个性化旅游系统总体方案设计报告

一、引言

（一）编写目的

本报告旨在依据《数据结构课程设计 评分原则-2025.pdf》（以下简称“评分标准”），为“个性化旅游系统”提供符合课程要求的总体设计框架。通过明确系统架构、模块划分、算法实现及部署方案，确保开发过程满足评分标准中的数据规模、功能完整性及算法设计要求，为团队开发、测试及验收提供标准化指导。

（二）背景

根据评分标准，课程设计需实现集旅游推荐、路线规划、场所查询、日记管理于一体的系统，核心要求包括：

- 数据规模：景区/校园≥200个，建筑物≥20个/景区，服务设施≥50个（≥10种），道路边数≥200条，用户≥10人；

- 算法要求：推荐功能需采用排序算法（如部分排序），路线规划需实现最短路径算法（如Dijkstra），场所查询需基于实际距离排序，日记管理需支持压缩与全文检索；

- 文档与代码一致性：报告需详细说明算法设计、模块分工，且代码需与报告描述一致。

（三）范围

- 功能范围：覆盖评分标准要求的核心功能（旅游推荐、路线规划、场所查询、日记管理），不包含选做功能（如室内导航、美食搜索）。

- 技术范围：采用Spring Boot（后端）+ Vue.js（前端）架构，MySQL存储数据，算法实现需符合评分标准指定的基础要求（如禁止直接调用第三方算法库）。

二、总体设计概述

（一）设计目标

1. 功能达标：严格满足评分标准的数据规模与功能要求，确保所有核心功能可演示运行（如推荐Top-10景点、多目标路径规划）。

2. 算法合规：核心算法（排序、最短路径、索引）均基于数据结构课程知识点设计，如用最大堆实现部分排序，用Dijkstra算法求解最短路径。

3. 性能优化：推荐响应时间≤1秒，路径规划时间≤3秒，支持≥10用户并发访问。

（二）设计原则

| 原则 | 说明 |

|---------------|----------------------------------------------------------------------|

| 算法优先 | 所有功能均以课程知识点为基础，如推荐功能基于排序算法，路线规划基于图论。 |

| 模块化设计| 前后端分离，各功能模块独立（如推荐模块、路线规划模块），降低耦合度。 |

| 数据合规性| 严格按照评分标准生成测试数据（如200+景点、200+道路边），确保验收通过。 |

（三）总体架构

```mermaid

graph LR

用户层 --> 前端层(Vue.js)

前端层 --> API网关(Spring Boot)

API网关 --> 服务层{推荐/路线/查询/日记}

服务层 --> 持久层(MyBatis-Plus)

持久层 --> MySQL数据库

服务层 --> 算法模块{堆排序/Dijkstra/R树/Huffman}

```

- 前端层：负责界面展示与用户交互，使用Vue组件实现景点列表、地图导航等功能，通过Axios调用后端API。

- 服务层：实现业务逻辑，如旅游推荐调用排序算法，路线规划调用Dijkstra算法，日记管理调用压缩算法。

- 持久层：基于MyBatis-Plus操作MySQL，存储景点、道路、用户、日记等数据，确保数据规模达标。

- 算法模块：独立实现评分标准要求的核心算法，与服务层解耦，便于测试与优化。

三、系统模块划分

（一）模块列表及功能描述

| **模块名称** | **功能描述** | **评分关联点** |
| --- | --- | --- |
| 旅游推荐模块 | 1. 按热度/评价/兴趣推荐Top-10景点/学校，采用最大堆排序实现部分排序；<br>2. 支持名称/类别/关键字搜索，结合哈希表与倒排索引实现模糊查询。 | 验证部分排序算法、查找算法 |
| 路线规划模块 | 1. 单目标导航：基于Dijkstra算法生成最短距离/时间路线；<br>2. 多目标导航：通过贪心算法规划闭环路线，支持交通工具混合策略（步行/自行车/电瓶车）。 | 验证最短路径算法、分层图模型 |
| 场所查询模块 | 1. 搜索周边设施，按实际步行距离排序（非直线距离），采用R树索引加速范围查询；<br>2. 支持类别过滤与关键词检索，结果排序采用快速排序。 | 验证空间索引、非直线距离排序 |
| 日记管理模块 | 1. 支持图文视频上传，内容采用Huffman编码压缩存储；<br>2. 提供名称精确查询（哈希表）、内容全文检索（文本搜索算法），按热度/评分推荐日记。 | 验证无损压缩、全文检索算法 |

（二）模块关系

图形用户界面

AI 生成的内容可能不正确。

- 数据依赖：各模块通过持久层获取基础数据（如景点列表、道路边数据），确保数据规模符合评分标准。

- 算法协同：推荐与日记模块共享排序算法，路线与查询模块共享图论与索引算法，避免重复开发。

四、技术选型

（一）开发语言

- Java（后端）：符合评分标准对算法自主实现的要求，Spring Boot框架可快速搭建API，集成JPA实现数据库操作。

- JavaScript（前端）：Vue.js便于实现动态交互界面，配合Element Plus组件库提升开发效率。

（二）框架与工具

| **工具/框架** | **作用** | **评分适配点** |
| --- | --- | --- |
| Spring Boot | 提供自动配置与依赖管理，快速实现RESTful API，支持跨域请求配置。 | 确保后端接口符合评分演示要求 |
| MyBatis-Plus | 简化数据库操作，支持批量插入数据（如生成200+景点），提升数据初始化效率。 | 满足数据规模快速填充需求 |
| 码上/通义灵码 | 辅助设计算法逻辑（如对比堆排序与快速选择性能），但核心代码需自主编写。 | 符合评分标准对大模型辅助的要求 |

（三）数据库设计

- MySQL 8.0：

- 核心表结构：

| **表名** | **关键字段** | **数据规模要求** |
| --- | --- | --- |
| attraction | id, title, hot, rating, area\_type（景区/校园）, coordinates（坐标） | ≥200条记录 |
| road | start\_id, end\_id, distance, crowdedness, transport\_type（步行/自行车/电瓶车） | ≥200条边 |
| user | id, username, interests（兴趣标签）, rated\_diary\_ids（评分日记ID列表） | ≥10个用户 |
| diary | id, user\_id, content（压缩后字节流）, tags, destination\_id, hot, rating | 日记数量≥用户数×5 |

五、系统接口设计

（一）外部接口（前端→后端）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **接口名称** | **方法** | **路径** | **请求参数** | **响应数据** | **评分验证点** |
| 获取推荐景点 | GET | /api/recommendations | sort=hot（热度）/<br>sort=rating（评分）/<br>interest=自然景观（兴趣标签） | List<Attraction>（前10项，含hot、rating） | 验证部分排序（非全量排序） |
| 规划路线 | POST | /api/route-planning | start=1&end=5&strategy=distance（最短距离）/<br>strategy=time（最短时间） | Route（节点序列、总距离/时间、路径坐标） | 验证Dijkstra算法及策略计算 |
| 搜索附近设施 | GET | /api/nearby-facilities | lat=30.12&lng=120.34&radius=500&type=餐饮 | List<Facility>（名称、距离、坐标，按距离升序） | 验证R树索引与实际距离排序 |
| 上传日记 | POST | /api/diaries | title, content, images[], destination\_id, tags | DiaryID | 验证Huffman压缩存储 |

（二）内部接口（模块间调用）



六、系统部署方案

（一）硬件环境

- 开发阶段：

- PC机：CPU≥i5-8400，内存≥16GB，硬盘≥500GB（存储测试数据）。

- 验收阶段：

- 云服务器（如阿里云ECS）：2核4GB配置，用于演示系统并发能力（≥10用户同时访问）。

（二）软件环境

| **组件** | **版本** | **作用** | **评分适配** |
| --- | --- | --- | --- |
| Java | 17 | 运行Spring Boot后端应用 | 符合课程指定开发环境 |
| MySQL | 8.0 | 存储数据 | 支持事务与复杂查询 |
| Node.js | 16+ | 运行前端开发服务器 | 确保Vue.js组件正常编译 |
| Postman | 最新版 | 接口测试 | 验证接口功能与性能 |

- 说明：前端静态文件通过Nginx部署，后端API通过Spring Boot提供，数据库与后端服务器部署在同一主机，确保低延迟数据交互。

注：本报告严格遵循评分标准，所有设计均围绕核心功能与算法展开，确保代码实现与报告描述一致，满足课程设计验收要求。