尊敬各位老师大家上午好，我叫汪思朋，来自计科院计算机科学与技术专业，我今天的答辩题目是基于VUE和spring boot的教考分离的考试系统的设计与实现，我的毕设指导老师是谯英教授。接下来，开始我的答辩内容。

首先是答辩目录，一共分为七个章节，分别是课题背景介绍、需求分析、核心数据库表的设计实现。这个考试系统的架构、考试系统详细设计实现、整个系统的测试、最后是对这个毕设的一个总结。

这部分介绍了我毕设论文的一些摘要，是对整体内容的一个概括。

文章的三个重点也就是：教考分离、考试系统、前后端分离

第一章

|  |
| --- |
| 然后开始第一章，课题背景的分析，也就是选题的目的和意义  教考分离是相对于教考合一而言的，教考合一指的是教学和考试均由同一位教师完成，经过长期的实践证明了教考合一有如下几个缺点。   * **各老师的试题封闭不互通** * **人工组卷工作量大** * **整体结果不易量化**   整体结果不易量化是指每位老师出的题目不同，不易去考察老师的教学质量和学生的知识掌握情况。  所以需要设计一个教考分离的考试系统来帮助教师减少工作量、统一题库、以大纲为准去检测教学质量。 |

第二章

|  |
| --- |
| 这章是需求分析部分，主要的工作是通过调研市场已有方案来确定考试系统的核心功能。  调研了超星学习网、中国大学MOOC、学堂在线等考试平台，踩在巨人的肩膀上选取了其中的主要功能进行实现。  需求分析从完整的考试流程说起。首先要进入考试系统就需要登录功能，登录之前的注册功能是必不可少的。  在注册时，还需要区别用户的身份，比如学生和教师，所以还需要用户管理、组织机构管理和权限管理功能。  登录之后进入首页有着不同的菜单，所以还需要一个功能（菜单）管理功能，这个根据用户的身份来显示不同的菜单。  然后老师新建题目，首先需要新建一个题库，所以需要一个题库管理功能，因为题库是以大纲为导向的，在新建题库之前还需要建立大纲，大纲管理应运而生。  上传完题目之后就是组卷、发布考试了，这一部分主要由教师完成，学生参加考试之后完成试卷的批阅。这就是大概的系统流程，整体功能如右图所示。 |

第三章

|  |
| --- |
| 本章是整个系统的核心数据库表的设计，具体讲述了表的关键字段和各个表之间的联系。  (1) 用户表是用户管理中的重要部分，许多操作都需要标明用户的身份才能进行，比如创建试题、试卷等操作。用户表是注册、登录、用户身份信息鉴权的重要部分。  用户可以使用用户名密码登录和邮箱登录，所有有name，pwd，email字段，然后使用role和orgcode标明用户的身份和组织。  (2) 菜单表是权限控制的重要部分，主要是由前端来控制，所以是单独隔离的一个表。  菜单具有多级目录，所以设计一个parentid来标明菜单的父级目录，click字段表示点击事件，一般是前端路由跳转的路径  Right表示菜单的权限，使用这个对用户进行权限的控制。  (3) 大纲表是创建试题、题库、试卷不可或缺的一部分，大纲相当于书本中的章节，创建试题的时候可以选择某个小节作为这个试题的大纲，也就是知识点，创建试卷的时候  选择某个章节作为试卷的大纲(知识点)，在之后的自动组卷的遗传算法中，就是根据大纲来设计的适应函数。 大纲的设计和菜单设计一样。  题库与问题之间进行连接。  (4) 在创建题目的时候，可以根据自己的选择创建不同的题目，这里将各个类型的题目分离开来，方便进行管理，单选、多选、填空、判断单独成一个表。  问题表存储题目的共同信息如大纲、题库、题目描述。同时试卷、答案都会出现问题，所以与问题表进行连接。  (5) 整个考试系统最重要的部分就是考试，所以试卷表是这个系统的核心，在创建的时候会选择大纲、难度和问题数量然后自动生成试卷。  试卷与问题之间通过 【问题-试卷表】进行连接，考生答案表中使用paper-id作为外键与试卷表进行连接。  单独设计一个表是因为一个题，有可能也同时出现在其他题库。  试卷与考生、考场、考试之间的关系使用考试信息表进行连接，考试信息表具体信息如下页所示。  (6) 考试信息表将考生、试卷、判分、答案、考场集合起来，实现一场完整的考试。  考场表是一个考场和一群考生的集合。  考试信息表将考试的判分表进行连接，方便客观题、主观题的判分  单独用一个信息表分开是因为一个考试有多个考场。(可以不说)  (7) 考生表会注明这场考试的所有信息，同时与考生答案表进行连接，保证以后的分数统计部分正常使用。(可以掠过) |

第四章

|  |
| --- |
| 分析完数据库的设计之后，就是整个系统的架构设计了，因为这个系统是前后端分离的。所以这章主要讲了前后端的系统架构以及技术路线的分析。  整个系统架构如图所示，分为展示层、业务层、数据层、数据存储部分。整体来讲就是后端根据业务层需求，在数据层存取数据，构建RESTful 的接口供前端调用，然后加以显示。  然后技术路线部分，前端部分构建页面使用了element-ui框架，帮助我快速构建页面，使用了webpack来构建前端的服务  使用vuejs框架来实现页面的逻辑。Mvvm，使用npm来管理前端的依赖。  后端部分使用了spring boot框架来实现服务的逻辑，使用tomcat来构建后端的服务，使用maven来管理后端的依赖，使用MySQL数据库来做数据的存储。  因为前后端运行在不同的端口，也就是不同的进程上，进行http通信的时候需要通过代理服务器进行接口的转发工作。 |

第五章

|  |
| --- |
| 注册的时候会使用正则表达式对用户的输入进行验证。只要有一项字段没有通过就不会发起http请求，否则发起注册请求。主要对手机号、邮箱和密码进行验证。  登录部分采用了传统的JWT验证方式，用户验证通过会在服务器通过加密算法生成一个token，这个token包含了用户的信息。  登录的时候有两个登录选项，一个是用户名密码登录，一个是邮箱验证码登录。使用验证码登录的时候，为了防止用户重复发起请求验证码的动作。会设置一个间隔时间，在这个间隔时间之内，前端禁止用户发起新的请求。然后邮箱的验证码因为有时效性，所以不会存储在数据库中，我存在了高速缓存服务器redis中，可以设置一个过期时间  在过期时间内去查询，如果有的话就通过验证。  注册部分分为接口注册和上传文件注册，接口注册会在前端对每个注册的字段进行验证，上传文件注册首验证文件的后缀。然后判断是否为大文件，如果是大文件就走大文件的流程，最后就是后端进行注册的一个流程。  然后是组织机构管理功能，这个就是管理用户的组织，比如计科院某某班级，这是一个高权限的操作，会进行日志的上报。  功能管理本质上是菜单的管理。  然后是大纲和题库管理。题库加密了之后需要输入密码进行访问。  题目管理部分。一开始，题目、选项、答案部分的数据库字段仅有文本部分，并未添加图片部分，后来在开发的过程中，参考到超星、慕课、腾讯课堂等平台的组卷系统，发现大多数平台的试题组织(题目)在添加题目的时候，都做了富文本的处理。包括文本的加粗、插入表情、插入图片等等。考虑到富文本的存储复杂以及前端实现的难度，因此放弃了添加富文本的计划，仅保留一个插入图片的功能。图片base64和文件上传  首先，每一种类型的题目都有一些共同的字段如题目描述、大纲节点、题库id、题解答案等，将这些通用的表单做成一个组件，嵌入在各个类型的题目中，作为公共字段进行提交。各个不同类型的题目做成不同类型的组件，既方便对其进行修改，也能保证表单最大限度的复用。题目的信息回填也是分不同的组件进行回填，保证不同字段之间的解耦，增强程序的健壮性。  自动组卷就是用户自己选择想出的题型及其数量，设置难度，知识点分布(大纲)，系统自动生成试卷。这里主要用到了遗传算法的相关知识。  遗传算法是一种启发式搜索，其灵感来自查尔斯·达尔文的自然进化论。该算法反映了自然选择的过程，其中选择最适合的个体进行繁殖以产生下一代的后代。  每个试卷时一个染色体，试卷中的题目就是基因。   1. 适应度函数的设计   适应度函数跟试卷难度系数和知识点覆盖率有关。  试卷难度系数公式：P=∑Di×Si/∑Si；其中i=1，2，…N，N是试卷所含的题目数，Ｄi，Ｓi分别是第i题的难度系数和分数。  知识点覆盖率系数公式：D = (1 – M/N)，  例如期望本试卷包含N个知识点，而一个个体中所有题目知识点的并集中包含M个（M<=N），则知识点的覆盖率为M/N。用户的期望难度系数EP与试卷难度系数P之差越小越好，知识点覆盖率越大越好，因此适应度函数为：f1- D \*f1-|EP-P|\*f2=。其中f1为知识点分布的权重，f2为难度系数所占权重。当f1=0时退化为只限制试题难度系数，当f2=0时退化为只限制知识点分布。这里(1- M/N)是因为覆盖率越大，M/N这个值就越大，因为适应度是一个正向的反馈值，所以要用(1-M/N)。  所以仅需要输入好试卷标题、试卷难度、组卷数量、各类型题目的题型和数量之后，系统将会根据输入的试 卷难度和课程设置的大纲下的各个章节的掌握度，自动计算出整张试卷每个章节的题目占所有题目的权重，在最后得到符合输入的试卷整体难度的试卷。  试卷数据的收集、多人提交的处理。 如何去收集试题的答案数据 首先在显示试卷上，复用了之前题目管理的试题组件，不再为每一道题目单独写template模板，给每一个题目类型的组件添加一个 returnAnswer 的方法，返回该题目的答案。  考试的试卷不同于常规的表单，数据是固定的，因为VUE是一个MVVM框架，数据与HTML模板进行了双向绑定，可以通过双向绑定获取输入框地内容，所以一开始可以将formData与表单进行强绑定。试卷的试题不是固定的，且试题类型不统一，给收集试题答案的数据带来了极大的困难。  因此设计了一种思路去动态地收集试卷的数据。首先进入考试界面通过HTTP请求拿到试题的数据之后，将数据转换为一个表单数组，这个数组的长度就是此张试卷试题的数量，然后通过表单数组去动态地自定义渲染模板中的数据，重新进行数据的绑定。数组中的每一个元素都绑定到了对应的组件上，就可调用组卷的returnAnswer方法获取数据，这样试卷的表单就和普通的表单一样，有着固定的字段绑定，无论是修改还是提交都非常地方便。 如何处理多人同时提交 在考试的时候，会遇到很多同学同时提交试卷的情况，这个时候后端是无法承受住如此高的并发量的，所以此时用到了RabbitMQ(Message Queue)高级消息队列协议。具体流程如图5.11所示。RabbitMQ相当于一个缓存的中间件，在多名同学同时发出交卷请求的时候，会将这个时刻所有的请求体的内容存到RabbitMQ中，而不是直接到后台去进行数据库的添加工作，因为数据库每秒执行的SQL数量有限。然后后台系统就从RabbitMQ中一条一条地读取这些数据，再进行操作，既没有影响用户体验，也很好地保护了服务器资源。 |