排课系统设计文档

1 引言

本设计文档旨在详细阐述排课系统的设计方案，为系统的开发、测试和部署提供明确的指导。

2 实体关系

本系统的核心实体及其关系如下：

1. **用户（User）：**系统操作者，分为管理员、教师和学生。
2. **管理员（Admin）：**排课系统管理员，负责系统的信息管理和用户的权限分配，进行排课操作。
3. **教师（Teacher）：**拥有个人信息、可授课程等属性。
4. **学生（Student）：**拥有个人信息、所上课程、所属专业等属性。
5. **课程（Course）：**排课内容的基本单元，包含课程编号、名称、学时等信息。
6. **班级（Class）：**学生的集合，也是教学活动的基本单位。
7. **教师（Room）：**上课的物理空间，包含教室编号、容量、类型（如普通教室、实验室）等属性。
8. **上课时间（Schedule）：**定义了上课的具体时间。
9. **学习资料（Study\_logs）:**包含每个课程对应的教材（教材名称，教材编号等），对应的试卷，以及学习文档。
10. **网络课程（Online\_course）:**包含线上课程信息（课程名称，课程编号等）。

实体间的关系描述：

一个教师可以教授多门课程，一门课程也可以由多名教师讲授。

一个学生可以选修多门课程，一门课程也可以被多名学生选修。

一个教师可以教授多名学生，一个学生也可以被多名教师教授。

一个学生关联唯一的班级。

教室和上课时间之间存在“占用”关系。

3 功能

|  |  |
| --- | --- |
| 功能模块 | 描述 |
| 基础信息管理 | **用户管理：**  对教师、学生角色的增删改查及权限分配。  **教师管理：**  维护教师基本信息、联系方式、职称、可授课程列表等。  **学生管理：**  管理学生学籍信息、所在班级、所上课程等。  **课程管理：**  维护课程数据库。包括课程名称、学时等。  **教室管理：**  管理教室资源。包括教室号、容量、教室类型等信息。  **班级管理：**  对教学班进行创建、编辑、删除和查询等操作。 |
| 排课规则设置 | **时间规则：**  课程必须安排在有效的时间段内。  **学期规则:**  排课是按学期进行的。  **教室规划：**  课程需要分配到可用教室；教室容量需满足班级人数；教室时间不能冲突。  **班级规则：**  班级时间不能冲突。  **教师规则：**  特定教师负责特定班级的特定课程；教师时间不能冲突。 |

4 分工

|  |  |  |
| --- | --- | --- |
| 蔡雨坤 | 张凯 | 关荣升 |
| 贡献度：40% | 贡献度：30% | 贡献度：30% |
| 网站搭建、前端功能实现、后端核心算法实现 | 部分后端算法实现、数据库设计 | 系统各功能检验、需求分析、交互优化 |

5 技术选择

在构建一个现代化的、功能复杂的排课系统时，技术选择是项目成功的关键。我们的选择旨在确保系统的性能、稳定性、可扩展性和可维护性。

### 5.1 前端：Vue.js + Element UI

**渐进式框架与上手难度：**Vue.js 被誉为“渐进式框架”，这意味着我们可以根据项目需求，从最核心的视图层渲染，逐步扩展到需要路由（Vue Router） 和状态管理 （Vuex/Pinia） 的复杂单页面应用（SPA）。

**卓越的性能：**Vue.js能够以最小的性能开销精确地更新视图，确保了在处理排课表这种需要频繁数据交互和视图更新的复杂界面时，依然能提供流畅的用户体验。

**Element UI：成熟且全面的组件库：**Element UI是一套专为Vue.js设计的桌面端组件库。它提供了排课系统所需的大量现成组件，如表格（Table）、表单（Form）、对话框（Dialog）、日期选择器（Date Picker） 和时间选择器（Time Picker） 等。使用Element UI可以极大地减少重复的界面开发工作，同时保证了整个系统视觉风格的统一。

### 5.2 后端：Java + Spring Boot + MySQL

**Java：**世界上最流行和成熟的编程语言之一。其庞大的开发者社区和海量的第三方库，为解决各种复杂问题提供了坚实的基础。对于需要处理复杂业务规则和高并发请求的排课系统而言，Java的稳定性和生态系统是毋庸置疑的优势。

**Spring Boot：**构建在强大的 Spring 框架之上，可以快速集成各种常用技术，如数据库、缓存、消息队列等。这使得后端开发团队可以快速启动项目，减少繁琐的配置工作，专注于业务逻辑的实现。

**MySQL：**全球最受欢迎的开源关系型数据库。对于排课系统而言，涉及大量的结构化数据，如学生、教师、课程、教室等，这些数据之间存在着复杂的关联关系。MySQL能够确保数据的完整性和一致性。

### 5.3 排课算法：遗传算法

排课问题是一个典型的**NP-hard**问题，具有大量的约束条件（如教师时间冲突、教室容量限制、学生课程冲突等）和庞大的解空间。传统的穷举搜索方法在问题规模稍大时便会变得不切实际。因此，在网上学习了各个开源项目和算法后，我们选择**遗传算法**这种启发式优化算法来求解。

**全局优化能力：**遗传算法模拟了自然界中生物的进化过程，通过选择、交叉和变异等操作，在复杂的解空间中进行全局搜索。它不容易陷入局部最优解，更有可能找到一个满足所有硬约束并尽可能优化软约束（如课程连贯性）的高质量排课方案。

**强大的约束处理能力：**遗传算法的适应度函数设计非常灵活，可以将排课问题中的各种硬性约束（必须满足）和软性约束（希望满足）量化为适应度评分。例如，每当一个排课方案违反了一个硬性约束（如教师冲突），就给予一个巨大的惩罚值；每满足一个软性约束（如满足教师的授课偏好），就给予一定的奖励值。通过最大化适应度函数，算法能够自动地朝着满足更多约束、质量更高的方向进化。

6 详细设计

本系统采用前后端分离的**B/S**架构。

**前端**：基于**Vue.js**构建单页面应用。用户通过浏览器访问，前端负责视图渲染、用户交互和向后端发送 API 请求。

**后端**：基于**Spring Boot**构建API服务。后端负责处理业务逻辑、数据持久化以及执行核心的排课算法。

**数据库**：使用**MySQL**数据库进行数据存储。

**通信**：前后端之间通过**HTTP/HTTPS**协议进行通信，数据交换格式为**JSON**。

### 6.1 前端模块设计

**登录/权限模块：**处理用户登录、注销，并根据用户角色（管理员、教师、学生）控制菜单和按钮的可见性。

**基础数据管理模块：**提供对教师、学生、课程、教室等基础信息的增删改查（CRUD） 界面。使用Element UI的el-table和el-form组件。

**排课任务管理模块：**

**创建排课任务：**允许管理员选择学期、需要排课的年级等信息，并设置排课的约束条件（如教师不可用时间、课程优先级等）。

**任务状态监控：**实时显示遗传算法的运行状态，如迭代次数、当前最优适应度分数等。

**课表展示与调整模块：**

**多维度课表视图：**以表格形式清晰地展示按班级、按教师、按教室的课表。

**可视化调整：**允许管理员在生成的课表基础上进行手动的拖拽式微调，以处理特殊情况。每次调整后，系统会实时检查是否产生新的冲突。

### 6.2 后端模块设计

**API接口层**：为前端提供清晰、无状态的 API 接口。例如：

GET /api/teachers: 获取所有教师列表

POST /api/courses: 新增一门课程

POST /api/scheduling-tasks: 创建一个新的排课任务

GET /api/schedules/class/{class\_id}: 获取指定班级的课表

**业务逻辑层：**

基础数据服务：处理对教师、课程等数据的业务逻辑。

排课服务（Scheduling Service）：核心服务。接收到排课请求后，准备排课所需的数据，调用遗传算法模块，并将最终结果存入数据库。

**数据访问层：**使用MyBatis持久化框架，负责与MySQL 数据库进行交互。

**遗传算法模块：**

**编码：**将一个完整的排课方案编码为一个“染色体”。染色体是一个一维数组，数组的每个基因代表一门需要安排的课程，基因的值则包含了该课程的授课教师、教室和时间段的ID。

**适应度函数：**算法的核心。

**初始化：**适应度分数为一个较高的初始值。

**硬约束惩罚：**遍历染色体代表的排课方案，检查所有硬约束。每发现一个冲突（如同一时间同一教师有两门课、教室容量不足、学生课程冲突等），就从适应度分数中扣除一个巨大的惩罚值。

**软约束奖励/惩罚：**检查软约束的满足情况。例如，满足教师的授课时间偏好则加分；将同一门课安排得过于分散则扣分。

**选择：**采用轮盘赌选择或锦标赛选择等策略，优先选择适应度分数高的个体（排课方案）进入下一代。

**交叉：**随机选择两个父代个体，通过单点交叉或多点交叉等方式，交换它们的部分基因，从而产生新的子代个体。这模拟了基因重组，有助于探索新的解空间。

**变异：**对新生成的子代个体，以一个较小的概率随机改变其某个或某些基因的值（例如，随机改变一门课的上课时间）。这有助于跳出局部最优解，增加解的多样性。

**算法流程：**

初始化种群：随机生成一批初始的排课方案（染色体）。

评估适应度：计算种群中每个个体的适应度分数。

循环迭代：

a. 选择：根据适应度选择优良个体。

b. 交叉：对选中的个体进行交叉操作，产生后代。

c. 变异：对后代进行变异操作。

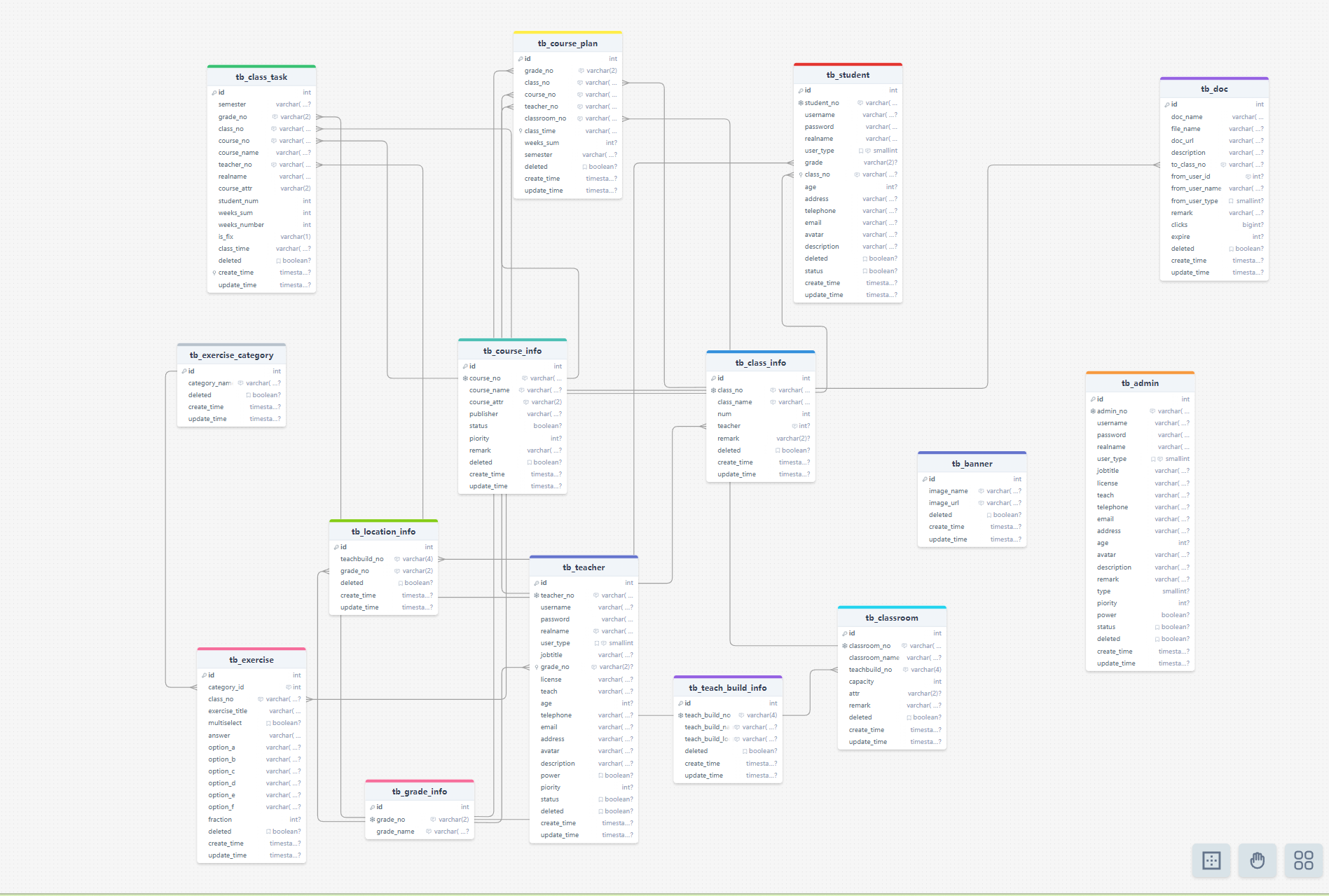
d. 评估：计算新一代个体的适应度。

e. 替换：用新一代替换旧一代种群。

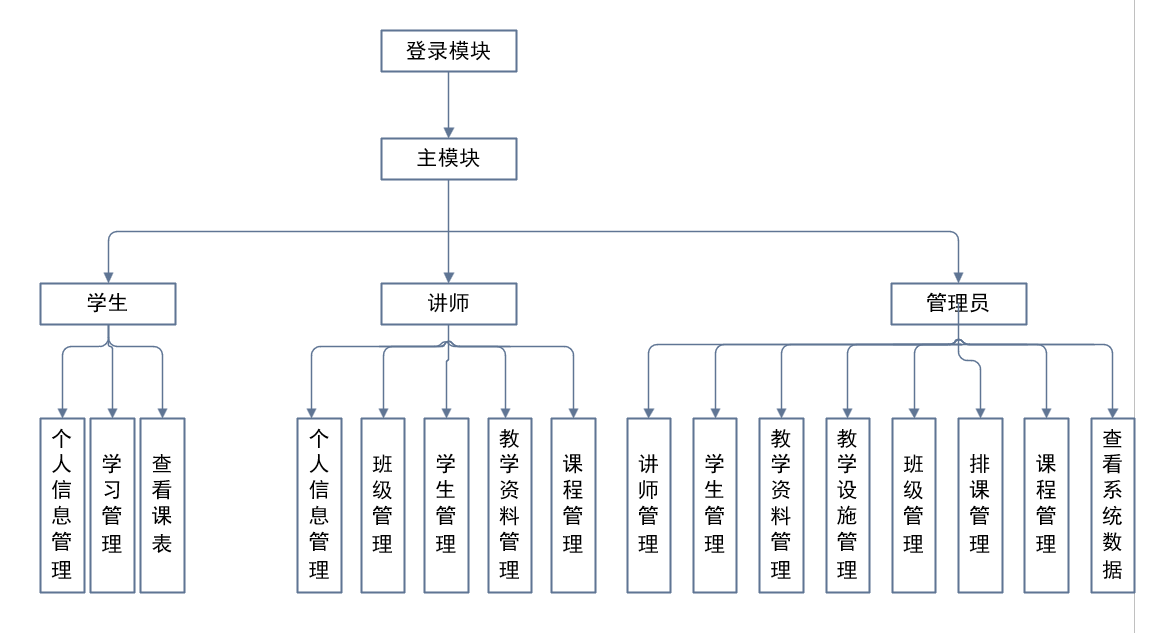
终止：当达到预设的最大迭代次数，或适应度分数连续多代没有显著提升时，算法终止。将此时适应度最高的个体作为最终的排课方案输出。

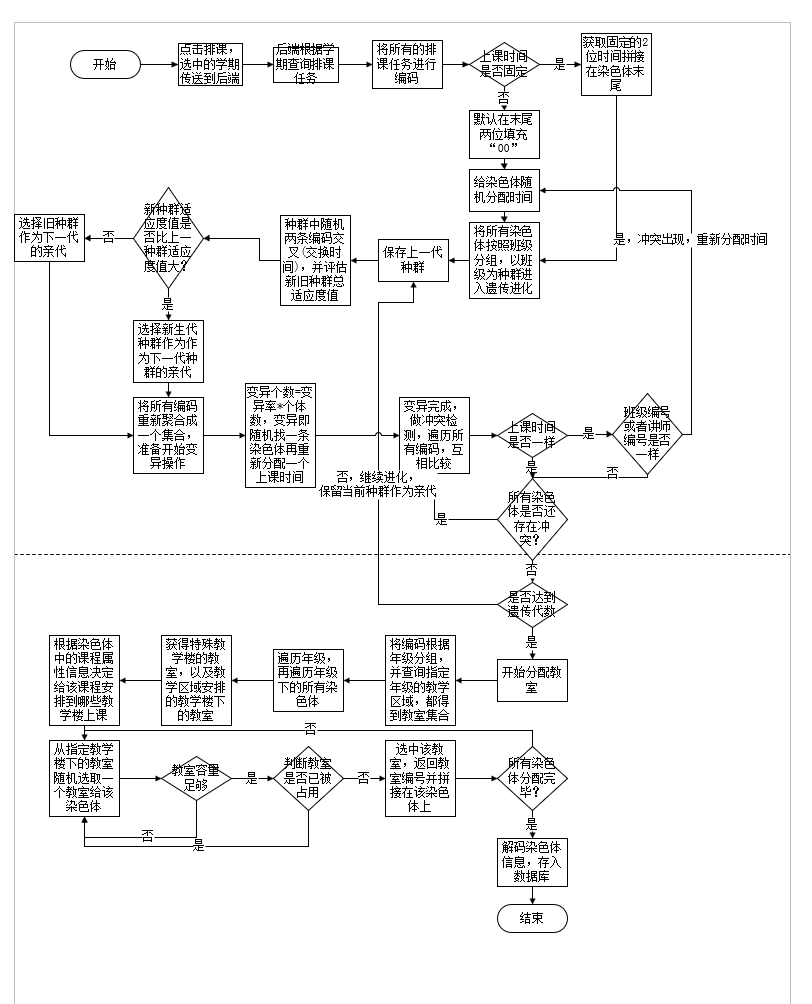
7 系统建模图

### 7.1 ER图

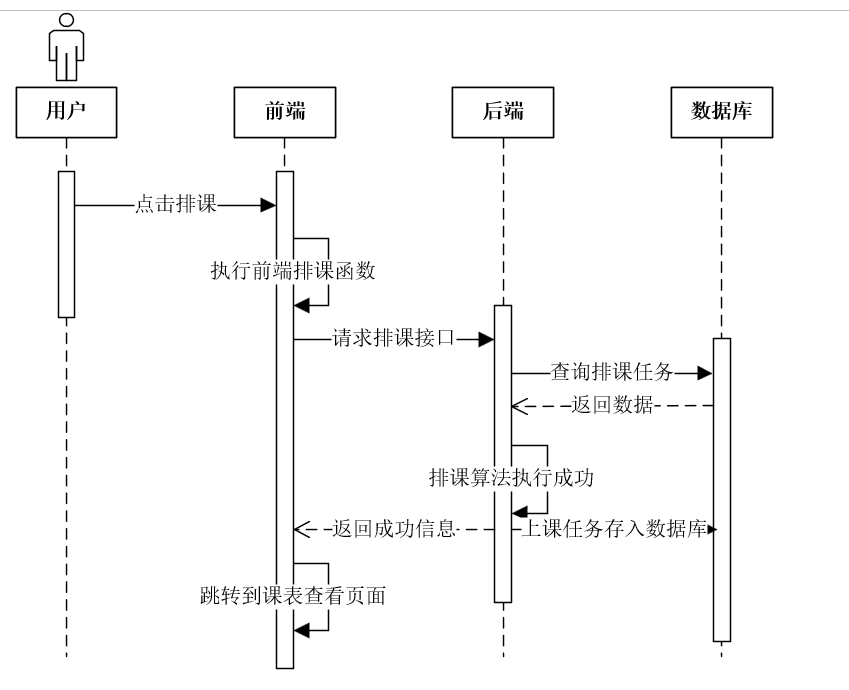


### 7.2 系统结构图



7.3 排课规则流程图  


### 7.4 排课时序图



8 功能演示  
 排课系统展示。



图 1 主界面



图 2 管理员/教师登录界面



图 3 学生登录界面

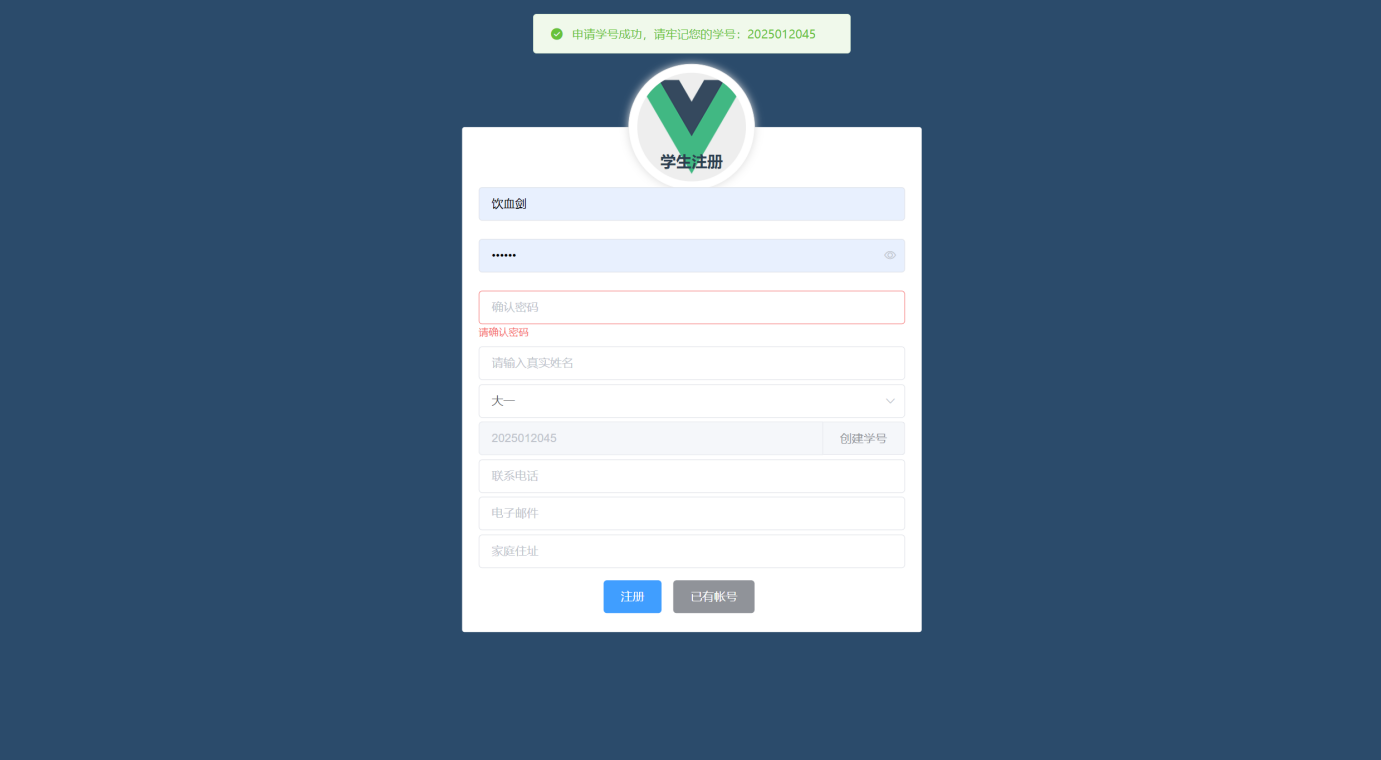


图 4 注册界面

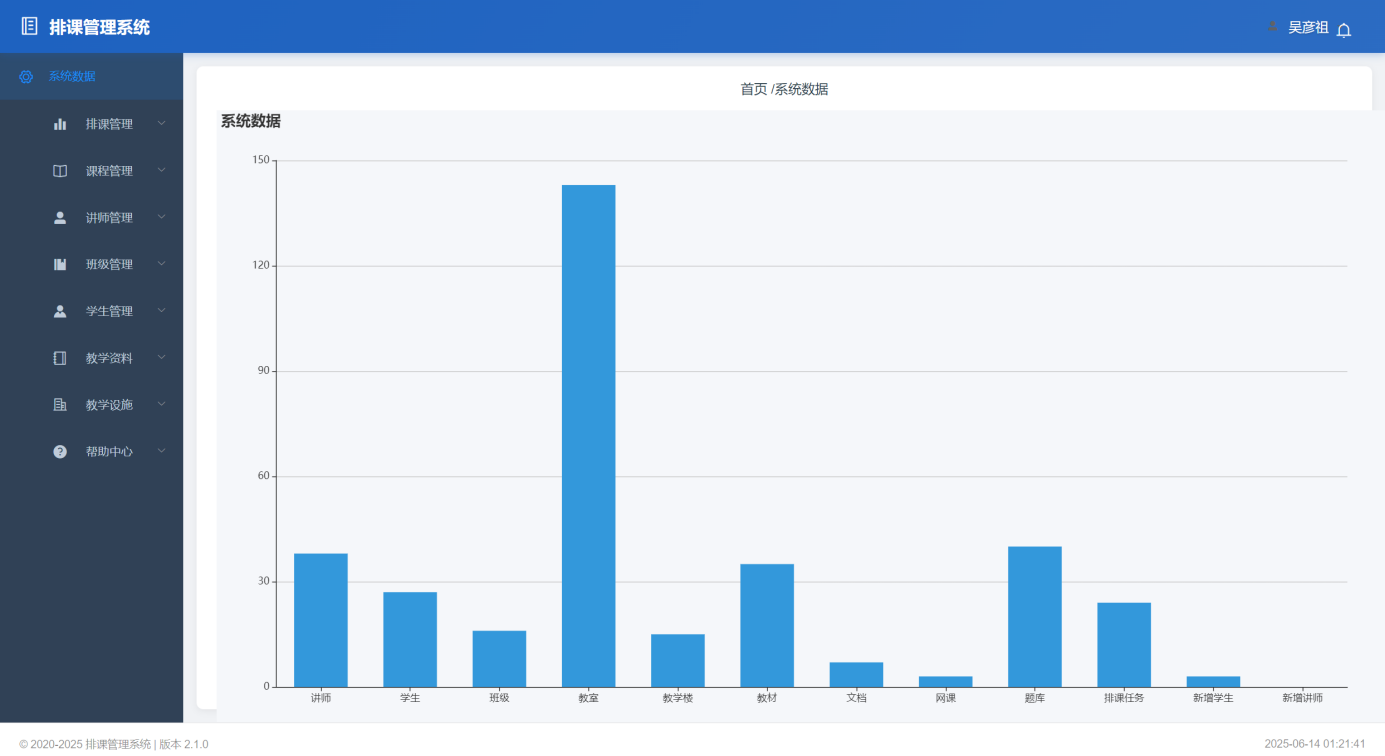


图 5 功能主界面