

数据库实践课程报告

题目： “桃子读书” 自主学习平台——阅读课程管理系统

组员姓名： 初阳

组员学号： 10165102158

完成时间： 2018/5/27

华东师范大学计算机科学技术系

目 录

一.	系统需求分析	1
1.	系统描述	1
2.	数据存储需求.....	1
3.	系统常做的查询与更新.....	2
4.	应用程序功能.....	3
二.	数据库概念设计	6
1.	确定实体和属性.....	6
2.	E-R 图	6
三.	数据库逻辑结构设计	8
1.	关系模式设计.....	8
2.	基本表设	8
四.	数据库物理设计和实施	13
1.	数据库的创建.....	13
2.	创建基本表	13
3.	存储过程设计.....	17
4.	函数设计	22
5.	视图设计	26
五.	应用程序设计	30
1.	开发及运行环境介绍.....	30
2.	主要功能设计.....	30
3.	主要界面	31
六.	心得体会	44

一. 系统需求分析

1. 系统描述

在儿童成长和学习的过程中，阅读是非常重要的教育手段，也是一项十分重要的任务，能够帮助儿童更加健康、积极的成长。儿童文学在儿童阅读中的地位不可取代，不仅能够培养儿童阅读的兴趣、提升他们的阅读能力。还能提高儿童的审美感受力、审美判断力以及审美理解力。只有在儿童早期提高阅读能力并让儿童拥有丰富的阅读经验，儿童才能在未来的成长和学习中获得更加突出的优势，儿童日后的提高和进步才能有一个较好的基础。因此，我们应该认真对待儿童早期的阅读，精心挑选阅读材料并进行更人性化的阅读指导工作是十分有必要的。

“桃子读书”自主学习平台就是立足于儿童阅读的线上平台。它能甄选最好的书籍给适合的孩子，邀请最好的老师为孩子讲述，并进行个性化指导，开发孩子们的潜能。而本阅读课程管理系统就是“桃子读书”自主学习平台的教师管理端和学生学习端。

本系统面向用户分为管理员、普通教师和学生。普通教师对学生作业进行批改，反馈，尤为重要是可以跟踪学生的学习记录。管理员可以进行课程和班级的管理和用户数据统计分析。本系统还模拟了用户的小程序端，学生可以进行书籍阅读、课程购买，同时可以进行作业提交与老师互动。

因此，阅读课程管理系统跨越了时间和空间的限制，给教育带来了变革，也给学生带来了便捷。并且可以使儿童阅读涵盖面更广，推行更远，让更多的孩子能在阅读的世界里培养出最好的自己。

2. 数据存储需求

阅读课程管理系统数据库需要存储如下信息。有关注册用户的信息涉及用户编号、用户名学分、用户问卷反馈、用户推荐结果与用户余额信息。有关教师信息涉及教师 ID，教师邮箱，教师姓名，教师登录密码，教师简介。有关套课（即，一套课程，包含多个单独的课程）信息涉及：套课 ID，套课的标题，套课所属的主题，套课适合的年龄阶段，套课封面图片的 url 地址，套课的介绍，套课相关书

籍的购买链接，套课的积分，套课的价格。有关班级信息涉及：班级 ID，任课教师 ID，配套课程 ID，开始时间，结束时间。有关单课（即，一个课程，多个课程组成套课）信息涉及：单课 ID，单课的标题，单课相关视频的 url 地址，单课的介绍，单课封面的地址。有关题目信息涉及：题目 ID，题干，选项题干，答案，题目类型（0：客观，1：主观），所属单课 ID，所在题号。有关作业信息涉及：作业 ID，班级 ID，学生 ID，单课 ID，标记位。

每个单课对应一套题目。每个套课对应多个单课，而单课可以组成多个不同的套课。每个班级对应一个套课，一个教师，而套课可由多个不同的班级进行学习。学生可以加入多个班级，教师也可负责多个班级。学生可以通过加入班级跟随套课进行单课的学习，学生也可以自由选择直接进行单课的学习。学生加入班级后，需要完成班级讲授的课程的作业，教师则对其负责的班级的学生作业进行反馈和批改。而自由选择单课的学习则无需完成作业，也无老师负责批改。

3. 系统常做的查询与更新

经常做的查询，或许对创建索引有影响的：

- 根据班级信息查询对应单课
- 根据班级和学生信息查询对应作业
- 根据单课的标题查询单课相关信息
- 根据套课的标题查询套课相关信息

根据经常做的查询，需要创建有关视图的：

- 根据学生 ID，查询其已学习过的全部单课（包括直接学习的单课和跟随班级学习的单课）
- 根据班级 ID 查询其对应单课及其顺序
- 根据学生 ID，班级 ID 对作业进行查询
- 建立教师与学生间的连接
- 查询学生可用余额

关于更新

- 添加、修改、删除单课
- 添加、修改、删除套课

- 添加、修改、删除班级
- 添加、修改、删除教师

4. 应用程序功能

前台用户的主要功能如下：

1) 个人中心：

- a) 个人信息查看：显示已获学分，问卷调查结果，推荐课程，余额
- b) 余额充值：进行余额充值操作
- c) 课程列表查看：显示在修课程，以及作业完成状态（未完成：灰色；已提交，未批改：绿色；已批改：蓝色）
- d) 作业查看：
 - 未完成作业：显示作业题目及作业提交框
 - 已提交，未批改作业：显示作业题目以及已提交结果
 - 已批改作业：显示作业题目，提交结果和教师反馈

2) 单课查看：

- a) 单课信息：显示是否已购（已购买：灰色；未购买：绿色），课程封面，课程简介
- b) 单课阅读：
 - 已购单课：直接进入阅读界面，显示单课题目，单课内容
 - 未购单课：先进行购买，再进入单课阅读界面

3) 班级查看：

- a) 班级信息：显示未购买的班级信息，包括班级 ID，教师名称，课程名称，开始时间，结束时间，课程主题，课程阶段，课程介绍，课程学分，课程价格
- b) 加入班级：

对余额进行检测，若不足则需充值。班级加入（购买）成功后跳转到学习界面。

前台教师的主要功能如下：

1) 待批改列表：

- 尚未批改的学生作业信息：包括班级课程信息，作业标号，作业已提交人数，班级总人数

2) 我的班级：

- a) 班级信息列表：包括班级号，课程名称，上次更新时间，在修课程人数
- b) 班级信息查看：
 - 课程信息：包括单课列表，单课信息
 - 课程完成情况：该班级的学生课程完成情况（未提交：灰色；已提交，未批改：绿色；已批改：蓝色）

3) 我的学生：

- a) 学生信息列表：包括学生 ID，班级号，课程号
- b) 学生信息查看：显示已获学分，问卷调查结果，推荐课程
- c) 课程完成情况：该学生所修的课程完成情况（未提交：灰色；已提交，未批改：绿色；已批改：蓝色）

4) 作业批改：

- a) 未批改作业：显示作业题目
- b) 已提交，未批改作业：显示作业题目，评语提交框
- c) 已批改作业：显示作业题目，作业评语

后台管理员的主要功能如下：

1) 管理教师：

- a) 教师信息列表：显示教师姓名，教师邮箱，教师介绍
- b) 教师信息添加
- c) 教师信息修改
- d) 教师信息删除

2) 管理学生：

- a) 学生信息列表：显示学生 ID，当前学分，问卷调查结果，课程推荐结果
- b) 学生信息查看：
 - 显示已获学分，问卷调查结果，推荐课程

- 课程完成情况：该学生所修的课程完成情况
- 作业显示：对各种状态的作业可进行查看

3) 管理班级：

- a) 班级列表信息：显示班级 ID，教师姓名，课程名称，开始时间，结束时间
- b) 班级信息添加
- c) 班级信息删除
- d) 班级休息修改：未开课时可修改（即没有学生加入）

4) 管理课程：

- a) 课程列表信息：显示课程名，课程主题，课程阶段，课程简介，课程学分，课程价格
- b) 课程信息查看：显示缩略信息和所对应的单课课程列表
- c) 课程信息修改：未开课时可修改（即没有学生加入）
- d) 课程信息删除
- e) 课程信息添加

5) 管理单课

- a) 单课信息列表：显示课程封面，课程名，课程简介
- b) 单课信息查看：显示作业题目
- c) 单课信息添加：包括作业题目添加
- d) 单课信息修改：包括作业题目修改
- e) 单课信息删除

二. 数据库概念设计

1. 确定实体和属性

分析阅读课程管理平台的系统需求，将系统中设计的人、物进行抽象，得到了系统的实体如下：

- 1) 学生信息实体集。属性包括：学生 ID，当前积分，问卷结果，推荐课程结果
- 2) 教师信息实体集。属性包括：教师 ID，邮箱，姓名，登录密码，教师简介
- 3) 管理员实体集。属性包括：管理员 ID，邮箱，姓名，登录密码
- 4) 套课信息实体集。属性包括：套课 ID，套课的标题，套课所属的主题，套课适合的年龄阶段，套课封面图片的 url 地址，套课的介绍，套课相关书籍的购买链接，套课的积分，套课的价格
- 5) 班级信息实体集合。属性包括：班级 ID，任课教师 ID，配套课程 ID，开始时间，结束时间
- 6) 单课信息实体集。属性包括：单课 ID，单课的标题，单课相关视频的 url 地址，单课的介绍，单课封面的地址
- 7) 题目信息实体集。属性包括：题目 ID，题干，选项题干，答案，题目类型（0：客观，1：主观），所属单课 ID，所在题号
- 8) 作业信息实体集。属性包括：作业 ID，班级 ID，学生 ID，单课 ID，标记位（1：未提交，未批改；2：已提交，未批改；3：已提交，已批改）

2. E-R 图

系统 E-R 图如图 2-1 所示：

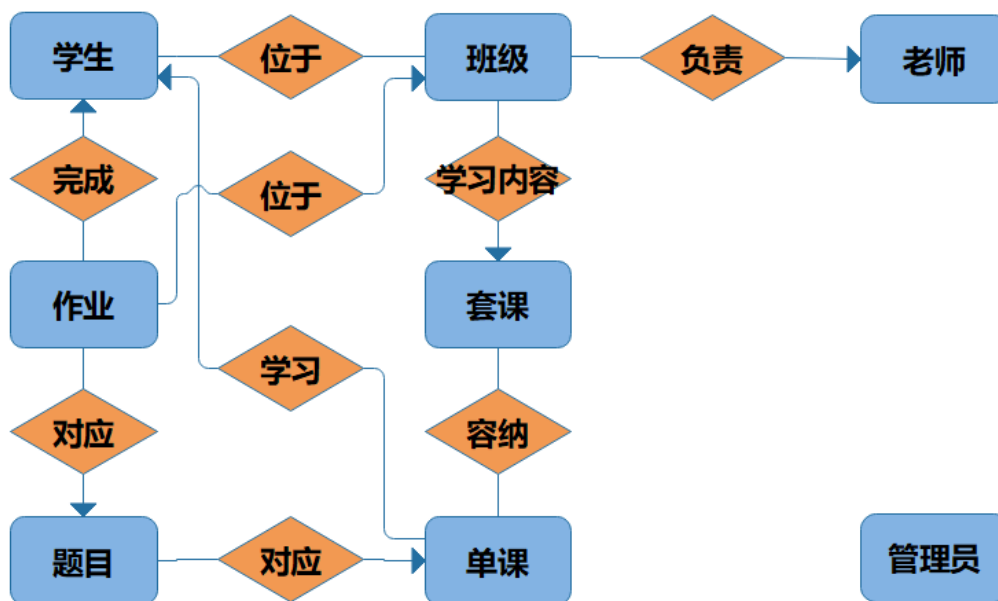


图 2-1 E-R 图

三. 数据库逻辑结构设计

1. 关系模式设计

根据概念结构设计得到的 E-R 图和转换规则,得到如下关系模式(主键用下划线标出,外键用斜体标出):

学生信息表(学生 ID, 当前积分, 问卷结果, 推荐课程结果, 当前余额)

教师信息表(教师 ID, 邮箱, 姓名, 登录密码, 教师简介)

管理员信息表(管理员 ID, 邮箱, 姓名, 登录密码)

套课信息表(套课 ID, 套课的标题, 套课所属的主题, 套课适合的年龄阶段, 套课封面图片的 url 地址, 套课的介绍, 套课相关书籍的购买链接, 套课的积分, 套课的价格)

班级信息表(班级 ID, *任课教师 ID*, *配套课程 ID*, 开始时间, 结束时间)

单课信息表(单课 ID, 单课的标题, 单课相关视频的 url 地址, 单课的介绍, 单课封面的地址)

题目信息表(题目 ID, 题干, 选项题干, 答案, 题目类型, *所属单课 ID*, *所在题号 ID*)

作业信息表(作业 ID, *班级 ID*, *学生 ID*, *单课 ID*, 标记位)

学生班级关系表(*班级 ID*, *学生 ID*)

套课单课关系表(*关系 ID*, *单课 ID*, *套课 ID*, 单课所在套课顺序)

学生直接学习单课关系表(*学生 ID*, *单课 ID*)

学生通过班级学习单课关系表(*学生 ID*, *单课 ID*)

2. 基本表设

基本表设计如表 3-1~3-12 所示。

表 3-1: 学生信息表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
stu_id	VARCHAR(15)	否			主键	学生 ID
credit	INT(11)	否		0		已获学分
qresult	VARCHAR(50)	是				问卷结果
rec_result	VARCHAR(255)	是				推荐结果
balance	DECIMAL(10,2)	否		0		账户余额

表 3-2：教师信息表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
tea_id	VARCHAR(15)	否			主键	教师 ID
tea_email	INT(11)	否				教师邮箱
tea_name	VARCHAR(50)	是				教师姓名
Password	VARCHAR(25)	是				教师密码
Tea_intro	VARCHAR(255)	是				教师介绍

表 3-3：管理员信息表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
Id	INT(11)	否			主键	管理员 ID
email	VARCHAR(50)	否				管理员邮箱
password	VARCHAR(50)	否				管理员密码
name	VARCHAR(255)	是				管理员姓名

表 3-4：套课信息表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
scourse_id	INT(11)	否			主键	套课 ID
scourse_title	VARCHAR(60)	否				套课名称
scourse_theme	VARCHAR(50)	是				套课主题
scourse_stage	VARCHAR(50)	是				套课阶段

pageimg_urls	VARCHAR(255)	是				套课封面
scourse_intro	VARCHAR(255)	是				套课介绍
buylink	VARCHAR(255)	是				购买链接
scourse_credit	INT(11)	是		0		套课学分
scourse_price	DECIMAL(10,2)	是		0.00		套课价格

表 3-5：班级信息表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
class_id	INT(11)	否			主键	班级 ID
tea_id	INT(11)	是				班级教师
scourse_id	INT(11)	是				班级课程
start_time	datetime	是				开始时间
end_time	datetime	是				结束时间

表 3-6：单课信息表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
pcourse_id	INT(11)	否			主键	单课 ID
vedio_url	VARCHAR(255)	是				视频地址
pcourse_intro	TEXT	是				单课介绍
pcourse_imgurl	VARCHAR(255)	是				单课封面
pcourse_title	VARCHAR(60)	是				单课标题

表 3-7：题目信息表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
pro_id	INT(11)	否			主键	题目 ID
pro_stem	TEXT	是				题干
choice	TEXT	是				选项
answer	CHAR(1)	是				答案

flag	INT(11)	是	0,1	1		题型 0: 客观 1: 主观
pcourse_id	INT(11)	是				所对应单课
pro_num	INT(11)	是				题号

表 3-8: 作业信息表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
homework_id	INT(11)	否			主键	作业 ID
class_id	INT(11)	是				班级 ID
stu_id	VARCHAR(15)	是				学生 ID
pcourse_id	INT(11)	是				单课 ID
done	INT(11)	是	0,1,2	0		完成状态
Comment	TEXT	是				老师评语
Store_url	VARCHAR(255)	是				作业存储地址

表 3-9: 学生班级关系表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
stu_id	VARCHAR(15)	否			主键	学生 ID
class_id	INT(11)	否			主键	班级 ID

表 3-10: 套课单课关系表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
scourse_id	INT(11)	否			主键	套课 ID
pcourse_id	INT(11)	否			主键	单课 ID
thenumber	INT(11)	否			主键	单课在套课内顺序

表 3-11：学生直接学习单课关系表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
Stu_id	VARCHAR(15)	否			主键	学生 ID
Pcourse_id	INT(11)	否			主键	单课 ID

表 3-12：学生通过班级学习单课关系表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
Stu_id	VARCHAR(15)	否			主键	学生 ID
Pcourse_id	INT(11)	否			主键	单课 ID

四. 数据库物理设计和实施

1. 数据库的创建

使用 MySQL 建立阅读课程管理系统的数据库，数据库基本信息如下：

Source Server : localhost_3306
Source Server Version : 50717
Source Host : localhost:3306
Source Database : readingmanagement

Target Server Type : MYSQL
Target Server Version : 50717
File Encoding : 65001

2. 创建基本表

```
CREATE TABLE `admin_info` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `email` varchar(50) NOT NULL,  
  `password` varchar(50) DEFAULT NULL,  
  `name` varchar(255) DEFAULT NULL,  
  PRIMARY KEY (`id`),  
  UNIQUE KEY `email` (`email`)  
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;  
  
CREATE TABLE `class_info` (  
  `class_id` int(11) NOT NULL AUTO_INCREMENT,  
  `tea_id` int(11) DEFAULT NULL,  
  `scourse_id` int(11) DEFAULT NULL,  
  `start_time` datetime DEFAULT NULL,  
  `end_time` datetime DEFAULT NULL,  
  PRIMARY KEY (`class_id`),  
  KEY `tea_id` (`tea_id`),  
  KEY `scourse_id` (`scourse_id`),  
  CONSTRAINT `class_info_ibfk_1` FOREIGN KEY (`tea_id`)
```

```

REFERENCES `teacher_info` (`tea_id`),
    CONSTRAINT `class_info_ibfk_2` FOREIGN KEY (`scourse_id`)
REFERENCES `setcourse_info` (`scourse_id`)
) ENGINE=InnoDB AUTO_INCREMENT=13 DEFAULT CHARSET=utf8;

CREATE TABLE `homework_info` (
  `homework_id` int(11) NOT NULL AUTO_INCREMENT,
  `class_id` int(11) NOT NULL,
  `stu_id` varchar(15) NOT NULL,
  `pcourse_id` int(11) NOT NULL,
  `done` int(11) DEFAULT '0',
  `comment` text,
  `store_url` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`homework_id`),
  KEY `pcourse_id` (`pcourse_id`),
  KEY `homework_info_ibfk_1` (`class_id`,`stu_id`),
  CONSTRAINT `homework_info_ibfk_1` FOREIGN KEY (`class_id`,
`stu_id`) REFERENCES `inclass` (`class_id`, `stu_id`) ON
DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `homework_info_ibfk_2` FOREIGN KEY
(`pcourse_id`) REFERENCES `percourse_info` (`pcourse_id`)
) ENGINE=InnoDB AUTO_INCREMENT=31 DEFAULT CHARSET=utf8;

CREATE TABLE `inclass` (
  `stu_id` varchar(15) NOT NULL,
  `class_id` int(11) NOT NULL,
  PRIMARY KEY (`stu_id`,`class_id`),
  KEY `class_id` (`class_id`,`stu_id`),
  CONSTRAINT `inclass_ibfk_1` FOREIGN KEY (`stu_id`)
REFERENCES `student_info` (`stu_id`) ON DELETE CASCADE ON
UPDATE CASCADE,
  CONSTRAINT `inclass_ibfk_2` FOREIGN KEY (`class_id`)
REFERENCES `class_info` (`class_id`) ON DELETE CASCADE ON
UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `percourse_info` (
  `pcourse_id` int(11) NOT NULL AUTO_INCREMENT,
  `vedio_url` varchar(255) DEFAULT NULL,
  `pcourse_intro` text,
  `pcourse_imgurl` varchar(255) DEFAULT NULL,
  `pcourse_title` varchar(60) DEFAULT NULL,
  PRIMARY KEY (`pcourse_id`)
) ENGINE=InnoDB AUTO_INCREMENT=103 DEFAULT CHARSET=utf8;

```



```
CREATE TABLE `problems` (  
  `pro_id` int(11) NOT NULL AUTO_INCREMENT,  
  `pro_stem` text,  
  `choice` text,  
  `answer` char(1) DEFAULT NULL,  
  `flag` int(11) DEFAULT NULL,  
  `pcourse_id` int(11) DEFAULT NULL,  
  `pro_num` int(11) DEFAULT NULL,  
  PRIMARY KEY (`pro_id`),  
  KEY `pcourse_id` (`pcourse_id`),  
  CONSTRAINT `problems_ibfk_1` FOREIGN KEY (`pcourse_id`)  
REFERENCES `percourse_info` (`pcourse_id`)  
) ENGINE=InnoDB AUTO_INCREMENT=310 DEFAULT CHARSET=utf8;  
  
CREATE TABLE `setcourse_info` (  
  `scourse_id` int(11) NOT NULL AUTO_INCREMENT,  
  `scourse_title` varchar(60) NOT NULL,  
  `scourse_theme` varchar(50) DEFAULT NULL,  
  `scourse_stage` varchar(50) DEFAULT NULL,  
  `pageimg_urls` varchar(255) DEFAULT NULL,  
  `scourse_intro` varchar(255) DEFAULT NULL,  
  `buylink` varchar(255) DEFAULT NULL,  
  `scourse_credit` int(11) DEFAULT '0',  
  `scourse_price` decimal(10,2) DEFAULT '0.00',  
  PRIMARY KEY (`scourse_id`),  
  UNIQUE KEY `scourse_id` (`scourse_id`)  
) ENGINE=InnoDB AUTO_INCREMENT=12 DEFAULT CHARSET=utf8;  
  
CREATE TABLE `student_info` (  
  `stu_id` varchar(15) NOT NULL,  
  `credit` int(11) DEFAULT NULL,  
  `qresult` varchar(50) DEFAULT NULL,  
  `rec_result` varchar(255) DEFAULT NULL,  
  `balance` decimal(10,0) DEFAULT NULL,  
  PRIMARY KEY (`stu_id`),  
  UNIQUE KEY `stu_id` (`stu_id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;  
  
CREATE TABLE `stu_class_read` (  
  `stu_id` varchar(11) NOT NULL,  
  `pcourse_id` int(11) NOT NULL,  
  PRIMARY KEY (`stu_id`, `pcourse_id`),  
  KEY `pcourse_id` (`pcourse_id`),
```

```
CONSTRAINT `stu_class_read_ibfk_1` FOREIGN KEY (`stu_id`)
REFERENCES `student_info` (`stu_id`) ON DELETE CASCADE ON
UPDATE CASCADE,
CONSTRAINT `stu_class_read_ibfk_2` FOREIGN KEY
(`pcourse_id`) REFERENCES `percourse_info` (`pcourse_id`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `stu_read` (
  `stu_id` varchar(255) NOT NULL,
  `pcourse_id` int(11) NOT NULL,
  PRIMARY KEY (`stu_id`,`pcourse_id`),
  KEY `stu_read_ibfk_2` (`pcourse_id`),
  CONSTRAINT `stu_read_ibfk_1` FOREIGN KEY (`stu_id`)
REFERENCES `student_info` (`stu_id`) ON DELETE CASCADE ON
UPDATE CASCADE,
CONSTRAINT `stu_read_ibfk_2` FOREIGN KEY (`pcourse_id`)
REFERENCES `percourse_info` (`pcourse_id`) ON DELETE CASCADE
ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `s_p_class_info` (
  `scourse_id` int(11) NOT NULL,
  `pcourse_id` int(11) NOT NULL,
  `thenumber` int(11) NOT NULL,
  PRIMARY KEY (`scourse_id`,`pcourse_id`,`thenumber`),
  UNIQUE KEY `scourse_id`
(`scourse_id`,`pcourse_id`,`thenumber`),
  KEY `pcourse_id` (`pcourse_id`),
  CONSTRAINT `s_p_class_info_ibfk_1` FOREIGN KEY
(`scourse_id`) REFERENCES `setcourse_info` (`scourse_id`) ON
DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT `s_p_class_info_ibfk_2` FOREIGN KEY
(`pcourse_id`) REFERENCES `percourse_info` (`pcourse_id`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `teacher_info` (
  `tea_id` int(11) NOT NULL AUTO_INCREMENT,
  `tea_email` varchar(50) NOT NULL,
  `tea_name` varchar(25) DEFAULT NULL,
  `password` varchar(25) DEFAULT NULL,
  `teacher_intro` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`tea_id`),
```

```

UNIQUE KEY `tea_id` (`tea_id`),
UNIQUE KEY `tea_email` (`tea_email`)
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=utf8;

```

3. 存储过程设计

- 1) 学生加入班级，输入为学生 ID 和班级 ID。学生加入班级后，应新建该学生的班级对应的作业，同时也应该更新“学生通过班级学习单课关系表”即表示该学生已将该班级对应的单课加入学习计划。

先获得当前课程的价钱(price)，再用学生的余额减去价钱(balance=balance-price)。获得当前课程的学分(t_credit)，用学生学分加上此课程学分(credit=credit+t_credit)。完成学生加入班级操作后。完成学生相关课程的作业创建，获得当前班级的所有单课 ID，对每个单课进行作业的创建，作业的状态都为 0（未提交，未批改）。

获得该班级对应的单课列表，更新“学生通过班级学习单课关系表”。

```

CREATE PROCEDURE `p_join_class`(
IN `p_stu_id` varchar(15),
IN `p_class_id` int)
BEGIN
DECLARE tt int DEFAULT 0;
DECLARE price float DEFAULT 0;
DECLARE t_credit int default 0;

DECLARE done INT DEFAULT 0; #repeat 结束标识

DECLARE cur1 CURSOR FOR SELECT pcourse_id FROM
class_percourse
WHERE class_id = p_class_id ORDER BY thenumber ;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

SELECT scourse_price into price from stu_class_tea_course

```

```

WHERE class_id = p_class_id;
SELECT scourse_credit into t_credit
from stu_class_tea_course WHERE class_id = p_class_id;
INSERT into
inclass(class_id,stu_id) VALUES(p_class_id,p_stu_id);
UPDATE student_info set balance = balance-price,credit =
credit+t_credit where stu_id = p_stu_id;
OPEN curl;
REPEAT
FETCH curl INTO tt;
IF NOT DONE THEN
INSERT into homework_info (class_id,stu_id,pcourse_id)
VALUES (p_class_id,p_stu_id,tt) ;
INSERT into stu_class_read (stu_id,pcourse_id)
VALUES(p_stu_id,tt);
END IF;
UNTIL done END REPEAT;
CLOSE curl;
END

```

- 2) 学生阅读单本书籍，输入为学生 ID 和书籍 ID（单课 ID）。先扣除学生的余额（书籍定价均为 1），再更新学生已读书籍表。

```

CREATE PROCEDURE `p_read_percourse` (
IN `p_stu_id` varchar(15),
IN `p_pcourse_id` int)
BEGIN
INSERT into stu_read(stu_id,read_pcourse_id)
VALUES (p_stu_id,p_pcourse_id);
UPDATE student_info set balance = balance-1

```

```
where stu_id = p_stu_id;  
END
```

- 3) 对班级进行创建时，防止重复创建，当教师，套课名称，开课时间，结束时间都与已经存在的班级相同，则不创建新的班级。

```
CREATE PROCEDURE `p_add_class`(  
in p_tea_id int ,  
in p_scourse_id int ,  
in p_start_time VARCHAR(50),  
in p_end_time VARCHAR(50))  
BEGIN  
DECLARE re int default 0;  
SELECT count(*) into re from class_info WHERE tea_id =  
p_tea_id and scourse_id = p_scourse_id  
and start_time = p_start_time  
and end_time = p_end_time;  
if re =0 THEN  
insert                into                class_info  
(tea_id,scourse_id,start_time,end_time)  
values (p_tea_id,p_scourse_id,p_start_time,p_end_time);  
end if;  
END
```

- 4) 向套课中添加单课时，因为不知道当前的单课的序号是多少，则调用此存储过程，先计算套课中已存在的单课数量(num)，则当前单课序号为(num+1)，再进行插入操作。

```
CREATE PROCEDURE `p_add_pcourse2scourse` (in p_scourse_id  
int,in p_pcourse_id int)  
BEGIN
```

```
DECLARE num int DEFAULT 0;
SELECT count(*) into num from s_p_class_info where
scourse_id = p_scourse_id;
set num = num+1;
INSERT                                     into
s_p_class_info(scourse_id,pcourse_id,thenumber)
values(p_scourse_id,p_pcourse_id,num);
END
```

5) 添加教师信息，输入教师相关信息，简化添加操作。

```
CREATE PROCEDURE `p_add_teacher`(
in p_tea_name VARCHAR(20),
in p_tea_email VARCHAR(20),
in pwd VARCHAR(20),
in intro VARCHAR(20)
)
BEGIN
    INSERT                                     into
teacher_info(tea_name,tea_email,password,teacher_intro)
values(p_tea_name,p_tea_email,pwd,intro);
END
```

6) 修改教师信息，输入教师相关信息，简化修改操作。

```
CREATE PROCEDURE `p_mod_teacher`(
in p_tea_id int,
in p_tea_name VARCHAR(20),
in p_tea_email VARCHAR(20),
in pwd VARCHAR(20),
in intro VARCHAR(20)
```

```
)  
BEGIN  
    update                teacher_info                set  
tea_name=p_tea_name,tea_email=p_tea_email,password=pwd,t  
eacher_intro=intro WHERE tea_id = p_tea_id;  
END
```

7) 对单课信息进行修改，输入单课相关信息，完成更新。

```
CREATE          DEFINER=`root`@`localhost`          PROCEDURE  
`p_mod_pcourse`(  
in p_pcourse_id int,  
in p_pcourse_title VARCHAR(255),  
in p_course_intro VARCHAR(500))  
BEGIN  
    update                percourse_info                set  
pcourse_title=p_pcourse_title,pcourse_intro=p_course_int  
ro WHERE pcourse_id = p_pcourse_id;  
END
```

8) 学生余额充值，输入学生 ID 和充值金额。

```
CREATE          PROCEDURE      `p_stu_recharge`(IN      `p_stu_id`  
varchar(15),IN `money` float)  
BEGIN  
    update student_info set balance = balance+money where  
stu_id = p_stu_id;  
END
```

4. 函数设计

- 1) 查看是否可以对套课信息进行修改，输入为套课 ID。若该套课所在的班级已经开课（即，已经有学生选课），则不能更新（返回值 0），否则，可以更新套课信息（返回值=0）。

```
CREATE FUNCTION `f_able_to_mod_scourse`(p_scourse_id
int) RETURNS int(11)
BEGIN
DECLARE num int default 0;
SELECT count(*) into num
from inclass NATURAL join class_info
where scourse_id =p_scourse_id;
return num;
end
```

- 2) 获得一个班级的学生数量，输入为班级 ID。

```
CREATE FUNCTION `f_getclass_student_num`(p_class_id int)
RETURNS int(11)
READS SQL DATA
BEGIN
DECLARE re int default 0;
SELECT count(*) into re FROM inclass
WHERE class_id = p_class_id;
RETURN re;
END
```

- 3) 由于本数据库一个班级有多个学生，多个单个课程和多个作业。为获得学生在当前班级的作业 ID，并且作业 ID 按照班级课程顺序显示。输入为学生 ID 和班级 ID，输出为作业列表（格式为：课程 1 对应的作业_ID，课程 2 对应

的作业_ID, 课程 3 对应的作业_ID..) 的字符串。

```
CREATE FUNCTION `f_gethomework` (p_stu_id
varchar(15),p_class_id int) RETURNS varchar(255) CHARSET
utf8
    READS SQL DATA
BEGIN
    DECLARE s VARCHAR(255) DEFAULT '';
    DECLARE t VARCHAR(255) default '';
    DECLARE tt VARCHAR(255) default '';

    DECLARE done INT DEFAULT 0; #repeat 结束标识

    DECLARE curl CURSOR FOR SELECT
CONCAT(homework_id ,',',don,',',thenumber,';') FROM
total_homework_info
    WHERE stu_id=p_stu_id and class_id = p_class_id ORDER
BY thenumber;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    OPEN curl;

    REPEAT
        FETCH curl INTO tt;

        IF NOT DONE THEN
            set s = CONCAT(s,tt);

        END IF;

    UNTIL done END REPEAT;

    return s;

    CLOSE curl;

END
```

- 4) 学生信息表中存在学分和实际余额, 学分可以部分转换为余额使用, 以激励学生学习。按照可用余额=当前学分*0.1+余额, 计算学生可用余额。输入为

学生 ID，输出为 float 格式的可用余额。

```
CREATE FUNCTION `f_get_balance`(`p_stu_id` varchar(15))
RETURNS float
BEGIN
    DECLARE re float default 0.0;
    SELECT credit/10+balance into re from student_info
    where stu_id = p_stu_id;
    RETURN re;
END
```

- 5) 由于本数据库一个课程对应多个作业。为获得当前课程对应的问题 ID，并且问题 ID 按照规定的问题顺序显示（例，第一题，第二题，第三题）。输入为单课 ID，输出为问题列表（格式为：问题 1_ID，问题 2_ID，问题 3_ID..）的字符串。

```
CREATE FUNCTION `f_get_pcourse_problem`( p_course_id int)
RETURNS varchar(255) CHARSET utf8
BEGIN
    DECLARE s VARCHAR(255) DEFAULT '';
    DECLARE tt VARCHAR(255) default '';
    DECLARE done INT DEFAULT 0; #repeat 结束标识
    DECLARE curl CURSOR FOR SELECT CONCAT(pro_id,',') FROM
    problems
    WHERE pcourse_id = p_course_id ORDER BY pro_num;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
    OPEN curl;
    REPEAT
        FETCH curl INTO tt;
```

```
        IF NOT DONE THEN

            set s = CONCAT(s,tt);

        END IF;

    UNTIL done END REPEAT;

    return s;

    CLOSE curl;

END
```

- 6) 由于本数据库一套课程对应多个单课。为获得当前课程对应的单课 ID，并且单课 ID 按照规定的顺序显示（例，第一课，第二课，第三课）。输入为 套课 ID，输出为单课列表（格式为：单课 1_ID，单课 2_ID，单课 3_ID..）的字符串。

```
CREATE          DEFINER=`root`@`localhost`          FUNCTION
`f_get_scourse_pcourse`(`p_scourse_id`      int)      RETURNS
varchar(255) CHARSET utf8
BEGIN
    DECLARE s VARCHAR(255) DEFAULT '';
    DECLARE tt VARCHAR(255) default '';

    DECLARE done INT DEFAULT 0; #repeat 结束标识

    DECLARE curl CURSOR FOR SELECT CONCAT(pcourse_id,',')
FROM s_p_class_info
    WHERE scourse_id = p_scourse_id ORDER BY thenumber;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    OPEN curl;

    REPEAT

        FETCH curl INTO tt;

        IF NOT DONE THEN

            set s = CONCAT(s,tt);
```

```
END IF;

UNTIL done END REPEAT;

return s;

CLOSE curl;

END
```

7) 获得老师的 email, 输入为教师 ID, 输出为教师 email。

```
CREATE FUNCTION `id_email`(id int) RETURNS varchar(20)
CHARSET utf8
    READS SQL DATA
BEGIN
declare email varchar(20);
SELECT email INTO email FROM teacher_info where
tea_id=id;
RETURN email;
END
```

5. 视图设计

- 1) 因为关系模式的设计中只存在学生直接学习的单课和学生跟随班级学习的单课。而查询经常做学生已经学习过的全部单课。所以建立学生-单课视图。本视图包含学生 ID, 单课 ID, 方便相关查询的执行。

```
CREATE view stu_pcourse as
SELECT * from stu_read UNION SELECT * from stu_class_read
```

- 2) 因为关系模式设计中只存在套课课程和单课课程间的对应, 而查询经常做关于班级和单课课程间的查询。所以建立班级-单课视图。本视图包含班级 ID, 单课 ID, 单课所在位置, 方便相关查询的执行。

```
CREATE VIEW `class_percourse` AS
select `class_info`.`class_id` AS `class_id`,
`s_p_class_info`.`pcourse_id` AS `pcourse_id`,
`s_p_class_info`.`thenumber` AS `thenumber`
from (`class_info` join `s_p_class_info` on
((`class_info`.`scourse_id` =
`s_p_class_info`.`scourse_id`))) ;
```

- 3) 为方便管理员对班级的管理和查询，建立管理员端的班级显示视图。本视图包含班级 ID，教师姓名，套课名称，开始时间，结束时间。

```
CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost`
SQL SECURITY DEFINER VIEW `class_tea_course` AS
select `class_info`.`class_id` AS `class_id`,
`teacher_info`.`tea_name` AS `tea_name`,
`setcourse_info`.`scourse_title` AS `scourse_title`,
`class_info`.`start_time` AS `start_time`,
`class_info`.`end_time` AS `end_time`
from ((`class_info`
join `setcourse_info` on((`class_info`.`scourse_id` =
`setcourse_info`.`scourse_id`)))
join `teacher_info` on((`class_info`.`tea_id` =
`teacher_info`.`tea_id`))) ;
```

- 4) 为方便学生对班级相关信息的查询，建立学生端的班级显示视图。本视图包含班级 ID，教师姓名，套课名称，开始时间，结束时间，套课主题，套课介绍，套课学分，套课价格。

```
CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost`
SQL SECURITY DEFINER VIEW `stu_class_tea_course` AS
select `class_info`.`class_id` AS `class_id`,
```

```

`teacher_info`.`tea_name` AS `tea_name`,
`setcourse_info`.`scourse_title` AS `scourse_title`,
`class_info`.`start_time` AS `start_time`,
`class_info`.`end_time` AS `end_time`,
`setcourse_info`.`scourse_theme` AS `scourse_theme`,
`setcourse_info`.`scourse_intro` AS `scourse_intro`,
`setcourse_info`.`scourse_credit` AS `scourse_credit`,
`setcourse_info`.`scourse_price` AS `scourse_price`
from ((`class_info`
join `setcourse_info` on((`class_info`.`scourse_id` =
`setcourse_info`.`scourse_id`)))
join `teacher_info` on((`class_info`.`tea_id` =
`teacher_info`.`tea_id`))) ;

```

- 5) 为方便对作业与学生，班级，单课，套课等实体间进行查询和管理，建立完成的作业信息视图。本视图包含作业 ID，作业状态，教师评语，作业存储位置，学生 ID，单课 ID，班级 ID，套课 ID，套课标题，作业顺序。

```

CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost`
SQL SECURITY DEFINER VIEW `total_homework_info` AS
select `homework_info`.`homework_id` AS `homework_id`,
`homework_info`.`done` AS `done`,
`homework_info`.`comment` AS `COMMENT`,
`homework_info`.`store_url` AS `store_url`,
`homework_info`.`stu_id` AS `stu_id`,
`homework_info`.`pcourse_id` AS `pcourse_id`,
`homework_info`.`class_id` AS `class_id`,
`class_info`.`scourse_id` AS `scourse_id`,
`setcourse_info`.`scourse_title` AS `scourse_title`,
`s_p_class_info`.`thenumber` AS `thenumber`

```

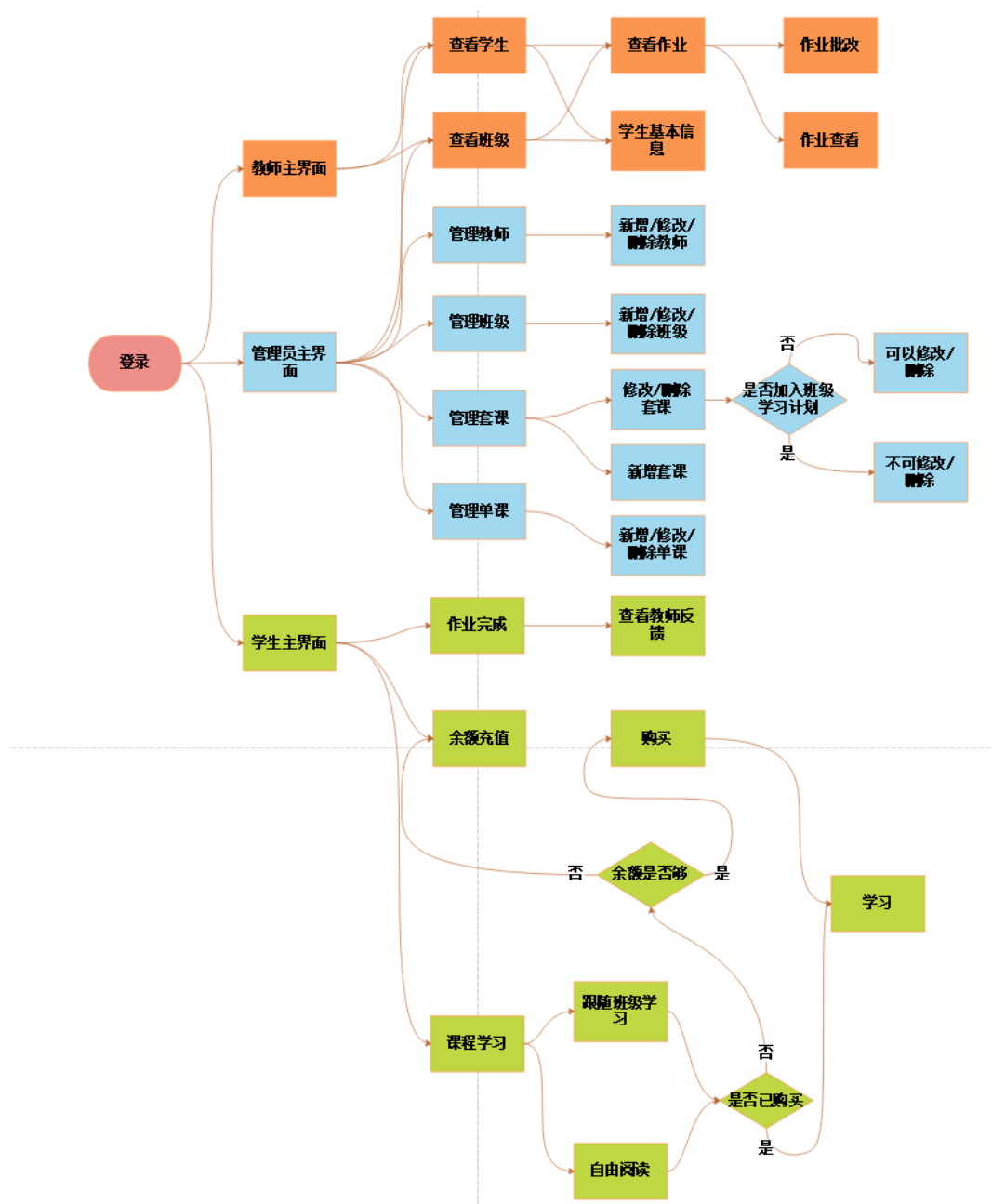
```
from (((`homework_info`  
join    `class_info`    on((`homework_info`.`class_id`    =  
`class_info`.`class_id`)))  
join  `s_p_class_info`  on(((`homework_info`.`pcourse_id`  
=                `s_p_class_info`.`pcourse_id`)          and  
(`class_info`.`scourse_id`                                =  
`s_p_class_info`.`scourse_id`))))  
join  `setcourse_info`  on((`class_info`.`scourse_id`  =  
`setcourse_info`.`scourse_id`))) ;
```

五. 应用程序设计

1. 开发及运行环境介绍

系统使用的开发工具是 JetBrain Pycharm 2018.3.4；开发采用 flask+Jinja 的实现方式；运行环境 win10 64 位系统；浏览器为 Chrome

2. 主要功能设计



3. 主要界面

1) 教师登录界面如图 5-1 所示。

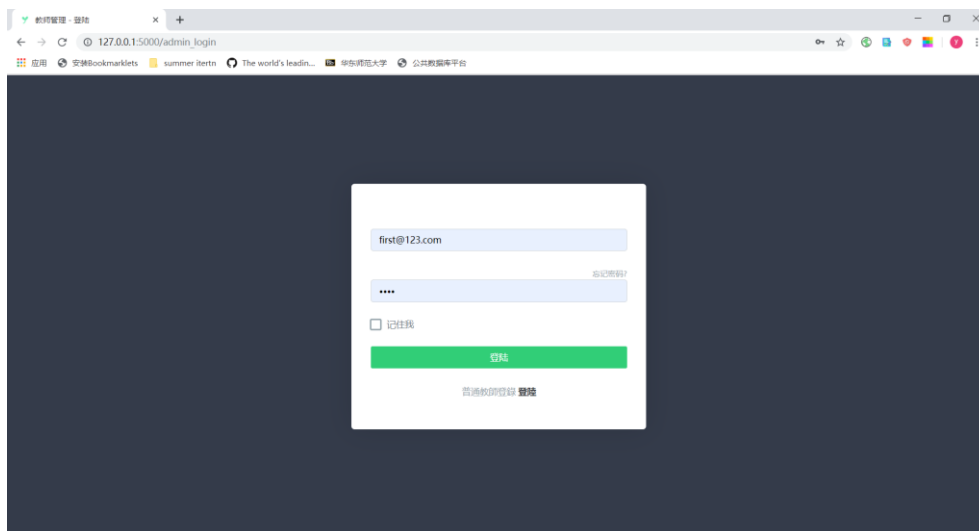


图 5-1 教师登录界面

2) 教师个人中心界面如图 5-2 所示。

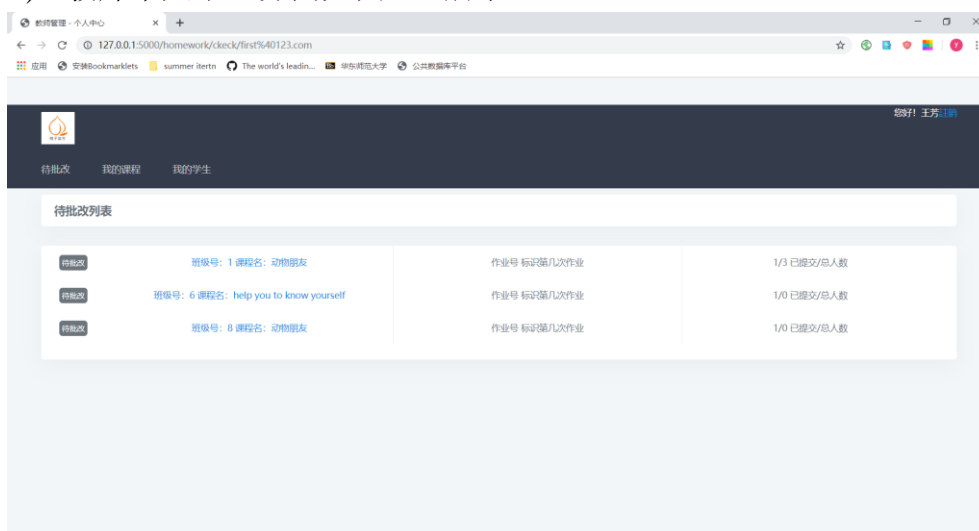


图 5-2 教师个人中心界面

3) 教师以班级为单位，查看班级提交作业情况。教师查看班级界面如图 5-3 所示。

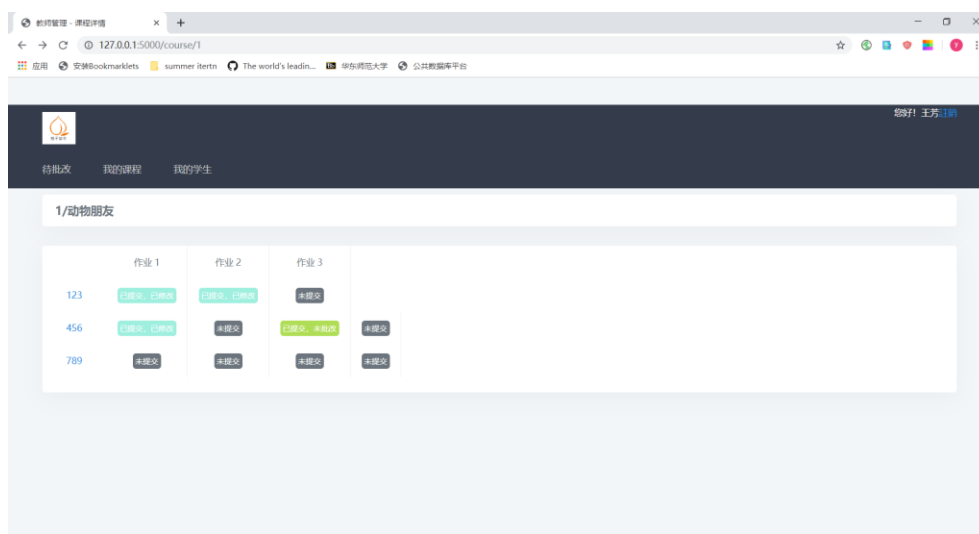


图 5-3 教师查看班级界面

- 4) 教师对已提交，未批改的作业，可进行批改。教师批改作业界面如图 5-4 所示。

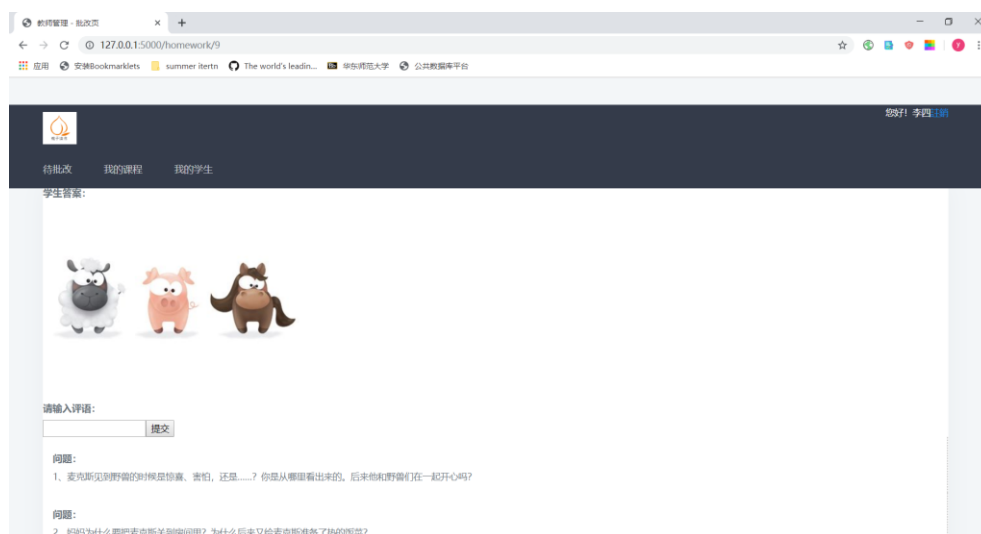


图 5-4 教师批改作业界面

- 5) 对已提交，已批改的作业，可查看批改记录。教师查看已批改作业界面如图 5-5 所示

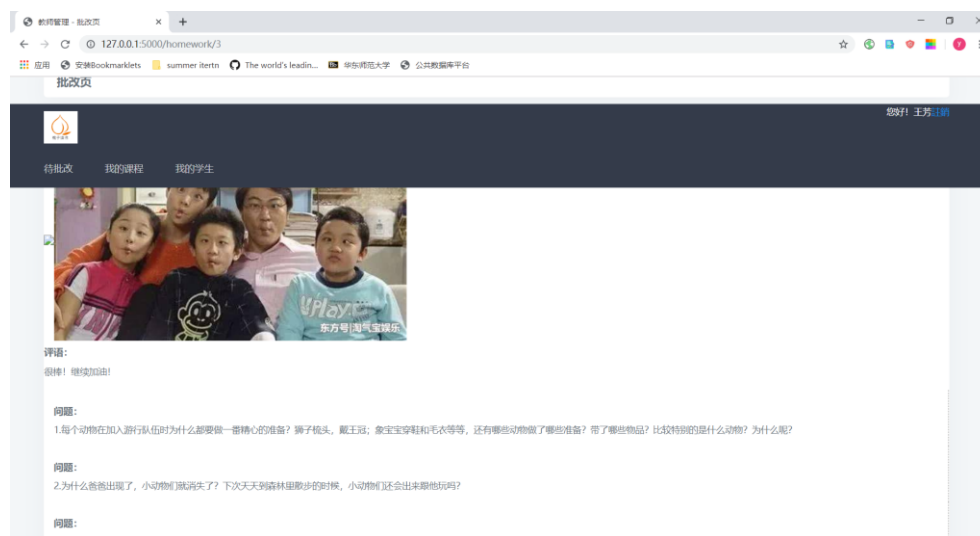


图 5-5 教师查看已批改作业界面

6) 教师查看所教学生列表界面如图 5-6 所示

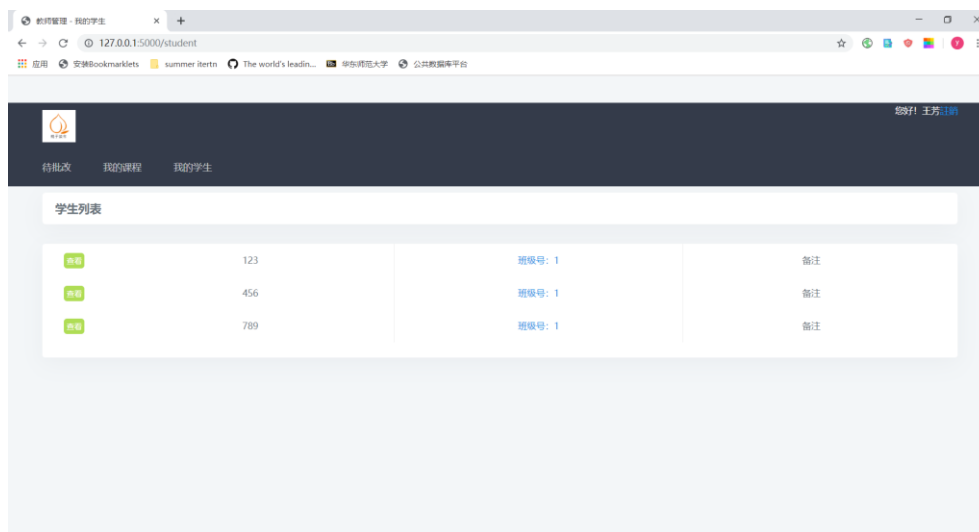


图 5-6 教师查看所教学生列表界面

7) 老师查看单个学生信息, 包括基本信息, 学习记录等。教师查看学生个人信息界面如图 5-7 所示

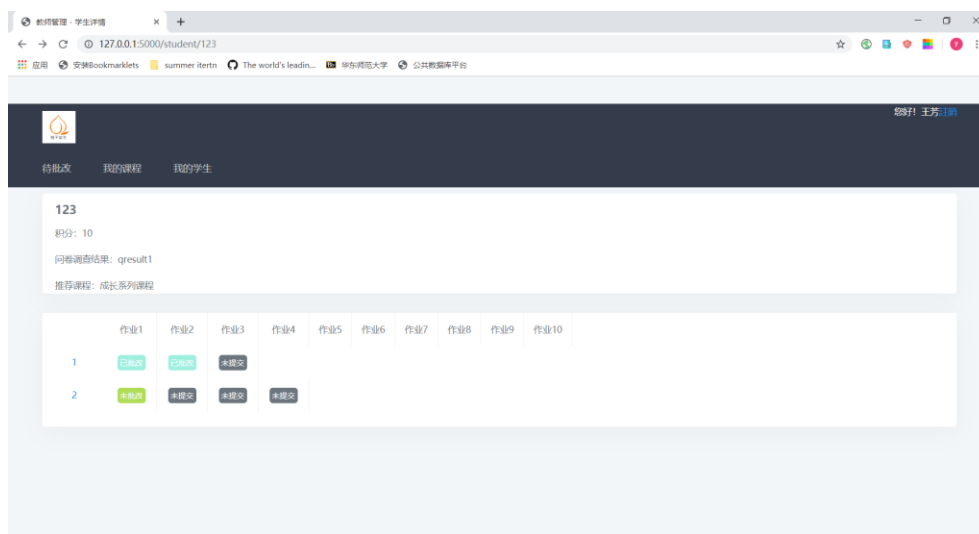


图 5-7 教师查看学生个人信息界面

8) 管理员登录界面如图 5-8 所示。

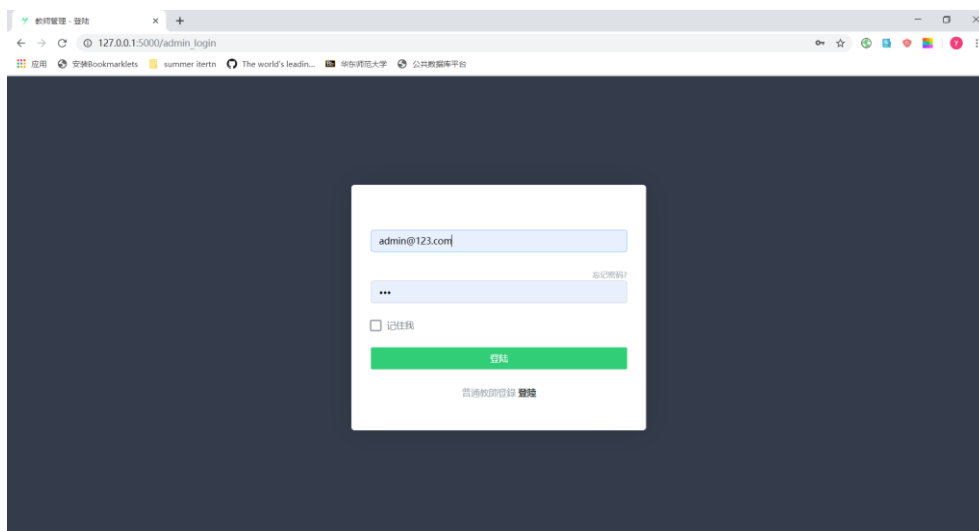


图 5-8 管理员登录界面

9) 管理员管理教师主页面，可进行编辑、删除、添加等操作。管理员管理教师主界面如图 5-9 所示。

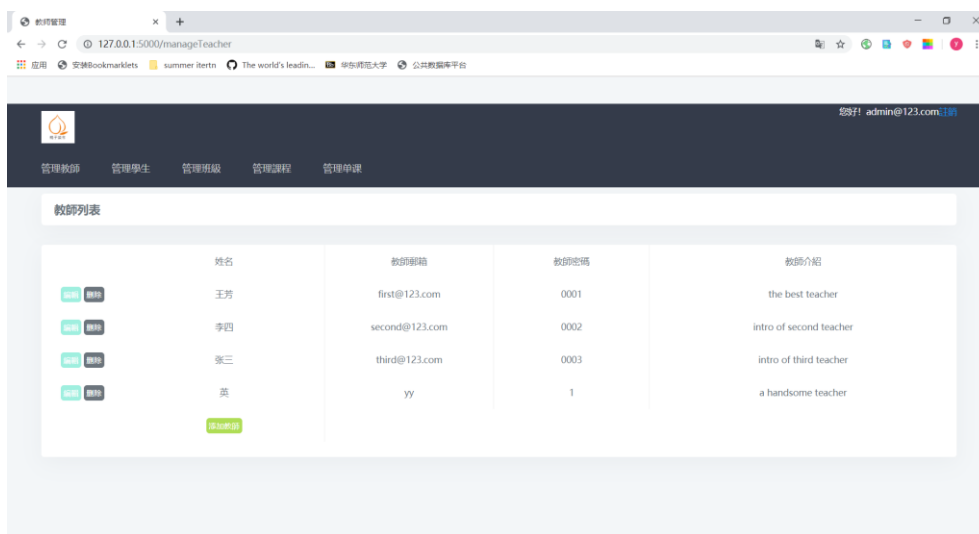


图 5-9 管理员管理教师主界面

10) 管理员管理学生主页面，可查看单个学生的详细信息。管理员管理学生主界面如图 5-10 所示。

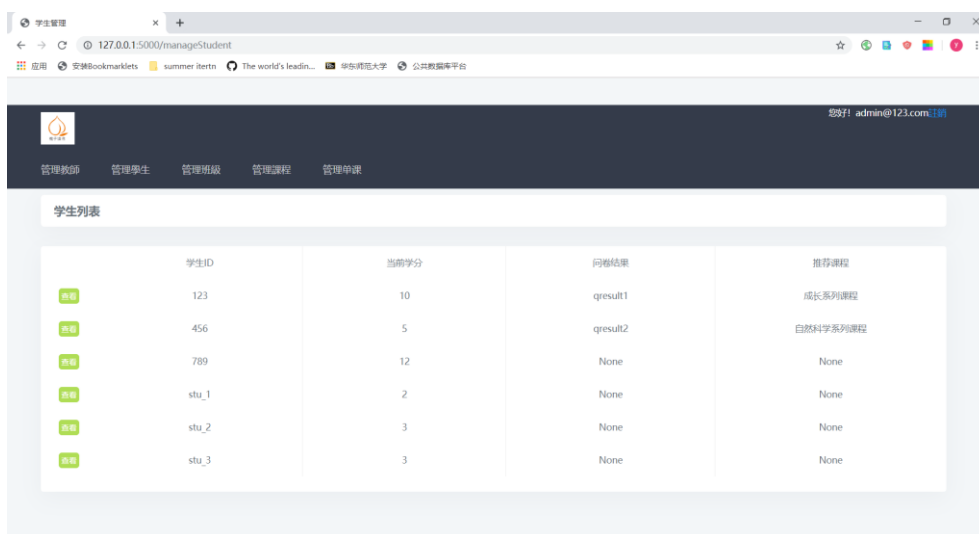


图 5-10 管理员管理学生主界面

11) 管理员管理班级主界面，可进行添加、删除等操作。管理员管理班级主界面如图 5-11 所示。

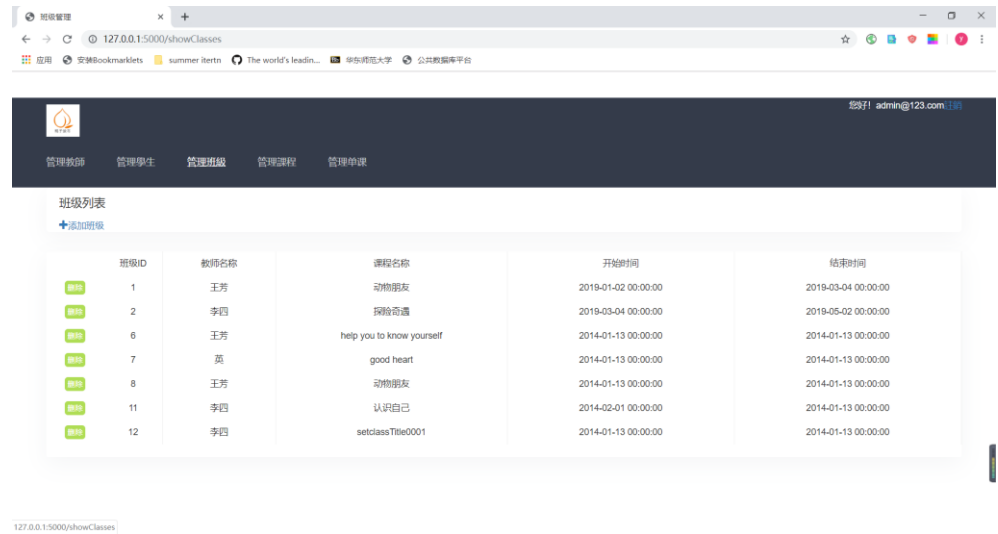


图 5-11 管理员管理班级主界面

12) 管理员管理课程主界面，可进行添加、编辑、查看详情等工作。管理员管理课程主界面如图 5-12 所示。

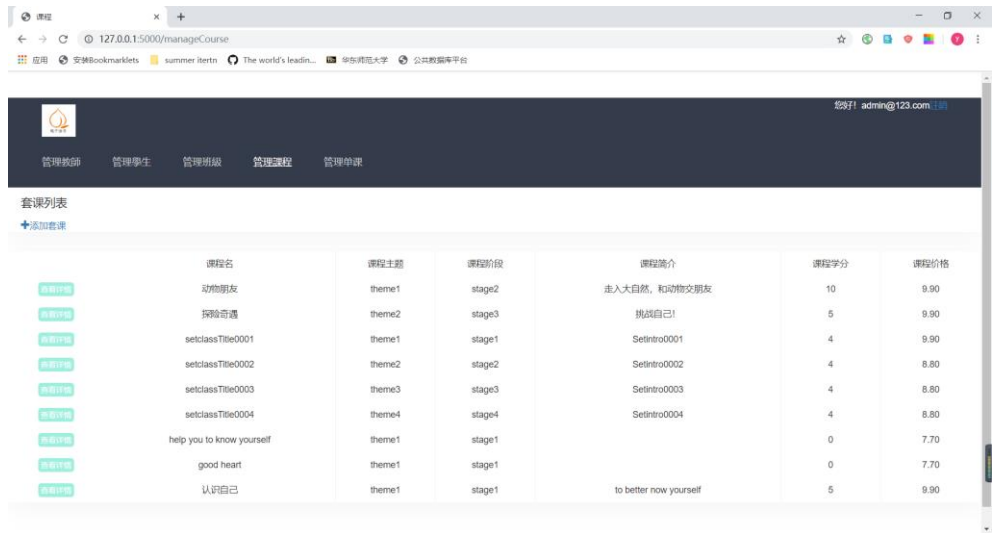


图 5-12 管理员管理课程主界面

- 13) 管理员添加课程界面，新建套课，并选择组成该套课的单课。单课列表由动态从数据库获取实现。管理员添加课程界面如图 5-13 所示。

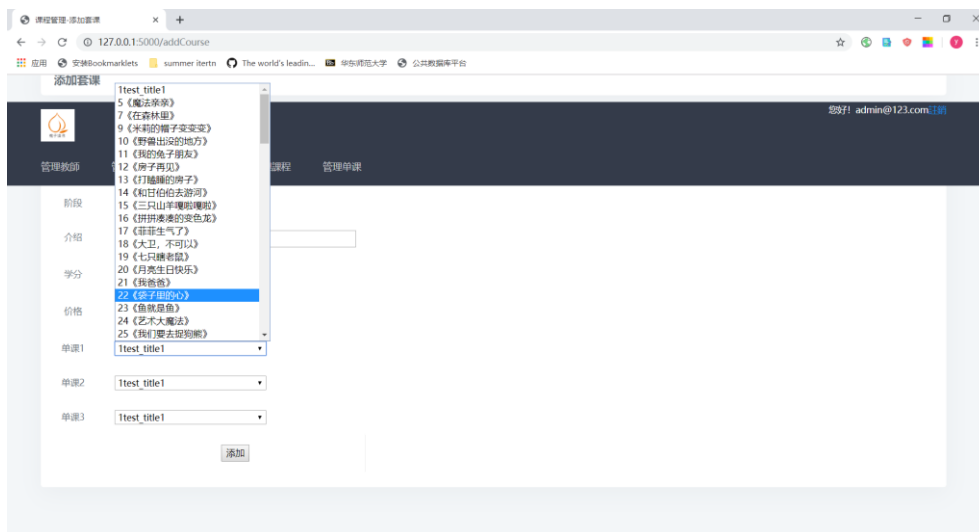


图 5-13 管理员添加课程界面

- 14) 对于已加入班级学习计划的课程，显示不可修改，界面如下。管理员修改课程界面如图 5-14 所示。

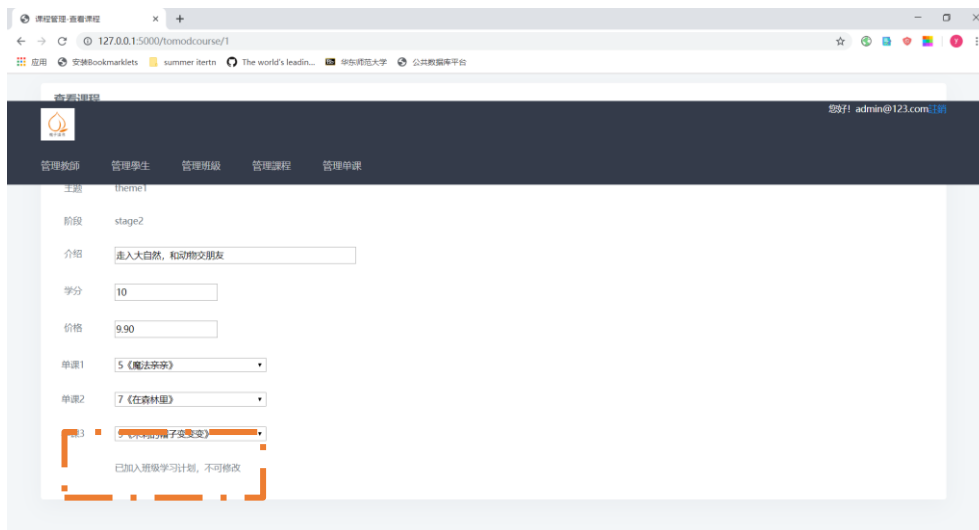


图 5-14 管理员修改课程界面

15) 对于未加入班级学习计划的课程, 显示可以修改, 界面如下。管理员修改课程界面如图 5-15 所示。

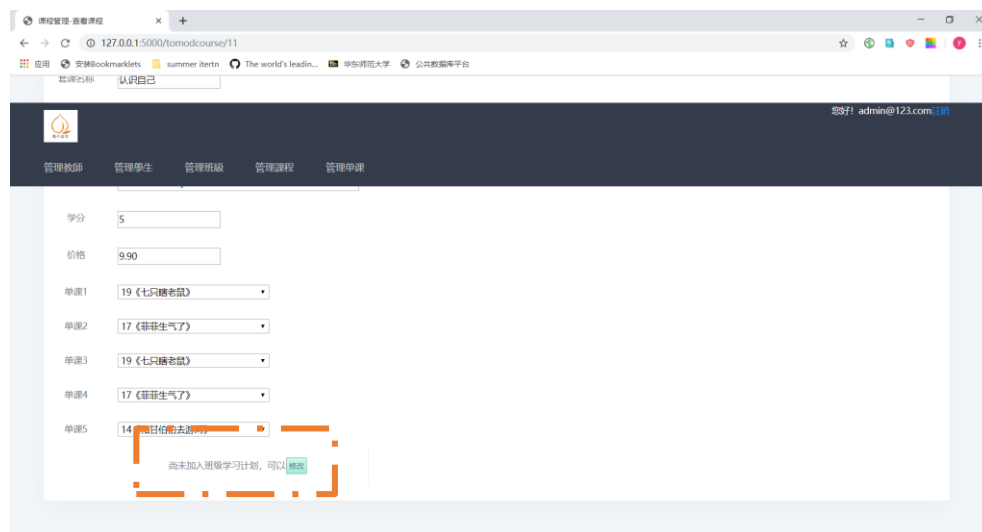


图 5-15 管理员修改课程界面

16) 管理员管理单课主界面, 显示单课列表, 包括课程封面、课程名、课程简介等信息, 可进行添加、修改、删除操作。管理员管理单课主界面如图 5-16 所示。



图 5-16 管理员管理单课主界面

- 17) 管理员添加单课界面，添加课程题目时，由单课的题型（客观或者主观）实现动态级联操作。管理员添加单课界面如图 5-17 所示。

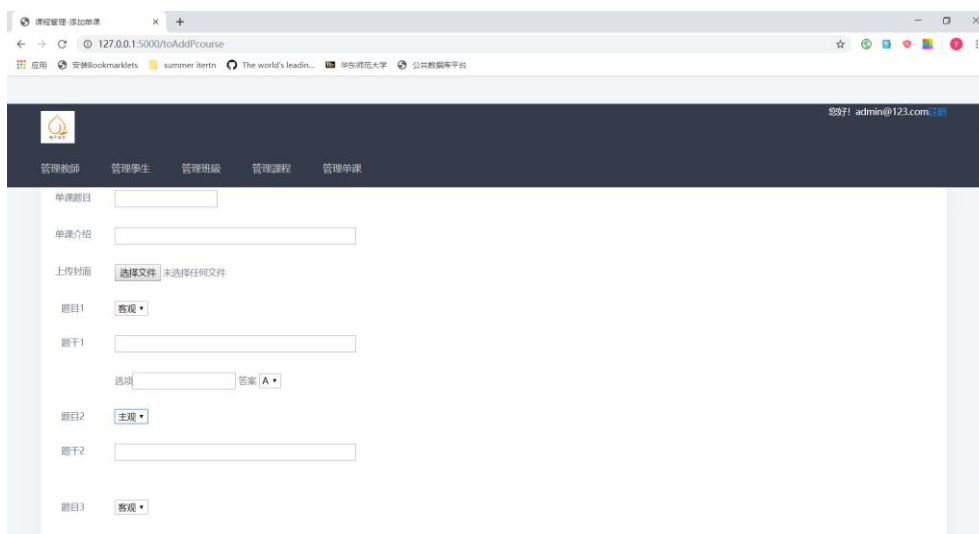


图 5-17 管理员添加单课界面

- 18) 学生登录界面，模仿微信扫码登录过程，遂没有输入密码操作。学生登录界面如图 5-18 所示。

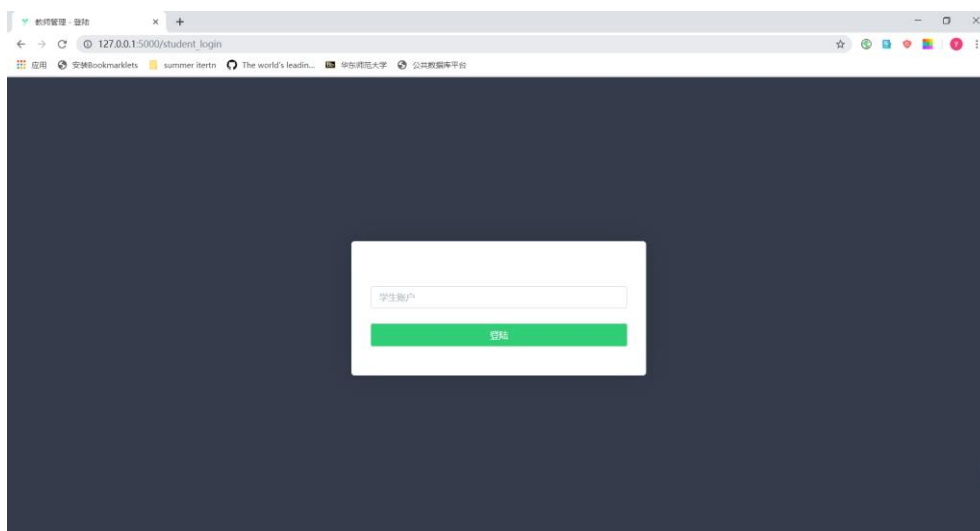


图 5-18 学生登录界面

19) 学生个人中心界面，显示学生基本信息和课程完成情况，可以实现账户充值、课程学习、作业完成等操作。学生个人中心界面如图 5-19 所示。

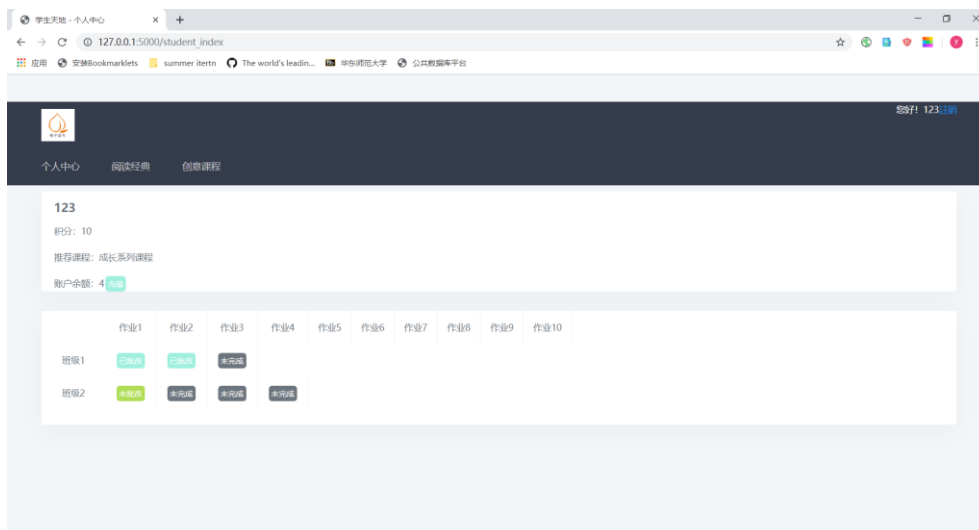


图 5-19 学生个人中心界面

20) 学生可对账户进行充值操作。学生充值界面如图 5-20 所示。

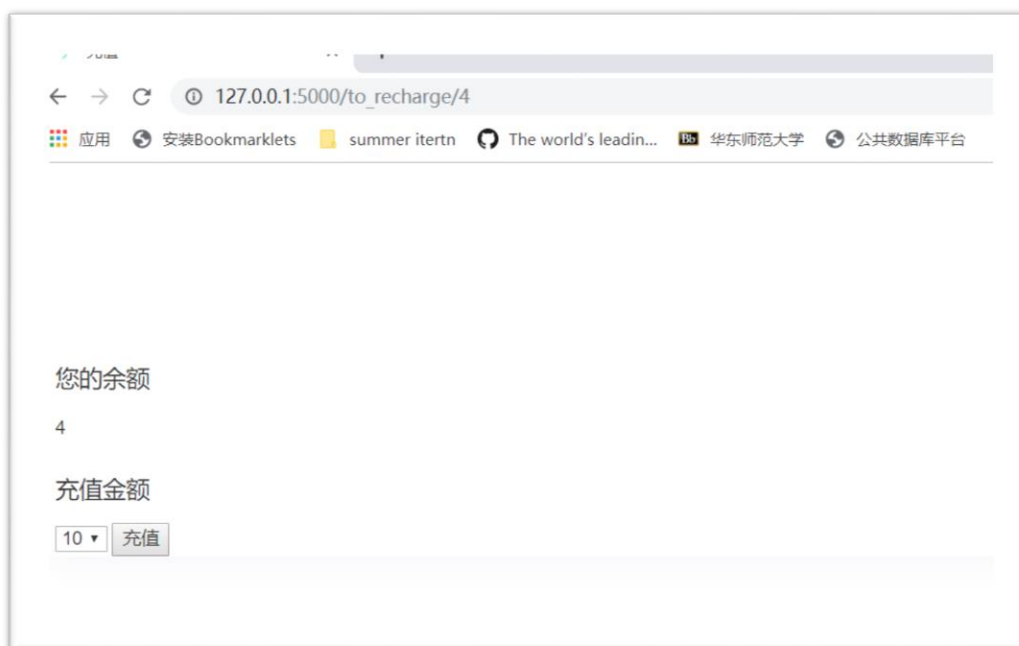


图 5-20 学生充值界面

21) 学生课程学习界面，对于未完成作业的课程，显示作业提交框。学生课程学习界面-未完成的课程界面如图 5-21 所示。

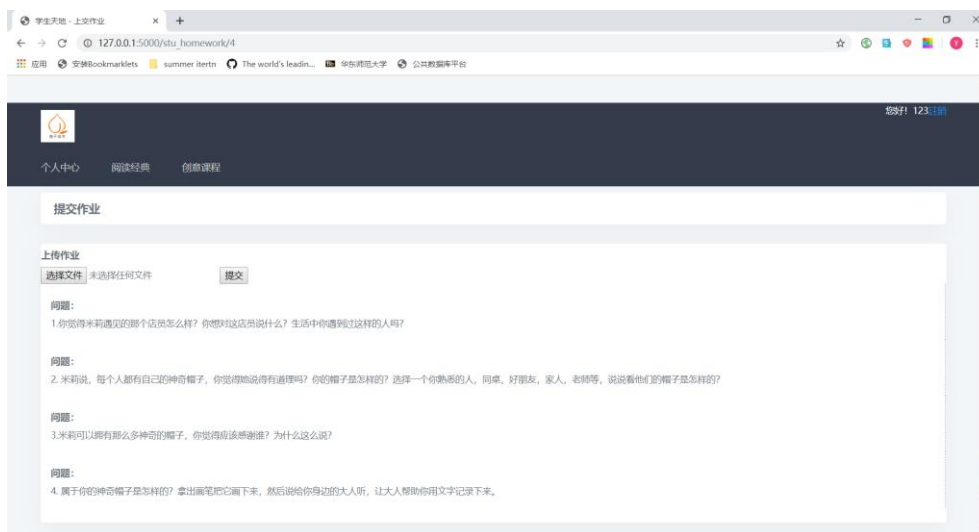


图 5-21 学生课程学习界面-未完成的课程

22) 学生课程学习界面，对已完成作业的课程，可进行课程回顾、查看教师评语等操作。学生课程学习界面-已完成的课程界面如图 5-22 所示。

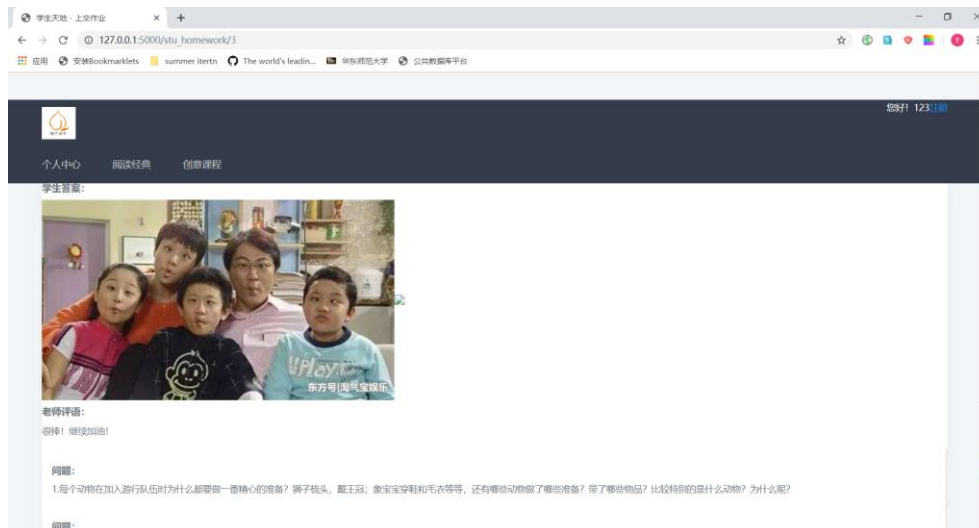


图 5-22 学生课程学习界面-已完成的课程界面

23) 学生自由阅读单课主界面，显示单课信息。对于已经学习过的单课（包括直接学习的单课和跟随班级学习的单课），显示阅读按钮。对于未购买的单课需先购买再阅读（价格为 0.1），显示购买按钮。学生自由阅读单课主界面如图 5-23 所示。

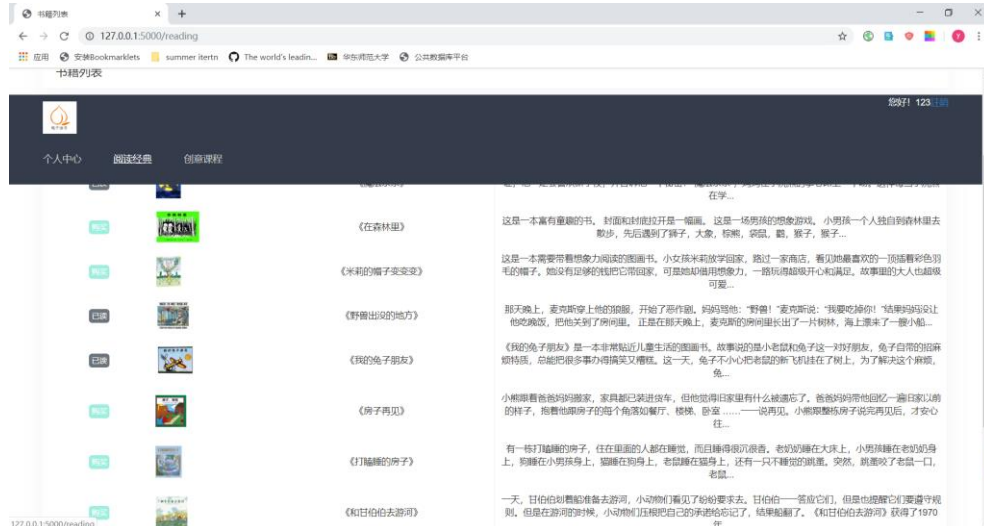


图 5-23 学生自由阅读单课主界面

24) 学生阅读界面如图 5-24 所示。

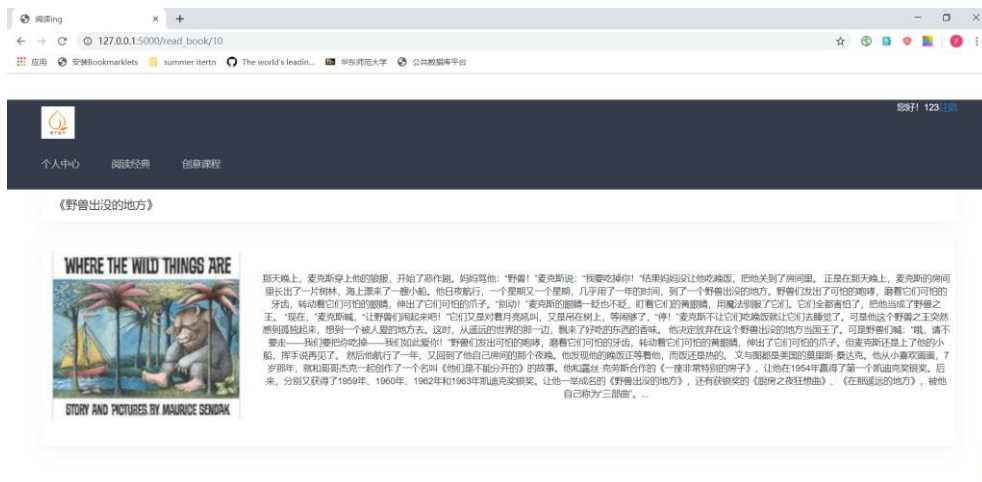


图 5-24 学生阅读界面

25) 学生跟随班级（课程）学习主界面，显示学生可加入的班级信息，若可用余额（余额+0.1*学分）小于课程价格，则需要先充值，再进行操作。学生跟随班级（课程）学习主界面如图 5-25 所示。

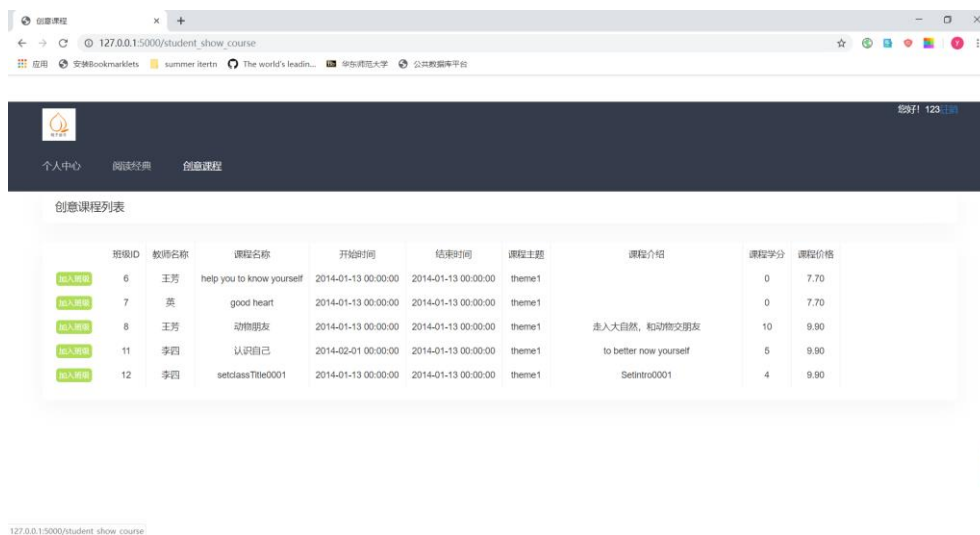


图 5-25 学生跟随班级（课程）学习主界面

六. 心得体会

1. 遇到的困难和解决方法

- 在一开始的测试数据中，为数据的方便表示，均采用英文字符串。但在实际的数据插入过程中，均为中文字符。在使用 pymysql 连接数据库时，发现传输的中文字符串都为问号，显示不出来。初以为是 Python 编码问题，后在排除这种情况后。才发现是数据库连接问题。后使用如下方式解决问题。

```
pymysql.connect('localhost',username ,password , 'readingmanagement',charset = 'utf8')
```

- 数据数据的预处理。在进行实际数据的输入时，才发现存在各种的格式问题及不匹配。例如，在进行问题集的插入过程中，发现原始数据并未对题目进行主观，客观分类，也没有将题干、选项和答案进行区分。所以，数据预处理是我们在实际建库的非常关键的一步。
- Mysql 中 cursor 使用。在对 select 单条结果进行 into 操作有较好的掌握之后，发现实际需要对 select 的结果集进行操作，无法使用 select into。所以，参考课程用例中进行循环判断，如下：

```
DECLARE exit HANDLER FOR NOT FOUND CLOSE cur1;
```

后发现此声明适用于 procedure，在循环结束后直接跳出，无法返回值。后经查阅资料，修改此条语句。修改结果见第四部分程序代码。

- 分页处理。实际数据中单课数据量很多，故无法在一页中显示，不方便且不美观。后采用 Pagination 实现分页。
- 遇到 Mysql 语句报错

```
1248 - Every derived table must have its own alias
```

经过查询得知此报错的意思为每一个派生出来的表都必须有一个自己的别名。后修改。并且学习到 Union 和 union all 的差别，前者会去除掉相同项。

- JavaScript 基础薄弱。无法很好实现联动查询，联动变化效果。

2. 收获与体会

经过大作业的实践，才发现实践才是最快最有效的学习方式。在完成大作业

的过程中，我首先学会到了如何将需求转换成 ER 图再转换成数据库中的关系模式；然后进一步深化了建表的关系模式设计方法，sql 查询语句的使用方法，然后将这学期学习的触发器、存储过程、函数等进行了实践；同时在设计的过程，知道了如何用这些方法（视图、触发器、存储过程、函数等）进行代码复用以及边界的判断。通过数据库实践大作业，我在对数据库的知识有了进一步的了解和掌握的同时，我也对如何通过网页实现一个完整的系统有了进一步的了解，关于 HTML，CSS，JavaScript，flask，Python 等知识。

数据库实践大作业最重要的是提高了我发现问题和解决问题的能力。从脑子中的一个 idea，变成眼前功能丰富的网站，每一步都会碰到问题并且需要我们一点点改进使它更完美，回想一路走来的点点滴滴，感慨万千。感谢老师和助教的指导！