项目开发计划书

项目名称：漫游手账

目录

[1. 引言 3](#_Toc24395)

[1.0 编写目的 3](#_Toc12099)

[1.1项目背景 3](#_Toc21407)

[1.2项目基本信息 4](#_Toc11439)

[1.3里程碑和提交作品 4](#_Toc1605)

[2.项目概述 5](#_Toc26023)

[2.1项目内容 5](#_Toc29635)

[2.2项目参加人员职责 6](#_Toc12951)

[2.3预期成果 6](#_Toc12498)

[2.3.1程序及文件 6](#_Toc25449)

[2.3.2服务 7](#_Toc27521)

[2.3.3非移交产品 8](#_Toc15503)

[2.4验收标准 9](#_Toc29326)

[2.5支持条件 10](#_Toc25989)

[2.6 WBS项目进度安排 11](#_Toc23669)

[2.7产品质量检测 12](#_Toc7195)

[2.8组间协调计划 12](#_Toc29694)

[2.9项目期限 13](#_Toc23361)

[3.开发计划 13](#_Toc32352)

[3.1开发过程 13](#_Toc22846)

[3.1.1需求分析 13](#_Toc20258)

[3.1.2系统设计 14](#_Toc8299)

[3.1.3代码实现 14](#_Toc101)

[3.1.4测试维护 15](#_Toc24293)

[3.1.5文档与产品部署 15](#_Toc6729)

[3.2生命周期 16](#_Toc11490)

[3.3项目框架 16](#_Toc2681)

[3.4项目管理 17](#_Toc18051)

[3.5 开发库内容 17](#_Toc5417)

[3.6可能出现的问题 23](#_Toc23896)

[4 项目协调与跟踪 23](#_Toc29461)

[4.1项目协调方式以及频率 24](#_Toc1784)

[4.2项目跟踪方式以及频率 24](#_Toc11311)

1. 引言

1.0 编写目的

本开发计划书旨在为《漫游手账》小程序的开发提供系统化指导，明确项目以"旅游打卡记录+社交化挑战"为核心定位，通过创新性的定位打卡和人脸识别技术实现趣味化旅行记录功能。计划书将详细规划项目实施方案，包括功能模块划分、技术实现路径（重点解决GPS定位精度，用户上传图片要求检验与人脸识别准确性问题）、开发周期安排以及资源调配方案，同时建立完善的质量管控体系（涵盖功能测试、性能优化和用户体验评估）和风险管理机制（针对技术实现难点和隐私合规要求制定应对预案）。通过规范化开发流程和明确责任分工，确保项目团队高效协作，最终交付一款兼具实用性和趣味性的优质产品。

本计划书将作为项目管理的核心依据，为开发团队提供明确的工作指引和进度把控标准。通过定期对照计划与实际开发进展，项目管理者能够及时发现问题并调整实施方案，保证开发工作始终围绕既定目标推进。同时，计划书也将作为团队成员间的协作契约，帮助各方统一认识、明确职责，共同推动《漫游手账》小程序从需求分析到交付的全流程工作，最终实现项目预期目标并通过验收评审。本计划面向项目组全体成员。

1.1项目背景

随着旅游消费的升级和个性化出行需求的增长，传统的旅游记录方式（如纸质手账、相册存储）已难以满足现代旅行者对数字化、社交化体验的追求。与此同时，移动互联网和位置服务技术的快速发展，为旅游场景下的趣味化记录和互动提供了更多可能性，推动了旅游类应用的创新升级。

"漫游手账"小程序是一款融合GPS定位、社交互动、创意打卡和企业团建的旅行记录工具，旨在为用户提供沉浸式的旅行记录体验。通过地图标记、图文记录和挑战任务等功能，帮助用户留存旅行足迹，同时借助社交分享和趣味打卡增强互动性。系统充分利用智能手机的便携性和微信生态的便利性，结合精准的GPS定位、图片识别功能和人脸识别验证，确保打卡记录的真实性和趣味性，解决传统旅游记录缺乏互动和激励的问题。

目前市场上，部分旅游类应用（如马蜂窝、小红书）虽提供游记分享功能，但缺乏深度的场景化互动和游戏化激励。本产品通过创新的挑战任务机制，支持用户自主发起基于地理围栏的打卡挑战，并进一步引入图片逻辑验证，人脸识别验证，增强打卡的趣味性和可信度。同时，系统提供个性化的旅行足迹地图和社交互动功能，让旅行记录更具情感价值和分享意义。

本项目的开发周期为8周，目标是为旅行爱好者、社交达人和企业团建等场景打造一款兼具实用性和趣味性的旅行记录工具，推动旅游记录方式向数字化、社交化和游戏化方向发展。

1.2项目基本信息

|  |  |
| --- | --- |
| **项目名称** | **漫游手账** |
| **项目类型** | ☑开发类 □升级类 □维护类 |
| **项目组长** | 葛子午 |
| **项目组员** | 张旭超，许开翔，曾武俊 |
| **项目开始时间** | 2025.3.20 |
| **项目结束时间** | 2025.6.1 |
| **项目预算** | 50元，在项目中主要用于地图API服务，AI情绪分析API和人脸识别服务进行付费 |
| **项目目标** | 本项目的核心目标是打造一款兼具趣味性和实用性的旅行记录小程序，为用户提供沉浸式的旅行打卡体验。系统将基于微信小程序平台，集成GPS定位、社交互动和创意挑战功能，实现个性化的旅行足迹记录。用户可通过地图标记、图文记录、温馨回忆与挑战任务留存旅行记忆，既能通过用户发送挑战进行定位和场景上的限制，也能帮助企业借助人脸识别验证确保打卡真实性。系统支持私密记录和公开分享两种模式，并提供社交互动功能（点赞、评论）增强用户粘性。管理后台将提供数据统计和内容管理功能，帮助运营者掌握用户活跃度和热门打卡点。最终目标是推动旅游记录方式向数字化、社交化和游戏化方向发展，为用户创造更多旅行乐趣。 |
| **成功标准** | 打造一款界面美观、交互流畅、功能完善的旅行记录小程序，既满足用户个性化记录需求，又提供创新的社交互动体验。严格按照项目计划完成开发，确保系统稳定性、数据安全性和用户体验满意度，最终通过验收评审并交付。 |
| **方式** | 以微信小程序形式呈现，进行迭代式开发。 |

1.3里程碑和提交作品

| 里程碑 | 提交产品 | 截止时间 | 负责人 |
| --- | --- | --- | --- |
| 项目计划确认 | 项目开发计划书 | 2024.3.30 | 葛子午 |
| 需求分析完成 | 软件需求说明书 | 2024.4.5 | 葛子午 |
| ​核心设计阶段 | 系统原型设计 | 2024.4.7 | 许开翔 |
|  | 系统架构与功能模块设计 | 2024.4.7 | 曾武俊 |
|  | 数据库设计 | 2024.4.10 | 葛子午 |
| ​开发实施阶段 | 前端界面设计 | 2024.4.12 | 张旭超 |
|  | 后端高级功能预研 | 2024.4.12 | 葛子午 |
|  | 接口设计文档 | 2024.4.14 | 许开翔 |
|  | 后端基础功能开发 | 2024.4.20 | 葛子午 |
| ​优化迭代阶段 | 测试与缺陷修复报告 | 2024.5.5 | 张旭超 |
|  | 高级功能模块代码 | 2024.5.21 | 葛子午 |
| ​功能验证阶段 | 功能演示视频 | 2024.5.25 | 葛子午 |
| ​答辩准备阶段 | 答辩PPT及演示素材包 | 2024.5.20 | 许开翔 |
|  | 系统功能演示视频 | 2024.5.25 | 葛子午 |
| ​项目结项 | 项目总结和源代码(终版) | 2024.5.30 | 曾武俊 |

2.项目概述

2.1项目内容

本项目旨在开发一款基于微信小程序的旅行记录应用，核心功能是利用GPS定位、图像识别技术和人脸识别技术，为用户提供趣味性和互动性兼具的旅行打卡体验。系统将包含微信小程序客户端和Web管理后台两部分，以满足普通用户和运营管理需求。微信小程序采用原生框架开发，管理后台采用Vue.js框架构建响应式界面。后端服务将采用Java Spring Boot3框架构建RESTful API服务，辅助使用Python Flask作为中间件，利用YOLO、Pytorch等处理人脸识别关键功能，数据存储方面选用MySQL数据库，并使用JPA进行高效的数据操作。为了保证系统的可扩展性和可维护性，我们将采用Docker进行容器化部署。

微信小程序端的核心功能包括：用户注册与登录，通过微信授权获取用户信息并建立账户体系；用户登录后，系统将使用JWT进行身份认证，确保会话安全；用户可以创建个人旅行博文，系统会根据GPS定位数据自动生成旅行足迹热力图，点击地图标记可查看历史博文打卡记录和AI生成的旅行回忆总结，给予用户足够的情绪价值；同时，用户可以发起基于地理围栏的图片挑战活动，要求参与者在指定位置拍摄包含特定场景或手势的照片，系统会通过图像识别技术验证照片是否符合要求；企业用户可额外使用人脸识别功能验证参与者身份；所有旅行记录可选择私密保存或公开分享，公开内容可在社区展示并获得点赞和评论；系统还将提供消息通知功能，及时推送挑战邀请和互动提醒。

Web管理后台主要面向运营管理人员，提供基础管理功能，包括：用户管理，可查看用户基本信息和活跃度；挑战活动管理，可监控活动参与情况；数据看板，可查看热门打卡点分布和用户增长趋势；企业服务模块，支持团队批量导入和管理，配置人脸识别功能。所有管理功能都以保证系统健康运行为目的，避免过度设计。

2.2项目参加人员职责

|  |  |  |
| --- | --- | --- |
| **角色** | **姓名** | **职责说明** |
| **小程序开发工程师** | 葛子午 | 负责微信小程序前端开发与界面设计，实现用户交互功能和地图可视化展示。协助基于python的人脸识别服务，搭建Flask中间件服务，处理小程序与核心后端系统的数据通信，确保跨平台交互的流畅性。 |
| **后端开发工程师** | 张旭超 | 负责SpringBoot+Vue后端服务开发，设计并实现RESTful API接口。进行数据库架构设计与优化，管理用户数据、地理位置信息等核心数据的存储与访问，保障数据安全性与查询效率。 |
| **后端开发** | 曾武俊 | 负责SpringBoot+Vue后端核心模块开发，实现用户认证、内容审核等业务逻辑。作为项目经理统筹项目进度，协调开发资源，组织代码评审，负责技术文档编写与交付成果质量把控。 |
| **算法开发工程师** | 许开翔 | 负责人脸识别算法开发，基于YOLO和深度学习框架实现人脸检测与特征提取功能。设计并训练专用识别模型，开发配套的活体检测模块，为系统提供安全可靠的身份验证服务。 |

**项目组成**

| **角色** | **责任承担人** |
| --- | --- |
| 项目组长 | 葛子午 |
| 技术总监 | 葛子午 |
| 后端开发工程师 | 曾武俊 |
| 全栈开发工程师 | 张旭超 |
| 算法开发工程师 | 许开翔 |
| 测试负责人 | 葛子午 |
| 数据库管理员(DBA) | 张旭超 |
| 配置管理员(SCM) | 曾武俊 |

2.3预期成果

2.3.1程序及文件

软件名称: 漫游手账

编程语言：

前端：vue3 + JavaScript + WXML+WXSS​+ HTML + CSS

后端：Java + Python，利用springboot和flask框架，并用YOLO、Pytorch进行人脸识别模块处理

存储方式：本地存储，后期部署服务器

文件：

1. 系统设计文档

相当于整个项目的“技术说明书”，用来告诉团队如何一步步把系统搭建起来。主要包含：

• 系统整体结构图（比如分几层、每部分做什么）

• 各个功能模块的作用和协作方式

• 数据库怎么设计、数据如何保护

• 性能要求（比如能承受多少用户同时使用）

• 服务器和网络环境怎么安排

• 配套的测试计划和图表说明

2. 代码说明文档

相当于“编程指南”，指导程序员怎么写代码。主要包含：

• 代码文件怎么分类存放

• 核心功能的实现思路（比如计算规则、操作步骤）

• 不同代码之间如何配合

• 操作数据库的注意事项

• 代码注释要求和更新记录

帮助团队成员写出风格一致、容易维护的代码。

3. 测试报告

相当于“质量检查表”，证明系统是否达标。主要包含：

• 测试了哪些功能

• 测试环境配置（用了什么设备或软件）

• 发现的问题和修复情况

• 速度、安全性等关键指标

• 最终结论和改进建议

让所有人清楚系统现在的真实表现。

4. 项目文档

使用手册：教用户如何操作系统，比如安装步骤、功能操作图解；开发记录；从需求分析到测试的完整过程文档，方便后续维护升级。

2.3.2服务

软件预期实现的主要功能：

用户层面：

（1）用户注册与微信授权登录

（2）个人旅行地图可视化展示

（3）创建个性化旅行记录（图文+定位）

（4）参与用户发起的旅行挑战

（5）发起基于地理围栏的图片挑战（设置特定拍照要求）

（6）查看博文点赞数与互动评论

（7）AI生成旅行回忆卡片（基于打卡内容的情感分析）

（8）私密/公开旅行记录管理

企业/团队层面：

（1）企业管理员批量导入团队成员形成名单

（2）配置专属团队挑战活动

（3）设置人脸识别验证要求

运营管理层面：

（1）用户行为数据分析

（2）热门打卡点运营管理

特殊功能：

（1）图像识别验证（手势/场景匹配）

（2）企业级人脸识别验证（需资质认证）

注：所有功能将分阶段实现，首期聚焦核心的旅行记录和挑战功能，企业级功能将在后续迭代中逐步完善。

2.3.3非移交产品

1. 技术可行性研究报告

项目启动阶段将组织技术团队进行全面的技术评估工作，对拟采用的技术方案进行可行性论证，包含主流技术对比分析、关键技术验证方案、潜在风险预测等内容，作为项目技术选型的重要依据文档。

2. 需求调研记录

通过用户访谈、问卷调查、竞品分析等方式收集的第一手需求素材，包括访谈录音、问卷原始数据、竞品功能拆解等，这些资料将保持原始形态作为需求分析的基础依据。

3. 原型设计稿

原型设计稿是UI/UX设计的初步成果，用于验证交互逻辑和界面布局。虽然它是设计阶段的重要产出，但最终移交的是高保真设计图和前端代码，原型稿仅作为内部评审使用。

4. 开发环境配置文档

为统一团队开发环境而编写的详细配置说明，包含开发工具清单、环境变量设置、依赖库安装等具体配置要求和使用规范。

5. 代码评审记录

在代码评审过程中形成的质量检查文档，记录每次代码提交的评审意见、缺陷标记、优化建议等质量管控信息。

6. 测试用例设计文档

测试团队编制的系统化测试指导文件，包含功能测试场景设计、测试数据准备方案、预期结果定义等完整的测试执行规范。

7. 项目周报/月报

按项目阶段定期编制的进展汇报材料，详细记录任务完成情况、存在的问题、风险预警及后续工作计划等项目管理信息。

8. 版本控制日志

通过版本控制系统自动生成的开发过程变更日志，完整记录代码库的演进历史和重要版本节点信息。

9 临时测试数据

为验证系统功能而专门准备的各类测试数据，包含正常业务流程数据和异常边界测试数据等。

10. 会议纪要

项目各类专题会议的完整记录，包含会议议题、讨论要点、决策结果和待办事项等关键信息。

2.4验收标准

2.4.1 **精度**

1. 定位精度：系统需在用户打卡时精准获取GPS位置，误差范围不超过20米（城市环境下）。

2. 图像识别精度：手势/场景匹配准确率≥90%，误识率≤10%。

3. 人脸识别精度（企业版）：识别准确率≥95%，误识率≤5%。

2.4.2 **响应时间**

1. 地图加载：热力图渲染时间≤3秒

2. 挑战验证：从提交照片到返回验证结果≤5秒

3. AI回忆生成：基于单篇游记的情感分析处理时间≤8秒

4. 后台管理：基础数据查询响应时间≤3秒

2.4.3 **兼容性**

1. 微信版本：支持微信8.0及以上版本

2. 设备兼容：

* 适配主流手机GPS模块
* 支持分辨率≥720P的摄像头
* 网络环境：支持4G/5G/WiFi网络

2.4.4 **安全性**

1. 数据安全：用户位置信息，人脸数据加密存储，传输过程使用SSL/TLS加密。

2. 防作弊机制：支持活体检测（如眨眼、摇头等动作验证），防止照片/视频伪造打卡。

3. 严格区分公开/私密内容，企业人脸数据独立加密

2.4.5 **用户体验**

1. 核心路径：

创建游记：操作步骤≤3步

发起挑战：设置流程≤5步

2. 界面规范：

字体大小适配系统设置

色彩对比度符合WCAG 2.0标准

3. 帮助系统：提供实时操作指引和常见问题解答

2.4.6 **可维护性**

1. 模块化设计：GPS定位、人脸识别、数据存储等模块解耦，便于独立升级。

2. 日志记录：完整记录用户操作、系统异常及GPS定位日志，保留至少20天。

3. API文档：提供完整的接口文档，说明调用规则、参数及返回示例。

2.4.7 **文档要求**

1. 用户手册：包含图文版操作流程、常见问题解答。

2. 技术文档：详细说明系统架构、数据库设计及相应API接口文档。

3. 测试报告：提供精度、响应时间、安全性等指标的第三方测试结果。

关键差异说明

1. 新增防作弊要求：针对人脸识别打卡场景，强调活体检测和防伪造能力。
2. 精度要求：针对旅游场景适当放宽，重点保证趣味性。
3. 响应时间：侧重用户体验关键路径。

2.5支持条件

2.5.1硬件要求

1. 开发环境硬件

处理器：Intel Core i5 或同等性能的 AMD 处理器（建议 4 核以上）。

内存：至少 8GB RAM（建议 16GB 以上以支持 Docker 和多任务处理）。

存储：SSD 硬盘，至少 256GB 可用空间（用于开发工具、数据库和容器镜像）。

网络：稳定的互联网连接（用于调用高德地图 API 和依赖库下载）。

2. 部署环境硬件

服务器：云服务器（如阿里云、腾讯云）或本地服务器，配置建议：

CPU：2 核以上

内存：4GB 以上

存储：50GB 以上（支持 MySQL 数据持久化和 Docker 容器运行）。

**2.5.2 软件要求**

1. 开发工具

前端开发：

微信开发者工具

浏览器开发者工具（Chrome DevTools）

VSCode（代码编辑器）

Vue：Vue CLI（脚手架） + Vuex（状态管理）

后端开发：

IntelliJ IDEA（集成开发环境）

Maven（项目管理工具）

MySQL数据库

Navicat（数据库辅助工具）

Docker（容器化工具）

版本控制：

Git 2.30+ 及 GitHub 仓库。

2. 依赖库与中间件

人脸识别：OpenCV 和 YOLO

定位服务：高德地图 API（JavaScript/Web 服务版）。

日志管理：Log4j2。

权限控制：Spring Security + RBAC

鉴权：JWT

3. 部署与运维

容器化：Docker 20.10+ 及 Docker Compose（用于环境隔离和快速部署）。

接口管理：Postman（用于API测试）。

4. 其他工具

协作工具：飞书（用于文档协作和会议记录）。

兼容性：系统支持 Windows 7+/macOS 10.15+/主流 Linux 发行版。

2.6 WBS项目进度安排

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 任务名称 | 任务描述 | 工作量（天） | 负责人 |
| 1 | 需求分析与功能确认 | 团队协同梳理用户需求，明确核心功能模块（如打卡、足迹地图、挑战活动等） | 3 | 葛子午、曾武俊、张旭超 |
| 2 | 系统架构设计 | 划分前后端模块（小程序端、SpringBoot服务、人脸识别模块），设计模块交互逻辑与数据流 | 3 | 葛子午、曾武俊 |
| 3 | 接口与数据结构设计 | 定义RESTful API接口规范（如/api/checkin打卡接口），设计数据库表结构与数据模型（含空间坐标存储） | 5 | 张旭超、曾武俊 |
| 4 | 小程序前端开发 | 实现微信小程序界面（Vant组件库），完成地图标点、图片上传、足迹展示等交互逻辑 | 5 | 葛子午 |
| 5 | 后端核心服务开发 | 搭建SpringBoot服务，实现用户认证、位置校验、数据存储等基础功能 | 5 | 曾武俊、张旭超 |
| 6 | 人脸识别模块开发 | 基于OpenCV实现人脸检测，训练轻量化识别模型（YOLO+MobileNet），封装为HTTP服务 | 7 | 许开翔 |
| 7 | 挑战活动模块开发 | 开发团队挑战功能，集成人脸识别签到、定位校验、参与名单管理 | 4 | 曾武俊、张旭超 |
| 8 | 数据可视化开发 | 实现足迹热力图、用户行为分析看板（ECharts集成） | 3 | 葛子午、许开翔 |
| 9 | 系统联调与测试 | 执行接口联调，完成功能测试（含定位偏移修正、人脸误识别率测试） | 5 | 全体成员 |

2.7产品质量检测

我们将严格遵循软件开发的标准化工作流程，定期对产品质量进行全面检测。

在代码质量与可维护性方面，应定期开展代码审查，确保严格遵循面向对象设计原则和模块化编程规范，以提升代码的可读性和可维护性；同时，实施完善的日志记录机制，全面记录用户操作日志、系统异常日志及性能日志，以便于故障排查和系统优化。

在界面与用户体验方面，应进行UI一致性检查，确保小程序内所有界面元素的风格统一，并组织真实用户参与用户体验测试，综合评估应用的易用性、操作流畅度及视觉舒适度。重点测试核心功能路径，确保用户能顺利完成旅行记录创建、挑战发布等关键操作。

在性能与稳定性方面，需进行全面的性能测试，包括页面加载时间、API响应速度、内存占用等指标，确保应用在常规使用场景下流畅运行。利用微信小程序自带的监控工具持续跟踪应用运行状态，及时发现并解决稳定性问题。

在数据安全方面，严格执行微信平台的数据安全规范，对敏感信息进行加密存储，确保用户隐私数据得到充分保护。同时建立内容审核机制，维护社区内容的安全健康。

此外，采用Git版本控制系统管理代码，确保开发过程的可追溯性，并制定版本回滚预案以应对紧急情况。所有功能更新都将经过严格的测试流程，包括单元测试、集成测试和用户验收测试，确保发布质量。

2.8组间协调计划

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **协调小组** | **协调方式** | **协调内容** | **问题处理方式** | **频率/时间** |
| ​全体成员 | 线下例会 | 同步本周进度、分配下周任务 | 当场讨论解决方案 | 每周一上午1小时 |
| ​全体成员 | 线下技术讨论会 | 关键技术选型与实现方案确认 | 组长决策争议点 | 按需召开（≤1小时） |
| ​全体成员 | 工位旁即时沟通 | 接口联调、数据交互问题处理 | 直接调试修改代码 | 开发过程中随时 |
| ​全体成员 | 微信群消息 | 日常进度同步、简单问题确认 | 快速文字/语音回复 | 工作日全天 |

2.9项目期限

本项目计划从2025年3月20日开始，到6月1日结束，共分为四个主要阶段。第一阶段（3.20-3.28）将进行需求分析工作，包括用户调研、竞品分析和需求确认，明确旅游打卡小程序的核心功能需求。第二阶段（3.31-4.12）是系统设计阶段，主要完成系统架构设计、前端界面设计、后端功能模块设计以及数据库设计。第三阶段（4.12-5.19）进入开发实施，将实现小程序前端界面、后端服务功能以及人脸识别模块的开发。最后阶段（5.19-6.1）将进行系统测试、问题修复、文档编写和答辩准备工作，确保项目按时完成并交付。整个项目将重点开发用户旅游打卡、足迹地图、团队挑战等核心功能模块，同时注重系统的稳定性和用户体验。

3.开发计划

3.1开发过程

3.1.1需求分析

1. 功能性需求

（1）支持用户通过微信授权快速注册与登录

（2）支持用户创建个性化旅行记录（图文+定位）

（3）基于地理围栏设定有效挑战范围，通过GPS定位验证用户位置

（4）图像识别验证用户提交的特定场景/手势照片

（5）企业管理员可配置团队挑战活动，设置人脸识别验证要求

（6）自动生成用户旅行足迹地图和统计数据

（7）支持内容审核机制处理违规内容

2. 非功能性需求

（1）性能需求：人脸识别≤1秒，地图加载时间≤3秒，图像识别验证响应≤5秒，AI回忆生成处理≤8秒

（2）安全需求：采用JWT技术进行接口鉴权，敏感数据加密存储

（3）兼容性需求：支持微信8.0+版本，支持Android 8.0+/iOS 12+，适配主流机型

（4）可靠性需求：关键操作异常自动恢复

3. 约束条件

(1）必须符合微信小程序平台规范

3.1.2系统设计

本项目采用分层架构设计，结合前后端分离模式与模块化开发理念，基于3.2章节的软硬件约束条件，形成以下技术实施方案：

1.架构设计

前端架构基于Vue 3和TypeScript构建响应式网页端，采用Vue CLI实现工程化开发，配合Vuex管理用户权限和考勤状态等全局数据。微信小程序端使用原生开发框架，与网页端共享业务逻辑抽象层。后端架构采用Spring Boot 3.x构建微服务体系，模块化拆分核心功能，包括人脸识别服务、定位服务和考勤业务服务。交互协议方面，前后端通过RESTful API交互，接口规范遵循OpenAPI 3.0标准，数据格式使用JSON并压缩传输体积。

2. 数据架构

存储方案采用MySQL 8.0存储用户档案、考勤记录和权限策略，使用InnoDB引擎和UTF8MB4字符集。空间数据通过MySQL GIS扩展存储地理围栏多边形，并建立R-Tree空间索引加速范围查询。日志存储采用ELK栈集中管理操作审计日志。完整性控制方面，通过主键约束和自增ID生成策略保证实体完整性，外键级联删除/更新规则保证参照完整性，触发器校验签到时间有效性等业务规则保证业务完整性。

3. 安全设计

传输层采用HTTPS和HSTS强制加密通信，敏感字段使用AES-256-GCM加密存储。鉴权体系包括JWT令牌无状态验证和动态配置的Token过期时间，权限控制通过Spring Security动态加载RBAC策略，支持接口级细粒度权限控制。审计追踪通过Log4j2记录关键操作并保留6个月日志。

4. 部署方案

采用容器化部署方案，基于Docker Compose编排服务依赖关系。云环境支持阿里云和腾讯云部署，通过Nginx实现负载均衡横向扩展后端实例，MySQL启用主从复制提升读性能。

5. 标准化设计

采用PlantUML进行UML建模，生成类图和时序图等设计文档。类图重点描述用户权限实体关系和考勤状态机模型，时序图详细展示人脸识别接口调用流程和围栏校验交互时序。同时建立统一的代码规范，包括命名规则、注释要求和目录结构标准，确保团队协作一致性。所有设计文档和代码规范统一存放在项目docs/standards/目录下，便于团队成员查阅和参考。

3.1.3代码实现

前端实现：

1. 框架搭建

2. 核心功能

• 路由控制：通过 Vue Router 实现页面跳转与权限拦截

• 地图组件：高德地图API渲染地理围栏，监听用户位置变更事件

3. 状态管理

Vuex 模块化管理考勤状态：

后端实现：

1. Spring Boot 配置

2. 人脸识别服务

3. 签到业务逻辑

4. 安全配置

数据交互：

1. OpenAPI 定义

2. 异常处理

关键设计验证点：

1. 性能测试：JMeter 模拟1000并发签到请求，响应时间≤800ms

2. 兼容性测试：微信小程序与Chrome/Firefox/Safari浏览器功能一致性验证

3. 灾备验证：模拟MySQL主节点宕机，从节点10秒内接管读请求

3.1.4测试维护

（1）测试工作安排

测试工作将分阶段有序推进，首先对每个功能模块进行单独测试，确保基础功能完整可用。在模块测试通过后，将进行联合测试，验证各模块间的交互和兼容性。最后开展完整测试，模拟真实使用场景进行全流程验证。同时建立自动化测试机制，通过工具自动检查代码质量，在每次代码更新后自动运行测试并生成报告。

（2）重点测试内容

功能测试将全面检查系统各项功能的正常运行情况，重点验证操作流程的正确性和数据显示的准确性。性能测试将模拟多用户并发场景，评估系统响应速度和长时间运行的稳定性。安全检查着重测试系统防护能力，包括数据保护措施和权限控制的有效性。

（3）系统维护措施

3.1.5文档与产品部署

（1) 编写代码说明文档，通过系统化记录小程序前端架构、后端服务接口、AI处理模块以及数据库设计，确保开发团队遵循统一的编码规范。该文档详细说明核心功能实现逻辑，包括地图定位集成、图像识别算法调用、人脸识别接口封装等关键技术的实现方案，并配套异常处理机制和性能优化建议，为后续功能扩展和维护提供技术依据。

（2) 编写用户手册和技术文档，详细介绍小程序的使用方法，包括微信授权登录、旅行记录创建、挑战发布与参与等核心功能操作流程。用户手册采用步骤化图文说明，重点展示地图标记、照片上传、挑战设置等特色功能；技术文档则深入解析系统架构设计、第三方服务集成方案和数据处理流程，为技术团队提供系统维护和优化指导。

（3) 使用Docker容器化部署后端服务，将Spring Boot应用、Python AI服务和MySQL数据库分别容器化，通过Docker Compose统一管理，在保证部署便捷快速的同时确保运行环境对系统的影响减至最小，并且使系统不同的模块之间的环境相互隔离减少系统的耦合度。

（4) 对系统进行运行测试，确保系统能够正常工作并提供稳定的服务。对测试中出现的问题进行分析以及进一步的改进，确保系统运行的完整性、安全性以及鲁棒性。

（5) 配置系统的安全性和可靠性，保证用户数据的安全和保密。

3.2生命周期

本项目采用迭代式开发模式，整个开发周期分为六个阶段：需求分析、系统设计、编码实现、系统测试、文档编写和项目答辩。

在需求分析阶段，团队将通过线下讨论确定旅游打卡小程序的核心功能需求，包括用户注册登录、打卡记录、足迹地图和团队挑战等主要功能模块。系统设计阶段将完成整体架构设计、数据库表结构定义和接口规范制定。

编码实现阶段采用前后端并行开发方式，优先实现基础打卡功能和地图展示模块。开发过程中通过每日站会和每周例会进行进度同步，遇到问题随时进行线下讨论解决。代码管理使用Git进行版本控制，建立dev、test、master三个主要分支。

系统测试阶段将对各功能模块进行完整测试，重点验证人脸识别准确性和地图定位精度。文档编写阶段将整理技术文档和用户手册。最后进行项目成果演示和答辩准备。

整个开发过程强调团队协作和及时沟通，通过线下会议和面对面讨论确保项目顺利推进。关键节点包括需求确认、设计评审、功能演示等，都将通过线下会议形式进行确认和决策。

**3.3项目框架**

本项目采用前后端分离架构，前端包含微信小程序和网页端，分别使用原生开发和Vue 3框架实现；后端基于Spring Boot构建，使用MySQL进行数据存储。系统核心功能包括用户权限管理、人脸识别打卡、定位校验以及考勤数据统计分析，同时提供系统管理模块进行规则配置和运行监控。在测试与维护方面，将通过功能测试、性能测试和安全测试确保系统质量。系统部署采用Docker容器化方案，开发环境支持本地调试，生产环境部署于云服务器。项目开发过程中将规范编写接口文档、用户手册等技术文档，确保系统可维护性和易用性。整体方案设计注重模块化和持续集成，以保障系统稳定可靠运行。

3.4项目管理

（1）制定项目开发计划，每周召开例会跟踪进度，确保按时完成各阶段任务。

（2）项目组由组长统筹协调，成员分别负责前端、后端、算法等不同模块的开发工

（3）保持沟通，使用任务管理工具跟踪工作进度。

（4）建立代码审查和测试制度，确保系统质量。

（5）使用版本控制工具管理代码，维护项目文档。

（6）项目完成后进行总结复盘，分析经验，给予团队成员相应的奖励和激励，鼓励他们在下一阶段中发挥更好的作用。

3.5 开发库内容

3.5.1 开发库目录结构

（1）根目录

漫游手账/

├── backend/ # 后端服务

│ ├── springboot/

│ └── flask/

├── frontend/ # 前端应用

│ ├── web/ # Vue3网页端

│ └── miniprogram/ # 微信小程序（WXML/WXSS + JavaScript）

├── docker/ # Docker相关配置

├── docs/ # 项目文档

├── scripts/ # 部署/测试脚本

└── README.md # 项目说明文档

（2）backend

springboot/

├── src/ # 主代码目录

│ ├── main/

│ │ ├── java/

│ │ │ └── com.javaclimb/ # 核心业务代码包

│ │ │ ├── config/ # Spring配置类等

│ │ │ ├── controller/ # REST API接口层

│ │ │ ├── service/ # 业务逻辑层

│ │ │ ├── repository/ # 数据持久层(DAO)

│ │ │ ├── entity/ # 数据库实体类

│ │ │ ├── exception/ # 自定义异常处理

│ │ │ ├── common/ # 通用工具类

│ │ │ └── Application.java # SpringBoot启动类

│ │ └── resources/

│ │ ├── mapper/ # MyBatis XML映射文件

│ │ ├── static/ # 静态资源文件

│ │ ├── application.yml # 应用配置文件

│ │ └── openapi/ # API文档规范

│ └── test/ # 单元测试代码

├── migrations/ # 数据库迁移脚本

├── logs/ # 应用日志目录

├── .mvn/ # Maven包装器配置

├── HELP.md # 项目帮助文档

├── mvnw # Maven包装器脚本(Linux)

├── mvnw.cmd # Maven包装器脚本(Windows)

└── pom.xml # Maven项目配置

flask/

├── app/ # 主应用包

│ ├── \_\_init\_\_.py # 应用

│ ├── routes/ # 路由控制器

│ │ ├── \_\_init\_\_.py

│ │ ├── face\_api.py # 人脸识别接口

│ │ └── health.py # 健康检查接口

│ ├── services/ # 业务逻辑层

│ │ ├── face\_detection.py # 人脸检测服务

│ │ └── model\_loader.py # 模型加载服务

│ ├── utils/ # 工具类

│ │ ├── decorators.py # 自定义装饰器

│ │ └── file\_util.py # 文件处理

│ └── config.py # 配置管理

│

├── models/ # 模型文件

│ ├── yolov5/ # 模型版本

│ │ ├── best.pt # 训练权重

│ │ └── classes.json # 类别标签

│ └── README.md # 模型说明

│

├── tests/ # 测试套件

│ ├── test\_routes/ # 接口测试

│ └── test\_services/ # 服务测试

│

├── requirements.txt # 依赖清单

├── Dockerfile # 容器配置

├── .flaskenv # 环境变量

（3）Frontend

web/

├── public/ # 静态资源 (直接拷贝到构建目录)

│ ├── favicon.ico

│ ├── index.html

│ └── assets/ # 公共图片/字体等

├── src/

│ ├── assets/ # 动态资源 (需打包处理)

│ │ ├── styles/ # 全局样式

│ │ └── images/ # 组件专用图片

│ ├── components/ # 通用组件 (按钮/卡片等)

│ │ └── ui/ # 基础UI组件 (可复用)

│ ├── pages/ # 页面入口

│ │ ├── Home/ # 首页

│ │ │ ├── components/ # 页面级组件

│ │ │ └── index.vue # 页面入口

│ │ └── User/

│ ├── routers/ # 路由配置 (Vue Router/React Router)

│ ├── stores/ # 状态管理 (Pinia/Redux)

│ ├── utils/ # 工具函数

│ │ ├── auth.js # 登录鉴权

│ │ └── request.js # Axios封装

│ ├── App.vue # 根组件

│ └── main.js # 应用入口

├── package.json

└── vite.config.js # 构建配置 (Vite)

miniprogram/

├── app.js # 小程序入口

├── app.json # 全局配置 (页面注册/窗口样式)

├── app.wxss # 全局样式

├── sitemap.json # 微信SEO配置

├── assets/ # 静态资源

│ └── icons/ # 小程序专用图标

├── components/ # 自定义组件

│ └── toast/ # 封装微信Toast组件

├── pages/ # 小程序页面

│ ├── home/

│ │ ├── index.js # 逻辑

│ │ ├── index.json # 页面配置

│ │ ├── index.wxml # 模板

│ │ └── index.wxss # 样式

│ └── user/

├── services/ # 服务层

│ ├── api.js # 封装wx.request

│ └── login.js # 微信登录逻辑

├── utils/

│ ├── util.js # 工具函数

│ └── wxPromise.js # Promise化微信API

└── project.config.json # 项目配置 (IDE依赖)

（4）Docker文档

├── docker/ # Docker相关配置

│ ├── Dockerfile # 后端Dockerfile

│ ├── docker-compose.yml # 容器编排（MySQL + Backend）

│ └── mysql/ # MySQL初始化脚本

1. 文档目录

├── docs/ # 项目文档

│ ├── api/ # API文档（Apifox导出）

│ ├── design/ # 系统设计文档

│ ├── database/ # 数据库设计（ER图/SQL）

│ ├── deployment/ # 部署指南

│ └── test/ # 测试报告

（6）测试脚本

└── scripts/ # 测试脚本

├── deploy.sh # 部署脚本

├── migrate.sh # 数据库迁移脚本

└── test.sh # 自动化测试脚本

**3.5.2 开发库访问权限**

|  |  |  |  |
| --- | --- | --- | --- |
| **成员** | **角色** | **目录** | **权限** |
| 曾武俊 | **后端协同开发** | $/wondering\_notes/backend/springboot/ | rwx |
|  | **小程序协同开发** | $/wondering\_notes/frontend/miniprogram/pages/checkin/ | rw- |
| 张旭超 | **后端主开发** | $/wondering\_notes/backend/springboot/src/main/resources/ | rwx |
|  | **Web前端开发** | $/wondering\_notes/frontend/web/src/views/admin/ | rwx |
| 葛子午 | **小程序主开发** | $/wondering\_notes/frontend/miniprogram/ | rwx |
|  | **Flask主开发** | $/wondering\_notes/backend/flask/routes/health.py | rw- |
|  | **项目组长** | $/wondering\_notes/docs | rwx |
| 许开翔 | **AI主开发** | $/wondering\_notes/backend/flask/models  $/wondering\_notes/backend/flask/app/service | rwx |
|  | **Web前端开发** | $/wondering\_notes/frontend/web/src/components | rwx |
| 全体成员 |  | $/wondering\_notes/scripts/test.sh | r-x |

**3.5.3 开发里程碑和存放地点**

| 里程碑 | 提交产品 | 时间 | 存放地点 |
| --- | --- | --- | --- |
| 需求分析 | **用例图** | 2024.3.31 | $/wondering\_notes/docs/design/use\_case/ |
|  | **系统流程图** | 2024.4.2 | $/wondering\_notes/docs/design/flow\_chart/ |
| 设计阶段 | **UI原型设计** | 2024.4.3 | $/wondering\_notes/docs/design/ui\_prototype/ |
|  | **类图** | 2024.4.6 | $/wondering\_notes/docs/design/class\_diagram/ |
|  | **数据库设计** | 2024.4.9 | $/wondering\_notes/docs/database/ |
|  | **接口文档** | 2024.4.9 | $/wondering\_notes/docs/api/ |
| 技术预研 | **基础工具类实现** | 2024.4.15 | $/wondering\_notes/backend/springboot/src/main/java/com.javaclimb/common/ |
|  | **人脸识别Demo** | 2024.4.15 | $/wondering\_notes/backend/flask/demo/ |
| 开发阶段 | **微信小程序功能模块** | 2024.5.12 | $/wondering\_notes/frontend/miniprogram/ |
|  | **后台管理系统页面** | 2024.5.12 | $/wondering\_notes/frontend/web/src/views/admin/ |
|  | **SpringBoot核心服务** | 2024.5.12 | $/wondering\_notes/backend/springboot/ |
|  | **Flask人脸识别接口** | 2024.5.12 | $/wondering\_notes/backend/flask/app/routes/ |
| 测试交付 | **测试报告** | 2024.5.16 | $/wondering\_notes/docs/test/ |
|  | **系统演示视频** | 2024.5.19 | $/wondering\_notes/docs/demo\_video/ |

3.6可能出现的问题

| **可能出现的问题** | **问题描述** | **解决方案** |
| --- | --- | --- |
| 技术储备不足 | 项目涉及多项新技术（如AI情感分析、图像识别等），团队成员可能存在学习曲线 | 每周组织技术分享会，建立内部知识库，并安排核心成员进行专项技术指导 |
| 架构设计经验欠缺 | 首次开发此类综合性小程序，在系统架构设计上可能存在考虑不周的情况 | 在进行设计时参考市场上已有项目的架构，并且询问相关资深从业者的意见，采用渐进式架构演进策略 |
| 开发规范不统一 | 多人协作开发可能导致代码风格和实现方式不一致 | 制定详细的编码规范文档，使用自动化代码检查工具，定期进行代码审查 |
| 时间不足 | 时间有限，可能无法实现产品预期功能 | 进行合理的项目规划和管理，优先保证核心功能完成 |

4 项目协调与跟踪

本项目作为软件开发综合实训的内容，旨在通过实际开发过程培养团队协作能力和工程实践能力。既需要我们交付一个功能完善的旅行记录打卡小程序，更重要的是希望我们在团队协作开发中掌握迭代软件的开发过程，提升运用现代软件开发方法和技术解决问题的能力。团队协作是本项目的核心要素，所以组内的协调对齐以及交流至关重要。我们将通过规范化的项目管理、完善的文档体系和高效的沟通机制，确保项目顺利推进。

4.1项目协调方式以及频率

1. 定期举行团队会议：本团队拟定每周举办1次工作会议，讨论项目进展。议将总结上周工作成果，制定下周开发计划，确保团队成员目标一致。
2. 沟通协作：日常使用飞书和微信进行即时沟通，快速解决开发过程中的技术问题。重要讨论和决策都会再文档中进行记录，确保信息透明可追溯。
3. 专项会议：针对技术难点召开专项会议，邀请相关成员深入讨论。会前准备技术方案文档，会后形成书面决策，避免问题遗留。
4. 建立GitHub远程仓库：通过GitHub进行代码协作，严格执行分支管理和Code Review流程。主分支设置保护机制，所有代码变更都需通过PR合并。
5. 建立完整的文档管理体系：包括会议纪要、技术文档和决策记录等。

4.2项目跟踪方式以及频率

**1.阶段目标跟踪：**

每三天检查一次项目进度与阶段里程碑的匹配情况。针对需求分析、UI设计、核心功能开发等关键阶段设立明确目标，通过进度对比及时发现偏差并调整开发计划。

**2.会议进度记录：**

在每次项目会议中记录项目的进展情况，包括已完成的任务、遇到的问题、下一步计划等。同时，在每次项目会议结束时记录项目进度，确保团队成员对项目进展有清晰的了解，并能够及时调整和安排工作。

**3. 代码交付监控**

每日检查GitHub提交记录，要求每个功能点或问题修复单独提交，commit信息需关联任务编号。项目经理每周五汇总代码提交情况，评估开发产出质量与进度匹配度。