南京邮电大学

专业课程设计 II 报告

（ 2018 / 2019 学年 第 一 学期）

题　　目：选修课程管理系统

专　　业	计算机科学与技术
学 生 姓 名	潘一帆
班 级 学 号	B15021030
指 导 教 师	张琳
指 导 单 位	计算机科学与技术系
日　　期	**2018.12.24-2019.1.4**

支撑指标点	评价准则	计分（每项 10 分）
2.2 能够通过文献研究表达复杂工程问题（40 分）	1、能够掌握计算机软件设计的相关基础知识，并能够针对求解的工程问题，进行合理的分析与设计	
	2、具备一定自学能力与探索创新意识，能够充分利用教科书及其资源（如网络等）自学新知识与新技能	
	3、能够给出数据结构和算法的设计描述，给出关键算法的流程图或伪代码，并给出各算法之间的结构关系描述	
	4、具备一定的人机交互设计意识，人机交互设计合理、友好，操作简便	
3.3 能综合利用专业知识对解决方案进行优化，体现创新意识，并考虑健康、安全以及环境等因素（40 分）	5、具备一定的算法与数据结构设计分析能力，能够完成课题要求的各项任务和指标	
	6、能够结合计算机软硬件资源，合理选用算法、数据结构、数据存储方式等技术手段，对求解的工程问题进行有效建模和求解	
	7、能够选择合适的程序设计语言与编程开发平台，对求解的工程问题进行编程实现	
	8、掌握调试方法与工具，对程序开发过程中出现的问题进行分析、跟踪与调试，并能够进行充分测试	
11.2 能够在多学科环境中应用工程管理原理和经济决策方法进行工程设计与实践，具有一定的组织、管理能力（20 分）	9、能够正确、完整地回答指导教师关于课题的问询，反映其对课题内容，以及相关的项目管理知识具有较好的理解和掌握	
	10、具备一定的语言表达能力与文字处理能力，能够结合复杂工程问题撰写报告，报告内容和实验数据详实，格式规范	
专业课程设计 II 能力测评总分		
指导教师：_____ _____年____月____日		
备注：		

选修课程管理系统

一、课题内容和要求

设计选修课程管理系统，主要功能如下：

（1）区分学生、教师、管理员等用户角色。

（2）对学生个人信息（学号，班级，学院等)、选修课程信息（课号，课程名称，开课学院，开放的年级，授课教师，等）及学生选课信息进行维护。功能包括：录入，修改，查询，删除等。

（3）按照平时成绩与期末考试成绩各占总评成绩的 30%和 70%的比例，录入并计算总评成绩。

（4）针对具体选修课程，算出处于优、良、中、及格、不及格的学生人数以及占总人数的百分比。其中 100-90 为优，89-80 为良，79-70 为中，69-60 为及格，60 分以下为不及格，建议用图表的形式显示。

（5）按要求分别输出成绩在优、良、中、及格、不及格各区间的学生学号。

二、需求分析和总体设计

2.1 本课题的主要功能包括：

（1）登录功能，不同角色（学生、教师和管理员）登录进入不同的页面。

（2）学生功能模块，具有选修课程、退选课程、查看已选课程、查看课程成绩、查看看个人信息和更改密码功能。

（3）教师功能模块，具有录入成绩、查看和修改成绩、统计各等第占比、统计各班级均分、查看看个人信息和更改密码功能。

（4）管理员功能模块，具有编辑（增加、删除、查询、修改）学生信息、编辑教师信息、编辑课程信息、查看看个人信息和更改密码功能。

用例图如图 1 所示：

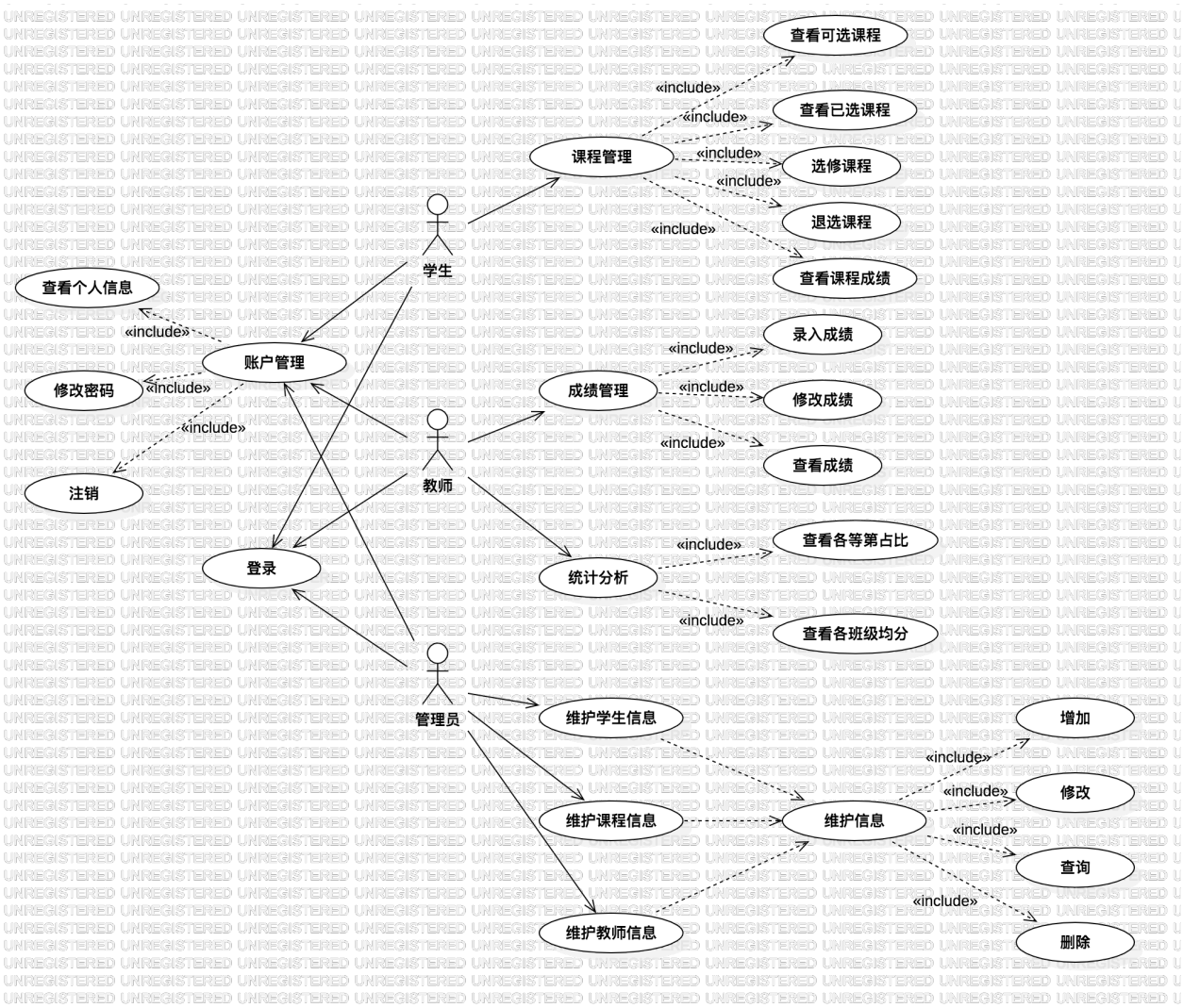

图 1 系统用例图

2.2 本课题的数据表单设计

关系模式集为：

学生（学号、姓名、班级、学院、年龄、性别、密码）

课程（课程号、课程名、开课学院、开放年级、教师号）

选课（ 学号、课程号 、平时成绩、考试成绩、总评成绩）

教师（工号、姓名、密码）

管理员（工号、姓名、密码）

其中，加下划线的字段表示该表中的主键，加虚线的字段表示该表中的外键，引用的主键名称与外键名称一致。关系模式集对应的 ER 图如图 2 所示：

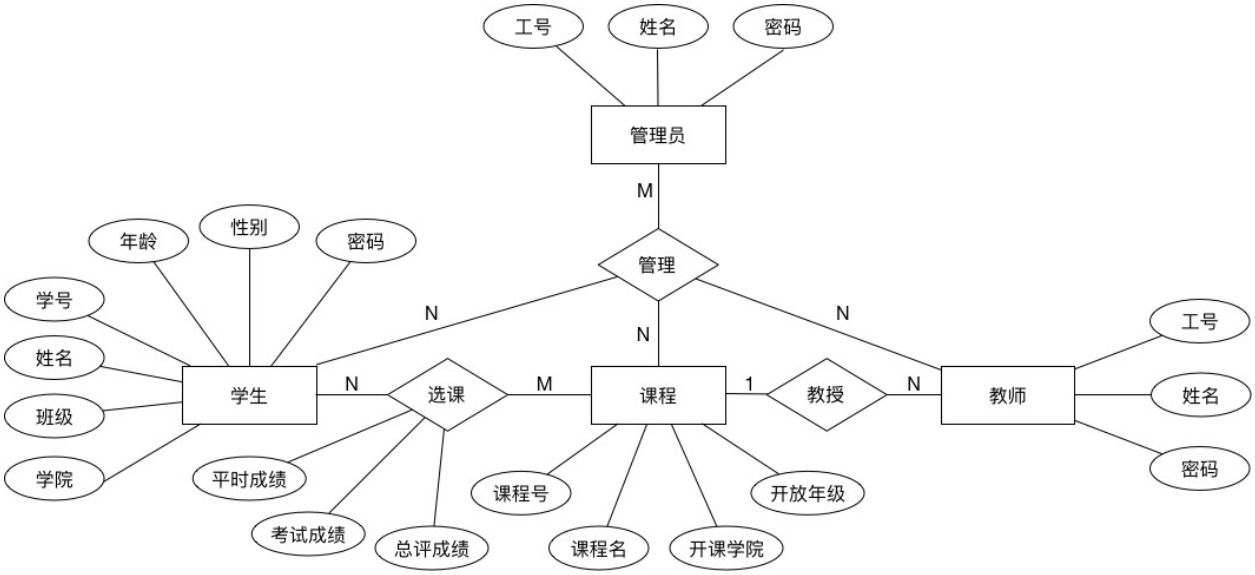

图 2 数据库设计 ER 图

2.3 本课题的体系结构设计

本系统形式为 Web 应用，采用 MVC 架构，分为模型层（model）、视图层（view）和控制层（controller），模型层主要负责处理数据库操作，为控制层提供数据存取的方法。控制层主要负责处理业务逻辑，将数据库中的原始数据加工处理，使其符合用户的需求，然后发送给视图层。视图层的作用就是处理用户交互，将用户的输入传递给控制层，接收控制层返回的处理结果并将其呈现给用户。

本系统数据库采用 MySQL，模型层和控制层采用 java 语言并结合 servlet 技术实现，这两层可以称之为“后端”，视图层采用 html 网页的形式，利用 vue 框架实现，这一层可以称之为“前端”。前后端之间是完全分离的，相互间的通信通过 http 协议传递 json 格式数据来完成。本系统的架构如图 3 所示：

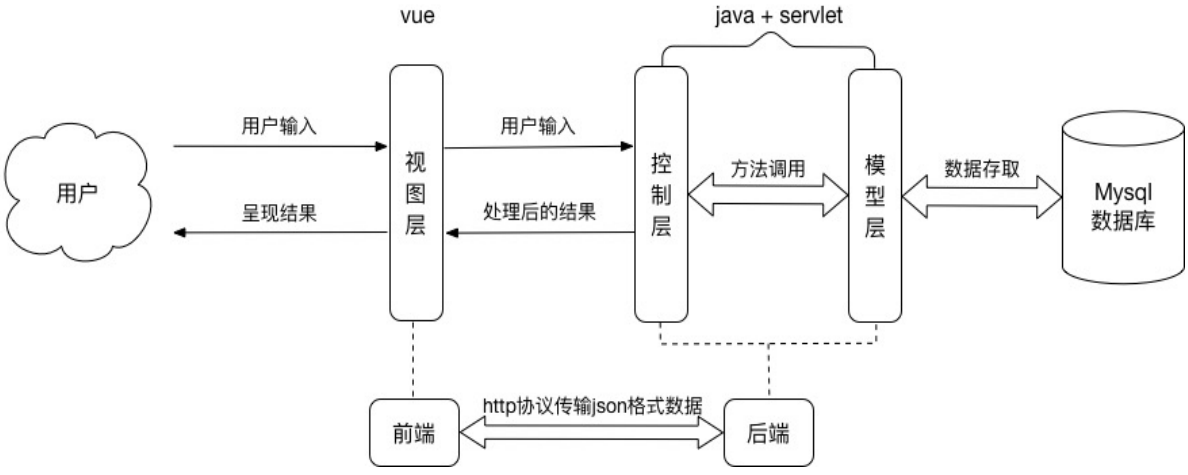

图 3 系统架构图

2.4 本课题的主要功能界面设计

登录界面如图 4 所示：

图 4 登录界面 UI 设计图

学生、教师和管理员的基本界面如图 5 所示：

图 5 角色基本界面 UI 设计图

三、相关功能模块详细设计

1 模块内的数据结构设计

本系统中用到的的数据结构主要是对象数组，首先定义了三个对象类：学生类

（Student）、教师类（Teacher）、课程类（Course），具体定义如图 6 所示：

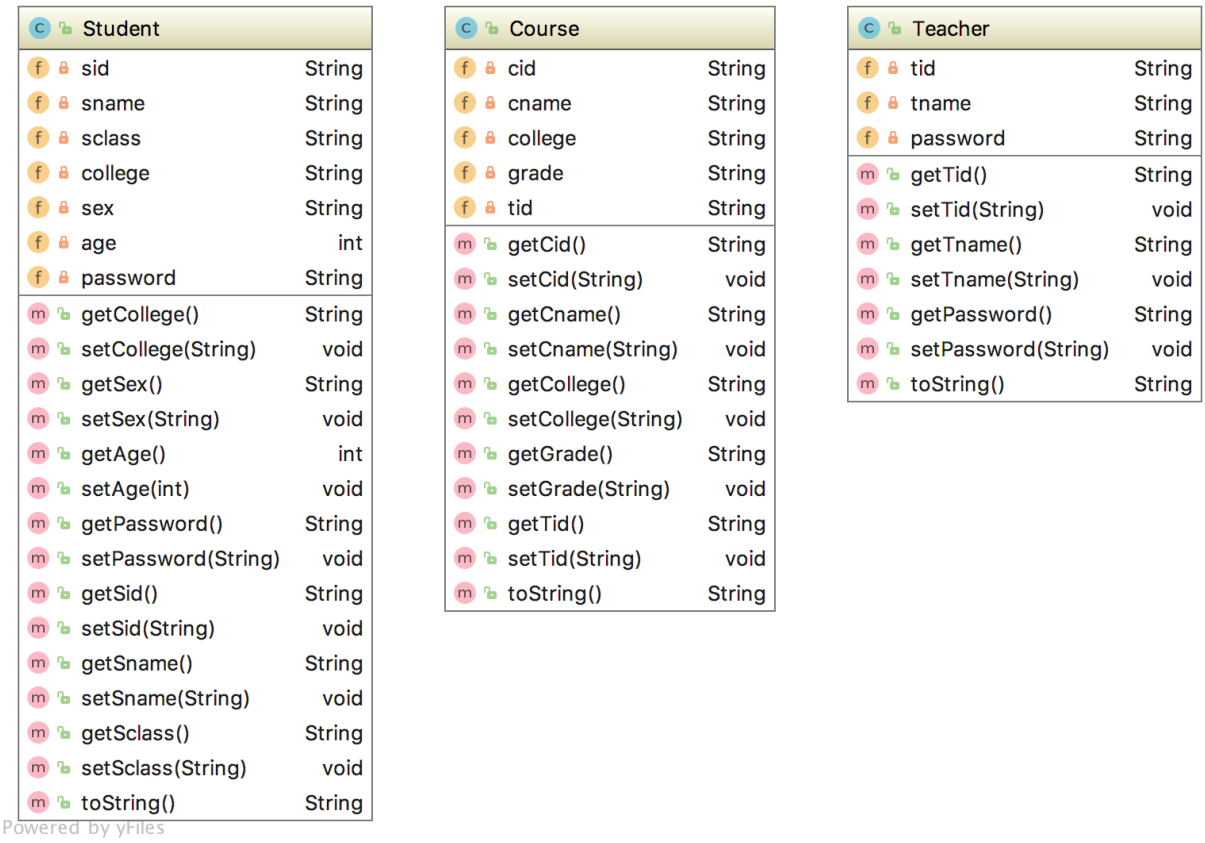

图 6 系统类图

对应的对象数组如表 1 所示：

表 1 数据结构表

定义	名称	功能
List<Student> students	学生对象数组	存储多个学生信息
List<Teacher> teachers	教师对象数组	存储多个教师信息
List<Course> courses	课程对象数组	存储多个课程信息

2 模块程序流程图设计

2.1 登录模块和账号管理模块

其中账户管理模块是学生模块、教师模块和管理员模块的子模块，因为和登录模块逻辑类似，就放在一起描述。

登录功能流程图和账户管理流程图分别如图 7、图 8 所示：

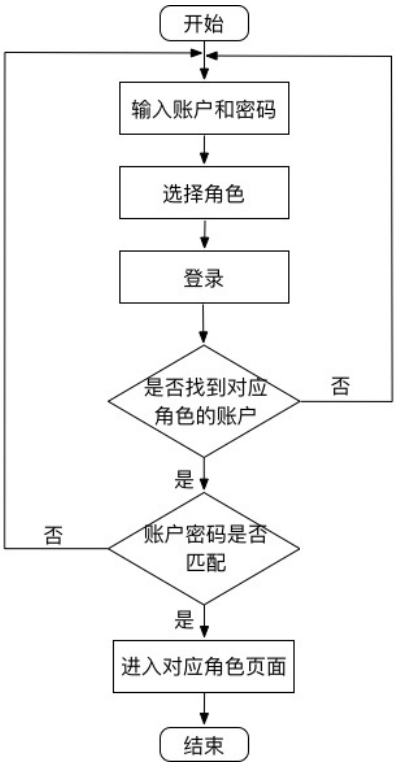

图 7 登录流程图

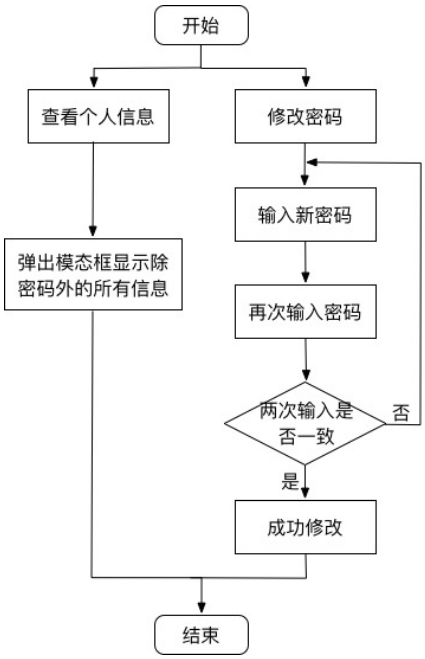

图 8 账户管理流程图

2.2 学生模块

学生模块流程图如图 9 所示：

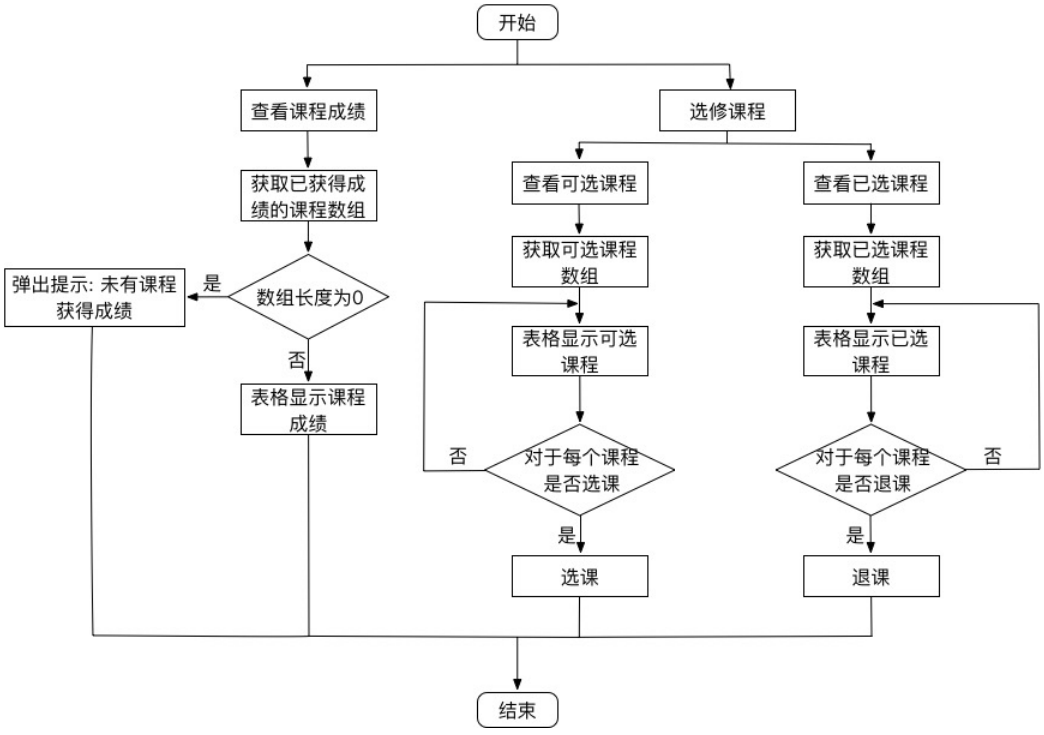

图 9 学生模块流程图

2.3 教师模块

教师模块流程图如图 10 所示：

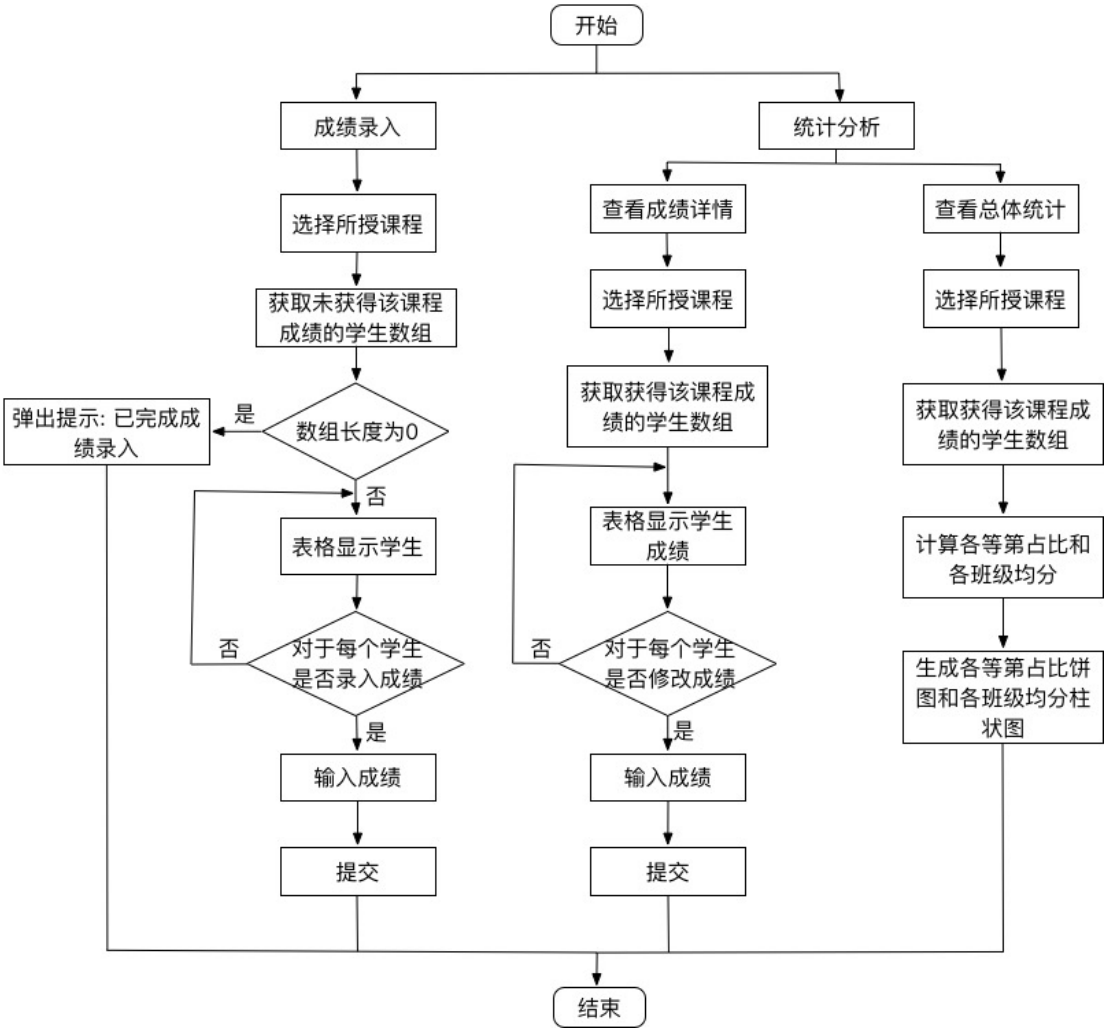

图 10 教师模块流程图

2.4 管理员模块

管理员模块主要是对学生、教师和课程信息的增加、修改、删除和查询，逻辑简单清晰，为节约篇幅本部分不再绘制流程图。

四、部分核心代码

1 登录模块

核心函数：validate，用于验证账户密码是否匹配。首先根据角色查找账号，若没有找到，直接返回 false 表示验证失败，找到后再进行密码匹配。

```
public boolean validate(String role, String id, String password){
    if(role.equals("student")){
        Student stu = sd.findStudent(id);
        if(stu == null)
            return false;
```

```
            if(password.equals(stu.getPassword()))
                return true;
            else
                return false;
        }else if(role.equals("teacher")){
            Teacher tch = td.findTeacher(id);
            if(tch == null)
                return false;
            if(password.equals(tch.getPassword()))
                return true;
            else
                return false;
        }else{
            Admin adm = ad.findAdmin(id);
            if(adm == null)
                return false;
            if(password.equals(adm.getPassword()))
                return true;
            else
                return false;
        }
}
```

2 学生模块

2.1 查看可选课程

模型层核心函数：

```
public List<Course> getOptionalCourse(Student stu) {
        String sql = "select * from Course where grade=? and cid not in (select cid from SelectCou
rse where sid=?)";
        Object[] params = {stu.getSid().substring(1,3), stu.getSid()};
        List<Course> optionalCourse;
        try {
            optionalCourse = qr.query(sql, new BeanListHandler<Course>(Course.class), params);
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
        return optionalCourse;
}
```

控制层核心函数：

```
public List<Map<String, Object>> getOptionalCourse(Student stu) throws DatabaseOperationException{
        List<Course> courses = null;
        List<Map<String, Object>> res = new ArrayList<Map<String, Object>>();
```

```
        try{
            courses = sd.getOptionalCourse(stu);
            addTnameForCourses(courses, res);
        }catch (Exception e){
            e.printStackTrace();
            throw new DatabaseOperationException("获取课程失败!");
        }
        if(courses.size() == 0)
            return null;
        return res;
}
```

2.2 选修课程

模型层核心函数：

```
public boolean selectCourse(Student stu, Course course) {
        String sql = "insert into SelectCourse values(?,?,null,null,null)";
        Object[] params = {stu.getSid(), course.getCid()};
        try {
            qr.update(sql, params);
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
        return true;
}
```

控制层核心函数：

```
public void selectCourse(Student stu, Course course) throws DatabaseOperationException{
        if(!sd.selectCourse(stu, course)){
            throw new DatabaseOperationException("选课失败!");
        }
}
```

查看已选课程和退选课程实现代码与上述代码相似，不再叙述。

3 教师模块

3.1 查看所授课程

模型层核心函数：

```
public List<Course> getTeachedCourse(Teacher tch) {
        String sql = "select * from Course where tid=?";
        Object[] params = {tch.getTid()};
        List<Course> teachedCourse;
        try {
            teachedCourse = qr.query(sql, new BeanListHandler<Course>(Course.class), params);
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
        return teachedCourse;
}
```

控制层核心函数：

```
public List<Map<String,Object>> getTeachedCourse(Teacher tch) throws DatabaseOperationException{
        List<Course> courses = null;
        List<Map<String,Object>> res = new ArrayList<Map<String,Object>>();
        try{
            courses = td.getTeachedCourse(tch);
            for(Course course : courses){
                Map<String,Object> map = BeanUtils.Bean2Map(course);
                int totalStudentNum = cd.getStudentTotalNum(course);
                map.put("totalStudentNum",totalStudentNum);
                res.add(map);
            }
        }catch(Exception e){
            e.printStackTrace();
            throw new DatabaseOperationException("查询课程失败!");
        }
```

```
        if(res.size() == 0)
            return null;
        return res;
}
```

3.2 查看未获得成绩的学生

模型层核心函数：

```
public List<Student> getUnscoredStudent(Course course) {
        String sql = "select * from Student where sid in (select sid from SelectCourse where cid=? and finalScore is null)";
        Object[] params = {course.getCid()};
        List<Student> unscoredStudent;
        try {
            unscoredStudent = qr.query(sql, new BeanListHandler<Student>(Student.class), params);
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
        return unscoredStudent;
}
```

控制层核心函数：

```
public List<Map<String,Object>> getUnscoredStudent(Course course) throws DatabaseOperationException{
        List<Student> unscoredStudent = null;
        List<Map<String, Object>> res = new ArrayList<Map<String, Object>>();
        try{
            unscoredStudent = cd.getUnscoredStudent(course);
            deletePasswordForStudent(unscoredStudent, res);
        }catch (Exception e){
            e.printStackTrace();
            throw new DatabaseOperationException("获取学生失败!");
        }
```

```
        if(unscoredStudent.size() == 0)
            return null;
        return res;
}
```

3.2 录入成绩

模型层核心函数：

```
public boolean scoreInput(Student stu, Course course, Score score) {
        String sql = "update SelectCourse SET regularScore=?,examScore=?,finalScore=? where sid=? and cid=?";
        Object[] params = {score.getRegularScore(), score.getExamScore(), score.getFinalScore(), stu.getSid(), course.getCid()};
        try {
            qr.update(sql, params);
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
        return true;
}
```

控制层核心函数：

```
public void scoreInput(Student stu, Course course, Score score) throws DatabaseOperationException{
        if(!cd.scoreInput(stu,course,score)){
            throw new DatabaseOperationException("录入失败!");
        }
}
```

其他功能代码结构与上述代码相似，故不再叙述。

4 管理员模块

管理员模块主要是对学生、教师和课程信息的维护（增加、修改、删除、查询），这里以维护学生信息为例。

模型层有四个函数，分别对应增加、修改、删除、查询功能：

```
public boolean addStudent(Student stu){   //增加
        String sql = "insert into Student values(?,?,?,?,?,?,?)";
        Object[] params = {stu.getSid(), stu.getSname(), stu.getSclass(), stu.getCollege(), stu.getAge(),
stu.getSex(),stu.getPassword()};
        try {
            qr.update(sql, params);
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
        return true;
}

public boolean delStudent(String stuId){ //删除
    String sql = "delete from Student where sid=?";
    Object[] params = {stuId};
    try {
        qr.update(sql, params);
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    }
    return true;
}

public Student findStudent(String stuId){ //查询
    String sql = "select * from Student where sid=?";
    Object[] params = {stuId};
    Student stu;
    try {
        stu = qr.query(sql, new BeanHandler<Student>(Student.class), params);
    } catch (SQLException e) {
        e.printStackTrace();
        return null;
    }
    return stu;
}

public boolean updateStudent(Student stu){ //修改
    String sql = "update Student SET sname=?,sclass=?,college=?,age=?,sex=? where sid=?";
    Object[] params = {stu.getSname(), stu.getSclass(), stu.getCollege(), stu.getAge(),
stu.getSex(),stu.getSid()};
    try {
        qr.update(sql, params);
```

```
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
        return true;
    }
```

控制层也有四个函数，分别对应模型层的四个函数：

```
    public void addStudent(Student stu) throws DatabaseOperationException{//增加
        if(!sd.addStudent(stu)){
            throw new DatabaseOperationException("添加学生失败!");
        }
    }

    public void delStudent(String id) throws DatabaseOperationException{//删除
        if(!sd.delStudent(id)){
            throw new DatabaseOperationException("删除学生失败!");
        }
    }

    public void updateStudent(Student stu) throws DatabaseOperationException{//修改
        if(!sd.updateStudent(stu)){
            throw new DatabaseOperationException("更新学生信息失败!");
        }
    }

    public Student findStudent(String id) throws DatabaseOperationException{ //查询
        Student stu = null;
        try{
            stu = sd.findStudent(id);
        }catch (Exception e){
            throw new DatabaseOperationException("查询学生信息失败!");
        }
        return stu;
    }
```

五、软件测试及其结果分析

1 登录模块

图 11 登录界面

分别以学生、教师和管理员身份进行登录测试，都能成功跳转到对应界面，若账号或密码错误会弹出提示。

2 学生模块

2.1 选修课程

图 12 课程选修界面

可以正确显示出所有可选的课程，点击旁边的多选框可以选择课程，点击“提交选课”按钮后就选修了对应课程。点击下拉菜单中的“已选课程”可以查看到选修的课程，如图 0 所示：

图 13 已选课程界面

点击旁边的多选框可以退选课程

2.2 成绩查询

图 14 成绩查询界面

2.3 账号设置

点击下拉菜单中的“个人信息”即弹出模态框显示出个人信息：

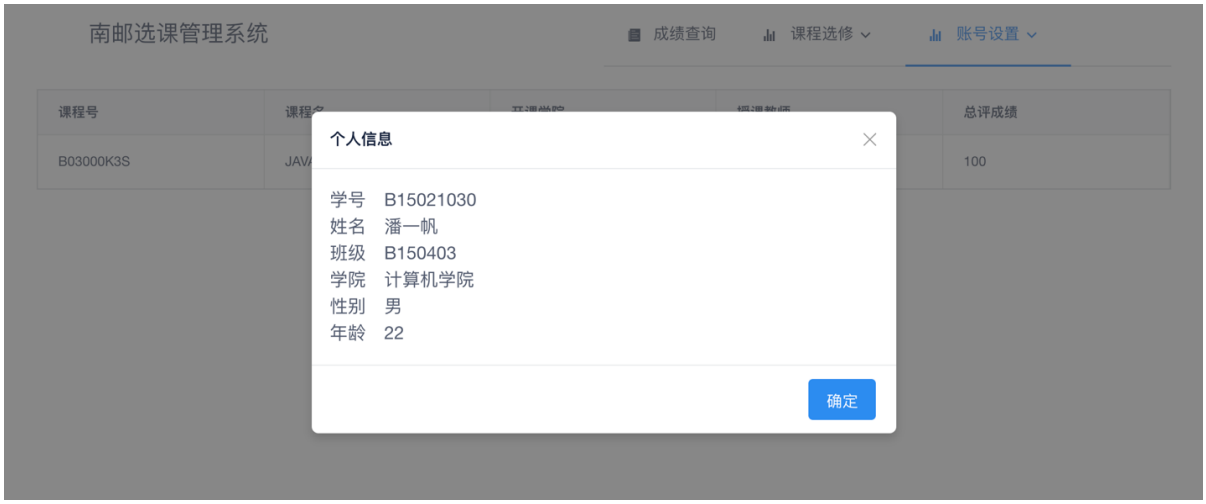

图 15 个人信息界面

点击下拉菜单中的“修改密码”即进入密码修改界面：

图 16 修改密码界面

3 教师模块

3.1 成绩录入

图 17 成绩录入界面

可以选择所授课程，并显示出未录入成绩的学生信息，并给出学生总数和未录入学生数。点击右边的录入按钮会弹出一个模态框用于成绩录入：

图 18 成绩录入

3.2 统计分析

点击下拉菜单中的“成绩详情”可以显示出已获得成绩的学生，并提供成绩修改按钮

图 19 成绩详情界面

点击下拉菜单中的“总体分析”可以显示出所授课程的各等第占比和各班级均分：

图 20 总体分析界面

4 管理员模块

4.1 学生管理

点击信息维护中的“学生信息”即进入学生管理界面，分页显示出所有学生信息：

图 21 学生管理界面

点击“搜索”按钮可以进行模糊搜索：

图 22 搜索界面

点击“添加”按钮弹出模态框用于添加学生：

图 23 添加学生

点击编辑按钮弹出模态框用于编辑学生信息，且默认值为当前学生的信息：

图 24 修改学生信息

4.2 教师管理

点击信息维护中的“教师信息”即进入教师管理界面，分页显示出所有教师信息：

图 25 教师管理界面

其他功能与学生管理相似，且测试通过。

4.3 课程管理

点击信息维护中的“课程信息”即进入课程管理界面，分页显示出所有课程信息：

图 26 课程管理界面

其他功能与学生管理相似，且测试通过。

六、课题完成过程中遇到的问题及解决方法

问题 1：数据库中文乱码，只要存储的信息是中文就无法正常显示。

解决方法：修改 MySQL 数据库配置文件，将默认编码改为 utf-8。

问题 2：请求的 url 参数如果是中文，那么会造成后端无法识别，导致前后端通信异常。

解决方法：url 参数编码方式为 iso-8859-1，后端接收到后先以 iso-8859-1 方式解码成二进制字节流，再以 utf-8 方式编码成能识别的字符串。

问题 3：运行时报 java 空指针错误。

解决方法：一般是因为定义了对象却没有开辟空间而直接使用造成的，或者是因为调用的方法返回了 null 值而不是一个对象。前者定义时直接开辟空间皆可解决，后者需

要对 null 值进行特殊处理，譬如以异常处理的形式来避免程序的运行崩溃。

七、总结

本学期由于较为空闲，一直都在自学 java web 的相关知识，这次课程设计正好给了我一个机会实践，用 java 来实现一个“麻雀虽小，五脏俱全”的 web 应用。同时，在暑假实习中了解到了目前主流的前后端分离技术，这次正好也尝试了一下：用 java 来开发后端，用目前的流行框架 vue 来写前端页面，两者相互独立，基于 http 协议传输 json 格式数据来相互通信。

开发过程总体来说还是比较顺利的，后端因为之前也写过几个小 demo，所以搭起来还是比较快的，前端由于 vue 官方提供了非常详细的文档和教程，上手也很快，而且有很多第三方现成的组件可以使用，不用自己再写 css 和 javascipt 来调样式，十分方便，可以只专注于业务逻辑。

对于开发出的系统总体还是比较满意的，功能很齐全，界面也很美观，但是还是存在一些不足。譬如数据库设计的时候没有考虑周全，没有考虑到教师和课程是多对多的关系，将其简化成了一对多，这样如果一门课程如果需要多个老师来教那么系统就无法实现了。还有就是密码应该加密存储而不是明文存储，以加强系统安全性。这些问题在以后的开发中会更加关注。