****

移动应用开发 课程设计报告

题 目： 基于Android的音乐平台

App设计与实现

学 院： 计算机科学与工程学院

专 业： 计算机科学与技术

班 级： 卓越班

姓 名： 鱼洲

学 号： 202231607206

教师姓名：

摘 要

随着移动互联网的飞速发展，音乐平台的需求日益增长，尤其是基于Android的移动应用因其普及性和便捷性成为用户音乐享受的主要平台。本文设计并实现了一款基于Android的音乐平台App，提供用户注册、登录、歌曲播放、动态发布、评论互动等核心功能，满足了用户对音乐的娱乐需求，同时也为用户提供了社交互动的空间。此外，系统根据用户的喜好，推荐热门歌曲及动态，增强了用户的互动体验。

在开发过程中，系统采用了Android原生技术，利用SQLite作为数据库存储用户、歌曲、评论、动态等信息。通过简单直观的UI设计，用户能够方便地进行歌曲播放、动态发布、评论点赞等操作。系统还为用户提供了个人空间，可以查看和修改个人信息、收藏歌曲，并进行动态的发布与管理。

此外，本应用还实现了多样化的播放功能，如单曲、顺序、随机播放以及点赞、评论等社交互动功能，使得用户在享受音乐的同时，能够与朋友分享、交流音乐感受。通过对Android开发技术的深刻理解与运用，结合实际需求，本文详细阐述了应用的开发流程、技术实现与优化策略，旨在为未来同类应用的开发提供有益的参考与借鉴。

关键词：Android；音乐播放器；动态发布；SQLite；社交互动；个人空间

目 录

[第一章 绪论 1](#_Toc32343)

[1.1 开发背景 1](#_Toc11064)

[1.2 开发意义 1](#_Toc16777)

[1.3 项目概述 1](#_Toc13930)

[第二章 开发环境及关键技术 3](#_Toc17466)

[2.1开发平台 3](#_Toc22944)

[2.2前后端技术栈 3](#_Toc4189)

[2.3数据库技术 4](#_Toc11533)

[2.4项目重点技术（音乐相关功能） 4](#_Toc25398)

[第三章 系统需求分析 6](#_Toc28475)

[3.1用户方面 6](#_Toc1566)

[3.1.1 注册与登录 6](#_Toc32075)

[3.1.2 个人空间 6](#_Toc24117)

[3.2音乐方面 6](#_Toc14939)

[3.2.1歌曲推荐机制 6](#_Toc29357)

[3.2.2歌曲搜索 6](#_Toc18198)

[3.2.3歌曲播放、暂停与切换 7](#_Toc26282)

[3.2.4播放模式的切换 7](#_Toc12979)

[3.2.5歌曲详情页 7](#_Toc2747)

[3.3社交方面 8](#_Toc22762)

[3.3.1动态发布 8](#_Toc2935)

[3.3.2动态推荐机制及详情页 8](#_Toc31921)

[3.3.3评论与收藏歌曲 8](#_Toc11729)

[3.3.4评论与点赞动态 8](#_Toc28091)

[3.4界面交互设计 9](#_Toc20239)

[3.4.1页面切换 9](#_Toc14094)

[3.4.2底部导航栏 9](#_Toc19859)

[3.4.3侧边栏与隐藏框设计 9](#_Toc1836)

[3.5其他需求 9](#_Toc26839)

[3.5.1性能需求 9](#_Toc18083)

[3.5.2安全性需求 10](#_Toc23109)

[3.5.3可用性需求 10](#_Toc14337)

[第四章 数据库设计 11](#_Toc13779)

[4.1系统概念结构设计 11](#_Toc5246)

[4.2系统逻辑结构设计 11](#_Toc22670)

[4.3数据库实施与连接 13](#_Toc29399)

[4.3.1创建数据库 13](#_Toc7294)

[4.3.2创建数据库表 14](#_Toc29237)

[4.3.3数据库的连接 16](#_Toc1148)

[第五章 界面设计及功能实现 18](#_Toc12562)

[5.1界面设计 18](#_Toc26617)

[5.1.1注册登录界面 18](#_Toc6266)

[5.1.2音乐界面 19](#_Toc9673)

[5.1.3动态界面 20](#_Toc1560)

[5.1.4个人界面 21](#_Toc24168)

[5.2前后台连接 22](#_Toc29680)

[5.3功能实现及截图展示  23](#_Toc31069)

[5.3.1注册登录功能 23](#_Toc29373)

[5.3.2搜索功能 24](#_Toc5786)

[5.3.3歌曲收藏及评论功能 25](#_Toc13373)

[5.3.4歌曲播放功能 26](#_Toc21523)

[5.3.5发布动态功能 28](#_Toc3325)

[5.3.6动态点赞及评论功能 28](#_Toc20712)

[5.3.7个人信息修改功能 29](#_Toc20833)

[5.3.8社交信息查看及删除功能 30](#_Toc25280)

[第六章 总结 33](#_Toc20465)

[6.1系统优点 33](#_Toc22367)

[6.2系统缺点 33](#_Toc31558)

[6.3改进方向 33](#_Toc11122)

[参考文献 35](#_Toc10236)

[项目组成员分工及贡献度 36](#_Toc10529)

# 第一章 绪论

## 开发背景

在信息技术日新月异的今天，移动互联网已经深入到人们日常生活的方方面面，极大地改变了我们的生活方式。尤其是随着智能手机的普及和技术的不断革新，移动应用成为了我们获取娱乐、沟通和信息的重要工具。而其中，音乐作为人类文化和情感表达的重要载体，早已不再局限于传统的音响设备和CD播放器。数字音乐的出现和流媒体平台的崛起，使得音乐欣赏更加便捷，几乎随时随地都能享受高品质的音响体验。

Android操作系统作为全球市场中最受欢迎的手机操作系统，已经成为开发移动应用的主流平台。通过Android平台开发的音乐应用，不仅满足了用户对高质量音乐的需求，同时也催生了更多创新功能的探索，诸如个性化推荐、社交互动、音乐分享等。因此，开发一款基于Android的音乐平台应用，既能推动技术的进步，也能满足日益增长的用户需求，具有十分重要的现实意义。

## 开发意义

随着数字音乐的蓬勃发展，越来越多的用户希望能够在移动设备上实现便捷的音乐播放和互动体验。传统的本地播放器虽能满足基本的音乐播放需求，但却缺乏与用户情感和社交需求的深度融合。因此，如何在现有的技术框架下，设计一个集音乐播放、社交互动和个性化推荐为一体的移动应用，成为了音乐平台应用开发的核心挑战之一。

本项目旨在通过Android开发技术，结合现代音乐平台的需求，打造一款既具备优质播放功能又能提供丰富社交互动的音乐平台App。通过开发这一应用，不仅能够满足用户对音乐的基本需求，还能让用户在平台上实现自我表达、社交互动和情感交流。通过对用户行为的精确分析，本平台能够提供更具个性化的服务，提升用户的使用体验，同时增强平台的粘性和活跃度。

## 项目概述及功能分解图

本项目的基本目标是设计并实现一款基于Android平台的音乐应用，能够为用户提供稳定流畅的音乐播放功能，以及便捷的社交互动功能。为此，本应用具备以下几项核心功能：

1. **音乐播放与管理**：实现本地音乐的在线播放、播放列表的添加与编辑，提供播放、暂停、切换歌曲等基本功能，并支持按歌曲名称、歌手、专辑等信息搜索歌曲以及多种播放模式（单曲循环、顺序播放及随机循环等）。用户可以收藏自己喜爱的歌曲，并在歌曲详情页进行评论，分享对歌曲的看法。

**（2）用户社交互动**：用户可以发布动态，评论和点赞歌曲及动态，与其他用户进行互动。系统还会根据用户的互动行为（评论及点赞）为用户进行热门推荐，进一步增强社交体验。

**（3）用户管理**：提供用户的注册、登录、个人信息管理等功能。用户可以通过邮箱注册，允许用户修改和编辑自己的个人资料，包括头像、昵称、个性签名和个人背景等。这些功能可以增强用户的个性化体验，使得用户能够展示自己的独特风格和个性，同时提升应用的社交性。

**（4）数据管理与存储**：利用Android内置的SQLite数据库进行本地数据的存储与管理，确保数据的持久性和高效查询。

在研究内容上，本项目将涉及以下几个方面：

**（1）Android开发技术的应用与实现**：在开发过程中，深入探讨Android开发环境、UI设计、数据库管理等技术的运用，解决实现过程中遇到的技术难题。

**（2）数据库设计与优化**：设计合理的数据表结构，确保数据存储与检索的高效性，保证用户数据、歌曲信息、评论、动态等内容的完整性与一致性。

**（3）用户体验的优化与提升**：通过简洁明了的界面设计、流畅的交互体验，提升用户的使用满意度。

**（4）社交互动与个性化推荐系统的实现**：研究社交互动功能和个性化推荐系统的设计，增强平台的社交属性和个性化服务。

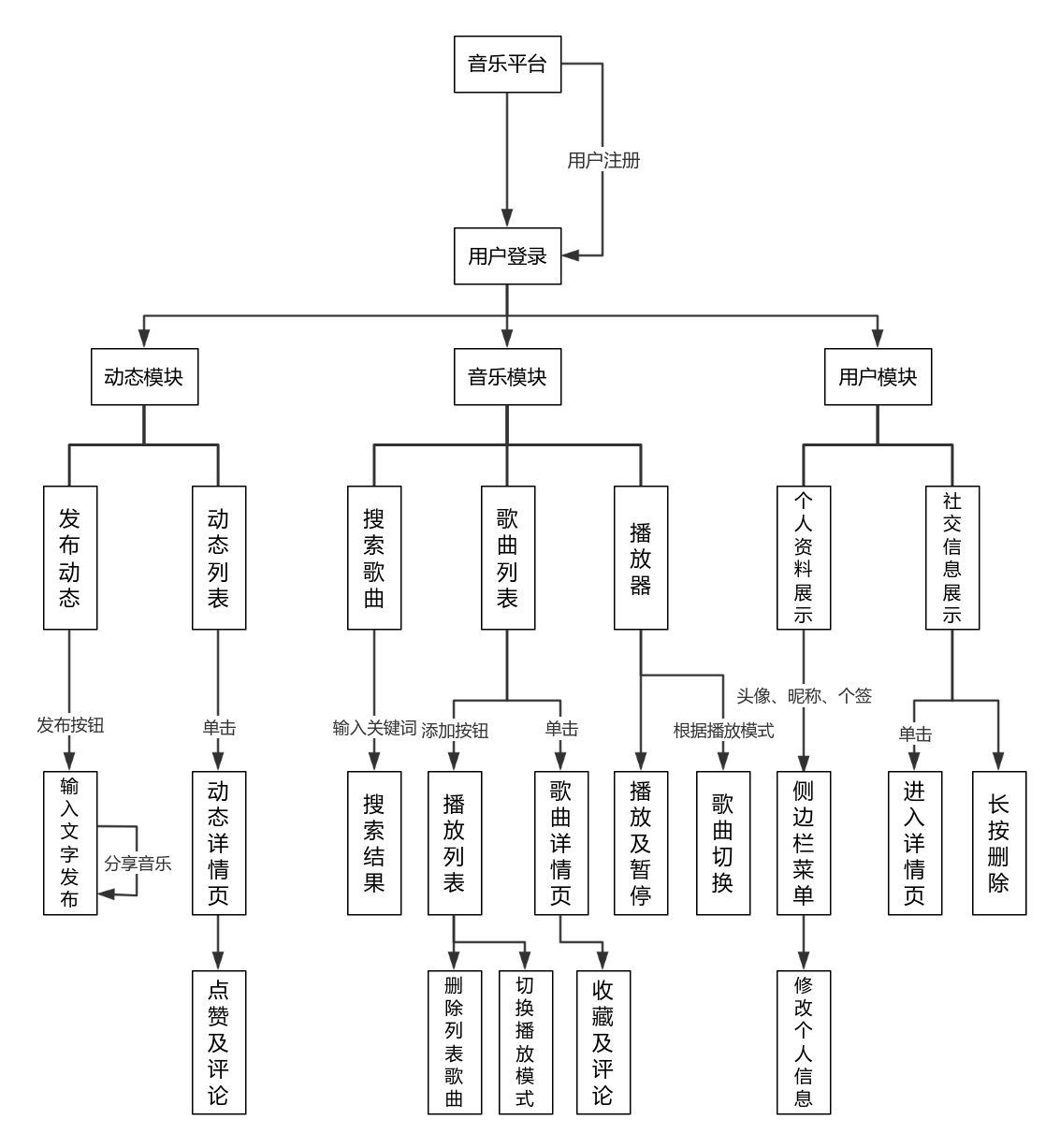


图1.1 项目功能分解图

# 第二章 开发环境及关键技术

## 2.1开发平台

本项目的开发平台基于 **Android操作系统**，充分利用其成熟的生态系统和强大的设备支持。选择Android平台，是因为其开放性、灵活性以及庞大的用户群体，为移动应用开发提供了丰富的资源和开发工具。具体来说，本应用的开发使用 **Android Studio** 作为主开发环境，Android Studio 是Google推荐的官方IDE，它为开发者提供了全面的工具支持，包括智能代码补全、UI设计、调试和性能分析等功能。

在此基础上，项目还依赖于 **Java编程语言**，利用Java的面向对象特性实现应用功能的模块化与可扩展性。对于更复杂的UI设计和交互，本项目使用了 **XML布局文件** 来定义界面，确保界面设计的清晰与高效。此外，Android开发平台的强大设备支持意味着本应用能够广泛适配多种尺寸和分辨率的设备，从智能手机到平板电脑，都能提供良好的使用体验。

Android的系统架构使得开发者能够直接访问底层硬件，极大地提升了应用的性能与响应速度。在本项目的实现过程中，我们通过Android的 **Intent机制** 和 **服务组件** 来实现应用内的各类交互，同时借助 **BroadcastReceiver** 完成后台任务的处理，为用户提供流畅的体验。

## 2.2前后端技术栈

在前端开发部分，本项目采用了Android自带的UI组件和布局系统。通过XML布局文件，设计了符合用户需求的交互界面。在UI组件的选择上，项目结合了常见的视图控件，如TextView、Button、ImageButton等，并通过RelativeLayout、LinearLayout等布局方式实现了页面元素的合理排列。

项目中，为了实现动态效果和更高的交互性，采用了ViewPager2来展示轮播图，RecyclerView来展示动态列表以及播放列表。通过这些组件的结合，用户能够体验到流畅的界面和简洁的操作。

为了提高用户体验，项目中还使用了多个UI自定义控件，如自定义按钮、进度条等，并结合动态更新、响应式布局等技术，使得界面在不同屏幕尺寸下都能够自适应展示，确保了应用的高兼容性。

前端开发过程中，主要依赖于Android的原生组件和布局系统，充分利用其丰富的视图组件和强大的UI框架，确保了良好的用户体验和界面交互。

后端技术部分，考虑到项目的轻量化和开发周期的要求，项目并未使用Spring系列技术框架、MyBatis、Spring MVC等常见的后端技术栈。为了简化开发流程，并减少系统复杂性，本项目后端主要采用了最基础的安卓开发技术。

具体来说，后端逻辑部分主要负责与数据库进行交互，包括用户信息的存储与管理、歌曲数据的保存与加载、动态内容的管理等。为了保证数据的持久性，采用了Android自带的SQLite数据库，通过编写自定义SQL语句和封装数据库操作类来进行增删改查。

此外，项目中的一些核心功能，如用户登录、歌曲播放、动态发布等，均通过Android的内部存储和文件管理功能进行实现。虽然没有引入复杂的后端框架，但这种轻量级的方式能有效避免不必要的复杂性，同时确保了应用在移动端的高效运行。

## 2.3数据库技术

为了保证数据的可靠存储和快速访问，项目采用了Android内置的SQLite数据库。这一数据库引擎内置于Android设备中，具有较高的性能和较小的占用空间，适合于单机应用存储数据。

项目中，数据库包含多个表格，如用户表、歌曲表、动态表等，每个表格都被精心设计以满足业务需求。例如，用户表中包含邮箱、密码、个性签名等字段，歌曲表则记录了歌曲的名称、艺术家、专辑等信息。此外，还使用了外键约束，确保数据的完整性和一致性。

通过SQLite数据库，应用能够在本地高效地处理大规模数据，如用户的点赞、评论、动态等信息。与此同时，数据库的设计也确保了数据表结构的清晰和操作的简便，使得后期的数据扩展和维护变得更加容易。

## 2.4项目重点技术（音乐相关功能）

本项目的核心技术原理涉及到音乐播放、暂停、歌曲切换以及播放模式切换等功能的实现。为了使音乐播放体验更加流畅和多样化，采用了安卓平台的 MediaPlayer 类来处理音频文件的播放、暂停、跳转等操作，其状态机时序图如图2.1所示。

在歌曲播放的过程中，MediaPlayer 用于加载和播放音频文件。歌曲播放由 MediaPlayerManager 类中的 playSong() 方法控制。首先，该方法会检查播放列表是否为空，并在歌曲路径有效的情况下开始播放。如果启用了随机播放模式，系统会随机选择一首歌曲。如果当前歌曲与上一首歌曲相同且播放器处于暂停状态，系统则恢复播放。否则，系统会释放当前播放器资源并加载新的歌曲文件，确保每次播放都能得到最好的效果。

歌曲文件的路径通过 Uri 进行处理，支持从本地存储或通过 content:// 协议从应用提供的资源中加载音乐。通过 MediaPlayer 的 setDataSource() 方法，音频文件被正确加载到播放器中。为了支持大部分音频格式，系统对文件路径做了充分的异常处理和错误捕获，确保在播放过程中如果出现问题，能够及时反馈并停止播放。

播放模式的切换包括顺序播放、单曲循环和随机播放。在顺序播放模式下，播放器按顺序播放播放列表中的歌曲，播放完最后一首后会重新开始。单曲循环模式允许用户选择某一首歌曲不断重复播放，而随机播放则随机选取播放列表中的歌曲。这些播放模式通过 isSingleLoop 和 isRandom 两个标志变量来控制，提供了更高的灵活性。

歌曲切换功能是通过更新当前播放索引来实现的。无论是顺序播放还是随机播放，播放器会根据当前播放状态和模式，动态决定下一个要播放的歌曲。例如，当用户请求播放下一首歌曲时，playNext() 方法会检查播放模式并根据情况选择播放列表中的下一首歌曲。在随机播放模式下，系统会随机选择一首歌曲；在顺序播放模式下，系统会播放当前歌曲的下一首。

为了提升用户体验，系统还实现了暂停与恢复播放的功能。当用户点击暂停按钮时，播放器会暂停当前歌曲的播放并记录播放状态，用户可以随时点击恢复按钮继续播放当前歌曲。此外，播放器还会在每首歌曲播放结束后自动调用 onCompletionListener，以便在歌曲播放完毕后根据播放模式切换到下一首歌曲。

为了确保播放体验的流畅性和系统资源的有效管理，MediaPlayer 在播放过程中会定期释放不再需要的资源。每当歌曲切换或退出播放时，release() 方法会被调用，释放 MediaPlayer 占用的系统资源，避免内存泄漏或性能问题。

通过这些技术原理的结合，项目实现了高效、稳定的音乐播放体验，支持多种播放模式和歌曲切换操作，同时提供了良好的用户交互界面和流畅的播放控制。

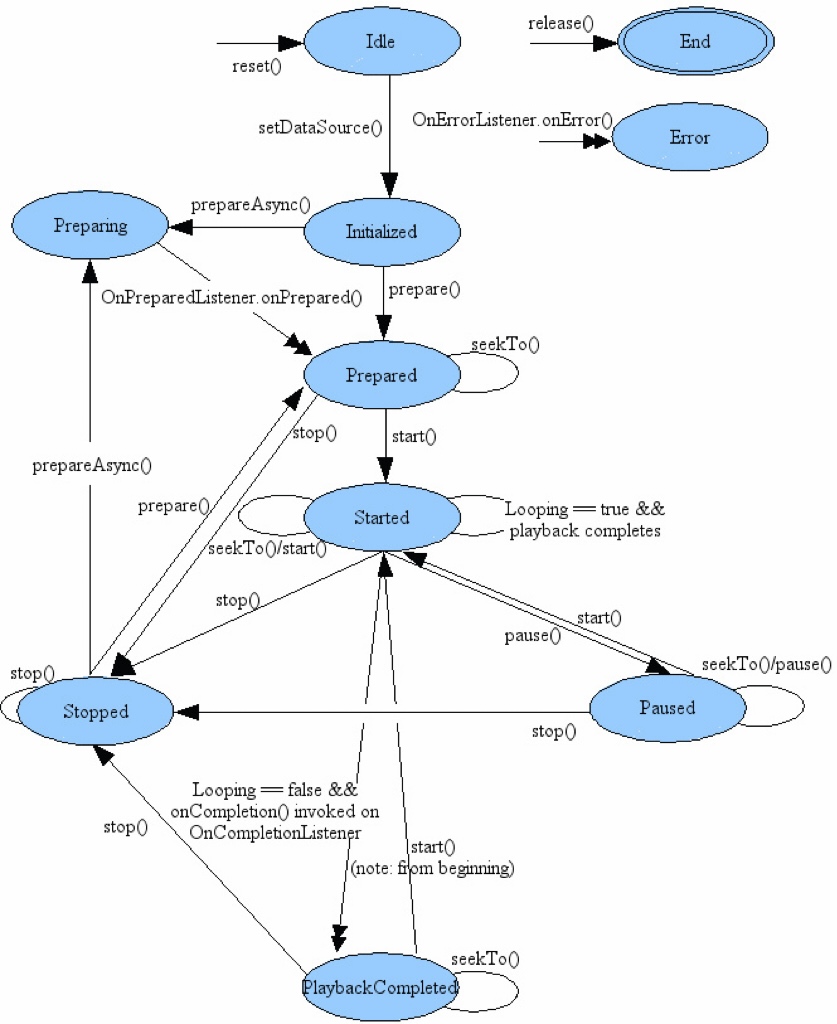


图2.1 MediaPlayer状态机时序图

# 第三章 系统需求分析

## 3.1用户方面

### 3.1.1 注册与登录

在本音乐平台应用中，用户的注册和登录功能是进入系统的第一步。用户可以通过邮箱、密码的方式完成注册，并能够通过相同的方式进行登录。为确保账户的安全性，用户密码存储采用了加密处理，防止数据泄露。此外，登录界面设计简洁，支持自动记住密码和一键登录功能，提升用户的便捷性。

### 3.1.2 个人空间

个人空间是用户在平台中的独立展示区，集成了用户的基本信息、社交动态、评论记录和收藏歌曲等内容。在个人空间中，用户可以查看和编辑自己的昵称、头像、个性签名等个人资料，同时还可以浏览自己的动态、评论和收藏歌曲，进一步加强个性化的展示和社交互动。

除了展示这些信息外，用户还可以直接点击按钮来播放自己收藏的歌曲。通过这种方式，用户能够方便快捷地访问和聆听自己喜爱的音乐，使得个人空间不仅仅是信息的展示平台，更成为一个互动的娱乐平台。这一功能的加入，不仅提升了用户的体验感，也增强了用户与平台内容之间的连接，使得个人空间成为了一个集娱乐、社交与自我展示于一体的重要功能模块。

## 3.2音乐方面

### 3.2.1歌曲推荐机制

在音乐平台中，歌曲的展示不仅仅依赖于其内容的丰富性，还与用户的互动息息相关。为了提升用户体验并让最受欢迎的歌曲得到更多曝光，本平台通过歌曲的**点赞数**来优先展示那些受欢迎的歌曲。具体而言，系统会根据每首歌曲的点赞数进行排序，确保点赞数较高的歌曲出现在用户的首页或推荐列表中。这种展示方式不仅让用户能够快速找到热门歌曲，还促进了用户之间对音乐的互动和分享。

通过这种动态的展示机制，平台能够充分调动用户的参与热情，使得热门歌曲得以优先展示，同时也鼓励用户对自己喜爱的歌曲进行点赞，形成一个良性循环，提升平台的活跃度和社交氛围。

### 3.2.2歌曲搜索

歌曲搜索功能是本应用中不可或缺的一部分，旨在提供用户快速找到自己喜爱歌曲的能力。通过集成强大的搜索引擎，用户能够根据歌曲名称、歌手、专辑等多种维度进行精准检索。为了提升搜索体验，系统支持模糊匹配，当用户输入部分关键词时，系统会自动推荐相关的歌曲，帮助用户迅速锁定目标。同时，搜索结果不仅展示歌曲名称，还会附带歌手、专辑封面等信息，提供多维度的视觉参考。

为了应对歌曲库庞大的情况，搜索结果页面通过动态方式加载更多结果，避免一次性加载过多数据导致界面卡顿。用户可以轻松浏览和点击任意一首歌曲进入其详情页，进一步了解歌曲信息或开始播放。

### 3.2.3歌曲播放、暂停与切换

歌曲播放功能是本应用的核心，旨在提供流畅的音乐体验。当用户选择一首歌曲进行播放时，系统会从本地或远程服务器加载该歌曲，并通过内置播放器进行播放。在播放过程中，用户可以通过控制按钮进行操作，如暂停、播放和停止。暂停功能能够让用户随时暂停当前播放的歌曲，并且用户重新点击播放按钮时，系统会从暂停的位置继续播放，提供无缝的使用体验。

同时，歌曲切换功能允许用户根据需求切换不同的歌曲。在播放列表中，用户可以通过点击下一首或上一首按钮来切换歌曲，系统会自动加载并播放目标歌曲。如果用户不想手动切换，可以启用自动播放功能，系统会根据预设的播放模式（顺序播放、单曲循环、随机播放）自动切换到下一首歌曲。歌曲切换时，系统保持流畅的界面过渡，不会出现卡顿或延迟现象，确保音乐播放的连续性。

### 3.2.4播放模式的切换

为了更好地满足用户对音乐播放的个性化需求，系统提供了多种播放模式切换功能。用户可以根据自己的偏好选择不同的播放模式，包括顺序循环、单曲循环和随机循环。

**（1）顺序循环模式**：在此模式下，歌曲会按照用户创建的播放列表顺序依次播放，当播放列表中的歌曲全部播放完毕后，系统会自动从第一首歌开始循环播放。该模式适用于那些希望按顺序欣赏整个播放列表的用户。此功能可通过简单的按钮切换进行选择，界面会清晰地显示当前所选的播放模式。

**（2）单曲循环模式**：此模式下，当前播放的歌曲会被重复播放，适合喜欢单曲的用户。无论歌曲的播放进度如何，当歌曲结束时，系统会自动从头开始播放该曲目，用户无需进行任何操作即可享受歌曲的重复播放。这一功能的便捷性和流畅度大大提升了用户的音乐享受。

**（3）随机循环模式**：在随机循环模式下，系统会随机选择播放列表中的歌曲进行播放。每首歌曲的播放顺序都是不可预测的，为喜欢发现新歌曲或改变播放节奏的用户提供了全新的体验。随机播放不仅为用户提供了更多的选择空间，还打破了按顺序播放的局限，让每次听歌都充满新鲜感。

播放模式的切换功能通过简单的图标按钮实现，用户可以在播放界面上方便地选择并切换到自己喜好的模式。此外，系统还会在界面上显示当前播放模式的图标，确保用户能够清晰了解当前的播放状态。

### 3.2.5歌曲详情页

歌曲详情页是用户了解每首歌曲详细信息的主要途径。在该页面中，用户可以看到歌曲的封面图、歌手、专辑信息等。同时，用户还可以查看歌曲的相关评论、点赞数等互动信息，并通过该页面参与到社交互动中，发表评论或点赞自己喜欢的内容。歌曲详情页通过清晰的界面和直观的功能设计，使得用户能够便捷地获取每首歌曲的核心信息，并享受流畅的播放体验。

## 3.3社交方面

### 3.3.1动态发布

社交互动是本平台的重要特色之一。用户可以发布个人动态，分享自己对歌曲、生活等方面的想法。动态发布界面简单易用，用户只需输入文本内容，并可选择是否附带一首歌曲，便能完成动态的发布。动态可以是对某首歌曲的评论、对某种生活现象的分享，或者是任何能够表达用户个性和想法的内容。通过动态发布，平台增强了用户之间的互动性，提升了社交体验。

### 3.3.2动态推荐机制及详情页

为了提升用户体验并确保优质内容得到更多的关注，本平台采用了**点赞数优先**的展示机制，即系统会根据每条动态的点赞数进行排序，优先推荐那些点赞数较高的动态。这种做法能够有效地突出用户间最受欢迎的内容，让它们得到更多的曝光。

动态点赞数的排序，不仅能够帮助用户更快地发现平台中的热门内容，还能够激励用户参与互动，点赞自己喜爱的动态，推动社交圈的活跃度和互动性。

而动态详情页为用户提供了查看和参与互动的空间。在此页面中，用户可以看到动态的详细内容、发布时间、发布者的昵称及头像，以及附带的歌曲信息（如果有）。同时，用户还可以对该动态进行评论、点赞或收藏。动态详情页的设计旨在鼓励用户对其他用户的动态进行互动，使得整个社交体验更加生动和富有层次感。

### 3.3.3评论与收藏歌曲

作为社交互动的核心部分，评论和收藏功能使得用户能够与其他用户及平台内容建立更深的连接。用户可以在歌曲详情页下进行评论，分享对歌曲的感受。而收藏功能则让用户能够将自己喜爱的歌曲保存到个人收藏夹，方便随时聆听。这些社交功能增强了用户在平台内的存在感和互动性，形成了一个良性循环的社区。

### 3.3.4评论与点赞动态

评论和点赞功能是平台社交互动的重要组成部分，极大增强了用户与内容以及用户之间的联系。用户可以在动态下发表个人评论，分享对内容的看法，评论不仅仅是文字交流，还可以包含表情符号、图片等多样化的形式。每个动态的评论将显示评论者的用户名和评论时间，帮助形成更直观的互动反馈。

**点赞功能**则提供了更加简便的互动方式，用户可以通过点击点赞按钮表达对动态内容的支持。点赞数的实时更新让用户能够看到哪些动态受到最多关注，促进了热门内容的传播与讨论。用户也可以随时取消点赞，灵活管理自己的互动。

## 3.4界面交互设计

### 3.4.1页面切换

本应用通过简单明了的页面切换设计，确保用户在不同功能之间流畅过渡。从主界面到个人信息页面、歌曲播放界面再到动态发布页面，每个页面的跳转都十分自然，没有冗余的操作步骤。页面之间的切换通过滑动、点击等方式完成，用户可以在短时间内找到所需功能，并进行相应操作。

### 3.4.2底部导航栏

底部导航栏是用户快速访问平台各个核心模块的重要工具。通过底部的导航栏，用户可以快速切换至“首页”、“动态”以及“我的”页面。导航栏通过清晰的图标和简洁的文字指示，使得用户能够直观地了解每个模块的功能。底部导航栏不仅提高了应用的易用性，还增强了整体的用户体验感。

### 3.4.3侧边栏与隐藏框设计

在界面的交互设计中，侧边栏与隐藏框的使用显著提升了应用的空间利用率。侧边栏为用户提供了设置、帮助、个人信息修改、退出登录等功能项，用户可以通过点击按钮轻松开启侧边栏。此外，隐藏框的设计使得某些不常用的功能能够隐藏起来，避免界面过于繁杂，同时保证了常用功能的可访问性。

这些精心设计的交互元素使得应用的使用流程更加流畅，帮助用户快速上手并提高了操作效率。每个细节的优化都考虑到用户的需求和习惯，力求为用户提供一个无缝连接的社交体验。

## 3.5其他需求

### 3.5.1性能需求

性能需求是衡量应用程序能否满足用户需求的关键因素之一。在本平台中，尤其是在歌曲播放、动态加载和用户交互等高频操作场景下，系统必须具备高响应性和低延迟性。特别是对于移动端应用，资源的消耗和加载速度显得尤为重要。用户期待平台能够快速加载歌曲信息、流畅播放音乐而不出现卡顿、延迟或掉线现象。

为了确保这一点，平台在数据处理上需高效精简。后台服务在处理用户请求时需要具备足够的吞吐能力，以支持并发请求。此外，为了避免无意义的资源浪费，系统在后台应能及时清理缓存数据，并在网络环境不稳定时自动调整音频的质量，确保用户始终获得平稳的使用体验。

### 3.5.2安全性需求

随着信息泄露和网络攻击事件的频发，平台的安全性已经成为重中之重。用户的个人信息、账号数据以及支付信息等都可能成为攻击者的目标。因此，必须采取严格的安全措施，保障用户数据的隐私和应用的整体安全性。

首先，应用应对用户的密码进行加密处理，避免明文密码泄露。此外，针对动态发布、评论和点赞等社交行为，系统应当进行权限控制，确保用户只能够访问或修改属于自己的数据，并防止恶意用户通过破解方式获取管理员权限。

在此基础上，系统还需进行日常的安全监控和漏洞修复，确保平台在面对不断变化的网络安全威胁时，能够及时响应并修复潜在问题。

### 3.5.3可用性需求

可用性是系统设计中最为重要的目标之一。无论是复杂的功能设计，还是简单的用户界面，都必须考虑到如何提高用户的便捷性和体验感。为了确保平台具备高可用性，系统的设计需要简单直观、易于理解与操作。

首先，用户界面应当简洁清晰，所有核心功能应尽可能通过清晰的图标和简便的操作流程展示给用户。对于功能较复杂的部分，提供简要的提示信息或引导页面，以帮助用户快速熟悉操作流程。

其次，平台应当具备较强的容错性。即使在网络中断或其他突发情况发生时，系统仍应能保持一定的稳定性，并在恢复时自动同步之前未完成的操作或数据，避免因不可预见的错误导致用户数据丢失或操作失败。

此外，系统需要支持多设备之间的数据同步，保证用户在任何设备上登录时，都能够继续之前的操作和获取最新的内容，这对于提升平台的可用性尤为重要。

性能、安全性和可用性需求是构建一个优秀音乐平台的基石，满足这些需求不仅能增强平台的稳定性、提升用户体验，还能为用户提供一个更安全、高效、便捷的互动空间。

# 第四章 数据库设计

## 4.1系统概念结构设计

系统的概念结构设计主要是对整个系统的数据需求进行抽象，通过实体-关系（ER）图来描述系统中的各个实体及其相互关系。在本系统中，涉及的主要实体包括：用户、歌曲、动态、评论、点赞等，图4-1为本系统的E-R图。

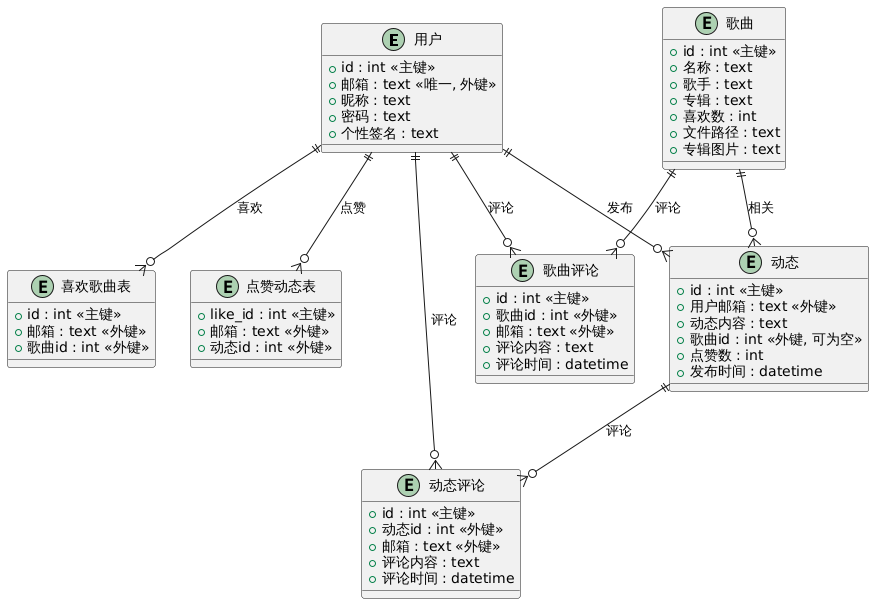


图4-1 系统E-R图

在系统的概念结构设计中，通过实体之间的外键关联，确保了数据的一致性和完整性。这些关系反映了系统的业务逻辑以及用户行为的各种交互。

## 4.2系统逻辑结构设计

在概念结构设计的基础上，系统逻辑结构设计进一步将实体关系模型转化为数据库表结构，定义了具体的数据类型、约束条件、主外键关系等。

表4-1 系统表清单

|  |  |
| --- | --- |
| 表名 | 用途 |
| Users（用户） | 存储用户基本信息 |
| Songs（歌曲） | 存储歌曲基本信息 |
| Dynamic（动态） | 记录用户发布的动态信息 |
| Comment（歌曲评论） | 用户对歌曲进行的评论 |
| Dycomment（动态评论） | 用户对动态进行的评论 |
| User\_likes（喜欢歌曲） | 记录用户对歌曲的收藏信息 |
| Dynamic\_likes（点赞动态） | 记录用户对动态的点赞信息 |

表4-2 用户表

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 字段描述 |
| id | integer | 4 | 用户ID,主键,自动增长 |
| email | text | 20 | 用户邮箱，唯一，外键 |
| userName | text | 20 | 用户昵称 |
| password | text | 20 | 用户密码 |
| profile | text | 50 | 用户个性签名 |

表4-3 歌曲表

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 字段描述 |
| id | integer | 4 | 歌曲ID,主键,自动增长 |
| name | text | 20 | 歌曲名称 |
| artist | text | 20 | 歌手名称 |
| album | text | 20 | 歌曲专辑名称 |
| likes | integer | 4 | 歌曲被喜欢数量 |
| filePath | text | 20 | 歌曲文件路径 |
| songImage | text | 20 | 歌曲图片路径 |

表4-4 动态表

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 字段描述 |
| id | integer | 4 | 动态ID,主键,自动增长 |
| UserEmail | text | 20 | 用户邮箱，外键 |
| content | text | 50 | 动态内容 |
| musicId | integer | 4 | 歌曲id，外键，可为空 |
| timestamp | datatime | 20 | 发布时间 |
| likes\_count | integer | 4 | 点赞数 |

表4-5 歌曲评论表

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 字段描述 |
| id | integer | 4 | 歌曲评论ID,主键,自动增长 |
| email | text | 20 | 用户邮箱，外键 |
| song\_id | integer | 4 | 歌曲id，外键 |
| comment\_text | text | 50 | 评论内容 |
| timestamp | datatime | 20 | 评论时间 |

表4-6 动态评论表

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 字段描述 |
| id | integer | 4 | 动态评论ID,主键,自动增长 |
| email | text | 20 | 用户邮箱，外键 |
| dynamic\_id | integer | 4 | 动态id，外键 |
| comment | text | 50 | 评论内容 |
| timestamp | datatime | 20 | 评论时间 |

表4-7 喜欢歌曲表

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 字段描述 |
| id | integer | 4 | 喜欢歌曲ID,主键,自动增长 |
| email | text | 20 | 用户邮箱，外键 |
| song\_id | integer | 4 | 歌曲id，外键 |

表4-8 点赞动态表

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 字段描述 |
| id | integer | 4 | 点赞动态ID,主键,自动增长 |
| email | text | 20 | 用户邮箱，外键 |
| dynamic\_id | integer | 4 | 动态id，外键 |
|  |  |  |  |

通过逻辑结构设计，我们不仅定义了每个表的字段类型和数据约束，还为表之间的关系（如外键）进行了详细的说明，确保数据的规范化管理。

## 4.3数据库实施与连接

在本项目中，我们使用了 SQLite 数据库来存储和管理应用程序的用户数据、歌曲信息、动态内容以及与之相关的评论和点赞信息。数据库实施包括数据库的创建、表结构的设计、索引的创建、存储过程的使用以及触发器的设置。以下是数据库实施的详细内容。

### 4.3.1创建数据库

数据库的创建是通过 SQLiteOpenHelper 类实现的，UserDatabaseHelper 类继承自 SQLiteOpenHelper，并重写了 onCreate 和 onUpgrade 方法。onCreate 方法会在数据库第一次创建时被调用，负责初始化所有的表和相关约束。

在创建数据库时，我们指定了数据库的名称为 musicAppDB，并设置版本号。当数据库版本更新时，onUpgrade 方法会自动执行，确保数据库结构与应用程序的需求保持一致。

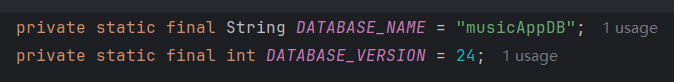


图4.2 数据库创建

### 4.3.2创建数据库表

数据库中的各个表用于存储不同类型的信息。下面是每个表的创建语句和其功能说明：

**（1）用户表 (**users**)**：存储用户的基本信息，包括邮箱、密码和个性签名等。

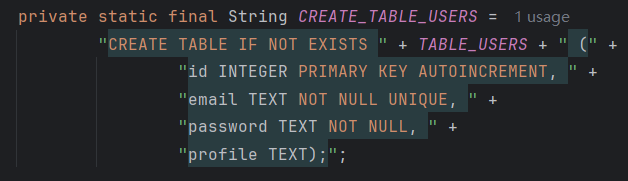


图4.3 用户表创建

1. **歌曲表 (**songs**)**：存储歌曲的信息，包括歌曲名称、歌手、专辑、歌曲文件路径以及专辑图片。

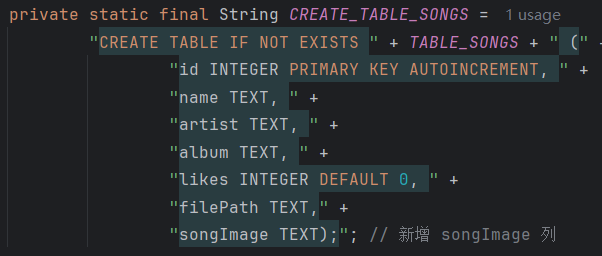


图4.4 歌曲表创建

1. **动态表 (**dynamic**)**：存储用户发布的动态内容，包括动态的文本内容、附带的歌曲（可为空）、点赞数及发布时间。

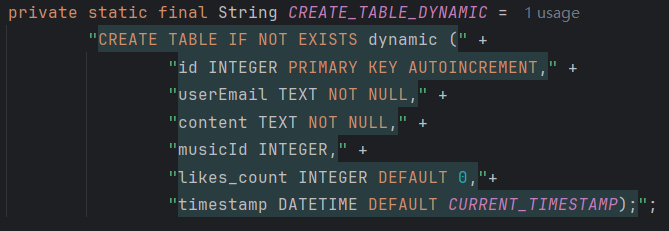


图4.5 动态表创建

1. **歌曲评论表 (**comment**)**：存储用户对歌曲的评论信息。

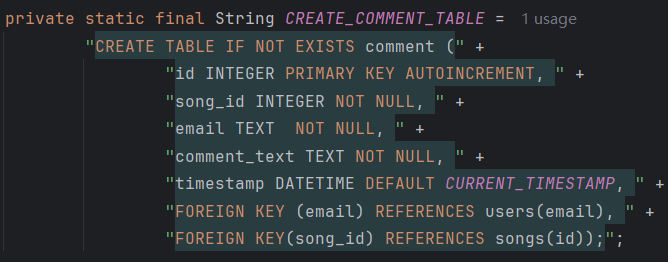


图4.6 歌曲评论表创建

1. **动态评论表 (**dycomment**)**：存储用户对动态内容的评论。

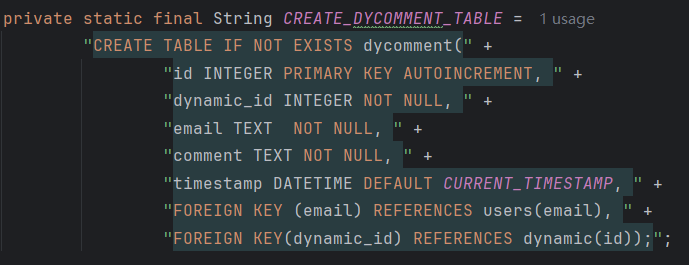


图4.7 动态评论表创建

1. 喜欢歌曲**表 (**user\_likes**)**：存储用户对歌曲的收藏信息，确保每个用户只能收藏一次同一首歌曲。

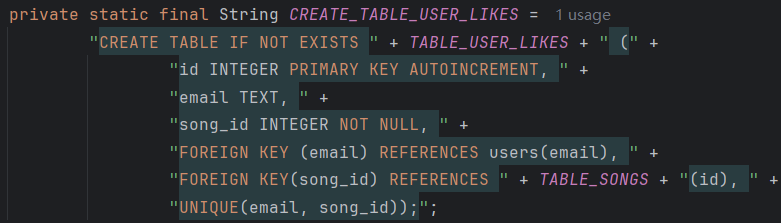


图4.8 喜欢歌曲表创建

1. **动态点赞表 (**dynamic\_likes**)**：存储用户对动态的点赞信息。

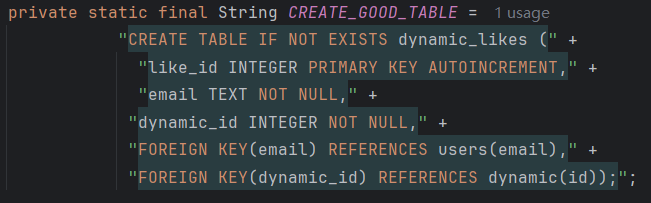


图4.9 动态点赞表创建

### 4.3.3数据库的连接

在本项目中，数据库连接采用了 Android 提供的 SQLiteOpenHelper 类来简化与 SQLite 数据库的交互。SQLiteOpenHelper 类主要负责数据库的创建、版本升级以及常见的数据库操作。通过继承这个类，开发者可以轻松地管理数据库的生命周期，从而避免了手动创建和管理数据库连接的复杂性。

数据库连接的实现从 SQLiteOpenHelper 子类开始。在这个子类中，我们重写了 onCreate() 和 onUpgrade() 方法。onCreate() 方法在首次创建数据库时被调用，其中包含了所有表的创建语句。onUpgrade() 方法则在数据库版本更新时被调用，主要用于处理数据库表结构的变化，比如添加新字段或表结构修改等。

在数据库连接的过程中，SQLiteOpenHelper 会根据需要自动调用 getWritableDatabase() 或 getReadableDatabase() 来获取数据库实例。调用 getWritableDatabase() 可以获取可读写的数据库对象，如图4-11所示。而 getReadableDatabase() 则返回一个只读的数据库实例。在获取到数据库实例后，开发者便可以利用该实例执行各类操作，如插入、查询、更新和删除数据。

当数据操作完成后，必须及时关闭数据库连接以释放资源，避免内存泄漏。在 Android 中，关闭数据库连接非常简单，只需调用 SQLiteDatabase 对象的 close()方法即可。通过这种方式，数据库连接的管理变得更加高效和可靠。



图4-10 数据库的连接

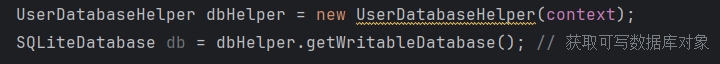


图4-11 数据库实例的获取

# 第五章 界面设计及功能实现

## 5.1界面设计

在本系统中，界面设计的核心目标是提升用户体验，使得用户能够方便地操作，同时确保界面的美观和易用性。每个模块的界面都进行了精心设计，力求达到直观、简洁和功能完备的效果。以下是各个主要界面的设计及实现描述。

### 5.1.1注册登录界面

注册和登录界面是用户使用本平台的入口，也是最重要的初始交互界面。为了简化用户操作，本界面提供了邮箱和密码的基本登录功能，并通过邮箱实现注册、修改密码等功能。注册流程要求用户提供有效的邮箱地址、密码信息。为了增强安全性，密码在输入时进行了加密存储，确保用户信息的安全。在登录成功后，用户能够顺利进入到首页界面。

登录及注册界面采用简洁的布局风格，背景图像和按钮设计均避免过多干扰，确保用户的注意力集中在注册和登录操作上。输入框采用清晰的标签和提示，确保用户可以直观地理解如何进行操作。



图5.1 桌面图标及开屏动画

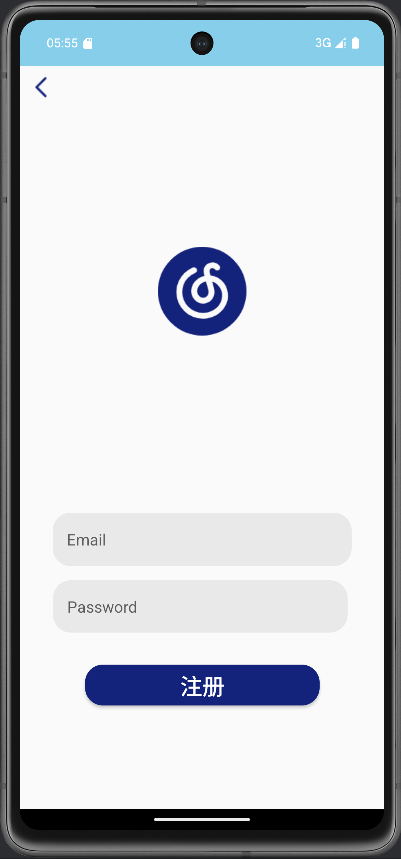
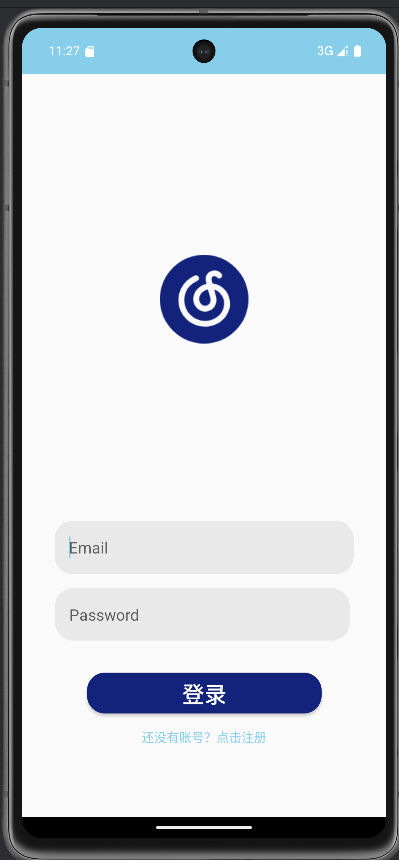


图5.2 登录及注册界面

### 5.1.2音乐界面

音乐界面是用户享受平台音乐内容的核心界面。在界面中，用户可以进行音乐的搜索、浏览、播放、暂停、切换等操作。界面中显示当前歌曲的封面、歌手、专辑信息，用户可以直接查看歌曲详情或者将歌曲添加到播放列表。在播放控制区域，提供了暂停、播放、上一首、下一首等功能按钮，保证用户的操作简单直观。

为了营造更好的音乐氛围，不仅在界面中采用了轮播图设计，并且界面背景和控件的设计都采取了柔和的颜色，避免了过于突兀的视觉冲击。按钮和图标采用了大尺寸设计，用户可以方便地进行触摸操作。



图5.3 首页及歌曲详情页

### 5.1.3动态界面

动态界面展示了平台内所有用户发布的动态，包括动态内容、附带的歌曲信息（可选）以及发布时间。点击每条动态后，用户会进入动态的详情页面，提供了点赞和评论按钮，用户查看详细的评论和点赞情况，并可以在该页面进行评论。

动态列表采用简洁的卡片式设计，每个卡片展示动态内容的简要信息。卡片之间有适当的间隔，避免视觉上的拥挤感。动态内容和图片信息被清晰分割，使用户可以迅速浏览并参与互动。



图5.4 动态页及动态详情页

### 5.1.4个人界面

个人界面是用户查看和编辑自己信息的地方。用户可以在此界面查看自己的昵称、头像、个性签名等基本信息，并有修改和更新的选项。该界面还展示了用户发布的动态、评论和喜欢的歌曲等社交互动信息。为了方便用户管理自己的个人内容，界面还提供了清晰的导航栏，让用户能够快速切换到各个模块。

个人页面设计以用户为中心，信息布局层次分明，用户可以一目了然地看到自己的个人资料、互动信息以及更多功能。头像和背景图等元素得到了精心设计，呈现出较高的视觉美感。



图5.5 个人界面

## 5.2前后台连接

在本系统中，前端界面通过与内置的 SQLite 数据库连接，实现在设备本地存储和管理数据的功能。SQLite 是一种轻量级的数据库，能够在没有网络连接的情况下高效存储数据，因此非常适合用于 Android 应用中。为了实现与数据库的连接，我们在项目中使用了 SQLiteOpenHelper 类，这个类封装了数据库的创建、升级和版本管理操作，从而简化了数据库的管理。

在应用启动时，UserDatabaseHelper 类被用来初始化数据库连接。这个类继承自 SQLiteOpenHelper，并通过它的构造方法接受一个 Context 对象。通过这种方式，数据库的实例得以在后台创建，并且系统能够自动管理数据库的版本控制。当数据库首次创建时，onCreate() 方法被调用，它负责执行数据库表的创建。如果数据库版本更新，onUpgrade() 方法会被触发，从而进行结构的升级，例如新增字段或更改表结构。

数据库操作主要通过 SQLiteDatabase 类来执行。无论是用户登录、点赞、评论，还是动态的发布、评论和点赞等操作，都会涉及到与数据库的交互。在前端界面中，用户触发某些行为时，应用会通过 SQLiteDatabase 执行相应的 SQL 查询或更新操作。比如，当用户登录时，应用会根据用户输入的邮箱和密码查询数据库中的记录，验证其是否存在。如果用户对某首歌曲进行点赞，系统则会在数据库中更新歌曲的点赞数，并通过查询更新后的数据实时显示在界面上。

数据的读取和写入是通过 Cursor 类来完成的，Cursor 类是一个游标，用来处理查询结果集。在查询数据库时，Cursor 会指向每一行数据，前端界面通过读取这些数据来更新显示。当用户查看个人收藏的歌曲时，应用会查询 user\_likes 表，并展示出所有已收藏的歌曲。对于动态发布功能，用户提交的内容会被插入到 dynamic 表中，随后前端界面会即时更新显示，反映用户的最新动态。

所有数据库操作都是通过 SQLiteDatabase 进行的，保证数据的一致性与完整性。在执行写操作时，系统会使用事务来确保操作的原子性和一致性，避免在出现异常时导致数据丢失或不一致。此外，系统在执行查询时也进行了优化，采用了合适的索引和查询策略，确保在数据量较大时也能快速响应用户的操作。通过这种设计，前端界面能够与 SQLite 数据库无缝连接，实时读取和更新数据，从而为用户提供流畅、即时的操作体验。

## 5.3功能实现及截图展示

### 5.3.1注册登录功能

在注册与登录功能中，系统通过邮箱和密码实现用户的身份验证，确保用户信息的安全。注册时，用户需要提供唯一的邮箱和有效的密码，在后台进行数据验证，验证成功会显示注册成功，如图5.6所示。并且系统提供了多种安全检测，如图5.7为输入错误以及输入为空的情况。登录成功后，用户进入系统首页。

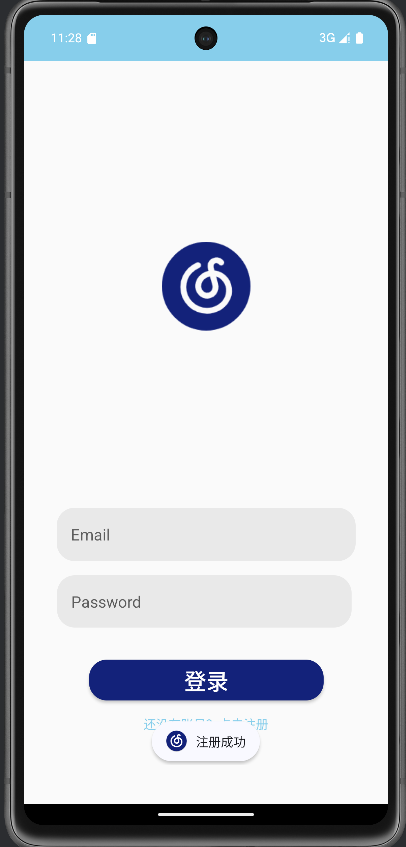


图5.6 注册功能

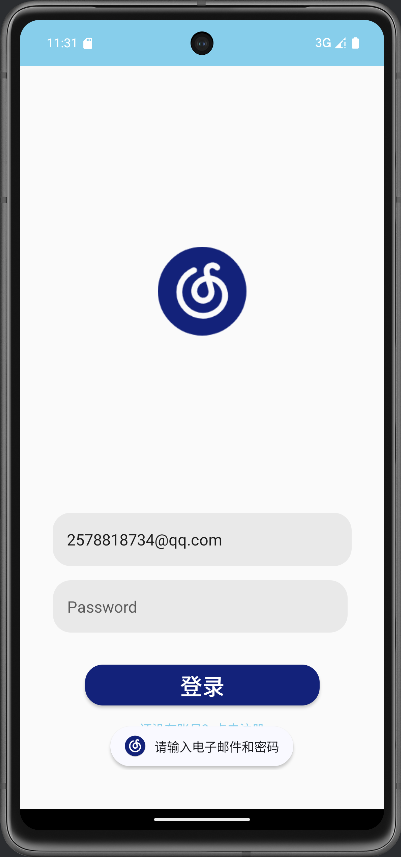


图5.7 登录验证功能

### 5.3.2搜索功能

搜索功能允许用户按歌曲名、歌手或专辑等关键词进行查找，快速找到自己感兴趣的音乐内容。搜索框设计简单直观，用户只需要输入关键词即可获得相应的搜索结果。

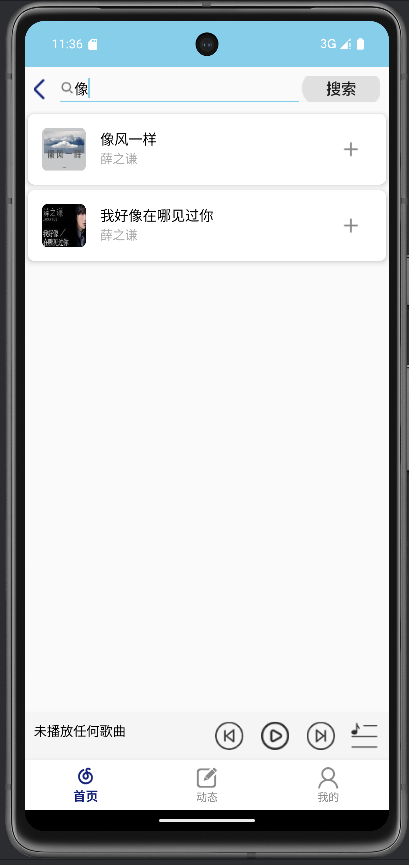


图5.8 歌曲搜索功能

### 5.3.3歌曲收藏及评论功能

用户可以点击爱心按钮将自己喜欢的歌曲添加到收藏夹，并在歌曲详情页发表评论，不仅如此，用户还可以在歌曲详情页查看歌曲的封面、歌手信息和专辑名称。如图5.9所示。而且每条评论会显示评论时间、评论内容以及评论者的昵称，增强了平台的社交互动性。



图5.9 歌曲收藏及评论功能

### 5.3.4歌曲播放功能

歌曲播放功能使得用户能够方便地听歌，用户可以点击歌曲旁边的添加按钮将歌曲添加到播放列表，如图5.10所示，通过播放控制按钮允许用户随时暂停、播放、切换歌曲，并且点击播放列表按钮，用户可以查看或删除当前列表的歌曲，以及切换播放模式等操作，享受更丰富的音乐体验。如图5.11，当前播放模式为随机循环，会按随机顺序播放歌曲。



图5.10 添加歌曲到播放列表功能



图5.11 歌曲播放及切换功能

### 5.3.5发布动态功能

用户可以在动态页面发布自己的心情或分享自己喜欢的歌曲。发布动态时，用户可以选择是否附带音乐，并可自由编辑文字内容，如图5.12所示。动态发布后，其他用户可以查看并进行评论或点赞。



图5.12 动态发布功能

### 5.3.6动态点赞及评论功能

用户可以点击动态页的动态进入其详情页，对发布的动态进行点赞和评论，如图5.13所示，对5.3.5中发布的动态进行了点赞和评论操作。通过这种社交互动提升平台的活跃度。每条动态下方都会显示点赞数和评论，增加了内容的互动性。

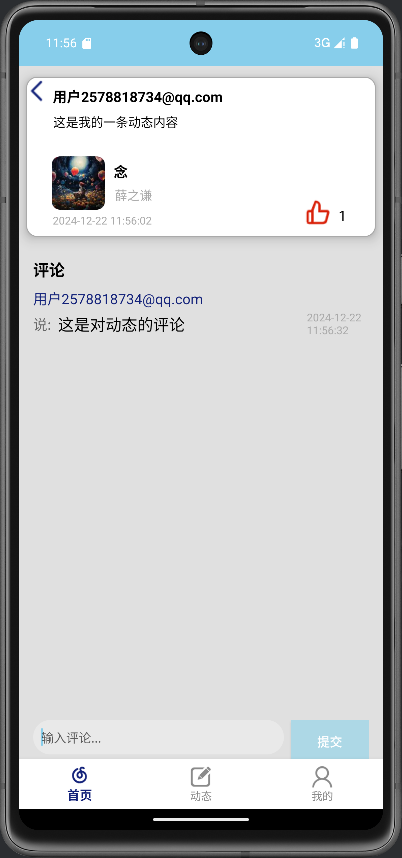


图5.13 动态点赞及评论功能

### 5.3.7个人信息修改功能

在个人信息页面，用户可以修改昵称、个性签名、头像等个人资料，如图5.13是修改个性签名以及用户昵称的示例操作。这些信息的修改会实时更新到数据库，并同步到前端界面，确保展示的是最新的用户资料。



图5.14 个人信息修改功能

### 5.3.8社交信息查看及删除功能

用户可以查看自己发布的动态、评论和收藏的歌曲等社交信息。如果用户不再需要某条动态或评论，也可以长按删除它们，确保个人信息的隐私性和管理便利。



图5.15 社交信息查看功能

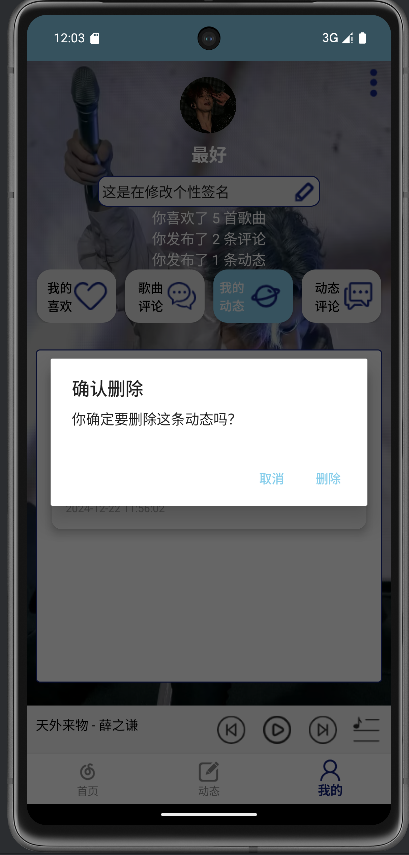
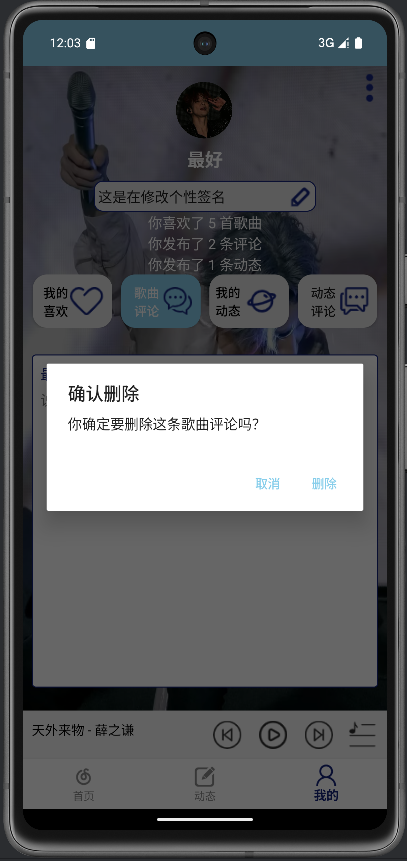


图5.16 社交信息删除功能

# 第六章 总结

## 6.1项目特色

本项目在设计和功能实现上具备多个独特的特色，致力于为用户提供更加个性化、互动性强且智能化的音乐体验。

首先，项目支持多种播放模式的切换，包括顺序播放、单曲循环和随机播放，用户可以根据个人需求灵活选择。无论是在听歌过程中希望单曲反复播放，还是想要随机享受不一样的歌曲体验，系统都能满足不同场景下的需求。通过这种灵活的播放模式切换，提升了用户的使用舒适度和个性化体验。

其次，社交互动是本项目的另一大亮点。用户不仅可以进行歌曲播放和管理，还能通过发布动态、评论歌曲以及点赞等功能与其他用户进行互动。动态功能允许用户分享自己的音乐心情，评论和点赞则提供了与其他用户进行情感交流和互动的渠道。社交元素的加入大大提升了用户的粘性，使得平台不仅是一个音乐播放器，更成为了一个互动性的社交平台。

另外，本系统的音乐推荐功能也非常突出。基于用户的行为和偏好，系统会推荐更多人喜欢的歌曲，从而提高用户粘性，创造个性化的音乐体验。这种智能推荐机制使得用户能够发现更多自己感兴趣的内容，增加了用户与平台之间的互动和留存。

除此之外，项目使用了安卓平台的 MediaPlayer 进行本地歌曲播放，确保了音频播放的高效性和稳定性。通过对歌曲播放过程的细致控制，系统能够实现无缝切换、播放列表管理以及丰富的播放模式切换，极大地提升了用户的使用体验。

## 6.2项目不足

当前系统虽然完成了基本的音乐播放和社交互动功能，但仍存在一些不足之处，影响了用户体验和系统的扩展性。首先，系统目前只能播放本地歌曲，无法支持在线流媒体音乐播放。随着用户对平台的依赖加深，在线音乐的需求越来越强烈。用户希望能通过平台访问更多的音乐资源，例如云端音乐库和流媒体服务。因此，无法播放在线歌曲成为了系统的一大局限。

另外，系统的社交功能相对基础，虽然提供了评论、点赞等基本互动方式，但缺乏更多形式的社交交流。例如，缺少私信、朋友圈、动态分享等功能，导致用户之间的互动较为单一，平台的社区氛围也未能充分形成。同时，推荐系统较为简单，无法根据用户的历史行为和偏好智能推荐音乐或动态内容，造成用户体验缺乏个性化和互动性。

## 6.3改进方向

未来的改进将集中在多个领域，以提升系统的功能和用户体验。首先，系统需要支持在线音乐播放。通过接入流媒体平台或构建自己的音乐库，平台将能够为用户提供丰富的在线音乐资源，打破仅能播放本地音乐的局限。这不仅能提升用户体验，还能增强平台的竞争力。

此外，社交功能的拓展也是未来改进的关键。平台可以引入私信、朋友圈、动态分享等社交互动功能，进一步增强用户间的互动性和粘性。通过丰富的社交功能，平台可以提升社区氛围，增加用户的活跃度和归属感。

为了提升用户体验，系统还需要引入更智能的推荐算法。目前，系统的推荐机制较为基础，缺乏根据用户偏好、行为习惯进行智能推荐的能力。未来可以通过引入机器学习和深度学习技术，为用户提供个性化的音乐推荐、动态推荐和社交互动建议。这将大大增强平台的智能化水平和用户体验。

另外，引入**AI大模型**也是未来的一项重要改进方向。通过结合大数据和AI技术，可以分析用户的听歌历史、评论互动等信息，精准预测用户的兴趣并提供高度个性化的推荐服务。AI大模型还可以在内容审核、评论智能生成、社交互动等方面提供更多智能支持，为用户带来更具创新性和沉浸感的体验。

最后，为了适应更大规模用户的需求，系统的架构也需要进行优化。随着用户量的增加，当前的架构可能面临扩展性问题，未来可以考虑将一些功能迁移到云端，支持跨设备的数据同步，减轻客户端的负担，提高系统的响应速度和处理能力。

通过这些改进，平台将能够提供更丰富的功能、更流畅的用户体验，并提升系统的智能化和个性化水平，进一步增强用户粘性和平台的竞争力。

参考文献

[1]许瑾.基于Android平台音乐播放器的设计与实现[D].北京邮电大学,2011.

[2]胡晓东.基于android平台音乐播放器的设计与实现[D].吉林大学,2014.

[3]崔毅明.基于Android平台的多功能音乐播放器的设计与实现[D].吉林大学,2015.

[5]苏航.基于Android的音乐播放器软件的设计与实现[D].电子科技大学,2016.

[6]李玲玲.基于Android平台在线音乐播放器的研究与实现[D].安徽理工大学,2012.

[7]陈振宇.基于Android的手机音乐播放器的设计与开发[D].西安电子科技大学,2014.

项目组成员分工及贡献度

|  |  |  |
| --- | --- | --- |
| 成员 | 分工 | 贡献度 |
| 鱼洲（组长） | 项目开发、数据库设计、PPT制作、  讲解视频录制、项目报告撰写 | 100% |
| 录浩 | 数据库设计、项目报告撰写 | 95% |
| 戴燊 | PPT制作、界面设计 | 90% |
| 焦琳通 | PPT制作、界面设计 | 90% |