项目实践——本地音乐播放器

一、 设计背景

随着智能手机的普及，人们对于音乐的需求也日益增长。安卓系统作为全球最流行的移动操作系统之一，拥有庞大的用户群体。安卓本地音乐播放器可以提供更好的用户体验。用户可以直接在手机上管理和播放自己的音乐库，无需依赖第三方音乐平台，具有更高的自主性和灵活性。其次，安卓本地音乐播放器可以更好地保护用户的隐私。用户无需将自己的音乐数据上传到云端，减少了数据泄露的风险。此外，开发本地音乐播放器还可以为开发者提供更多的创新空间。例如，通过添加独特的功能和界面设计，开发者可以打造出与众不同的音乐播放器，提升产品的竞争力。综上所述，进行安卓开发本地音乐播放器是为了满足用户对于更好的音乐体验、隐私保护和个性化需求的追求。

二、 需求分析

(一)登陆界面

1. 功能点

1) 用户隐私权限

需用户同意《服务条款》和《隐私协议》，其中条款和协议突出显示且可以点击，用户同意后方可进行账户登录。

2) 用户登录

在用户输入用户名和密码后，可点击登录按钮进行登录。本地存储用户信息。

2. 涉及技术点

基础布局，基础 UI，用户信息存储验证，数据库 SQLite（定义表，用户数据，用户喜好——点赞，收藏）。

(二)音乐播放详情页

1. 功能点

1) 歌曲暂停/播放，下一首，上一首

若当前音乐正在播放中，则当用户点击暂停后，歌曲停止播放，同时按键转换为开始播放按钮；若当前无音乐播放，则当用户点击播放按钮后，音乐继续播放，同时按键转换为暂停按钮。当用户点击下一首/上一首按键时，暂停当前正在

播放的音乐，转为播放当前列表中的下一首/上一首。若当前音乐播放完毕，自动播放下一首。

涉及音频播放 MediaPlayer: 首先需要创建出一个 MediaPlayer 对象，然后调用 setDataSource()方法来设置音频文件的路径，再调用 prepare()方法使 MediaPlayer 进人到准备状态，接下来调用 start()方法就可以开始播放音频，调用 pause()方法就会暂停播放，调用 reset()方法就会停止播放。申请 WRITE_EXTERNAL_STORAGE 权限，用于访问 SD 卡。

2) 进度拖拽

当用户拖拽正在播放的音乐的进度条后，从用户拖拽的停止处开始播放音乐。涉及拖动条 SeekBar 长度，时间设定。

3) 音乐点赞、收藏

当用户点击点赞/收藏时，可以将音乐加入用户账户下的点赞/收藏列表

2. 涉及技术点

基础布局，基础 UI，按键监听，图标转换，滑动监听，旋转动画，图片加载，用户点赞收藏数据用数据库存储。

(三)主页

1. 功能点

1) 顶部显示搜索和个人界面跳转按钮

当用户点击搜索按钮时，跳转到搜索界面。当用户点击个人界面按钮时，显示用户的账户信息。

2) 底部显示最近播放的歌曲

底部显示最近播放的歌曲的封面以及暂停/播放按钮，用户可以通过底部按钮实现暂停、播放的功能，同时可以显示当前播放列表。

3) 中部显示本地全部歌单

当用户打开软件时，可以看见本地歌曲的封面、歌名、歌手信息，当用户点击歌曲名称时，跳转到播放界面，进行歌区播放。

2. 涉及技术点

基础布局，基础 UI，数据解析，文本监听，滑动列表，图片加载。

(四)搜索界面

1. 功能点

1) 搜索记录

当用户进入搜索界面时，显示用户搜索历史，若没有搜索历史则不显示。

2) 搜索功能

根据用户输入的信息进行搜索，以列表形式展现搜索结果。

2. 涉及技术点

基础布局，基础 UI，数据存储，数据解析，文本框内容读取文本监听，本地 sd 卡扫描。

(五)个人界面

1. 功能点

1) 显示当前账户信息

若用户已经登录，则显示用户名和头像；若未登录，则显示基础的头像，以及用户登录提示信息。

2) 显示点赞歌曲列表

若用户已经登录，则点击后显示点赞歌单；若未登录，则显示用户登录提示信息。

3) 显示音乐收藏列表

若用户已经登录，则点击后显示收藏歌单；若未登录，则显示用户登录提示信息。

2. 涉及技术点

基础布局，文本监听，DrawerLayout 实现向右滑个人信息栏，歌单跳转。

三、 概要设计

(一)注册、登陆界面

1. 用户登陆

登录界面的 UI 设计如下图 1 所示。左侧为控件设计图，右侧为功能设计图。界面主要展示用户进行用户名和密码输入的范围。同时，涉及用户选择是否记住密码，以及是否同意用户协议。在用户输入框下侧，放置两个按钮，分别用于用户登录，和注册页面的跳转。在界面底部涉及“游客登录”提示信息。

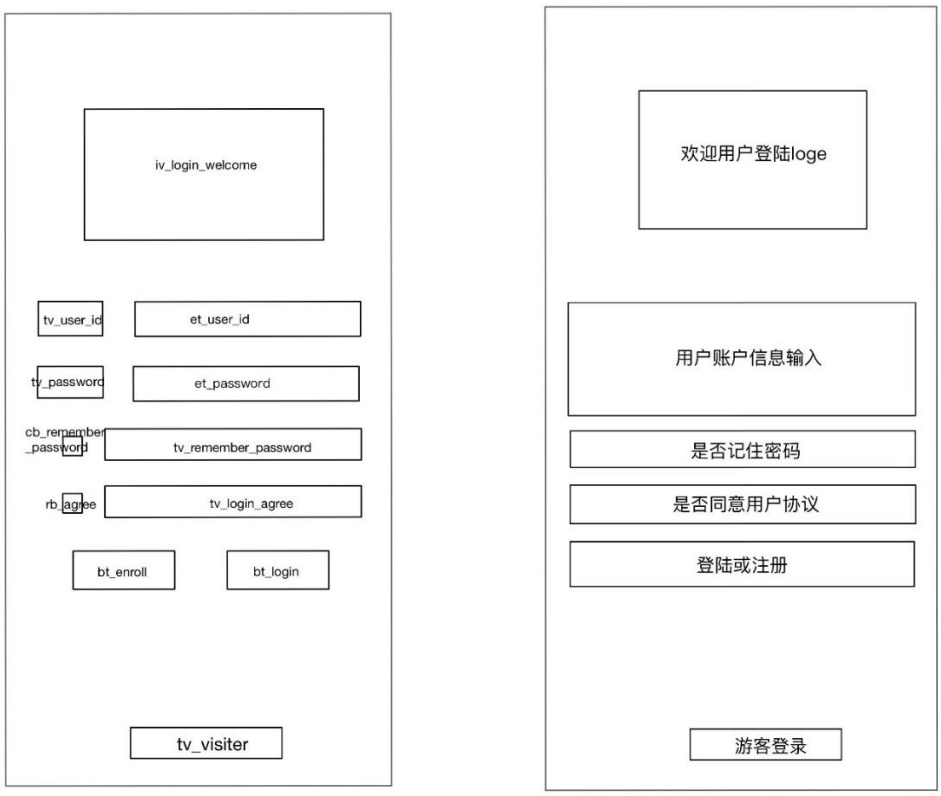

图 1　用户登陆界面概要设计图

2. 用户注册

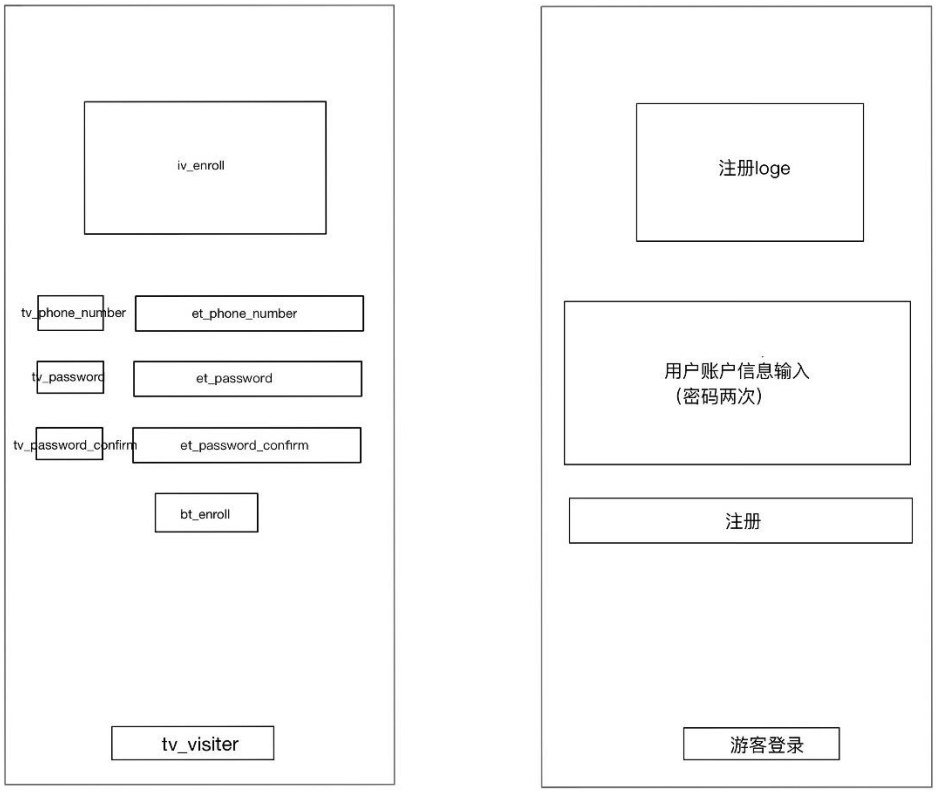

图 2　用户注册界面概要设计图

注册界面的 UI 设计如上图 2 所示。左侧为控件设计图，右侧为功能设计图。界面主要展示用户进行用户名和密码输入的范围。在注册时用户需要输入两次，第二次用于确认密码。在用户输入框下侧，放置一个按钮，用于用户注册页面。若两次密码一致，则会注册成功，跳转至登陆界面。若密码不一致，则会弹出提示信息。在界面底部涉及“游客登录”提示信息。当用户选择游客登录模式时，会以未登陆状态进入主界面。

(二)音乐播放详情页

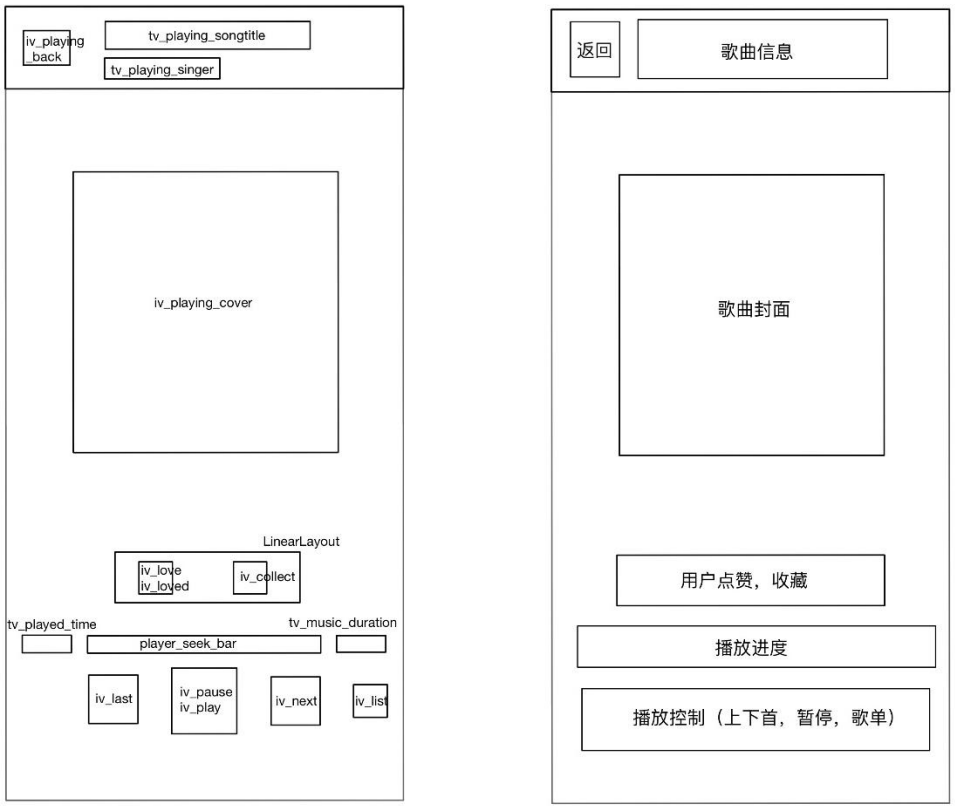

图 3　音乐播放界面概要设计图

音乐播放界面的设计如上图 3 所示。左侧为控件设计图，右侧为功能设计图。播放界面主要显示当前正在播放的音乐的信息，歌名和歌手显示在播放界面的顶部，歌曲的封面显示在中间，以圆形的形式呈现，当音乐播放时，封面跟着旋转；当暂停时，封面动画也暂停。进度条同理，当用户点击进度条某个位置时，播放进度跳转至相应位置。

进度条上方设置点赞、收藏按键，当用户点击后，图标会进行切换。进度条下侧设置上、下一曲切换按键，以及暂停、播放按键，同时设置列表按钮，便于用户查看当前歌单。

(三)主页

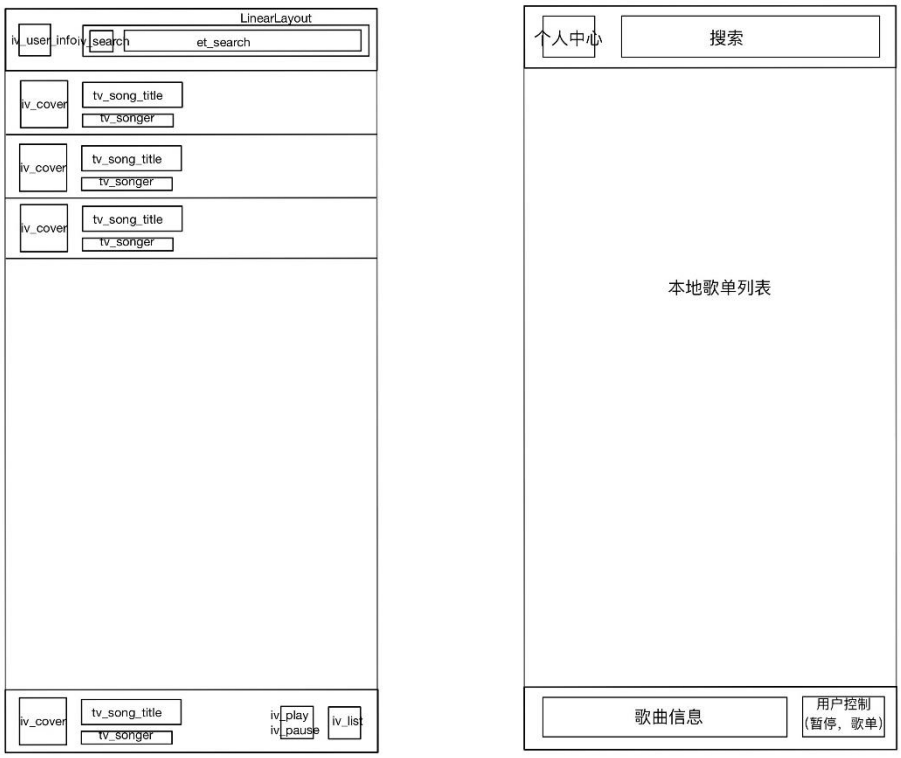

图 4　播放器主界面概要设计图

音乐播放界面的设计如上图 4 所示。左侧为控件设计图，右侧为功能设计图。主页主要用于显示用户信息，便于用户查看个人的歌单。顶部设置搜索框便于用户进行本地搜索。主界面的底部用于显示当前正在播放的歌曲，为当前播放的歌曲设置暂停播放键，以及歌单信息查看键。主界面的中部主要显示本地获取的所有 MP3 歌曲列表。

(四)搜索界面

搜索界面的设计如下图 5 所示。左侧为控件设计图，右侧为功能设计图。主要用于显示用户的搜索记录。若没有搜索记录则不显示最近搜索。顶部设置返回按钮和用户输入框。返回键便于用户返回主页，输入框输入需要搜索的信息，搜索文本用于进行搜索功能。最近搜索记录的右侧有删除按钮，便于用户对某条记录进行删除。

当用户进行搜索后，如果有对应的搜索结果，则结果以歌单列表的形式显示。搜索的结果界面的概要设计如下图 6 所示。左侧为控件设计图，右侧为功能设计图。搜索结果同时显示歌曲的名称、歌手、歌曲的封面信息。搜索结果的顶部同样也显示搜索框，便于用户多次搜索。

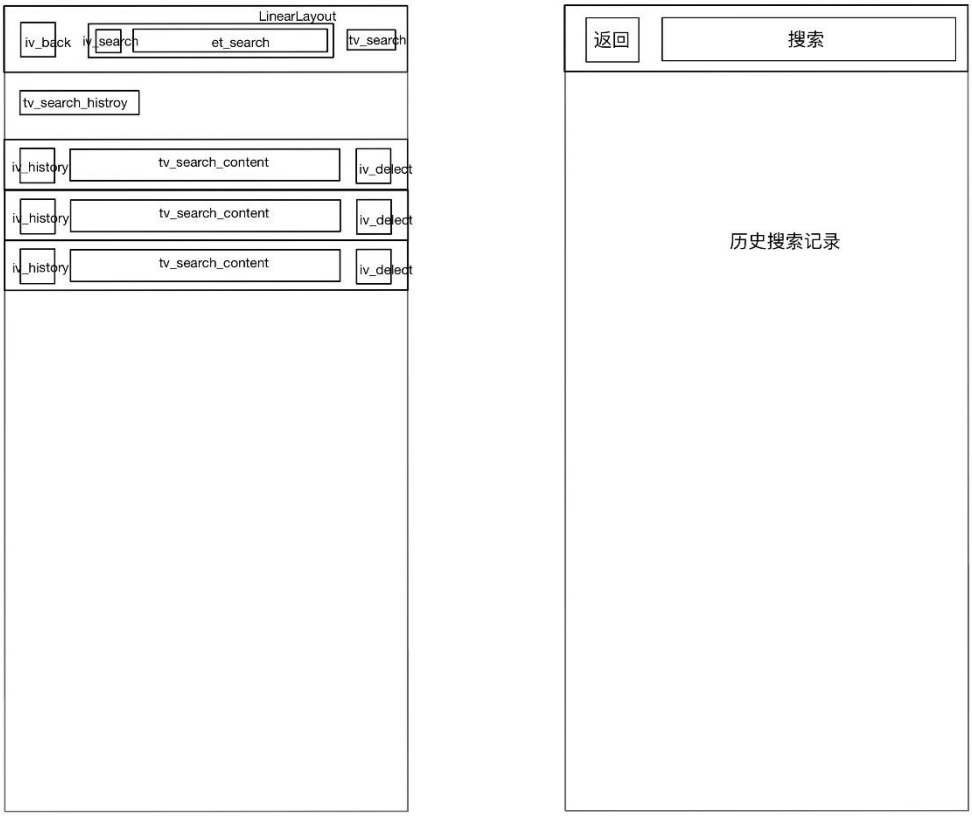

图 5　搜索界面概要设计图

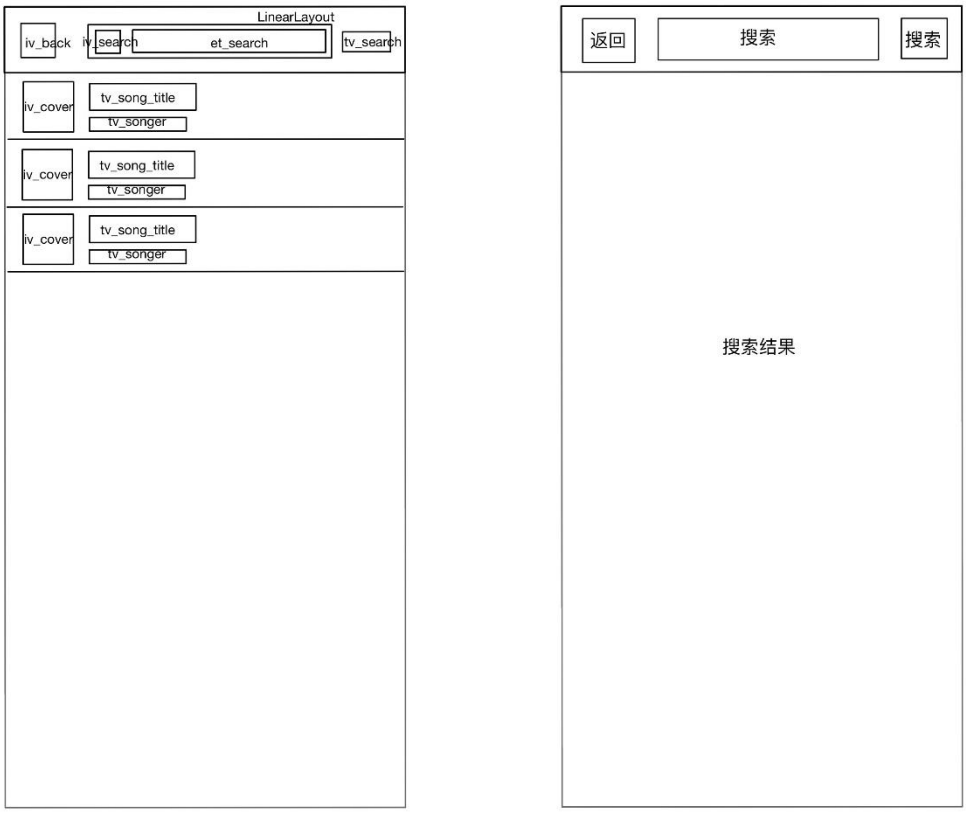

图 6　搜索结果界面概要设计图

(五)个人界面

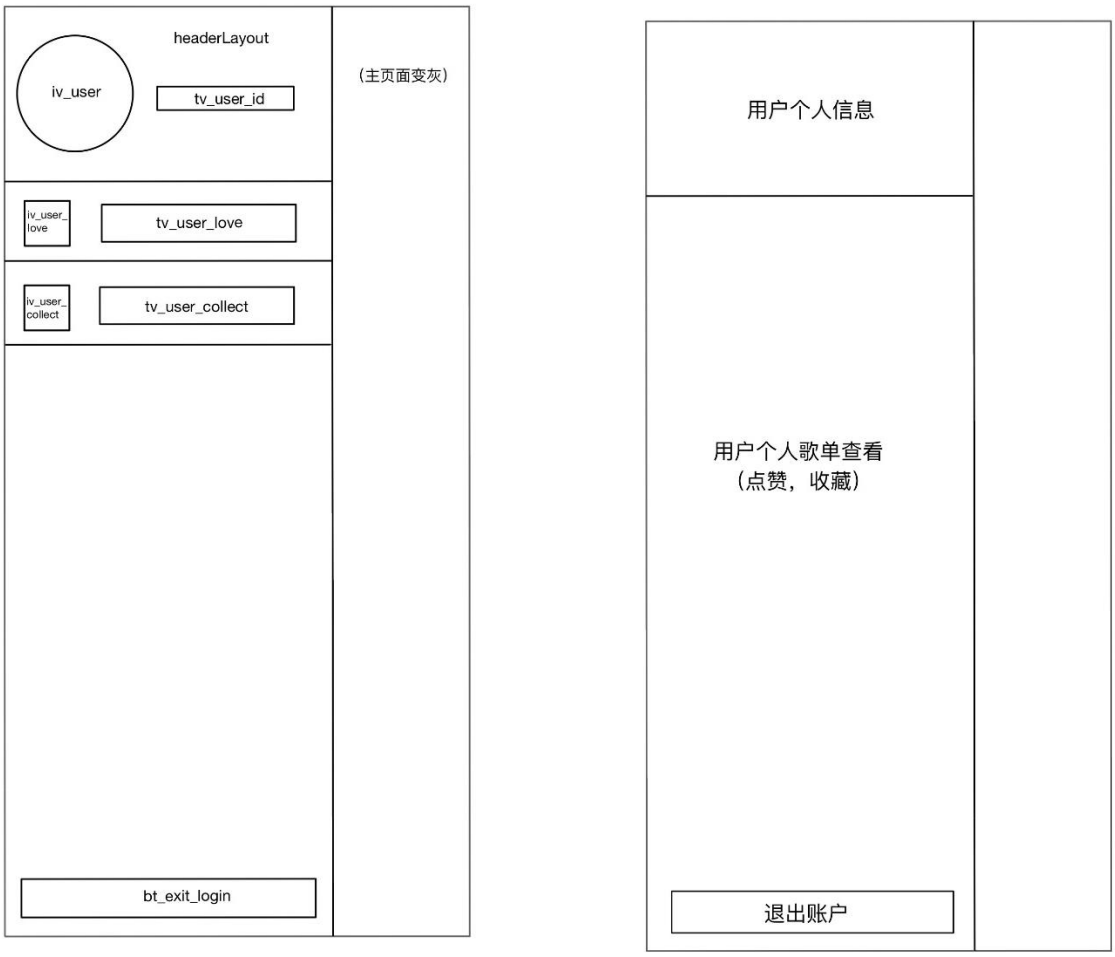

图 7　个人信息界面概要设计图

个人信息计如上图 7 示。左侧为控件设计图，右侧为功能设计图。主要用于显示当前用户的个人信息，如用户名。个人信息界面基于主界面实现，上层显示用户信息界面，下层显示音乐播放器主界面。

底部显示退出登录按钮，当用户已经登录时，退出登录按键完全显示；当用户还未登陆时，用户名处显示请登录，头像处显示基础未登录的头像，此时未登录按键为灰色，不可点击。个人信息界面还涉及点赞和收藏列表。用户处于已登录状态时，可以进行歌单的查看。

(六)歌单列表界面

歌列表界面入下图 8 所示。左侧为控件设计图，右侧为功能设计图。歌单列表的顶部设置返回按钮，便于关闭当前列表。紧接，设置歌单名称和封面图标。下方设置歌单内的歌曲信息，包括歌曲的封面、歌手、歌名信息。

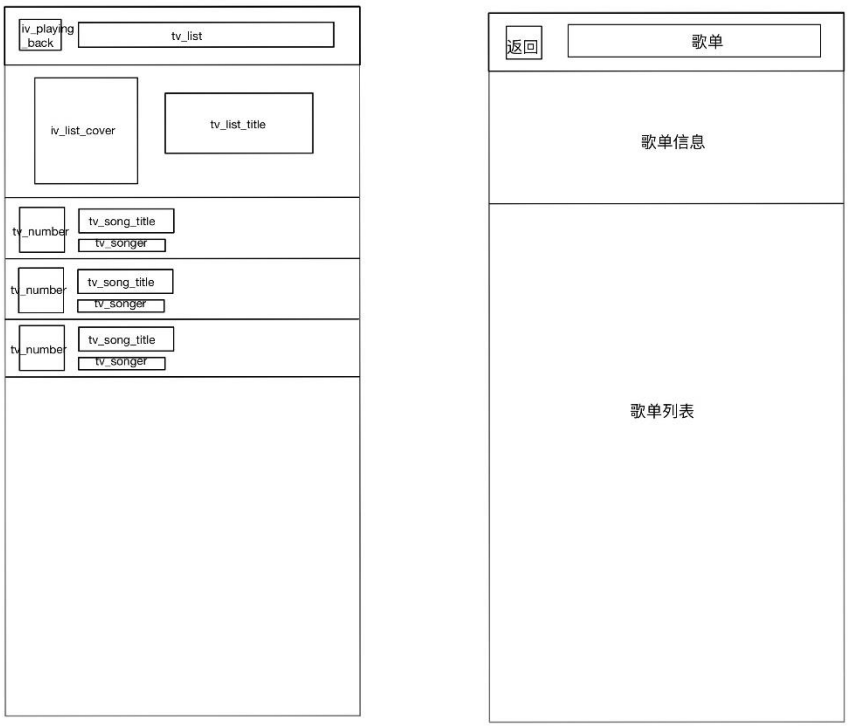

图 8　歌单界面概要设计图

(七)数据库设计

1. 歌单信息数据库

该数据库主要用于存放从本地读取的歌单信息，为功能实现提供数据源。主要包含歌曲的路径，点赞、收藏信息等。歌单数据库的概要设计如下表 1 所示。

表 1　歌曲信息数据库表 Songs

编号	歌名	歌手	歌曲路径	点赞	收藏
id	title	singer	filePath	love	collect
integer	text	text	text	boolean	boolean

2. 用户信息数据库

该数据库主要用于存放用户注册、登录的信息。当用户注册成功时，会向数据库插入信息。当用户进行的登录操作时，会核对数据库内的信息，判断是否成功登录。用户表设计如下表 2 所示。

表 2　用户信息数据库表 Users

编号	用户名	密码
id	usesrName	password
integer	text	text

四、 功能实现

项目整体代码文件框架设计如下图 9 所示。左侧为功能设计文件架构，右侧为布局文件的设计架构。整体包含 2 个数据库设计文件，2 个适配器文件，6 个活动 Activity 文件和 3 个碎片 Fragment 文件。

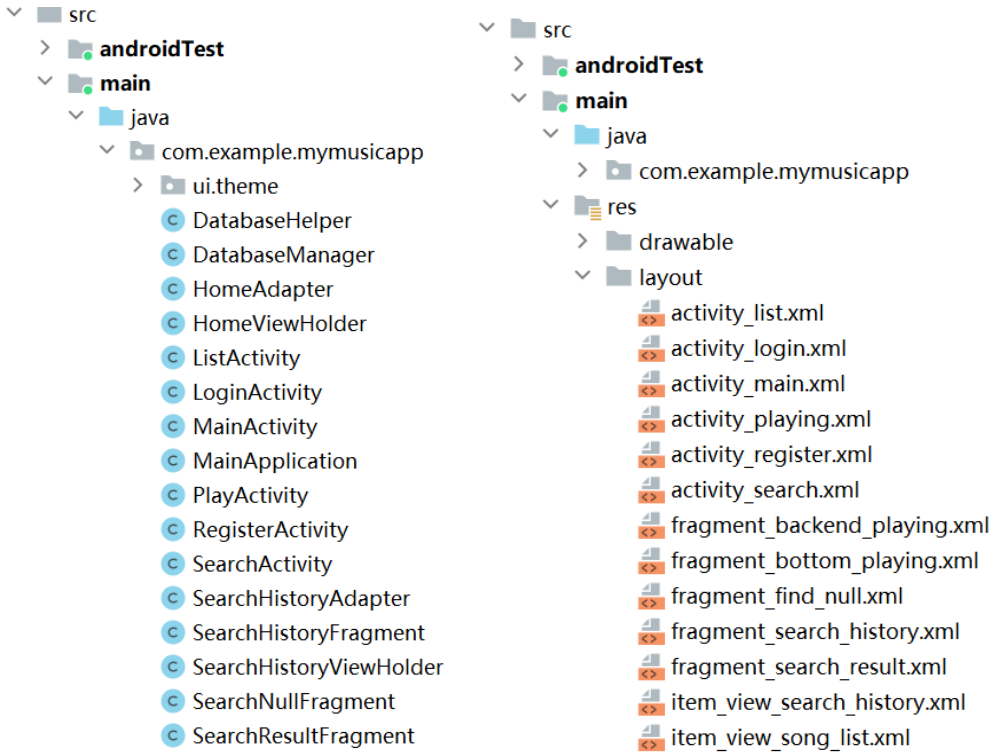

图 9　项目文件设计图

(一)数据库实现

1. 数据库构建、基础功能实现

数据库的初始化、更新，以及数据筛选，数据的获取功能都在 DatabaseHelper 类中实现，该类继承自 SQLiteOpenHelper 数据库类。初始化时，在 onCreate()中实现两个数据库表格的创建。onUpgrade()函数用于数据库的更新。

initSongsDB()函数用于初始化数据库，该函数调用 getMusicPaths()函数，用于实现 SD 卡目录下文件的遍历，isMusicFile()判断文件格式，将所有以.mp3 结尾的文件路径加入 List 列表中，然后调用 insertMusicFilesInfo()函数实现路径列表放入数据库。

自定义 getStoredMusicPath()函数实现获取数据表 Songs 中所有歌曲的路径。该函数先获取数据库的读权限，其次，将获取数据库的结果放在 Cursor 实例中，随后遍历 cursor 实例，将所获取到的路径放在 List 中，最后关闭数据库和 cursor

实例，返回 List。获取点赞列表 getLovedMusicPath()，获取收藏列表 getCollectedMusicPath()同理。代码实现如下图 10 所示。

```
public List<String> getStoredMusicPath() {
    SQLiteDatabase db = this.getReadableDatabase();
    List<String> storedMusicFiles = new ArrayList<~>();
    // 查询数据库中的音乐文件路径
    String query = "SELECT * FROM Songs";
    Cursor cursor = db.rawQuery(query,  selectionArgs: null);
    if (cursor.moveToFirst()) {
        do {
            String filePath = cursor.getString(cursor.getColumnIndexOrThrow( s: "filePath"));
            storedMusicFiles.add(filePath);
        } while (cursor.moveToNext());
    }
    cursor.close();
    db.close();
    return storedMusicFiles;
}
```

图 10　getStoredMusicPath()函数代码实现图

自定义 getTitle()函数，根据传递进来的音频文件的路径，初始化媒体 MediaMetadataRetriever，根据属性 METADATA_KEY_TITLE 返回歌曲文件的歌名。同理，自定义 getSinger()函数，属性 METADATA_KEY_ARTIST 获取歌手，getCover()获取封面的 bitmap 信息。代码实现如下图 11 所示。

```
public String getTitle(String filePath) {
    String title = "unknown";
    MediaMetadataRetriever mmr=new MediaMetadataRetriever();
    try {
        mmr.setDataSource(filePath);
        title = mmr.extractMetadata(MediaMetadataRetriever.METADATA_KEY_TITLE);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return title;
}
```

图 11　getTitle ()函数代码实现图

```
public void setLoved(String filePath){
    SQLiteDatabase db = this.getReadableDatabase();
    ContentValues values=new ContentValues();
    values.put("love",1);
    db.update( table: "Songs",values,  whereClause: "filePath = ?",new String[]{filePath});
}
```

图 12　setLoved ()函数代码实现图

自定义 setLoved()函数，用于对传入的音频文件，设置对应数据库的 love 列的值，用于当用户对歌曲点赞时，数据库的更新管理。同理，setLove()设定歌曲

数据库歌曲取消点赞，setCollected()设定指定歌曲数据库信息为收藏，setCollecte()设定指定歌曲数据库信息为取消收藏。代码实现如上图 12 所示。

自定义 getLastFilePath()函数，用于实现获取上一曲的歌曲路径。该函数先确定当前歌曲在歌单列表的位置，若不是第一首，则返回前一首歌的路径，若为第一首，则依旧返回当前歌曲的路径。同理 ，getNextFilePath()函数用于获取下一首歌的路径。代码实现如下图 13 所示。

```
public String getLastFilePath(String filePath,String ListName){
    List<String> list=new ArrayList<>();
    if(ListName.equals("collected")){
        list=getCollectedMusicPath();
    } else if(ListName.equals("loved")){
        list=getLovedMusicPath();
    }else {
        list=getStoredMusicPath();
    }
    Log.d( tag: "TAG", msg: "nowList: "+list.toString());
    // 使用循环查找目标字符串的下标
    int index = -1;
    for (int i = 0; i < list.size(); i++) {
        if (list.get(i).equalsIgnoreCase(filePath)) {
            index = i;
            break;
        }
    }
    Log.d( tag: "TAG", msg: "nowIndex: "+index);
    if(index==0){
        Log.d( tag: "TAG", msg: "returnPath: "+filePath);
        return filePath;
    }else{
        Log.d( tag: "TAG", msg: "returnPath: "+list.get(index-1));
        return list.get(index-1);
    }
}
```

图 13　getLastFilePath ()函数代码实现图

```
public int isLoved(String checkedFilePath){
    int love=0;
    List<String> storedMusicFiles=getLovedMusicPath();
    for (String item : storedMusicFiles) {
        if(item.equals(checkedFilePath)){
            love=1;
            break;
        }
    }
    return love;
}
```

图 14　isLoved ()函数代码实现图

自定义 isLoved ()函数，用于实现获取当前歌曲是否是点赞的歌曲。通过查询本歌路径是否在点赞歌单中，若是则返回 1，若不是，则返回 0.同理，isCollected()函数用于实现获取当前歌曲是否是收藏的歌曲。代码实现如上图 14 所示。

2. 数据库管理器

由于多个 Activity 需要使用数据库信息，避免多次获取 DatabaseHelper 实例，导致多次数据创建，设定一个数据库管理器，统一维护一个全局的 databaseHelper。设置 MainApplication 继承自 Application，用于获取全局的 context，同时修改 mainfiest 中 Application 的 name 属性为自定义的 MainApplication。将用户需要的列表如整体的全部歌单路径，点赞歌单路径，收藏歌单路径等放入管理器的 hashMap 中统一管理，当 hashMap 中不为空时，可以直接传递列表，不用再次进行数据库查询的耗时操作。

(二)注册、登陆功能实现

1. 登陆功能

界面创建的时候，先加载登陆界面的布局文件，进行控件的绑定。然后获取本地关于“记住密码”的数据，若有，则将记录的账户密码输入文本框中，并设置页面中“记住密码”按钮为选中状态；否则，文本框中不显示文本，同时“记住密码”按钮为默认的未选中状态。

为“注册”按钮设置监听，若用户点击，则发起一个注册的活动。为“《用户协议》”和“《隐私协议》”文本设置监听，当用户点击时，出现一个 toast 提示。如下图 15 所示。

图 15　查看协议运行图

为“登录”按钮设置监听，当用户点击时，判断协议是否已同意，若不同意则弹出 toast 提示，如下图 16 所示。若同意则获取用户输入的用户名和密码。若获取内容为空，则弹出 toast 进行提示。若不为空，则发起一个子线程，进行用户数据表 Users 内容比对。在 handler 中处理比对结果。若比对成功，则进入主界面，若用户同时选择记住密码，则更新本地 sharedPreference 存储内容。若比对失败，则根据判断，提示“用户名或密码错误”或“账户不存在”。当密码错误时，运行结果如下图 17 所示。

图 16　勾选协议运行图

图 17　“用户名或密码错误”运行图

```
protected void onDestroy() {
    super.onDestroy();
    if(handler!=null){
        handler.removeCallbacksAndMessages( token: null);
    }
}
```

图 18　活动销毁时释放 handler 代码实现图

为登录界面底部的“游客登陆”设置监听，当用户点击游客登陆时，将本地 sharedPreference 中当前用户名清空，并发起主界面活动。当登陆界面销毁时，若 handler 不为空，则移除，释放。销毁代码实现如上图 18 所示。

2. 注册功能

界面创建的时候，先加载登陆界面的布局文件，进行控件的绑定。为“注册”按键设置点击监听，当用户点击后，判断两次输入的密码是否一致，一致则将注册的饿用户信息放入数据库用户表 Users 中，否则，弹出 toast 提示信息，密码不一致。密码不一致的提示运行如下图 19 所示。注册界面的“游客登录”设计实现与登陆界面的“游客登陆”功能设计实现类似。

图 19 注册失败运行图

(三)主界面、个人界面实现

主界面的根布局文件选择 DrawerLayout，其中，第一部分的布局用于设置主界面的显示控件，第二部分的布局用于显示侧拉内容的布局文件。第二部分的布

局文件的宽度低于父布局，便于两个界面的悬浮显示，即当侧拉界面显示时，主界面的内容变灰，当点击灰色部分时，侧拉布局收回，显示主界面的布局。个人界面的显示如下图 20 所示。

进入该界面以后会申请 SD 卡读写权限。若权限获取成功，则会开启子线程，对 SD 卡进行扫描，获取数据，存入数据库中，将获取的数据列表，设置到 recyclerView 中。在实现之前，先需要自定义 HomeViewHolder 类，继承自 RecyclerView.ViewHolder，用于实现 itemView 中控件的绑定。自定义 HomeAdapter 类继承自 RecyclerView.Adapter<HomeViewHolder>，该类用于实现列表内容传递给控件，以及控件的监听。加载成功的主界面实现如下图 21 所示。

图 20　个人界面登陆后运行图

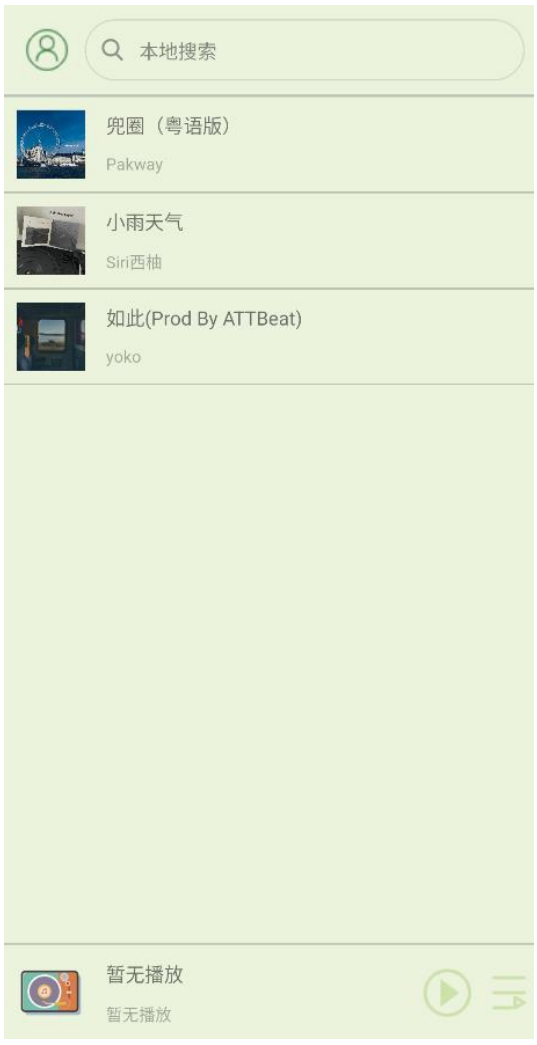

图 21　主界面加载成功运行图

为主界面上侧的用户图标设置监听，当用户点击该图标时，程序使用 drawerLayout.openDrawer(GravityCompat.START);语句实现 drawerLayout 第二层布局的显示。若本地 sharedPrefernece 存储的用户名为空，则个人信息显示为未

登录，个人点赞、收藏歌单、退出登录不设置点击事件，“退出登录”设置为灰。若用户点击，则提示“未登录”，为用户名处设置点击事件，当用户点击时，跳转至登陆界面。若当前有用户登录，则显示当前用户登陆名，为个人点赞、收藏歌单、退出登录设置点击事件，分别显示点赞、收藏歌单，以及返回登陆界面。

为上侧搜索框设置监听事件，当用户点击搜索框时，跳转至搜索界面。在布局文件中，为播放按钮设置 tag，当点击时，根据 tag 的内容，实现图标的切换。为列表图标设定监听事件，当点击该图标时，显示当前正在播放的列表。

在活动销毁时，判断当前播放器 mediaPlayer 是否为空，若不为空，则暂停并释放，handler 同理。

(四)播放实现

当进入播放界面时，先接收从其他 Activtiy 传递过来的 intent，从中获取需要播放的歌曲的路径。在子线程中，进行数据库内容的获取，将得到的歌曲信息，放在 bundle 中，设置一个标签，标注类别，通过 message 传递给 handler，在 handler 中将收到的信息放置到对应的控件中。播放界面的实现效果如下图 22 所示。

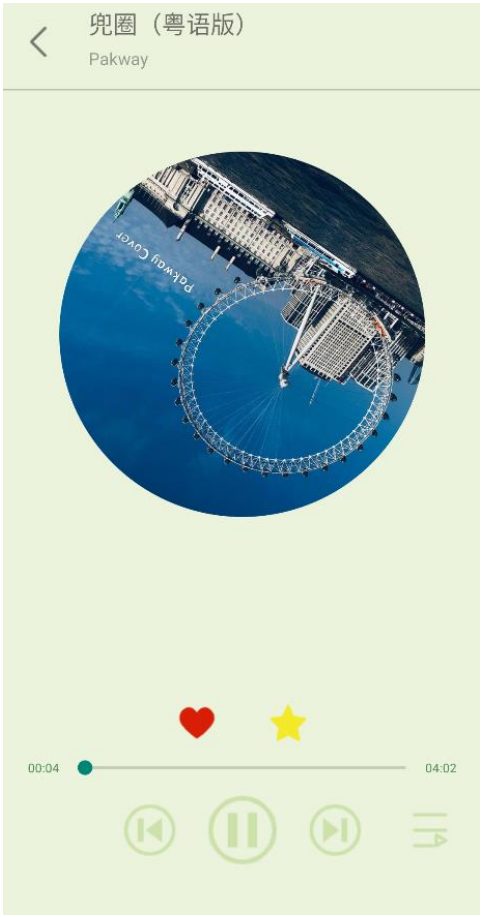

图 22　播放界面正在播放歌曲运行图

在界面控件绑定时，设置 animator 绑定歌单封面，设定旋转一圈的时间，速度，以及循环情况，实现播放时封面旋转功能。

根据获取的路径，初始化媒体播放器。当播放器准备完毕后，设定页面进度条的总时长，同时增加一个计时器，用于实现当前播放进度事件的更新，随后启动播放器，实现播放。

对进度条控件设置监听事件。在 onStopTrackingTouch()滑动停止时，更新媒体播放器的播放进度，在 onProgressChanged()进度条当前进度结束时，进行播放器内容的更新。进度条代码实现图下图 23 所示。

```
seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    no usages  Rong-r
    @Override
    public void onProgressChanged(SeekBar seekBar, int i, boolean b) {
        //进当滑动条到末端时，结束动画
        if (i==seekBar.getMax()){
            animator.pause();//停止播放动画
            mediaPlayer.pause();
            animator.pause();
            String newFilePath=databaseManager.getNextFilePath(filePath,fromList);
            Log.d( tag: "TAG", msg: "newFilePath: "+newFilePath);
            filePath=newFilePath;
            Log.d( tag: "TAG", msg: "filePathNew: "+filePath);
            onResume();
        }
    }
    //滑动条开始滑动时调用
    no usages  Rong-r
    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {
    }
    //滑动条停止滑动时调用
    no usages  Rong-r
    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {
        //根据拖动的进度改变音乐播放进度
        int progress=seekBar.getProgress();//获取seekBar的进度
        mediaPlayer.seekTo(progress);//改变播放进度
    }
});
```

图 23　进度条监听代码实现图

为界面的暂停按键设置监听，当用户点击时，通过该图标的 tag 信息，判断是否进行图片和 tag 的切换，如果有媒体正在播放，则设置媒体暂停、动画暂停。

为上一曲、下一曲图标设置点击监听事件，当用户点击时，通过数据库管理者获取上一曲/下一曲的歌曲路径，同时，重新设定播放界面。为点赞、收藏按键设置点击事件，当用户未登录且点击时，提示登录；当用户登陆后，点击图标，修改对应的 tag 和图片资源，同时，根据当前的歌曲路径，更新数据库中的数值。

(五)歌单列表实现

当执行列表界面的 Activity 时，先获取从其他 Activity 传来的请求列表的名称，开启一个子线程，根据请求内容，获取对应的个单列表，并对列表界面的歌单信息和封面进行设置。运行结果如下图 24 所示，获取对应歌单的代码实现如下图 25 所示。为顶部返回图标设置监听事件，当用户点击后，结束当前 Activity。

图 24　获取当前全部歌单列表运行图

```
private List<String> getList(String listName){
    List<String> listToShow=new ArrayList<>();
    if(listName.equals("collected")){
        listToShow.addAll(databaseManager.getMusicListCollected());
        textViewListTitle.setText("我的收藏");
        imageViewListIcon.setImageResource(R.drawable.playing_collected);
    } else if (listName.equals("loved")) {
        listToShow.addAll(databaseManager.getMusicListLoved());
        textViewListTitle.setText("我的喜爱");
        imageViewListIcon.setImageResource(R.drawable.playing_loved);
    }else {
        listToShow.addAll(databaseManager.getMusicListAll());
        textViewListTitle.setText("全部歌曲");
    }
    return listToShow;
}
```

图 25　获取对应歌单列表以及对应图片、标题设置代码实现图

(六)搜索功能实现

搜索界面的顶部控件固定设置，导航栏下方为 Fragment 控件存放区域。当用户进入搜索界面时，先调用 isShowHistory()函数，用于获取本地的搜索历史内容，如果有历史记录，则调用历史记录的 SearchHistoryFragement。

SearchHistoryFragement 继承自 Fragment。在 SearchHistoryFragement 中，绑定对应的 recyclerView，同时为 recyclerView 设置适配器 SearchHistoryAdapter。SearchHistoryAdapter 继承自 RecyclerView.Adapter<SearchHistoryViewHolder>。SearchHistoryViewHolder 用于绑定 itemView 中的控件。在 SearchHistoryAdapter 中，将历史记录的列表内容与控件进行绑定，实现历史记录的显示，同时为各条控件的删除图标设置监听事件，实现该条记录的删除，同时更新本地的历史记录内容。搜索记录的实现如下图 26 所示。

为该界面的“搜索”文本设置监听事件，当用户点击该文本时，先隐藏软键盘，调用 addHistory()函数，进行历史记录更新。然后开启子线程，处理用户文本框中输入的信息，调用 canSearch()函数，如上图 26 所示。实现数据库内容搜索。将搜索结果类型通过 handler 传递到主线程中，由主线程根据传递结果选择 Fragment。若能搜索到结果，则放置 SearchResultFragment，将搜索结果列表传递到适配器中。若没有搜索到，则放置 SearchNullFragment，提示没有搜到。搜索成功如下图 27 所示，搜索失败如下图 28 所示。

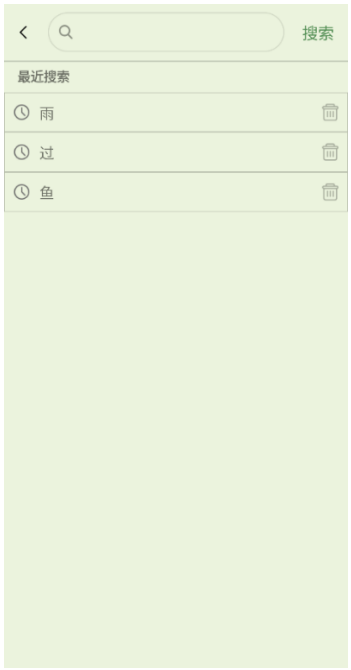

图 26　搜索历史运行图

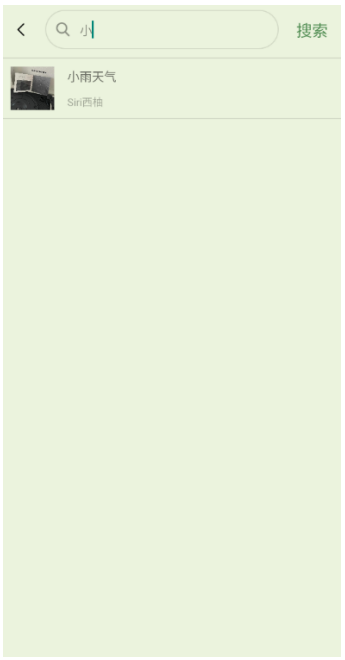

图 27　可搜索运行图

图 28 无搜索结果运行图

五、 总结与反思

通过三周的安卓开发的学习与实践，对应用程序完整的开发流程有了基础的认知，意识到了前期需求分析与概要设计对后期项目开发的影响，同时，需要注重沟通与学习，对于不确定的问题要适当向导师询问，明确方向后继续学习。

在本次项目的开发中，完成了基础的项目设计，以及功能的实现，但也还存在一些不足之处。音乐播放时，界面转换，播放器的控制，以及后台服务 Service 的部分有待深入学习，并对项目进行改进。对于数据库部分，在导师的指导下，对数据库的使用有了初步的改进，能够设定一个全局的数据库管理器，避免多次创建数据库实例。但数据库管理器的使用还存在部分缺陷，数据的存取，以及数据库管理器的缓存数据部分的设计需要优化。对于本地 SD 卡权限的声情，以及对应 SDK 版本权限管理的部分处理也有待改进。进一步实现后，可以考虑设定不同用户维护不同的点赞、收藏歌单，以及喜好推荐等，进而可以尝试实现网络歌单的获取以及歌单推荐，不断优化、丰富项目的功能。

总之，目前的项目实现内容是个简洁基础的本地播放器，能够进行本地歌单获取，以及动画播放，用户注册登录等功能。未来通过深入学习安卓开发的内容，熟悉版本适配等内容，可以对该播放器进行不断地改进优化，争取做到一个合格的上线的音乐播放器应用程序。