大连海事大学 2020-2021（2）学期 序号：8

移动应用开发课程实践报告

项目名称：网上音乐系统

项目组名称：4A+景区

项目组成员：彭小锐 董俊豪 刘天 赵纪淘

姓名	学号	任务	成绩
彭小锐	2220180665	主要为：APP 音乐部分实现	
董俊豪	2220180312	主要为：APP 后端、云库部分实现	
刘天	2220180423	主要为：APPxml 设计、管理员页面	
赵纪淘	2220180289	主要为：组长，APP 个人信息部分实现。	

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
彭小锐	2220180665	音乐界面，歌单界面，音乐本地流控制，APP 串联逻辑。登录逻辑。APP 架构、控制逻辑，文件上下流控制，APP Util 文件控制。GIT 控制。Navigation 主要控制	25%
董俊豪	2220180312	后端搭建、云端数据库架构、搭建、云端数据库接口设计、控制。云端控制逻辑。APP 用户信息修改。APP 端用户登录。GIT 控制。	25%
刘天	2220180423	音乐管理员界面设计逻辑。音乐界面、歌单界面、我的界面、功能界面 UI 设计。本地数据库设计、本地数据库逻辑控制、本地数据库处理。	25%
赵纪淘	2220180289	选题，项目方向设计，项目整体控制，架构，功能，上传，我的界面逻辑串联。APP 控制逻辑，本地流控制。文档。Navigation 辅助控制。	25%

目录

1. 需求分析

1.1 背景描述

在国家针对性的政策与方针支持、互联网经济高速发展、以及社会各群体的精神文化消费需求不断增长这三大作用力的积极推动下，我国音乐产业的规模得以稳定、快速地壮大，其在人们文化生活中占据的位置也愈发重要。根据中国传媒大学音乐与录音艺术学院音乐产业项目组于 2018 年底发布的《2018 中国音乐产业发展报告》的数据显示，2017 年中国音乐产业总规模已经达到 3470.94 亿元，环比增长 6.7%。其中，以音乐演出和数字音乐产业为主的音乐产业核心层产值规模达到 772.66 亿元，占产业总值的 22.26%。得益于产业核心层产值与收入的快速增长，音乐演出与数字音乐产业，尤其是处于产业链条上游及核心位置的音乐内容创作群体也就获得了与产业产值增长水平相匹配的获得增长。

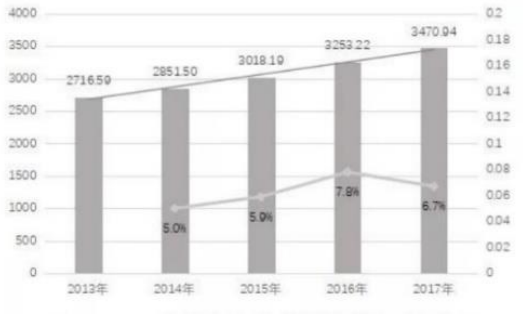


Figure 1 **中国** 2013-2017 **音乐市场**

1.2 痛点分析

目前国内比较活跃的音乐 APP 主要是采用 PGC 模式。并且由于版权的问题，国内很多音乐爱好者不得不使用多个不同的音乐 APP 来完成自己的听歌体验。而不同音乐 APP 是互斥的，一个用户不能够方便的同时听两个 APP 的歌曲。

而我们的 APP 正式针对这一个痛点而产生的。我们的 APP 关注于管理用户的歌曲。让用户在移动端可以自由的享受所有自己所能获取到的歌曲。并且，我们的平台还支持 UGC 模式。用户可以从别的用户那里获取到他们上传的歌曲。拓展了用户对音乐的接受面。

1.3 需求描述

根据 1.1 和 1.2 的内容我们可以得出，整个系统需要两个大板块的功能，普通用户模块和歌曲管理模块。

对于这两个模块当中，将需求详细描述下来，也就是如下的内容。

<1. 普通用户模块（P1）

普通用户模块主要是在手机 APP 端口，为用户提供个人听歌、检索歌曲、歌曲管理的个人音乐助手，主要需要的功能有：

1）个人信息控制。用户能够区别于别的用户，产生自己独立的信息，也就是我们所说的注册、登录功能。用户能够对自己的个人信息做改动，来满足个人的需要。

2）播放歌曲功能。用户在 APP 当中可以尽情的享受歌曲的快乐。能够比较好的完成听歌这项活动。而这项活动当中，包括了对播放歌曲的暂停、播放、切歌、设置播放顺序、后台播放等一系列的功能。

3）歌单管理功能。用户在 APP 上可以使用歌单来管理的自己的歌曲。一首歌曲可以存在于多个歌单当中，一个歌单也可以添加多个歌曲。查询、更改歌单内容是必不可少的。

4）音乐上下行功能。在 APP 平台当中，用户一方面可以将手机当中的本地歌曲上传至服务器当中，另一方面，也可以将云当中的歌曲下载下来听。这两个方面将解决用户获取云数据，云保存用户状态的功能。

5）歌曲查找功能。在 APP 平台当中，用户可以在 APP 当中使用查找功能，对所有在云当中的歌曲进行检索，获取自己想要的歌曲信息。而查找功能应该是模糊的，并且可以通过不同的索引来查找的。

<2. 歌曲管理模块（P2）

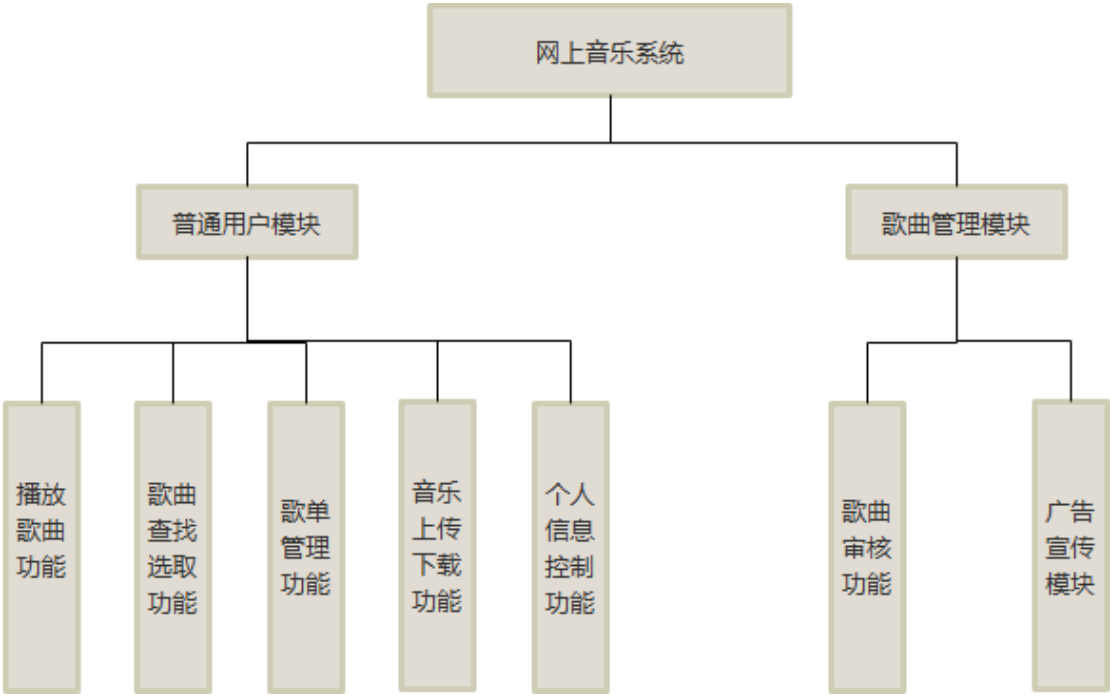
歌曲管理模块主要面向的是音乐系统的管理员。他们主要是通过 APP 来对上传至云的公开的歌曲进行审核。于此同时也负责平台一些基础的管理。歌曲的管理员不能被注册，需要后端直接授予权限。

1）歌曲审核功能。歌曲管理员可以在 APP 上进行对当前需要审核的内容进行查看，并对这些内容进行审核。以此达到控制 APP 端所有公共音乐的目的。

2）广告功能。歌曲管理元可以通过一定手段来设置广告和广告信息，能够让所有使用此 APP 的普通用户在使用这款 APP 的过程当中看到广告信息。

1.4 核心需求及结构图

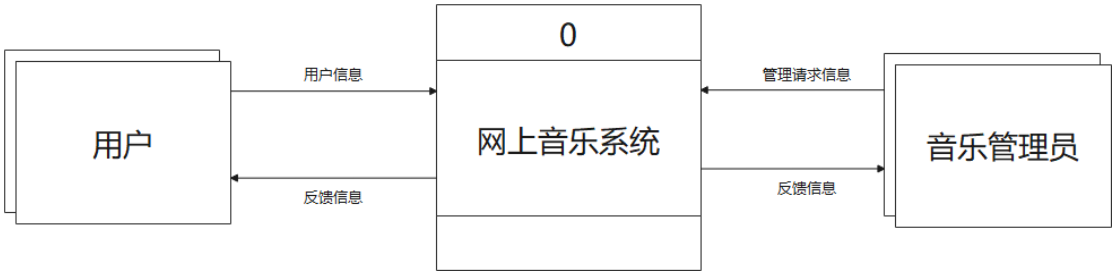
为解决 1.1 和 1.2 当中提出的问题，我们 APP 的核心功能也就出现了。面向 APP 的用户来说，可以得出以下功能图（此图仅为需求分析时期的图，具体面向编程所需要的图会在之后的功能设计当中体现）。

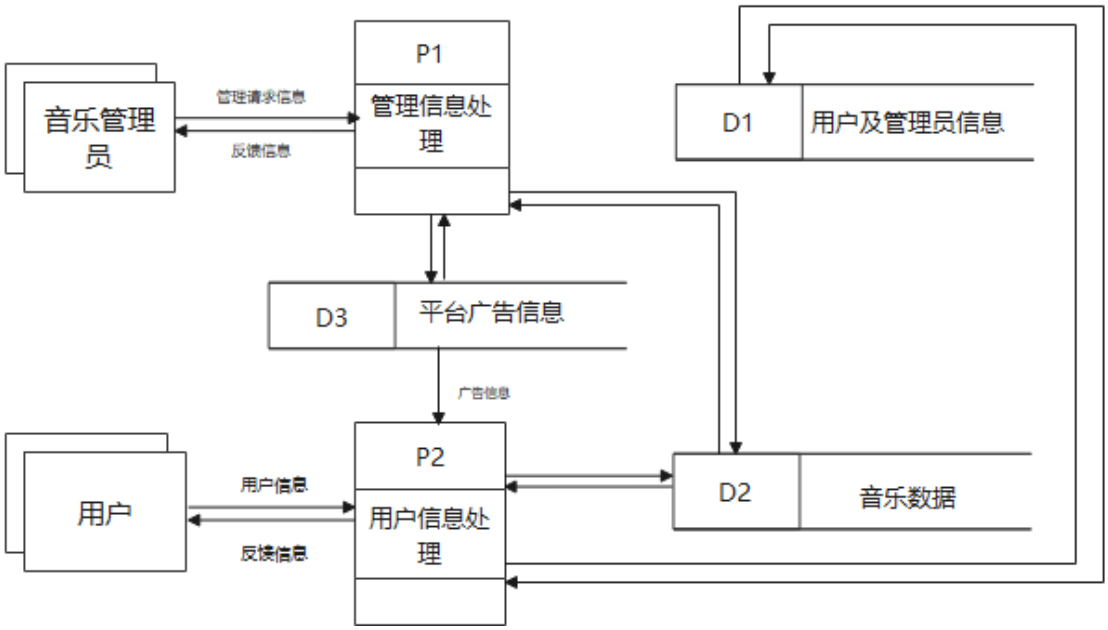


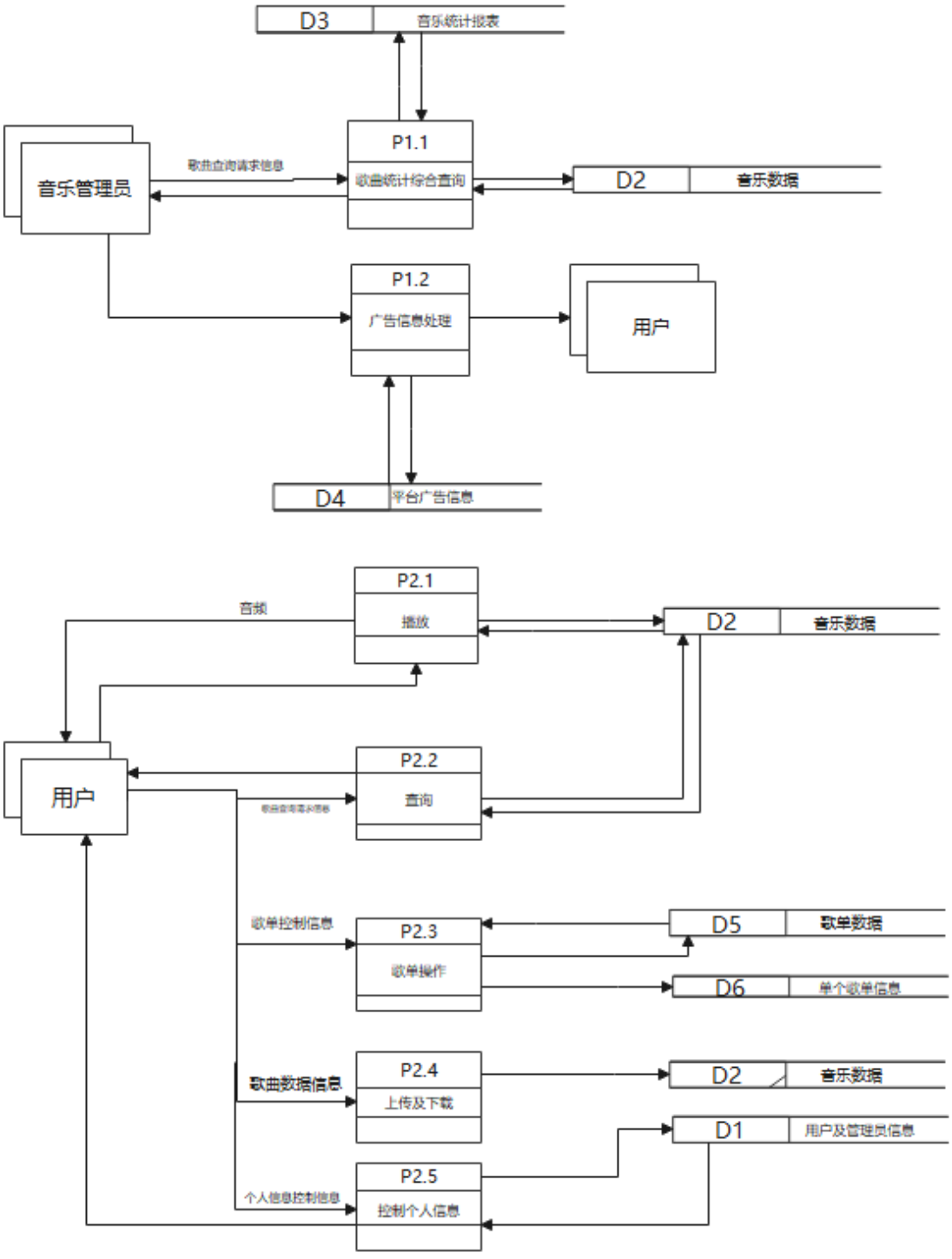
1.5 数据流程图

将 1.4 中的功能再次抽象出来得出数据流程图。这里数据流图仅用于为明确后文的开发设计。

注：此数据流图仅限于完成逻辑构建，详细数据字典见设计文档


0
用户
用户信息
反馈信息
网上音乐系统
管理请求信息
反馈信息
音乐管理员

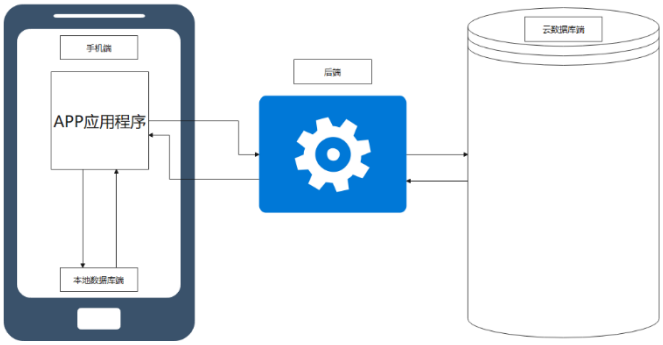

音乐管理员
管理请求信息
反馈信息
P1
管理信息处理
D1
用户及管理员信息
D3
平台广告信息
广告信息
用户
用户信息
反馈信息
P2
用户信息处理
D2
音乐数据


D3
音乐统计报表
P1.1
歌曲统计综合查询
歌曲查询请求信息
音乐管理员
D2
音乐数据
P1.2
广告信息处理
用户
D4
平台广告信息
P2.1
播放
音频
D2
音乐数据
用户
P2.2
查询
歌曲查询请求信息
歌单控制信息
P2.3
歌单操作
D5
歌单数据
D6
单个歌单信息
歌曲数据信息
P2.4
上传及下载
D2
音乐数据
D1
用户及管理员信息
P2.5
个人信息控制信息
控制个人信息

2. 功能设计

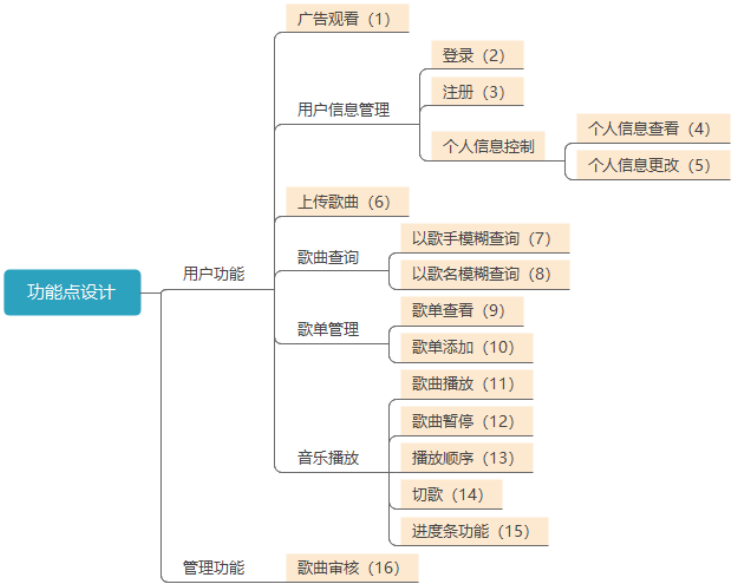
2.1 宏观设计思路

逻辑串联思路：如下图所示，手机 APP 端口通过接口访问后端，后端对云数据库进行操控，云数据库处理信息之后返给后端，后端将信息传回给手机。于此同时，手机可以与本地数据库进行交互访问。



2.2 功能点设计（功能点）

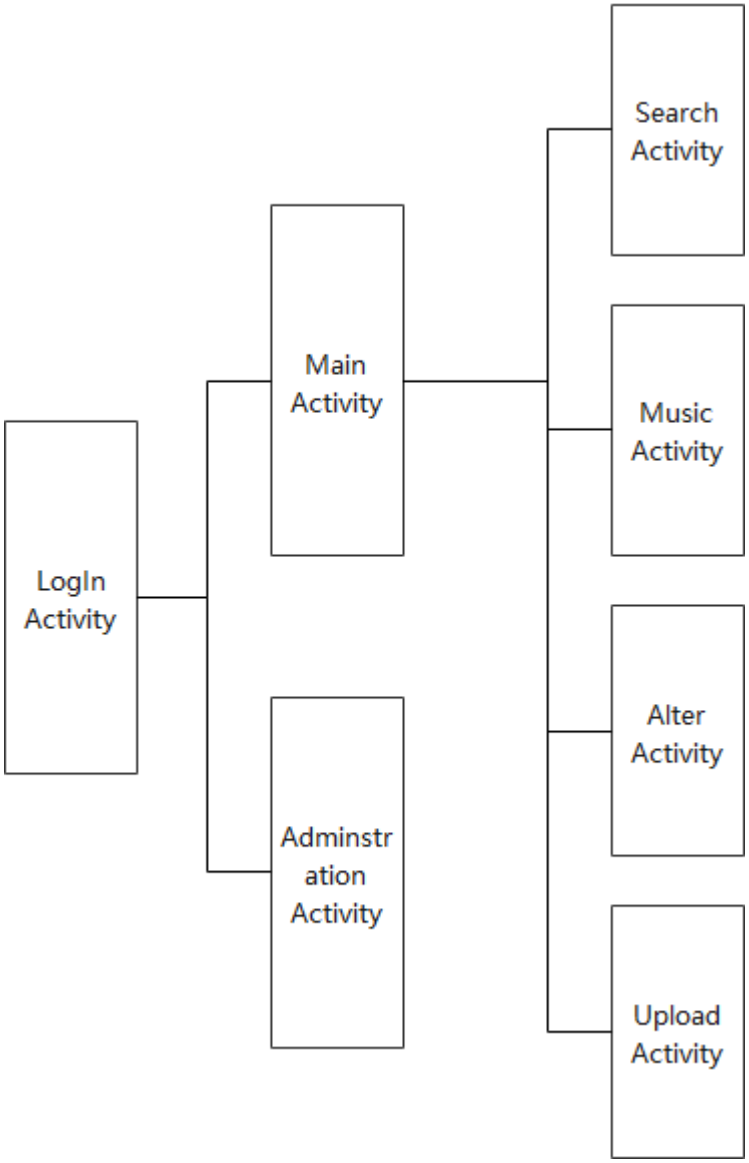
将第一部分的内容具体结构化，使其功能尽量内聚，得出如下图。可以看到，本 APP 一共有 16 个功能点。



2.3 APP 设计

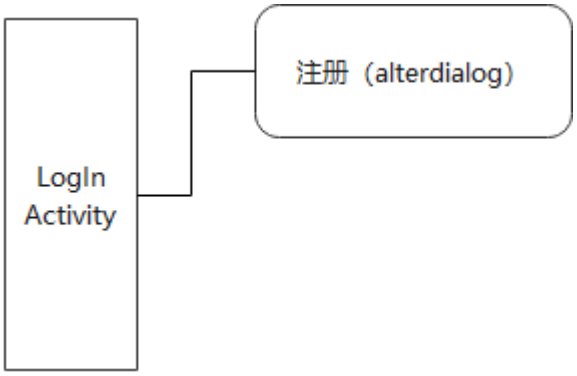
2.3.1 Activity 链接设计（Activity 点）

根据功能点之间的逻辑关系，共设计了 7 个 Activity 来完成此功能



2.3.2 Activity 内部设计

2.3.2.1 LoginActivity

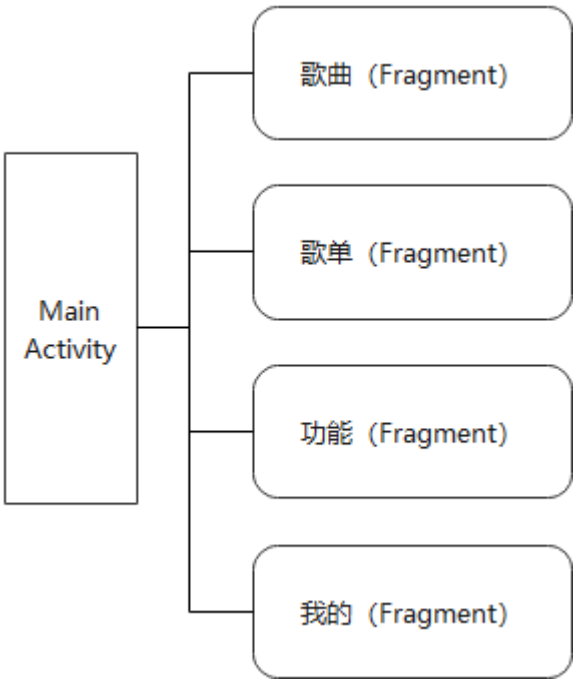


LoginActivity 内部包含注册 alterdialog，完成功能：注册、登录跳转。内涵逻辑：注册信息形式检查、记住账号密码。

2.3.2.2 AdministrationActivity

单独的 Activity，内部包含音乐管理员界面，管理歌曲。

2.3.2.3 MainActivity

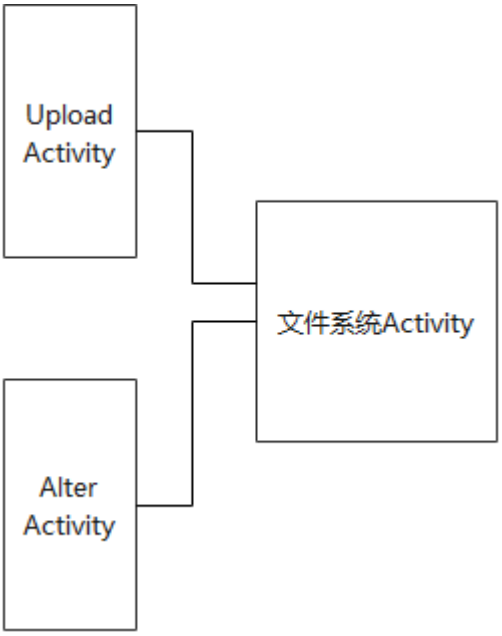


MainActivity 使用 Navigation 框架，内涵四个 navigation fragment，使

用 navigationController 来控制。通过重写其中 fragmentNavigator 类来实现 Fragment 的切换而不死亡。

实现功能：歌曲 Fragment：音乐的播放、暂停、切歌、播放顺序调整、进度条、添加歌单功能。我的 Fragment：个人信息查看。功能 Fragment：广告查看。

2.3.2.4 UploadActivity & AlterActivity



这两个 Activity 分别涉及音乐文件的上传和个人信息（包含图片）的上传。需要使用到文件系统的界面来选取所需要的文件。获取到其 URL 之后，通过工具进行上传。

完成功能：上传歌曲，修改个人信息。

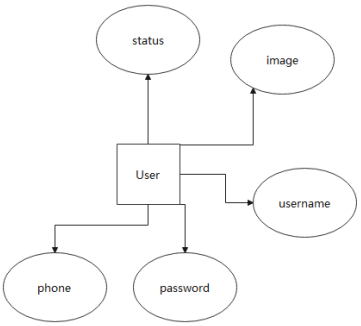
2.3.2.5 Others

SearchActivity 和 MusicActivity。这两个分别是用于显示歌单内歌曲和显示搜索歌曲结果。

完成功能：歌单查看，歌曲按人名模糊搜索，歌曲按名字模糊搜索。

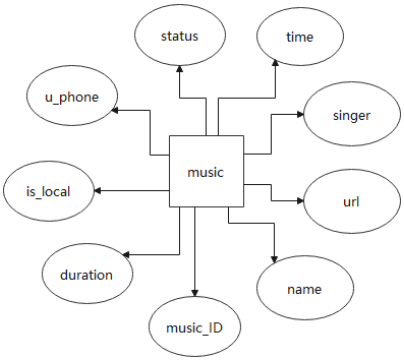
2.3.3 数据类基础设计

2.3.3.1 基础类设计



a. User 类(静态)：User 类承载着用户的基本信息，这个类是用户方法的核心。

重要方法：void updateInfo()：对接数据库，更新用户信息。

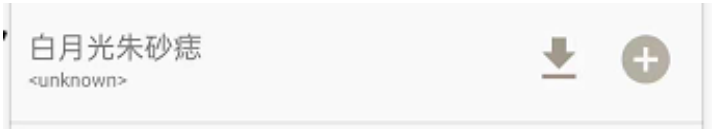


b. Music 类：Music 类承载着歌曲的基本信息，这个类是歌曲播放的核心。

2.3.3.2 歌条和歌单

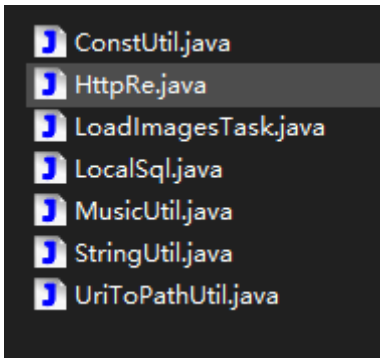
歌条和歌单是两个主要的 UI 形式。也是整个承载着音乐系统的核心部分：音乐的播放和音乐的管理。

歌条在 playerFragment 当中是以 listView 当中的 item 而出现的。使用 adapter 产生一个个歌条填充到 listView 当中。



2.3.4 Util 设计

安卓 APP 当中需要复用很多相同的文件传输方法。比如和后端链接的 Http 请求，比如 String 和 JSON 互相转换等方法。



1） 使用 Http 请求向后端请求信息数据。
2） 使用 Http 像后端上传信息数据。
3） String、URL、JSON 转换器。
4） 使用 Http 向后端发送文件。
5） 使用 Http 向后端获取文件。
6） Uri 和 Path 相互转换。
7） Local 的 Sqlite 调用接口

2.3.5 界面设计（通过 PathData 绘制 VavtorImage）

界面设计主题风格为简约风格。简约风格更高效、更清楚的展示我们的程序。相对来说 UI 代码也简单一些。

由于为简约的设计，我们更需要的是 Vector 形式的向量。主要界面的图片使用 Vector。使用 PathData 文法规则绘制自己想要的矢量图。

2.4 数据库设计

2.4.1 云端数据库设计

云端数据库采用 MySql，使用 Spring Boot 后端与其交互

云端数据库主要存储 User 用于云端用户信息的管理。Music 表来控制所有云端上传的歌曲。Music_list 对所有云端的歌单进行管理。Music_music_list 来控制多对多的歌曲歌单关系。

表名	字段	键	格式	字段说明
用户表				
user	phone	主键	varchar(1	用户注册使用的手机号(用户唯一标识)
	passward		varchar(2(	登入密码
	username		varcahr(1(	昵称
	image		varchar(1(	头像文件url
	status		int	用户身份(0为普通用户，1为总管理员
音乐表				
music	music_id	主键	int(increm	音乐唯一标识
	name		varchar(1(	音乐名
	url		varchar(1(	音乐文件url
	singer		varchar(1(	歌手名
	time		data	上传时间
	status		int	是否公开(0为私密，1为公开，2为待审核)
		外键	varchar(1	用户手机号
歌单表				
music_list	list_id	主键	int(increm	歌单id(唯一标识)
	u_phone	外键	varchar(1	用户手机号
	list_name		varchar(1(	歌单名
	time		date	歌单创建日期
歌单-歌曲关系表				
music_musicList	music_id	外键	int	音乐id
	list_id	外键	int	歌单id

2.4.2 本地数据库设计

安卓本地 Sqlite 数据库主要通过安卓 Utils 下的 LocalSql 类进行调用和程序交互。

Local_music 主要存储本地歌曲信息。每次链接数据库时会更新本地歌曲信息。Music_musicList 存储本地音乐和歌单的多对多关系。Music_list 信息会存储到云端库。查看时会先查询云端库的信息再查询本地库的信息，之后集成。

SQLLITE				
表名	字段	键	格式	字段说明
本地歌曲				
local_music	id	主键	varchar(1(	音乐标识
	name		varchar(1(	歌名
	singer		varchar(1(	歌手名
	isLocal		int	是否为下载歌曲(0为本地，1为下载歌曲)
	path		varchar(1(	文件存储路径
本地音乐和歌单的关系				
music_musicList	id	外键	varchar(1(	音乐id
	list_id	外键	varchar(1(	歌单id

2.5 后端接口设计（公网后端 阿里云服务器）

后端通过 Spring Boot 管理，架设在阿里云服务（47.94.160.152:8000）以下为设计好的后端接口。

接口描述	接口	接口参数	请求方式	返回值
增加user	/addUser	phone=?&password=?&username=?	POST	true/false
删除user	/deleteUser	phone=?	POST	true/false
更改user	/alterUser	phone=?&password=?&username=?	POST	true/false
查全部user	/findAllUser	null	GET	List<user>
登入	/login	phone=?,passsword=?	POST	user/false
更改头像	/changeImage	phone=?/FILE	POST	true/false
通过手机号获得用户所有信息	/getUserInfo	phone=?	POST	user
增加歌曲	/addMusic	name=?,singer=? ,status=?,u_phone=?,FILE	POST	true/false
删除歌曲	/deleteMusic	id=?	POST	true/false
更改歌曲	/alterMusic	id=?,name=?,singer=?,status=?	POST	true/false
通过歌名搜索歌曲	/searchByName	name=?	POST	List<music>
通过歌手名搜索歌曲	/searchBySinger	singer=?	POST	List<music>
查全部歌曲	/findAllMusic	null	GET	List<music>
根据用户歌单获取歌单歌曲	/findSongByList	list_id=?	POST	List<music>

2.6 编码设计

2.6.1 编码三级顺序及分布式开发（Git 管理）

编码顺序：

登录 Activity 设为一级界面，主页面、管理员界面设为二级界面，其余功能拓展设为三级界面。编码顺序按照，先一级界面，之后二级界面，扩展三级界面的方式来进行。

编码分布式开发：

Xml 界面和逻辑界面分离。先进行 Xml 界面的生成，后串行逻辑结构。统计页面逻辑先内部分布式编码，再外部集成。以彭小锐机器为核心，进行集成。以董俊豪为核心进行后端测试。

工具类提前规划，于第一次使用时进行开发顺便测试，测试成功后进行集成化管理。

Git 上传：

由于开发前期框架没有搭起来，整个的开发类结构、静态值比较松散，直接 Git 上传会导致每次 Git 集成过于复杂。所以我们以董俊豪、彭小锐两人的主机作为集成机，赵纪淘与彭小锐，刘天与董俊豪对接。对接之后两人再进行 Git 上传合并。

2.6.2 编码规范

统一使用 Android 编码格式。分布式开发 java 内所有内容命名为：NameParents，xml 所有命名格式为 name_parents。遵循驼峰原则。

编码分布式开发过程当中，所有新建类使用姓名首字母小写作为开头，后加命名格式。统一使用 Git 管理，能够自动识别改动文件。

2.7 **navigation 实现 fragment 不重生复用设计

Navigation 是一种导航概念，是安卓执行一种单 Activity 实现多功能的一种设计。通过 Navigation 的源地址和目标地址（将 Fragement 抽象成 Destination）实现对 Fragment 的管理。我们所构想的页面符合 Navigation 底

部菜单栏的设计。所以我们决定使用 Navigation 作为模板实现我们用户主界面（MainActivity）。

Navigation 通过 AppBarConfiguration、NavigationUI 分别存储目的地和控制参数。再同 NavigationUI 的 setupWithNavController 绑定两者。

但是，这种方法在给我们提供设计上、逻辑上的方便的同时，它也给我们带来了不好的以面：在 Navigation 下，将 Fragment 的切换写死成了重画，也就是说每次切换都会导致 fragment 的重建。而我们的一个 fragment 预想是歌曲播放，切出 fragment 停止播放或者再次加载报错显然是不合理的。而如果使用 ViewModel 来传递信息，希望借此解决这个问题也是会有逻辑上的不通顺。

最终我们决定使用覆盖的方法来解决这个问题。如果想要解决这个问题，需要将转换的过程变成 hide()和 show()。所以我们将 FragmentNavigation 继承下来，并在新类重写其 navigate 方法。而这也会导致符类的 Stack 管理上出问题。所以要对符类的 mBackStack 进行获取，并将符类的 generateBackStackName 也拿过来。最后使用我们自己的 navigator 来进行导航即可达成。

3．功能实现

3.1 登入、注册

3.1.1 Code

该功能在 LoginActivity 提供用户登录功能和注册功能，通过 EditTextView 输入手机号和密码，点击登录进行用户登录，通过编写异步工具类 HTTPTask 类结合 LoginHandler，实现异步获取是否登入信息。或者通过点击注册，会弹出 AlertDialog，用户输入手机号和密码进行注册，这里也是使用 http 请求完成的，因此也是使用 HttpTask。

LoginActivity 的部分代码

```
/**
 * register 控件点击事件
 */
private void doRegister() {
    final AlertDialog.Builder builder = new AlertDialog.Builder(this);
    @SuppressLint("InflateParams") final View alert =
getLayoutInflater().inflate(R.layout.register_alertdialog_layout, null);
    final EditText name = alert.findViewById(R.id.r_name_et);
    final EditText username = alert.findViewById(R.id.r_phone_et);
```

```
    final EditText password = alert.findViewById(R.id.r_password_et);
    final Button register = alert.findViewById(R.id.register_btn);
    builder.setView(alert);
    AlertDialog dialog = builder.create();
    register.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String names = name.getText().toString();
            String usernames = username.getText().toString();
            String passwords = password.getText().toString();

            if (StringUtil.isEmpty(names)
|| !StringUtil.isNumber(usernames) || StringUtil.isEmpty(passwords)){
                Toast.makeText(LoginActivity.this, "数据格式有误，请重新
输入", Toast.LENGTH_LONG).show();
            }else{
                Toast.makeText(LoginActivity.this, "正在注册，请稍
后...", Toast.LENGTH_LONG).show();
                new HttpTask(ConstUtil.REGISTER_URL, new
LoginHttpHandler(LoginActivity.this,R.id.register_btn,dialog)).execute(S
tringUtil.transformToRegister(names, usernames, passwords));
            }
        }
    });
    dialog.show();

}

/**
 * login 控件点击事件
 */
private void doLogin() {
    saveUser();//保存用户账户信息
    String phone = username.getText().toString();
    String password = passWord.getText().toString();
    if (StringUtil.isEmpty(password) || !StringUtil.isNumber(phone)
|| !StringUtil.isPhone(phone)){
        Toast.makeText(LoginActivity.this, "数据格式有误，请重新输入",
Toast.LENGTH_LONG).show();
    }else{
        new HttpTask(ConstUtil.LOGIN_URL, new LoginHttpHandler(this,
R.id.login_btn,null)).execute(StringUtil.transformToLogin(phone,
password));
```

```
    }
}
```

HttpTask 代码

```
/**
 * 用于进行 Http 请求的异步任务类（POST 请求）
 * 使用：
 * 构造方法传入一个用于接收请求结果的 Handler 和要进行请求的 URL
（String）
 * execute 方法传入要进行请求的参数（只能传一个 String 型参数）
 */
public class HttpTask extends AsyncTask<String, Integer, String> {
    private HttpMessageHandler handler;
    private String URL;

    public static final String CONNECT_OR_READ_TIMEOUT = "timeout";
    public static final String HTTP_REQUEST_RESULT = "result";

    public HttpTask(String URL, HttpMessageHandler handler) {
        this.URL = URL;
        this.handler = handler;
    }

    @Override
    protected String doInBackground(String... strings) {
        String result =
"{ \"res\":\""+CONNECT_OR_READ_TIMEOUT+"\",\"data\":[]}";
        HttpURLConnection con;
        try {
            con = (HttpURLConnection) new URL(URL).openConnection();
            con.setConnectTimeout(2000);
            con.setReadTimeout(2000);
            con.setRequestMethod("POST");
            con.setDoOutput(true);
            con.setDoInput(true);

con.getOutputStream().write(strings[0].getBytes(StandardCharsets.UTF_8))
;
            result = new BufferedReader(new
InputStreamReader(con.getInputStream(),
StandardCharsets.UTF_8)).readLine();
        } catch (IOException e) {
            e.printStackTrace();
```

```
        }
        Bundle bundle = new Bundle();
        bundle.putString(HTTP_REQUEST_RESULT, result);
        Message message = Message.obtain();
        message.setData(bundle);
        handler.sendMessage(message);
        Log.d("TTT_send", String.valueOf(strings[0]));
        Log.d("TTT_result", result);
        return result;
    }
}
```

使用 LoginHandler 负责让 activity 和 HttpTask 通信，HTTPTask 即获得后台用户登入或者注册的信息，通过 LoginHandler 处理收到的信息，并决定是否登入成功，是否跳转到 MainActivity。

对手机号和密码进行校验，如果用户名是已注册用户且用户密码正确则获取到用户的身份信息，如果是普通用户通过 Intent 跳转到主界面 MainActivity。如果用户登录的是管理员账户，则在 MainActivity 中启用管理员页面。

说到 Handler，因为我们发现使用 HTTP 请求的 Handle 有很多的共同方法，于是我编写 HTTPMessageHandler 抽象类，这样它的子类只需继承它，实现构造函数和 control 方法即可。这里还同时使用了弱引用，防止 GC 装置无法回收 Activity 而出现内存泄露。

HTTPMessageHandler

```
public abstract class HttpMessageHandler extends Handler {
    private WeakReference<AppCompatActivity> activity;



    public WeakReference<AppCompatActivity> getActivity() {
        return activity;
    }

    public void setActivity(WeakReference<AppCompatActivity> activity) {
        this.activity = activity;
    }

    public HttpMessageHandler(AppCompatActivity activity){
        this.activity = new WeakReference<>(activity);
    }

    @Override
    public void handleMessage(@NonNull Message msg) {
        super.handleMessage(msg);
```

```
        control(msg);
    }

    /**
     * 子类重新此方法即可
     * @param msg
     */
    public abstract void control(Message msg);

}
```

LoginHttpHandler

```
.       public class LoginHttpHandler extends HttpMessageHandler {

    private int view;
    private AlertDialog alertDialog;

    public LoginHttpHandler(AppCompatActivity activity,@IdRes int
view,AlertDialog alertDialog){
        super(activity);
        this.view = view;
        this.alertDialog = alertDialog;
    }



    @Override
    public void control(Message msg) {
        try {
            String result =
msg.getData().getString(HttpTask.HTTP_REQUEST_RESULT);
            JSONObject json = new JSONObject(result);
            JSONArray dataArr = json.getJSONArray("data");
            JSONObject data = null;
            if(dataArr.length()>0){
                data = dataArr.getJSONObject(0);
            }

            String res = json.getString("res");
            switch (view) {
                case R.id.login_btn:
                    switch (res) {
```

```
                        case "false":
                            Toast.makeText(super.getActivity().get(), "
账号或者密码错误" , Toast.LENGTH_LONG).show();
                            break;
                        case HttpTask.CONNECT_OR_READ_TIMEOUT:
                            Toast.makeText(super.getActivity().get(), "
网络开小差儿啦~~~", Toast.LENGTH_LONG).show();
                            break;
                        case "true":
                            if(data == null){

Toast.makeText(super.getActivity().get(), "服务器错误，登入不成功",
Toast.LENGTH_LONG).show();
                                break;
                            }
                            if(data == null){

Toast.makeText(super.getActivity().get(), "服务器错误，登入不成功",
Toast.LENGTH_LONG).show();
                                break;
                            }
                              //将所有的用户信息过得，存到静态变量中
                            ConstUtil.user = new
User(data.getString("phone"),data.getString("password"),data.getString("
username"),data.getString("image"),data.getInt("status"));
                            Toast.makeText(super.getActivity().get(), "
登录成功", Toast.LENGTH_LONG).show();
                            //页面跳转
                            Intent intent = new Intent();
                            if(ConstUtil.user.getStatus()==1){

intent.setClass(super.getActivity().get(), AdminstrationActivity.class);
                            }else{

intent.setClass(super.getActivity().get(), MainActivity.class);
                            }

intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NEW_TASK);

super.getActivity().get().startActivity(intent);
                            break;

                    }
```

```
                    break;
                case R.id.register_btn:
                    switch (res) {
                        case "true":
                            Toast.makeText(super.getActivity().get(), "
注册成功！", Toast.LENGTH_LONG).show();
                            break;
                        case HttpTask.CONNECT_OR_READ_TIMEOUT:
                            Toast.makeText(super.getActivity().get(), "
网络开小差儿啦~~~", Toast.LENGTH_LONG).show();
                            break;
                        case "false":
                            Toast.makeText(super.getActivity().get(), "
电话号码已被注册！" , Toast.LENGTH_LONG).show();
                            break;
                    }
                    break;
            }
            if (alertDialog != null)
                alertDialog.dismiss();
        }catch (JSONException e){
            e.printStackTrace();
        }

    }
}
```

除了上述功能之外，为了使用的便捷我们还加了记住密码的功能，使用 Android 的轻量级数据存储 sharedpefermance 记住用户的账号和密码。

```
/**
 * 读取用户账户信息
 */
private void readUser() {
    username.setText(sharedPreferences.getString("username", ""));
    passWord.setText(sharedPreferences.getString("password", ""));
    saveAccount.setChecked(sharedPreferences.getBoolean("isSave",
false));
}

/**
 * 保存用户账户信息
 */
```

```
private void saveUser() {
    SharedPreferences.Editor editor = sharedPreferences.edit();
    if (isSave) {
        editor.putString("username", username.getText().toString());
        editor.putString("password", passWord.getText().toString());
    } else {
        editor.putString("username", "");
        editor.putString("password", "");
    }
    editor.putBoolean("isSave", saveAccount.isChecked());
    editor.apply();
}
```

重写返回事件，防止用户误触返回.
LoginActivity 中部分代码

```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if (keyCode == KeyEvent.KEYCODE_BACK) {
        if (System.currentTimeMillis() - lastKeyDownTime < 1500) {
            System.exit(0);
        }
        Toast.makeText(this, "再按一次退出程序",
Toast.LENGTH_LONG).show();
        lastKeyDownTime = System.currentTimeMillis();
    }
    return true;
}
```

3.1.2 布局

都是使用 LinearLayout 进行布局，注册是使用 AlerDialog，对应的使用线性布局。

3.2 音乐播放，本地歌曲扫描，歌曲展示，歌曲播放控制等

3.2.1 code

这些功能主要在 FragmentPlayer 中进行实现的，通过编写的 MusicUtil 类扫描本地的 map3 歌曲返回 List<Music>，由此获得本地歌曲。

MusicUtil

```
public class MusicUtil {
    /**
     * 扫描系统里面的音频文件，返回一个 list 集合
     */
    public static List<Music> getMusicData(Context context) {
        MediaScannerConnection.scanFile(context, new
String[]{Environment
                .getExternalStorageDirectory().getAbsolutePath()}, null,
null);

        // 媒体库查询语句（写一个工具类 MusicUtils）
        Cursor cursor = context.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;
        List<Music> musicList = new ArrayList<Music>();
        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,filePath,singer,null,0,null,true,duration);
                //大于 30 秒的
                if (duration > 30000) {
                    musicList.add(song);
                }
            }while (cursor.moveToNext());
            // 释放资源
            cursor.close();
        }
        return musicList;
    }
```

将获得的歌曲列表，通过 MusicAdapter 生成 view 附加到 ListView 上，进而实现歌曲显示，同时为每个 item 添加 listener，使得用户点击歌曲即可播放歌曲或者切换到此歌曲。同时为为每个歌曲条的下载按钮和添加至歌单添加 onclick 方法。

MusicAdapter 的部分代码

```
public View getView(int position, View convertView, ViewGroup parent) {
    Music song = getItem(position);
    View view=
LayoutInflater.from(getContext()).inflate(resId,parent,false);
    TextView tvMusicName =
view.findViewById(R.id.music_name_music_item);
```

```
    TextView tvSinger = view.findViewById(R.id.singer_music_item);
    Button btDownload = view.findViewById(R.id.down_load_music_item);
    Button btAddMusicList =
view.findViewById(R.id.add_musiclist_music_item);
    btDownload.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            MusicUtil.downloadMusic(song);
        }
    });
    btAddMusicList.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            addToMusicList(song);
        }
    });

    tvMusicName.setText(song.getName());
    tvSinger.setText(song.getSinger());

    return view;
}

/**
 * 添加到歌单 music list
 */
private void addToMusicList(Music music) {
    final AlertDialog.Builder builder = new
AlertDialog.Builder(getContext());
    @SuppressLint("InflateParams") final View alert =
LayoutInflater.from(getContext()).inflate(R.layout.music_list_alertdialo
g_layout,null);

    final ListView lvMusiclists =
alert.findViewById(R.id.music_list_alertdialog);
    lvMusiclists.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
        }
    });
    MusicListAdapter adapter = new
MusicListAdapter(getContext(),R.layout.music_list_item,ConstUtil.musicLi
```

```
sts);
    lvMusiclists.setAdapter(adapter);
    builder.setView(alert);
    builder.show();
}
```

PlayerFragment 的部分

```
lsAudios = root.findViewById(R.id.current_music_list);
MusicAdapter adapter = new MusicAdapter(getActivity(),
R.layout.music_item_show, currentList);
lsAudios.setAdapter(adapter);
lsAudios.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
        Music song = currentList.get(position);
        initMediaPlayer(song);
    }
});
```

在音乐控制面板上有上一首，下一首，播放/暂停按钮，可以控制音乐的播放，同时可以选择播放模式，随机播放，循环播放或者单曲循环，这些功能的实现需要更改音乐播放顺序的逻辑。

PlayerFragment 的部分(status 表示三种播放模式)

```
btPalyMode.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        ++status;
        status %= 3;
        switch (status){
            case 0:

btPalyMode.setBackgroundResource(R.drawable.loop_playback);
                break;
            case 1:

btPalyMode.setBackgroundResource(R.drawable.single_loop_play);
                break;
            case 2:

btPalyMode.setBackgroundResource(R.drawable.random_playback);
```

```
                break;
        }
    }
});
Button btSkipPrevious = root.findViewById(R.id.btn_skip_previous);
btSkipPrevious.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(mediaPlayer == null) return;
        switch (status){
            case 0:
                currentPositon--;
                currentPositon%=currentList.size();
                initMediaPlayer(currentList.get(currentPositon));
                break;
            case 1:
                initMediaPlayer(currentList.get(currentPositon));
                break;
            case 2:
                currentPositon = new
Random().nextInt(currentList.size());
                initMediaPlayer(currentList.get(currentPositon));
                break;
        }
    }
});

Button btSkipNext = root.findViewById(R.id.btn_skip_next);
btSkipNext.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        switch (status){
            case 0:
                currentPositon++;
                currentPositon%=currentList.size();
                initMediaPlayer(currentList.get(currentPositon));
                break;
            case 1:
                initMediaPlayer(currentList.get(currentPositon));
                break;
            case 2:
                currentPositon = new
Random().nextInt(currentList.size());
                initMediaPlayer(currentList.get(currentPositon));
```

```
                break;
            }
        }
});
```

Seekbar 拖动进度条，歌曲进度跟进

```
sbSeek.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;
        tvLeft.setText(m + ":" + s );
    }
});
```

暂停与播放逻辑

```
btPlay.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (mediaPlayer == null) return;
        if (isPlay){
            isPlay = false;
            mediaPlayer.pause();
            mAnimator.pause();
            btPlay.setBackgroundResource(R.drawable.play);
        }else {
            isPlay = true;
            mediaPlayer.start();
```

```
            mAnimator.resume();
            btPlay.setBackgroundResource(R.drawable.pause);
        }
    }
});
```

添加到歌单

```
btAddToMusiclist.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (mediaPlayer == null) return;
        addToMusicList(currentList.get(currentPositon));
    }
});
```

使用线程和 Handler 进行更新进度条，和更新播放时间

```
Handler handler = new Handler(){
     @Override
     public void handleMessage(Message msg) {
         if (seekBarChange) return;
         sbSeek.setProgress(mediaPlayer.getCurrentPosition());
         int m = mediaPlayer.getCurrentPosition() / 60000;
         int s = (mediaPlayer.getCurrentPosition() - m * 60000) / 1000;
         tvLeft.setText(m + ":" + s );
     }
 };
new Thread(new Runnable() {
    @Override
    public void run() {
        while(true){
            handler.sendEmptyMessage(0);
            try {
                Thread.sleep(1000);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}).start();
```

3.2.2 布局

Fragment_player.xml

整体为 contrainlayout 布局，有两个子 constraintlayout 布局，上面的布局 up_fragment_play 中有一个 textView 和一个 ImageView，和两个 constraintLayout 组成，这里分别是进度条时间为一个，控制按钮在一个。下面的 down_fragment_play 包含一个 ListView。

3.3 歌单管理，歌曲管理

3.3.1 code

显示用户歌单，管理歌单。将在 Mainactivity 中获得到的用户歌单即 ConstUtil.获得使用 MusicListAdapter，填充列表。

注册搜索按钮，将用户查询的信息用 Intent 传给 searchActivity，跳转到搜索结果页面，searchActivity 通过传递过来的关键字使用异步 Http 请求从后端获取查找结果。

给歌单项添加 OnclickListener，点击歌单项进入歌单，跳转到 MusicActivity，需要将选中的 musiclist 是 Bundle 打包传入 MusicActivity。

MusicListFragment.java

```
public class MusicListFragment extends Fragment {
```

```
    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle
savedInstanceState) {
        View root = inflater.inflate(R.layout.fragment_music_list,
container, false);
        ListView music_lists =
root.findViewById(R.id.music_list_view_fragment_music_list);
        MusicListAdapter adapter = new
MusicListAdapter(getContext(),R.layout.music_list_item,
ConstUtil.musicLists);

        music_lists.setAdapter(adapter);

        music_lists.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view,
int position, long id) {
                //歌单跳转到 MusicActivity，将选中的 musiclist 传入
                Intent intent = new Intent(getActivity(),
MusicActivity.class);
                Bundle bundle = new Bundle();

bundle.putSerializable("musiclist",ConstUtil.musicLists.get(position));
                intent.putExtras(bundle);
                startActivity(intent);
            }
        });

        root.findViewById(R.id.search_img_search).setOnClickListener(new
View.OnClickListener(){
            public void onClick(View v){
                Intent intent=new Intent(getActivity(),
SearchActivity.class);
                EditText editText =
root.findViewById(R.id.search_text_search);
                //将用户查询的信息传给 searchActivity
                intent.putExtra("key",editText.getText().toString());
                intent.putExtra("type","singer");
                startActivity(intent);
            }
        });

        return root;
```

```
    }
}
```

SearchActivity

```
public class SearchActivity extends AppCompatActivity{

    private ListView lvMusicList;

    private List<Music> musics;// 数据

    private MusicAdvanceAdapter adapter;

    private MusicList musicList;


    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_search_layout);

        String data = getIntent().getStringExtra("key");
        String type = getIntent().getStringExtra("type");


        lvMusicList =
findViewById(R.id.music_list_music_activity_layout);

        musicList =
(MusicList)getIntent().getSerializableExtra("musiclist");

        //搜索歌曲数据
        if(type.equals("name")){
            new HttpTask(ConstUtil.SEARCH_NAME_URL,new
SearchMusicHandler(this)).execute(("name="+data));
        }else{
            new HttpTask(ConstUtil.SEARCH_SINGER_URL,new
SearchMusicHandler(this)).execute(("singer="+data));
        }


    }
```

```
    public void initListView(List<Music> musics){
        this.musics = musics;
        adapter = new
MusicAdvanceAdapter(this,R.layout.music_item_search,musics);
        lvMusicList.setAdapter(adapter);
        lvMusicList.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view,
int position, long id) {

            }
        });
    }


}
```

使用 HttpTask 当搜索完毕时，通过 SearchMusicHandler 处理收到的搜索结果。SearchMusicHandler 保留了 activity 的引用，SearchMusicHandler 在数据处理完后更新 SearchActivity。

SearchMusicHandler

```
public class SearchMusicHandler extends HttpMessageHandler {
    private WeakReference<SearchActivity> activity;

    public SearchMusicHandler(SearchActivity activity) {
        super(activity);
        this.activity = new WeakReference<>(activity);
    }

    @Override
    public void control(Message msg) {
        try {
            String result =
msg.getData().getString(HttpTask.HTTP_REQUEST_RESULT);
            JSONObject json = new JSONObject(result);
            JSONArray dataArr = json.getJSONArray("data");
            JSONObject data = null;
            if(dataArr.length()>0){
                data = dataArr.getJSONObject(0);
```

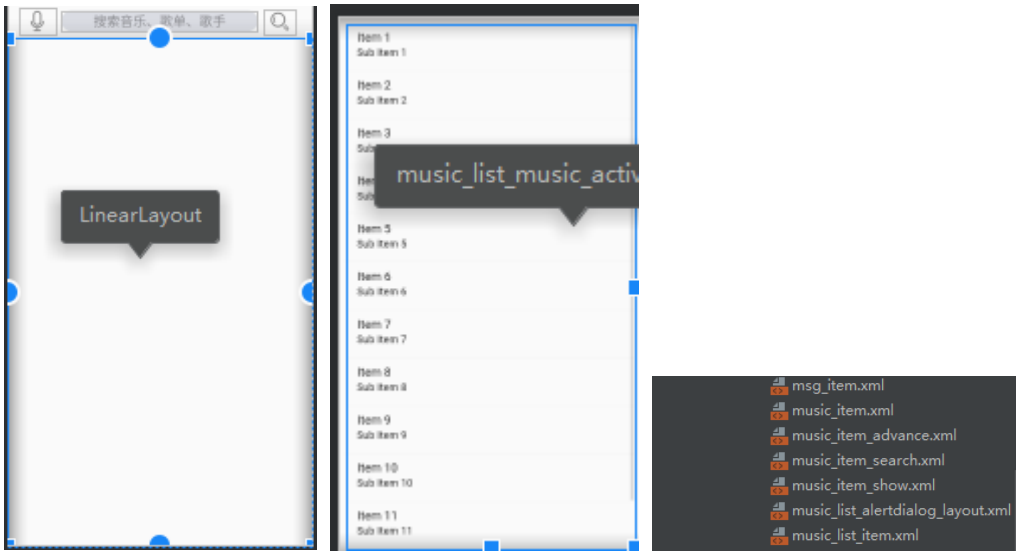
```
            }
            String res = json.getString("res");

            switch (res) {
                case "false":
                    Toast.makeText(super.getActivity().get(), "出错了",
Toast.LENGTH_LONG).show();
                    break;
                case HttpTask.CONNECT_OR_READ_TIMEOUT:
                    Toast.makeText(super.getActivity().get(), "网络开小
差儿啦~~~", Toast.LENGTH_LONG).show();
                    break;
                case "true":
                    if(data == null){
                        Toast.makeText(super.getActivity().get(), "服务
器错误，无法获得歌单", Toast.LENGTH_LONG).show();
                        break;
                    }
                    List<Music> musics = new ArrayList<>();
                    for (int i = 0;i<dataArr.length();i++){
                        JSONObject _temp = dataArr.getJSONObject(i);

                        Date time = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss").parse(_temp.getString("time"));
                        musics.add(
                                new
Music(_temp.getInt("music_id"),_temp.getString("name"),

_temp.getString("url"),_temp.getString("singer"),
                                        time,_temp.getInt("status"),

_temp.getString("u_phone"),false,0)
                        );
                    }
                    //通过获得到的 musics 更新 searchActivity
                    activity.get().initListView(musics);
                    break;
            }
        }catch (JSONException e){
            e.printStackTrace();
        } catch (ParseException e) {
            e.printStackTrace();
        }
```

```
    }
}
```

同样的 GetMusicHandler 也基本一样，只是用在 MusicActivity 的更新。。

3.3.2 布局

这几个是都是使用 LinearLayout，item 有一个布局，通过 adapter 生成 item。主要是 item 的布局有些不一样，歌单要要显示图片，歌单名，歌单创建时间，歌曲在列表因为有管理功能所以后面会有一个删除按钮，而搜索页面的歌曲项不提供删除按钮。



3.4功能信息页面

3.4.1 Code：

OnCreateView 方法，在这个方法当中，重点对 updata_lo_functions 和 upload_music_functions 同时添加约束条件，保证无论点到 layout 还是 button 上都可以进行跳转至上传界面。

```
public View onCreateView(@NonNull LayoutInflater inflater,
                         ViewGroup container, Bundle savedInstanceState)
{
    Log.e("a",String.valueOf(count));
```

```
    View root = inflater.inflate(R.layout.fragment_functions,
container, false);

root.findViewById(R.id.updata_lo_functions).setOnClickListener(new
View.OnClickListener(){
        public void onClick(View v){
            Intent intent=new Intent(getActivity(),
UploadActivity.class);
            startActivity(intent);
        }
    });


root.findViewById(R.id.upload_music_functions).setOnClickListener(new
View.OnClickListener(){
        public void onClick(View v){
            Intent intent=new Intent(getActivity(),
UploadActivity.class);
            startActivity(intent);
        }
    });

    root.findViewById(R.id.search_img_search).setOnClickListener(new
View.OnClickListener(){
        public void onClick(View v){
            Intent intent=new Intent(getActivity(),
SearchActivity.class);
            EditText editText =
root.findViewById(R.id.search_text_search);
            intent.putExtra("key",editText.getText().toString());
            intent.putExtra("type","name");
            startActivity(intent);
        }
    });
    return root;
}
```

3.4.2 布局：

使用 constraintlayout 布局，各个控件位置如下图

3.4.3 上传界面 Code

重点方法 pickFile，跳转至 Os 的文件界面，并且完成时产生回调。

```
public void pickFile() {
    Intent intent = new Intent(Intent.ACTION_GET_CONTENT);
    intent.setType("*/*"); // 代表所有文件类型
    //可选择性打开的文件类型设置
    intent.addCategory(Intent.CATEGORY_OPENABLE);
    //设置回传结果
    this.startActivityForResult(intent, 1);
}
```

重点方法重写回调函数，当文件界面返回时拿到选取的文件的 Uri，通过 Util 下的方法来获取 path，最后使用 Util 的上传方法上传文件。

```
protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
    super.onActivityResult(requestCode,resultCode,data);
    if (resultCode == Activity.RESULT_OK) {//是否选择，没选择就不会继续
        Uri uri = data.getData();//得到 uri，后面就是将 uri 转化成 file 的过
程。
        music_path = UriToPathUtil.getImageAbsolutePath(this, uri);
        Log.e("path",music_path);

        //*
        music_n = music_name.getText().toString();
```

```
        singer_n = singer_name.getText().toString();

        //*/

        map.put("name", music_n);
        map.put("singer", singer_n);
        map.put("status", public_status);
        map.put("u_phone", ConstUtil.user.getPhone());

        try {
            new Thread(new Runnable() {
                public void run(){
                    try {

HttpRe.uploadFile(music_path,map,ConstUtil.ADD_MUSIC_URL);

                    } catch (Exception e) {
                        Log.e("uploadinth","wrong!!!");
                    }
                }
            }).start();


            onBackPressed();
        } catch (Exception e) {
            Log.e("upload","wrong!!!");
        }
    }
}
```

重点方法 onCreate，给选取的状态添加背景，达到交互的作用。以及给上传调用添加约束。

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_upload);


    View return_button = this.findViewById(R.id.return_upload);
    public_button = this.findViewById(R.id.state_public_upload);
    unpublic_button = this.findViewById(R.id.state_unpublic_upload);
    final Drawable d =  public_button.getBackground();
    music_name = this.findViewById(R.id.text_name_music_upload);
    singer_name = this.findViewById(R.id.text_name_singer_upload);
```

```
    upload_button = this.findViewById(R.id.upload_music_upload);
    warning_text = this.findViewById(R.id.warning_text_upload);


    public_status = -1;


    public_button.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v){
            public_status = 2;
            public_button.setBackgroundColor(Color.rgb(    192,255,
62));
            unpublic_button.setBackground(d);
        }
    });
    unpublic_button.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v){
            public_status = 1;
            unpublic_button.setBackgroundColor(Color.rgb(  192,255,
62));
            public_button.setBackground(d);
        }
    });

    upload_button.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v){
            music_n = music_name.getText().toString();
            singer_n = singer_name.getText().toString();
            if(music_n.equals("") || singer_n.equals("")||
public_status == -1){
                warning_text.setVisibility(View.VISIBLE);
                return;
            }
            pickFile();

        }
    });

    return_button.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v){
            onBackPressed();
        }
    });
}
```

3.4.4 上传界面布局

使用 constraintlayout 布局，各个控件位置如下图

3.5 个人信息页面

3.5.1 Code

updateShowMessage() 加载用户信息，每次返回或者跳转到这个页面时，都会重新加载则会个方法

```
private void updateShowMessage(){
    Log.e("a", ConstUtil.user.getUsername());
    this.user_name_my_info.setText(ConstUtil.user.getUsername());
    this.user_id_my_info.setText(ConstUtil.user.getPhone());

    this.imagePath = ConstUtil.user.getImage();
    //获取头像
    if(ConstUtil.user.getImage()!=null){
        ImageView imageView = root.findViewById(R.id.user_image_my_info);
        new LoadImagesTask(imageView).execute(ConstUtil.user.getImage());
    }
}
```

重写 onActivityResult 方法

```
@Override
    public void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {

//        new MyThread().run();
```

```
        updateShowMessage();

        super.onActivityResult(requestCode, resultCode, data);

    }
```

在初始化方法 onCreateView()中动态监听 *set_info* 和 *all_upload_music* 按钮，分别对应跳转到 AlterActivity、AllMusicList 页面

```
public View onCreateView(@NonNull LayoutInflater inflater,
                         ViewGroup container, Bundle savedInstanceState) {

    root = inflater.inflate(R.layout.fragment_my_information, container, false);
    user_name_my_info = root.findViewById(R.id.user_name_my_info);
    user_id_my_info = root.findViewById(R.id.user_id_my_info);
    user_image_my_info = root.findViewById(R.id.user_image_my_info);
    this.updateShowMessage();

    root.findViewById(R.id.set_info).setOnClickListener(new View.OnClickListener(){
        public void onClick(View v){
            Intent intent=new Intent(getActivity(), AlterActivity.class);
            startActivityForResult(intent, 1);
        }
    });

    root.findViewById(R.id.all_upload_music).setOnClickListener(new View.OnClickListener(){
        public void onClick(View v){
            Intent intent=new Intent(getActivity(), MusicActivity.class);
            Bundle bundle = new Bundle();
            bundle.putSerializable("musiclist",ConstUtil.musicLists.get(0));
            intent.putExtras(bundle);
            startActivityForResult(intent, 1);
        }
    });

    return root;
}
```

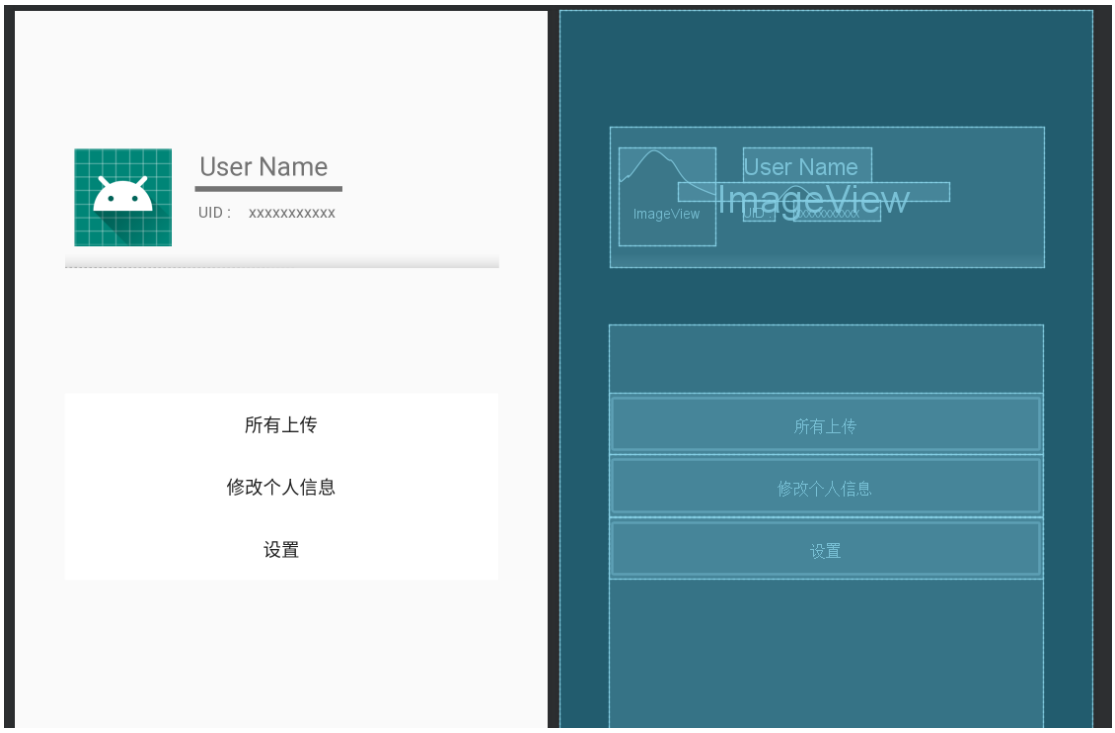
对头像框圆角的设计和布局，使用画布工具

```
private Bitmap bimapRound(Bitmap mBitmap,float index){
    Bitmap bitmap = Bitmap.createBitmap(mBitmap.getWidth(), mBitmap.getHeight(),
Bitmap.Config.ARGB_4444);

    Canvas canvas = new Canvas(bitmap);
    Paint paint = new Paint();
```

```
    paint.setAntiAlias(true);

    //设置矩形大小
    Rect rect = new Rect(0,0,mBitmap.getWidth(),mBitmap.getHeight());
    RectF rectf = new RectF(rect);

    // 相当于清屏
    canvas.drawARGB(0, 0, 0, 0);
    //画圆角
    canvas.drawRoundRect(rectf, index, index, paint);
    // 取两层绘制，显示上层
    paint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.SRC_IN));

    // 把原生的图片放到这个画布上，使之带有画布的效果
    canvas.drawBitmap(mBitmap, rect, rect, paint);
    return bitmap;

}
```

3.5.2 布局：

使用 constraintlayout 布局，各个控件位置如右下图



3.5.3 code

“修改个人信息”跳转至 AlterActivity
“所有上传”跳转至 AllMusicList （复用”我的上传”歌单列表）

修改个人信息页面---AlterActivity
Code:
onCreate 阶段初始化整个 activity 获取 image_button 和 info_button 并动态监听
image_button:对应着 pickFile()方法选择文件 并上传为头像;
info_button:对应着 new HttpTask 向服务器发送一个修改个人信息的请求

```
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_alter);

        View return_button = this.findViewById(R.id.return_upload);
        image_button = this.findViewById(R.id.alter_image);
        info_button = this.findViewById(R.id.alter_info_button);
        Log.e("a", info_button.toString());

        username = this.findViewById(R.id.text_name_music_upload);
        password = this.findViewById(R.id.text_name_singer_upload);
        warning_text = this.findViewById(R.id.warning_text_upload);

        image_button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v){
                pickFile();
            }
        });
        info_button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v){
                name = username.getText().toString();
                pass = password.getText().toString();
                if(name.equals("") || pass.equals("")){
                    warning_text.setVisibility(View.VISIBLE);
                }else{
                    Log.e("a", StringUtil.transformToAlter(name, ConstUtil.user.getPhone(),
pass));
                    new HttpTask(ConstUtil.ALTERINFOURL, new
AlterHttpHandler(AlterActivity.this,
R.id.alter_info_button)).execute(StringUtil.transformToAlter(name,
ConstUtil.user.getPhone(), pass));
                }
```

```
            }
        });
        return_button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v){
                onBackPressed();
            }
        });
    }
```

pickFile()打开文件选择器，并且设置回传结果

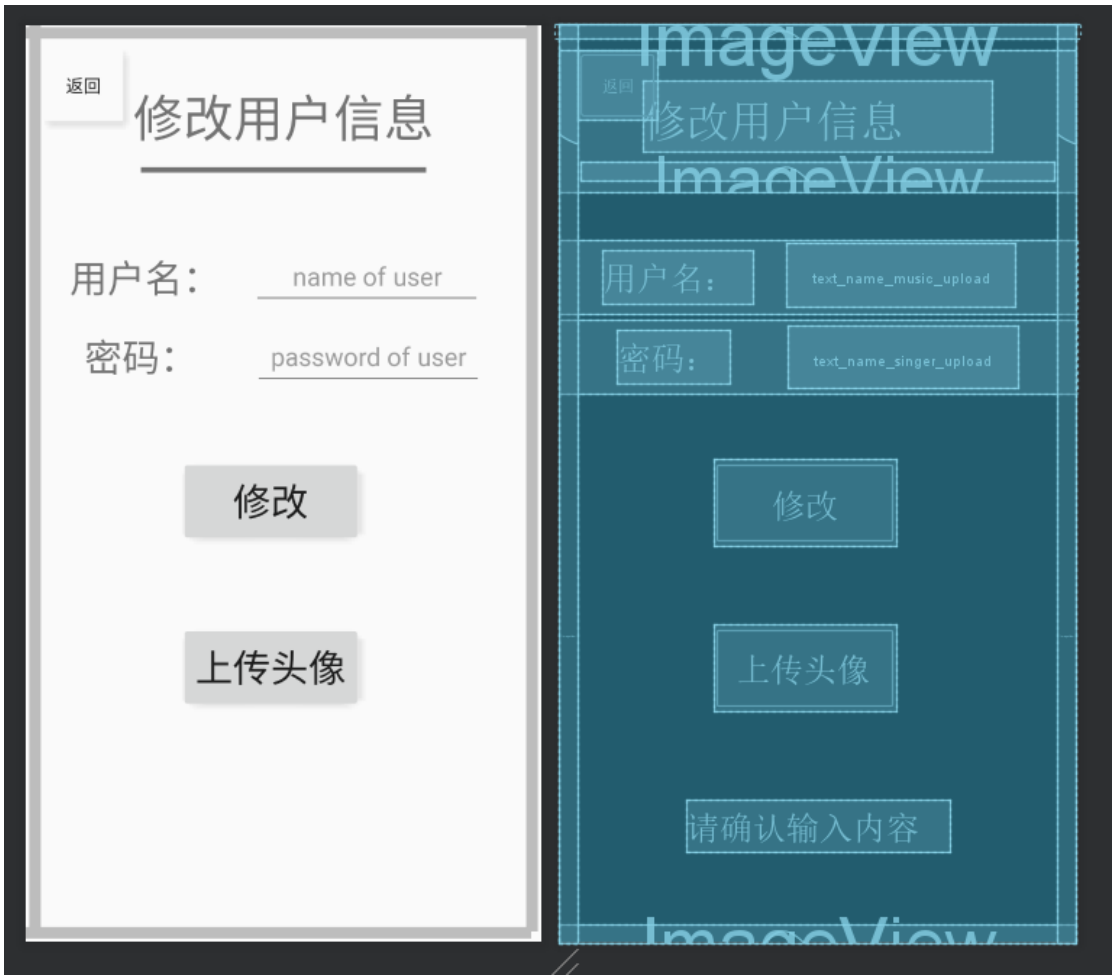
```
public void pickFile() {
    Intent intent = new Intent(Intent.ACTION_GET_CONTENT);
    intent.setType("*/*"); // 代表所有文件类型
    //可选择性打开的文件类型设置
    intent.addCategory(Intent.CATEGORY_OPENABLE);
    //设置回传结果
    this.startActivityForResult(intent, 1);
}
```

重写 onActivityResult()接受回传结果，为了保持歌曲播放的 fragment 不销毁，我们修改了 navigation 的方法，使得 fragment 切换不会重新运行 oncreateview 方法，而我们修改信息后需要重新刷新的，因此我们通过该回调函数完成信息的刷新。

```
@Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (resultCode == Activity.RESULT_OK) {//是否选择，没选择就不会继续
            Uri uri = data.getData();//得到uri，后面就是将uri转化成file的过程。
            image_path = UriToPathUtil.getImageAbsolutePath(this, uri);
            map.put("phone", ConstUtil.user.getPhone());
            try {
                new Thread(new Runnable() {
                    public void run() {
                        try {
                            HttpRe.uploadFile(image_path, map, ConstUtil.ALTER_IMAGEURL);
                        } catch (Exception e) {
                            Log.e("uploadinth","wrong!!!");
                        }
                    }
                }).start();
            } catch (Exception e) {
                Log.e("upload","wrong!!!");
            }
        }
    }
```

3.5.4 布局

使用 constraintlayout 布局，各个控件位置如右下图



3.6 制作工具类

3.6.1 HttpRe

用于上传文件
使用 java 模仿网页，post 请求上传文件。用于上传歌曲模块，头像上传。

```
public static void uploadFile(String fileName, Map<String, Object>
map,String link) throws Exception {
        // 换行符
        final String newLine = "\r\n";
```

```
        final String boundaryPrefix = "--";
        // 定义数据分隔线
        String BOUNDARY = "========7d4a6d158c9";
        // 服务器的域名
        URL url = new URL(link);
        HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
        // 设置为 POST 情
        conn.setRequestMethod("POST");
        // 发送 POST 请求必须设置如下两行
        conn.setDoOutput(true);
        conn.setDoInput(true);
        conn.setUseCaches(false);
        // 设置请求头参数
        conn.setRequestProperty("Connection", "Keep-Alive");
        conn.setRequestProperty("Charset", "UTF-8");
        conn.setRequestProperty("Content-Type", "multipart/form-data;
boundary=" + BOUNDARY);
        try (
                OutputStream outputStream = conn.getOutputStream();
                DataOutputStream out = new
DataOutputStream(outputStream);
        ) {
            //传递参数

            if (map != null) {
                StringBuilder stringBuilder = new StringBuilder();
                for (Map.Entry<String, Object> entry : map.entrySet()) {
                    stringBuilder.append(boundaryPrefix)
                            .append(BOUNDARY)
                            .append(newLine)
                            .append("Content-Disposition:form-data;
name=\"")
                            .append(entry.getKey())
                            .append("\"").append(newLine).append(newLine
)
                            .append(String.valueOf(entry.getValue()))
                            .append(newLine);
                }

out.write(stringBuilder.toString().getBytes(Charset.forName("UTF-8")));
            }

            // 上传文件
```

```
            {
                File file = new File(fileName);
                StringBuilder sb = new StringBuilder();
                sb.append(boundaryPrefix);
                sb.append(BOUNDARY);
                sb.append(newLine);
                sb.append("Content-Disposition: form-
data;name=\"file\";filename=\"").append(fileName)
                        .append("\"").append(newLine);
                sb.append("Content-Type:application/octet-stream");
                sb.append(newLine);
                sb.append(newLine);
                out.write(sb.toString().getBytes());

                try (
                        DataInputStream in = new DataInputStream(new
FileInputStream(file));
                ) {
                    byte[] bufferOut = new byte[1024];
                    int bytes = 0;
                    while ((bytes = in.read(bufferOut)) != -1) {
                        out.write(bufferOut, 0, bytes);
                        Log.e("wenjian","dudaole");
                    }
                    out.write(newLine.getBytes());
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }

            // 定义最后数据分隔线，即--加上 BOUNDARY 再加上--。
            byte[] end_data = (newLine + boundaryPrefix + BOUNDARY +
boundaryPrefix + newLine)
                    .getBytes();
            // 写上结尾标识
            out.write(end_data);
            out.flush();

        } catch (Exception e) {
            e.printStackTrace();
        }

        //定义 BufferedReader 输入流来读取 URL 的响应
        try (
```

```
                InputStream inputStream = conn.getInputStream();
                InputStreamReader inputStreamReader = new
InputStreamReader(inputStream);
                BufferedReader reader = new
BufferedReader(inputStreamReader);
        ) {
            String line = null;
            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

3.6.2 ConstUtil

用于存放该应用的 app 常量，如 url 等。各个模块都使用。

```
public static final String SERVER_URL = "http://47.94.160.152:8000";
public static final String LOGINURL = SERVER_URL+"/user/login/";
public static final String REGISTERURL = SERVER_URL+"/user/addUser/";
public static final String MUSICLISTURL =
SERVER_URL+"/list/findAllByPhone/";

public static final String GET_MUSIC_BY_LIST =
SERVER_URL+"/music/findSongByList";
```

3.6.3 StringUtil

用于对字符串转化，如转化为 url 格式，转化为 json 格式，判断是否为手机号等。下面是部分代码。在多个模块使用，如登入注册等需要格式转化的模块。

```
/**
 * 将指定的字符串传化为时间格式:
 * 2019-09-01 01:01
 * @param time 1
 * @return 1
 */
public static String transformToDate(int...time) {
```

```
    if (time.length < 5)
        return "";

    String[] s = new String[5];
    for (int i = 0; i <time.length; i++){
        s[i]= String.valueOf(time[i]);
    }

    StringBuilder sb = new StringBuilder();
    for (int i = 1; i < s.length; i++) {
        if (s[i].length() == 1)
            s[i] = "0".concat(s[i]);
    }
    sb.append(s[0].concat("-"));
    sb.append(s[1].concat("-"));
    sb.append(s[2].concat(" "));;
    sb.append(s[3].concat(":"));;
    sb.append(s[4]);

    return sb.toString();
}

public static String transformToLogin(String  phone,String password){
    return "phone="+phone+"&"+"password="+password;
}
public static String transformToRegister(String  username,String
phone,String password){
    return transformToLogin(phone,password)+"&"+"username="+username;
}
public static String transformToAlter(String  username,String
phone,String password){
    return transformToLogin(phone,password)+"&"+"username="+username;
}

public static String transformToPOST(Map<String,String> values){
    StringBuffer stringBuffer = new StringBuffer();

    for (Map.Entry<String,String> _t : values.entrySet()){
        stringBuffer.append("&"+_t.getKey()+"="+_t.getValue());
    }
    return stringBuffer.substring(1);
}
```

3.6.4 UriToPathUtil

Uri 转化为文件路径。读取文件上传模块需要使用。

```
public static String getRealFilePath(Context context, final Uri uri) {
    if (null == uri)
        return null;
    final String scheme = uri.getScheme();
    String data = null;
    if (scheme == null)
        data = uri.getPath();
    else if (ContentResolver.SCHEME_FILE.equals(scheme)) {
        data = uri.getPath();
    } else if (ContentResolver.SCHEME_CONTENT.equals(scheme)) {
        Cursor cursor = context.getContentResolver().query(uri, new
String[] { MediaStore.Images.ImageColumns.DATA }, null, null, null);
        if (null != cursor) {
            if (cursor.moveToFirst()) {
                int index =
cursor.getColumnIndex(MediaStore.Images.ImageColumns.DATA);
                if (index > -1) {
                    data = cursor.getString(index);
                }

            }
            cursor.close();
        }
        if (data == null) {
            data = getImageAbsolutePath(context, uri);
        }

    }
    return data;
}

public static Uri getUri(final String filePath) {
    return Uri.fromFile(new File(filePath));
}
```

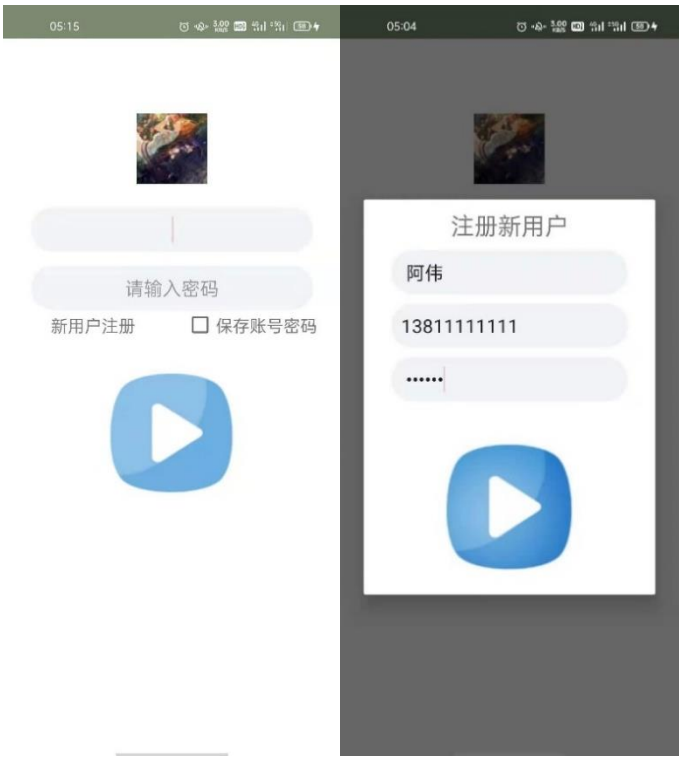
4. 运行测试

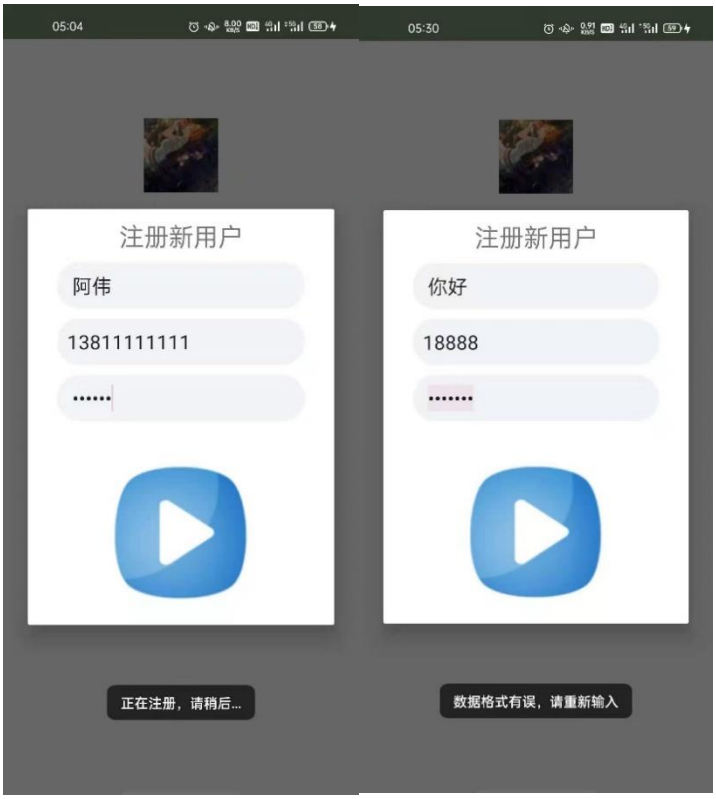
4.1 登录及注册测试

4.1.1 注册模块：

操作：点击“新用户注册“---填写新用户信息后

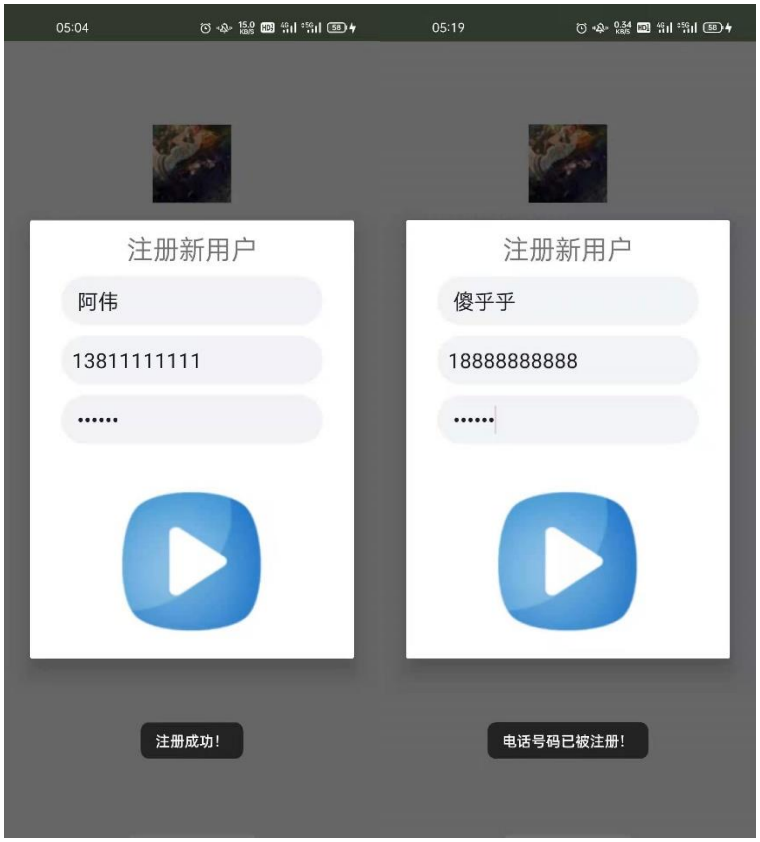
效果：在输入都合法时，会弹出”正在注册，请稍后”，否则弹出“数据格式有误，请重新输入”





过程：输入合法---“等待注册”---注册结果

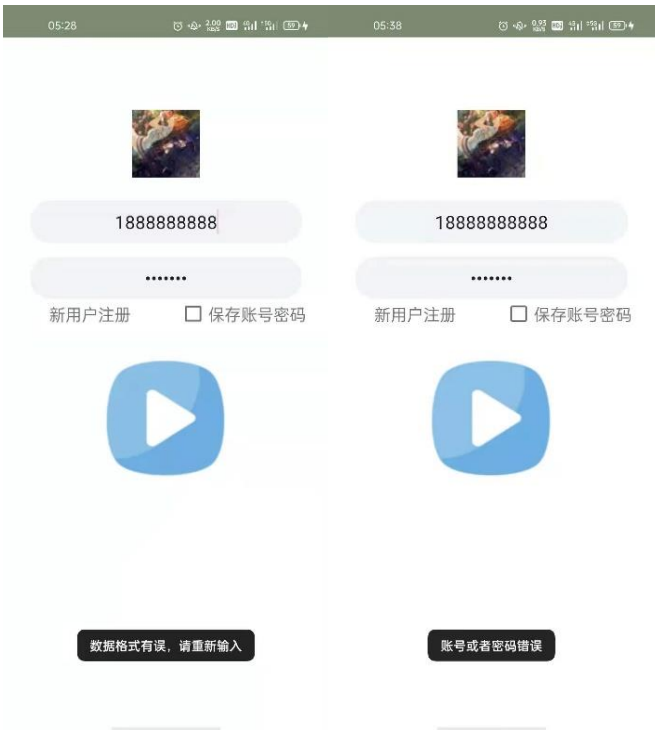
效果：如果电话号码没有被注册过则弹出“注册成功”;否则弹出“电话号码已被注册



4.1.2 登录模块：

操作：输入手机号码和密码

效果：如果输入不合法，则弹出“数据格式有误，请重新输入”；若合法则会判断手机号与密码是否匹配，若不匹配会弹出“账号或者密码错误“



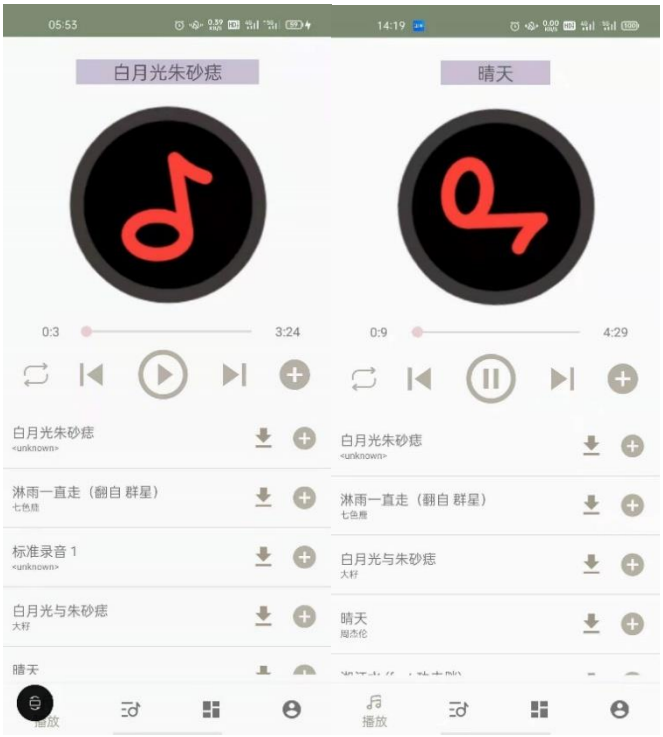
效果：在账号密码相互匹配下，弹出“登录成功”，再判断该用户是管理员还是普通用户，若是普通用户则跳转到“播放”页面；若是管理员会跳转到“音乐管理员”页面

4.2 音乐播放测试

4.2.1 暂停播放模块：

操作：点击暂停/播放按钮

效果：点击暂停则停止播放音乐，点击播放则开始播放音乐“



4.2.2 切歌循环模式模块：

操作：点击切换循环模式按钮、下一首/上一首按钮

效果：点击切换循环模式按钮可以切换循环模式，点击下一首/上一首按钮可以切换到下一首/上一首歌曲并播放

4.2.3 进度条模块：

操作：点击/拖拽进度条到指定位置

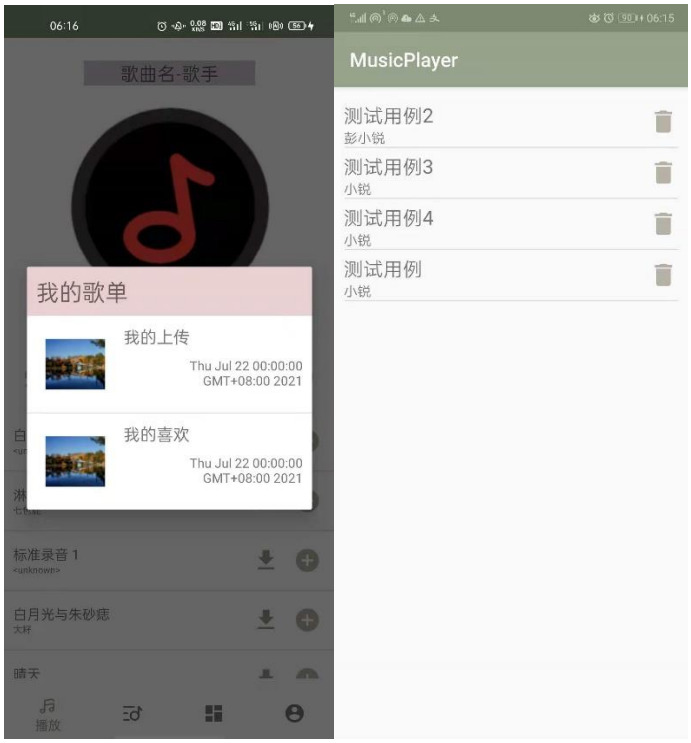
效果：点击/拖拽进度条到指定位置可以将歌曲前进/后退到指定位置进行播放

4.3 歌单管理测试

4.3.1 添加至歌单模块：

操作：点击“ + ”号将指定歌曲加入到指定歌单

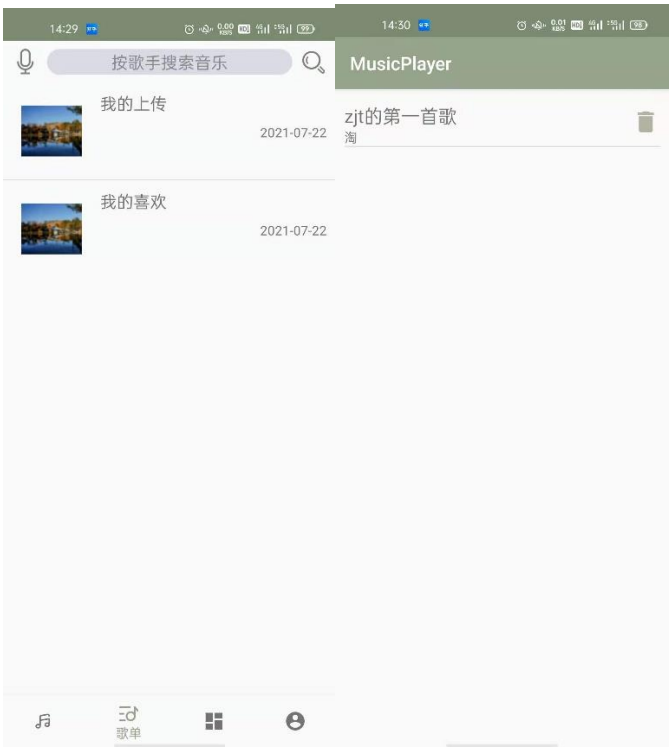
效果：被添加歌曲会出现在指定歌单



4.3.2 歌单查看模块：

操作：点击并进入指定歌单后

效果：展示该歌单下所有歌曲



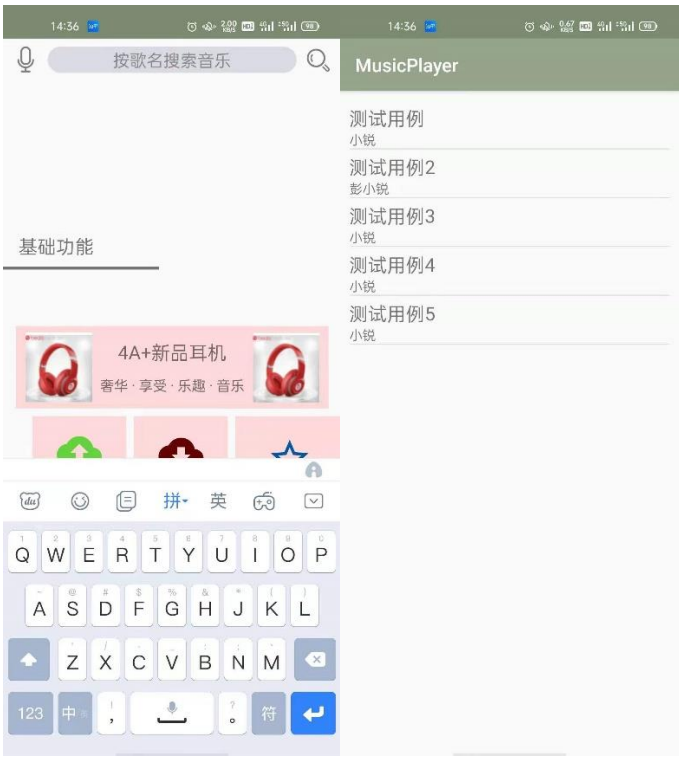
4.3.3 歌单管理模块：

操作：点击右侧删除按钮
效果：被删除的歌曲在该歌单消失

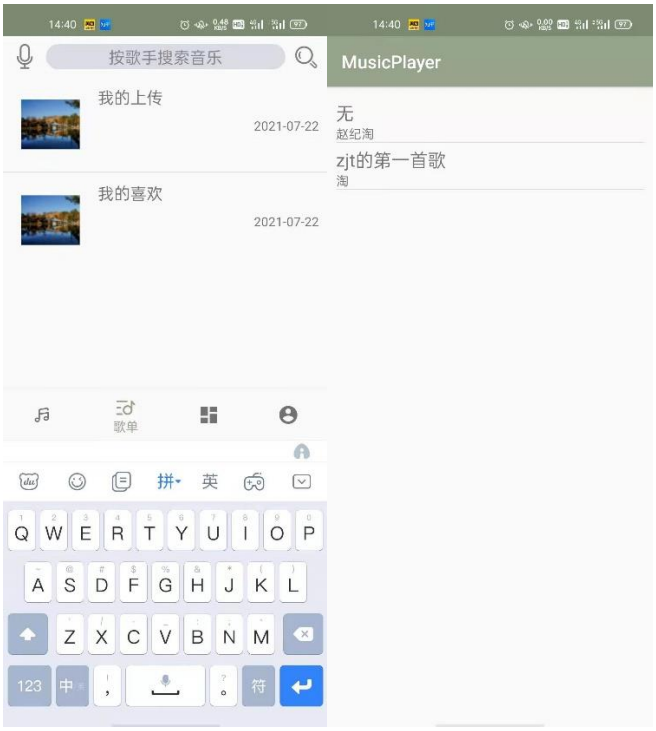
4.4 功能页面测试

4.4.1 搜索模块：

操作：在“按歌名搜索歌曲”框输入要模糊搜索的文本
效果：展示歌名中含有该文本的歌曲



操作：在“按歌手搜索歌曲”框输入要模糊搜索的文本
效果：展示歌手名中含有该文本的歌曲



4.4.2 上传歌曲模块：

操作：点击“上传歌曲”将指定歌曲上传到“我的上传“歌单
效果：上传的歌曲会出现在我的上传歌单以及所有上传列表


14:43
按歌名搜索音乐
基础功能
4A+新品耳机
奢华 · 享受 · 乐趣 · 音乐
上传歌曲
下载歌曲
开通VIP
曲库全搜索
功能
06:13
MusicPlayer
返回
上传歌曲
歌曲名：
name of music
歌手名：
name of singer
公开
私密
上传


MusicPlayer
返回
上传歌曲
歌曲名：
测试用例5
歌手名：
小锐
公开
私密
上传
MusicPlayer
测试用例2
彭小锐
测试用例
小锐
测试用例4
小锐
测试用例5
小锐
测试用例3
小锐

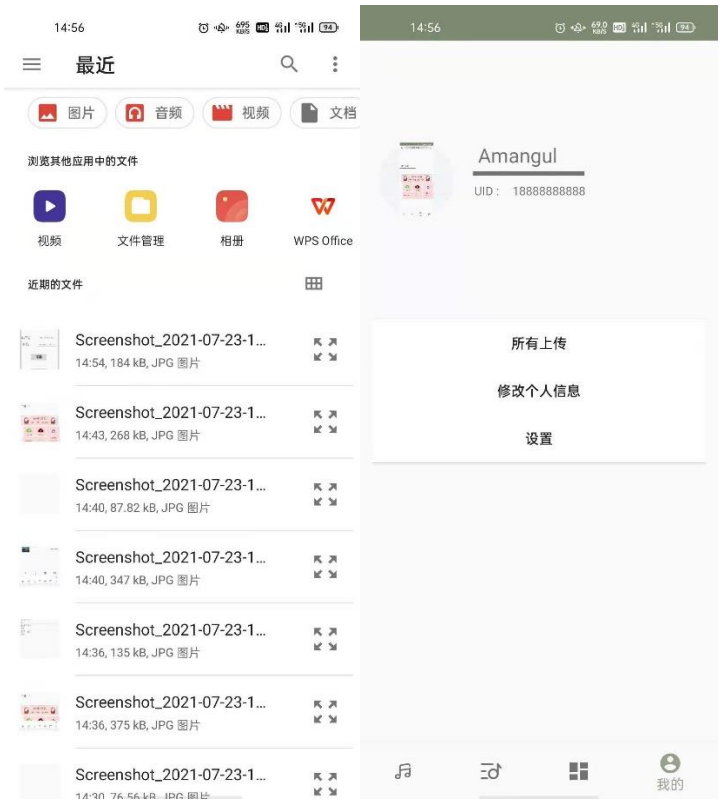
4.5 我的页面测试

4.5.1 修改个人信息模块：

操作：点击“修改个人信息“---更改完用户信息后---返回“我的“页面
效果：用户名区域会刷新为更改后的用户名

4.5.2 更改头像模块：

操作：点击“所有上传 “---跳转“我的上传”歌单页面
效果：显示所有已经上传的歌曲



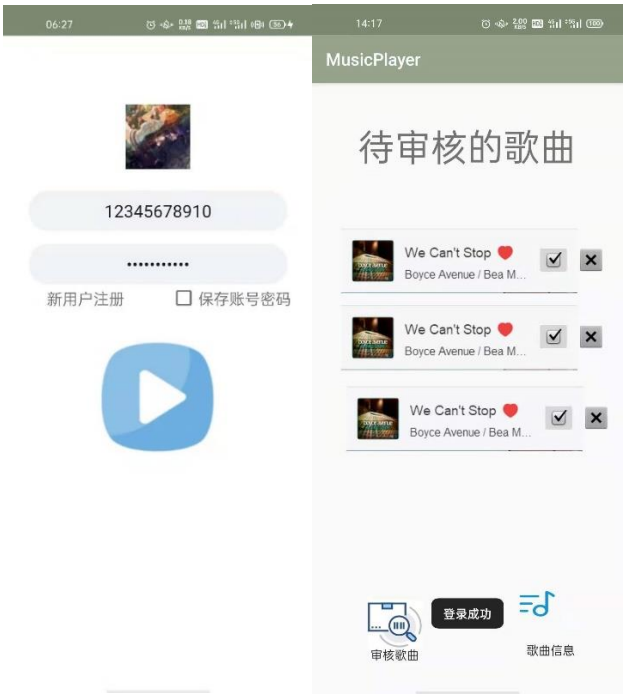
4.5.3 所有上传模块：

操作：点击“所有上传 “---跳转“我的上传”歌单页面
效果：显示所有已经上传的歌曲

4.6 音乐管理员页面测试

操作：输入管理员账户信息(手机号：12345678910；密码：12345678910)

效果：进入音乐管理员平台



5. 总结及展望

5.1 彭小锐

通过这周的移动应用开发，我学习到了很多东西，首先是加深我对 Android 系统的移动设备的应用系统开发方法、开发过程、开发技术、开发工具的了解和掌握。通过制作一个音乐系统，参与了从应用需求调研、分析、设计与实现过程。对 Android 开发有了完整的认识，掌握了 Android 程序的开发。

我完成了登入注册，音乐播放板块的功能，和歌单管理的功能等模块，但是在开发过程中，我遇到了许多的困难，但是通过不断的查资料，看官方文档逐渐克服，虽然花了很多的时间，但是学习到了很多的东西，特别是学习到 Android 平台底层技术原理及架构比如 Handler，Message 机制，异步类等，最终也完成了想要的功能。

5.2 董俊豪

为期一周的移动应用开发实践结束了，由于之前没有选过安卓移动开发的课程，所以在这周的前一两天，通过学习 Android 的控件、布局、Activity、Service 等一系列基础知识，对整个 Android 的开发有了大致的了解。例如：要的布局（或者控件），在学习界面中，我发现 Android 为我们提供了很好的类似反射机制，通过 Layout 文件夹下的配置文件，可以快速的形成界面，在配置文件可以设置属性或者样式都是很快捷方便。还有对一些点击、选中、按键等处理的事件，界面之间的跳转 Intent 管理，通过 Bundle 对数据在界面之间进行传输。另外，也学习了关于 spring boot 后端开发结合数据库以及部署到服务器的网络开发体系知识。

这次实践我不仅仅学到了很多基础知识，在整个过程中，感受到在开发前期，对整体系统的架构、设计以及分工是万分重要的，因为臃肿冗余的体系和不明确的分工会导致开发后期的模块集成环节会经常出现问题。而且我们小组每个成员为了把我们的项目做到最好都付出了极大努力，连续熬了两个通宵，尽最大可能在这短短几天做出比较完美成型的 APP。同时，我相信这些宝贵经验会成为我今后工作中的重要基石！

5.3 刘天

安卓编程和我原来想的架构编程是有很大差别的。我原来会认为安卓的架构会让我们非常容易上手，而且能够很好的开发。

但是，很显然，我预估错误了。

是的，他确实是要比我们直接从 java 上到 android 简单，因为很多东西 android studio 帮我们集成了。但也带来了更多的问题。

复杂的架构会导致我在使用很多组件的时候都是知其然不知其所以然。没有办法很好的掌控。有的时候代码出错了，查了半天是因为我使用的那个方法在这个组件当中不成立。这

对我影响很大。我也通过这次编程学习了非常多的 APP。

5.4 赵纪淘

作为本组的组长，我认为在本次我从安卓开发当中学到的不仅仅是一个开发的知识，更有很多项目控制、面对新问题如何解决的问题。

在周一开始了解安卓之后，我的心就凉了半截。因为整个框架环境和我之前所认知的环境有着很多很多的不同。架构写代码，更方便，但约束也更多。眼花缭乱的界面让我不知道该如何去下手。

预先想好的进程速度因为一个有一个看上去极其不合理的机制而打破。在一天半夜，仍然被 permission 问题困住的我不禁对安卓环境产生了厌烦。单当现在再回首时，跳出来反而却看到了很多安卓编程的优良。

给我的另外一个触动是：技术难点的攻克是要比开发进度要重要的。在整个开发的前 3 天，也就是一直到周四中午左右吧，进度一直都非常慢。当时做完的内容差不多是最后成品的一半。但最后一天的开发速度着实让我感到震惊。当然，这也是可以遇到的。因为主要的技术难点已经被攻克了。