1. 系统概述

1.1 项目背景

志愿填报系统旨在帮助学生科学合理地选择高考志愿，提供数据分析、推荐和填报功能，解决志愿填报复杂、信息不足的问题。

1.2 项目目标

提供多维数据查询，支持考生快速筛选合适的学校及专业。

实现智能推荐功能，通过算法分析匹配度。

支持志愿模拟填报和保存，为考生提供多样化选择。

2. 系统需求分析

2.1 功能需求

用户模块

用户注册、登录及权限管理。

用户信息管理（成绩、意向区域等）。

数据查询模块

学校及专业信息查询（按分数线、区域、排名等筛选）。

数据统计及可视化（历史录取分布等）。

智能推荐模块

基于用户成绩和偏好的志愿推荐。

模拟填报及分析建议。

管理模块

数据管理（学校信息、专业信息、分数线等）。

系统设置（志愿批次、填报规则等）。

2.2 非功能需求

性能：支持并发用户数 ≥ 10,000。

安全性：采用 HTTPS 通信，用户数据加密存储。

可用性：系统可用性 ≥ 99.9%，支持高可用部署。

扩展性：支持学校、专业数据的动态更新和扩展。

3. 系统架构设计

3.1 技术栈

前端：React + Next.js + Tailwind CSS

后端：Node.js（Express）或 Python（Django/FastAPI）

数据库：MySQL/PostgreSQL

算法模块：Python（推荐系统使用机器学习或基于规则的算法）

部署：Docker + Kubernetes，云服务（如 AWS/GCP/阿里云）

3.2 系统架构图

前端：提供用户交互界面。

后端：提供数据接口及业务逻辑。

数据库：存储用户信息、院校数据、分数线等。

推荐系统：使用算法模块对志愿进行推荐。

4. 模块设计

4.1 用户模块

功能：

用户注册、登录。

修改个人信息，上传成绩。

接口设计：

POST /api/register：用户注册。

POST /api/login：用户登录。

GET /api/user/profile：获取用户信息。

4.2 数据查询模块

功能：

学校、专业查询，筛选条件包括分数线、区域等。

数据可视化展示。

接口设计：

GET /api/schools：获取学校列表。

GET /api/majors：获取专业列表。

GET /api/statistics：获取录取分数线分布。

4.3 智能推荐模块

功能：

基于考生信息和数据的推荐算法。

提供志愿模拟填报建议。

接口设计：

POST /api/recommend：获取推荐学校和专业。

4.4 管理模块

功能：

增删改查院校、专业及分数线数据。

配置志愿填报批次规则。

接口设计：

POST /api/admin/schools：添加学校信息。

DELETE /api/admin/schools/:id：删除学校。

5. 数据库设计

5.1 数据表

用户表（users）

id, username, password, email, score, region

学校表（schools）

id, name, province, ranking, category

专业表（majors）

id, name, school\_id, min\_score, max\_score

推荐记录表（recommendations）

id, user\_id, school\_id, major\_id, recommendation\_score

5.2 数据关系

用户与推荐记录：一对多。

学校与专业：一对多。

6. 接口设计

6.1 接口规范

请求格式：

使用 RESTful API。

数据格式为 JSON。

响应结构：

{

"success": true,

"data": {...},

"message": "Operation completed successfully"

}

7. 测试计划

7.1 测试类型

功能测试：验证各模块功能是否按预期工作。

性能测试：模拟并发访问，测试系统稳定性。

安全测试：验证用户数据是否加密存储，防止 SQL 注入。

用户体验测试：收集用户反馈，优化系统设计。

7.2 测试用例示例

测试模块测试项预期结果

用户模块注册功能用户成功注册，返回用户 ID。

数据查询模块按分数线筛选学校返回符合筛选条件的学校列表。

推荐模块提供智能推荐功能返回与用户成绩匹配的推荐学校。

8. 部署方案

8.1 部署环境

操作系统：Linux（Ubuntu 20.04）

服务器：Nginx + Docker

数据库：MySQL 8.0

推荐系统：独立服务通过 gRPC 或 REST API 与主系统交互。

8.2 部署流程

1.拉取代码仓库并安装依赖。

2.构建前端项目并部署至 Nginx。

3.配置后端服务，设置环境变量，启动 Docker 容器。

4.初始化数据库，导入基础数据。

9. 附录

9.1 参考资料

高考志愿填报指南。

数据算法参考书籍。

系统设计模式相关文档。

9.2 团队联系方式

开发负责人：XXX

技术支持邮箱：XXX@domain.com

这个文档可作为志愿填报系统的全面指导，具体内容可以根据项目的实际需求进行调整和扩展。