**上海电力学院**

**软件设计实训**



题　　目：　 学生信息管理系统

院　　系：　 计算机科学与技术学院

专业年级：　 软件工程2011级

学生姓名：　 　学号：　20113092

指导教师：　　李海明、刘小洛、张超、杨吟冬

2014年7月 5日

目录

[1前言 3](#_Toc392639092)

[1.1目的 3](#_Toc392639093)

[1.2背景 3](#_Toc392639094)

[1.2.1开发的软件系统的名称 3](#_Toc392639095)

[1.2.2项目的提出者 4](#_Toc392639096)

[1.2.3项目的开发者 4](#_Toc392639097)

[1.2.4项目的用户 4](#_Toc392639098)

[1.2.5与其他软件、系统的关系： 4](#_Toc392639099)

[1.3定义 4](#_Toc392639100)

[1.4参考资料 5](#_Toc392639101)

[2系统分析 6](#_Toc392639102)

[2.1可行性分析 6](#_Toc392639103)

[2.1.1法律方面的可行性 6](#_Toc392639104)

[2.2.2使用方面的可行性 6](#_Toc392639105)

[2.2.3技术条件方面的可行性 6](#_Toc392639106)

[2.2.4经济条件方面的可行性 7](#_Toc392639107)

[2.2需求分析 7](#_Toc392639108)

[2.2.1 开发目标及作用范围 7](#_Toc392639109)

[2.2.2 系统基本情况及模块介绍 7](#_Toc392639110)

[2.2.3用户功能介绍 9](#_Toc392639111)

[2.2.4用户功能介绍 9](#_Toc392639112)

[3系统设计 11](#_Toc392639113)

[3.1项目规划 11](#_Toc392639114)

[3.1.1任务分解及人员分工 11](#_Toc392639115)

[3.1.2项目进度 11](#_Toc392639116)

[3.2系统流程图 12](#_Toc392639117)

[3.3处理流程图 12](#_Toc392639118)

[4系统详细设计 13](#_Toc392639119)

[4.1模块设计 13](#_Toc392639120)

[4.2数据库设计 18](#_Toc392639121)

[4.2.1概念结构设计 18](#_Toc392639122)

[4.2.2物理结构设计 18](#_Toc392639123)

[5系统实现及其关键代码 20](#_Toc392639124)

[5.1登录模块 20](#_Toc392639125)

[5.2学生管理模块 22](#_Toc392639126)

[5.2.1查询学生信息 22](#_Toc392639127)

[5.2.2增加学生信息 23](#_Toc392639128)

[5.2.3修改课程信息 24](#_Toc392639129)

[5.2.4删除课程信息 26](#_Toc392639130)

[6总结 27](#_Toc392639131)

# 1前言

## 1.1目的

学生信息管理系统，是针对学校人事处的大量业务处理工作而开发的管理软件，是典型的管理信息系统。它是一个教育单位不可缺少的部分,它的内容对于学校的决策者和管理者来说都至关重要,它能够为用户提供充足的信息和快捷的查询手段。能有效的帮助学校和老师掌握学生的情况，为学生提供成绩跟课程方面的查询。

在传统模式下利用人工进行学生信息管理，存在着较多的缺点，如：效率底，保密性差，时间一长将产生大量的文件和数据，更不便于查找，更新，维护等。诸如这些情况，令学校管理者对学生的信息管理带来了很大困难，严重影响了教育工作者的工作效率。

随着科学技术的不断提高,计算机科学日渐成熟, 使用日趋成熟的计算机技术来代替传统的人工模式，来实现学生信息的现代化管理，其强大的功能已为人们深刻认识,它已进入人类社会的各个领域并发挥着越来越重要的作用。作为计算机应用的一部分,使用计算机对学生信息进行管理,具有着手工管理所无法比拟的优点.例如:检索迅速、查找方便、易修改、可靠性高、存储量大、数据处理快捷、保密性好、寿命长、成本低、便于打印等。这些优点能够极大地提高学生信息管理的效率,也是学校的科学化、正规化管理,与世界接轨的重要条件。用计算机制作的学生信息管理系统还可以通过功能强大的Internet网及时的向学生的家长传递该生在校的最新成绩，有助于学校与家长的信息互动，为更好地把握学校的教育方针一定的促进作用！因此，开发这样一套管理软件成为很有必要的事情。

## 1.2背景

## 1.2.1开发的软件系统的名称

学生信息管理系统

## 1.2.2项目的提出者

上海电力学院计算机科学与技术学院

## 1.2.3项目的开发者

凤翔、陈龙龙

## 1.2.4项目的用户

管理员

## 1.2.5与其他软件、系统的关系：

系统：Windows7

数据库：MongoDB

IDE：Webstorm8

版本控制软件：Git  
编辑器：Sublime Text 3

Google查资料用代理工具：Goagent

## 1.3定义

学生信息管理系统（Student Information Management System）：为用户提供充足的信息和快捷的查询手段。能有效的帮助学校和老师掌握学生的情况，为学生提供成绩跟课程方面的查询。

Node.js： 是一个基于Chrome JavaScript 运行时建立的一个平台， 用来方便地搭建快速的， 易于扩展的网络应用· Node.js 借助事件驱动， 非阻塞I/O 模型变得轻量和高效， 非常适合 运行在分布式设备 的 数据密集型 的实时应用

MongoDB：一个基于分布式文件存储的数据库。由C++语言编写。旨在为WEB应用提供可扩展的高性能数据存储解决方案。

CoffeeScript：CoffeeScript是一套JavaScript的转译语言。受到Ruby、Python与Haskell等语言的启发，CoffeeScript增强了JavaScript的简洁性与可读性。此外，CoffeeScript也新增了更复杂的功能，例如列表内涵（List comprehension）、模式匹配（Pattern matching）等。

## 1.4参考资料

[Node.js v0.10.29 Manual & Documentation](http://nodejs.org/api/)

[The MongoDB 2.6 Manual](http://docs.mongodb.org/manual/)

[Mongoose Schemas v3.8.12](http://mongoosejs.com/docs/guide.html)

[NODE.JS UNIT TESTING BY @朴灵](http://html5ify.com/unittesting/slides/index.html#/)

[Express 4.x API Reference](http://expressjs.com/4x/api.html)

[How to use templates with EJS]( https://code.google.com/p/embeddedjavascript/wiki/Templates)

[Bootstrap 中文文档](http://v3.bootcss.com/)

[LESS 语法](http://www.bootcss.com/p/lesscss/)

[Pro Git](http://git-scm.com/book/zh/)

[初识 mocha in NodeJS](http://cnodejs.org/topic/516526766d38277306c7d277)

[Mocha - the fun, simple, flexible JavaScript test framework](http://visionmedia.github.io/mocha/)

[Wikipedia](http://zh.wikipedia.org/)

[Stackoverflow](http://stackoverflow.com/)

# 2系统分析

## 2.1可行性分析

### 2.1.1法律方面的可行性

本项目为小组成员自主开发，使用正版软件，故在法律方面可行

### 2.2.2使用方面的可行性

本系统开发秉承着操作简洁，界面友好，响应式布局的原则。所以只要使用者具有计算机基础即可在一段时间的熟悉后便能熟练使用，故在使用方面可行。

### 2.2.3技术条件方面的可行性

本系统全部使用开源技术，可以在Windows、OS X、Linux全平台通用。

* 服务器环境：nodejs 0.10.29
* 服务器语言：CoffeeScript
* MVC框架：Express 4
* 数据库：MongoDB
* ORM框架：Mongoose
* 模板引擎：ejs
* 单元测试框架：Mocha
* 前端页面语言：HTML5
* 层叠样式表：CSS3、LESS
* 页面脚本语言：Javascript、CoffeeScript
* 页面开源框架：JQuary、Bootstrap
* 版本控制软件：Git
* 版本控制服务：Github
* 数据库托管平台：Mongohq
* IDE：JetBrain Webstorm
* 编辑器：Sublime Text 3
* Google查资料用代理工具：Goagent

利用现有技术基本可以达到目标，另外开发期间还有老师指导，故技术方面可行。

### 2.2.4经济条件方面的可行性

本次项目开发小组成员为2人，开发时间9天，期间均使用自己的计算机，无额外投入，故经济方面可行

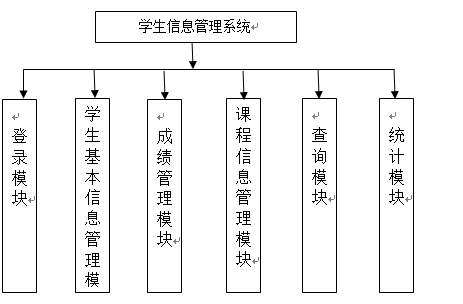
## 2.2需求分析

### 2.2.1 开发目标及作用范围

对于开发的学生信息管理系统，主要实现以下功能：对学生的基本信息、课程安排、成绩、选课、密码，进行基本的管理。包括：学生的基本信息和学科成绩的存储、输入、删除、修改、查询。老师可以录入成绩、查询学生课程信息及修改密码、编辑学生的信息。管理员：课程管理、学生管理、修改密码及查询。

### 2.2.2 系统基本情况及模块介绍

本学生信息管理系统共包含6个模块，分别为登录模块、学生基本信息管理模块、成绩管理模块、课程信息管理模块、查询模块、统计模块，模块结构图如下图所示。



每一个模块都有其固有的功能，下面将每个模块的基本功能介绍如下：

（1）学生基本信息管理模块：对学生的基本信息进行综合管理，可以添加、修改及删除学生的基本信息。

（2）成绩管理模块：对学生所选课程的成绩信息进行综合管理，可以添加、修改及删除基本信息。

（3）课程信息管理模块：对课程信息进行综合管理，可以添加、修改及删除课程的基本信息。

（4）查询模块：

1）学生基本信息的查询：根据学生的已知条件来查询学生的详细信息，对姓名、学号、班级、系名等支持模糊查询。

2）课程基本信息的查询：根据课程的信息来查询课程的详细信息。

3）查询学生的选课情况、查询学生所选课程的成绩。

（5）统计模块：根据不同课程对学生成绩进行统计，求平均分；根据不同的分数区间进行人数统计等。

### 2.2.3用户功能介绍

* 管理员



### 2.2.4用户功能介绍

每一个模块中的用例均有其特有的功能描述，因时间有限，下面将以学生基本信息管理模块中的添加学生信息用例和修改学生信息用例为例，进行用例分析。

|  |  |
| --- | --- |
| 用例名称 | 添加学生信息 |
| 用例编号 | 1 |
| 执行者 | 管理员 |
| 用例简述 | 该用例描述了管理员如何为系统添加学生的过程 |
| 前置条件 | 1. 管理员登录系统成功 2. 输入正确的学生信息 |
| 后置条件 | 学生添加成功 |
| 基本流程 | 1. 管理员登录 2. 系统显示登录后首页 3. 管理员点击【学生信息维护】按钮 4. 系统显示学生列表页面 5. 单击【添加】按钮，系统显示学生添加界面 6. 输入正确的学生信息后，点击【确认】 7. 系统提示学生添加成功 8. 系统返回学生列表界面 |
| 替代流程 | 任何时刻发生以下状况，添加用户信息用例将失败：   1. 管理员未输入其中任一一项学生信息 2. 系统提示“XXX信息未输入，请返回重新输入” 3. 管理员输入错误的用户信息 4. 系统提示“XXX信息输入有误，请重新输入” |

|  |  |
| --- | --- |
| 用例名称 | 修改学生信息 |
| 用例编号 | 3 |
| 执行者 | 管理员 |
| 用例简述 | 该用例描述了管理员如何为修改学生信息的过程 |
| 前置条件 | 1. 管理员登录系统成功 2. 输入正确的学生信息 |
| 后置条件 | 学生信息修改成功 |
| 基本流程 | 1. 管理员登录 2. 系统显示登录后首页 3. 管理员点击【学生信息维护】按钮 4. 系统显示学生列表页面 5. 单击任意一名学生，系统显示该学生信息界面 6. 单击【编辑】按钮，系统进入编辑界面 7. 输入正确的学生信息后，点击【确认】 8. 系统提示学生信息修改成功 9. 系统返回学生列表界面 |
| 替代流程 | 任何时刻发生以下状况，修改学生信息用例将失败：   1. 管理员输入错误的学生信息 2. 系统提示“XXX信息输入有误，请重新输入” |

# 3系统设计

## 3.1项目规划

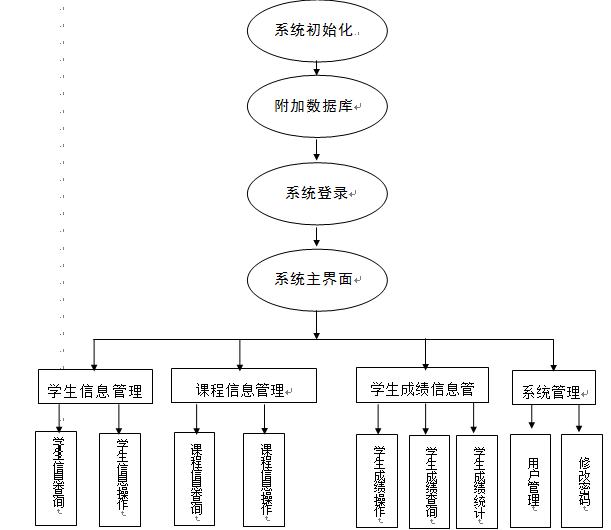
### 3.1.1任务分解及人员分工

|  |  |
| --- | --- |
| 任务 | 负责人 |
| 可行性分析报告 | 陈龙龙 |
| 项目开发计划 | 凤翔 |
| 需求规格分析说明书 | 陈龙龙 |
| 项目概要设计规格说明书 | 凤翔 |
| 项目详细设计规格说明书 | 陈龙龙 |
| 项目架构 | 凤翔 |
| 项目界面 | 陈龙龙 |
| 项目学生、成绩、统计模块编写 | 凤翔 |
| 项目课程模块编写 | 陈龙龙 |
| 项目测试 | 凤翔 |
| 项目总结 | 陈龙龙 |

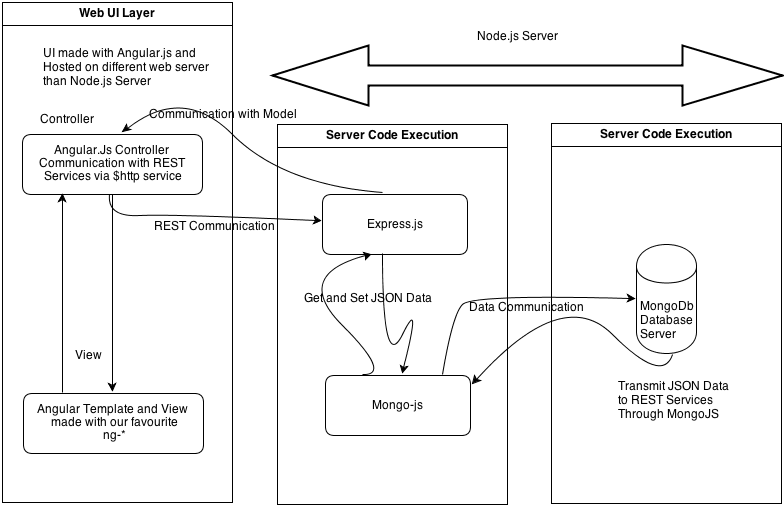
### 3.1.2项目进度

|  |  |
| --- | --- |
| 时间 | 预期工程进度 |
| 第一天 | 可行性分析、需求分析、项目架构 |
| 第二天 | 数据库设计、数据库连接 |
| 第三天 | 概要设计、详细设计 |
| 第四天 | 编码 |
| 第五天 | 编码 |
| 第六天 | 编码 |
| 第七天 | 测试 |
| 第八天 | debug |
| 第九天 | 撰写总结报告 |

## 3.2系统流程图



## 3.3处理流程图



# 4系统详细设计

## 4.1模块设计

* 课程信息管理模块

|  |  |
| --- | --- |
| 用例名称 | 添加课程信息 |
| 用例编号 | 1 |
| 执行者 | 管理员 |
| 用例简述 | 该用例描述了管理员如何为系统添加课程的过程 |
| 前置条件 | 1. 管理员登录系统成功 2. 输入正确的课程信息 |
| 后置条件 | 学生添加成功 |
| 基本流程 | 1. 管理员登录 2. 系统显示登录后首页 3. 管理员点击【课程信息维护】按钮 4. 系统显示课程列表页面 5. 单击【添加】按钮，系统显示课程添加界面 6. 输入正确的课程信息后，点击【确认】 7. 系统提示课程添加成功 8. 系统返回课程列表界面 |
| 替代流程 | 任何时刻发生以下状况，添加用户信息用例将失败：   1. 管理员未输入其中任一一项课程信息 2. 系统提示“XXX信息未输入，请返回重新输入” 3. 管理员输入错误的用户信息 4. 系统提示“XXX信息输入有误，请重新输入” |

|  |  |
| --- | --- |
| 用例名称 | 修改课程信息 |
| 用例编号 | 2 |
| 执行者 | 管理员 |
| 用例简述 | 该用例描述了管理员如何为修改课程信息的过程 |
| 前置条件 | 1. 管理员登录系统成功 2. 输入正确的课程信息 |
| 后置条件 | 课程信息修改成功 |
| 基本流程 | 1. 管理员登录 2. 系统显示登录后首页 3. 管理员点击【课程信息维护】按钮 4. 系统显示课程列表页面 5. 单击任意一名课程，系统显示该课程信息界面 6. 单击【编辑】按钮，系统进入编辑界面 7. 输入正确的课程信息后，点击【确认】 8. 系统提示课程信息修改成功 9. 系统返回课程列表界面 |
| 替代流程 | 任何时刻发生以下状况，修改课程信息用例将失败：   1. 管理员输入错误的课程信息 2. 系统提示“XXX信息输入有误，请重新输入” |

|  |  |
| --- | --- |
| 用例名称 | 删除课程 |
| 用例编号 | 3 |
| 执行者 | 管理员 |
| 用例简述 | 该用例描述了管理员如何删除课程的过程 |
| 前置条件 | 1. 管理员登录系统成功 |
| 后置条件 | 课程删除成功 |
| 基本流程 | 1. 管理员登录 2. 系统显示登录后首页 3. 管理员点击【课程信息维护】按钮 4. 系统显示课程列表页面 5. 单击任意一名课程后的【删除】按钮 6. 系统提示课程删除成功 7. 系统返回课程列表界面 |
| 替代流程 | 无 |

* 成绩管理模块

|  |  |
| --- | --- |
| 用例名称 | 录入成绩信息 |
| 用例编号 | 1 |
| 执行者 | 管理员 |
| 用例简述 | 该用例描述了管理员如何为学生录入成绩的过程 |
| 前置条件 | 1.管理员登录系统成功 |
| 后置条件 | 成绩录入成功 |
| 基本流程 | 1. 管理员登录 2. 系统显示登录后首页 3. 管理员点击【学生成绩维护】按钮 4. 系统显示学生成绩列表页面 5. 单击【录入成绩】按钮，系统显示录入成绩界面 6. 录入成绩后，点击【确认】 7. 系统提示成绩录入成功 8. 系统返回学生列表界面 |
| 替代流程 | 任何时刻发生以下状况，录入成绩将失败：  管理员未录入成绩信息  1）系统提示“XXX成绩未输入，请返回重新输入”   1. 系统提示“XXX成绩输入有误，请重新输入” |

|  |  |
| --- | --- |
| 用例名称 | 修改成绩信息 |
| 用例编号 | 2 |
| 执行者 | 管理员 |
| 用例简述 | 该用例描述了管理员如何修改学生成绩信息的过程 |
| 前置条件 | 1.管理员登录系统成功 |
| 后置条件 | 课程信息修改成功 |
| 基本流程 | 1. 管理员登录 2. 系统显示登录后首页 3. 管理员点击【学生成绩维护】按钮 4. 系统显示学生成绩列表页面 5. 单击任意一名学生的【修改成绩】按钮，系统显示该修改成绩界面 6. 输入正确的成绩信息后，点击【确认】 7. 系统提示成绩修改成功 8. 系统返回学生成绩列表界面 |
| 替代流程 | 任何时刻发生以下状况，修改课程信息用例将失败：   1. 管理员输入错误的成绩信息 2. 系统提示“XXX成绩输入有误，请重新输入” |

|  |  |
| --- | --- |
| 用例名称 | 删除成绩 |
| 用例编号 | 3 |
| 执行者 | 管理员 |
| 用例简述 | 该用例描述了管理员如何删除成绩信息的过程 |
| 前置条件 | 管理员登录系统成功 |
| 后置条件 | 成绩删除成功 |
| 基本流程 | 1. 管理员登录 2. 系统显示登录后首页 3. 管理员点击【学生成绩维护】按钮 4. 系统显示学生成绩列表页面 5. 单击任意一项成绩后的【删除】按钮 6. 系统提示成绩删除成功 7. 系统返回学生成绩列表界面 |
| 替代流程 | 无 |

* 统计模块

|  |  |
| --- | --- |
| 用例名称 | 按成绩区间统计 |
| 用例编号 | 1 |
| 执行者 | 管理员 |
| 用例简述 | 该用例描述了管理员如何按成绩区间统计学生成绩的过程 |
| 前置条件 | 1.管理员登录系统成功 |
| 后置条件 | 统计成功 |
| 基本流程 | 1. 管理员登录 2. 系统显示登录后首页 3. 管理员点击【学生成绩维护】按钮 4. 系统显示学生成绩列表页面 5. 单击【按成绩区间统计】按钮，系统显示统计界面 6. 输入正确的成绩区间后，点击【确认】 7. 系统提示成绩统计完成 8. 系统返回统计结果页面 |
| 替代流程 | 任何时刻发生以下状况，修改课程信息用例将失败：   1. 管理员输入错误的成绩区间 2. 系统提示“输入有误，请重新输入” |

* 查询模块

|  |  |
| --- | --- |
| 用例名称 | 查看学生信息 |
| 用例编号 | 1 |
| 执行者 | 管理员 |
| 用例简述 | 该用例描述了管理员如何查看学生信息的过程 |
| 前置条件 | 管理员登录系统成功 |
| 后置条件 | 学生信息查看成功 |
| 基本流程 | 1. 管理员登录 2. 系统显示登录后首页 3. 管理员点击【学生信息维护】按钮 4. 系统显示学生列表页面 5. 单击任意一名学生，系统显示学生信息界面 |
| 替代流程 | 任何时刻发生以下状况，查看学生信息用例将失败：   1. 管理员点击错误的按钮 2. 系统停留在原界面不跳转 3. 系统跳转至其它页面 |

|  |  |
| --- | --- |
| 用例名称 | 查看课程信息 |
| 用例编号 | 2 |
| 执行者 | 管理员 |
| 用例简述 | 该用例描述了管理员如何查看课程信息的过程 |
| 前置条件 | 管理员登录系统成功 |
| 后置条件 | 课程信息查看成功 |
| 基本流程 | 1. 管理员登录 2. 系统显示登录后首页 3. 管理员点击【课程信息维护】按钮 4. 系统显示课程列表页面 5. 单击任意一名课程，系统显示课程信息界面 |
| 替代流程 | 任何时刻发生以下状况，查看课程信息用例将失败：  管理员点击错误的按钮   1. 系统停留在原界面不跳转 2. 系统跳转至其它页面 |

|  |  |
| --- | --- |
| 用例名称 | 查看学生选课信息 |
| 用例编号 | 3 |
| 执行者 | 管理员 |
| 用例简述 | 该用例描述了管理员如何查看学生选课信息的过程 |
| 前置条件 | 1.管理员登录系统成功  2.管理员输入正确学生信息 |
| 后置条件 | 学生选课信息查看成功 |
| 基本流程 | 1. 管理员登录 2. 系统显示登录后首页 3. 管理员点击【学生信息维护】按钮 4. 系统显示学生列表页面 5. 管理员点击任意一名学生后的【选课情况】按钮 6. 系统显示该学生选课信息界面 |
| 替代流程 | 任何时刻发生以下状况，查看课程信息用例将失败：  管理员点击错误的按钮   1. 系统停留在原界面不跳转 2. 系统跳转至其它页面 |

|  |  |
| --- | --- |
| 用例名称 | 模糊查询 |
| 用例编号 | 4 |
| 执行者 | 管理员 |
| 用例简述 | 该用例描述了管理员根据学生的已知条件来查询学生的详细信息 |
| 前置条件 | 管理员登录系统成功 |
| 后置条件 | 学生信息查看成功 |
| 基本流程 | 1. 管理员登录 2. 系统显示登录后首页 3. 管理员点击【学生信息维护】按钮 4. 管理员点击【模糊查询】按钮 5. 管理员输入姓名、学号、班级、系名等已知信息，点击【确定】按钮进行模糊查询 6. 系统显示查询到的学生信息界面 |
| 替代流程 | 任何时刻发生以下状况，查看学生信息用例将失败：  管理员点击错误的按钮   1. 系统停留在原界面不跳转 2. 系统跳转至其它页面 |

## 4.2数据库设计

### 4.2.1概念结构设计



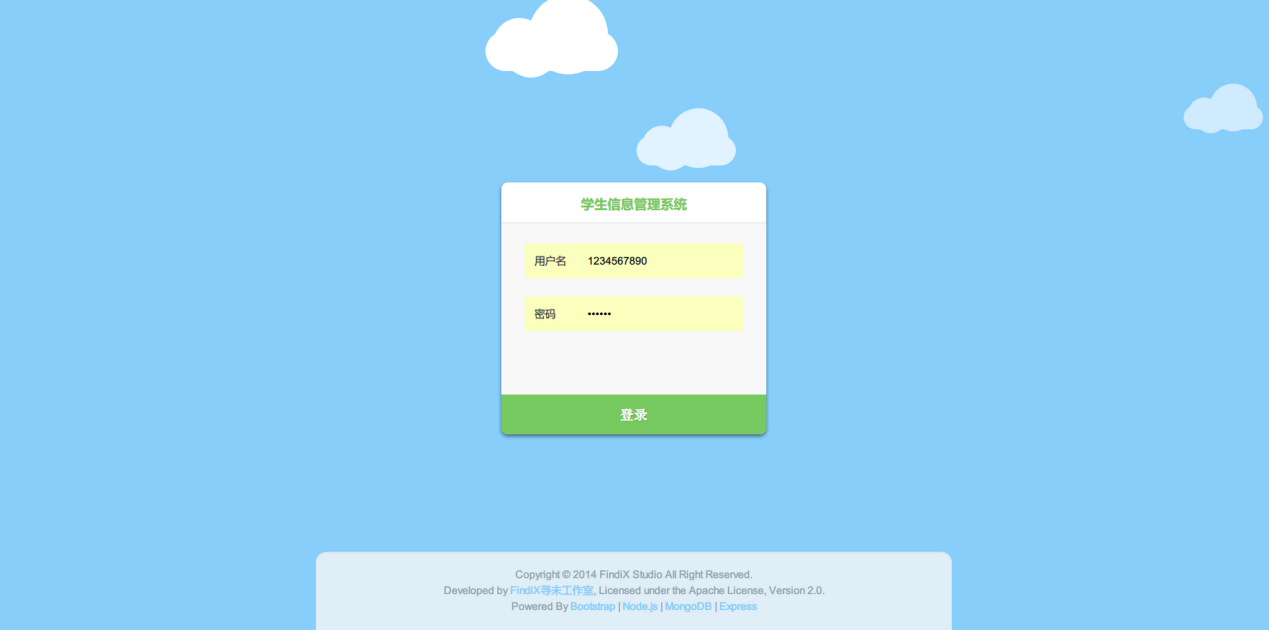
### 4.2.2物理结构设计

学生信息管理系统部分E-R图如下图所示。



# 5系统实现及其关键代码

## 5.1登录模块



用户访问网站，判断是否登录，如未登录，则渲染显示登录页面，否则进入管理系统。

|  |
| --- |
| router.get "/", (req, res) ->  # 判断是否登录  if req.session.username?  res.render 'index',  username: req.session.username  res.redirect '/student'  # 如果没有登录，渲染登录页面  else res.render "login",  status: '' |

用户输入工号和密码，点击登陆，开始前端和后台数据库两层验证，首先是前端验证。

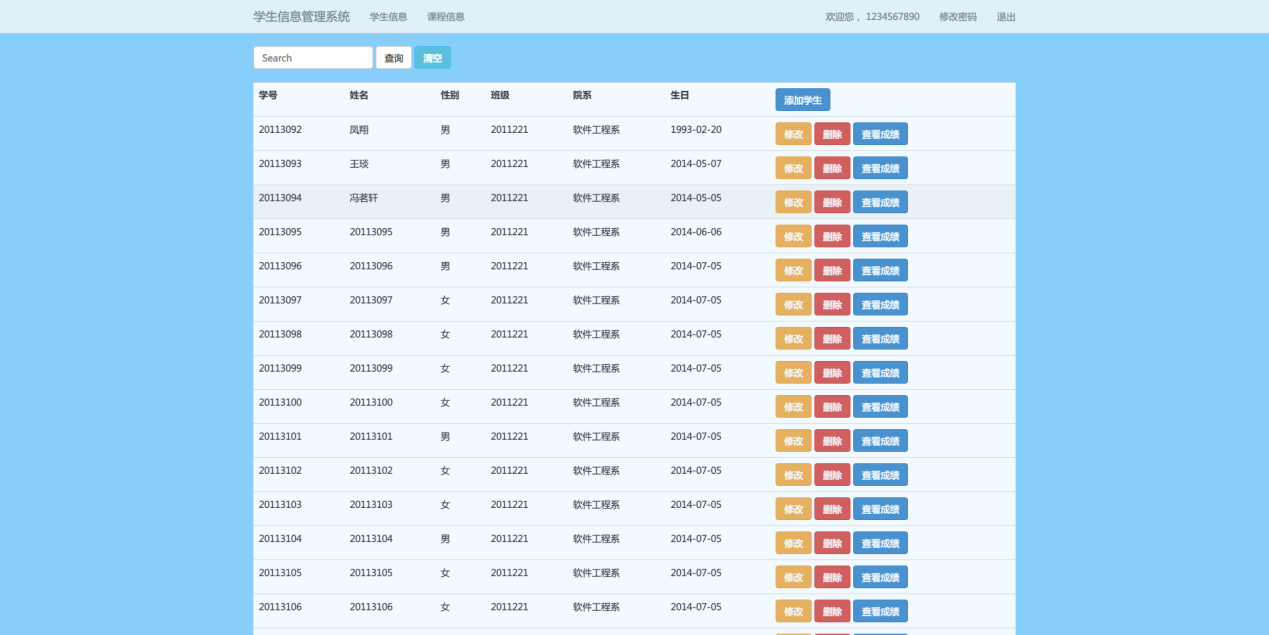
|  |
| --- |
| /\*\*  \* 表单验证  \*/  function checkForm() {  //对函数的第一个参数进行判断  if (arguments[0]) {  //获取当前表单  var form = arguments[0];  } else {  return false;  }  //验证规则汇总  var id = new RegExp(/[0-9]{10}/);//匹配数字10次，用于工号匹配  var password = new RegExp(/\w{6,16}/);//字母6到16次，用于密码匹配  //用户名username验证  if (form.username != undefined && !id.test(form.username.value)) {  //不符合匹配规则  onClick($(form.username)[0], '工号不规范，应为10位数字');  //阻止页面跳转  return false;  }  //密码password验证  if (form.password != undefined && !password.test(form.password.value)) {  //不符合匹配规则  onClick($(form.password)[0], '新的密码不规范，应为6~16位字母和数字');  //阻止页面跳转  return false;  }  } |

如果前端验证通过，进行后台登录验证

|  |
| --- |
| router.post '/Login', (req, res) ->  username = req.body.username  password = req.body.password  # 查询用户名和密码  user.find  username: username  password: password  , (err, data) -> #回调函数  if data.length is 0 #长度为0表示没有查询到结果，即该用户名和密码不存在  then res.render 'login', #渲染页面，加入错误信息  status: '<div class="option\_result">用户名或密码错误</div>'  else  req.session.username = username #登录成功，存储session  res.redirect "/" |

## 5.2学生管理模块

### 5.2.1查询学生信息

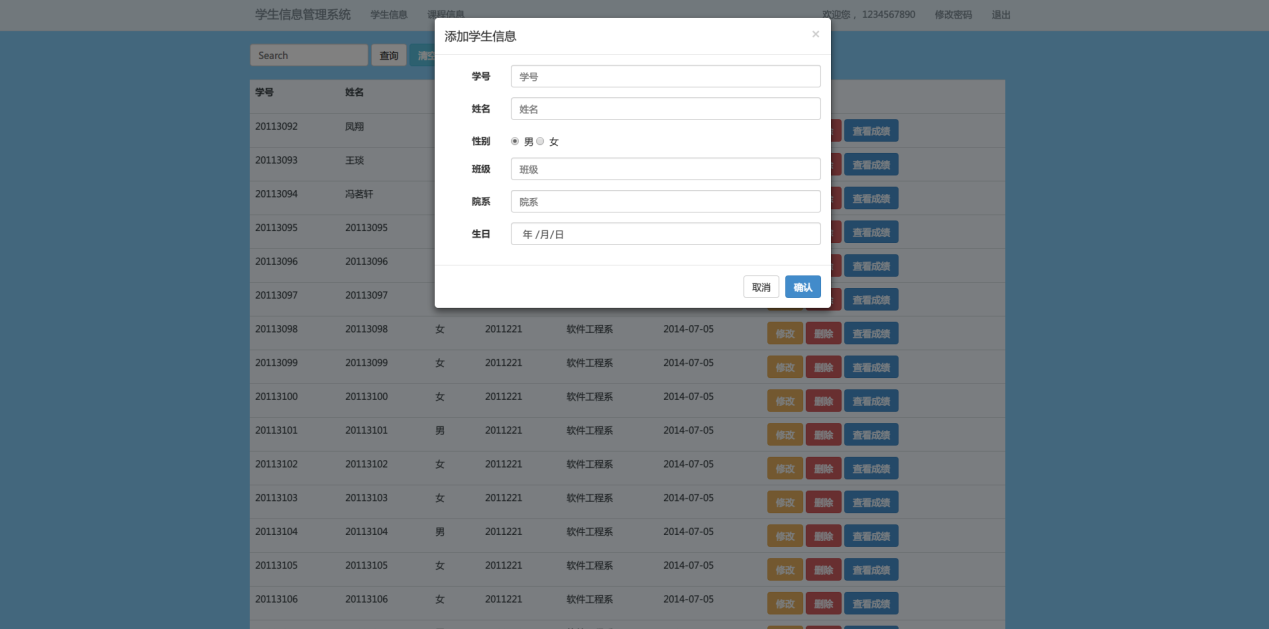


登录成功后，首先进入学生管理页面，此时没有查询条件，所以页面上显示所有信息。

当用户在搜索框中输入信息，并点击查询，即会重新渲染页面，此时query参数不为空，find函数开始根据正则表达式匹配内容，最终渲染页面输出满足匹配的学生信息。

|  |
| --- |
| router.get "/", (req, res) ->  # 登录验证  unless req.session.username?  res.redirect '/'  # 获取查询参数  query = req.query.query  # 查询  studentModel.find("$or": [  {sname: new RegExp(query)}  {sno: new RegExp(query)}  {gender: new RegExp(query)}  {class: new RegExp(query)}  {department: new RegExp(query)}  ],null, {sort:  'sno': 1  },# 以学生学号升序排序  (err, data)->  if err?  console.log(err)  res.render 'student',  students: data  username: req.session.username  query: query  status: ''  ) |

### 5.2.2增加学生信息



点击添加学生，将弹出添加框，使用js脚本清空该模态框

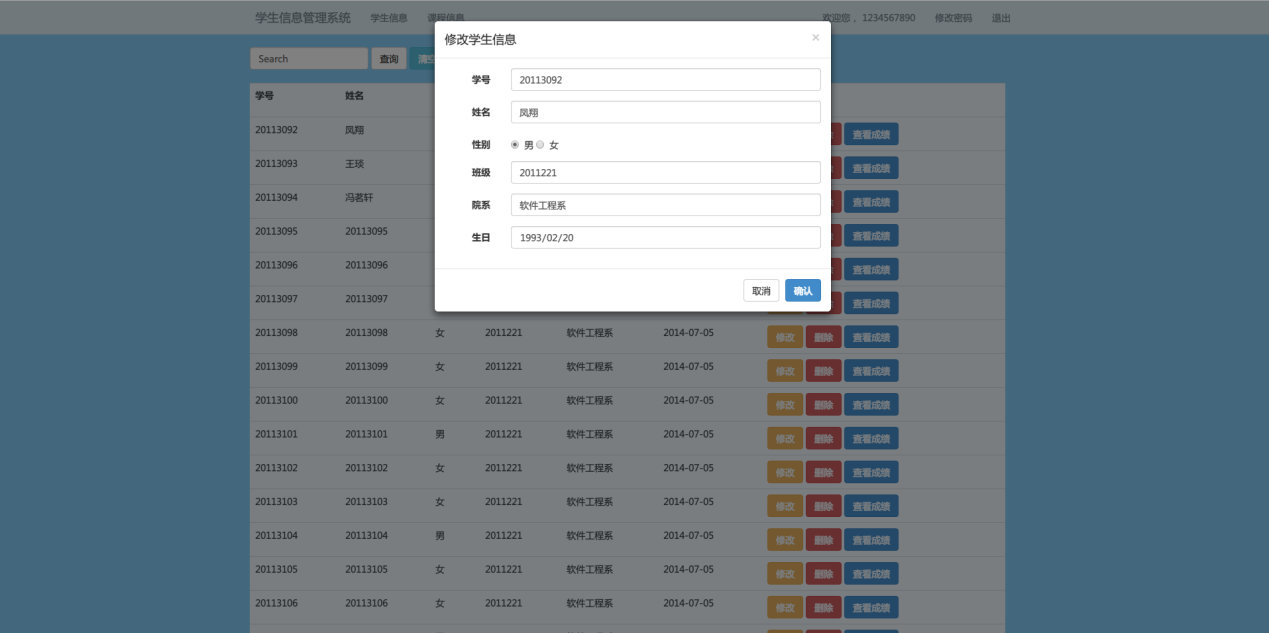
|  |
| --- |
| var add = function () {  $('#form').attr('action', '/student/add');  $('#modal-title').text('添加学生信息');  $('#sno').val('');  $('#sname').val('');  $('input[id=gender]')[0].checked = true;  $('#class').val('');  $('#department').val('');  $('#birthday').val('');  }; |

在添加框内输入正确信息，点击确认

系统返回学生列表页面，添加成功

|  |
| --- |
| # 增加一个学生  router.post "/add", (req, res) ->  # 登录验证  unless req.session.username?  res.redirect '/'  #解析post请求body  \_sno = req.body.sno  \_sname = req.body.sname  \_gender = req.body.gender  \_class = req.body.class  \_department = req.body.department  \_birthday = req.body.birthday  # 创建student实体  studentEntity = new studentModel  sno: \_sno  sname: \_sname  gender: \_gender  class: \_class  department: \_department  birthday: \_birthday  # 向数据库内插入该学生实体  studentEntity.save((err)->  if err?  console.log(err)  )  res.redirect '/' |

### 5.2.3修改课程信息



点击任意学生后的修改按钮，将弹出修改框。

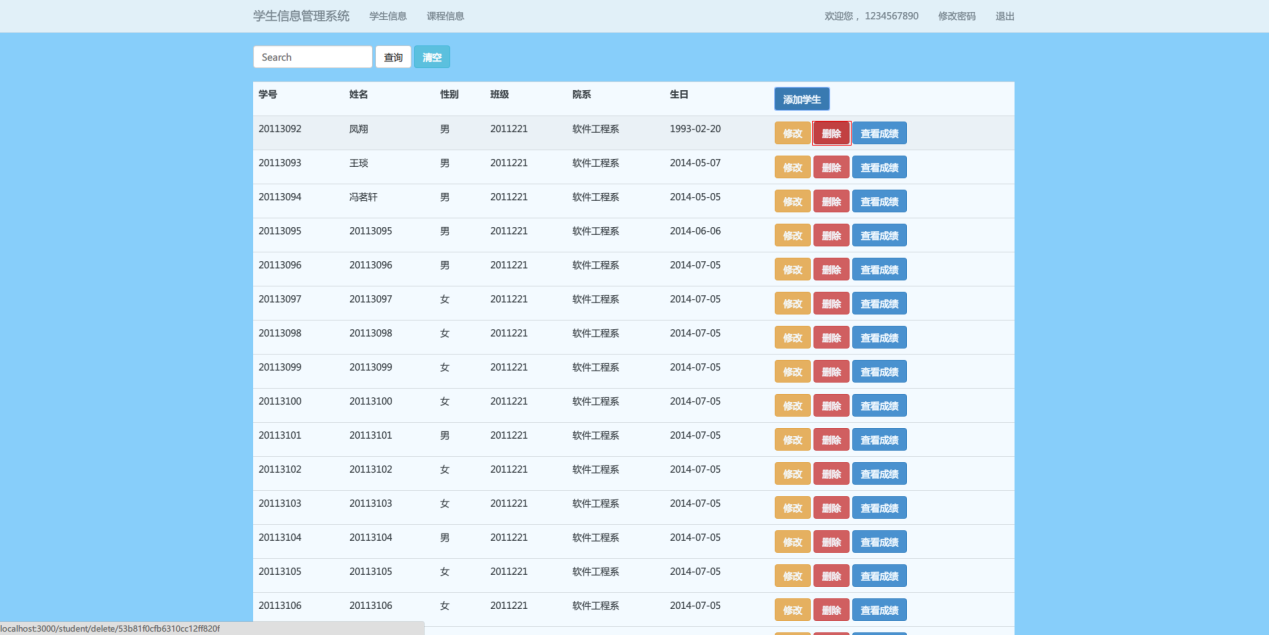
使用前端js脚本绑定修改内容

|  |
| --- |
| var update = function (\_id) {  $('#form').attr('action', '/student/update/' + \_id);  $('#modal-title').text('修改学生信息');  var id = '#' + \_id;  $('#sno').val($($(id + ' td')[0]).text());  $('#sname').val($($(id + ' td')[1]).text());  if ($($(id + ' td')[2]).text() == '男') {  $('input[id=gender]')[0].checked = true;  } else {  $('input[id=gender]')[1].checked = true;  }  $('#class').val($($(id + ' td')[3]).text());  $('#department').val($($(id + ' td')[4]).text());  $('#birthday').val($($(id + ' td')[5]).text());  }; |

在修改框内输入正确修改信息，点击确认，系统返回学生列表页面，修改成功

|  |
| --- |
| # 修改一个学生  router.post "/update/:sid", (req, res) ->  # 登录验证  unless req.session.username?  res.redirect '/'  sid = req.params.sid #获取学生ObjectId  \_sno = req.body.sno  \_sname = req.body.sname  \_gender = req.body.gender  \_class = req.body.class  \_department = req.body.department  \_birthday = req.body.birthday  #查询并修改  studentModel.findByIdAndUpdate(sid,  sno: \_sno  sname: \_sname  gender: \_gender  class: \_class  department: \_department  birthday: \_birthday,  (err)->  if err?  console.log(err)  )  res.redirect '/student' |

### 5.2.4删除课程信息



点击任意学生后的删除按钮，即可删除该名学生。

|  |
| --- |
| # 删除一个学生  router.get "/delete/:sid", (req, res) ->  unless req.session.username?  res.redirect '/'  sid = req.params.sid  console.log(sid)  # 查询并删除对应学生信息  studentModel.remove(  {\_id: sid},  (err)->  if err?  console.log(err)  )  res.redirect '/student' |

# 6总结

大三这一年，课堂上和课外，都做了很多项目。慢慢的对“软件工程”这个概念有了更多自己的认识。

#### 人月神话

人们一般很难意识到工程的伟大，但是仔细想想，从摩天大楼到宽阔马路，从高速动车到手机电脑。这一切的的建造和制造者，都是那些并不懂得其中原理的人。他们既不懂土木工程、建筑学，也不可能懂电路、微电子技术。但是在他们手里，伟大的工程奇迹一个个出现了。这，就是工程的力量。

软件行业也一样。互联网时代，用户对于高质量软件的需求越来越大，而真正哪些高素质的高级软件工程师总是少数。难道让他们一个个都来写增删改查么？当然不可能。另一方面，软件并不是一个单一的概念。只懂算法永远无法完成一款完整的软件。界面（UI）、用户体验（UX）、交互对于一款软件的重要性与日俱增。程序员们怎样和设计师沟通合作，也是一门很大的学问。

我相信，真正规范的软件开发一定是伟大的，我一直很喜欢标准化，但是在“软件工程”这个问题上还是犯了难。

软件工程、面向对象分析设计、软件过程、Scrum、XP……这些我们耳熟能详的名字，之间到底是什么关系，我们到底应该遵循哪一种方法。我感受到的是，在“人月神话”之后，大家开始寻找工程化的开发方式。并且已经找到了很多很完善的方案。但是在实际使用中，不是所有人都愿意完全遵照方案开发——很明显，这样太麻烦了。

首先，报告一定是给别人看的，对于自己，代码中的注释已经足够了，甚至写的好的代码，应该不用注释也可以轻易理解。所以这就几乎决定了这些方案都是为多人团队开发准备的工具。在项目没有到达一定规模，团队没有到一定人数的时候，很难体会到这些方法的优势。

其次，经典的软件工程方法，是要求在Coding之前完成所有的设计，但是现实中，据我对自己接触公司的了解（当然都是一些比较小的公司），大家多数采取的还是原型迭代的快速开发方法。毕竟，只有对系统真的非常了解的情况下才有可能完整的架构一个项目。对于现在的我来说，写代码之前，很难想象这些功能应该怎么实现。而频繁变更的需求，也总会让已经写好的文档多次回炉。在研发的时候，与其选择纸上谈兵的文档推动，不如选择原型推动。

对于现在的我们来说，从未见过标准化的文档(网上是有很多文档，太多了，以至于还是无法得知到底怎样才算是标准的)，从未参与过标准化的开发。没当过小兵直接当将军实在是困难的。还不算程序员，直接架构师也实在是令人困扰的。

#### 没有银弹

这次课程设计，正好不规定使用技术，确实非常适合对新技术进行实践。确实对沉重的Java EE和严谨的SQL数据库有些厌烦了，于是很自然的选择了自己有过一些了解的Node.js+MongoDB这一对好搭档，事件驱动和异步I/O的组合，在带来极佳并行性能的情况下，也是一种全新的编程体验。

相信写过Python的人没有不爱它的，脚本语言可以十分方便的以最短的代码写出各种复杂的功能，还有优雅的map reduce、lambda表达式等。相对于Javascript类似Java的语法，我个人还是更倾向于Python的缩进脚本语法。于是干脆新技术用到底，上Coffeescript。

这里也不详述技术选型问题了，主要还是谈谈使用感受吧。最主要的感觉还是更加理解“没有银弹”这句话了。

新技术的使用，在带来方便的同时，一定也会带来问题。首先就是我简直作死的全部使用了自己不熟悉的技术，真的是全部，包括IDE都用了新的。而且自己绝对的低估了学习成本，高估了自己的英文水平。新技术嘛，那一定是很少有中文教程的，唯一的方法就是看官方文档，去Stackoverflow上提问。这几天一通文档看下来，感觉自己的英文水平大涨，这是始料未及的。此外也开始意识到官方文档的优势——既然有第一手资料，作者一定是最熟悉的人，他告诉你该怎么写，那当然比看别人转述的教程更加清楚了。

此外就是技术本身的问题了，比如Node的异步，在带来高效率的同时，增加了编码难度。异步回调的嵌套，毫无可读性可言。更是遇到了一个到最后都只是用讨巧方案解决的问题，浪费了大量时间。所以，使用新技术节约下来的时间，都被学习时间以及适应时间消耗了，总体下来，还真不一定比自己用熟悉的工程语言Java完成更快。确实没错，“没有银弹”。

#### 没有使用到却希望使用的技术

既然是总结嘛，那就记下来，后面希望能有机会尝试，毕竟学会新技术还是很有意思的

* Async：Node.js异步流程控制
* AngularJS：页面模板，这次因为用了ejs就没用Angular
* Mocha：自动化单元测试
* Passport：登录验证中间件