

**《软件工程》项目报告**

**题目： 珞珈行迹-武汉市旅行交友攻略手帐微信小程序**

**课程名称： 软件工程**

**专业班级： CS2202**

**学 号： U202215404**

**姓 名： 郑云鲲**

**同组成员： 王雯琪 U202215680 （CS2203）**

**黄俊 U202215389**

**李宗泽 U202215384**

**指导教师： 范晔斌**

**报告日期： 2024.11.15**

**计算机科学与技术学院**

**任 务 书**

**一 总体要求**

1. 综合运用软件工程的思想，协同完成

一个软件项目的开发，掌软件工程相关的技术和方法；

2. 组成小组进行选题，通过调研完成项目的需求分析，并详细说明小组成员的分工、项目的时间管理等方面。

3. 根据需求分析进行总体设计、详细设计、编码与测试等。

**二 基本内容**

根据给出的题目任选一题，自行组队，设计与开发中软件过程必须包括：

**1. 问题概述、需求分析：**正确使用相关工具和方法说明所开发软件的问题定义和需求分析，比如NABCD模型，Microsoft Visio，StarUML等工具 (20%)；

**2. 原型系统设计、概要设计、详细设计**：主要说明所开发软件的架构、数据结构及主要算法设计，比如墨刀等工具（35%）；

**3. 编码与测试**：编码规范，运用码云等平台进行版本管理，设计测试计划和测试用例（30%）；

**4．功能创新**：与众不同、特别吸引用户的创新（10%）；

**5. 用户反馈**：包括用户的使用记录，照片，视频等（5%）。

**目 录**

**[1 问题定义 4](#_Toc1500119297)**

[1.1项目背景与意义 4](#_Toc1047008899)

[1.2 项目基本目标 5](#_Toc1590580453)

[1.3 可行性分析 6](#_Toc1430625484)

[1.4人员管理和项目进度管理 8](#_Toc1894354378)

**[2 需求分析 10](#_Toc1007664728)**

[2.1 E-R图、数据流图 10](#_Toc765043254)

[2.2 用例图等 13](#_Toc1097375389)

[2.3 原型系统设计 14](#_Toc998602487)

**[3 概要设计和详细设计 18](#_Toc583925774)**

[3.1 系统结构 18](#_Toc40216828)

[3.2 类图等 18](#_Toc1614363038)

[3.3关键数据结构定义 18](#_Toc1291183468)

[3.4 关键算法设计 23](#_Toc598293741)

[3.5 数据管理说明 25](#_Toc1004469733)

**[4 实现与测试 29](#_Toc1188765089)**

[4.1实现环境与代码管理 29](#_Toc1534482782)

[4.2 关键函数说明 29](#_Toc921000251)

[4.3 测试计划和测试用例 32](#_Toc189090981)

[4.4 结果分析 34](#_Toc1217466417)

**[5 总结 35](#_Toc733881903)**

[5.1 用户反馈 35](#_Toc1354559000)

[5.2 全文总结 35](#_Toc598971153)

**[6 体会 37](#_Toc1652314982)**

**1 问题定义**

**1.1项目背景与意义**

**1.1.1 N(Need) 需求**

现代都市人，尤其是生活在快节奏中的武汉市民或来访的游客，在进行旅行时，往往需要整合多个信息来源，如景区介绍、攻略、地图等。

许多用户还希望将自己的旅行经历通过手帐、图片等形式记录下来，同时与朋友或其他旅友分享交流。

而市场上大多数旅行类应用较为单一，无法满足从信息获取到个性化记录、再到社交分享的一站式需求。



图 1-1研发意义

**1.1.2 B(Benefit) 好处**

用户可以轻松获得全面的武汉市景区信息。在此基础上，用户可以自由编写个性化的旅行手帐，记录旅途中的点点滴滴，无论是精彩的瞬间还是感人的故事，都能通过文字、图片和贴纸的形式生动呈现。同时，平台提供了便捷的发布和查阅功能，让旅行攻略的分享变得简单高效，用户可以在这里交流旅行心得，互相启发旅行灵感。

这种多种功能的结合，不仅让用户能够享受到深度的旅行体验，更在旅途中增加了社交互动和乐趣。特别是平台提供的个性化足迹地图和成就系统，让用户能够清晰地看到自己的旅行轨迹，感受到每一步的成就。这些功能不仅增强了用户的归属感，也让旅行变得更有仪式感和纪念意义。在这样的互动和记录中，用户不仅收获了美好的回忆，还激发了探索更多未知领域的热情，真正实现了旅行与生活的完美融合。

**1.1.3 C(Competitors) 竞争**

目前市场上虽然有许多旅游类的APP和小程序，但大多功能较为单一，主要集中在景点介绍或攻略分享。部分应用可能会提供手帐功能，但设计较为简陋，缺乏灵活性。

此外，市场上的大多数旅游推荐产品，覆盖全国各个地方，但是体量也会较大，还有很多广告干扰和隐私安全问题。

因此，我们的小程序结合了景区展示、个性化手帐、旅行攻略、足迹地图，提供了一站式的全方位服务，同时避免了广告干扰和隐私安全问题，能更好地满足用户需求。

**1.2 项目基本目标**

课程时间内的基本项目目标为以微信小程序的形式实现我们的“珞珈行迹”项目，为武汉市内的游客和居民提供一站式的旅游信息、个性化手帐记录及交友服务。项目代码和资源将通过 Gitee 平台进行管理，并租用服务器部署后端服务，实现基于微信平台的用户数据存储和景点信息展示。

**1.2.1 A(Approach) 做法**

景区展示：涵盖武汉市所有景区，并提供详细的地图和景点介绍，用户可以按季节、热度及区域查询景点。

旅行手帐：提供丰富的贴纸、组件，可自由编写、设计个性化的旅行手帐，生成预览图并保存至相册。

攻略发布与发现：支持用户发布多图、多标签的旅行攻略，并通过发现页的瀑布流展示，用户可以为攻略点赞，获取实时消息通知。

足迹地图：基于用户的旅行攻略生成个性化足迹地图，点击足迹地图中的地点会展示相关的攻略卡片。

个性化服务：包括“我的足迹”、“我的成就”等模块，帮助用户管理个人内容，提供定制化的旅行服务体验。

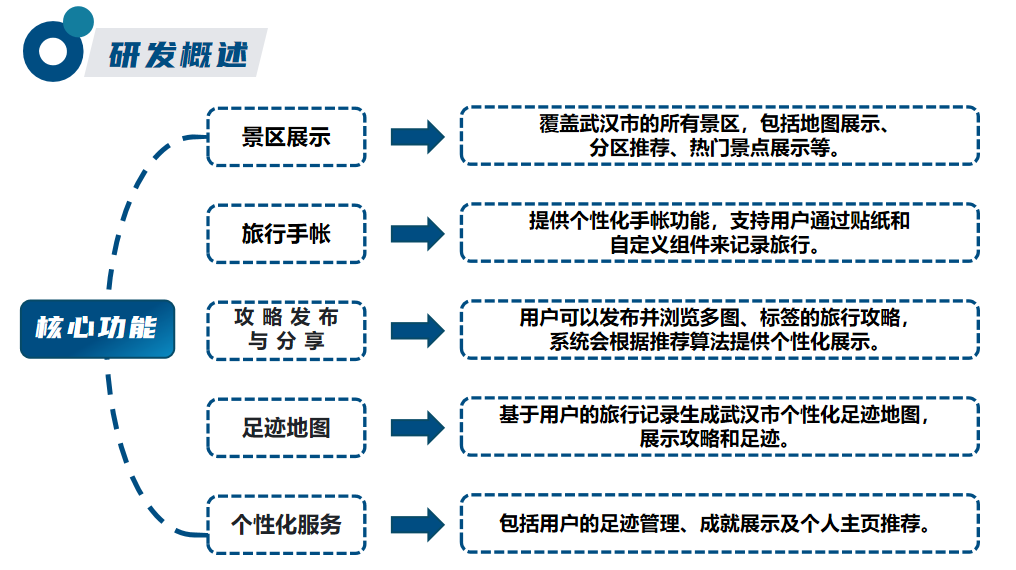


图 1-2研发概述

**1.2.2 D(Delivery) 推广**

社交媒体推广：在微信、微博等平台上，结合用户的社交圈进行嵌入式推广，邀请用户分享手帐、足迹地图，吸引更多潜在用户。

与旅行社合作：联合武汉市的旅行社、景区，开展小程序内的合作宣传，为景点导流，同时为用户提供更多实用信息。

活动推广：不定期举办线上或线下活动，鼓励用户参与发布攻略、分享手帐的比赛，通过打卡送礼等形式增加用户粘性。

KOL推广：邀请旅行达人、旅游博主进行小程序体验，发布使用心得，提升产品的知名度。

**1.3 可行性分析**

**1.3.1市场需求分析：**

需求存在性：目前市场上缺乏将旅行信息、个性化手帐记录和交友服务结合于一体的应用，特别是针对武汉市。珞珈行迹小程序能够填补这一市场空缺。

目标用户群体：武汉市游客和本地居民，特别是喜欢旅行和记录生活的人群。

**1.3.2技术可行性：**

平台选择：

微信小程序优势：微信小程序拥有庞大的用户基础，易于分享和传播，且开发成本相对较低。微信平台提供了丰富的API接口，支持地图、支付、社交等功能，非常适合珞珈行迹小程序的开发。

技术栈：

前端技术：使用微信小程序官方推荐的技术栈，如WXML、WXSS、JavaScript，这些技术成熟且社区活跃，能够满足项目的前端开发需求。

后端技术：可以选择Node.js、Python、Java等后端语言，结合云开发或自建服务器。这些技术都具有良好的扩展性和稳定性。

功能实现：

景区展示：利用微信小程序的地图组件和API，可以实现景区的地图展示、分区推荐、热门景点展示等功能。

旅行手帐：通过Canvas API实现个性化手帐的绘制和编辑功能，用户可以自由创作和保存。

旅行攻略：利用富文本编辑器和数据库存储，用户可以分享和浏览旅行攻略。

足迹地图：结合地理位置API，记录用户的旅行足迹，并在地图上展示。

数据管理：

数据库选择：可以选择MySQL、MongoDB等数据库，根据项目需求进行数据存储和管理。

数据安全：采用加密技术保护用户数据，确保数据传输和存储的安全性。

性能优化：

前端性能：尝试通过代码优化、资源压缩等技术提高前端性能。

后端性能：采用缓存策略、数据库优化等技术提高后端处理能力。

测试与部署：

测试：编写单元测试、集成测试和性能测试，确保代码质量和功能完整性。

部署：利用微信小程序的云开发平台或自建服务器进行部署，确保稳定性和可扩展性。

**1.3.3经济可行性：**

开发成本：考虑到项目规模和所需技术，开发成本相对可控。

盈利模式：通过广告、会员服务、合作伙伴赞助等方式实现盈利。

**1.3.4 竞争分析：**

现有竞争：市场上的旅游类应用功能单一，珞珈行迹通过一站式服务脱颖而出。

差异化优势：结合旅行、手帐和社交功能，提供更丰富的用户体验

**1.4人员管理和项目进度管理**

**1.4.1 分工**

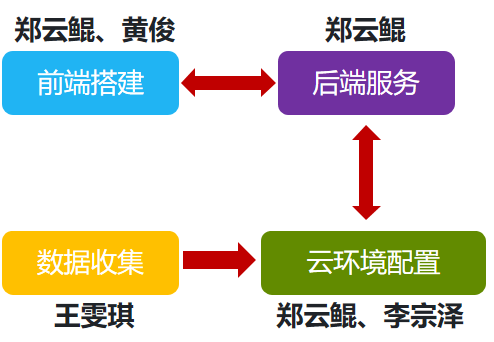


图1-3项目分工总览

**1.4.2 任务分解**

**1.前端搭建：郑云鲲、黄俊**

* 创建微信小程序项目
* 设计页面结构和布局
* 实现基本页面实现用户登录和注册功能
* 实现主要功能模块
* 与后端API对接

**2.后端搭建：郑云鲲**

* 创建微信小程序云开发项目
* 设计数据库结构
* 实现用户认证和授权
* 实现主要功能模块的云函数（如景区展示等）
* 部署后端服务

**3.数据收集：王雯琪**

* 确定数据来源（如公开API、合作伙伴等）
* 编写数据爬虫或脚本收集数据
* 清洗和处理数据
* 存储数据到数据库
* 定期更新数据
* 确保数据的准确性和完整性

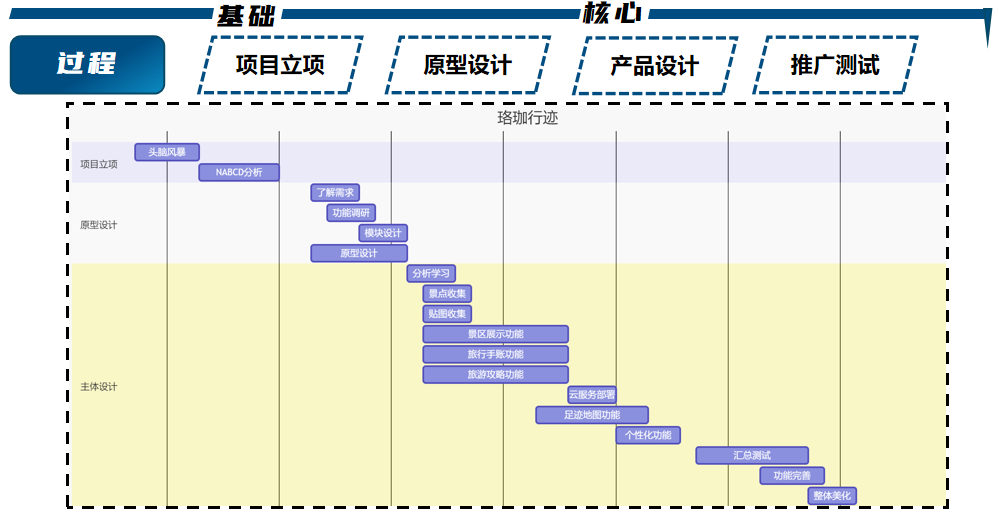
**4.云环境配置：郑云鲲、李宗泽**

* 配置微信小程序云开发环境
* 配置数据库
* 配置存储服务
* 配置云函数

**1.4.3 进度安排**

甘特图链接：

[https://github.com/zhengyunkun/2024-Software-Engineering/wiki/%E8%BF%9B%E5%BA%A6%E8%AE%A1%E5%88%92](https://gantt.mindsup.cn/share/YtXDaUWQ7)



**2 需求分析**

**2.1 E-R图、数据流图**

**2.1.1 E-R图**

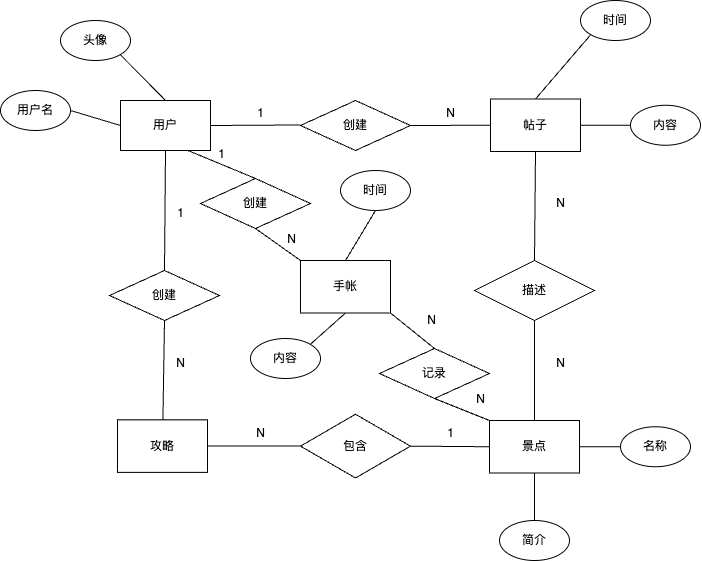
****

图2-1 E-R图

表2-1珞珈行迹需求分析表

|  |  |
| --- | --- |
| 功能种类 | 功能说明 |
| 景区展示 | 涵盖武汉市所有景区，并提供详细的地图和景点介绍，用户可以按季节、热度及区域查询景点。 |
| 旅行手帐 | 提供丰富的贴纸、组件，可自由编写、设计个性化的旅行手帐，生成预览图并保存至相册。 |
| 攻略发布分享 | 支持用户发布多图、多标签的旅行攻略，并通过发现页的瀑布流展示，用户可以为攻略点赞，获取实时消息通知。 |
| 足迹地图 | 基于用户的旅行攻略生成个性化足迹地图，点击足迹地图中的地点会展示相关的攻略卡片。 |
| 驴友招募 | 提供招募驴友的功能，用户可以发布、筛选并查找旅行伙伴，建立线下或线上交流圈。 |
| 个性化服务 | 包括“我的足迹”、“我的成就”等模块，帮助用户管理个人内容，提供定制化的旅行服务体验。 |
|  |  |

**2.1.2数据流图**

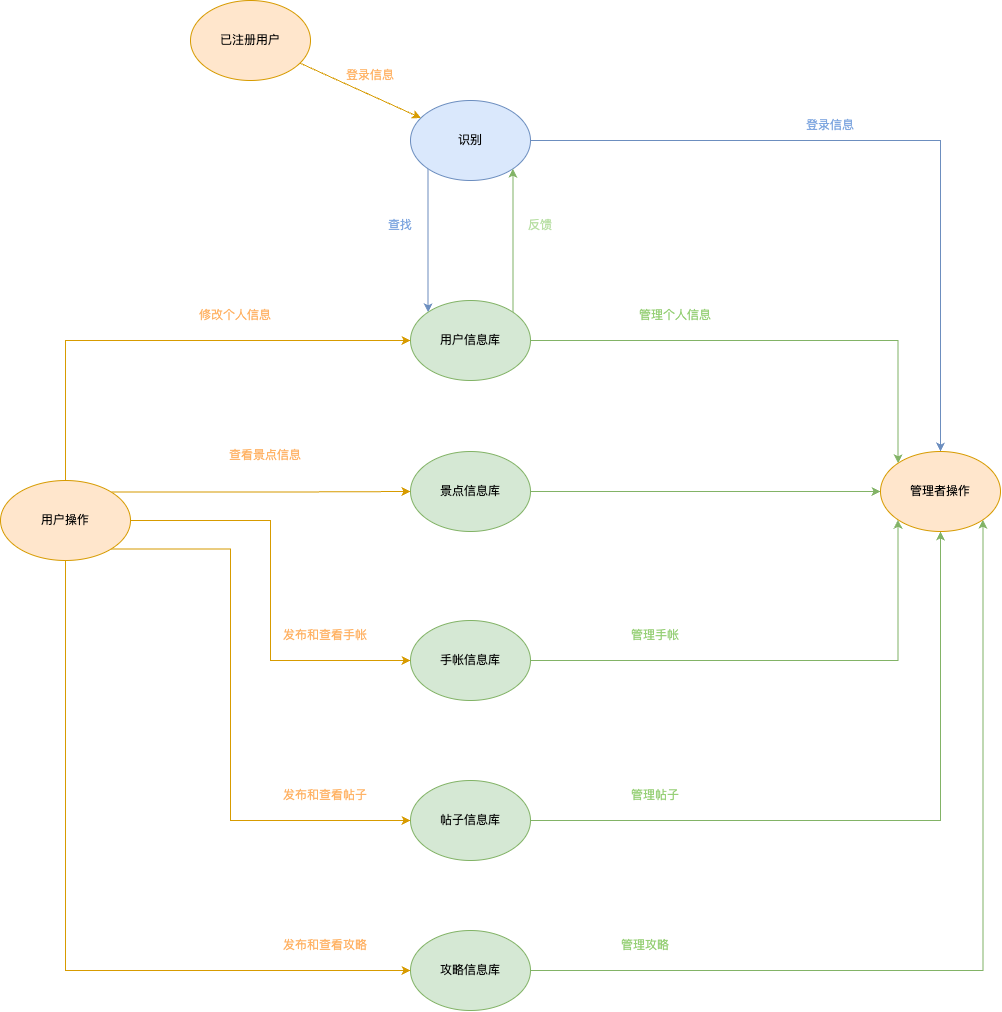


图2-2 项目数据流图

**2.2 用例图等**

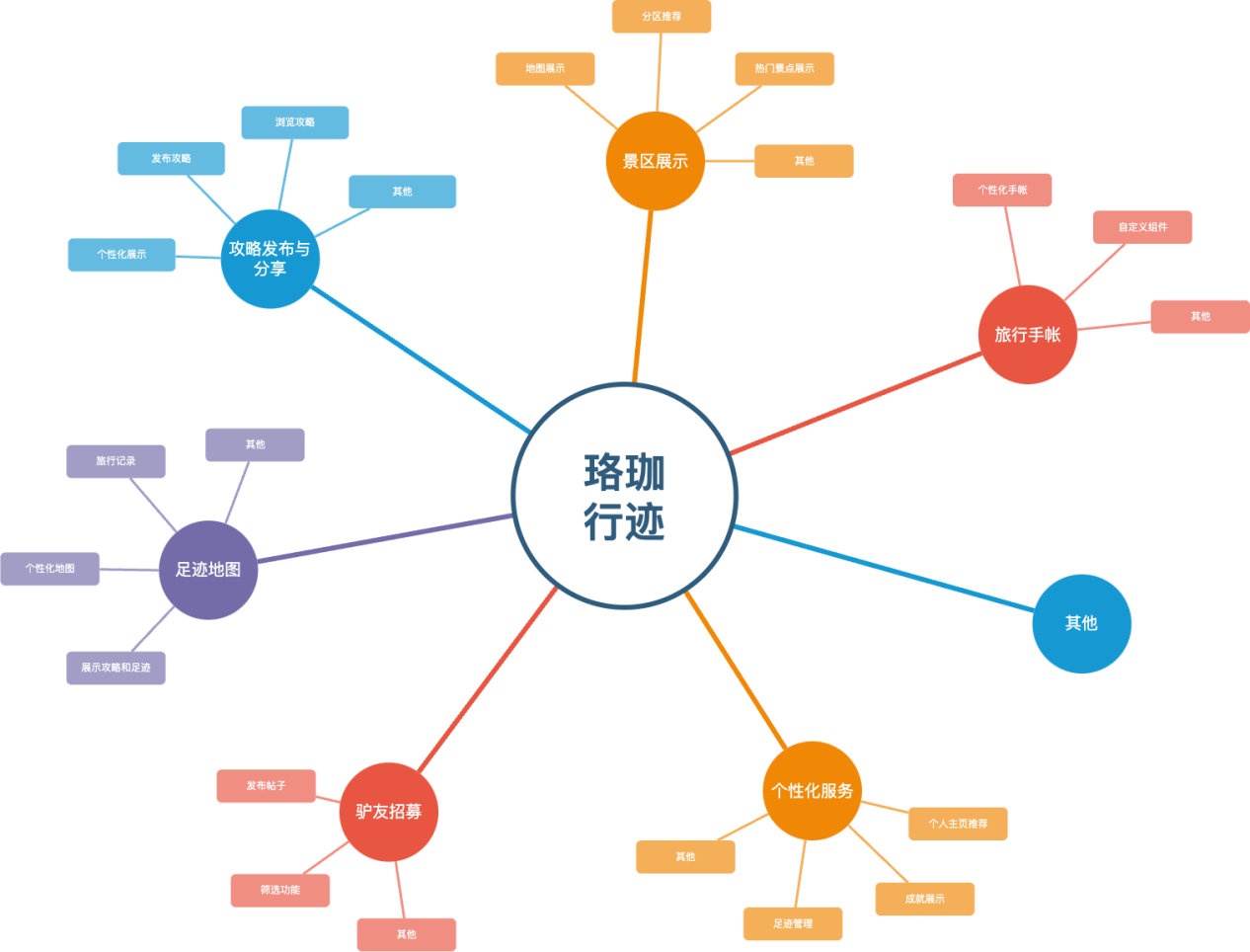


图2-3 用户用例图

**2.3 原型系统设计**

原型系统设计图使用墨刀进行设计。项目初期对于应用界面规划尚有部分不合理的地方，所以项目成果的应用界面和结构可能与原型系统中的设计有较大出入，原型系统设计图仅供参考。

**2.3.1 用户功能页面与简述**

我们为用户提供个性化的旅游与社交服务。通过该页面，用户可以管理个人信息、查看旅游记录、参与交友活动，并体验一系列定制化功能，旨在提升用户的旅行和交友体验。

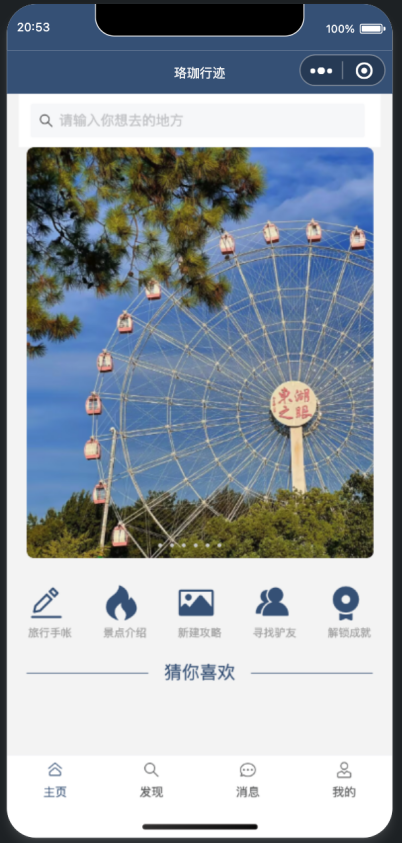
****

图2-4 主页面1-4

首页导航栏有四个选项：主页、发现、消息、我的。如上图所示。

其中主页包含了旅行手账、景点介绍、新建攻略、寻找驴友、解锁成就五个功能入口。

下面对几个已实现的功能进行简述：

**1.旅行手账**

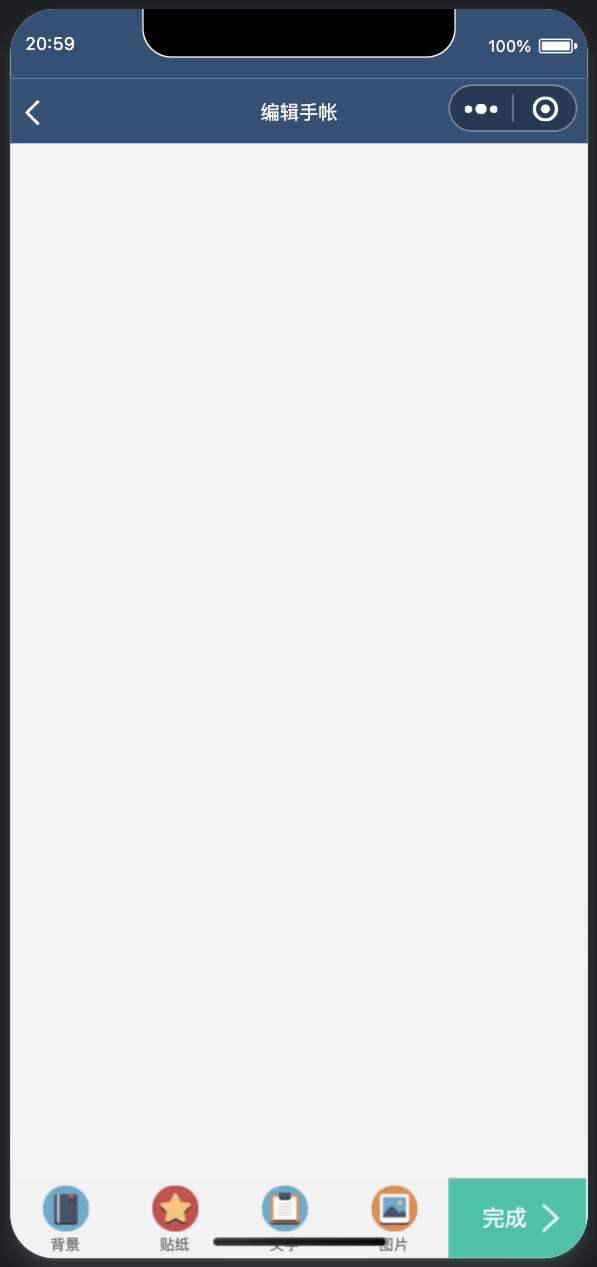


图2-5 旅行手账编辑页面

如图为旅行手账功能，在本编辑页面下如导航栏所示，我们为用户提供了丰富的贴纸、组件，可自由编写、设计个性化的旅行手帐，完成编辑后用户可生成预览图并保存至相册。

1. **景点介绍**

涵盖武汉市所有景区，并提供详细的地图和景点介绍，用户可以按季节、热度及区域查询景点。另外还设计了地图导览功能，如图所示。



图2-6 旅行地图

1. **攻略发布与分享**

支持用户发布多图、多标签的旅行攻略，并通过发现页的瀑布流展示，用户可以为攻略点赞，获取实时消息通知。

1. **招募驴友**

提供招募驴友的功能，用户可以发布、筛选并查找旅行伙伴，建立线下或线上交流圈。



图2-7 新建攻略和招募驴友页面

**3 概要设计和详细设计**

**3.1 系统结构**

项目系统主要分为三大模块，分别为：用户端模块、管理员端模块以及核心算法模块。

1. 用户端模块：用户端模块是普通用户（旅行者）可以直接接触的主要功能模块，负责为用户提供旅行辅助功能和交友互动体验。功能包括：提供景区推荐、旅行攻略查询和手账记录功能；实现景区评分、评论和旅行手账的分享功能；支持驴友招募信息的发布与参与，方便用户组队旅行；提供用户认证与权限管理功能，确保数据安全；集成底层算法模块的地理位置导航和推荐系统。
2. 管理员端模块：管理员端模块为平台管理人员提供高效管理工具，用于维护系统稳定运行，确保用户体验。功能包括：提供对景区、攻略和用户手账的内容审核功能；实现用户信息管理，包括禁用违规账号、更新用户权限等；管理驴友招募活动，检查发布内容是否合法合规；查看并分析用户使用数据，通过可视化工具辅助决策；支持从核心算法模块获取推荐数据和统计分析结果。
3. 核心算法模块：核心算法模块为系统提供技术支撑，是整个项目的基础模块，为用户端和管理员端提供智能化功能。功能包括：提供景区推荐算法，通过分析用户行为数据推送个性化景区；实现旅行攻略和驴友招募信息的全文检索算法，提高用户查询效率；支持用户足迹的路径规划与导航功能；为管理员端提供基于大数据的统计分析和趋势预测。

**3.2 类图等**

图略，详情请见3.3关键数据结构定义一节的介绍(包括系统中关键数据结构的类图)

**3.3关键数据结构定义**

### **3.3.1 前端代码框架**

首先介绍微信小程序的前端代码框架。微信小程序的前端框架是一个轻量级框架，专为小程序设计，基于 MVVM（Model-View-ViewModel）模式，旨在帮助开发者高效构建用户界面。

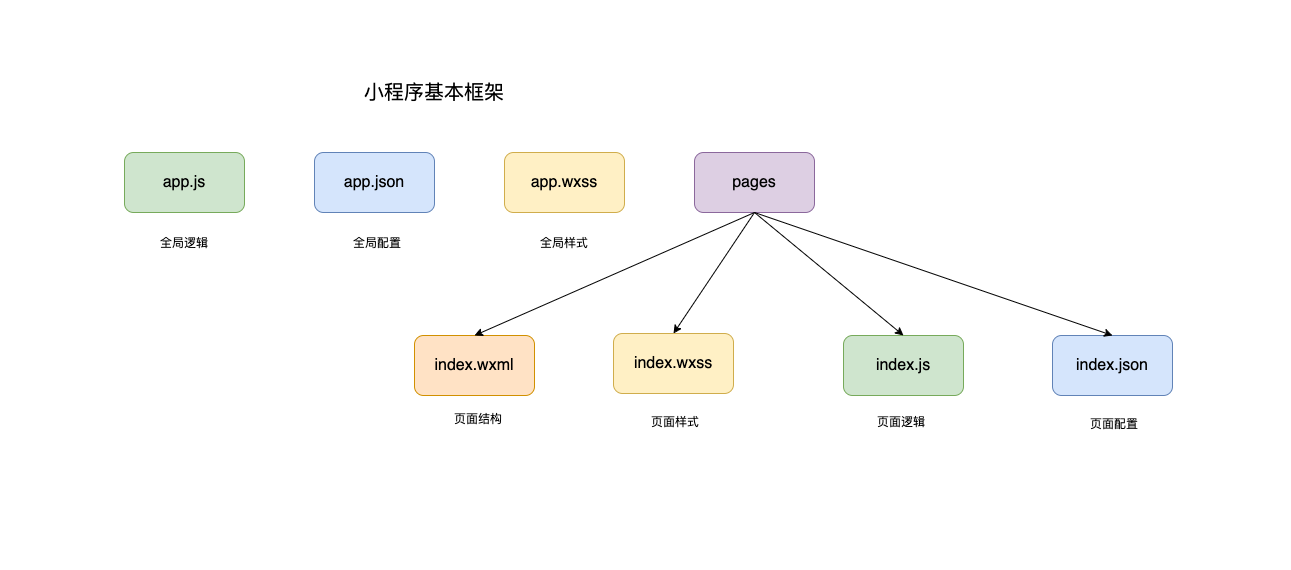


图3-1 前端页面框架

微信小程序主要有以下组成部分：

* ·WXML（WeChat Markup Language）

类似于HTML，用于描述页面的结构，提供了一些专有的标签（如 view、text）和属性，支持数据绑定。支持条件渲染（wx:if）和列表渲染（wx:for）。

* ·WXSS（WeChat Style Sheets）

类似于 CSS，用于页面样式的描述。支持大部分 CSS 样式规则，包含小程序独有的尺寸单位（如 rpx）以适应不同屏幕尺寸。

* ·JS（JavaScript）

用于实现页面的逻辑和数据处理。数据驱动：通过数据绑定（this.setData()）实现界面与数据的自动同步。支持调用微信提供的 API，比如网络请求、地理位置、支付等。

* ·JSON 配置文件

用于全局和页面级的配置，定义小程序的路由、导航栏样式以及页面窗口表现这些。

下面以主页的页面代码为例对数据结构进行说明：

*// pages/components/mainpage/mainpage.js*

var utils = require("../../utils/util.js")

Component({

  properties: {

  },

*/\*\**

   \* 组件的初始数据

   \*/

  data: {

    imgList: [

      {id: 166,imgSrc: '../../images/scenery/东湖樱园.jpg'},

      {id: 151,imgSrc: '../../images/scenery/长江大桥.jpg'},

      {id: 158,imgSrc: '../../images/scenery/黄鹤楼.jpg'},

      {id: 169,imgSrc: '../../images/scenery/古德寺.jpg'},

      {id: 21,imgSrc: '../../images/scenery/巴公馆.jpg'},

      {id: 678,imgSrc: '../../images/scenery/东湖之眼.jpg'}

    ],

    swiperCurrent: 0,

    showAuthButton: false,

    navList: [ // 宫格导航

      {name: '旅行手帐',  icon: 'edit', pageUrl: '../diary/diary'},

      {name: '景点介绍',  icon: 'fire', pageUrl: '../touristAttraction/touristAttraction'},

      {name: '新建攻略', icon: 'photo',pageUrl:'../publish/publish'},

      {name: '寻找驴友',  icon: 'friends',pageUrl:'../friend/friend'},

      {name: '解锁成就', icon: 'award',pageUrl: '../achievements/achievements'},

    ],

    skip: 0,//Math.ceil(Math.random()\*10),

    num: 5,

    leftList: [],

    rightList: [],

    guideData: [],

    noMore: false,

    backTopValue: false, // “返回顶部”

  },

*/\*\**

   \* 组件的方法列表

   \*/

  lifetimes:{

    attached: function(){

      this.getGuideData(this.data.skip, this.data.num)

    }

  },

  methods: {

*// 轮播图事件改变*

    swiperChange: function(e){

      this.setData({swiperCurrent: e.detail.current})

    },

*// 用户点击搜索栏,界面跳转到搜索*

    onSearchTap:()=>{

      console.log("tap")

      wx.navigateTo({

        url: "../search/search"

      })

    },

*// 点击宫格导航栏,界面跳转到对应的界面*

    tapNav:function (e){

      if(this.data.navList[e.target.id].pageUrl){

        wx.navigateTo({

          url: this.data.navList[e.target.id].pageUrl,

        })

      }else{

        wx.showToast({

          title:'点击:'+this.data.navList[e.target.id].name

        });

      }

    },

*// 进入到景区的详细界面*

    goAttractionDetail: function(e){

      var id\_ = e.target.id *// 景区id*

      utils.goattrDetail(id\_)

    },

    onClose: function(e){

      this.setData({showAuthButton:false})

    },

    getGuideData: function(){

      var that = this

       wx.cloud.callFunction({

         name: 'getEntry',

         data: {collection: 'guide', skip: that.data.skip,num: that.data.num, main:true},

        }).then(res=>{

         console.log(res)

         var left = that.data.leftList, right = that.data.rightList;

        //  if(res.result.data.length==0){

*//   that.setData({*

*//     noMore: true*

*//   })*

*// }*

*//  for(var i=0;i<res.result.data.length;i++){*

*//    if(i%2)*

*//      right.push(res.result.data[i])*

*//    else*

*//      left.push(res.result.data[i])*

*//    that.data.guideData.push(res.result.data[i])*

*//  }*

         var newSkip = that.data.skip+that.data.num

         that.setData({

           guideData: that.data.guideData,

           leftList: left,

           rightList: right,

           skip: newSkip

         })

       })

    },

    handleItemLeftChange:function(e){

      console.log(e.detail)

      var lList = this.data.leftList

      if(e.detail.type=="delLike"){

       lList[e.detail.index].like-=1

        lList[e.detail.index].amILike=false

      }else if(e.detail.type="addLike"){

        lList[e.detail.index].like+=1

        lList[e.detail.index].amILike=true

      }

      this.setData({

        leftList: lList

      })

    },

    handleItemRightChange:function(e){

      console.log(e.detail)

      var rList = this.data.rightList

      if(e.detail.type=="delLike"){

        rList[e.detail.index].like-=1

        rList[e.detail.index].amILike=false

      }else if(e.detail.type="addLike"){

        rList[e.detail.index].like+=1

        rList[e.detail.index].amILike=true

      }

      this.setData({

        rightList: rList

      })

    },

  }

})

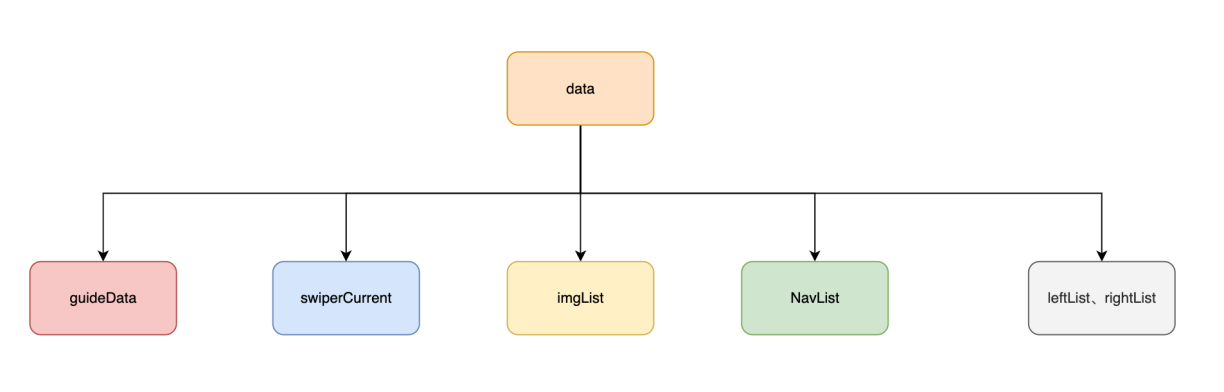


图3-2 data数据结构

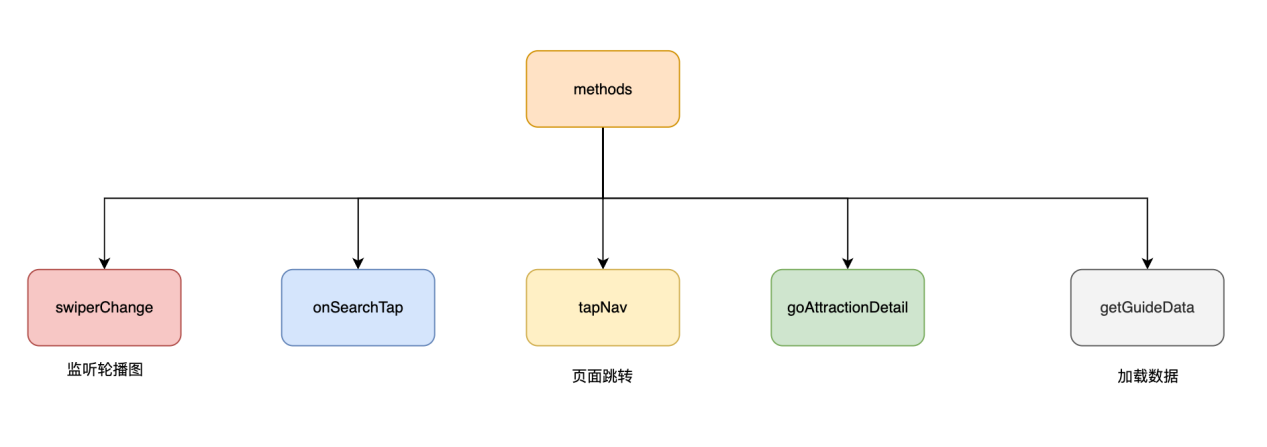


图3-3 methods数据结构

### **3.3.2 后端代码框架**

微信小程序的后端架构是基于微信云开发 (Cloud Development) 的，属于无服务器架构（Serverless Architecture）。后端的主要分为以下几个核心部分的数据结构。这里以deleteEntry云函数进行说明，代码如下：

*/\**

    模版函数：增加数据库中一条记录

    @param event (event.collection -表名, event.id -项id)

\*/

const cloud = require('wx-server-sdk')

cloud.init({env: 'luojia1cloud-7gbweippb2dee1e5'})

const db = cloud.database()

exports.main = async (event, context) => {

  if( event.isMessage!=null){

    console.log("del message!")

    return await db.collection('message').where({

       guide\_id: event.guide\_id

    }).remove().then(res=>{

    console.log(res)

  })

  }

  try{

    return await db.collection(event.collection).doc(event.\_id).remove().then(res=>{

      console.log(res)

    })

  }catch(e){

    console.log(e)

    return e

  }

}

云环境**const** cloud = require('wx-server-sdk') 作为开发环境，配置了

luojia1cloud-7gbweippb2dee1e5 作为我们的云，然后将云函数和数据库绑定，云函数通过 cloud.database() 获取数据库实例，与云开发环境直接通信，云数据库const db = cloud.database()。

**3.4 关键算法设计**

微信小程序的生命周期是通过生命周期函数进行管理的，包括多个关键节点： （1）onLaunch 是小程序初始化时触发的，仅在小程序启动或重新进入时调用一次，通常用于初始化全局数据，例如用户登录、获取用户信息或初始化云开发环境；

（2）onShow 会在小程序启动或从后台进入前台时触发，用于刷新全局数据、恢复应用状态或获取最新的通知信息；

（3）onHide 在小程序从前台进入后台时触发，主要用于保存状态、清理资源或停止后台任务，例如断开 WebSocket 连接；

（4）onError 会在小程序运行发生错误时触发，用于捕获全局错误并记录日志或上报错误；

（5）onUnhandledRejection 则用来捕获未处理的 Promise 拒绝（需启用特定配置），用于记录或处理此类错误；

（6）最后，onPageNotFound 在用户访问一个不存在的页面路径时触发，例如页面路径错误或页面被删除，这时可以用来记录日志或引导用户跳转到默认页面。

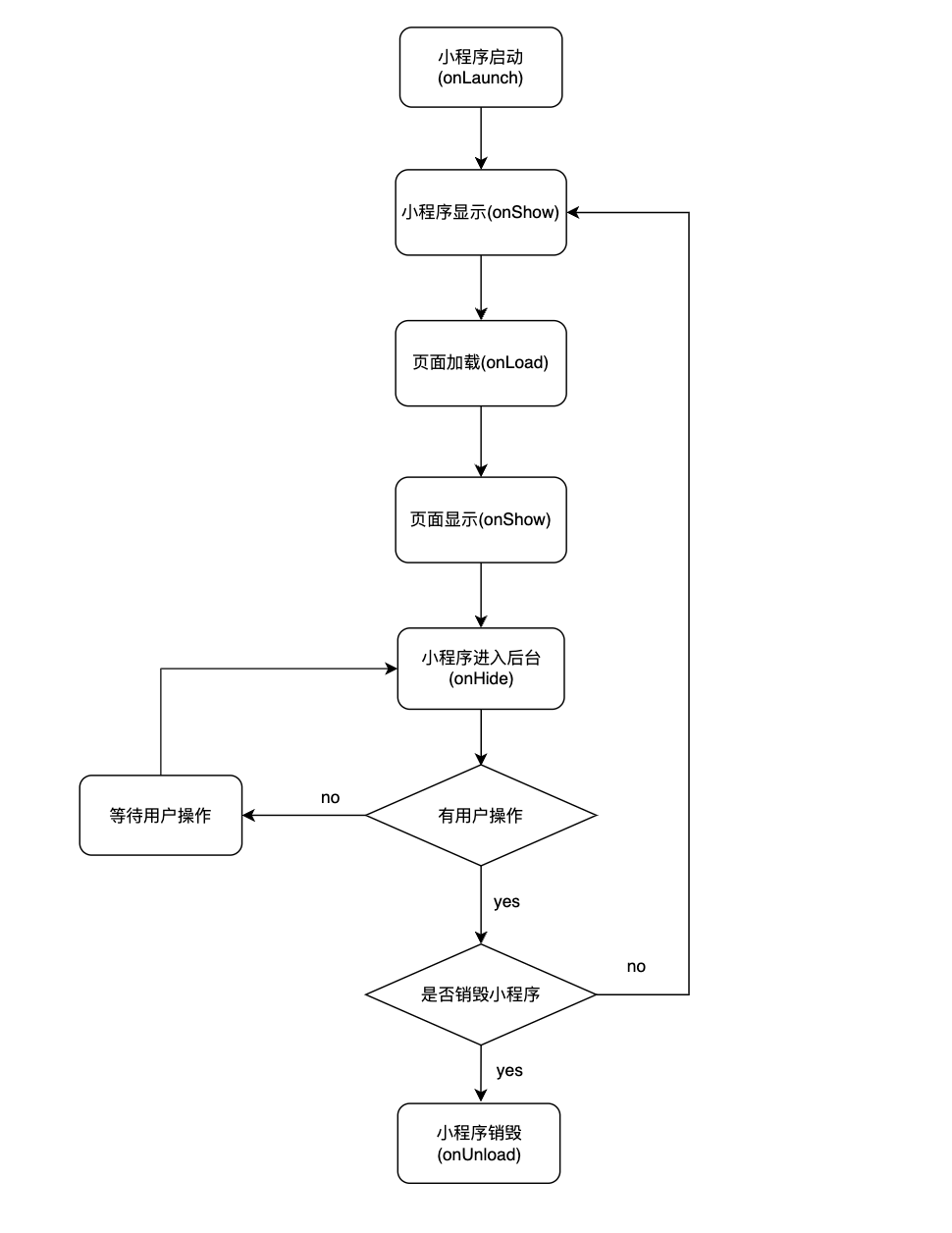


图3-4 小程序生命周期流程图

**3.5 数据管理说明**

### **3.5.1 数据来源** https://travel.qunar.com/p-cs300133-wuhan-jingdian

图3-5 数据来源网站图

**3.5.2 数据来源**  
文件名为武汉景点.csv，包括武汉景点263条数据，其中有很多属性大部分缺失且用不到，进行预处理后只保留地点、缩略图、排行、足迹、简介属性。

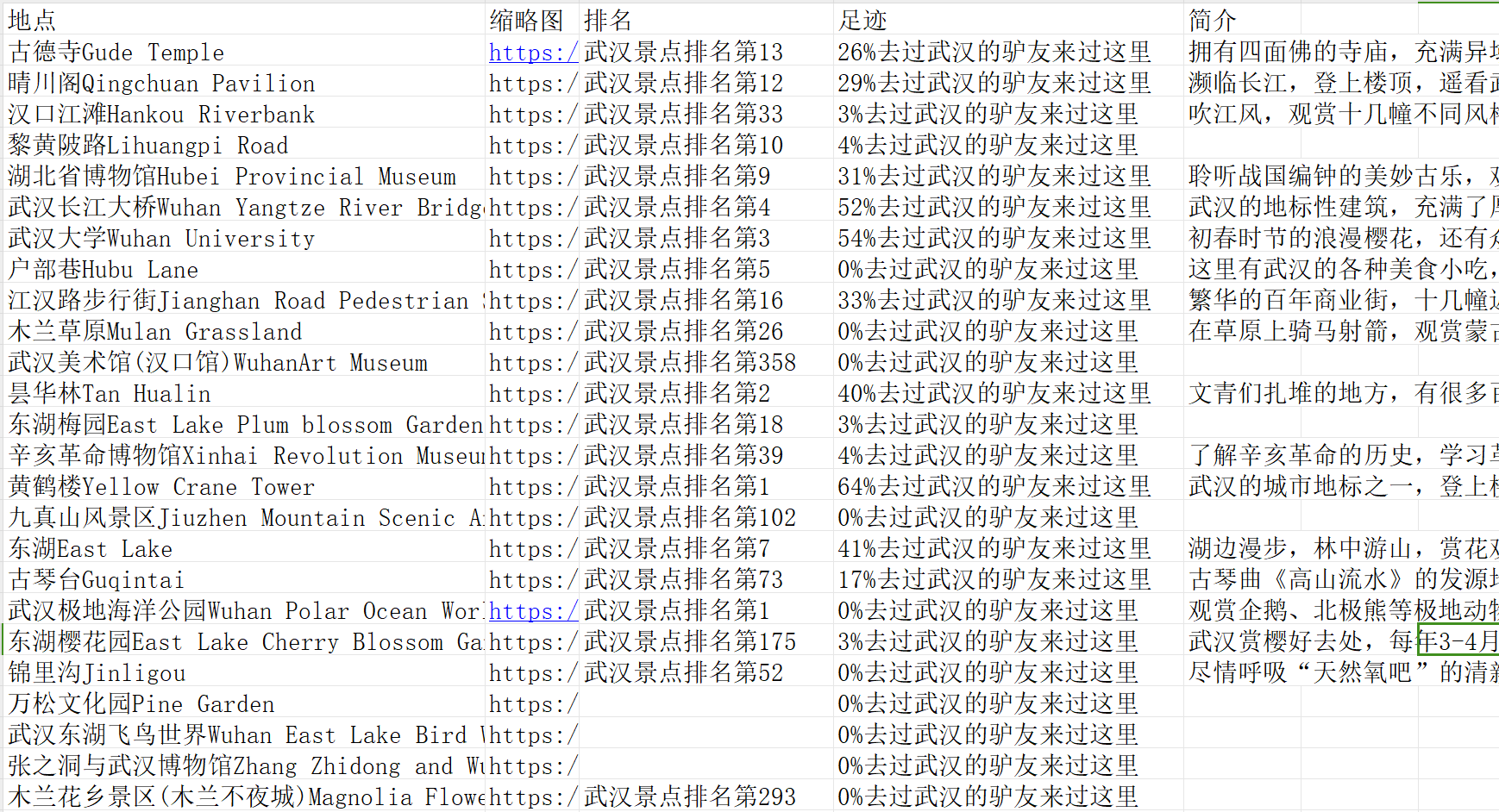


图3-6数据集内容图

**3.5.3 数据获取**  
数据集使用python爬虫获取  
1.使用模块  
**request**包用于向网站发送 HTTP 请求，并获取响应结果。  
**csv**包用于创建、读写 、修改文件  
**beautifulsoup**包用于从网页中抓取数据  
2.函数定义  
def get\_one\_page(i)  
包含url，head，data，用于爬取每一页上的内容，失败时会提示：“爬取失败”。  
def parse\_one\_page(html, ulist)  
解析网站每一页的数据并存入列表中。  
def processing\_data(ulist)  
通过检查列表中字符串长度来检测到格式不同的日期，并利用切片修改  
def save\_contents(ulist)  
预处理，新建csv文件，使用utf-8-sig编码，选择需要的内容，从列表中获取写入文件中。  
main  
通过每一页网页url根据页数有序变化的特性，利用循环调用函数，逐页获取内容，写入函数。

import requests from bs4

import BeautifulSoup

import csv

# 爬取每一页内容

def get\_one\_page(i):

url = f"http://example.com/page={i}" # 替换为实际网站的 URL

headers = { "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.0.0 Safari/537.36" }

try:

response = requests.get(url, headers=headers, timeout=10) response.raise\_for\_status() response.encoding = response.apparent\_encoding return response.text

except requests.RequestException as e:

print(f"爬取失败: {e}")

return None

# 解析每一页内容

def parse\_one\_page(html, ulist):

soup = BeautifulSoup(html, 'html.parser')

items = soup.find\_all('div', class\_='item-class') # 根据实际页面调整标签和类名

for item in items:

data = {

"title": item.find('h2').get\_text(strip=True),

"date": item.find('span', class\_='date-class').get\_text(strip=True),

"link": item.find('a')['href']

}

ulist.append(data)

# 处理数据

def processing\_data(ulist):

for entry in ulist:

date = entry['date']

if len(date) != 10: # 假设标准日期格式为 YYYY-MM-DD

entry['date'] = date[:10] # 切片获取标准长度日期

# 根据需要添加其他处理逻辑

# 保存内容到 CSV 文件

def save\_contents(ulist, filename="output.csv"):

with open(filename, 'w', newline='', encoding='utf-8-sig') as csvfile:

fieldnames = ['title', 'date', 'link']

writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

writer.writeheader()

for entry in ulist:

writer.writerow(entry)

# 主函数

def main():

ulist = []

num\_pages = 10 # 爬取的页数

for i in range(1, num\_pages + 1):

print(f"正在爬取第 {i} 页内容...")

html = get\_one\_page(i)

if html:

parse\_one\_page(html, ulist)

else:

print(f"跳过第 {i} 页")

print("数据处理...")

processing\_data(ulist)

print("保存内容到文件...")

save\_contents(ulist)

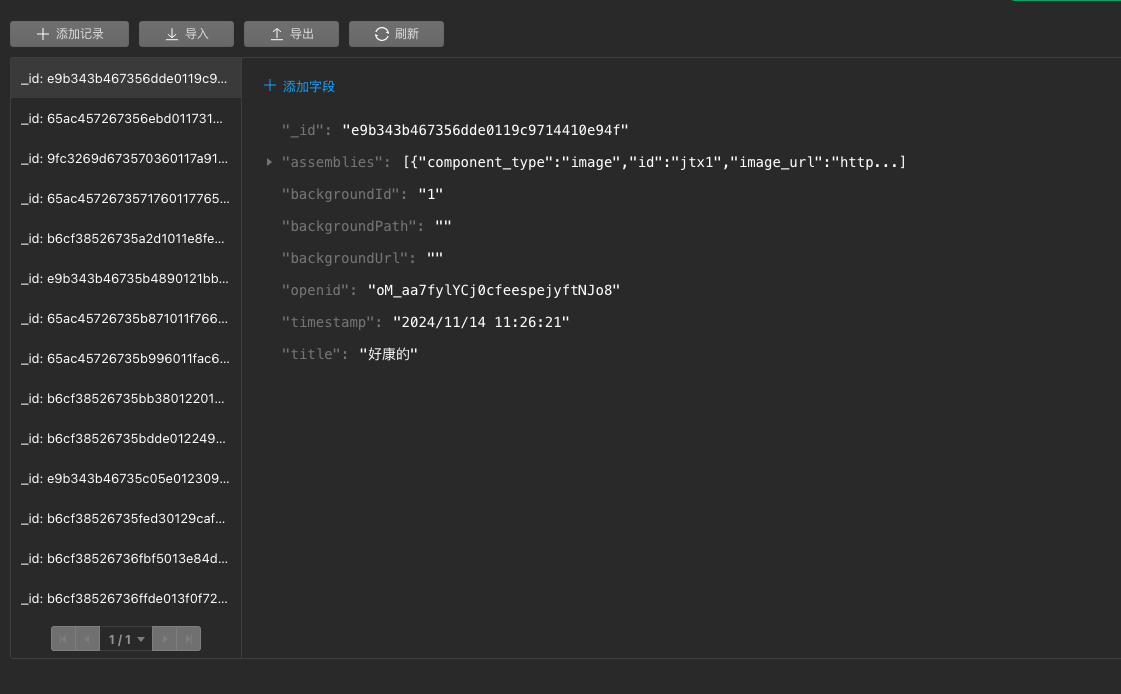
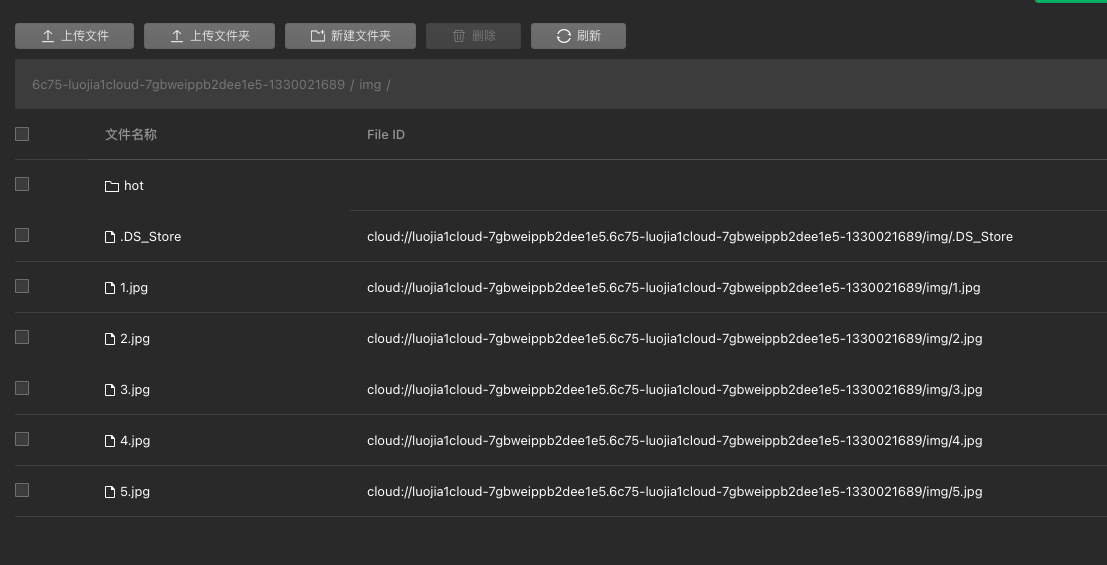
print("完成！")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**3.5.3 数据管理**

所有数据都存放在云环境中，通过云存储访问。在抓取到数据后上传到云存储，以便后续的数据处理和使用。

****图3-5 数据来源网站图

**4 实现与测试**

**4.1实现环境与代码管理**

* + 1. **实现环境**

在mac端进行软件开发，操作系统为macOS Sonoma 14.6.1，使用微信开发者工具进行微信小程序的开发，后续的测试也在微信开发者工具的模拟器中进行。

* + 1. **代码版本管理的签入记录**

****

图4-1 github平台代码签入示例

**4.2 关键函数说明**

在所有微信小程序的开发中，app.js 是小程序的全局文件，用于定义小程序的生命周期函数和全局数据。它在整个小程序的运行中起着核心的管理作用，是小程序的入口文件。这里针对app.js进行关键函数的说明：

*//app.js*

App({

  onLaunch: function () {

*// 开启云开发*

    if(!wx.cloud){

      console.error('请升级版本使用云开发功能')

    }else{

      wx.cloud.init({

        env: 'luojia1cloud-7gbweippb2dee1e5',

      })

    }

*// 登录*

    wx.login({

      success: res => {

*// 发送 res.code 到后台换取 openId, sessionKey, unionId*

        console.log("code",res.code)

      }

    })

*// 从缓存中加载用户信息*

    var user = wx.getStorageSync('user')

    if(user){

      this.globalData.userInfo = user

*// 打印用户信息*

      console.log("用户昵称:"+this.globalData.userInfo.nickName)

      console.log("用户头像地址:"+this.globalData.userInfo.avatarUrl)

*// 开启攻略点赞监听*

     const db=wx.cloud.database()

     const watcher\_ = db.collection('message').where({

        target\_nickname: this.globalData.userInfo.nickName,

      }).watch({

        onChange: function(snapshot){

         if(snapshot.type == 'init') {

           wx.cloud.callFunction({

             name: 'cf',

             success: function(res) {

               wx.cloud.callFunction({

                 name: 'cf3',

                 success: function(res2) {

                   if(res.result.message\_count < res2.result.message\_count)

                     that.globalData.\_hasNewMessage = true

                 }

               })

             }

           })

         }

         else if(snapshot.docChanges[snapshot.docChanges.length - 1].dataType == 'add')

           that.globalData.\_hasNewMessage = true

         wx.cloud.callFunction({

           name: 'cf2'

         })

        },

        onError:function(err){

         console.error('watch closed because of error',err)

       }

      })

    }

  },

  globalData: {

*// 全局信息： userInfo(用户信息), \_hasNewMessage(是否有新的点赞消息)*

    userInfo: null,

    \_hasNewMessage: false

  },

*// 监控hasNewMessage*

  watch: function(method){

    var obj = this.globalData

    Object.defineProperty(obj, "\_hasNewMessage",{

      configurable: true,

      enumerable: true,

      set: function(value){

        method(value)

      },

      get:function(){

        return this.\_hasNewMessage

      }

    })

  },

*/\**

      云数据库查询

      @param: setName 查询的数据库名称

      @param: ruleObj 查询的字段

      @param: callback 查询成功的回调函数

  \*/

  getInfoWhere: function(setName, ruleObj, callback){

    const db=wx.cloud.database()

    db.collection(setName).where(ruleObj)

      .get({

        success: callback,

        fail: console.error

      })

  },

})

（1）onLaunch：小程序的生命周期函数，初始化时触发一次，用于全局设置和初始化。检查是否支持云开发，支持时初始化云开发环境，设置 env 为 'luojia1cloud-7gbweippb2dee1e5'。使用 wx.login 获取用户的登录凭证 code，可以将其发送到后端换取用户的 openId, sessionKey, unionId。

（2）watcher\_：通过云数据库的实时数据监听功能 (db.collection().watch) 实现对用户点赞消息的实时监控。监听云数据库 message 集合中目标昵称（target\_nickname）为当前用户昵称的记录，实时更新 \_hasNewMessage 标志。

（3）watch：动态监控 \_hasNewMessage 变量。

（4）getInfoWhere：封装了云数据库查询的通用函数。根据集合名称 (setName) 和查询条件 (ruleObj) 执行查询操作，并在查询成功时调用指定的回调函数 (callback)。

（5）globalData：存储小程序的全局数据，所有页面都可以访问。

**4.3 测试计划和测试用例**

1.登陆测试：

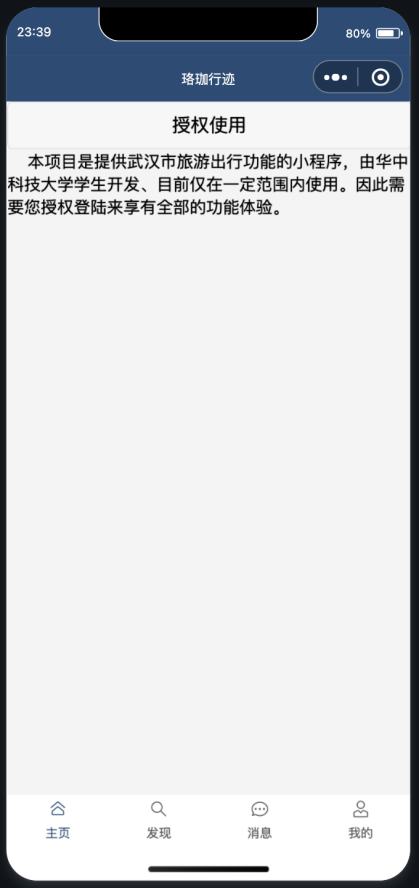
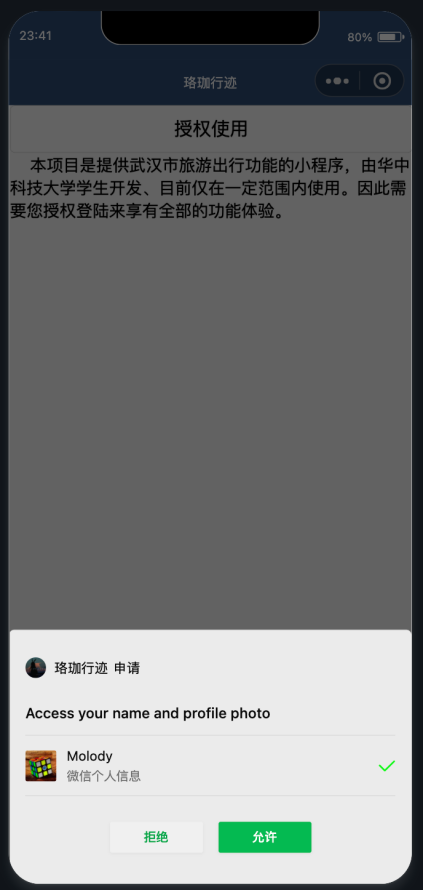
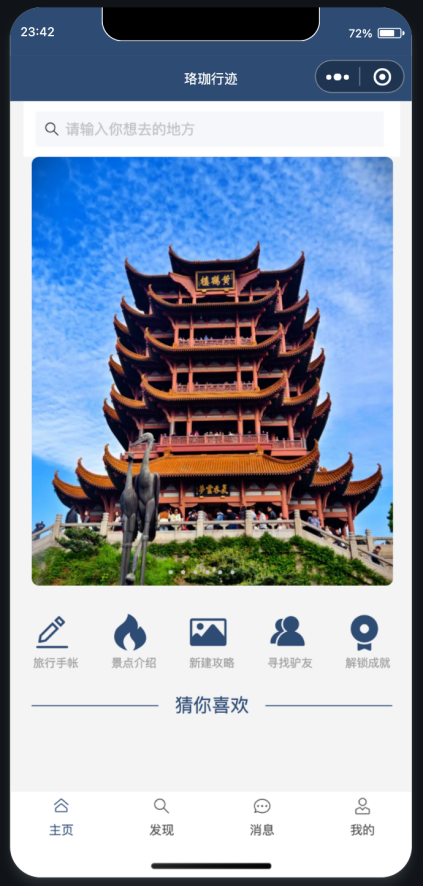


图4-2 登陆测试

2.个性地图测试：



图4-3 个性地图测试

3.旅行手帐测试：

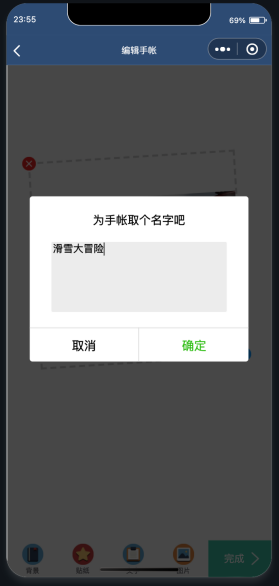
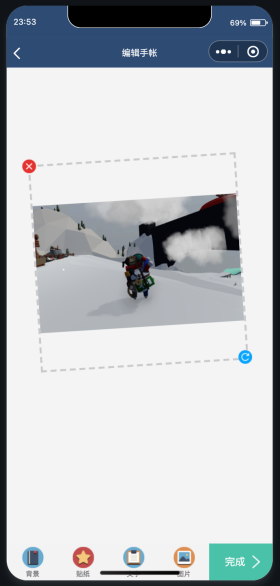
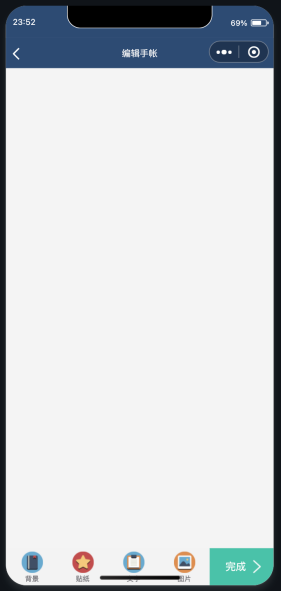
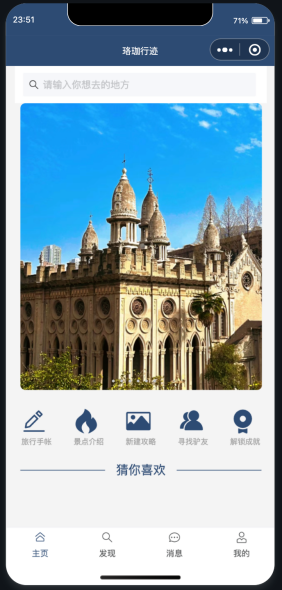


图4-4 旅行手帐测试

4.景点介绍测试：



图4-5 景点介绍测试

**4.4 结果分析**

根据测试分析运行结果，用户登录、旅行手帐制作、景点介绍等功能均正确，已经实现了用户身份验证、个性化内容创建以及信息查询的核心需求。此外，宫格导航、景点详情浏览和搜索功能操作流畅，数据同步及时，整体交互符合用户预期。通过模拟不同用户场景和高并发访问测试，系统表现稳定，关键功能响应迅速，满足应用上线的基本要求，为后续的用户体验优化和功能扩展打下了坚实基础。

**5 总结**

**5.1 用户反馈**

我们邀请了三位同学来体验“珞珈行迹——武汉市旅行交友攻略手账“微信小程序，并收集了它们在使用过程中的意见反馈。

用户1：APP的画面简洁美观，看起来很舒服，操作也方便，可以快速找到自己想要使用的功能，可以分享自己旅行过程中的体验，也能记录旅行过程中的人与事，体验很好。对景点的介绍可以再详细一些，这样能更清楚的了解这些景点。

用户2：涵盖了武汉市的热门景点，可以了解景点的信息，也能看别人的分享评价，还内置地图，不用再开一个地图软件，出去游玩的时候很好用。但是可以调整下地图入口的位置，把它放到主界面，这样打开地图看路更方便。

用户3：APP里面有各个热门景点的攻略评价，自己也能在里面分享旅游攻略，而且还能在这个平台上认识更多的朋友，大家可以一起旅游，很有特点，出去旅游很有用。但是APP界面可以做得再丰富一些，让人能打开就感觉到赏心悦目。

**5.2 全文总结**

市场上虽然有许多旅游类的APP和小程序，但大多功能较为单一，主要集中在景点介绍或攻略分享。部分应用可能会提供手帐功能，但设计较为简陋，缺乏灵活性。此外，市场上的大多数驴友招募类产品，筛选功能不够强大，无法满足用户的精准需求。

因此我们制作了“珞珈行迹——武汉市旅行交友攻略手账“微信小程序，包含六大核心功能，景区展示：覆盖武汉市的所有景区，包括地图展示、分区推荐、热门景点展示等。旅行手帐：提供个性化手帐功能，支持用户通过贴纸和自定义组件来记录旅行。攻略发布与分享：用户可以发布并浏览多图、标签的旅行攻略，系统会根据推荐算法提供个性化展示。足迹地图：基于用户的旅行记录生成武汉市个性化足迹地图，展示攻略和足迹。驴友招募：支持用户发布招募驴友的帖子，提供筛选功能，帮助用户找到合适的旅行伙伴。个性化服务：包括用户的足迹管理、成就展示及个人主页推荐。

在这款旅行交友微信小程序中，用户可以轻松获得全面的武汉市景区信息、自由编写个性化的旅行手帐，并方便地发布、查阅旅行攻略。此外，用户可以通过我们的平台找到志同道合的驴友，组织丰富的线下或线上旅行活动。多种功能结合，让用户既能享受深度旅行体验，又能在过程中收获更多社交互动和乐趣。特别是提供的个性化足迹地图和成就系统，增强了用户的归属感和成就感。

**6 体会**

在这次开发作“珞珈行迹——武汉市旅行交友攻略手账“微信小程序的过程中， 我们经历了项目立项，原型设计，产品设计，推广测试几个阶段。这几个阶段中我们遇到了一些有代表性的问题。

微信小程序需要搭建云开发环境，但此前我们没有搭建云开发环境的经验。在搭建过程中，发现程序出现了在本地环境能运行，但在云开发环境上不能运行的情况，经过不断调试后，发现本地代码没有问题，但是云环境配置出现错误，导致代码在本地运行正常，在云环境下运行错误，我们随后修改了云环境配置，使得代码能正常运行。

在开发“珞珈行迹——武汉市旅行交友攻略手账“微信小程序的过程中，我们边学边开发，一边学习软件开发的相关知识，一边开发软件。整个开发流程下来，我们不仅学到了很多软件开发的知识，也增强了编写代码的实践能力，对软件开发积累了经验，更为将来的深入学习打下基础。