草木诗经（Plants In Books of Songs）
《Android 技术开发基础》开发文档

1 App 的运行与开发环境

（1） **运行环境：** 10.0 以上版本 Android 的 Android 手机
（2） **部署方法：** 直接安装草木诗经.apk 即可
（3） **开发环境：** Android Studio Hedgehog | 2023.1.1 Patch 2
（4） **手写代码行数：**手机端约 4366 行（Kotlin）

2 App 功能说明：

App 图标如下。

图片 1 App 图标展示

APP 总共有四个主要的页面，分别集成了四方面不同的功能。接下来将根据每一个页面介绍每个页面具体功能。

（1）“遇见”页面

此页面为主页面。实现的功能主要是为用户推荐诗经中的植物介绍卡片和收藏卡片功能。

在此页面中，会显示有关诗经中的植物的介绍卡片，卡片中，介绍了一种诗经中的植物及其出现的诗句。用户可以点击“换一换”更换推荐的植物。

图片 2 遇见页面展示

用户也可以点击“收藏”，将此卡片加入到用户收藏列表中。

图片 3 遇见页面收藏功能展示

用户也可以点击“分享”，将此卡片截图分享到其他 APP，此功能因为时间不足，尚未具体实现，只呈现了 UI 界面。

用户也可以点击“喜欢”，增加该卡片的喜爱度，此功能因为时间不足，尚未具体实现，只呈现了 UI 界面。

（2）“发现”页面

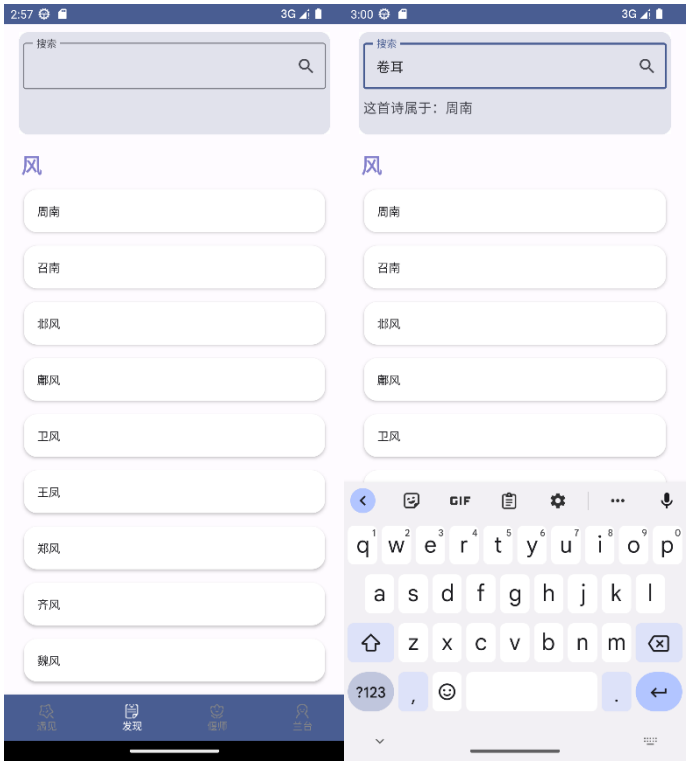
此页面实现的功能主要是为用户提供诗经原文阅读目录及搜索诗经中诗目具体位置的功能。

图片 4 发现页面第一级页面展示

目录总共有两级，页面共有三级，第一级目录包含风、雅、颂等诗经的类别，同时也为

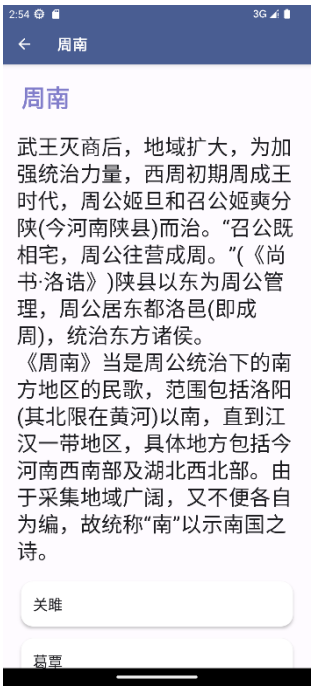
第一级页面；第二级目录包含诗经的具体类别的信息介绍和每一类别包含的诗目，同时也为第二级页面；第三极页面为诗目的诗句展示。

第一级页面中，用户可以使用搜索功能，查询一首诗具体属于哪个类别。



图片 5 发现页面搜索功能展示

第二级页面中，用户可以阅读了解该类别的具体信息。



图片 6 发现页面第二级页面展示

第三级页面中，用户可以点击悬浮按钮切换含汉语拼音的版本。

图片 7 发现页面第三级页面展示及拼音功能展示

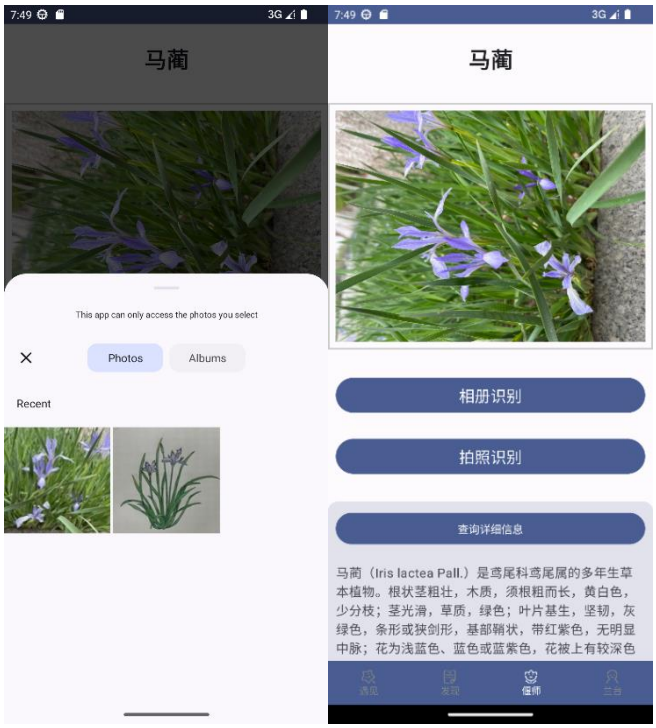
由于时间有限，诗经诗目共有 305 首，需录入和处理的资料又很多，APP 此处只录入了周南的具体诗目和关雎的拼音版本为例展示。

（3）“偃师”页面

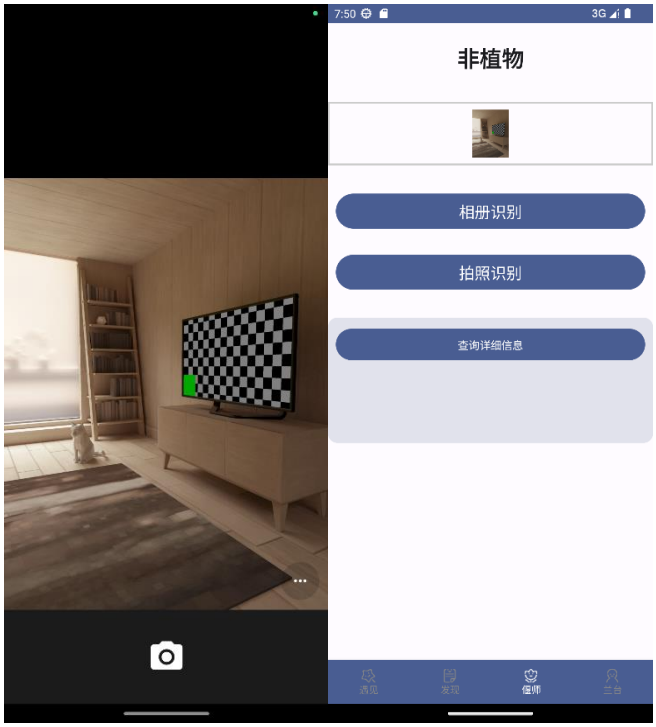
此页面实现的功能主要是为用户提供植物识别功能。用户可以选择相册图片识别或者拍照识别，识别成功后，会直接显示识别植物的类别。

图片 8 偃师页面展示

此时，用户可以选择查询详细信息，该类植物的详细信息会显示在卡片里。



图片 9 偃师页面相册识别功能及查询信息功能展示

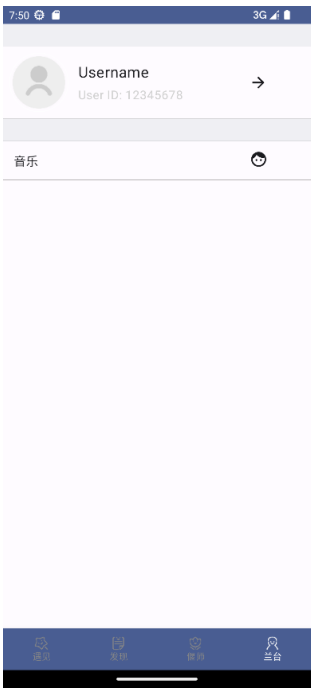


图片 10 偃师页面拍照识别功能展示

（4）“兰台”页面

此页面是个人页面，当用户未登录时，显示的是游客页面；当用户已登录时，显示的是用户页面。实现的主要功能是注册、登录、退出登录，以及登录后查看个人信息、查看个人收藏列表、背景音乐设置等功能。

用户未登录时，游客页面如下，此页面中可点击右上角箭头进入登录页面进行注册登录。

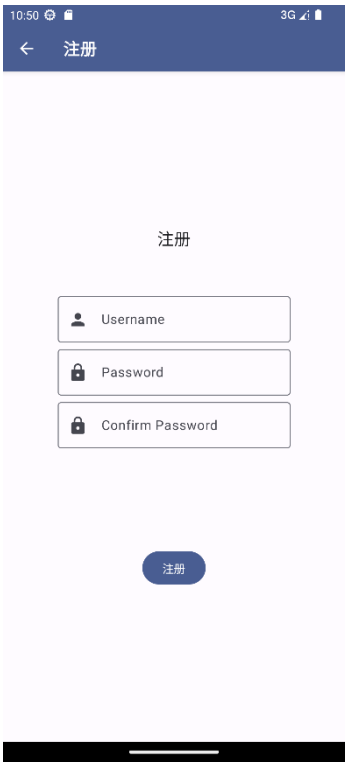


图片 11 游客页面展示

登录页面如下。注册后用户可在此页面登录。未注册用户可点击“去注册”进入注册页面进行注册。

图片 12 登录页面展示

注册页面如下。用户可在此页面注册，注册成功后用户自动进入登录页面。



图片 13 注册页面展示

用户登录后，用户页面如下。此页面中，用户可点击不同选项使用不同功能。

图片 14 用户页面展示

点击“音乐”，可以开始和暂停背景音乐。

点击“收藏”，可以进入收藏页面，查看收藏的卡片列表。

图片 15 收藏列表展示

点击“设置”，可以进入个人信息页面。个人信息页面中，点击“退出登录”，确认后可退出登录。

图片 16 个人信息页面及退出登录功能展示

3 App 架构设计及技术实现方案

程序整体由 PlantUML 插件生成的 UML 图如下。

图片 17 整体 UML 图

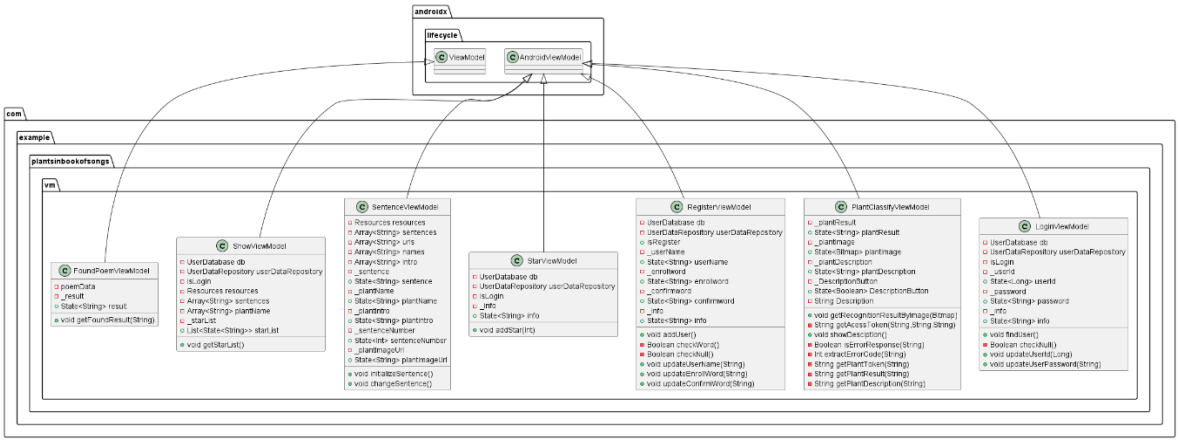
程序共分为十四个包，每个包中程序的解释及 UML 图如下。

（1）Screen

Screen 包中为四个主要页面以及其他页面的页面代码。

（2）vm

Vm 包中为各个页面的 ViewModel，UML 图如下。

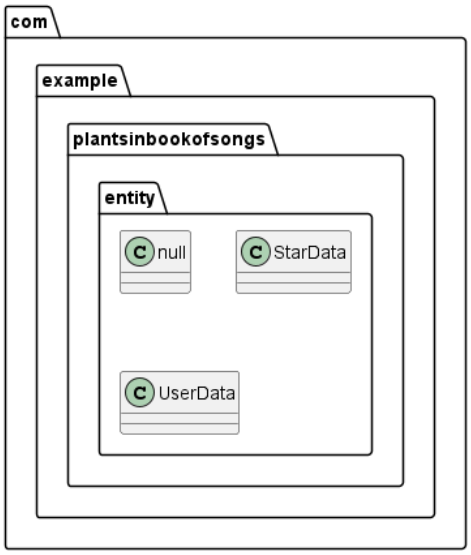


图片 18 vm 包 UML 图

（3）entity、dao、db、repository、helper

这五个包主要是用 Room 来实现数据库的创建、使用和测试。

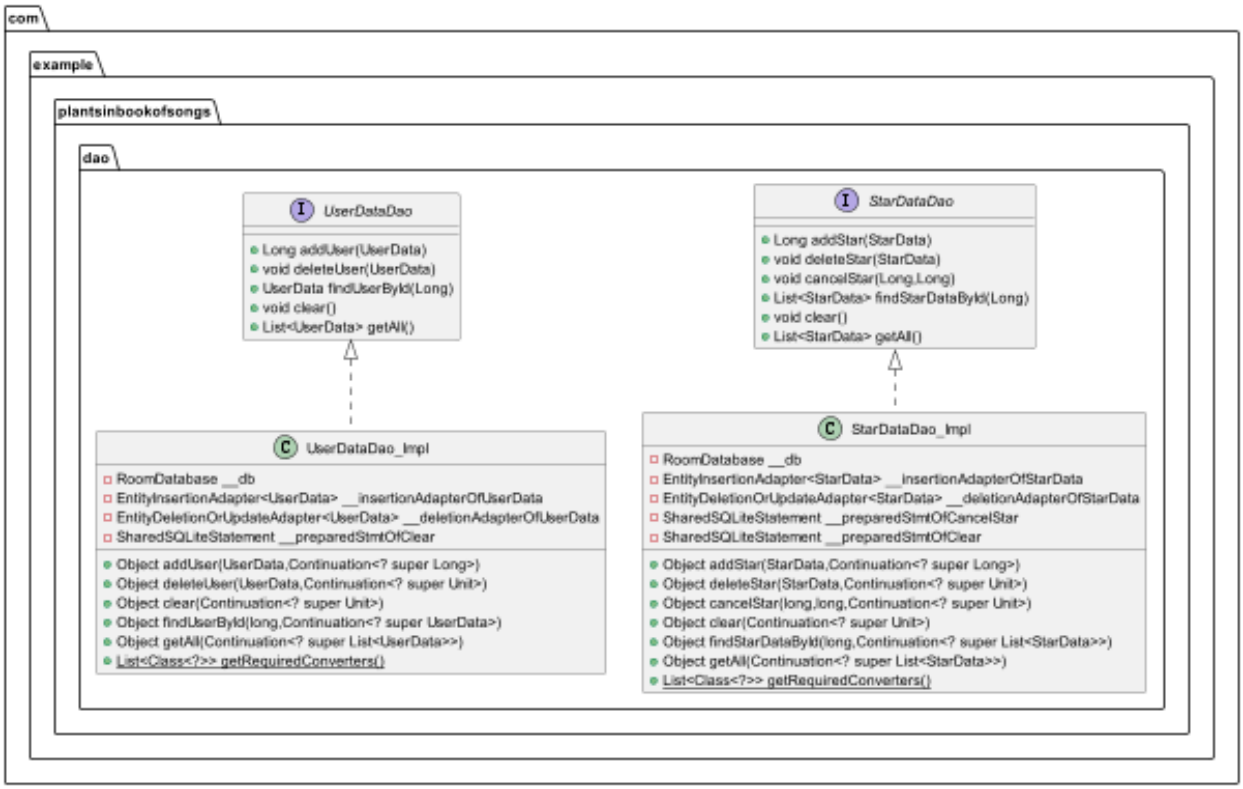
其中，entity 中存放两张表，user 表和 star 表。UML 图如下。



图片 19 entity 包 UML 图

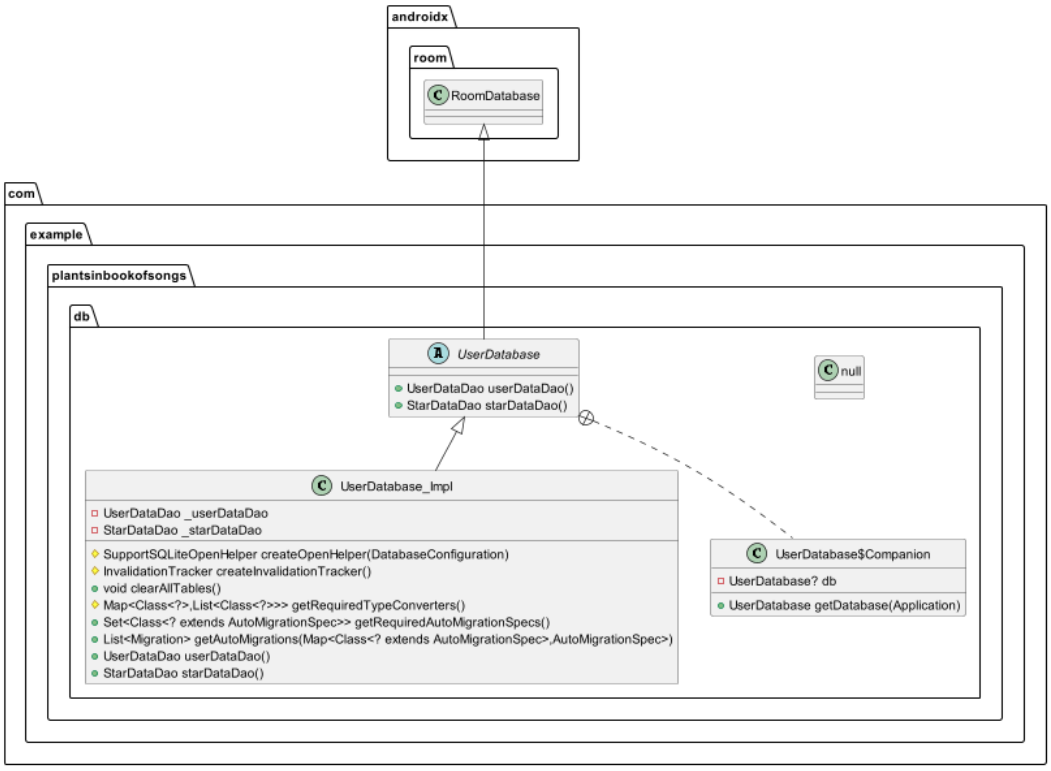
Dao 中的 dao 接口用来将数据库语言对表的操作转换为 Room 注解的函数。UML 图如

下。



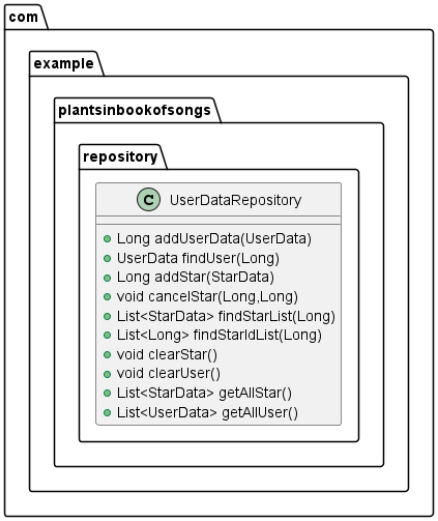
图片 20 dao 包 UML 图

Db 用来创建存放数据库 Database。UML 图如下。



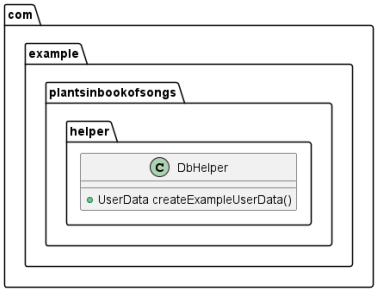
图片 21 db 包 UML 图

Repository 用来封装数据库操作，UML 图如下。



图片 22 repository 包 UML 图

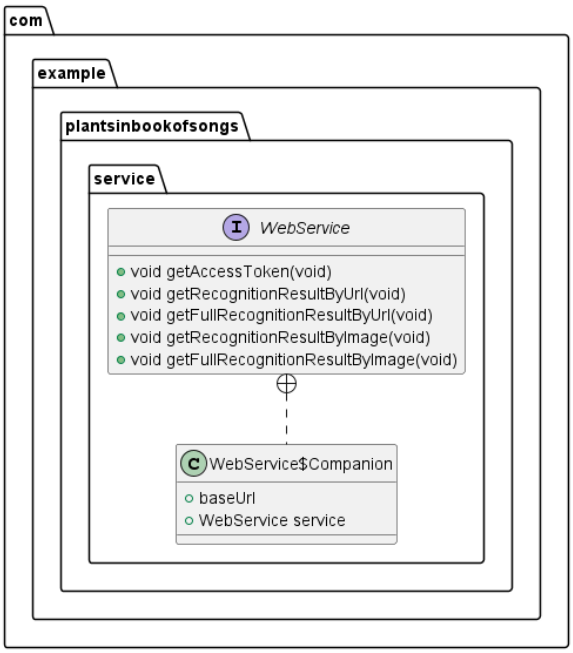
Helper 用来生成数据库测试时测试数据库的数据，UML 图如下。



图片 23 helper 包 UML 图

（4）service

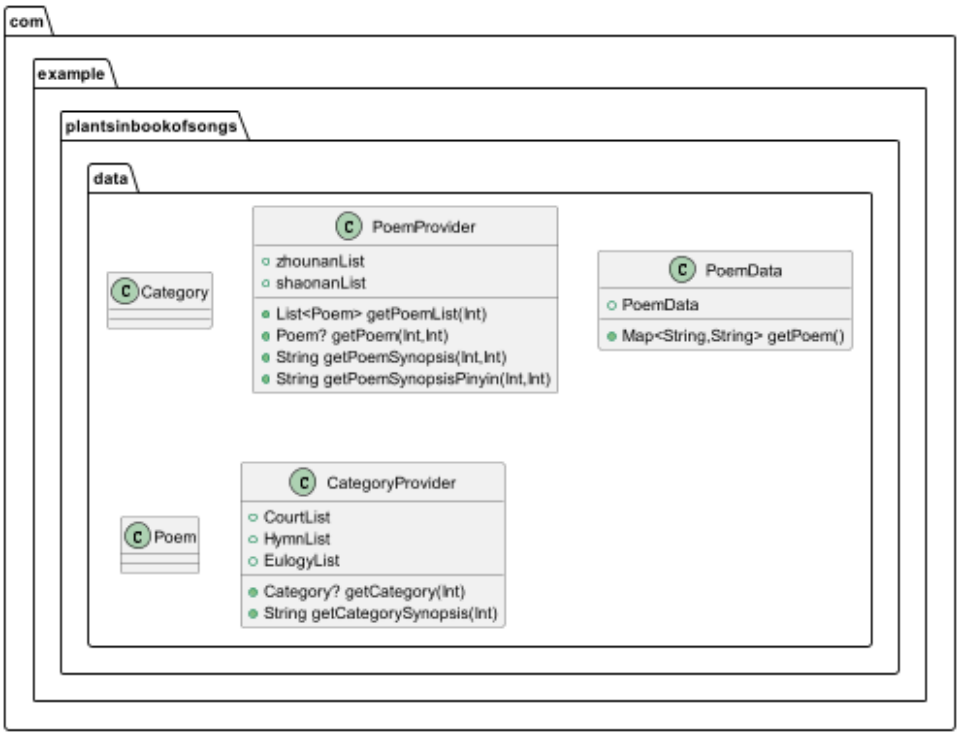
Service 用来提供网络服务，这里主要用来提供百度植物识别 api 的 Access_Token 获取和植物识别信息获取。UML 图如下。



图片 24 service 包 UML 图

（5）data

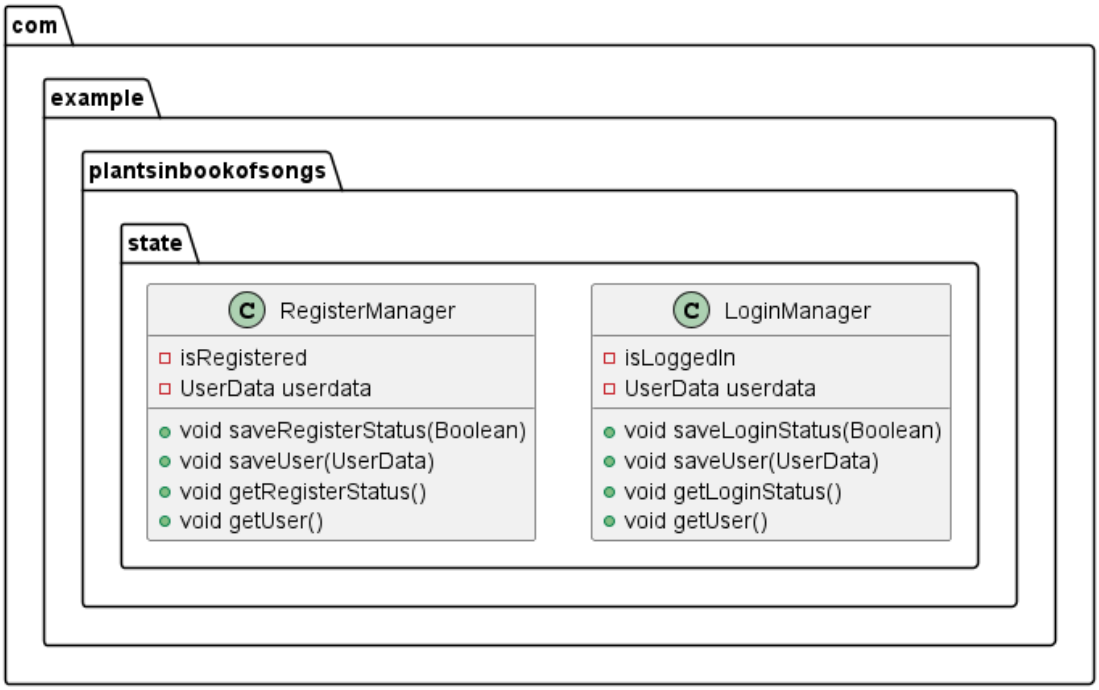
Data 中主要提供发现页面（FoundScreen）中的诗经浏览页面的列表数据。UML 图如下。



图片 25 data 包 UML 图

（6）state

State 中主要实现登录和注册状态的记录。UML 图如下。

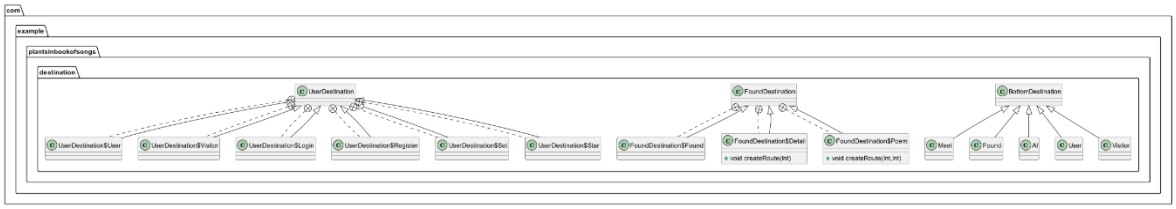


图片 26 State 包 UML 图

（7）nav、destination

这两个包主要用来实现导航功能，destination 记录不同页面的导航类，nav 中为底部导

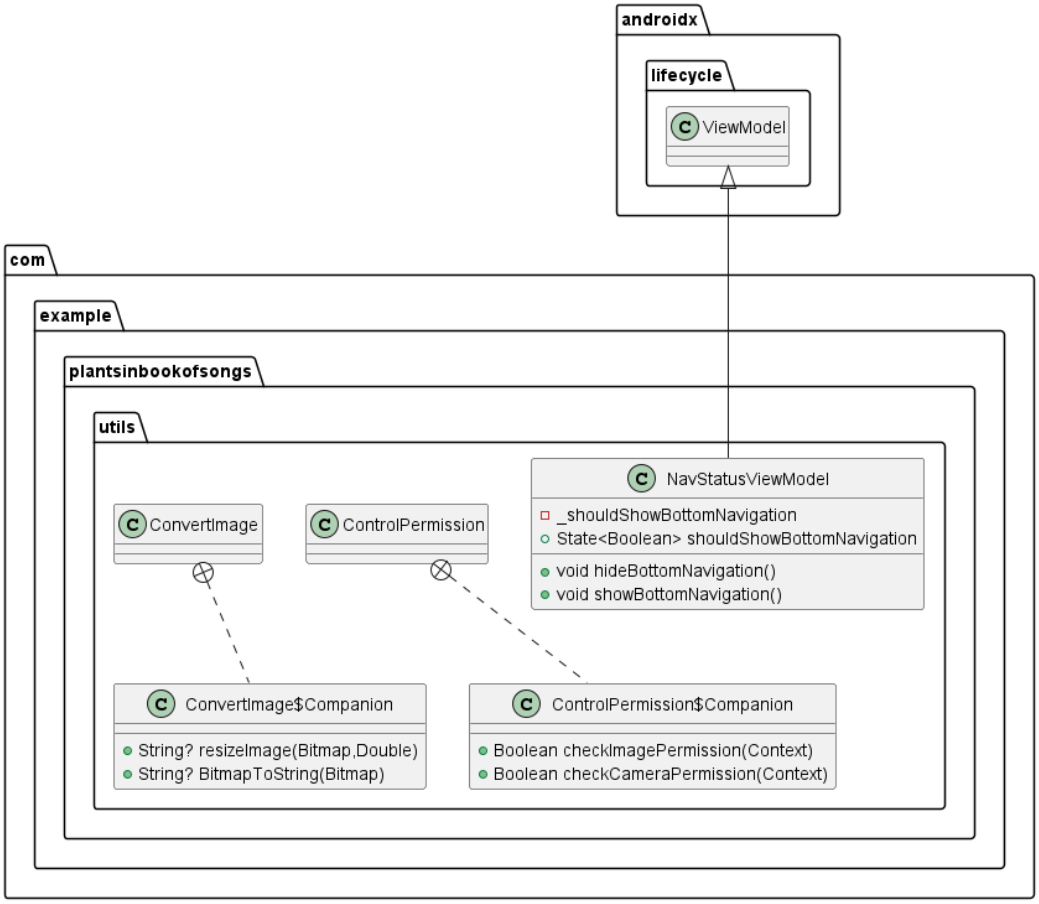
航。UML 图如下。



图片 27 destination 包 UML 图

（8）utils

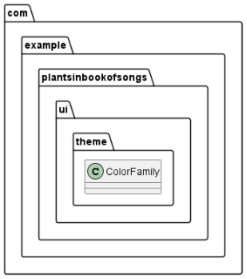
Utils 中主要包含了一些工具，如图片 bitmap 和 base64 编码压缩转换的工具、检测是否取得权限、密码 md5 编码、底部导航是否显示等。UML 图如下。



图片 28 utils 包 UML 图

（9）ui.theme

Ui 包中主要是 App 的主题设计，在此使用的是 Compose 官方的主题构建器工具进行设计的紫色主题。UML 图如下。



图片 29 ui 包 UML 图

4 技术亮点、技术难点及其解决方案

- 与其他类似的 App 相比，你这个 App 有哪些与众不同之处？

与其他 App 相比，App 的与众不同之处在于选题独特——诗经中的草木。在当前的应用市场上，虽然关于古诗文和关于植物的 App 很多，但关于诗经的 App 很少，几乎没有，而关于诗经中的植物的 App 是没有的，这是一个小而独特的点。

另外，与其他同类型植物识别和诗文等 App 相比，App 整体是用 Android Compose 实现的，整体上使用的技术更新，App 的页面设计更加简洁美观，操作也更加简单方便。

- 你这个 App 中，你认为最得意的地方在哪里？

这个 App，我最满意的地方首先是使用了百度植物识别的 api 实现了植物识别功能。先是用百度的官网注册了账号和 App，用 App 的 client_id 和 client_secret 获得了 Acess_Token，然后使用了课上讲的 Retrofit，根据百度植物识别 api 给出的示例代码进行编写 Webservice，最后成功的集成了植物识别功能和植物详细信息获取功能。

同时，在实现植物识别功能中的相册识别和拍照识别时，我使用了 Compose 的 rememberLauncherForActivityResult 进行获取相机权限和相册图片，Compose 的这个类用于获取相册图片和获取拍照权限时更加方便便捷。

另外，是用 Room 和数据库自主实现了登录注册和收藏功能的集成。在网络上我浏览了很多其他 App 的登录注册功能的实现，大多数都只实现到做一个登录注册功能，不能满足本 App 中对已登录的用户能进行收藏植物信息卡片的需求。所以，我使用了老师上课讲过的 Room，在数据库中建立了用户 User 表和收藏 Star 表，其中，User 表以 Star 表的主键为外键，从而实现了用户与收藏卡片的关联，这是收藏功能实现的关键。然后，用 Room 进行数据库操作，并使用 Repository 内的挂起函数进行存取数据，进而实现注册、登录、收藏功能。

- 你开发这个程序中遇到了哪些具体困难，又是如何解决的？

（1）图像识别传输问题

百度通用图像识别error_msg":"inputoversize""error_code":216205

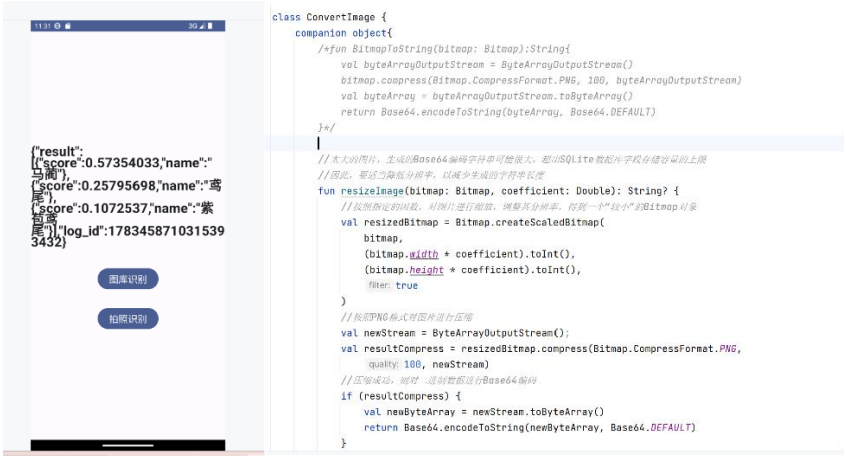
图片 30 百度植物识别传输图片过大

问题描述：

使用百度植物识别 api 时，传输的图片分辨率过高，转换的 base64 字符串过长，造成 Input Oversize 问题。

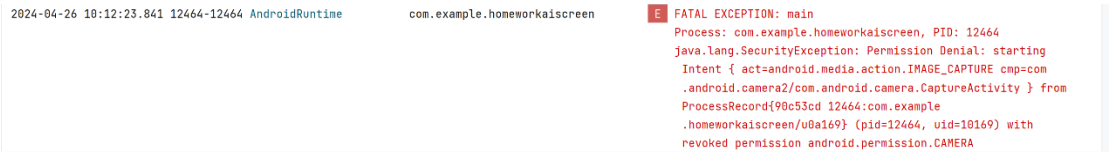
最终的解决方案：

编写根据图片大小降低图片分辨率的函数，减短 base64 字符串长度。



图片 31 降低图片分辨率问题解决

（2）拍照权限问题



图片 32 获取拍照权限报错

问题描述：

使用 ActivityResultContracts.TakePicturePreview()发出拍照请求时，显示无拍照权限。

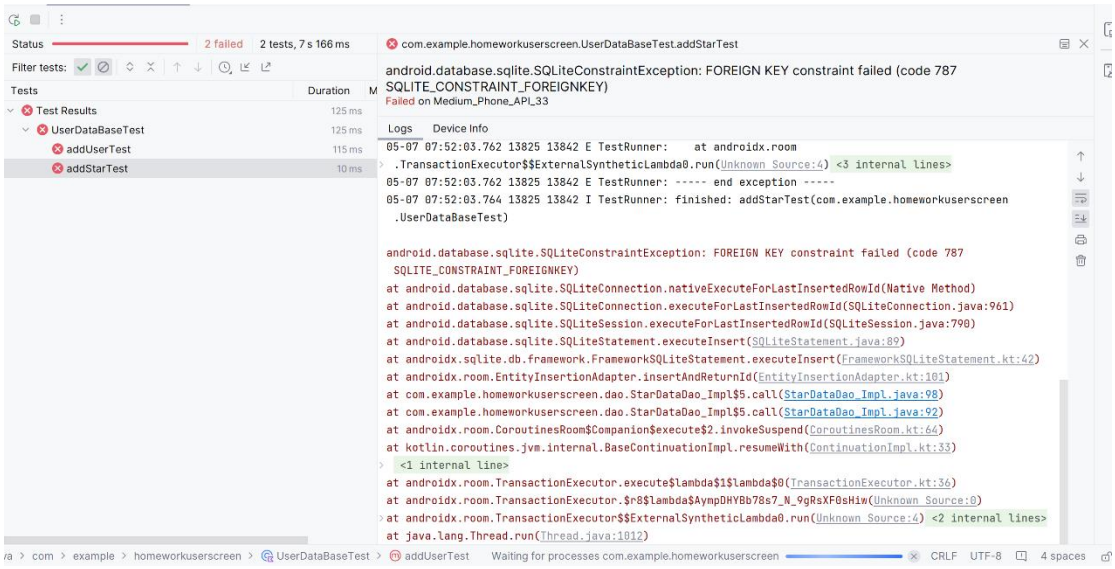
最终的解决方案：

编写根据图片大小降低图片分辨率的函数，减短 base64 字符串长度。先使用 ActivityResultContracts.RequestPermission()在第一次使用 app 时获得拍照权限，再使用 ActivityResultContracts.TakePicturePreview()发出拍照请求。

```
val permissionLauncher= rememberLauncherForActivityResult(
    contract = ActivityResultContracts.RequestPermission() ,
    onResult = { it: Boolean
        if(it){
            log( msg: "have permission")
            log( msg: "----------------------------------------------------")
        }else{
            log( msg: "no permission")
            log( msg: "----------------------------------------------------")
        }
    })
```

图片 33 先进行权限请求再进行拍照请求

（3）数据库外键约束问题



图片 34 外键约束问题

问题描述：

使用一个数据库建立两张表，star 表中以 user 表的主键 userId 为外键，进行数据库测试时，测试 Insert 功能时，显示不符合外键约束条件。 后重读代码，发现是外键的约束要求 user 表中必须已存在 userId，才能在 star 表中插入以该 userId 为外键的 star 数据。

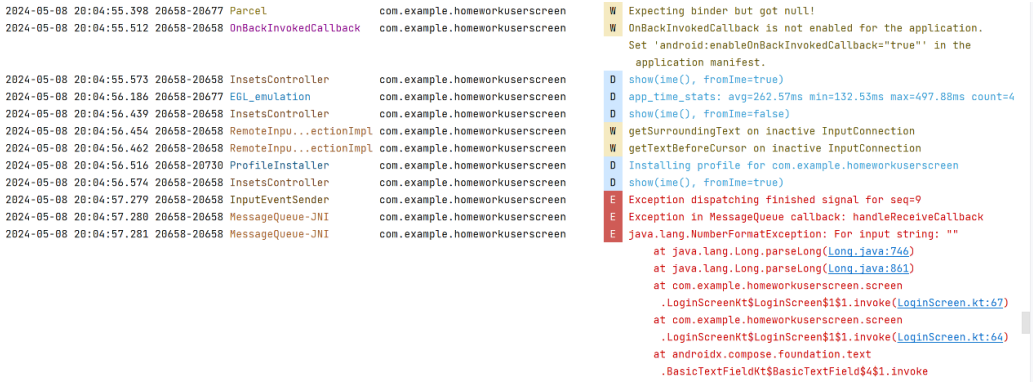
最终的解决方案：

先插入 user 表中的 userData，得到 userId 后，再根据该 userId 产生 starData 并插入 star 表。

```
val userData=DbHelper.createExampleUserData()
val userNumber=userDataDao.addUser(userData)
showLog( info: "$userData 已插入")
val starData=StarData( number: 0,userNumber, StarId: 0)
val starNumber=starDataDao.addStar(starData)
showLog( info: "$starData 已插入")
```

图片 35 依据外键约束进行数据插入

（4）字符转换问题



图片 36 NumberFormatException

问题描述：

当输入 TextField 为空时，输入内容不可直接由 String 类型转换为 Long 类型，控制台报错 java.lang.NumberFormatException: For input string。查阅资料后发现，当 Java 在将 String 字符串转换为数字的时候，如果遇到没有办法转换的情况，Java 将会抛出一个 NumberFormatException 异常。此处，应该是字符串为空无法转换产生的错误。

最终的解决方案：

因为 Java 的 Number API 不能处理字符串导致的，只需要将输入的字符串进行调整，保持为数字类型即可。所以此处，利用异常捕获判断是否可以转换为 Long 类型进行更新，如果不能，弹出消息提示用户输入数字格式不对，请重新输入。

```
private fun canConvertToLong(str: String): Boolean {
    return try {
        str.toLong()
        true
    } catch (e: NumberFormatException) {
        false
    }
}
```

图片 37 进行异常捕获

（5）导航嵌套问题

```
2024-05-16 09:57:26.176 10139-10139 AndroidRuntime          com.example.plantsinbookofsongs
E FATAL EXCEPTION: main
  Process: com.example.plantsinbookofsongs, PID: 10139
  java.lang.IllegalStateException: ViewModelStore should be set before setGraph call
```

图片 38 导航嵌套报错

问题描述：

当使用多个导航时，试图使用多个 NavHost 嵌套，结果导致在 setGraph 调用异常之前设置 ViewModelStore。

最终的解决方案：

使用一个单一的 NavHost，根据所在的目的地隐藏和显示底部导航。此时使用一个状态变量记录，根据该状态变量判断是否需要隐藏和显示底部导航。感觉这个方法不是特别好，需要在每个界面都进行修改状态变量的值，但是目前也没有更好的方法了。

```
class NavStatusViewModel:ViewModel(){
    // 使用MutableStateOf来持有一个可观察的状态变量

    private var _shouldShowBottomNavigation= mutableStateOf( value: true)
    val shouldShowBottomNavigation: State<Boolean>
        get()=_shouldShowBottomNavigation


    // 一个函数，用于在导航到Detail页面时隐藏底部导航栏
    fun hideBottomNavigation() {
        _shouldShowBottomNavigation.value = false
    }

    // 如果需要，你也可以有一个函数来重新显示底部导航栏
    fun showBottomNavigation() {
        _shouldShowBottomNavigation.value = true
    }
}
```

```
@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter")
@Composable
fun MainScreen(navController: NavHostController){
    val navViewModel:NavStatusViewModel= viewModel()
    var showNav= remember {
        navViewModel.shouldShowBottomNavigation
    }
    Scaffold {
        bottomBar = {
            if(showNav.value){
                BottomNav(navController = navController)
            }
                    },
        content = { it:PaddingValues
            NavHost(navController =navController, startDestination = "meet" ){ this: NavGraphBuilder
                composable(BottomDestination.Meet.route){ this: AnimatedContentScope  it: NavBackStackEntry
                    navViewModel.showBottomNavigation()
                    MeetScreen()
                }
                composable(BottomDestination.Found.route){ this: AnimatedContentScope  it: NavBackStackEntry
                    navViewModel.showBottomNavigation()
                    FoundScreen()
                }
                composable(BottomDestination.AI.route){ this: AnimatedContentScope  it: NavBackStackEntry
                    navViewModel.showBottomNavigation()
                    AIScreen()
                }
```

图片 39 状态变量控制底部导航显示

（6）Compose 性能问题

```
I  Skipped 110 frames!  The application may be doing too much work on its main thread.
D  app_time_stats: avg=40.43ms min=15.91ms max=169.88ms count=24
I  Skipped 47 frames!  The application may be doing too much work on its main thread.
```

图片 40 Compose 出现卡顿

问题描述：

当集成所有功能构成 App 时，在虚拟机上出现了掉帧卡顿的情况，并显示主线程任务过多。

最终的解决方案：

根据 Compose 官网上的性能改进的建议对代码进行了修改和微调，尽量减少了函数重组次数，同时，尽量把耗时的任务改到其他线程进行，掉帧的情况得到了一定改善。

不过，安装完 apk 进行实机测试时，发现基本未出现卡顿情况。

5 简要开发过程

4 月 19 号　查找资料确定要设计的应用和要实现的功能
4 月 20-22 号 进行 Found Screen 浏览诗经功能开发
4 月 23-25 号 进行 AI Screen 植物识别功能开发
4 月 26-27 号 进行 AI Screen 相册使用和拍照功能开发
4 月 27-30 号 进行 Meet Screen 每日一句功能开发
5 月 1-5 号　进行 User Screen 注册登录功能开发
5 月 6-10 号　进行 User Screen 收藏功能开发
5 月 11-16 号 进行各个页面的整合修改
5 月 17-20 号 使用 Material 主题构建器工具进行主题设计，导出 Compose 主题代码
5 月 23-25 号 功能开发完成，进入测试阶段，对程序进行集成测试
5 月 26 号　程序开发工作完毕，整理开发文档

6 学习感悟及对本课程的建议

经过一个学期的理论学习和最后作业的开发，我感觉进行开发是很有趣的。在整个进行开发的过程中，一点点实现自己想要的功能和页面，能带给自己很强的满足感。遇到问题时，进行很多实现上的设计和思考，这个过程虽然艰难，但是思考很多种方法、找到一种达到目的，本身也很有趣。这学期我选择了一门学习诗经中的草木的公共选修课，在开发这个 App 上给了我很多灵感，切实地体会到用技术实现理想的功能的快乐。

另外，就是感到在开发中适当使用 AI 是很方便的。其实，虽然经历了一个学期的学习，我的开发技术还是很不到位，在一些小细节上都会遇到困难，但是，AI 在实现一些很小的功能时表现得非常完美，这种地方适当地让 AI 进行编写，自己进行优化，这种策略对我的开发有很大帮助。

最后，感觉 App 还有很多的不足之处。很不满意的一个地方是，最开始进行开发时，我没有考虑到 App 的总体性能问题，到开发尾声的时候才意识到使用 Compose 的 Android App 在开发时实际上有一些减少重组次数和减小重组范围的方法，但是，这时候已经没有太多时间进行大范围修改程序了，只能做一些小的改动尽量提高性能。不过，在实机测试时整体未表现出卡顿。

当然，最后做出的 App 成品基本实现了想象的功能，设计的图标和配色也基本上符合自己的审美，总而言之，非常感谢这一学期老师干货满满的教导，也非常感谢这一学期依然坚持编程努力进步的自己，在五月末敲下这一段话，以此纪念几乎每天都在写代码的 24 年春夏。