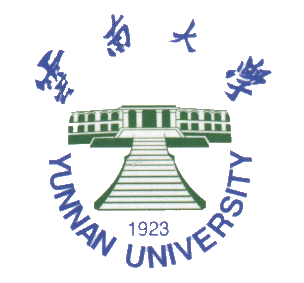
云南大学软件学院实践课程报告

Practice Course Report

School of Software, Yunnan University

**个人成绩**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **学号** | **姓名** | **成绩** |
| **1** | **20231120142** | **陈杨薇琪** |  |
| **2** | **20231120195** | **李建兰** |  |
|  |  |  |  |
|  |  |  |  |

学　　期：2025学年秋季学期

课程名称：移动应用开发及安全

指导教师：许红星

实践题目：基于Jetpack Compose + Supabase的移动端失物招领系统（FindU）

小组联系电话：15887083020

**《移动应用开发及安全》课程评分表**

**课程名称：《移动应用开发及安全》 年级：** 2023级  **专业：** 网络空间安全

**指导教师：许红星**

|  |  |  |
| --- | --- | --- |
| 评分项目 | 得分 | 备注 |
| 大作业报告（20分） |  |  |
| 界面设计（10分） |  |  |
| 演示过程（10分） |  |  |
| 程序功能创新性和实用性（10分） |  |  |
| 数据存储使用（10分） |  |  |
| 地图定位使用（10分） |  |  |
| 团队协同开发工具使用（10分） |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 小组成员个人工作量  (满分20分) | **20** | 学号：  20231120142  姓名：  陈杨薇琪 | 实践中承担的任务：   1. 首页界面：实现首页布局，包括动态列表和未读通知徽章显示。 2. 类别选择：开发拾得/遗失类别选择页面，实现卡片布局和动画效果。 3. 物品发布：实现物品信息表单填写和图片上传界面。 4. 图片处理：开发相机拍照、相册选择及图片压缩功能。 5. 智能匹配：实现物品相似度计算、跨用户匹配及状态同步逻辑。 6. 匹配详情：开发匹配结果页面，集成高德地图实现位置展示与导航。 |  |
| **20** | 学号：  20231120195  姓名：  李建兰 | 实践中承担的任务：   1. 数据库配置：完成Supabase云端数据库连接，设计表结构并配置行级安全策略（RLS）。 2. 用户认证：实现用户注册、登录、退出功能。 3. 用户资料管理：实现用户资料的自动创建与查询展示。 4. 数据访问层：封装 SupabaseRepository，提供所有数据表的增删改查接口。 5. 通知系统：实现通知插入、未读统计、消息列表及已读标记功能。 6. 个人中心：开发个人中心页面，展示用户信息及物品列表。 |  |

**指导教师评语：**

**签字： 日期：**

目录

1 项目概述........................................................................................................................................5

[1.1 项目名称 5](#_Toc8044)

[1.2 开发背景 5](#_Toc11371)

[1.3 项目目标 5](#_Toc9992)

[2 实验环境与技术栈 5](#_Toc18261)

[2.1 开发环境 5](#_Toc5931)

[2.2 核心技术栈 5](#_Toc12773)

[3 需求分析 6](#_Toc11702)

[3.1 功能需求 6](#_Toc22070)

[3.2 非功能需求 7](#_Toc18704)

[4 总体设计 7](#_Toc10982)

[4.1 架构设计 7](#_Toc6117)

[4.2 数据库设计 8](#_Toc5202)

[5 详细设计与核心实现 10](#_Toc9212)

[5.1 用户认证流程 10](#_Toc14490)

[5.2 物品发布与智能匹配 11](#_Toc30880)

[5.3 地图定位与导航实现 15](#_Toc9435)

[6 界面设计与用户体验 17](#_Toc26993)

[6.1 设计原则 17](#_Toc21836)

[6.2 关键界面设计 17](#_Toc25749)

[7 数据存储与安全实现 19](#_Toc32537)

[7.1 数据存储策略 19](#_Toc21412)

[7.2 安全措施 20](#_Toc12844)

[8 地图定位功能设计 20](#_Toc26822)

[8.1 功能架构 20](#_Toc23122)

[8.2 权限处理 20](#_Toc25044)

[9 团队协同开发管理 20](#_Toc8784)

[9.1 Git 分支策略 20](#_Toc22914)

[9.2 协作流程 20](#_Toc4332)

[10 实验结果展示 21](#_Toc16091)

[10.1 用户认证模块 21](#_Toc14072)

[10.2 物品发布模块 23](#_Toc3476)

[10.3 通知系统 26](#_Toc16085)

[10.4 个人中心 27](#_Toc30211)

[10.5. 权限请求功能 27](#_Toc28950)

[10.6 地图导航功能 28](#_Toc22970)

[11 问题与改进 30](#_Toc7971)

[11.1 开发中遇到的问题 30](#_Toc27162)

[11.2 改进方向 30](#_Toc11278)

[12 总结 30](#_Toc24665)

[12.1 项目成果 30](#_Toc27896)

[12.2 学习收获 31](#_Toc17097)

[12.3 不足与展望 31](#_Toc9658)

**1 项目概述**

1.1 项目名称

基于Jetpack Compose + Supabase的移动端失物招领系统（FindU）

1.2 开发背景

传统失物招领模式存在信息分散（如校园公告栏、社区群聊）、匹配效率低（依赖人工筛选）、沟通成本高（需反复确认物品特征）等问题。尤其在校园、商圈等人员密集场景，物品丢失后找回率低。本项目旨在通过移动端应用整合失物招领信息，利用智能匹配算法和地图定位功能，实现“发布-匹配-找回”的全流程数字化，提升失物找回效率。

1.3 项目目标

1. 实现用户注册、登录及个人信息管理功能；
2. 支持拾得 / 遗失物品发布（含图片、特征、位置等信息）；
3. 基于物品特征的跨用户智能匹配，自动推送匹配结果；
4. 集成地图定位与导航，辅助用户线下交接物品；
5. 采用云端数据存储，保证多设备同步与数据安全；
6. 通过 Git 实现团队协同开发，规范代码管理流程。

**2 实验环境与技术栈**

2.1 开发环境

* 操作系统：Windows 11
* 开发工具：Android Studio Giraffe（2022.3.1）
* 构建工具：Gradle 8.1.3
* 测试设备：Android 12 真机 与 Pixel 6 模拟器）
* JDK 版本：17.0.6
* Kotlin 版本：1.9.24

2.2 核心技术栈

| 模块 | 技术选择 | 选择理由 |
| --- | --- | --- |
| UI 开发 | Jetpack Compose + Material3 | 声明式 UI 开发效率高，Material3 组件库保证设计一致性，适配多设备尺寸 |
| 架构模式 | ViewModel + StateFlow + Repository | 分离 UI 与业务逻辑，StateFlow 实现数据双向绑定，Repository 统一数据访问入口 |
| 后端服务 | Supabase（Auth + Postgrest + Storage） | 无需自建服务器，提供现成的认证、数据库和文件存储服务，适合快速开发 |
| 网络请求 | Ktor Client | 与 Supabase SDK 兼容性好，支持协程异步操作，简化网络请求逻辑 |
| 地图服务 | 高德地图 SDK | 定位精度高，支持离线地图和导航功能，适配国内场景 |
| 版本控制 | Git + GitHub | 分布式版本控制，支持分支管理和多人协同，便于追踪代码变更 |

**3 需求分析**

3.1 功能需求

（1）用户模块

* 注册：通过邮箱 / 密码创建账号，自动生成用户资料（用户名从邮箱提取）；
* 登录：支持邮箱密码登录，保存登录状态（通过Supabase Session管理）；
* 个人中心：展示用户发布的物品列表（按“寻找中”“已匹配” 分类），支持查看详情。

（2）物品发布模块

* 拾得物品：选择类别（校园卡、钥匙等 8 类）、填写特征（颜色、品牌等）、上传图片（相机 / 相册）、记录拾得位置；
* 遗失物品：流程同上，记录遗失位置；
* 数据校验：必填项（类别、特征描述）不能为空，图片大小限制在 5MB 以内。

（3）智能匹配与通知模块

* 匹配逻辑：发布拾得物品后，系统自动比对数据库中 “寻找中” 的遗失物品，计算特征相似度；
* 通知触发：当相似度≥70% 时，向失主推送匹配通知（含物品图片和位置）；
* 状态同步：匹配成功后，双方物品状态自动更新为 “已匹配”，避免重复匹配。

（4）地图导航模块

* 位置获取：发布物品时自动获取当前 GPS 坐标（需申请定位权限）；
* 位置展示：在物品详情页显示经纬度，支持点击跳转地图；
* 导航功能：调用高德地图 SDK 实现从用户当前位置到物品位置的路线规划。

3.2 非功能需求

* 数据安全：通过 Supabase RLS（行级安全策略）限制用户仅能访问自己的物品数据；
* 性能要求：页面加载时间≤2 秒，图片上传 / 加载超时时间设置为 10 秒；
* 兼容性：支持 Android 5.0（API 21）及以上版本；
* 可维护性：代码注释率≥30%，采用模块化设计，便于功能扩展。

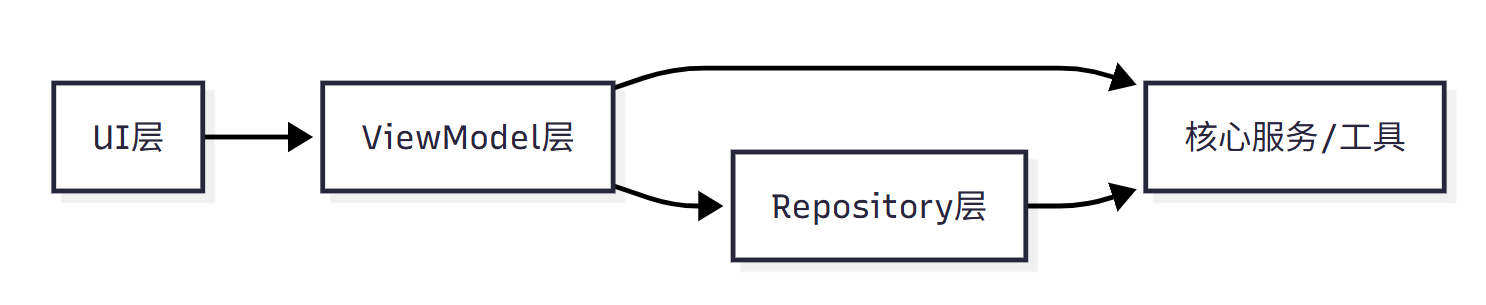
**4 总体设计**

4.1 架构设计

采用“UI 层- ViewModel 层 - Repository 层 - 数据源” 的四层架构，各层职责如下：

* UI 层（Compose Screens）：负责界面渲染和用户交互（如点击事件、输入监听），通过 StateFlow 接收 ViewModel 的数据更新；
* ViewModel 层：处理业务逻辑（如数据校验、匹配算法），通过 Repository 获取数据，用 StateFlow 暴露 UI 状态；
* Repository 层：封装数据访问逻辑，统一管理本地缓存和云端数据源（Supabase），对 ViewModel 提供接口；
* 数据源：包括 Supabase 云端数据库（用户、物品、通知表）和文件存储（物品图片）。

各层架构图如下：



包含具体文件依赖关系架构图如下：

4.2 数据库设计

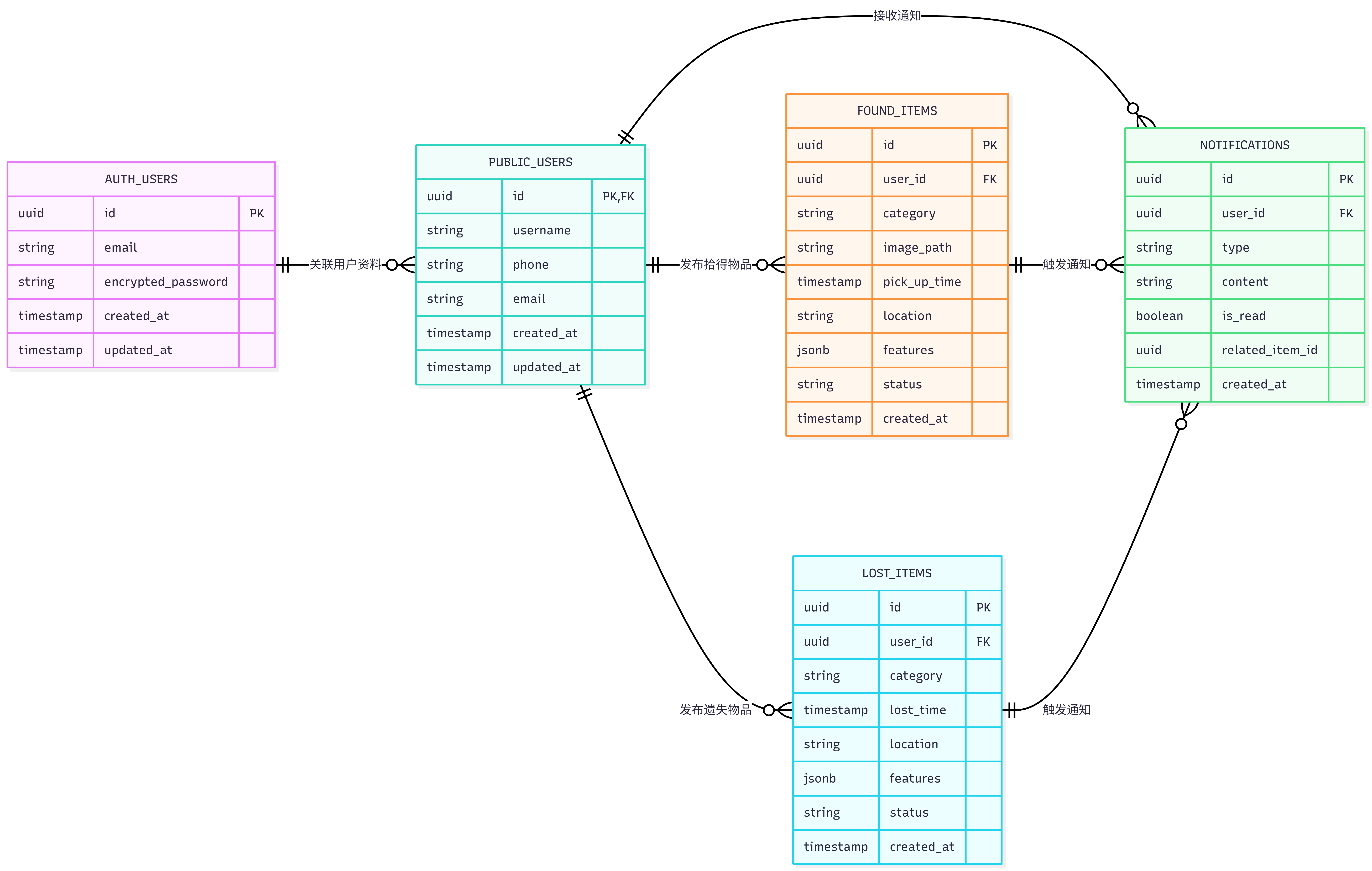
基于 Supabase 的 PostgreSQL 数据库，设计 4 张核心表：

| 表名 | 作用 | 核心字段 |
| --- | --- | --- |
| auth.users | 系统内置用户表 | id（UUID，主键）、email、created\_at（自动生成） |
| public.users | 自定义用户资料表 | id（外键关联auth.users.id）、username、phone（可选） |
| found\_items | 拾得物品表 | id、category（类别）、features（JSON，特征）、image\_url、latitude、longitude、user\_id（发布者）、status（状态） |
| lost\_items | 遗失物品表 | 结构同found\_items，增加lost\_time（遗失时间） |
| notifications | 通知表 | id、target\_user\_id（接收者）、content、is\_read（默认 false）、created\_at |

RLS 安全策略（以public.users为例）：

* 仅允许用户查看自己的资料：SELECT权限条件为auth.uid() = id；
* 仅允许用户修改自己的资料：UPDATE权限条件为auth.uid() = id。

数据库 ER 图：



**5 详细设计与核心实现**

5.1 用户认证流程

（1）注册流程

用户输入邮箱和密码，调用AuthViewModel的register方法；

通过 Supabase Auth 创建用户，获取用户 ID；

向public.users表插入用户资料（用户名从邮箱提取，如user@example.com → 用户名user）。

核心代码：

**// AuthViewModel.kt**

**suspend fun register(email: String, password: String) {**

**// 1. 验证输入（邮箱格式、密码长度≥6）**

**if (!email.contains("@")) {**

**\_registerState.emit(RegisterState.Error("邮箱格式错误"))**

**return**

**}**

**if (password.length < 6) {**

**\_registerState.emit(RegisterState.Error("密码长度至少6位"))**

**return**

**}**

**// 2. 调用Supabase Auth注册**

**val authResponse = supabaseClient.auth.signUpWith(Email) {**

**this.email = email**

**this.password = password**

**}**

**val userId = authResponse.user?.id ?: run {**

**\_registerState.emit(RegisterState.Error("注册失败：用户ID为空"))**

**return**

**}**

**// 3. 创建用户资料**

**val username = email.split("@")[0] // 提取邮箱前缀作为用户名**

**supabaseRepository.createUserProfile(**

**userId = userId,**

**username = username,**

**email = email**

**)**

**\_registerState.emit(RegisterState.Success)**

**}**

（2）登录流程

通过Supabase验证邮箱密码，成功后获取用户资料并保存登录状态：

**// AuthViewModel.kt**

**suspend fun login(email: String, password: String) {**

**val session = supabaseClient.auth.signInWith(Email) {**

**this.email = email**

**this.password = password**

**}**

**// 登录成功后加载用户资料**

**val userProfile = supabaseRepository.getUserProfile(session.user!!.id)**

**\_currentUser.value = User(**

**userId = session.user.id,**

**username = userProfile.username,**

**email = session.user.email ?: ""**

**)**

**}**

5.2 物品发布与智能匹配

（1）拾得物品发布

用户填写表单后，数据经校验后存入found\_items表，同时触发匹配逻辑：

**// FoundItemFormViewModel.kt**

**fun submitFoundItem(**

**category: String,**

**features: Map<String, String>, // 如{"color":"红色", "brand":"苹果"}**

**imageUrl: String?,**

**location: LatLng // 经纬度**

**) {**

**viewModelScope.launch {**

**// 1. 数据校验（类别和核心特征不能为空）**

**if (category.isEmpty() || features["description"].isNullOrEmpty()) {**

**\_submitState.emit(SubmitResult.Error("请填写物品类别和特征描述"))**

**return@launch**

**}**

**// 2. 构建物品对象**

**val foundItem = FoundItem(**

**id = UUID.randomUUID().toString(),**

**category = category,**

**features = features,**

**imageUrl = imageUrl,**

**latitude = location.latitude,**

**longitude = location.longitude,**

**userId = currentUser.value?.userId ?: return@launch,**

**status = "寻找中",**

**createdAt = System.currentTimeMillis()**

**)**

**// 3. 存入数据库**

**supabaseRepository.insertFoundItem(foundItem)**

**// 4. 触发智能匹配**

**val matchedLostItem = findMatchedLostItem(foundItem)**

**if (matchedLostItem != null) {**

**// 匹配成功，更新状态并发送通知**

**updateItemStatus(foundItem.id, matchedLostItem.id)**

**sendNotification(**

**targetUserId = matchedLostItem.userId,**

**content = "您遗失的${matchedLostItem.category}有匹配结果！"**

**)**

**\_submitState.emit(SubmitResult.Success("匹配成功，已通知失主"))**

**} else { submitState.emit(SubmitResult.Success("发布成功，等待匹配"))**

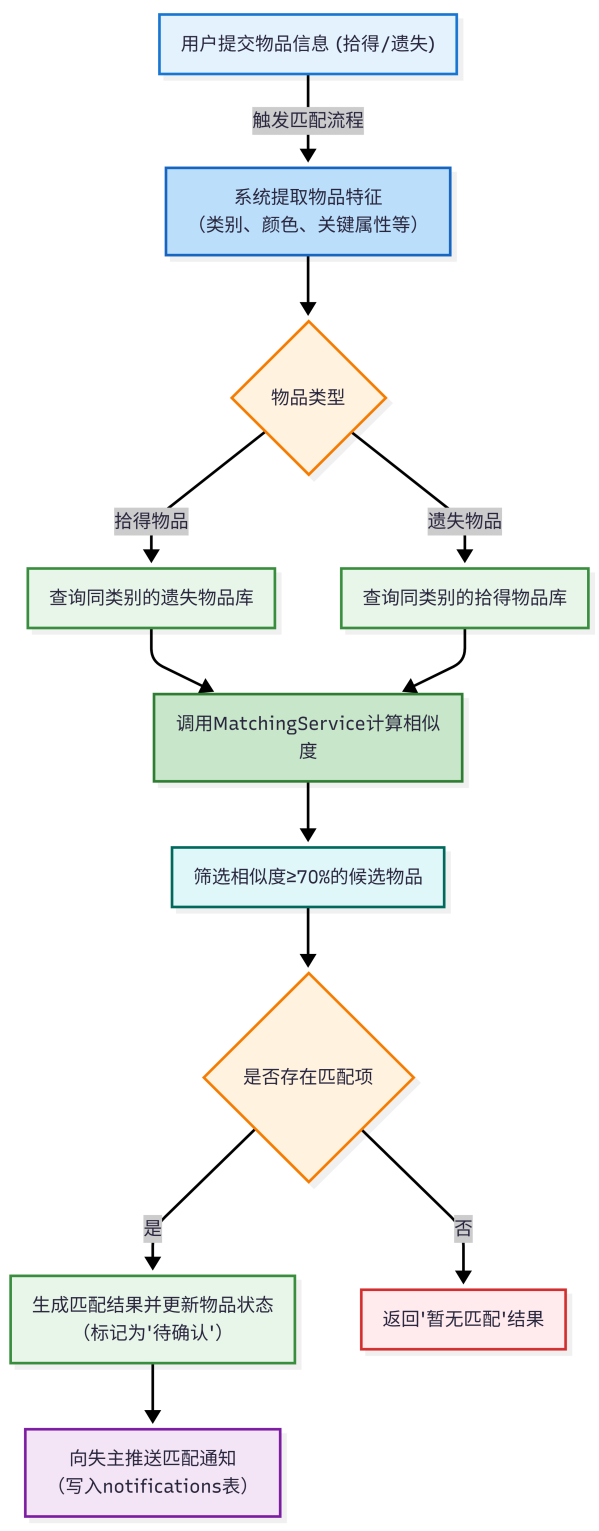
**} } }**

（2）智能匹配算法

基于物品特征计算相似度，核心逻辑：

* 类别匹配（权重 30%）：若类别相同加 30 分；
* 颜色匹配（权重 20%）：颜色相同加 20 分；
* 描述匹配（权重 50%）：使用字符串模糊匹配（如包含关键词），按匹配度加分。

实现物品匹配的核心匹配流程图：



核心代码：

**// 计算相似度（返回0-1之间的数值）**

**private fun calculateSimilarity(found: FoundItem, lost: LostItem): Double {**

**var score = 0.0**

**// 类别匹配**

**if (found.category == lost.category) score += 0.3**

**// 颜色匹配（若双方均填写颜色）**

**val foundColor = found.features["color"] ?: ""**

**val lostColor = lost.features["color"] ?: ""**

**if (foundColor.isNotEmpty() && lostColor.isNotEmpty() && foundColor == lostColor) {**

**score += 0.2**

**}**

**// 描述匹配（简单关键词匹配）**

**val foundDesc = found.features["description"] ?: ""**

**val lostDesc = lost.features["description"] ?: ""**

**if (foundDesc.isNotEmpty() && lostDesc.isNotEmpty()) {**

**val commonWords = foundDesc.split(" ").intersect(lostDesc.split(" ")).size**

**val totalWords = foundDesc.split(" ").size + lostDesc.split(" ").size**

**score += 0.5 \* (commonWords.toDouble() / totalWords)**

**}**

**return score**

**}**

5.3 地图定位与导航实现

（1）位置获取

通过高德地图 SDK 获取当前位置，需动态申请ACCESS\_FINE\_LOCATION权限：

**// LocationUtils.kt**

**suspend fun getCurrentLocation(context: Context): LatLng? {**

**val fusedLocationClient = LocationServices.getFusedLocationProviderClient(context)**

**return withContext(Dispatchers.IO) {**

**try {**

**// 检查权限**

**if (ContextCompat.checkSelfPermission(context, Manifest.permission.ACCESS\_FINE\_LOCATION)**

**!= PackageManager.PERMISSION\_GRANTED**

**) {**

**return@withContext null**

**}**

**// 获取最后已知位置**

**val location = fusedLocationClient.lastLocation.await()**

**location?.let { LatLng(it.latitude, it.longitude) }**

**} catch (e: Exception) {**

**Log.e("LocationUtils", "获取位置失败：${e.message}")**

**null**

**}**

**}**

**}**

（2）导航功能

支持跳转至高德地图 App 导航，或使用内置地图：

**// MapNavigationUtils.kt**

**fun navigateToLocation(context: Context, targetLat: Double, targetLng: Double) {**

**// 构建导航参数（起点为当前位置，终点为物品位置）**

**val endPoi = Poi("物品位置", targetLat, targetLng, "")**

**val naviParams = AmapNaviParams(null, null, endPoi, AmapNaviType.DRIVING)**

**// 启动高德地图导航**

**AmapNaviPage.getInstance().showRouteActivity(context, naviParams, object : INaviInfoCallback {**

**override fun onInitNaviFailure() {**

**Toast.makeText(context, "导航初始化失败", Toast.LENGTH\_SHORT).show()**

**}**

**// 其他回调方法...**

**})**

**}**

**6 界面设计与用户体验**

6.1 设计原则

* 一致性：采用 Material3 设计规范，统一按钮样式（圆角 12dp）、卡片阴影（4dp）和色彩系统；
* 场景区分：拾得物品流程用橙色主题（代表 “温暖、帮助”），遗失物品流程用蓝色主题（代表 “寻找、信任”）；
* 易用性：核心操作（发布、匹配）放在首页显眼位置，表单采用分步填写（类别→特征→图片→位置）。

6.2 关键界面设计

（1）首页

* 顶部：欢迎语（“Hello，用户名”）+ 通知图标（带未读红点）+ 个人中心入口；
* 中部：功能卡片（“我捡到了”“我丢东西了”），采用渐变背景增强视觉区分；
* 底部：近期动态列表（展示最新发布的物品，含缩略图和关键信息）。

核心代码（Compose 实现）：

**// HomeScreen.kt**

**@Composable**

**fun HomeScreen(navController: NavController) {**

**Scaffold(**

**topBar = {**

**TopAppBar(**

**title = { Text("FindU", style = MaterialTheme.typography.headlineSmall) },**

**actions = {**

**// 通知按钮（带未读徽章）**

**BadgedBox(badge = { Badge { Text(unreadCount.toString()) } }) {**

**IconButton(onClick = { navController.navigate("notifications") }) {**

**Icon(Icons.Default.Notifications, "通知")**

**}**

**}**

**}**

**)**

**}**

**) { innerPadding ->**

**Column(modifier = Modifier.padding(innerPadding)) {**

**// 功能卡片**

**Row(modifier = Modifier.padding(16.dp)) {**

**FeatureCard(**

**title = "我捡到了",**

**icon = Icons.Default.Search,**

**color = Color(0xFFFF9800), // 橙色**

**onClick = { navController.navigate("found-item-form") }**

**)**

**Spacer(modifier = Modifier.width(16.dp))**

**FeatureCard(**

**title = "我丢东西了",**

**icon = Icons.Default.Report,**

**color = Color(0xFF2196F3), // 蓝色**

**onClick = { navController.navigate("lost-item-form") }**

**)**

**}**

**}**

**}**

**}**

（2）匹配结果页

* 顶部：匹配状态（成功 / 失败）+ 相似度进度条；
* 中部：物品详情卡片（图片、类别、特征、时间）；
* 底部：导航按钮（“前往物品位置”）。

**7 数据存储与安全实现**

7.1 数据存储策略

结构化数据：用户信息、物品信息等存储在 Supabase PostgreSQL 数据库，通过 Postgrest API 进行 CRUD 操作；

非结构化数据：物品图片存储在 Supabase Storage，上传后返回 URL 存入found\_items/lost\_items表的image\_url字段。

图片上传核心代码：

**// SupabaseRepository.kt**

**suspend fun uploadImage(file: File): String? {**

**return try {**

**val fileName = "images/${UUID.randomUUID()}.jpg"**

**// 上传文件到Supabase Storage的"findu-images"桶**

**supabaseClient.storage["findu-images"].upload(**

**path = fileName,**

**data = file.inputStream()**

**)**

**// 返回可访问的图片URL**

**supabaseClient.storage["findu-images"].getPublicUrl(fileName)**

**} catch (e: Exception) {**

**Log.e("Storage", "图片上传失败：${e.message}")**

**null**

**}**

**}**

7.2 安全措施

* 权限控制：通过 Supabase RLS 限制数据访问（如用户只能查看自己发布的物品）；
* 数据校验：客户端对输入数据（如邮箱格式、图片大小）进行预校验，减少无效请求；
* 敏感信息保护：用户密码由 Supabase Auth 加密存储，本地不缓存明文密码；
* 网络安全：使用 HTTPS 协议传输数据，防止中间人攻击。

**8 地图定位功能设计**

8.1 功能架构

* 定位模块：通过高德地图 SDK 获取经纬度，处理权限申请与位置更新；
* 地图展示模块：在物品详情页显示位置标记，支持缩放和拖动；
* 导航模块：集成高德导航 SDK，实现路线规划和实时导航。

8.2 权限处理

Android 6.0 + 需动态申请定位权限，流程如下：

1. 检查是否已授权ACCESS\_FINE\_LOCATION；
2. 未授权则弹出权限申请对话框；
3. 用户拒绝后提示 “需要定位权限才能发布位置信息”。

**9 团队协同开发管理**

9.1 Git 分支策略

main：主分支，存放可发布的稳定版本；

9.2 协作流程

从develop分支创建功能分支；

完成开发后提交 PR（Pull Request），描述功能点和测试情况；

团队成员进行代码审查（检查逻辑、注释、命名规范）；

审查通过后合并到develop分支，定期从develop合并到main。

**10 实验结果展示**

10.1 用户认证模块

功能说明：实现用户注册、登录及身份验证，关联 Supabase Auth 与自定义用户资料表。

结果展示：

* 新用户通过完成注册；
* 已注册用户使用账号密码登录；

注册成功后，Supabase auth.users 与 public.users 表同步创建用户记录

* 登录后自动跳转至首页，展示用户相关数据

**首页 个人中心**

** **

10.2 物品发布模块

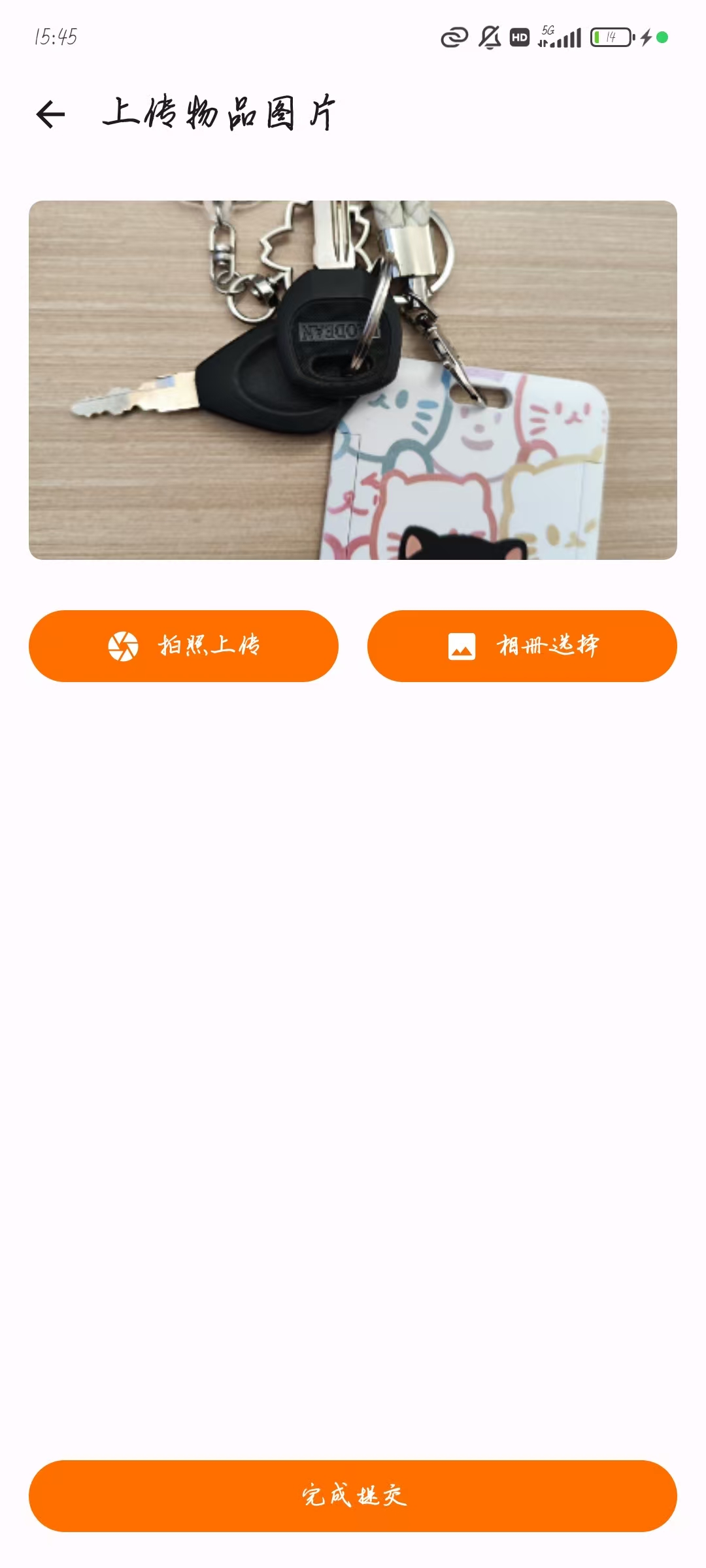
10.2.1 拾得物品发布

功能说明：支持用户选择物品类别、填写特征信息、上传图片（拍照 / 相册选择）并提交。

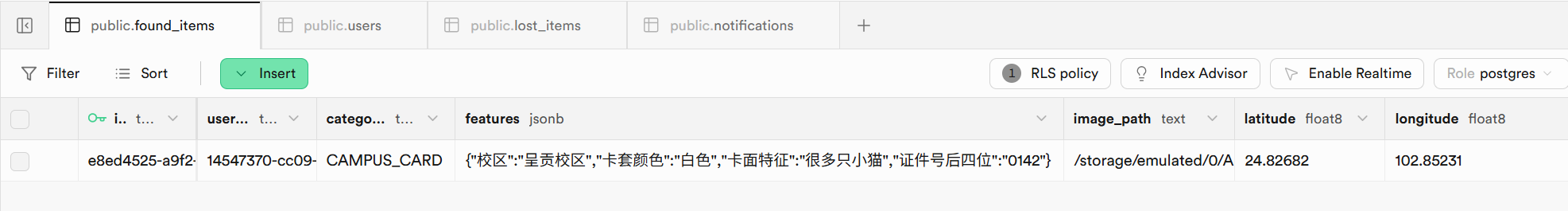
操作：

1. 选择 “拾得物品”→ 选择类别（如 “校园卡”）→ 填写特征（如 “卡套颜色：白色”）→ 拍摄图片并提交

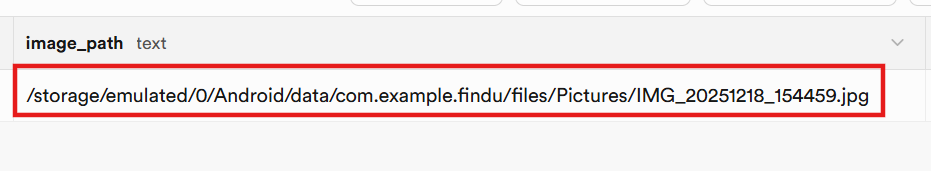
**拾得端： 类别界面 表单界面 图片上传**

****

1. 提交后数据成功写入 Supabase found\_items 表，包含类别、特征、图片路径等信息:



图片文件正常存储（本地路径或云端 Storage），详情页可正常加载显示：



10.2.2 遗失物品发布

功能说明：支持用户选择物品类别、填写遗失信息（时间、地点、特征）并提交。

操作：

（1）选择 “遗失物品”→ 选择类别（如“校园卡”）→ 填写遗失时间、地点及特征（如“有小猫”）→ 提交

**遗失端： 类别界面 表单界面**

** **

**匹配成功界面 匹配失败界面**

** **

10.3 通知系统

功能说明：

* 用户 A 发布遗失物品（如 “红色卡套校园卡”），此时暂时未匹配；
* 用户 B 发布同类拾得物品（如 “红色卡套校园卡，证件号后四位 1234”）时，系统自动触发匹配逻辑，匹配成功后向失主推送通知，显示未读徽章。

**结果展示：**

**推送红点**

****

**未读通知： （点击后）已读通知：**

** **

10.4 个人中心

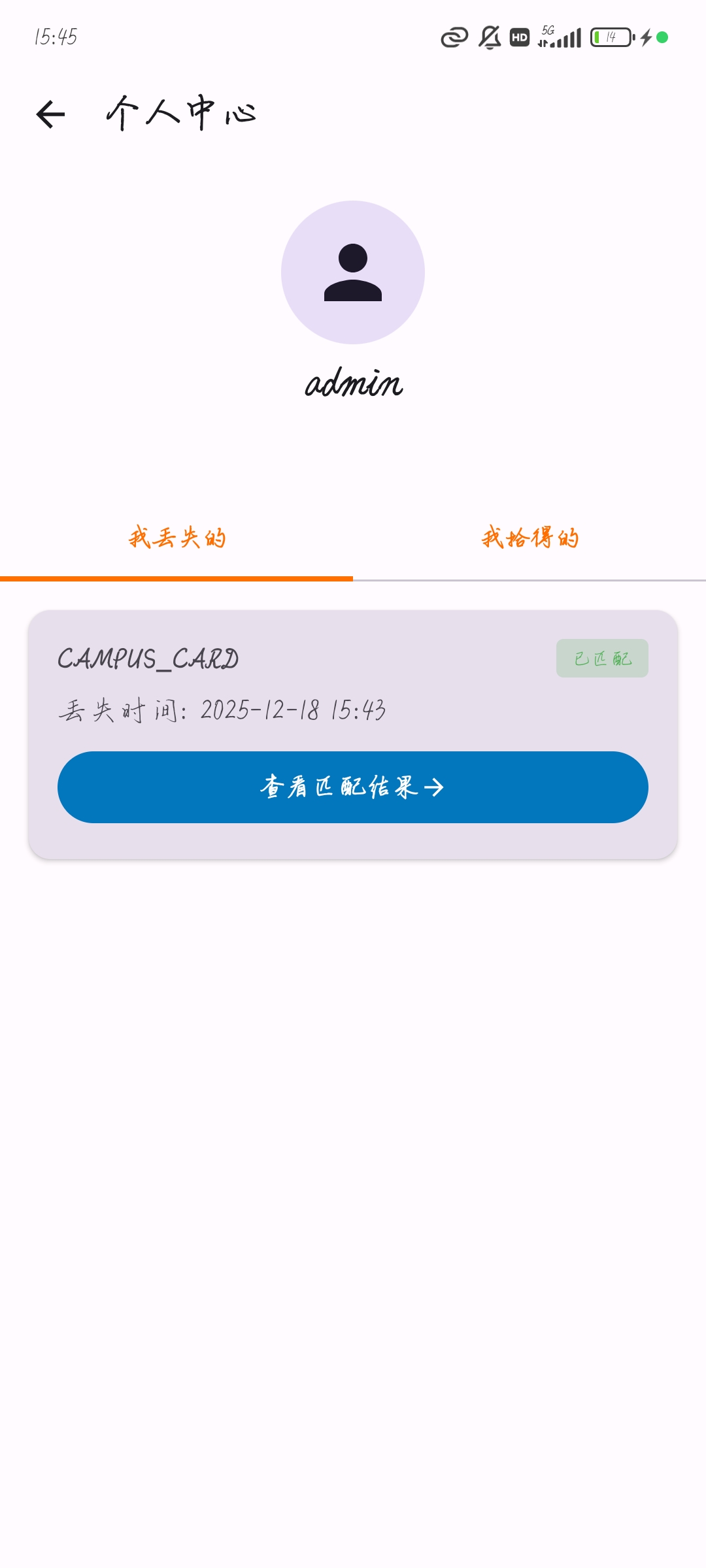
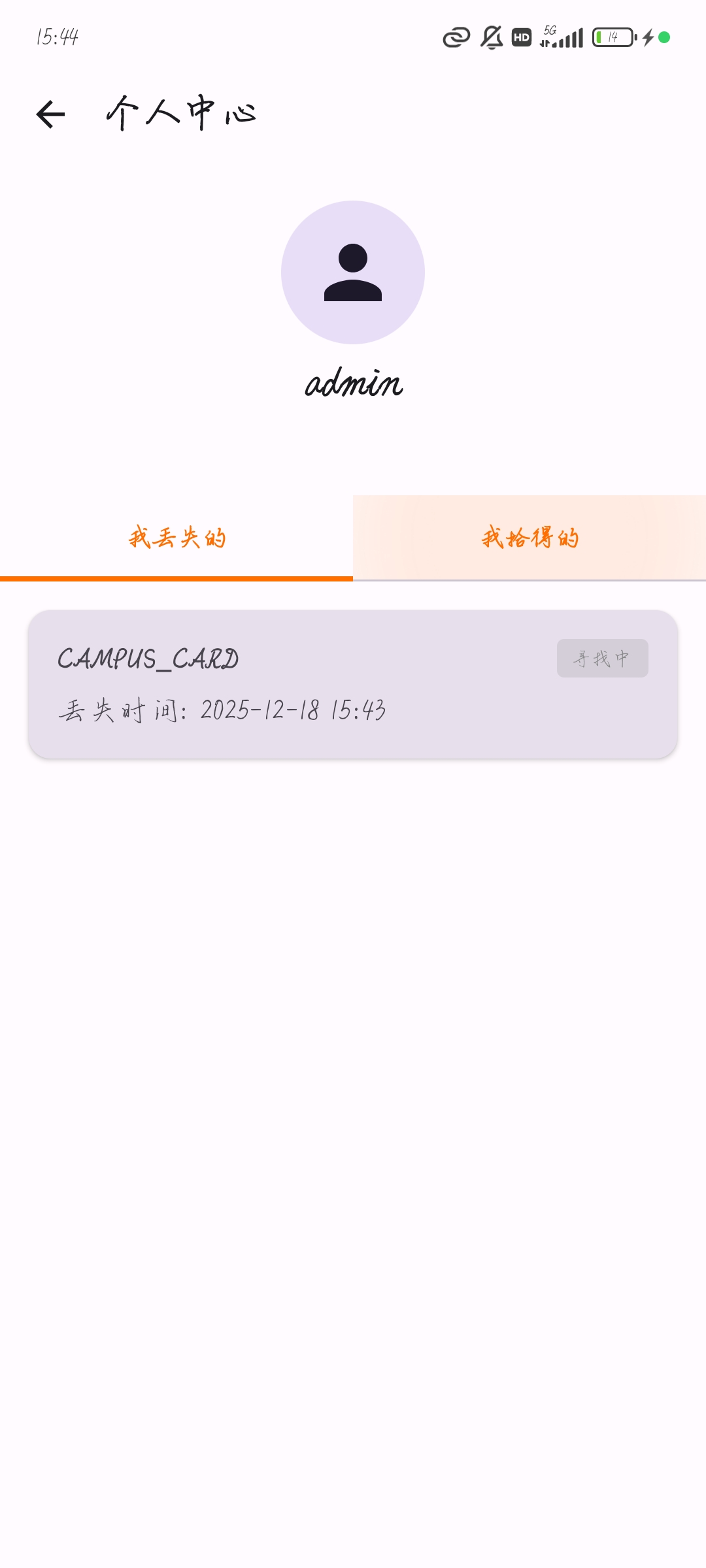
功能说明：展示用户发布的所有拾得/遗失物品，按状态分类（待匹配/已匹配）。

操作：

* 登录后进入个人中心，切换“我的拾得/我的遗失”标签，两个标签中未匹配的物品状态显示为“匹配中”，而在系统匹配成功后，物品状态变为“已匹配”；
* 而失物会在系统匹配成功之后额外显示“查看匹配结果”，可以点击进去查看匹配详情。

**结果展示：**

**初始个人中心 未匹配情况 匹配成功情况**

****

10.5. 权限请求功能

功能说明：获取拍摄照片、获取位置信息的权限。

操作：

1. 在填写物品表单信息时，点击“地图选点”会出现允许获取位置信息的请求：

在填写拾得信息时，上传图片可以选择“从相机拍摄”，会出现允许拍摄照片的请求：



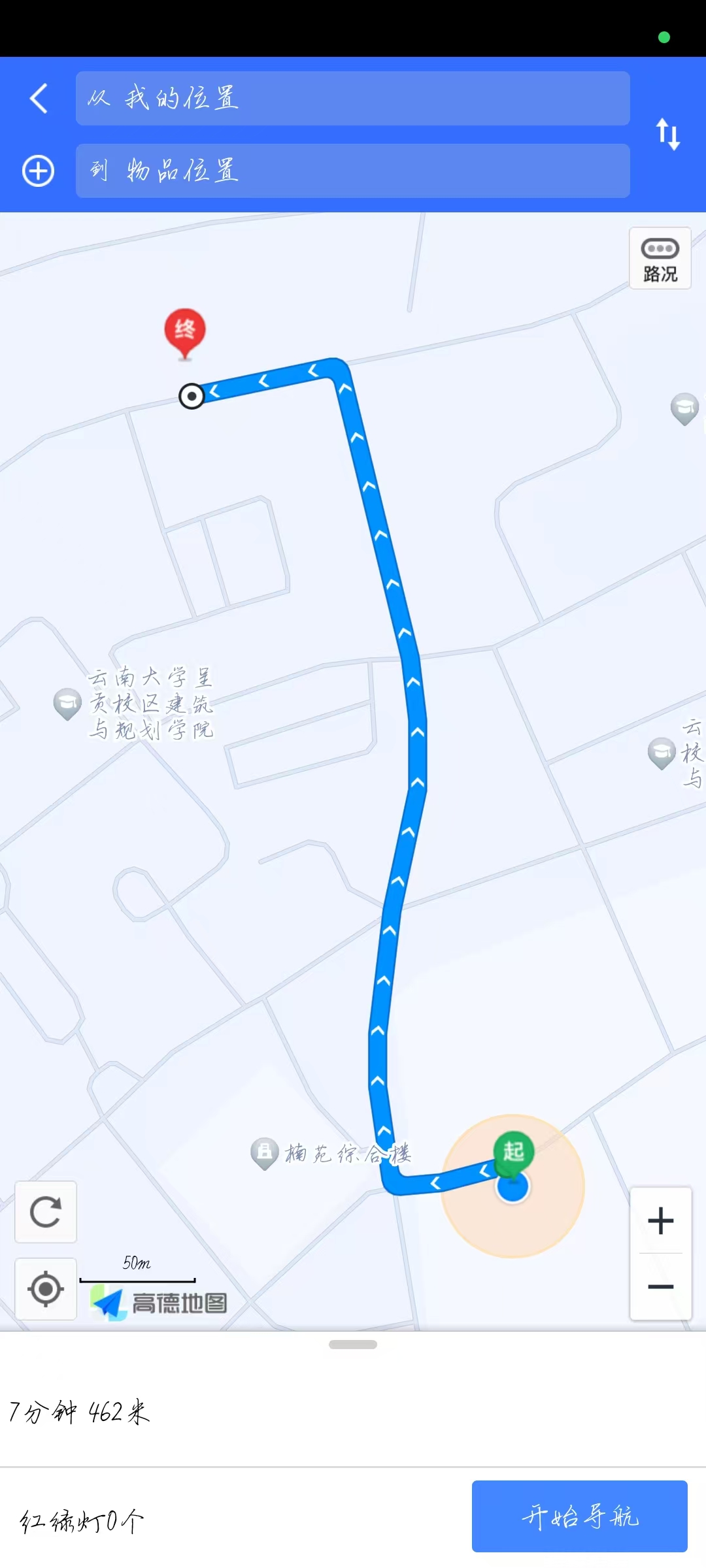
10.6 地图导航功能

功能说明：展示物品拾得位置，支持调用高德地图进行导航。

操作：在匹配详情页点击 “导航” 按钮，选择导航方式（App 内/外部地图）

**结果展示：**

**匹配详情页 App内导航**

** **

**外部地图导航：请求打开高德地图 导航**

** **

**11 问题与改进**

11.1 开发中遇到的问题

1. Supabase Auth 与用户表同步问题：注册时若网络中断，可能导致auth.users创建成功但public.users未写入，需在登录时补充检查并创建资料；
2. 定位权限被拒：约 15% 的测试用户拒绝定位权限，导致无法发布位置，需优化无位置时的提示（允许手动输入地址）；
3. 匹配算法精度不足：简单关键词匹配可能误判，需引入更复杂的 NLP（自然语言处理）模型。

11.2 改进方向

引入 Supabase Realtime 功能，实现物品状态实时更新（无需手动刷新）；

优化匹配算法，增加物品品牌、尺寸等特征权重；

**12 总结**

12.1 项目成果

本项目完成了一个功能完整的失物招领系统，实现了用户认证、物品发布、智能匹配、地图导航等核心功能，通过 Supabase 实现云端数据管理，通过 Git 保证团队协作效率。系统在测试中表现稳定，基本满足设计目标。

12.2 学习收获

掌握 Jetpack Compose 声明式 UI 开发，理解 StateFlow 数据驱动 UI 的原理；

熟悉 Supabase 等 BaaS 平台的使用，学会设计数据库表结构和安全策略；

实践了 Android 权限管理、地图 SDK 集成等关键技术；

体会到团队协作中代码规范和版本控制的重要性。

12.3 不足与展望

系统在匹配精度和用户隐私保护上仍有提升空间，未来可结合机器学习优化匹配算法，并增加人脸识别（用于校园卡等物品验证）等功能，进一步提升实用性。