null) return;
            if(playmode==0){// 0 为列表循环
                music_id=(--music_id) % musicList.size();
                initMediaPlayer(musicList.get(music_id));
            }else if(playmode==1){// 1 为单曲循环
```

```
            }else{// 2 为随机播放
                music_id = new Random().nextInt(musicList.size());
                initMediaPlayer(musicList.get(music_id));
            }
        }
});
btn_music_next.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if(mediaplayer==null) return;
            if(playmode==0){// 0 为列表循环
                music_id=(++music_id) % musicList.size();
                initMediaPlayer(musicList.get(music_id));
            }else if(playmode==1){// 1 为单曲循环

            }else{// 2 为随机播放
                music_id = new Random().nextInt(musicList.size());
                initMediaPlayer(musicList.get(music_id));
            }
        }
});


//进度条 开始时间 结束时间
seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
        @Override
        public void onProgressChanged(SeekBar seekBar, int progress, boolean
fromUser) {
        }

        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {
            seekbarchange = true;
```

```
        }

        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {
            seekbarchange = false;
            mediaplayer.seekTo(seekBar.getProgress());
            int m = mediaplayer.getCurrentPosition() / 60000;
            int s = (mediaplayer.getCurrentPosition() - m * 60000) / 1000;
            start_time.setText(m+":"+s);
        }
});

//初始化播放列表
recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
MusicAdapter adapter=new MusicAdapter( musicList,getContext());
recyclerView.setAdapter(adapter);
adapter.setOnItemClickListener(new MusicAdapter.OnItemClickListener() {
        @Override
        public void onItemClick(View view, int position) {
            Music music = musicList.get(position);
            initMediaPlayer(music);
        }

        @Override
        public void onItemLongClick(View view, int position) {

        }
});
        MediaScannerConnection.scanFile(getContext(), new
String[]{Environment.getExternalStorageDirectory().getAbsolutePath()}, null, null);
            Cursor cursor = getContext().getContentResolver().query(
                    MediaStore.Audio.Media.EXTERNAL_CONTENT_URI,
                    new String[]{MediaStore.Audio.Media._ID,
                            MediaStore.Audio.Media.DISPLAY_NAME,
```

```
                                MediaStore.Audio.Media.TITLE,
                                MediaStore.Audio.Media.DURATION,
                                MediaStore.Audio.Media.ARTIST,
                                MediaStore.Audio.Media.ALBUM,
                                MediaStore.Audio.Media.SIZE,
                                MediaStore.Audio.Media.DATA},
                        MediaStore.Audio.Media.MIME_TYPE + "=?",
                        new String[]{"audio/mpeg"}, null);
            String fileName, title, singer,size, filePath = "";
            int duration, m, s;
            Music song;

            if (cursor.moveToFirst()) {
                do{
                    fileName = cursor.getString(1);
                    title = cursor.getString(2);
                    duration = cursor.getInt(3);
                    singer = cursor.getString(4);
                    size = (cursor.getString(6) == null) ? "未知" : cursor.getInt(6) /
1024 / 1024 + "MB";
                    if (cursor.getString(7) != null) filePath = cursor.getString(7);

                    song = new Music(null,title,singer,filePath,null,duration);
                    musicList.add(song);
                    //大于 30 秒的
//                    if (duration > 30000) {
//                        musicList.add(song);
//                    }
                }while (cursor.moveToNext());
                // 释放资源
                cursor.close();
            }
            int len=musicList.size();
```

```
            Toast.makeText(getContext(),"歌曲数量为
"+len,Toast.LENGTH_LONG).show();//  扫描到的歌曲数量
```

3.3 歌曲上传下载功能

通过 UploadActivity 和 DownloadActivity 来实现歌曲上传下载功能。用户进入上传界面，输入要上传的歌曲文件以及歌曲信息，点击上传，将信息打包上传至服务器，同时在数据库中更新歌曲信息，并输出操作成功的信息。用户选择一首歌曲发起下载请求，客户端将歌曲 ID 及下载请求发送给服务器，服务器接收到请求后，根据歌曲 ID 在数据库中搜索歌曲的存储路径，将歌曲发送给客户端，客户端接收到歌曲后将歌曲存储在本地。

UploadActivity 的主要代码

```
    Submit.setOnClickListener(new View.OnClickListener() {
                public void onClick(View v) {
                    String song_n= songname.getText().toString().trim();//获取歌曲
名字
                    String singer_n= singername.getText().toString().trim();//获取歌
手名字
                    String url = url_text.getText().toString().trim();//获取歌曲 url
                    DatabaseHelper dbsqLiteOpenHelper = new
DatabaseHelper(UploadActivity.this);//创建本地数据库操控接口
                    SQLiteDatabase sdb =
dbsqLiteOpenHelper.getWritableDatabase();
                    String sql="SELECT s_id FROM Singer WHERE s_name=?";//
根据歌手名字查询歌手 ID
                    Cursor cursor=sdb.rawQuery(sql, new String[]{singer_n});
                    UserApplication application1 = (UserApplication)
UploadActivity.this.getApplication();//获取 application
                    String id = application1.getValue();//获得用户 ID
//                    if(cursor.moveToFirst()){
//                        s_id = cursor.getInt(0);
//
Toast.makeText(LoginActivity.getInstance(),cursor.getString(0),
Toast.LENGTH_LONG).show();
```

```
//                          cursor.close();
//                      }else{
//                          s_id = 1;
//                      }
                    s = new Music(null,song_n,singer_n,url,id,0);//存入歌曲对象中
                    sql="insert into Music(m_name,m_url,m_singer,m_type,m_userid)
values(?,?,?,0,?);";
                    Object obj[]={s.getName(),s.getUrl(),s.getSinger(),s.getUser()};
                    sdb.execSQL(sql, obj);
                    upload();
                    Toast.makeText(UploadActivity.this, "上传成功",
Toast.LENGTH_LONG).show();
                    Intent intent = new
Intent(UploadActivity.this,MainActivity.class);//上传成功后，回到主页面
                    startActivity(intent);
                }
            });
    /**
     * 回调方法，获取歌曲的uri
     */
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if(requestCode==1){
        if(resultCode==RESULT_OK){
            Uri uri = data.getData();
            //Log.e("图片 URI： ", uri.toString());
            UriTool ut=new UriTool();
            url_text.setText(ut.UriToPath(this, uri));
        }
    }
}
private void upload(){
    new Thread(new Runnable() {//使用多线程，防止主线程堵塞
```

```
        @Override
        public void run() {
            OkHttpClient okHttpClient = new OkHttpClient();
            Uri uri = Uri.parse(s.getUrl());
            File file = new File(String.valueOf(uri));
            String filename = s.getUrl().substring(s.getUrl().lastIndexOf("/") + 1);

            RequestBody fileBody =
RequestBody.create(MediaType.parse("application/octet-stream"), file);
            RequestBody requestBody = new MultipartBody.Builder()//创建
requestbody
                    .setType(MultipartBody.FORM)
                    .addPart(Headers.of(
                            "Content-Disposition",
                            "form-data; name=\"getUpTime\""),
                            RequestBody.create(null, "2022-6-29"))
                    .addPart(Headers.of(
                            "Content-Disposition",
                            "form-data; name=\"originalData\"; filename=\"" +
filename + "\""), fileBody)
                    .build();
            String url = "http://192.168.1.104:7506/solution";
            System.out.println("------------------------------------------------");
            Request request = new Request.Builder()
                    .url(url)
                    .post(requestBody)
                    .build();
            Call call = okHttpClient.newCall(request);
            call.enqueue(new Callback() {//回调方法
                @Override
                public void onFailure(Call call, IOException e) {
                    Log.e("text", "failure upload!" + e.getMessage());
                }
```

```
                    @Override
                    public void onResponse(Call call, Response response) throws
IOException {
                        Log.i("text", "success upload!");
                        String json = response.body().string();
                        Log.i("success.....", "成功" + json);
                    }
                });
            }
        }).start();
}
```

DownloadTool 的主要代码

通过接收后端返回的 url 和文件名，下载文件并存储在本地音乐文件夹中。

```
public class DownloadTool {
    int count = 0;
    long total = 0;
    int progress = 0;
    public static String url="http://192.168.1.110:7506";
    public void download(String site,String filename) throws IOException {
        URL url = new URL(site);
        URLConnection con = url.openConnection();
        con.connect();
        int fie_length = con.getContentLength();
        InputStream is = url.openStream();
        File path = new File("/storage/emulated/0/Music");
        FileOutputStream fos = new FileOutputStream(path+"/"+filename);
        byte data[] = new byte[1024];
        while ((count=is.read(data)) != -1){
            total += count;
            int progress_temp = (int)total*100/fie_length;
            if(progress_temp%10 == 0 && progress != progress_temp){
                progress = progress_temp;
```

```
                }
                fos.write(data, 0, count);

            }
            is.close();
            fos.close();
        }
}
```

3.4 歌曲搜索功能

通过 SearchActivity 来实现歌曲搜索功能。用户通过搜索框输入要搜索的歌曲，客户端将用户的搜索词发送给服务器，服务器通过 SQL 语句模糊查询，查看是否存在该歌曲，如果不存在则输出错误信息，存在就输出歌曲信息。

SearchActivity 的主要代码

```
public class SearchActivity extends AppCompatActivity {
    String ans;
    String key;
    private List<Music> musicList =new ArrayList<Music>();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_search);
        Intent intent=getIntent();
        key =intent.getStringExtra("key");
        ask();
        RecyclerView
recyclerView=(RecyclerView)findViewById(R.id.recyclerView);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        recyclerView.setAdapter(new MusicAdapter(musicList,this) );
        ImageButton search_btn = (ImageButton)findViewById(R.id.search);
        search_btn.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                TextView tex = (TextView)findViewById(R.id.s_key);
                Intent intent=new Intent(SearchActivity.this,
```

```
SearchActivity.class);
                    intent.putExtra("key",tex.getText().toString());
                    startActivity(intent);
                }
            });
            ImageButton up_btn = (ImageButton)findViewById(R.id.upload) ;
            up_btn.setOnClickListener(new View.OnClickListener() {
                public void onClick(View v) {
                    Intent intent=new Intent(SearchActivity.this,
UploadActivity.class);
                    startActivity(intent);
                }
            });
        }

        public void    ask(){
            SearchActivity.askMusic callable = new SearchActivity.askMusic();
            //将实现 Callable 接口的对象作为参数创建一个 FutureTask 对象
            FutureTask<String> task = new FutureTask<>(callable);
            //创建线程处理当前 callable 任务
            Thread thread = new Thread(task);
            //开启线程
            thread.start();
            //获取到 call 方法的返回值
            try {
                String result = task.get();
                ans=result;
                Log.e("text", "result" +ans );
                thread.join();
            } catch (Exception e) {
                e.printStackTrace();
            }
            jsonToList(ans);
        }
        class askMusic implements Callable<String>
        {     String ans;
```

```
        @Override
        public String call() throws Exception {//创建带回调方法的线程进行网络请求
            OkHttpClient okHttpClient = new OkHttpClient();
            RequestBody requestBody = new MultipartBody.Builder()//创建requestbody
                    .setType(MultipartBody.FORM)//传参
                    .addPart(Headers.of(
                            "Content-Disposition",
                            "form-data; name=\"m_name\""),
                            RequestBody.create(null, key))
                    .build();
            String url = DownloadTool.url +"/searchMusic";
            System.out.println("------------------------------------------------");
            Request request = new Request.Builder()
                    .url(url)
                    .post(requestBody)
                    .build();
            Call call = okHttpClient.newCall(request);
            call.enqueue(new Callback() {//回调方法
                @Override
                public void onFailure(Call call, IOException e) {
                    Log.e("text", "failure upload!" + e.getMessage());
                }


                @Override
                public void onResponse(Call call, Response response) throws IOException {
                    Log.i("text", "success upload!");
                    ans = response.body().string();
                }
            });
            while(ans==null){}//等待赋值
            return ans;
        }
```

```
    }
    public    void jsonToList(String json) {
        Gson gson = new Gson();
        musicList = gson.fromJson(json, new TypeToken<List<Music>>()
{}.getType());//对于不是类的情况，用这个参数给出
        for (Music music1 : musicList) {
            System.out.println(music1.getM_name()+" "+music1.getM_singer());
        }
    }
}
```

后端代码

```
    @ResponseBody
@RequestMapping("/searchMusic")
public String search(@RequestParam(value = "m_name")String m_name) throws
SQLException {
    DATA_BASE db = new DATA_BASE();
    String st= db.select("SELECT * FROM(SELECT
Music.m_id,Music.m_name,s_name as
m_singer,Music.m_url,Music.m_userid,Music.m_duration FROM
Music,Singer WHERE Music.m_singer=Singer.s_id AND Music.m_name like
'%"+m_name+"%' ) AS A for json auto");
    return st;
}
```

3.5 用户信息修改功能

通过 UserActivity 来实现用户信息修改功能。通过点击信息修改，将用户 ID 通过 Intent 传递给 ChangeUserInfoActivity，并向服务器发送信息请求，服务器将数据库中的信息发送给前端，并显示在界面中，通过界面，修改信息，并将修改后的信息发回给后端，操作数据库。

ChangeUserInfoActivity 的主要代码

```
Intent intent =    getIntent();
searchid = intent.getStringExtra("id");
```

```
        ask_user();
        findViews();
        username.setText(person.getU_username());
        password.setText(person.getU_password());
        sex.setText(person.getU_sex());
        age.setText(person.getU_age()+"");
        phone.setText(person.getU_phone());
        xingquregister.setText(person.getU_hobby());
        change_info.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                String name=username.getText().toString().trim();
                String pass=password.getText().toString().trim();
                int ag= Integer.parseInt(age.getText().toString().trim());
                String ph=phone.getText().toString().trim();
                String xingqustr=xingquregister.getText().toString().trim();
                Log.i("TAG",name+"_"+pass+"_"+xingqustr+"_");
                UserService uService=new UserService(ChangeUserInfoActivity.this);
                User user=new User();
                user.setU_username(name);
                user.setU_password(pass);
                user.setU_age(ag);
                user.setU_phone(ph);
                user.setU_hobby(xingqustr);
                register(user);
                Toast.makeText(ChangeUserInfoActivity.this,"修改成功
",Toast.LENGTH_SHORT).show();
                finish();
            }
        });
}


public void    ask_user(){
        ChangeUserInfoActivity.askUser callable = new
ChangeUserInfoActivity.askUser();
        //将实现 Callable 接口的对象作为参数创建一个 FutureTask 对象
```

```
        FutureTask<String> task = new FutureTask<>(callable);
        //创建线程处理当前 callable 任务
        Thread thread = new Thread(task);
        //开启线程
        thread.start();
        //获取到 call 方法的返回值
        try {
            String result = task.get();
            ans=result;
            Log.e("text", "result" +ans );
            thread.join();
        } catch (Exception e) {
            e.printStackTrace();
        }
        System.out.println("answer");
        Gson gson = new Gson();
        person = gson.fromJson(ans, User.class);
        System.out.println(person);
}
class askUser implements Callable<String>
{       String ans;
        @Override
        public String call() throws Exception {//创建带回调方法的线程进行网络请求
            OkHttpClient okHttpClient = new OkHttpClient();
            RequestBody requestBody = new MultipartBody.Builder()//创建
requestbody
                    .setType(MultipartBody.FORM)//传参
                    .addPart(Headers.of(
                            "Content-Disposition",
                            "form-data; name=\"u_id\""),
                            RequestBody.create(null, searchid))
                    .build();
            String url = DownloadTool.url +"/askUser";
            System.out.println("-----------------------------------------------");
            Request request = new Request.Builder()
                    .url(url)
```

```
                    .post(requestBody)
                    .build();
            Call call = okHttpClient.newCall(request);
            call.enqueue(new Callback() {//回调方法
                @Override
                public void onFailure(Call call, IOException e) {
                    Log.e("text", "failure upload!" + e.getMessage());
                }


                @Override
                public void onResponse(Call call, Response response) throws
IOException {
                    Log.i("text", "success upload!");
                    ans = response.body().string();
                }
            });
            while(ans==null){}
            return ans;
        }
}

private void findViews() {
        username = (EditText) findViewById(R.id.usernameRegister);
        password = (EditText) findViewById(R.id.passwordRegister);
        age = (EditText) findViewById(R.id.ageRegister);
        sex = (TextView)findViewById(R.id.sex);
        phone = (EditText) findViewById(R.id.phoneRegister);
        xingquregister = (EditText) findViewById(R.id.xingquRegister);
        change_info = (Button) findViewById(R.id.change_info);
}
public String    register(User user){
        this.user=user;
        ChangeUserInfoActivity.rejicall callable = new
ChangeUserInfoActivity.rejicall();
        //将实现 Callable 接口的对象作为参数创建一个 FutureTask 对象
```

```
        FutureTask<String> task = new FutureTask<>(callable);
        //创建线程处理当前 callable 任务
        Thread thread = new Thread(task);
        //开启线程
        thread.start();
        //获取到 call 方法的返回值
        try {
            String result = task.get();
            ans=result;
            Log.e("text", "result" +ans );
            thread.join();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return ans;
}
class rejicall implements Callable<String>
{
    @Override
    public String call() throws Exception {//创建带回调方法的线程进行网络请求
        OkHttpClient okHttpClient = new OkHttpClient();
        RequestBody requestBody = new MultipartBody.Builder()//创建
requestbody
                .setType(MultipartBody.FORM)//传参
                .addPart(Headers.of(
                        "Content-Disposition",
                        "form-data; name=\"u_username\""),
                        RequestBody.create(null, user.getU_username()))
                .addPart(Headers.of(
                        "Content-Disposition",
                        "form-data; name=\"u_password\""),
                        RequestBody.create(null, user.getU_password()))
                .addPart(Headers.of(
                        "Content-Disposition",
                        "form-data; name=\"u_id\""),
                        RequestBody.create(null, searchid))
```

```
                .addPart(Headers.of(
                        "Content-Disposition",
                        "form-data; name=\"u_age\""),
                        RequestBody.create(null, user.getU_age()+""))
                .addPart(Headers.of(
                        "Content-Disposition",
                        "form-data; name=\"u_phone\""),
                        RequestBody.create(null, user.getU_phone()+""))
                .addPart(Headers.of(
                        "Content-Disposition",
                        "form-data; name=\"u_hobby\""),
                        RequestBody.create(null, user.getU_hobby()+""))
                .build();
        String url = DownloadTool.url +"/changeUser";
        System.out.println("------------------------------------------------");
        Request request = new Request.Builder()
                .url(url)
                .post(requestBody)
                .build();
        Call call = okHttpClient.newCall(request);
        call.enqueue(new Callback() {//回调方法
            @Override
            public void onFailure(Call call, IOException e) {
                Log.e("text", "failure upload!" + e.getMessage());
            }


            @Override
            public void onResponse(Call call, Response response) throws
IOException {
                Log.i("text", "success upload!");
                ans = response.body().string();
            }
        });
        while(ans==null){
```

```
            }
            return ans;
        }
}
```

后端代码

```
    @ResponseBody
@RequestMapping("/changeUser")
public String rejister(@RequestParam(value = "u_id")String u_id
            ,@RequestParam(value = "u_username")String u_sername
            ,@RequestParam(value = "u_password")String password
            ,@RequestParam(value = "u_age")String u_age
            ,@RequestParam(value = "u_phone")String u_phone
            ,@RequestParam(value = "u_hobby")String u_hobby) throws
SQLException {
        DATA_BASE db = new DATA_BASE();
        String insert_sql="UPDATE UserList SET u_username
='"+u_sername+"',u_password='"+password+"',u_age="+u_age+",u_phone='"+u
_phone+"',u_hobby='"+u_hobby+"' WHERE u_id='"+u_id+"'";
        db.exec(insert_sql);
        System.out.println(insert_sql);
        return "ssss";
}
```

3.6 管理员功能

通过 AdminActivity 来实现管理员功能。管理员可以看到所有用户的账号、歌单、上传的歌曲等公开信息，通过服务器查询数据库的相关信息返回到 app 端，管理员能够查看系统概况。

AdminActivity 的主要代码

```
    private String[] MusicMames;
private List<Music> MusicList=new ArrayList<Music>();
private List<Singer> SingerList = new ArrayList<Singer>();
private List<User> UserList= new ArrayList<User>();
```

```
private RecyclerView recyclerView;
private String ans;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.admin_fragment);
    MusicList.clear();
    ask_music();//获取歌曲列表
    recyclerView=(RecyclerView)findViewById(R.id.recyclerView);
    recyclerView.setLayoutManager(new LinearLayoutManager(this));
    recyclerView.setAdapter(new MusicAdapter(MusicList,this) );
    ImageButton search_btn = (ImageButton)findViewById(R.id.search);
    search_btn.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            TextView tex = (TextView)findViewById(R.id.s_key);
            Intent intent=new Intent(AdminActivity.this, SearchActivity.class);
            intent.putExtra("key",tex.getText().toString());
            startActivity(intent);
        }
    });
    ImageButton up_btn = (ImageButton)findViewById(R.id.upload) ;
    up_btn.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            Intent intent=new Intent(AdminActivity.this, UploadActivity.class);
            startActivity(intent);
        }
    });
    ImageView songa = (ImageView) findViewById(R.id.change_user) ;
    songa.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            Intent intent=new Intent(AdminActivity.this, LoginActivity.class);
            startActivity(intent);
        }
    });
    ImageView singermanager = (ImageView) findViewById(R.id.singer_manage);
    singermanager.setOnClickListener(new View.OnClickListener() {
```

```
            public void onClick(View v) {
                ask_singer();
                recyclerView.setLayoutManager(new
LinearLayoutManager(AdminActivity.this));
                recyclerView.setAdapter(new
SingerAdapter(SingerList,AdminActivity.this) );
            }
        });
        ImageView music_mangager = (ImageView)
findViewById(R.id.music_manager);
        music_mangager.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                ask_music();//获取歌曲列表
                recyclerView.setLayoutManager(new
LinearLayoutManager(AdminActivity.this));
                recyclerView.setAdapter(new
MusicAdapter(MusicList,AdminActivity.this) );
            }
        });
        ImageView user_mangager = (ImageView) findViewById(R.id.user_manager);
        user_mangager.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                UserList.clear();
                ask_user();//获取用户列表
                recyclerView.setLayoutManager(new
LinearLayoutManager(AdminActivity.this));
                recyclerView.setAdapter(new
UserAdapter(UserList,AdminActivity.this) );
            }
        });
```

第四章 系统测试

4.1 注册

输入用户名、密码、性别、年龄、手机号、歌曲兴趣等信息，点击注册按钮，跳转到登录界面。

图 4.1 注册

图 4.2 提示信息

	u_id	u_usern...	u_passw...	u_sex	u_age	u_phone	u_type	u_hobby
▶	)00000001	1234	1234	男	32	1876778...	1	*NULL*
	U00000...	6666	6666	女	23	1559339...	0	I like mu...
	U00000...	66667	66666	男	32	1876774...	0	爱情 日语
	U00000...	12341	213	男	31231	3123	0	
	U00000...	3135	3123	男	321312	31231	0	music
	U00000...	123421	1234	男	213	312	0	
	U00000...	663225	663225	男	18	1876678...	0	国风
*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

图 4.3 数据库界面

4.2 登录

输入刚才注册的用户名和错误的密码，点击登录按钮，显示登陆失败。输入正确的密码，显示登录成功，跳转到 app 主界面。

图 4.4 登录界面

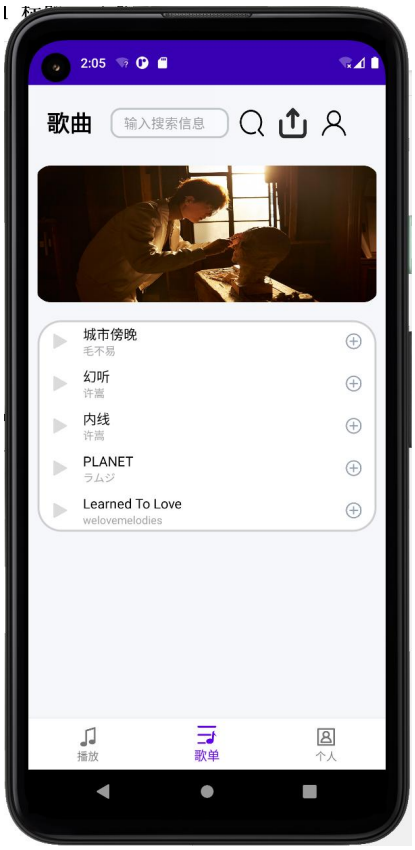

图 4.5 主界面

4.3 上传歌曲

点击上传图标样式的按钮，跳转到上传音乐界面，输入歌曲名、歌手名，选择歌曲文件的路径和歌曲类别，点击提交按钮，显示上传成功，跳转到主界面，在歌曲列表中显示出刚才上传的歌曲名和歌手。

图 4.6 上传歌曲界面

图 4.7 主界面

	m_id	m_name	m_url	m_singer	m_type	m_durat...	m_userid
▶	)00000001	城市傍晚	http://1...	S000000...	0	251496	U00000...
	M00000...	幻听	http://1...	S000000...	0	273241	U00000...
	M00000...	内线	http://1...	S000000...	0	247458	U00000...
	M00000...	PLANET	http://1...	S000000...	0	243435	U00000...
	M00000...	Learned ...	http://1...	S000000...	0	176875	U00000...
	M00000...	清明雨上	http://1...	S000000...	0	219899	U00000...
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

图 4.8 数据库界面

4.4 下载歌曲

进入界面后，系统自动读取服务器端和本地歌曲的差异，向服务器发起请求，服务器返回下载地址和文件名，发送给本地。

图 4.9 提示信息

图 4.10 本地歌曲界面

4.5 歌曲播放、切换、进度条

点击歌曲播放模式的按钮，可以进行单曲循环、列表循环、随机播放三种播放模式的切换；点击暂停按钮，可以暂停歌曲，再次点击，歌曲继续播放；点击上一首下一首的按钮，可以切换播放列表中的歌曲；单击下方的播放列表，可以播放指定的歌曲，并显示指定歌曲的歌曲名和歌手。

图 4.11 歌曲播放界面

图 4.12 歌曲播放界面

图 4.13 歌曲播放界面

图 4.14 歌曲播放界面

4.6 修改个人信息

点击修改按钮，从服务器获得个人信息，修改之后提交。

图 4.15 修改个人信息界面

	U00000...	6666	6666	女	23	1559339...	0	I like mu...

图 4.16 数据库界面

4.7 管理员界面

输入管理员账号密码 1234，进入管理员页面，能够对用户，歌曲，歌手信息进行管理和修改。

图 4.17 管理员界面

图 4.18 管理员界面


2:17
管理员
输入搜索信息
欢迎 管理用户
音乐管理
歌单管理
用户管理
歌手管理
列表
毛不易
许嵩
ラムジ
welovemelodies

图 4.19 管理员界面

第五章 总结

我们小组用了三周时间完成了这次的课程设计，工作量很大，但是很有收获。我们实现了一个完整的安卓 APP，并使用前后端分离的架构。这次的课程设计综合了之前所学的《Android 应用开发技术》和《Android 应用开发技术实验》等课程的知识，不仅检验了我们所学习的知识，也培养了我们的实践能力和动手能力。

在团队合作方面，在一开始，我们就进行了分工，我们制作了分工表，每个人负责一部分任务，最后进行整合，在这个过程中，我们锻炼了团队合作的能力，由于第一次合作，一开始有些地方配合的还不是很好，比如 Android 开发的同步问题，后来我们决定使用 Git 进行版本的控制和同步，达到了很好的效果，大大提升了开发效率，在团队合作中组员之间的沟通是必不可少的，当遇到问题时，沟通是解决问题最有效的方法，我们小组每隔一两天就进行一次汇报工作，汇报自己的进展和遇到的问题，平时我们使用微信群和 qq 群进行实时沟通，遇到棘手的问题，大家一起想办法解决。通过这次实践，我们理解了团队合作和团队管理的重要性，合理的分工，有效的配合，能够达到 1 加 1 大于 2 的效果。我们制定了合理的分工计划和沟通计划，四位组员各司其职，紧密合作，使得效率最大化。

在知识方面，我们首先进行了需求分析，确定了要实现的功能。接着进行了系统的整体设计，包括 APP 设计、SpringBoot 后端设计、SQL Server 数据库设计等多个模块。之后进行系统的实现，分工进行编码工作，然后进行集成。最后进行系统测试，修复一些 bug，保证系统稳定可用。在这个过程中，我们学到了许多之前没有了解过的技术，比如使用 navigation+fragment 实现导航栏的功能，使用 Springboot 架构实现前后端的分离，使用 http 接口实现 app 和服务器端的通信。通过这次实践，我们复习了书本上学习过的知识，也学到了很多书本上学不到的知识，将知识和实践相结合，对 Android 应用开发的过程有了更深刻的理解。

在实现的过程中，我们也遇到了很多问题，经过讨论和查找资料，我们最终顺利完成了本次课程设计。比如在数据库的开发过程中，我们设置了一个自增的主键，但是一直无法实现这个效果，后来我们查阅资料发现 SQL Server 和 MySql 的自增语法不一样，SQL Server 是 identity(1,1)，而 MySql 是 auto_increment，更正错误后，问题就迎刃而解了。在系统测试阶段，我们也发现了许多 bug，经过调试寻找问题，不断进行改进和完善。

课程设计是我们专业课程知识综合应用的实践训练，这也是我们迈向社会，从事职业工作前一个必不少的过程。“千里之行始于足下”，通过这次课程设计，

我们深深体会到这句千古名言的真正含义。这次小组分工配合，让我们深刻认识到，我们现在能够认真地进行课程设计，学会脚踏实地迈开这一步，就是在为明天能够稳健地在社会大潮中奔跑打下坚实的基础。在这次设计过程中，体现出我们整体设计并不断完善一个系统的能力以及综合运用知识的能力，能够学以致用，感受收获劳动成果的喜悦，同时，能够从中发现我们平时学习的不足和薄弱环节，从而加以弥补。