**类的简单设置**

class ClassRoom {

string name;

int size;

int type; // 比如 TYPE\_MEDIA, TYPE\_COMPUTER 什么的，可以用 | 连接同时满足多个 type

};

class Restriction {

string teacherName;

enum CourseTime courseTime;

int priority; // 越高表示越优先考虑该教师的特殊限制

};

// Course 结构由第一组给出

class ScheduledCourse {

Course course;

enum CourseTime courseTime;

};

class ToScheduleData {

void addCourse(Course course);

void addClassRoom(ClassRoom classroom);

void addRestriction(Restriction restriction);

vector<Course> getCourses();

vector<ClassRoom> getClassrooms();

vector<Restriction> getRestrictions();

};

class ScheduledData {

void addScheduledCourse(ScheduledCourse course);

vector<ScheduledCourse> getScheduledCourse();

}

// 定义一个接口，可以用多种算法实现多个类，都能使用，假装灵活度很高的样子...

interface SchedulingPolicy {

ScheduledData schedule(ToScheduleData data);

};

**任务概述**

**1. 目标**

实现自动排课系统的相关业务逻辑并开发用户界面，包括录入教室信息，自动排课，手动调课，导出课表等。

**2. 用户特点**

自动排课系统的用户有两类。

1. 系统管理人员：主要进行教室信息录入，自动排课，手动调课等绝大部分操作。

2. 各个课程的教师：主要进行课表的读取操作，并提交一定量的特殊需求给系统管理人员。

**3. 需求概述**

考虑到大部分用户均非计算机专业人员，该系统应该界面直观、简单且易操作。

对于系统管理人员主要使用的功能：

1. 教室信息录入功能的使用频率在学期初始时较高，但同时操作的人数较少，需要着眼于数据库读写速度的优化，而允许在并发量上不作太大关注。

2. 自动排课功能的使用频率较低但较为重要，所以自动排课功能运行可以消耗一定的时间与资源，但是结果应尽量达到较高的要求。

对于课程教师主要使用的功能：

导出课表操作使用频率较高，涉及较多读取，系统需要在数据读取上进行一定优化。

**4. 假定和约束**

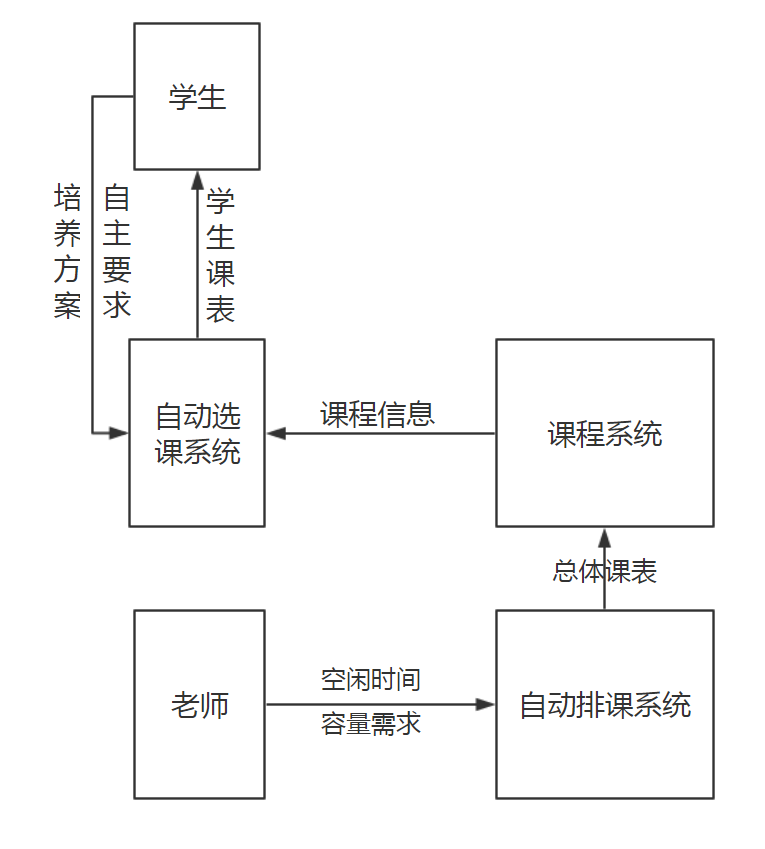
本次自动排课系统的开发出于教学目的，故资源有限，所以开发小组需充分利用其现有资源。

由于自动排课功能是子系统的核心功能，需要预留较多时间进行调查研究与开发，并进行相当的性能与正确性测试。

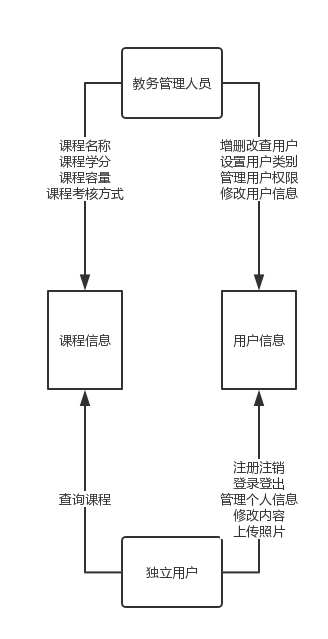
由于整个大系统开发所允许的时间较短，因此子系统的开发必须预留最后集成的时间，且该时间需要掌握得当。

**数据流动图**

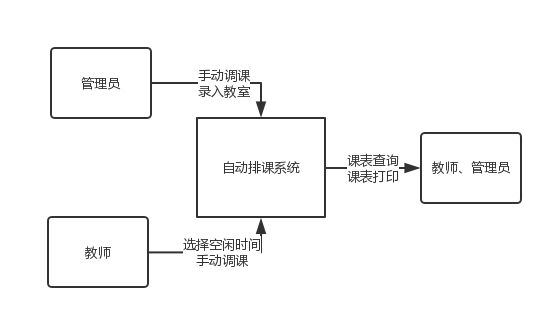
**课程系统顶层**



**基础信息管理**



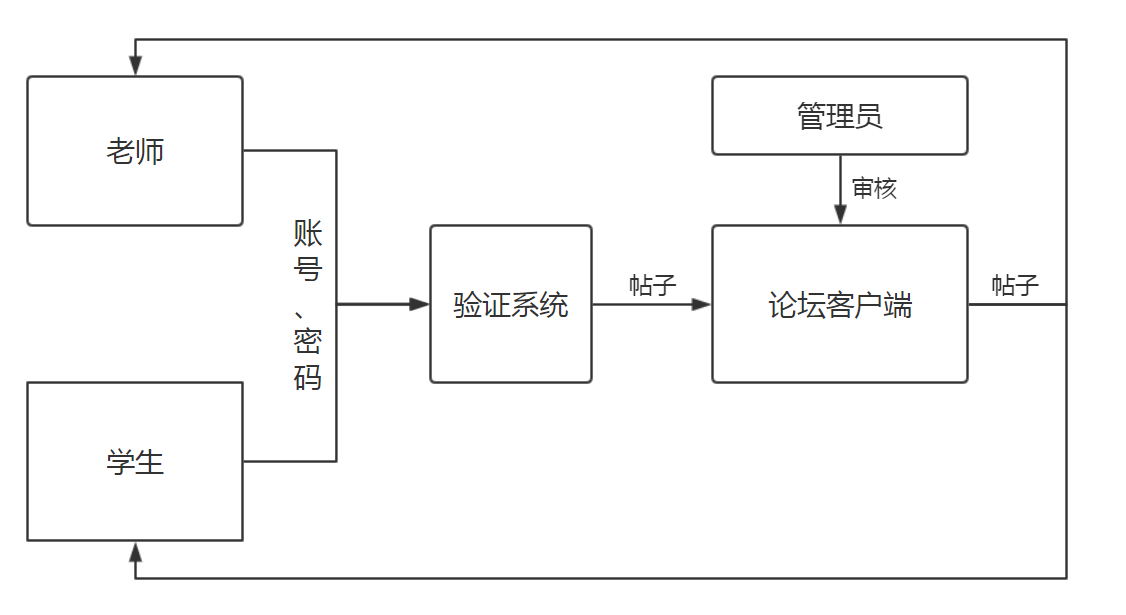
**自动排课**



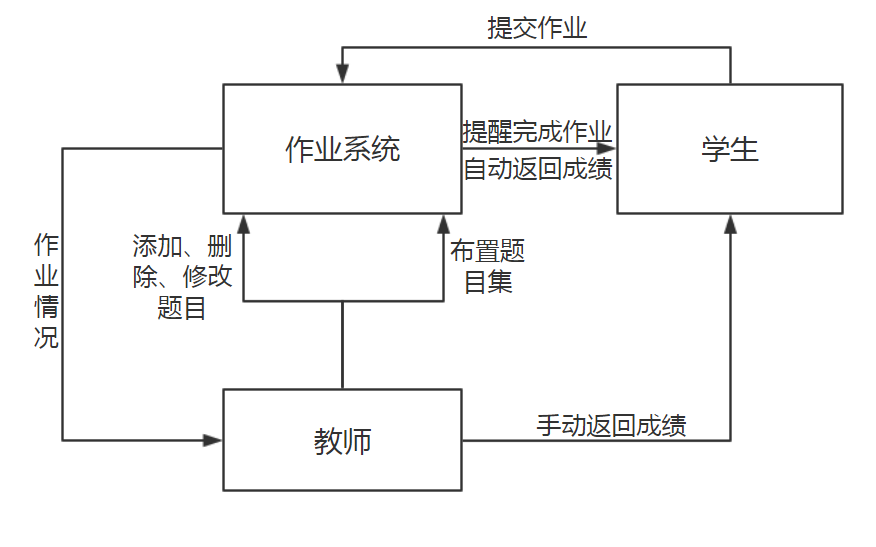
**选课**



**论坛**



**作业系统顶层**



**验收标准**

**数据库测试**

1. 测试计划：验证教室信息的数据库表设计符合需求

测试步骤：

* 1. 在MYSQL选择项目数据库。
  2. 查看classroom表结构并分析。
  3. 预期结果：classroom数据表中具备了需求设计中涉及的所有必要信息并且roomid,roomname属性的限制符合逻辑。

1. 测试计划：验证排课信息的数据库表设计符合需求。

测试步骤：

* 1. 在MYSQL选择项目数据库。
  2. 查看course\_arrange表结构并分析。
  3. 预期结果：course\_arrange数据表中具备了需求设计中涉及的所有必要信息并且courseid,roomid,teacherid属性的限制符合逻辑。

**自动排课模块：**

1. 测试计划：验证教师自动排课的功能

测试步骤：

1. 模拟多名教师登录排课系统，分别选择各自教授的课程所需教室大小、各自的空闲时间等选项
2. 启动自动排课系统
3. 预期结果为：排出合理的课表满足所有（或尽可能多）的需求，并且写入数据库，使得学生和老师都可以依此选课。
4. 若无法正确排出课表，需检查自动排课模块。若无法正确写入，则检查不同模块的耦合。
5. 模拟一名教师登录排课系统，修改其需求
6. 预期结果为：系统根据修改后需求再次排出合理的课表
7. 若无法正确排出课表，需检查业务逻辑。若无法正确写入，则检查不同模块的耦合。

**性能测试**

1. 测试计划：验证低数据量下后端接受请求道返回数据的响应时间。

测试步骤：

* 1. 在数据库中放入500个教师信息，1000个排好的课程信息。
  2. 由一名教师登录后，进行课程录入、空闲时间选择、排课、手动调课、打印课表等操作
  3. 预期结果：统计后端接收到请求道返回数据的响应时间，确保用户在课程录入、空闲时间选择、打印课表等功能在0.5秒内收到返回的数据信息，保证排课、手动调课功能在5秒内收到返回信息。

1. 测试计划：验证高数据量下后端接受请求道返回数据的响应时间。

测试步骤：

1. 在数据库中放入5000个教师信息，10000个排好的课程信息。
2. 由一名教师登录后，进行课程录入、空闲时间选择、排课、手动调课、打印课表等操作
3. 预期结果：统计后端接收到请求道返回数据的响应时间，确保用户在课程录入、空闲时间选择、打印课表等功能在1秒内收到返回的数据信息，保证排课、手动调课功能在10秒内收到返回信息。

**负载测试**

1. 测试计划：验证网站能同时为少量 (5名)用户在线提供自动排课服务

测试步骤：

* 1. 模拟5名教师在线并发进行论坛操作，包括课程录入、空闲时间选择、排课、手动调课、打印课表等。
  2. 若服务器能正常运作，事务操作均能正常执行，并能为同时进行服务器操作的用户提供较快的加载速度，则说明在少量用户并行下功能可以完成，基本功能的设计没有问题。
  3. 若在少量用户并行下出现问题，则说明基本的并行逻辑设计有误，需要检查自动排课基本的逻辑设计。

1. 测试计划：验证网站能同时为中量 (50名)用户在线提供自动排课服务

测试步骤：

* 1. 模拟50名教师在线并发进行论坛操作，包括课程录入、空闲时间选择、排课、手动调课、打印课表等。
  2. 若服务器能正常运作，事务操作均能正常执行，并没有明显卡顿，则说明服务器对中量数据下的并行基本完成。并行逻辑与负载功能基本完成。
  3. 若在中量用户并行下出现问题，则说明并行的逻辑和优化不足，需要进一步优化设计结构。

1. 测试计划：验证网站能同时为大量 (200名)用户在线提供自动排课服务

测试步骤：

* 1. 模拟200名教师在线并发进行论坛操作，包括课程录入、空闲时间选择、排课、手动调课、打印课表等。
  2. 若服务器能正常运作，事务操作均能正常执行，并没有明显卡顿，则说明服务器对大量数据下的并行完成。结构设计及优化都已完成。
  3. 若在大量用户并行下出现问题，则说明并行的逻辑和优化不足，需要进一步优化设计结构。也应当考虑服务器的硬件带宽性能等因素。