**公共课室管理系统**

**软件设计文档**

**Version x.x**

|  |  |  |
| --- | --- | --- |
| 文件状态：  [ ] 草稿  [ √ ] 正式发布  [ ] 正在修改 | 文件标识： |  |
| 当前版本： | 2.0 |
| 作 者： 者： | 陈梓浩 梁喻源 李宇捷 |
| 完成日期： | 2016.7.14 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **修订历史** | | | | |
| **修订日期** | **修订人** | **版本号** | **审核人** | **修订说明** |
| 2016.7.11 | 陈梓浩 | V1.0 | 李宇捷 | 撰写软件设计文档模板 |
| 2016.7.13 | 梁喻源 | V1.1 | 李宇捷 | 完成软件设计文档引言、总体设计 |
| 2016.7.14 | 梁喻源 | V1.2 | 李宇捷 | 完成软件设计文档架构设计、运行环境以及测试计划 |
| 2016.7.14 | 梁喻源 | V2.0 | 李宇捷 | 完成软件设计文档程序系统结构与研发风险 |

目录

[1. 引言 4](#_Toc456130175)

[1.1. 编写目的 4](#_Toc456130176)

[1.2. 项目背景 4](#_Toc456130177)

[1.3. 相关定义 4](#_Toc456130178)

[1.4. 参考文献 4](#_Toc456130179)

[2. 总体设计 4](#_Toc456130180)

[2.1. 系统目标概要 4](#_Toc456130181)

[2.2. 系统方案（用例）概要 4](#_Toc456130182)

[2.3. 系统功能 4](#_Toc456130183)

[2.4. 系统性能 4](#_Toc456130184)

[2.5. 子系统设计 4](#_Toc456130185)

[2.6. 项目使用到的软件技术 4](#_Toc456130186)

[2.7. 技术选型理由 4](#_Toc456130187)

[3. 程序系统的结构 4](#_Toc456130188)

[3.1. 模块划分 4](#_Toc456130189)

[3.2. 模块之间的关系 5](#_Toc456130190)

[3.3. 模块对应的软件技术 5](#_Toc456130191)

[4. 运行环境 5](#_Toc456130192)

[5. 外部因素 5](#_Toc456130193)

[6. 架构设计 5](#_Toc456130194)

[6.1. 系统体系结构 5](#_Toc456130195)

[6.2. 类图 5](#_Toc456130196)

[6.3. 模块视图 5](#_Toc456130197)

[6.4. 用户界面 5](#_Toc456130198)

[7. 测试计划 5](#_Toc456130199)

[8. 研发风险 5](#_Toc456130200)

[8.1. 技术风险 5](#_Toc456130201)

[8.2. 团队管理风险 5](#_Toc456130202)

# 引言

## 编写目的

经过我们的讨论，发现一个面向高校学生的公共课室管理系统（Public Classroom Management System以下简称PCM）对于高校学生的学习活动将有很大效率的提高，也将减少同学们在学校内各部门之间来回奔波的繁琐。于是我们决定编写一个PCM系统。

## 项目背景

该项目为软件综合设计课程的课程项目，由李宇捷、林勇洁、林育新、陈梓浩、梁喻源共同开发。本项目也是我们小组在课程《系统分析与设计》中的课程项目。

## 相关定义

|  |  |  |
| --- | --- | --- |
| **术语** | **英语名词** | **定义** |
| 公共课室管理系统 | Public Classroom Management (PCM) System | 本次的开发任务，能够进行课室申请、占用状态查询、在线审批的系统。 |
| 普通用户 | User | 自行注册并且能够查询课室、申请课室的用户，但此类用户不能进行审批。 |
| 管理员 | Admin | 由注册员注册、能够查看申请记录并且审批申请的用户。 |
| 申请记录 | Record | 指普通用户对使用公共课室的申请，与用户相关联。 |
| 课室占用 | Occupation | 指课室在该时间段以被使用，也即已有申请已被审批通过了。 |
| 申请队列 | Applicant Queue | 申请某课室、某时间段使用的申请者队列，其中越早申请者，其优先级越高。 |
| 审批记录 | Approval Record | 指由管理员对普通用户提交的申请进行审批。有两种可能的结果：通过与拒绝。 |

# 总体设计

## 系统目标概要

PCM系统目标允许用户通过移动移动终端浏览器访问该系统。对于用户, PCM系统具有用户注册登录、课室申请、课室查询、申请状态查询等功能。对于管理员用户，PCM系统管理员能够查看过去及未来所有课室的空闲、待审核和占用状态，并且能够审核待审核的申请。

## 系统方案（用例）概要



## 系统功能

用户注册:用户可以通过邮箱注册成为PCM用户，注册提供邮箱、密码和名字信息。完成注册后，用户需进行邮箱验证，以激活该账号。

用户登录：用户通过邮箱账号以及密码登录PCM用户，首次登录会要求用户完善个人用户信息，用户进入本系统后才能够进行申请、取消申请的操作。

找回密码：用户如果忘记密码可以通过绑定的邮箱验证身份并重置密码。

课室查询：PCM系统允许用户查询当前以及未来一段时间课室的占用、申请队列状态，登录后的用户可以通过输入查询时间段等一系列条件来查询课室的占用情况。

课室申请：用户通过PCM系统可以申请特定时间段的特定的公共课室 (如不指定则随机选择一个可用的公共教室) 的使用，如果该时间段的该课室已被其他用户申请，则提醒用户该非空申请状态，若用户确定申请则增加一条申请记录，如果未被其余用户申请，则直接增加一条记录。除此之外，用户还能够取消其未来的申请。

审核申请：管理员能够查看过去及未来所有课室的空闲、待审核和占用状态，并且能够审核待审核的申请。所谓审核，即通过或者拒绝用户的申请。一旦一条申请记录的状态改变，系统即向相关用户的绑定邮箱发送一封提醒邮件，以告知用户其申请是否被允许通过。对于某个时间段、某个课室的申请队列长度大于1，则优先通过较早的申请同时拒绝队列中其他申请。

申请状态查询：用户可以查看申请课室的状态，及时了解申请处于哪一过程以及申请结果。

## 系统性能

**时间性能**

(1) 为了提高用户体验，系统应该能够快速地进行页面响应，当点击对应选项之后，进入下一个页面的响应时间一般小于3s。

(2) 因为本系统是基于JS所开发的，当运行时需要利用浏览器打开，故响应的时间还与用户当前的网络状况有关系。

**适应性**

本系统适用于当前所有主流的浏览器，不需要安装多余的插件，操作简单方便。

## 子系统设计

PCM 系统包括三个子系统：用户管理子系统、课室申请子系统和申请审批子系统。这三个子系统向主系统提供接口，由主系统调用这三个子系统的接口，以实现其功能。其关系如下图所示。

* **用户管理子系统**

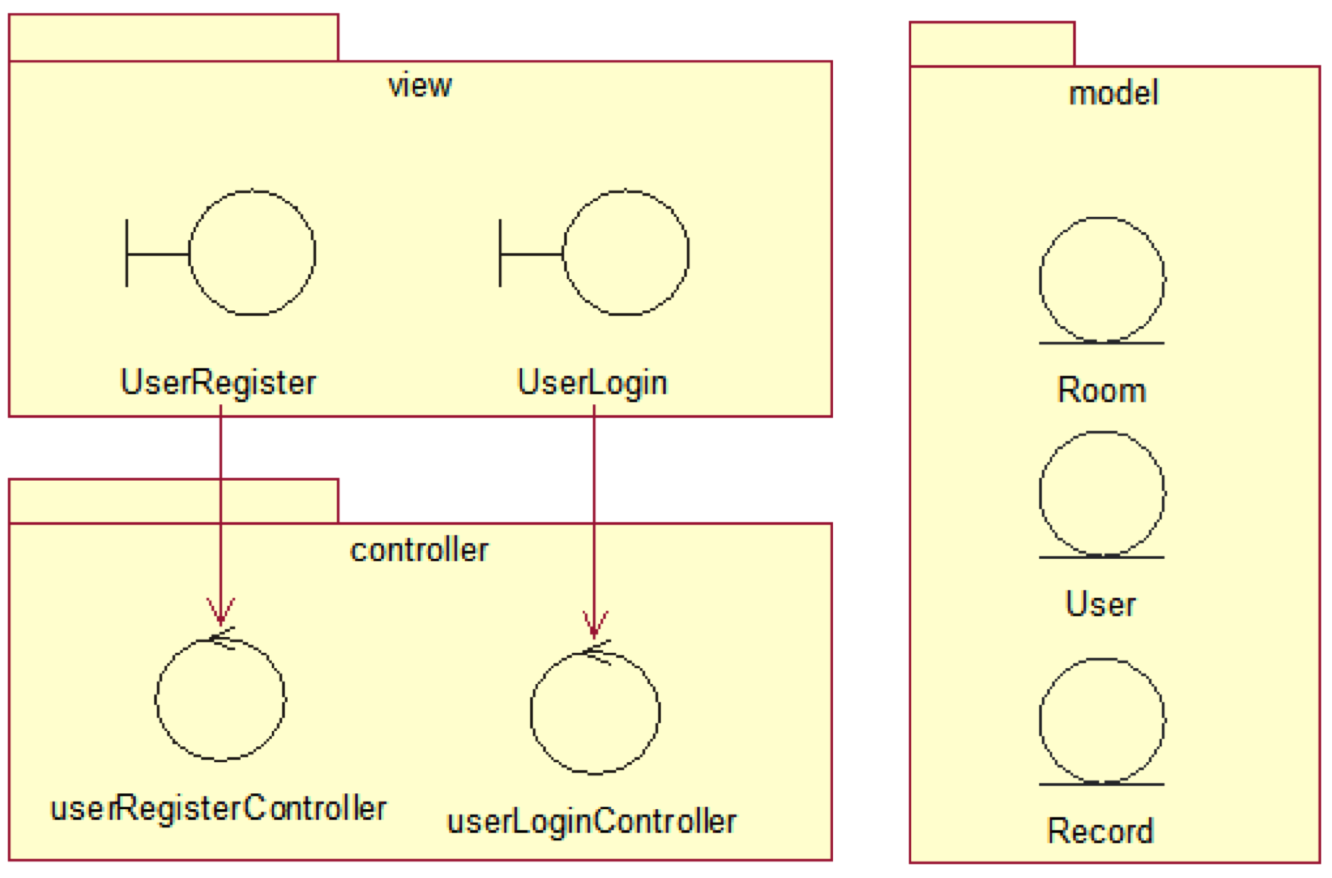
用户管理子系统遵循总体架构的分层策略，分别为视图层、控制层、实体层。架构图如下图所示。

视图层:

由边界类组成，包括 UserRegister 和 UserLogin 两个边界类。提供数据显示和操作界面。

控制层:

由控制类组成，包括 userRegisterController 和 userLoginController 两个边界类。负责处理用户管理的业务逻辑。



用户管理子系统架构图

* **课室申请子系统**

课室申请子系统分为视图层、控制层、实体层。架构图下图所示。

视图层：

由边界类组成，包括 RoomApply和 RoomQuery两个边界类。提供课室数据显示和操作界面。

控制层：

由控制类 roomApplyController和 roomQueryController组成。负责处理课室数据查询、申请等业务逻辑。



课室申请子系统架构图

* **审批管理子系统**

审批管理子系统遵循总体架构的分层策略，分别为视图层、控制层、实体层。架构图如下图所示。

视图层：

由边界类组成，包括 ApprovalRecord类。提供用户申请记录数据显示和操作界面。

控制层：

由控制类组成，包括 approvalRecordController类。控制层负责处理审批申请的业务逻辑。

审批管理子系统架构图

审批管理子系统架构图

## 项目技术

前端--vue.js

前端ui 框架--Vux

后端--express

数据库--mysql

后端构建工具--gulp

## 技术选型理由

**vue.js：**一个精简的 MVVM。从技术角度讲，Vue.js 专注于 MVVM 模型的 ViewModel 层。它通过双向数据绑定把 View 层和 Model 层连接了起来。实际的 DOM 封装和输出格式都被抽象为了Directives 和 Filters。Vue.js和其他库相比是一个小而美的库，作者的主要目的是通过一个尽量简单的 API 产生可反映的数据绑定和可组合的视图组件。

1. vue.js 官方文档清晰，比angular 更容易上手

2. 通过异步批处理的方式来进行dom 更新

3. 低耦合，通过组件化的方式，构建应用

4. 对各个模块更为友好，可以通过npm bower 等方式进行安装，不像angular 强迫各种语法设定，适用场景很多

5. 响应式编程：Vue 已经把 object 对象改造成了一个『响应式对象』。当你修改 object.message 的值时，渲染的 HTML 会自动更新。更关键的是，你不需要担心你是否因为是在一个 timeout 里面修改了状态而需要调用 $apply等。

6. 组件化：组件可以套其他组件，最终形成一个代表了你的 UI 视图的树状结构。为了让组件之间能够有效的进行动态组构，Vue 组件可以：

用 props 来定义如何接收外部数据；

用自定义事件来向外传递消息；

用 <slot> API 来将外部动态传入的内容（其他组件或是 HTML）和自身模板进行组合。

**Vux：**与vue.js 完美锲合 一脉相承 简化开发

**Express：**express 作为一个web framework ，既不是最底层的，又不是最抽象的，但是应用广泛，又有典型架构，发展时间较久，官方文档完整，api 完整，自带Router 和路由规则等，很适合作为初学的目标

**Mysql：**根据需求，各表之间关系较大，并且数据量不大，关系数据结构较为简单，小组讨论最终使用mysql 更为方便

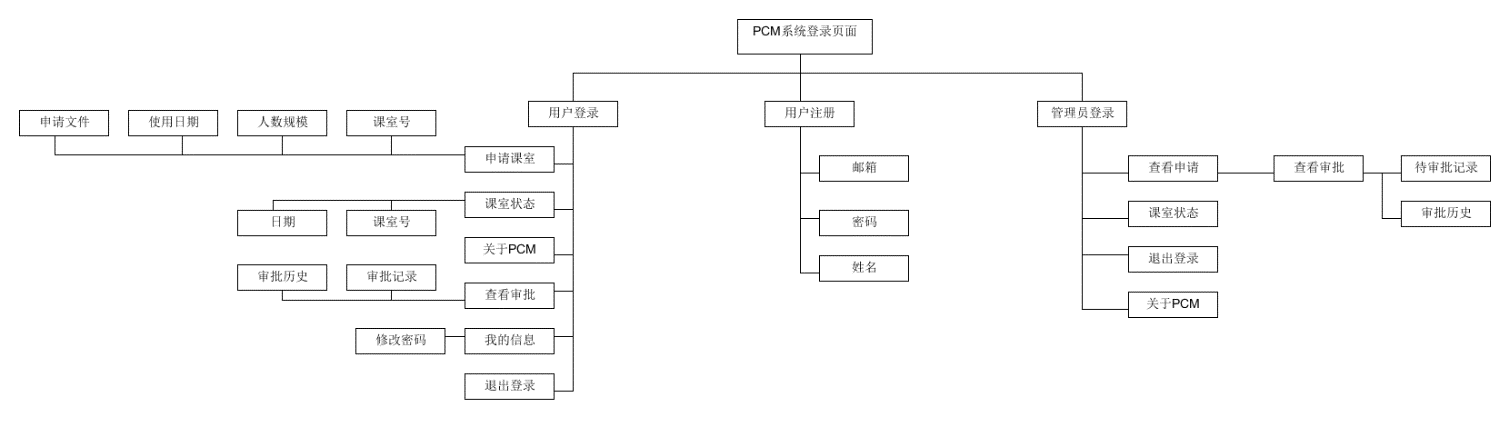
**Gulp：**后端代码构建工具以及实现自动化的构建工具，开发过程中，不必要过多关注过多重复性操作，实现文件的压缩，并且打包至相对应的目录

# 程序系统的结构

## 模块划分

PCM系统分为用户登录模块、管理员登录模块、用户注册模块。

用户登录模块又往下细分为查看审批、课室查询、申请课室模块和取消课室申请模块；管理员登录模块下则有查看申请、审批申请模块。



## 模块功能

用户登录模块：可以调用申请课室、课室查询、查看审批、取消课室申请等模块。

用户注册模块：可以通过邮箱、密码、姓名进行用户注册。

管理员登录模块：可以调用查看申请、审批申请等模块。

课室查询模块：可以查询想要的课室在特定时间是否空闲。

申请课室模块：申请课室。

查看审批模块：可以查看未来申请课室的请求是否审核通过。

取消课室申请模块：取消未来的申请课室请求。

查看申请模块：查看学生用户课室申请列表。

审批申请模块：对学生用户课室申请进行审批。

## 技术实现



# 运行环境

各主流现代浏览器。

# 架构设计

## 系统体系结构

本系统基于B/S 架构，使用 MVC (Model-View-Controller) 的架构模式，分别为视图层、控制层及实体层。

* 视图层

视图层系统提供的用户交互界面，负责展现实体的状态及获取用户的请求与信息。本系统视图层的模块包括：用户管理模块（登录／注册）、用户个人首页、课室申请界面模块、后台管理员查看信息模块、课室状态查看模块等。

* 控制层

控制层是实现系统业务逻辑的核心，负责对业务逻辑代码进行抽象和封装。控制层接收用户的请求和信息、调用实体层的数据接口、执行相应的逻辑操作以及将结果返回给视图层。本系统控制层的模块包括：用户管理模块、课室申请及审批、课室状态查询、课室匹配模块等。

* 实体层

实体层负责系统的实体对象的数据访问，数据访问对象提供了控制层访问实体层的接口方法，实现对数据库的增删改查的操作。本系统实体层包括：普通用户、管理员用户、课室、申请记录。

系统框架设计图：



## 类图

系统总类图:



* **用户注册**

用户能够注册成为本系统普通用户，注册需提供邮箱、密码与名字信息。完成注册后，用户需进行邮箱验证，以激活该账号。

边界类：

边界类为用户注册界面类（UserRegister）。该界面以表单的形式呈现，用于获取用户输入的用户邮箱、密码等信息。

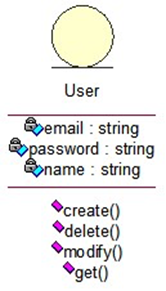


控制类：

用户注册的控制类为 userRegisterController类，该类负责将边界类的信息校验后，将数据信息经由数据访问对象接口，将用户信息加入数据库。



实体类：



对应关系：



* **用户登录**

边界类：

用户登录用例的边界类是 userLogin 类，该类以界面形式直接与用户交互，接收用户输入的邮箱地址与密码，并判断是否非空。



控制类

用户登录用例的控制类是 userLoginController 类，该类负责判断从边界类得到的用户邮箱与密码与实体类中的对应邮箱地址与密码是否匹配。经分析，该类类图如图所示。



实体类：



对应关系：



* **用户查询课室状态**

边界类：

用户查询课室的边界类是RoomQuery，负责接收用户所设置的查询条件，并将条件转交给控制类。



控制类：

queryController 是该用例的控制类。该类根据从边界类得到的查询条件，相应地调用实体类的查询方法，并将查询结果组装返回给边界类。经分析，其类图如图 14 所示。



实体类

实体类为 Record 与 Room。



实体类 Room



实体类 Record

对应关系：



* **用户申请课室**

边界类：

RoomApply 类是用户申请课室用例的边界类，用于获取用户输入的申请时所要求的信息。



用户申请课室边界类

控制类：

本用例控制类为 roomApplyController 类，该类接收边界类提交的申请信息，检验所填信息是否满足申请要求，然后调用实体类的接口增加一条申请记录。



用户申请课室控制类

实体类：

实体类 Record 包含了一条申请的所有信息，包括申请用户的信息、使用时间信息、对应课室信息等。



实体类 Room



实体类 Record

* **管理员审批课室**

边界类：

ApprovalRecord 类是管理员审批课室的边界类，其界面给用户提供对课室状态更改的操作，包括审批通过与拒绝等。



管理员审批课室用例边界类

控制类：

审批课室的控制类是 approvalRecordController，负责将边界类的信息转交给实体层的响应类完成相应操作。



管理员审批课室用例控制类

实体类：

本用例的实体类也即 Record 类，控制类可调用 Record 类的更新方法来改变状态信息。



实体类 Room



实体类 Record

审批申请类之间的关系：



## 用户界面

初始页面：



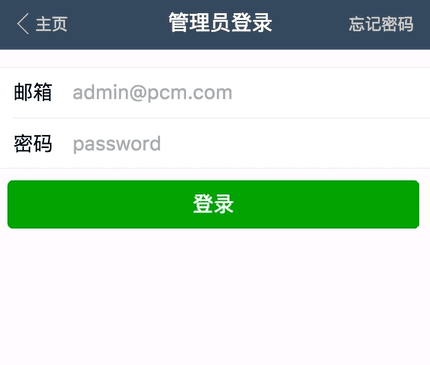
用户注册页面：



学生用户界面：



管理员登录界面：



管理员用户界面：



# 测试计划

## 测试目标

模块内部逻辑正确，模块间功能完备，交互准确，系统整体运行无差错

## 测试设计

与迭代周期相吻合，每次迭代之后进行下述的三层相吻合

第一层：单元测试，在模块内部主要逻辑处进行测试，保证无差错

第二层：模块测试，主要设计各种各样的测试输入，测试模块是否达到预期的逻辑与功能

第三层：集成测试，模块间拼接后，要进行功能性2测试，保证模块间逻辑与交互正确，符合预期效果

第一层测试以白盒测试为主，侧重逻辑

第二、三层测试以黑盒测试为主，侧重功能

# 研发风险

## 技术风险

Vue作为一个个人项目的前端库，虽然已经研发两年之久，但是进展缓慢，现在虽然也有人在生产环境使用它，但是兼容性和稳定性还是需要讨论与评测风险后才可以去使用。

Express 是基于 callback 来组合业务逻辑。Callback 有两大硬伤，一是不可组合，二是异常不可捕获。Express 的中间件模式虽然在一定程度上解决这两个问题，但没法彻底解决。

对于gulp，需要使用者有非常强的处理错误能力，如果遇到插件bug，就需要联系作者，这个是一件非常棘手的事情。虽然这种事情很少出现，但是这种风险是存在的。