

TONGJI UNIVERSITY

《WEB技术》

实验报告（大作业）

|  |  |
| --- | --- |
| 实验名称 | Online Exam在线考试系统 |
| 小组成员 | 2053190 刘峥 |
|  | 2053039 陈一鸣 |
| 学院（系） | 电子与信息工程学院 |
| 专 业 | 计算机科学与技术 |
| 任课教师 | 郭玉臣 |
| 日 期 | 2022年 6 月 22 日 |

目录

[一：项目背景 2](#_Toc106901697)

[二：需求分析 2](#_Toc106901698)

[2.1 用户需求分析 2](#_Toc106901699)

[2.2 功能需求分析 3](#_Toc106901700)

[三：整体设计 3](#_Toc106901701)

[3.1 技术设计 3](#_Toc106901702)

[3.2 功能设计 4](#_Toc106901703)

[四：前端搭建 5](#_Toc106901704)

[五：后端搭建 8](#_Toc106901705)

[5.1 数据库搭建 8](#_Toc106901706)

[5.2 交互逻辑搭建 9](#_Toc106901707)

[六：测试数据 14](#_Toc106901708)

[6.1 考试大厅页面数据测试 14](#_Toc106901709)

[6.2 我的考试测试数据 15](#_Toc106901710)

[6.3 问题测试数据 17](#_Toc106901711)

[七：实验结果展示 19](#_Toc106901712)

[八：小组分工 22](#_Toc106901713)

[九：心得体会 22](#_Toc106901714)

[十：web技术认识与体会 23](#_Toc106901715)

一：项目背景

今年是新冠病毒出现的第三年，但新冠疫情仍不断反复，对各行各业正常的生产生活秩序都带来了不利影响。对于学校而言，为保障学生的生命健康安全，在遇到疫情时通常会采用停止线下上课的措施，将课堂等教学活动转移到线上，在这样的情况下，线上教学的重要性和必要性不断凸显，需求不断增加。在日常教学过程中，平时的测验与期中期末等考试是教学的重要环节，可以检验学生阶段性学习效果并及时督促学习，并且也是最终成绩评定的重要依据之一。在疫情影响下的线上教学中如何实施测验便成为了一个需要解决的现实问题。

在这样的背景下，我们使用课上所学的web相关技术，开发出一套在线考试系统Online Exam，主要使用Vue cli脚手架结合Ant Design Vue搭建前端，使用SpringBoot结合MyBatis搭建后端，并采用MySQL作为数据库，实现登录注册、查看考试、新建考试、参加考试、考试管理和个人设置等功能，能够基本满足线上测试的需求，并按照功能进行了页面的设计与优化，尽量体现简洁与美观的效果，以实现较好的交互效果。

二：需求分析

2.1 用户需求分析

疫情期间线上教学与线上考试成为老师与学生交流的主要方式，因此，一个较好满足老师和学生需求的在线考试系统能够带来巨大便利。考虑到一般考试系统都包含学生与老师两个用户，因此我们分别对其需求进行分析。

对于老师用户，各学科的老师应具有创建考试的权限，并规定其中一些常见的考试属性，如考试时间，考试日期，考题分数，以及对应的考题。同时还应该具有对固定答案问题的自动判题功能，减少老师批改试卷的负担。老师还应可以查看考试详情，对一些非固定性答案的题目进行批阅。

对于学生用户，学生应可以查看当前所有考试的分数，考试时间等属性，并参与考试答题。在考试结束后，学生应该可以查看自己已经完成的考试，并查看对应的分数等属性。

2.2 功能需求分析

针对以上需求，将考试系统拆分为四个板块来实现，分别是登陆注册板块、考试大厅板块、考试板块和我的考试板块。

登陆注册板块提供学生与老师的注册功能，老师与学生用户应该具有不同的权限（目前还未完成这个功能，即权限开放），根据登陆的用户账号来划分权限。

考试大厅板块提供目前所有考试的展示，学生可以在这个界面参与考试，老师可以在这个界面创建新考试。

考试板块提供考试功能，学生在这个页面参与考试答题，在进入考试页面未完成考试之前，该页面具有防止切屏功能，可以在一定程度上防止学生作弊，学生还可以对每个题目进行标记，便于对不确定答案的题目进行二次检查。在考试限时结束或者学生点击提交试卷后，考试结束，并自动跳转到我的考试页面。

我的考试板块记录了每个学生用户参与过的考试，并且展示对应的考试分数等信息。可以点击右侧的详情来查看考试详情，考试详情会列出错误与正确的题目。

三：整体设计

3.1 技术设计

本项目是基于BS架构的在线考试系统，主要实现了用户注册登录、首页展示、考试列表展示、新建考试、参加考试、查看我的考试和个人中心七个功能。全程使用IDEA进行开发，并借助GitHub进行代码版本管理。

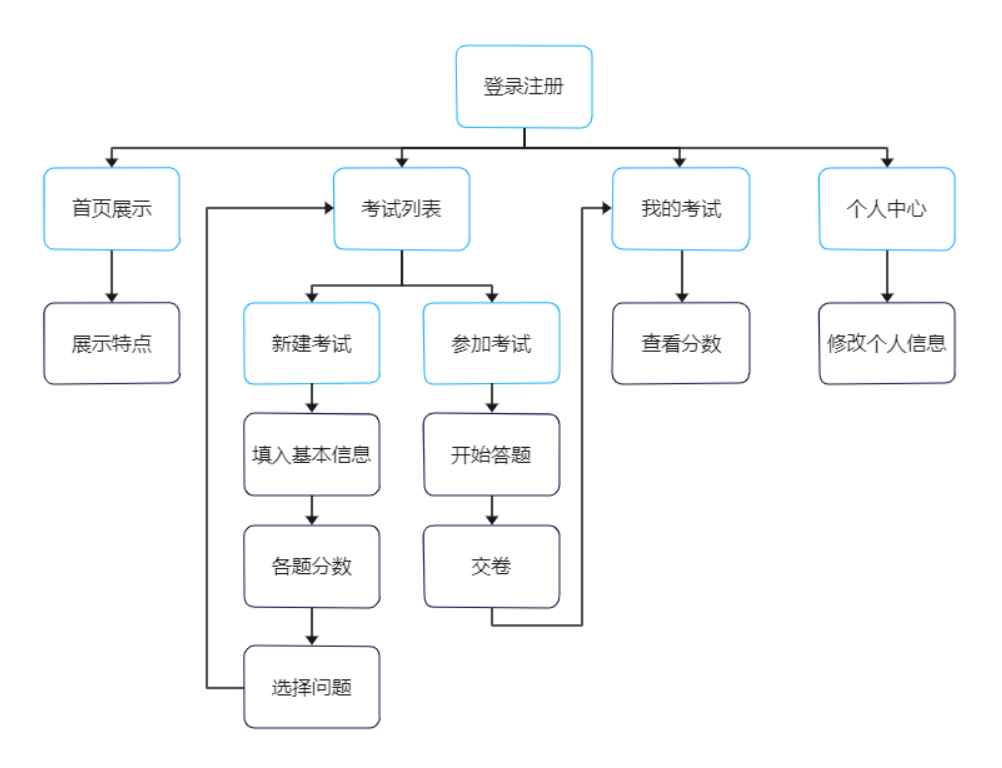
项目使用前后端分离的策略，通过特定接口进行前后端交互。

前端主要使用Vue cli脚手架作为框架，搭建用户交互界面，其中使用阿里巴巴Ant Design Vue库进行组件和样式的美化，借助vue-router进行页面的跳转，使用vuex和localStorage进行临时存储，并借助Vue 2语法完成前端工作。

后端主要使用较为流行的Java后端框架SpringBoot，选用开源的MySQL数据库作为系统数据库，并使用MyBatisPlus框架与数据库交互。使用maven作为依赖管理工具管理项目所需依赖。

3.2 功能设计

Online Exam系统整体功能架构图如下图所示。



系统主要有以下七个主要功能。

1. 登录注册

在未登录的状态下，系统会阻止游客访问，登录注册界面帮助用户进行考试系统的用户名和密码注册，并通过注册的信息进行登录，从而进入到考试系统。

1. 首页展示

首页为欢迎界面，用以展示网站主要功能和用户登录状态，并且通过网格化展示网站在设计用到的设计理念和构建技术，帮助用户更好的理解Online Exam考试系统。

1. 考试列表展示

在考试列表页面，用户可以查看当前已有考试，通过多页列表形式展示考试的名称、科目、难度、总分数和考试时长。并且在该页面，用户可以点击进行新建考试，也可以参加考试列表中的考试。

1. 新建考试

点击考试列表中的“新建”按钮，系统会弹窗开始新考试的创建流程。首先需要用户输入考试名称、考试限时、考试科目和难度等本场考试的基本信息，随后进入下一步，用户为考试中的单选题、多选题和判断题进行赋分，之后用户从问题列表中选择单选、多选和判断题作为考试题目，至此新考试创建完成。

1. 参加考试

用户点击考试列表中任意一个考试的“参加考试”按钮，可以进入到该项考试的答题页面。用户通过在左侧题目划分栏中选择题目编号跳转到相应题目，并进行答题操作，已答的题目会显示笑脸图标。用户可以进行题目标记操作。在答题过程中网页监测用户是否有切屏行为，并进行提示。在全部答完后，用户可以点击交卷按钮进行试卷的提交，系统在后端会进行判分。

1. 查看我的考试

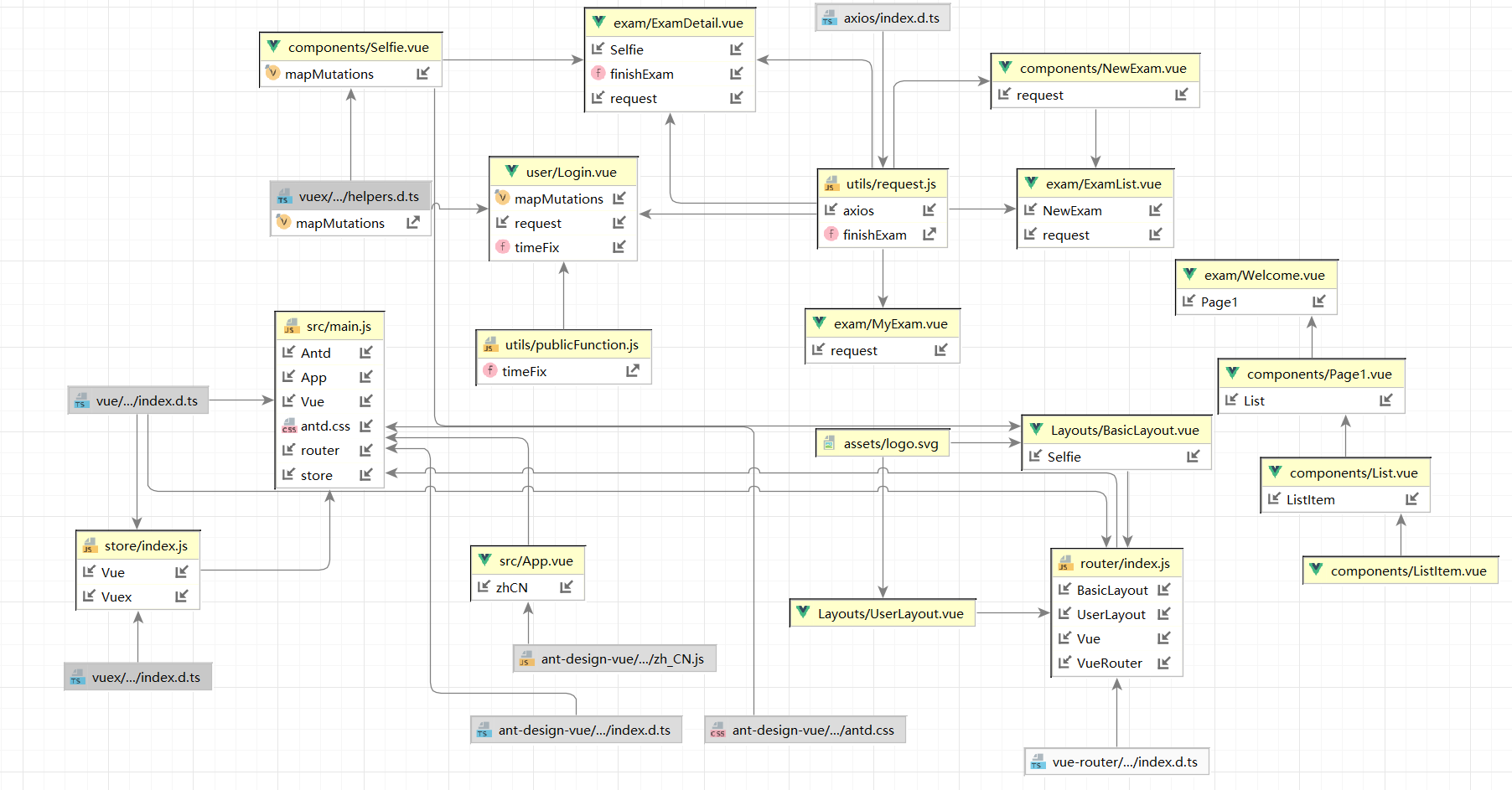
在参加完考试后，页面跳转到我的考试页面，这里用户可以查看已经参加过的考试并查看得分。网站支持多人答题，每个不同账号都可以查看到现有的所有考试，并且进行个人作答，进行完答题后可以在我的考试界面查看到用户自己的考试成绩。

1. 个人中心

点击每个页面的右上角小人图标，用户可以点击“账户设置”进入个人中心，修改个人信息。

四：前端搭建

前端整体使用Vue cli脚手架框架进行开发，使用模块化开发理念，创建整体布局与各子模块，资源文件整体结构如下。



下面按照通用结构和页面顺序对前端结构进行介绍。

1. 整体布局文件
   1. router/index.js

该文件使用vue-router插件库，实现前端页面的url的存储与跳转。主要设置有/welcome,/examList,/myList,/settings,/about,/user/login,/exam/:id等路由。并设置全局前置路由导航守卫，在跳转之前首先判断用户是否登录，即在localStorage中是否存在用户的token令牌，如果该用户未登录，则跳转至登录页面，并提醒用户进行登录。

* 1. store/index.js

该文件使用vuex，实现用户信息的存储与修改。在用户登录时调用Login函数，将用户的token令牌存储至vuex与localStorage中，以便之后进行判断。在用户退出登录或登录时间过长时调用Logout函数，清除vuex和localStorage中的数据。

* 1. utils/request.js

该文件使用axios，实现前端向后端发送请求。在每次前端需要请求后端数据，或与后端进行交互时，发送axios请求获取数据。在每次请求的请求头前加上该用户的token令牌，用以识别该用户。

1. 登录注册界面

在登录注册界面中使用UserLayout.vue文件创建整体背景，整体水平居中，并为中间插入登录注册版块提供<router-view>接口。Login.vue文件实现登录注册的主要功能，整体使用<a-tab-pane>控件，用以交换登录和注册界面，在每个选项卡下设置一个表单form，通过输入框读入数据。在输入时通过watch属性监视输入框内容，用以判断用户是否用户名密码均已输入，若没有，则登录注册键置灰色。当用户点击登录，若与后端数据库用户名密码一样则跳转至欢迎页，当点击注册，若用户名没有被注册且两次输入密码一致，则刷新页面。在登录注册界面还通过v-decorator属性设置了实时监测，要求表单必填项，对密码的长度，密码的强度进行判断并实时图形化显示，但是由于Vue相关文件结构使得最后的效果无法体现。

1. 首页

首页用来展示系统界面与系统亮点。使用BasicLayout.vue文件创建整体框架，上方使用<a-layout-header>内嵌<a-menu>创建横向导航栏，并使用Selfie.vue文件创建用户头像区域，用于用户管理。下方在Welcome.vue文件中实现，首先使用< a-carousel>走马灯控件创建左右轮播图，下方设置一个两行三列的亮点展示栏，主要使用模块化布局，将每一个小的亮点作为一个模块，通过props进行传值，使之显示不同内容。

1. 考试列表

考试列表页面通过ExamList.vue文件实现，主要使用<a-table>控件创建表格，数据由向后端发送请求获得，展示考试的名称、科目、难度、总分、时长等信息，其中内嵌新建考试和参加考试操作。

1. 新建考试

新建考试通过NewExam.vue文件实现，整体使用<a-modal>对话框浮层控件，将创建考试的界面浮动显示于考试列表界面之上，通过<a-steps>控件设置三步完成，并在下面通过v-show属性在不同步骤显示不同的表单界面，在表单内使用文本、数字、下拉选项等不同格式的输入框用以处理不同数据，最终将整个表单打包发送给后端，新建考试。

1. 考试答题页面

考试答题页面通过ExamDetail.vue文件实现。整体采用<a-layout>布局，划分顶栏和侧栏，顶栏显示考试名称、科目、限时等信息，提供交卷按钮。使用<a-layout-sider>在侧栏显示问题索引，使用v-for属性列出问题序号。在右侧内容区，每次根据所选题目序号，向后端发送请求获得当前题目内容。每次作答完一道题后将该题保存至已答列表answersMap，将answersMap中的题目在界面上标记为已答。每次标记一道题后将该题保存至已答列表starMap，将starMap中的题目在界面上标记为已标记。若在考试答题过程中进行切屏，网页通过对visibilityState的监测会进行弹窗提示，防止作弊。点击交卷，前端将答题情况打包发送给后端，由后端进行判分并跳转到我的考试界面。

1. 我的考试

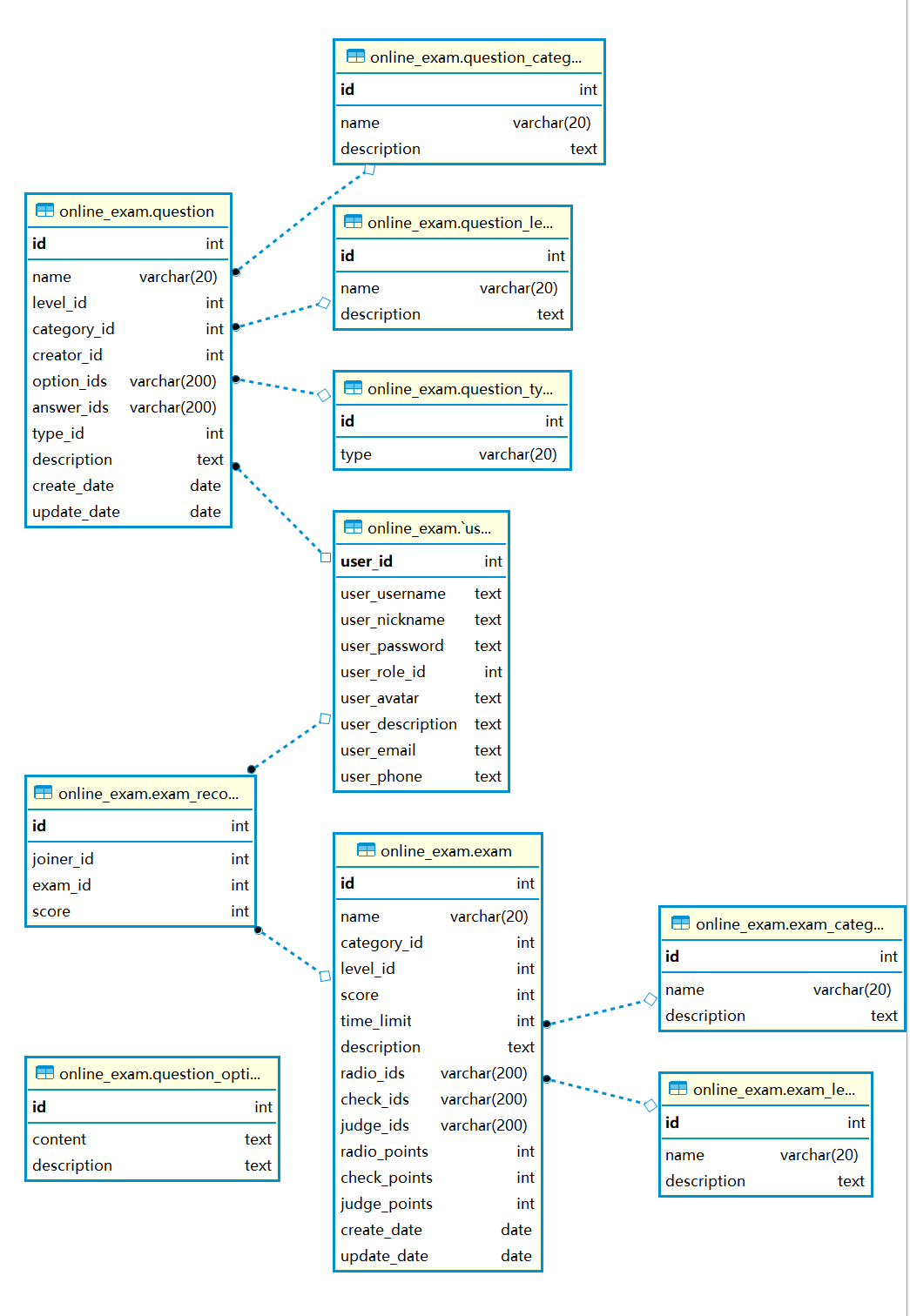
我的考试页面布局与考试列表基本一致，由MyExam.vue文件实现。

加页面

五：后端搭建

后端搭建主要包括数据库搭建与交互逻辑搭建两个部分。下面分别来讲述。

**5.1 数据库搭建**

数据库结构图如右：

本次实验使用了MySQL数据库，我们将数据库分为用户，考试，考题三大板块：

1. 用户

使用user表来存储所有的用户信息，包括用户名密码等，存储的密码经过SHA-256 +随机盐+[密钥](https://so.csdn.net/so/search?q=%E5%AF%86%E9%92%A5&spm=1001.2101.3001.7020)加密，保证了数据库的安全。其中user\_role\_id存储用户角色，系统为不同的用户提供不同功能。

1. 考试

使用exam表来存储考试信息，包括考试名称、考试难度、考试限时等信息。该表与exam\_category和exam\_level关联，exam\_category用于存储考试科目，exam\_level用于存储考试难度，通过关联表的方式提高了数据库的可扩展性。该表还以字符串的形式存储了考试题目对应的id，id通过”-”分隔后以字符串形式存储，在需要读取考试题目时，首先读取对应的字符串，然后根据”-”拆分成id数组后从question表中查询即可。在数据库的多处都通过存储id数组字符串的方式来实现一对多的数据关系。

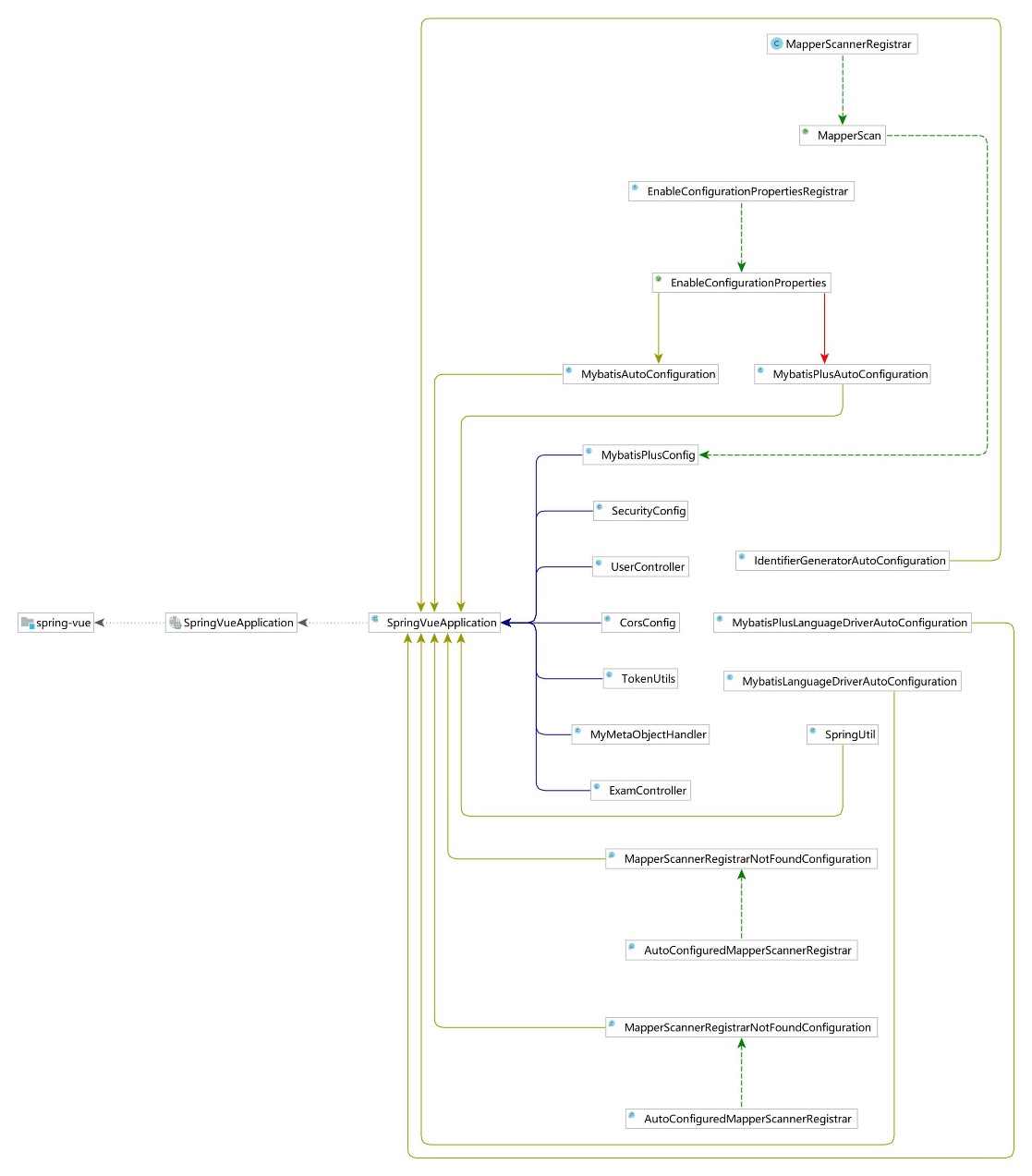
使用了exam\_record表来存储考试记录，即学生用户参加过的考试都会存储到这个表中，该表与exam和user表相关联，代表user中的用户参与了exam表中的哪个考试，当学生用户点击我的考试界面时，后端通过关联查询将相关的信息查询出来并传递给前端展示。

1. 问题

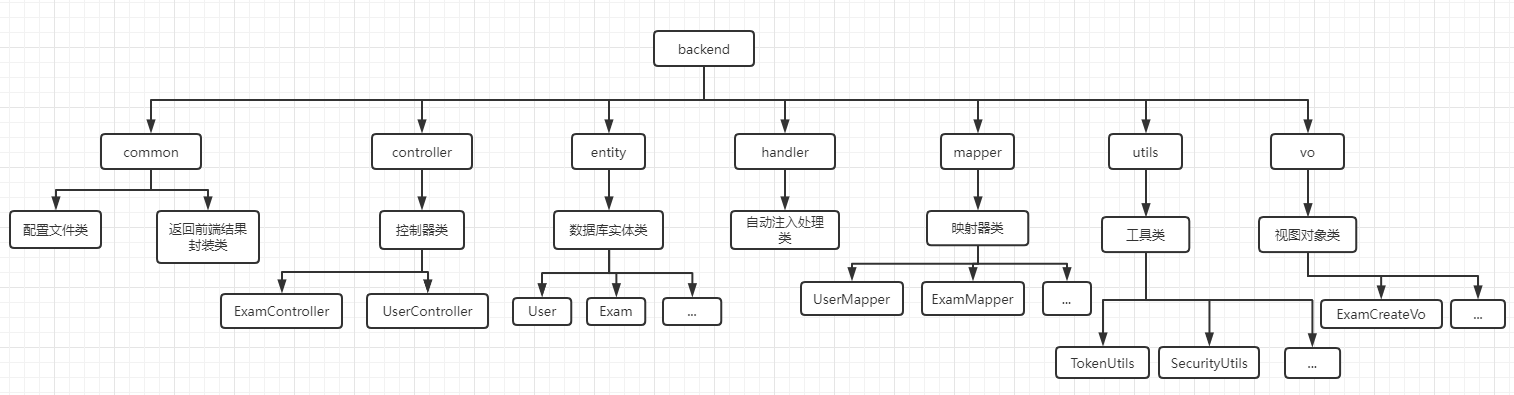
使用question表来存储问题信息，包括问题名称、问题科目、问题选项等。问题选项同样是存储了以”-”分隔的id的字符串，也存储了正确答案对应的id串，用于自动判分。该表与question\_category,question\_level,question\_type和question\_option关联，四个表分别用于存储问题所属科目，问题难度，问题种类与问题选项。通过对应的id与四个表相关联。同样也是使用关联查询进行id与信息的对应。

**5.2 交互逻辑搭建**

交互逻辑主要通过java spring boot + mybatisplus来实现。spring项目结构图如下：



项目文件层次图如下：



1. common

该文件夹下包含主要包含配置文件，在Spring中配置文件以java类的形式存在，进行了跨域配置、mybatisplus配置和springSecurity配置。如跨域配置如下：

|  |
| --- |
| @Configuration **public class** CorsConfig {   *// 当前跨域请求最大有效时长。这里默认1天* **private static final long *MAX\_AGE*** = 24 \* 60 \* 60;   **private** CorsConfiguration buildConfig() {  CorsConfiguration corsConfiguration = **new** CorsConfiguration();  corsConfiguration.addAllowedOrigin(**"\*"**); *// 1 设置访问源地址* corsConfiguration.addAllowedHeader(**"\*"**); *// 2 设置访问源请求头* corsConfiguration.addAllowedMethod(**"\*"**); *// 3 设置访问源请求方法* corsConfiguration.setMaxAge(***MAX\_AGE***);  **return** corsConfiguration;  }   @Bean  **public** CorsFilter corsFilter() {  UrlBasedCorsConfigurationSource source = **new** UrlBasedCorsConfigurationSource();  source.registerCorsConfiguration(**"/\*\*"**, buildConfig()); *// 4 对接口配置跨域设置* **return new** CorsFilter(source);  } } |

使用Configuration注解来将类配置到Spring中。

1. controller

该文件夹下包含控制器类，用于接收前端发送的请求，主要将控制器分为两个部分编写，分别是UserController类用于接收与用户有关的请求，如用户注册，用户登陆注销等；ExamController类用于接收与考试有关的请求，如创建考试，展示所有考试，展示我的考试等。在控制器中通过不同的注释接收get或者post请求，以及对应请求的参数。登陆对应接受函数如下：

|  |
| --- |
| @PostMapping(**"/login"**) **public** Result<?> login(@RequestBody User user) {  QueryWrapper<User> wrapper = **new** QueryWrapper<>();  wrapper.eq(**"user\_username"**,user.getUserUsername());  User res = **userMapper**.selectOne(wrapper);  String msg = **""**;  **if** (res == **null**) {  msg = **"不存在该用户名"**;  **return** Result.*error*(**"-1"**, msg);  }   **if** (!SecurityUtils.*matchesPassword*(user.getUserPassword(), res.getUserPassword())) {  msg = **"输入密码错误"**;  **return** Result.*error*(**"-1"**, msg);  }   *// 生成token* String token = TokenUtils.*genToken*(res);  res.setToken(token);  **return** Result.*success*(res); } |

注释化编写模式相对于使用原始的Servlet，解决了类爆炸的问题，并且减少了很多配置代码。

1. entity

该文件夹下存储了数据库实体类，对数据库中的每个表都建立对应的实体类，通过注释TableName使其相互关联。之后便可以通过创建类的对象来处理数据。如实体类User部分如下：

|  |
| --- |
| @TableName(**"user"**) @Data *// @Data注释自动为类生成get，set和构造函数* **public class** User {  *// 自增* @TableId(type= IdType.***AUTO***)  **private** Integer **userId**; |

1. handler

该文件夹下存储自动处理类，用于处理mybatisplus中自动插入的默认赋值或者动态默认赋值等。创建了MyMetaObjectHandler用于处理自动填充数据插入时间或者更新时间等。

1. mapper

该文件夹下存储mybatis映射类，该类的对象能够进行数据库的增删改查操作。理论上对于没个实体类都应该创建对应的映射类，但此实验中只创建了部分映射类，其余都使用注释的方式来实现关联查询，示例如下：

|  |
| --- |
| **public interface** ExamMapper **extends** BaseMapper<Exam> {  @Select(**"select e.id as serial,e.`name` as `name`,ec.`name` as `subject`,el.`name` as difficulty,e.score as score,e.time\_limit as duration from exam e,exam\_category ec,exam\_level el where e.category\_id = ec.id and e.level\_id = el.id;"**)  List<ExamVo> findAllExamVos(); |

这里采用了三表关联查询来获取前端所需数据。

1. utils

该文件夹下存放工具类，包括Token生成与解析类、密码加密与解密类和字符串解码类。这些类中的方法都以静态方法的方式编写，可以通过类名直接调用，而不需要先构建类的对象后调用。其中Token工具类如下：

|  |
| --- |
| @Slf4j @Component **public class** TokenUtils {   @Autowired  **private** UserMapper **userMapper**;  **private static** UserMapper *staticUserMapper*;   @PostConstruct  **public void** init() {  *staticUserMapper* = **userMapper**;  }   */\*\*  \* 生成token  \** ***@param user*** *\** ***@return*** *\*/* **public static** String genToken(User user) {  **return** JWT.*create*().withExpiresAt(DateUtil.*offsetDay*(**new** Date(), 1)).withAudience(user.getUserId().toString())  .sign(Algorithm.*HMAC256*(user.getUserPassword()));  }   */\*\*  \* 获取token中的用户信息  \** ***@return*** *\*/* **public static** Integer getUserId() {  **try** {  HttpServletRequest request = ((ServletRequestAttributes) RequestContextHolder.*getRequestAttributes*()).getRequest();  String token = request.getHeader(**"Authorization"**);  String aud = JWT.*decode*(token).getAudience().get(0);  **return** Integer.*valueOf*(aud);  } **catch** (Exception e) {  ***log***.error(**"解析token失败"**, e);  **return null**;  }  } } |

该类具有生成token与解码token两个方法，在控制器的中处理登陆请求时，若登陆成功则调用该方法将token传递给前端。

1. vo

该文件夹下存储视图类，这些类的实例用于接收前端传递过来的数据，并将数据以特定格式传递到前端，即完成了前后端的数据交互。由于数据库的存储方式与前端展示数据的方式不完全相同，因此编写一个视图对象类专门用于接收或者发送数据十分重要。例如在创建考试过程中，前端需要从后端获取考试难度、考试科目和所有问题，因此封装了ExamCreateVo来将对应的数据在后端包装后送到前端，ExamCreateVO编写如下：

|  |
| --- |
| @Data **public class** ExamCreateVo {  **private** Integer **radioScore**;   **private** Integer **checkScore**;   **private** Integer **judgeScore**;   **private** List<QuestionSelectVo> **radios**;   **private** List<QuestionSelectVo> **checks**;   **private** List<QuestionSelectVo> **judges**;   **private** String **name**;   **private** Integer **subject**;   *// 考试时间* **private** Integer **elapse**;   *// 难度* **private** Integer **level**;    **public** Integer getLevel() {  **if** (**level**== **null**)  **return** 2;  **return level**;  } } |

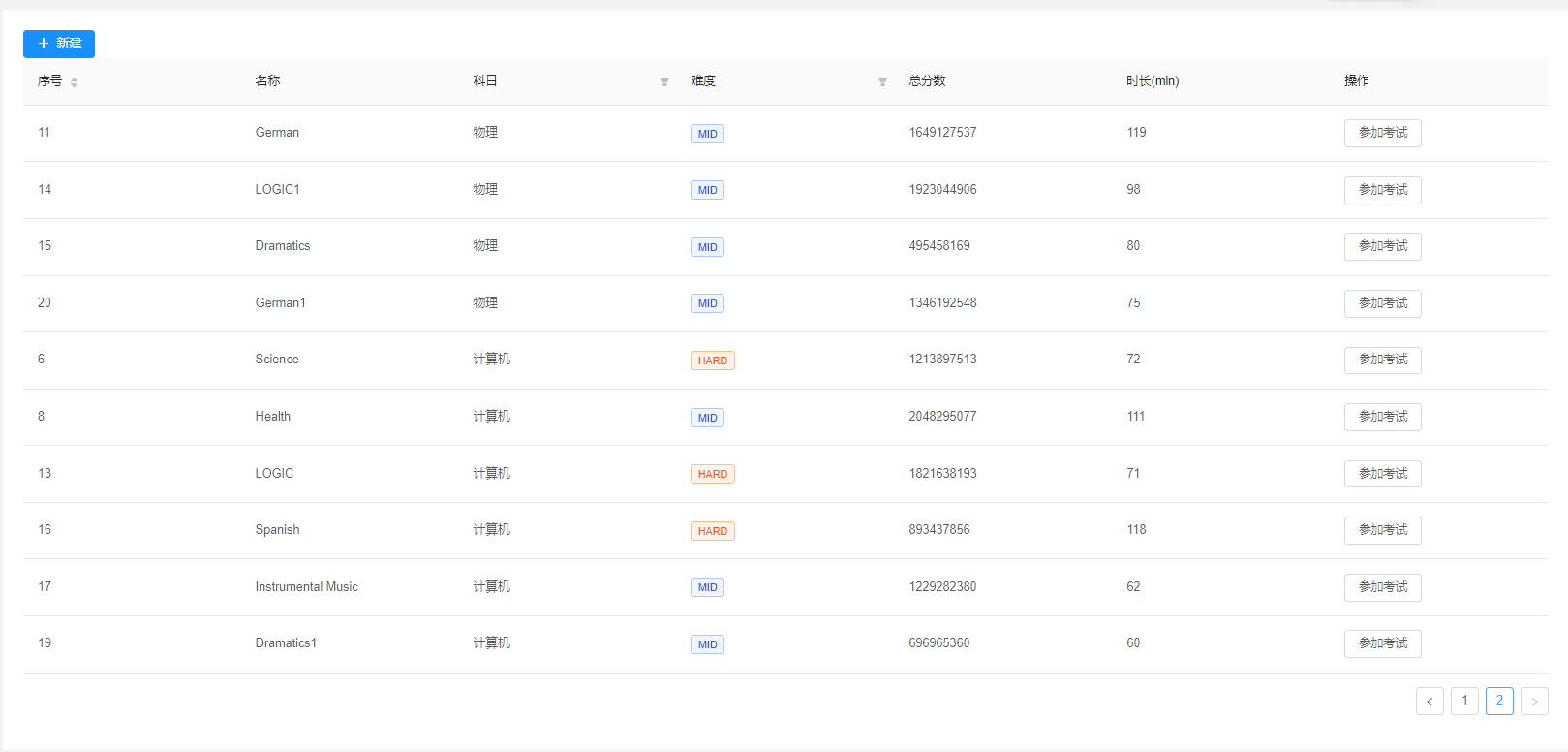
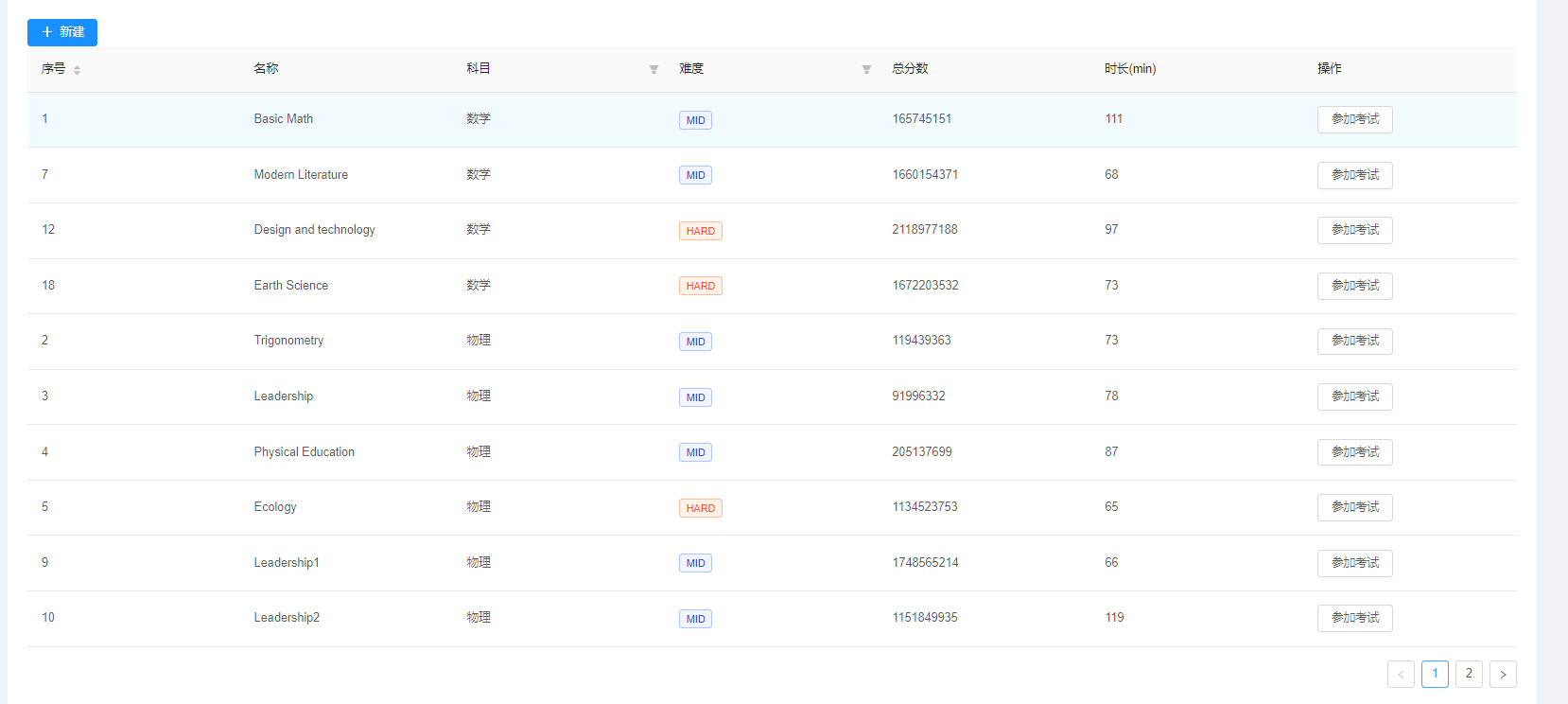
通过这种方式能够与前端完成流畅的数据交互，并且结构清晰。

六：测试数据

**6.1 考试大厅页面数据测试**

|  |
| --- |
| INSERT INTO exam(`name`,category\_id,level\_id,score,time\_limit) VALUES('Basic Math',1,2,100,111);  INSERT INTO exam(`name`,category\_id,level\_id,score,time\_limit) VALUES('Trigonometry',2,2,100,73);  INSERT INTO exam(`name`,category\_id,level\_id,score,time\_limit) VALUES('Leadership',2,2,120,78);  INSERT INTO exam(`name`,category\_id,level\_id,score,time\_limit) VALUES('Physical Education',2,2,120,87);  INSERT INTO exam(`name`,category\_id,level\_id,score,time\_limit) VALUES('Ecology',2,1,120,65);  INSERT INTO exam(`name`,category\_id,level\_id,score,time\_limit) VALUES('Science',3,1,100,72);  INSERT INTO exam(`name`,category\_id,level\_id,score,time\_limit) VALUES('Modern Literature',1,2,120,68);  INSERT INTO exam(`name`,category\_id,level\_id,score,time\_limit) VALUES('Health',3,2,121,111);  INSERT INTO exam(`name`,category\_id,level\_id,score,time\_limit) VALUES('Leadership1',2,2,122,66);  INSERT INTO exam(`name`,category\_id,level\_id,score,time\_limit) VALUES('Leadership2',2,2,121,119);  INSERT INTO exam(`name`,category\_id,level\_id,score,time\_limit) VALUES('German',2,2,121,119);  INSERT INTO exam(`name`,category\_id,level\_id,score,time\_limit) VALUES('Design and technology',1,1,122,97);  INSERT INTO exam(`name`,category\_id,level\_id,score,time\_limit) VALUES('LOGIC',3,1,211,71);  INSERT INTO exam(`name`,category\_id,level\_id,score,time\_limit) VALUES('LOGIC1',2,2,211,98);  INSERT INTO exam(`name`,category\_id,level\_id,score,time\_limit) VALUES('Dramatics',2,2,121,80);  INSERT INTO exam(`name`,category\_id,level\_id,score,time\_limit) VALUES('Spanish',3,1,121,118);  INSERT INTO exam(`name`,category\_id,level\_id,score,time\_limit) VALUES('Instrumental Music',3,2,121,62);  INSERT INTO exam(`name`,category\_id,level\_id,score,time\_limit) VALUES('Earth Science',1,1,121,73);  INSERT INTO exam(`name`,category\_id,level\_id,score,time\_limit) VALUES('Dramatics1',3,2,121,60);  INSERT INTO exam(`name`,category\_id,level\_id,score,time\_limit) VALUES('German1',2,2,121,75); |

插入数据后页面展示如下：



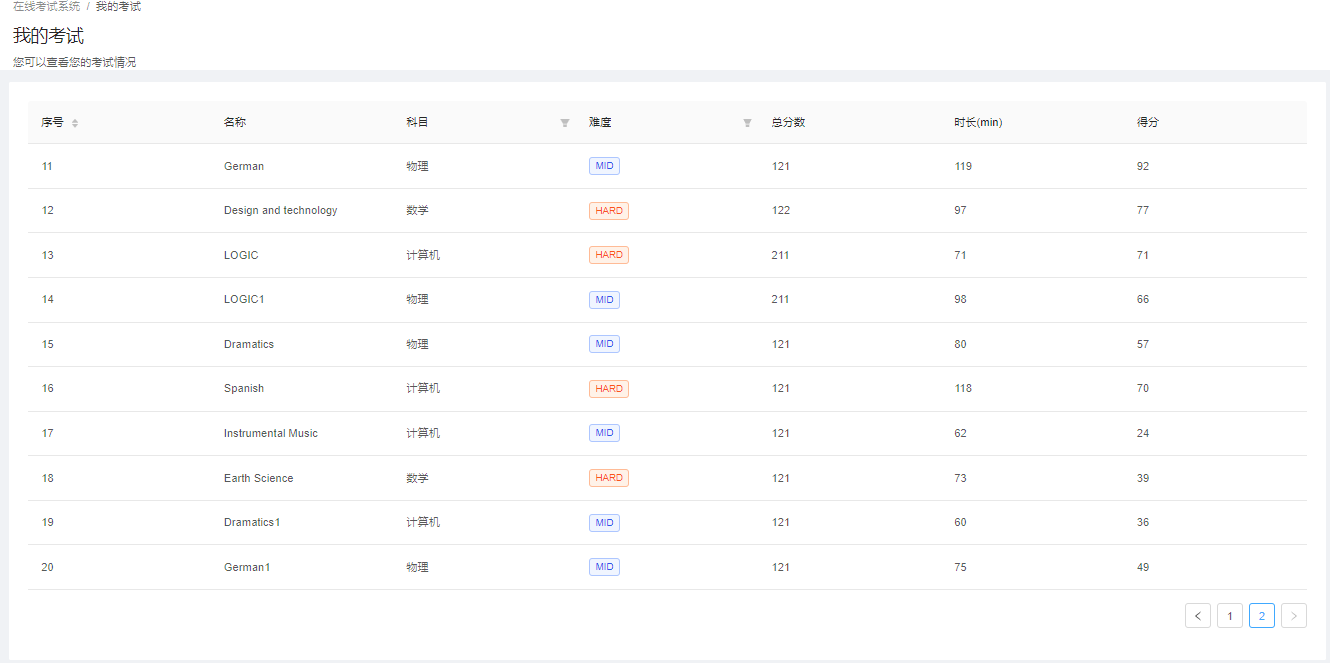
可以到展示正确，且分页展示正常。

**6.2 我的考试测试数据**

|  |
| --- |
| INSERT INTO exam\_record(joiner\_id,exam\_id,score) VALUES(1,1,22);  INSERT INTO exam\_record(joiner\_id,exam\_id,score) VALUES(1,2,59);  INSERT INTO exam\_record(joiner\_id,exam\_id,score) VALUES(1,3,1);  INSERT INTO exam\_record(joiner\_id,exam\_id,score) VALUES(1,4,25);  INSERT INTO exam\_record(joiner\_id,exam\_id,score) VALUES(1,5,45);  INSERT INTO exam\_record(joiner\_id,exam\_id,score) VALUES(1,6,94);  INSERT INTO exam\_record(joiner\_id,exam\_id,score) VALUES(1,7,82);  INSERT INTO exam\_record(joiner\_id,exam\_id,score) VALUES(1,8,34);  INSERT INTO exam\_record(joiner\_id,exam\_id,score) VALUES(1,9,95);  INSERT INTO exam\_record(joiner\_id,exam\_id,score) VALUES(1,10,2);  INSERT INTO exam\_record(joiner\_id,exam\_id,score) VALUES(1,11,92);  INSERT INTO exam\_record(joiner\_id,exam\_id,score) VALUES(1,12,77);  INSERT INTO exam\_record(joiner\_id,exam\_id,score) VALUES(1,13,71);  INSERT INTO exam\_record(joiner\_id,exam\_id,score) VALUES(1,14,66);  INSERT INTO exam\_record(joiner\_id,exam\_id,score) VALUES(1,15,57);  INSERT INTO exam\_record(joiner\_id,exam\_id,score) VALUES(1,16,70);  INSERT INTO exam\_record(joiner\_id,exam\_id,score) VALUES(1,17,24);  INSERT INTO exam\_record(joiner\_id,exam\_id,score) VALUES(1,18,39);  INSERT INTO exam\_record(joiner\_id,exam\_id,score) VALUES(1,19,36);  INSERT INTO exam\_record(joiner\_id,exam\_id,score) VALUES(1,20,49); |

我的考试前端页面展示如下：





数据展示与分页都正常。

**6.3 问题测试数据**

|  |
| --- |
| insert into question(`name`,type\_id,option\_ids,answer\_ids) values('完整的计算机系统应包括',1,'1-2-3-4','4');  insert into question(`name`,type\_id,option\_ids,answer\_ids) values('计算机硬件能直接识别的只有',1,'5-6-7-8','7');  insert into question(`name`,type\_id,option\_ids,answer\_ids) values('冯·诺依曼计算机中指令和数据均以二进制形式存放在存储器中， CPU 区分它们的依据是',1,'9-10-11-12','4');  insert into question(`name`,type\_id,option\_ids,answer\_ids) values('下列属于计算机硬件的是',2,'15-16-17-18','15-18');  insert into question(`name`,type\_id,option\_ids,answer\_ids) values('下列属于属于高级语言的是',2,'19-20-21-22','19-20');  insert into question(`name`,type\_id,option\_ids,answer\_ids) values('下列属于计算机外接设备的是',2,'23-24-25-26','23-24-25');  insert into question(`name`,type\_id,option\_ids,answer\_ids) values('同一个CPU 周期中，可以并行执行的微操作叫相容性操作',3,'13-14','14');  insert into question(`name`,type\_id,option\_ids,answer\_ids) values('就微命令的编码方式而言，若微操作命令的个数已确定，则直接表示法比编码表示法的微指令字长短 ',3,'13-14','13');  insert into question(`name`,type\_id,option\_ids,answer\_ids) values('相对于微程序控制器，硬布线控制器的特点是指令执行速度慢，指令功能的修改和扩展容易',3,'13-14','14');  insert into question\_option(content) values('运算器、存储器、控制器');  insert into question\_option(content) values('外部设备和主机');  insert into question\_option(content) values('主机和实用程序 ');  insert into question\_option(content) values('配套的硬件设备和软件系统');  insert into question\_option(content) values('高级语言');  insert into question\_option(content) values('自然语言');  insert into question\_option(content) values('汇编语言');  insert into question\_option(content) values('机器语言');  insert into question\_option(content) values('指令操作码的译码结果');  insert into question\_option(content) values('指令和数据的寻址方式');  insert into question\_option(content) values('指令周期的不同阶段');  insert into question\_option(content) values('指令和数据所在的存储单元');  insert into question\_option(content) values('是');  insert into question\_option(content) values('否');  insert into question\_option(content) values('CPU');-- 15  insert into question\_option(content) values('腾讯QQ');  insert into question\_option(content) values('微信');  insert into question\_option(content) values('显卡');  insert into question\_option(content) values('C++');-- 19  insert into question\_option(content) values('java');  insert into question\_option(content) values('汇编');  insert into question\_option(content) values('机器语言');  insert into question\_option(content) values('键盘');-- 23  insert into question\_option(content) values('鼠标');  insert into question\_option(content) values('音响');  insert into question\_option(content) values('CPU'); |

前端创建考试页面显示如下：



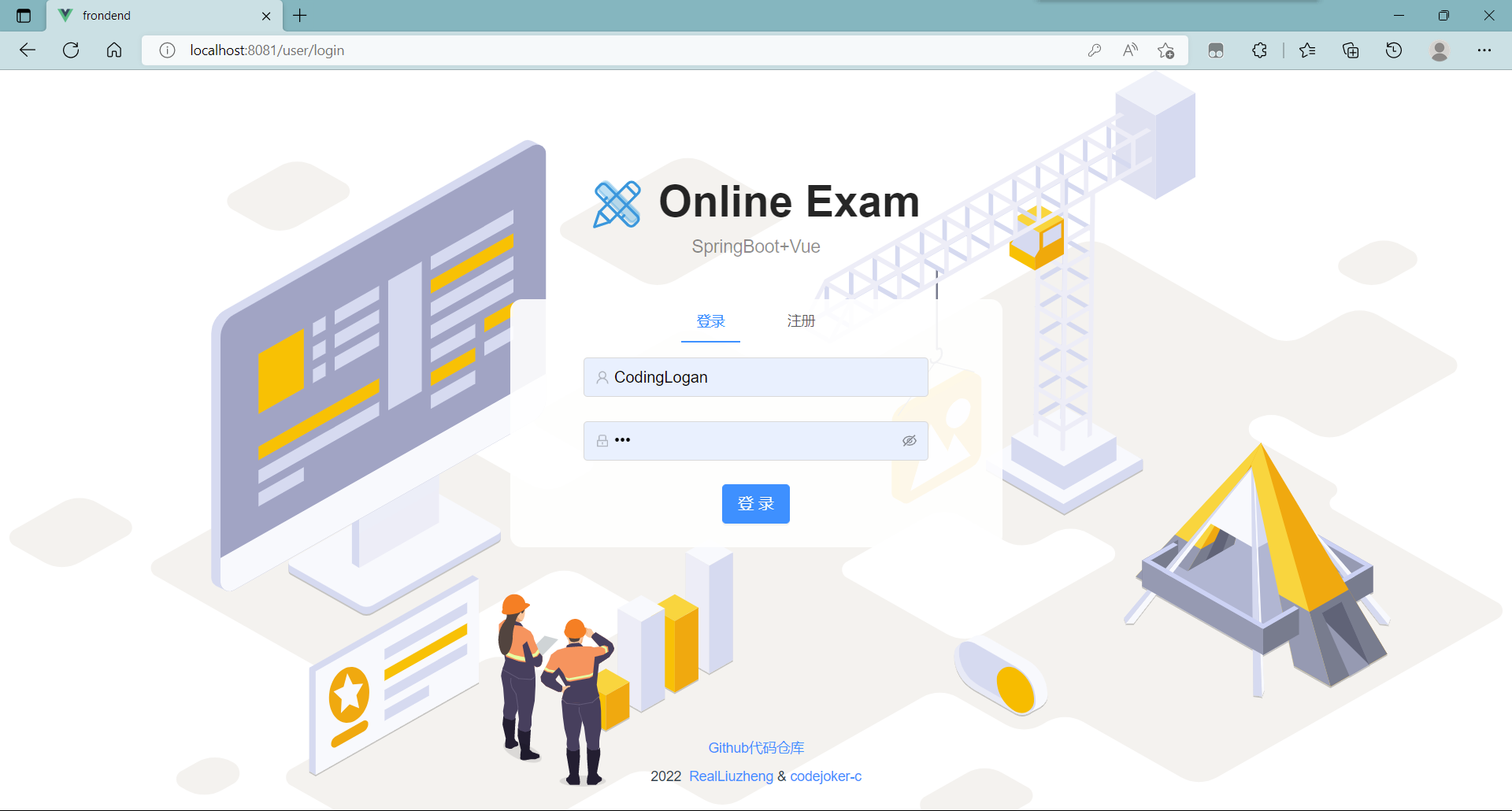
成功从数据库中获取对应的题目信息。考试界面如下：



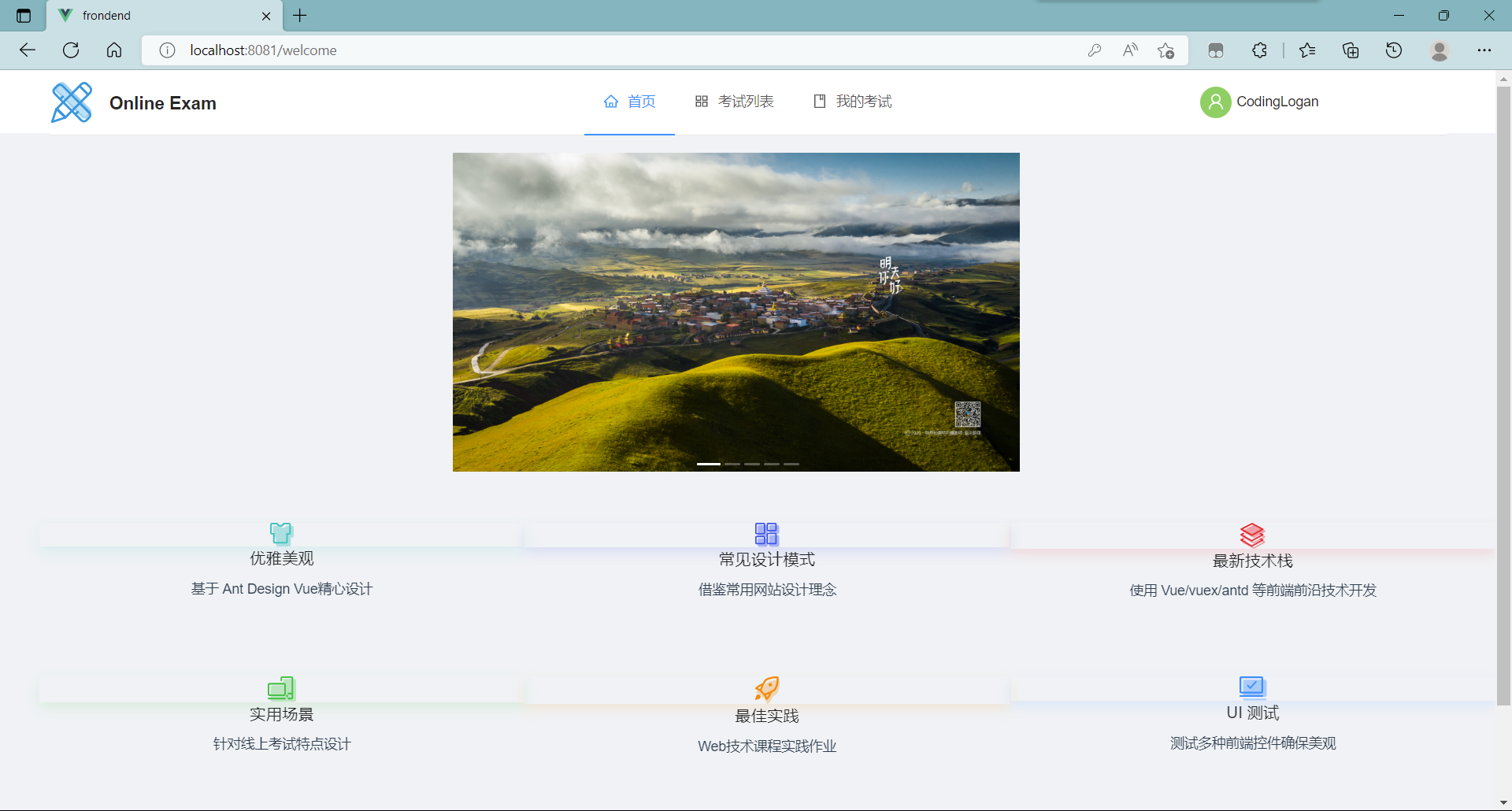
选项信息也能够正确获取。

七：实验结果展示

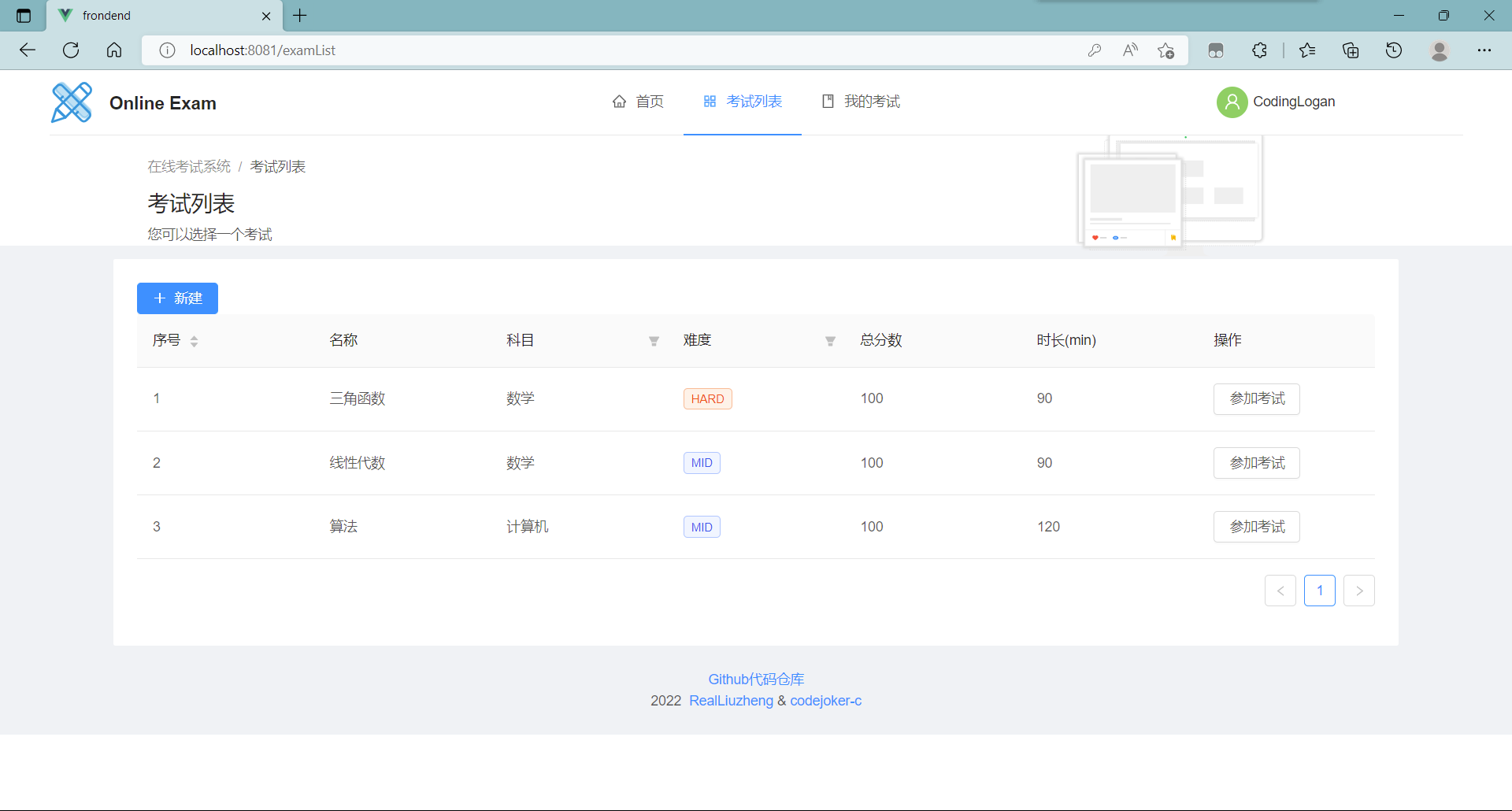
登录注册页面：



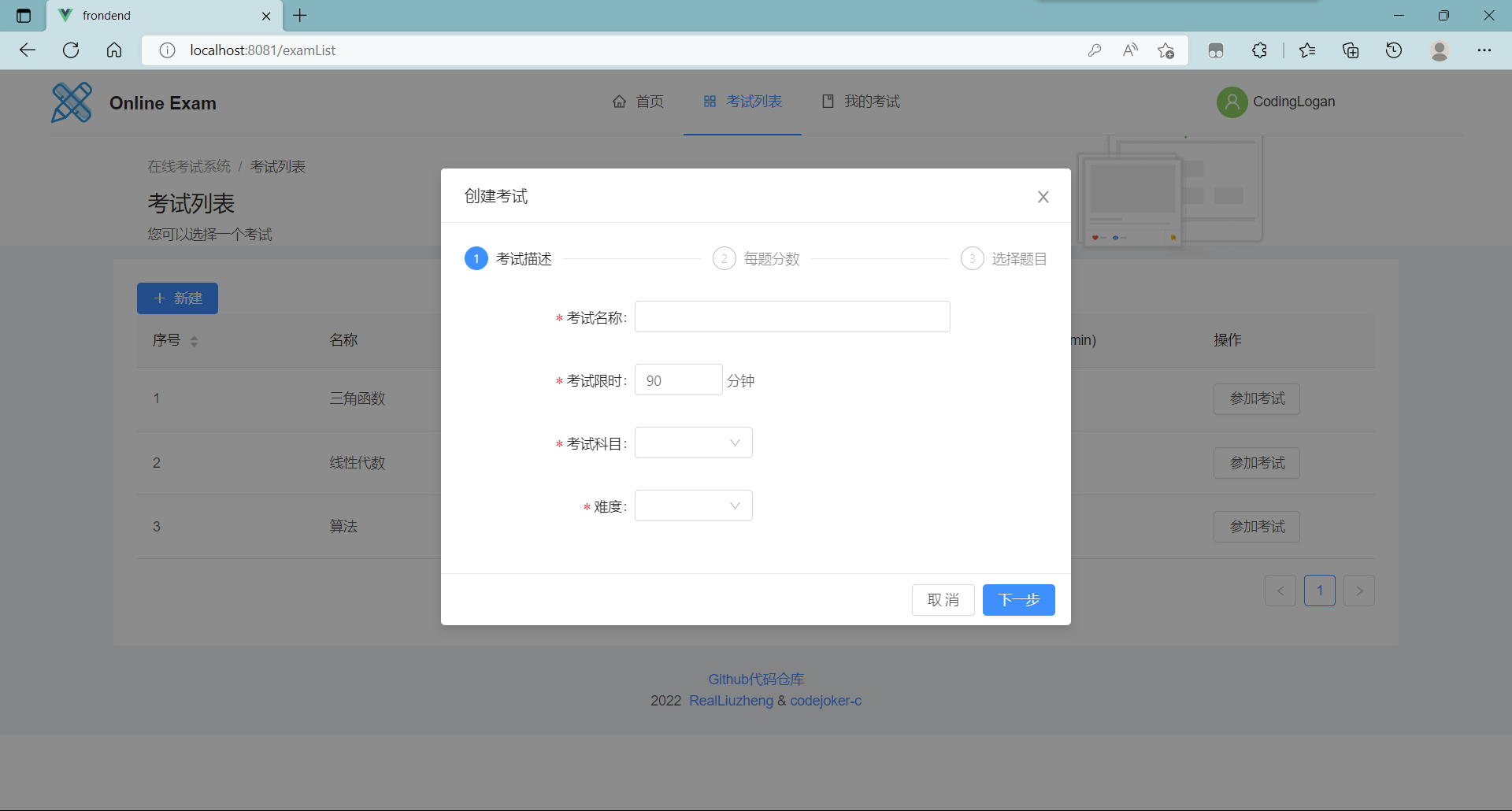
首页：



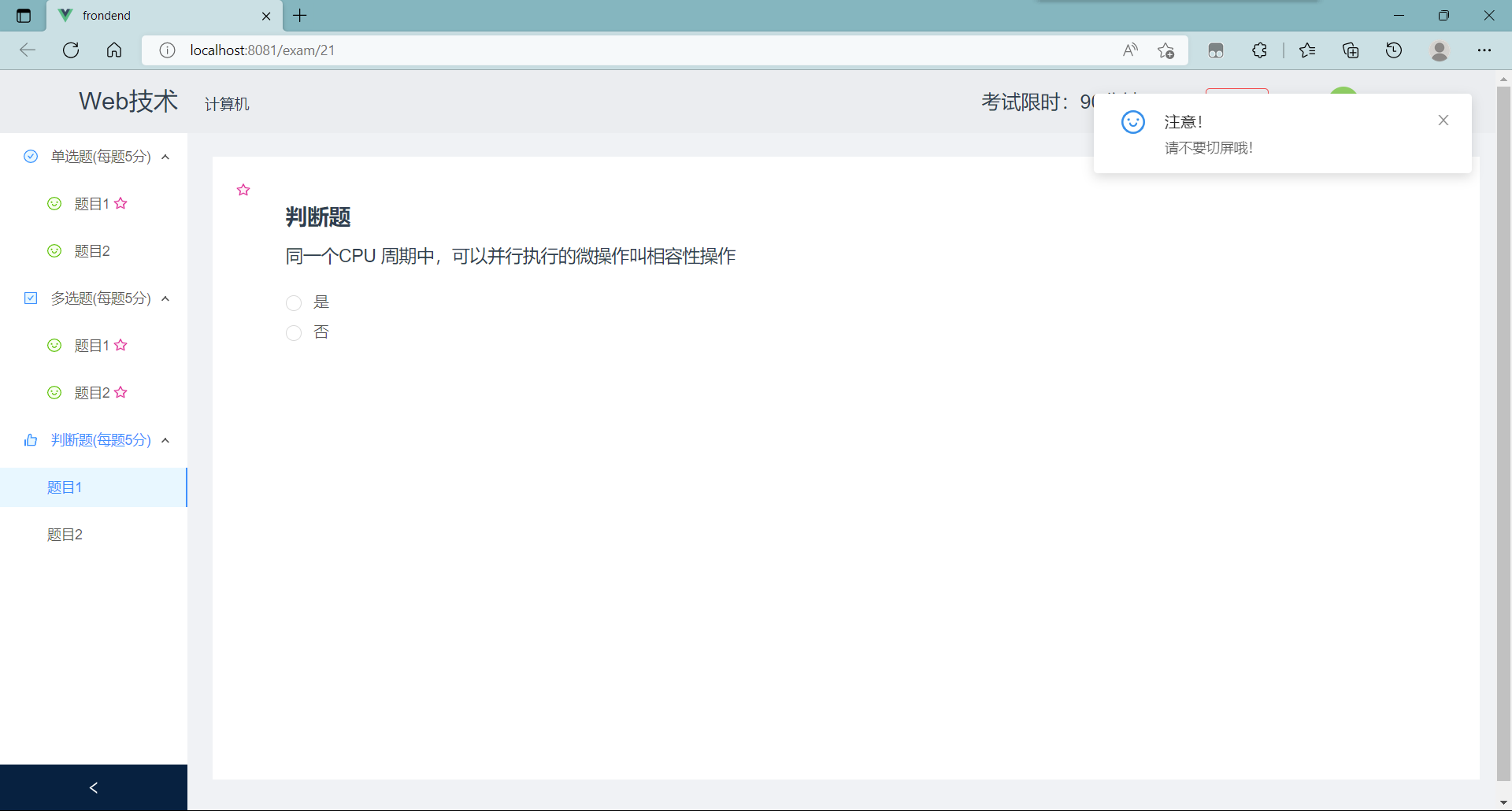
考试列表：



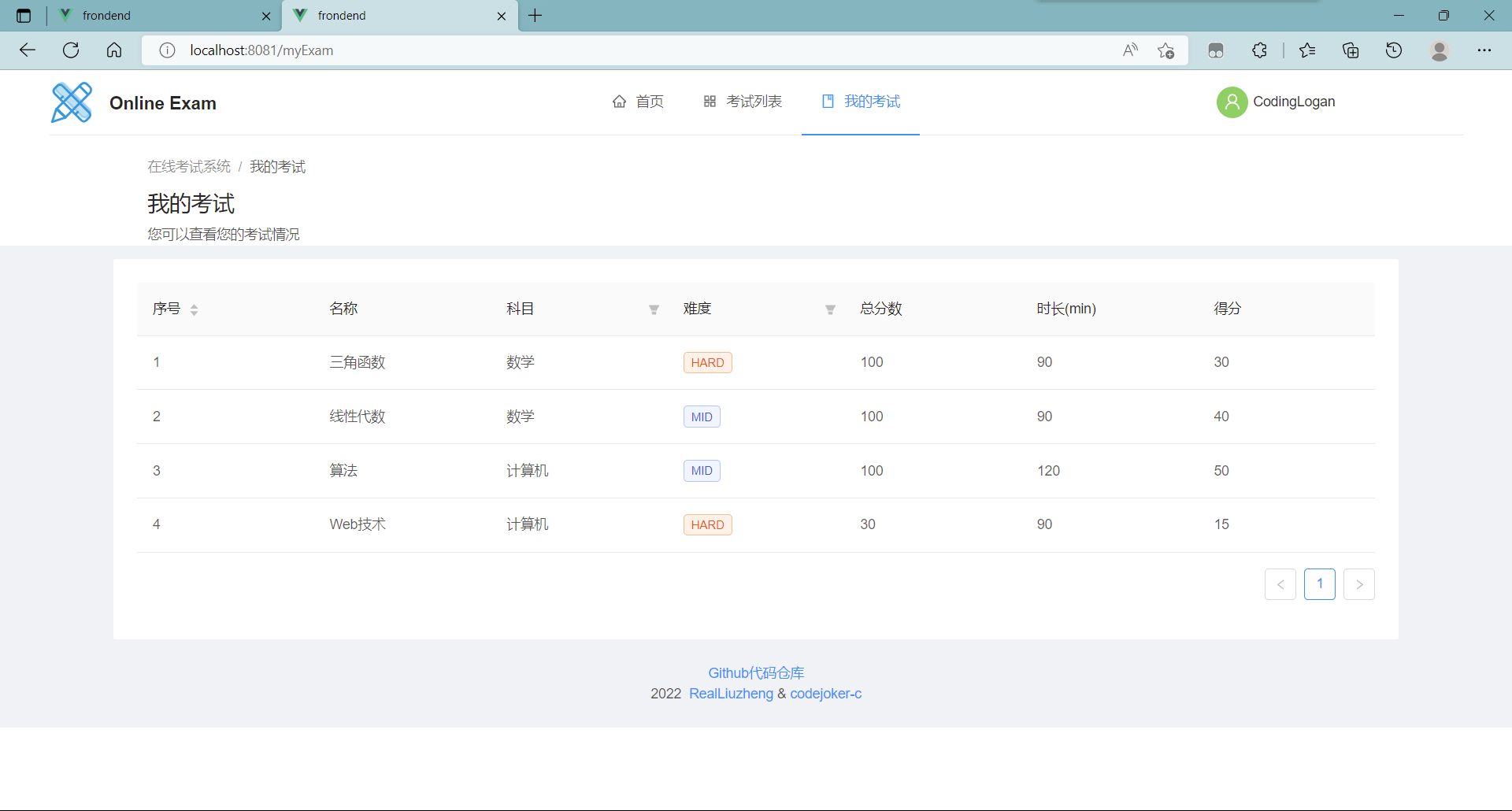
新建考试：



考试答题：



我的考试：



八：小组分工

在本次项目中，二人分工如下：

刘峥：负责前端展示页面的搭建与美化，处理与用户进行交互的过程，对数据进行展示和初步处理，与后端进行交互。

陈一鸣：负责后端框架的搭建，数据库的维护与处理，与前端进行数据交互，管理项目依赖。

团队二人全程共同参与、共同学习、共同完成，相互监督和修改完善项目，碰撞出了许多优秀的想法，解决了许多棘手的问题，建立了深厚的信任、友谊和团队感，同时对Web网页前后端的相关知识有了更加深刻的认知。

九：心得体会

本次实验中，我们使用了当前最为流行的web前后端开发框架Vue + SpringBoot + MybatisPlus开发了一个前后端分离的在线考试系统。开发过程中我们对网络的各种协议以及数据传输都有了十分深刻的理解，特别是数据传输部分，其在前后端分离中显得尤为关键。当前后端运行在不同的服务器上时，两者的耦合度大大下降，使得前后端开发人员能够专注于自己的工作，只需约定好数据接口即可，即前端通过向后端对应接口的url来请求数据，即发送请求报文，后端接受报文并发回响应报文。通过Http协议完成不同服务器之间的数据传输。

在项目的搭建过程中，我们首先学习了前后端主要知识，之后对项目整体进行布局，一起讨论考虑系统的主要功能，进行页面的划分和数据库结构的设计。之后前后端分别开始项目的搭建。整个实践过程中前后端不断交流，密切沟通，使得各自部分的搭建与前后端交互部分连接的效率都得到了较大的提升，最大限度的发挥了前后端分离开发的优势。

实验过程中我们还对Http的无状态性有了更加深刻的理解，为了实现用户登陆状态的保持，我们使用了token来保存用户的会话状态，并根据token给不同用户展示不同的信息。

视图对象在本次实验中也起到了至关重要的作用，由于前端页面展示的数据与数据库中存储的数据有较大差异，因此我们通过创建视图对象层将视图数据与数据库中的数据相关联，即通过关联查询从多个表中获取数据并填充到视图对象中，然后将视图对象传递到前端用于数据的展示；视图对象还能够接受前端的json数据，通过这种方式实现了前后端数据的顺利交互。

这次的项目是我们的第一个Web项目实践，让我们认识到Web技术的强大功能与广泛的适用性，为之后的Web项目开发打下基础。

十：web技术认识与体会

目前国内外web技术已经朝着微服务的方向开始发展，微服务的思想与前后端分离其实有些类似，它是将一个系统拆分为多个子系统，各个子系统之间相互独立，他们之间使用Http协议相互通信。子系统相互独立意味着其可以使用不同的编程语言，不同数据库来实现，这样针对不同的业务，就可以使用适合其的编程语言，从而避免了使用同一种语言开发带来的麻烦。微服务架构的一个重要特性就是代码的解耦合度高，对于一个子系统来说，功能业务的单一性使其容易维护，但是维护多个子系统较为复杂，因此也有如Eureka等服务集中管理框架用户对微服务的集中化管理。

单页面应用程序也是近些年来发展的主流，实际上Vue可以较好的实现单页面应用程序。单页面应用程序通过动态重写当前页面来与用户交互，避免了页面之间切换打断用户体验。国外有一些网站已经遵循了单页面应用程序的设计模式，如谷歌的云端硬盘。

上面谈到的两种web技术发展方向都具有一定的设计难度，特别是微服务架构，对于初次接触web开发项目来说较难实现。在本次web实验中，我们使用了目前较为流行的前后端分离开发模式，以合理的分工完成了本次项目。

不足之处在于我们没有很好的实现用户的权限分离，使得学生与教师浏览的页面相同，这对于一个考试系统来说还是存在一定的不足；由于时间原因，创建题目的页面也没有完成，导致题目只能够使用后端预置的数据，后续会在此处进行一定的完善。