

**《专业实习》**

**实习报告**

题 目： 在线教育平台

学生姓名： 王博伟

学 号： 20211120122

专 业： 计算机科学与技术

任课教师： 杨辉

评分（百分制）：

2025年1月

**目 录**

[**第一章 前 言 3**](#_Toc10035)

[1.1. 项目背景 3](#_Toc6628)

[1.2. 系统概述 3](#_Toc20146)

[1.2.1. 项目目标 *3*](#_Toc1171)

[1.2.2. 功能模块概述 *3*](#_Toc13633)

[1.2.3. 项目预期成果 *5*](#_Toc32434)

[1.3. 相关工作计划安排 5](#_Toc22688)

[**第二章 本人负责模块的需求分析 8**](#_Toc3587)

[2.1. 需求分析 8](#_Toc12930)

[2.1.1. 功能介绍 *8*](#_Toc3200)

[2.1.2. 基础信息管理 *8*](#_Toc10178)

[2.1.3. 业务办理及信息查询 *9*](#_Toc6969)

[2.1.4. 实时数据采集与显示 *9*](#_Toc28236)

[2.2. 示例用例 11](#_Toc13598)

[2.2.1. 更新实时信息 *11*](#_Toc1851)

[2.2.2. 管理人员信息 *12*](#_Toc10524)

[**第三章 本人负责模块的详细设计 13**](#_Toc5884)

[3.1. 模块概述 13](#_Toc15937)

[3.2. 功能设计 13](#_Toc29560)

[3.2.1. 数据库扩展功能 *13*](#_Toc6580)

[3.2.2. 前端显示功能 *13*](#_Toc16525)

[3.2.3. 答题提交功能 *14*](#_Toc32203)

[3.2.4. 评阅端口功能 *14*](#_Toc30053)

[3.2.5. 数据同步功能 *14*](#_Toc29323)

[3.3. 数据库设计 14](#_Toc14697)

[3.3.1. 主观题信息表 *14*](#_Toc11084)

[3.3.2. 选择题信息表 *15*](#_Toc15840)

[3.3.3. 判断题信息表 *16*](#_Toc23130)

[3.3.4. 填空题信息表 *17*](#_Toc8574)

[3.4. 系统界面设计 18](#_Toc27337)

[**第四章 本人负责模块的编程实现 21**](#_Toc5486)

[4.1. 项目结构 21](#_Toc6625)

[4.2. 关键代码及说明 22](#_Toc29885)

[4.2.1. 前端试题显示功能实现代码 *22*](#_Toc29740)

[4.2.2. 教师端口主观题评阅功能实现代码 *32*](#_Toc9239)

[4.2.3. 学生端试卷信息显示与答案处理功能实现代码 *41*](#_Toc4319)

[4.2.4. 教师端主观题答案获取与分数上传功能实现代码 *43*](#_Toc32606)

[**第五章 本人负责模块的测试与功能展示 45**](#_Toc22025)

[5.1. 测试 45](#_Toc27011)

[5.2. 功能展示 45](#_Toc11985)

[**第六章 实训总结 52**](#_Toc19745)

# 前 言

## 项目背景

随着互联网技术的快速发展和在线教育模式的普及，教育资源的数字化和便捷化已成为现代教学的重要方向。然而，现有在线教育平台普遍存在功能单一、交互性不足、用户体验不佳等问题，无法充分满足师生的多样化需求。为此，本项目旨在设计并实现一个功能完善、用户体验优良的在线教育平台，涵盖教师端的课程创建与管理、视频上传、作业布置和考试评阅，学生端的课程学习、作业完成和互动交流，以及后台的用户管理、数据统计等功能。通过采用现代技术架构（如微服务架构、分布式存储、实时消息推送等）和优质的视频处理方案，项目力求打造一个稳定、高效、可扩展的在线教育平台，为教师与学生提供便捷、丰富的教学与学习体验。

## 系统概述

### 项目目标

本人在该项目中的设计目标是设计并实现一个支持在线考试功能的高效平台，重点解决考试管理、答题体验和评阅效率等问题。通过开发限时答题、自动判卷、主观题手动评阅等功能，确保考试形式的多样性和灵活性，同时通过防作弊机制（如切屏限制、禁止复制粘贴）提高考试的公平性和严肃性。平台支持教师灵活设置考试内容、答题时间和评分标准，并可实时查看学生考试进度和成绩分布；学生端提供流畅的在线答题体验，支持保存答题进度和提交结果。结合后台数据统计功能，平台还将为管理员提供考试相关数据的分析与管理能力，以满足教学评估和持续优化的需求。

### 功能模块概述

1、课程管理模块

该模块主要用于教师对课程的创建、编辑、发布、管理等操作。教师可以通过系统创建新的课程，填写课程名称、简介、封面图片、章节内容等信息，并上传课程视频。每门课程可设置不同的章节内容，支持文字、图片和视频的形式展示课程知识点。课程上线后，教师可以管理课程的状态（上线、下线），根据需要随时编辑或删除课程内容。此模块还包括课程的分类管理，教师可以对课程进行学科、难度等方面的分类，方便学生根据需求进行筛选和选择。主要使用 VUE和JS来创建动态表单和视频上传界面，提供丰富的交互体验。同时使用Spring Boot 用于开发课程管理功能，使用 MySQL 存储课程相关数据，如课程名称、章节内容等。视频资源通过 MinIO 或 FastDFS 存储，确保高效、可靠的文件存储和访问。

2、考试模块

该模块为教师提供布置作业和考试的功能。教师可以根据课程内容，创建在线考试，设置考试的题目（选择题、填空题、判断题、主观题等）、答题时间、分值和评分标准。学生可以在限定的时间内在线完成作业或考试，并提交答案。系统会自动对选择题、判断题等客观题进行判卷，评分结果实时反馈给学生。对于主观题，教师可以手动评分，并为学生提供评语。成绩统计与分析功能可帮助教师和管理员了解每个学生的学习情况、错题分布以及整体考试成绩。通过 JavaScript 实现考试计时、答题过程控制与交互逻辑。前端界面使用 React.js 提供动态题目呈现与答题界面。另外Spring Boot 提供题库管理、考试生成、判卷等功能，自动评分通过算法实现，支持手动评阅主观题。成绩和数据将存储在 MySQL 数据库中，并通过 API 接口提供给前端。

3、用户管理与互动模块

该模块主要涉及教师和学生的注册、登录、角色管理及权限分配功能。学生和教师通过平台进行注册，系统会根据用户角色（教师、学生）自动分配不同的权限。教师可以创建、管理课程、布置考试，学生可以报名学习课程并参与互动。此外，系统还提供了实时互动功能，包括课程讨论区、提问和答疑等。学生可以在讨论区提问，教师或其他学生可以回复；教师还可以发布课程通知、发布答疑内容等。

4、视频处理与播放模块

该模块负责处理平台上传的课程视频，确保视频格式的兼容性以及优化视频播放体验。平台支持多种视频格式的上传，使用 FFmpeg 工具进行视频格式转换和码率调整，以适应不同网络环境下的播放需求。同时，视频支持断点续播、流畅播放。学生在观看视频时，可以使用倍速播放、视频进度记忆和笔记功能，使学习过程更加高效和便捷。视频处理通过 FFmpeg 实现，完成视频格式转换、码率调整和切片处理。视频存储使用 MinIO 或 FastDFS 等分布式存储系统。在前端通过 Video.js 插件实现视频播放，提供倍速播放、进度记忆等功能，支持在视频播放过程中插入笔记。通过前端与后端的协作，确保视频数据流的顺畅传输和播放，视频文件的存储与访问将得到优化。

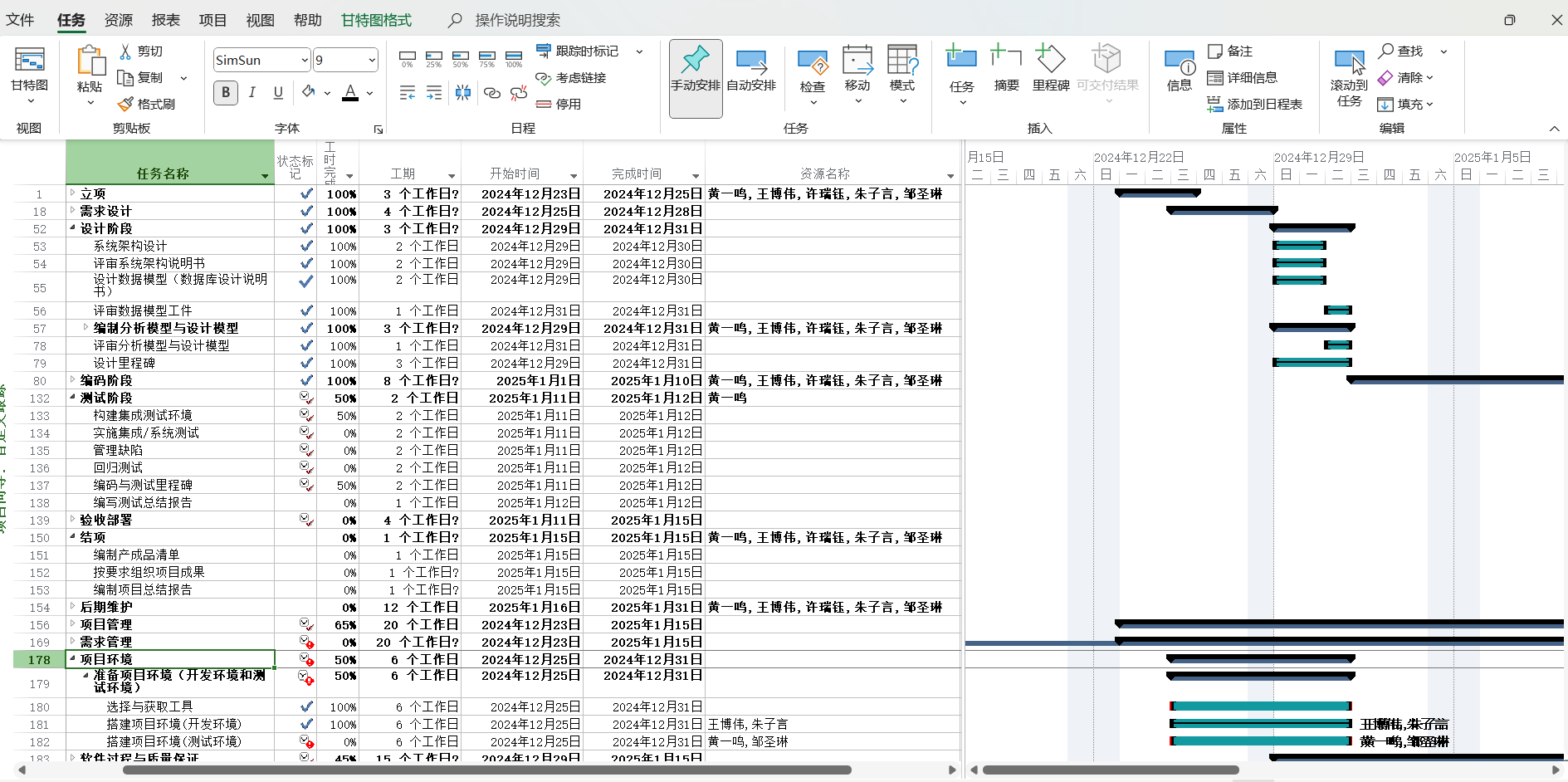
### 项目预期成果

本项目的预期成果是开发一个功能完备、稳定高效的在线教育平台，能够为教师、学生和管理员提供一体化的教学与学习体验。具体成果包括完整的课程管理功能，教师可以轻松创建、编辑、发布课程，并上传视频和其他教学材料，课程内容可以按章节进行管理，支持视频、文本等多种形式的课程内容呈现，提升教学效率；高效的考试与作业管理系统，平台提供了在线考试、作业布置和自动评分功能，教师能够灵活设计各种题型，学生可以实时在线答题，系统自动评分，教师可以对主观题进行手动评分，并查看成绩分析报告；实时互动与讨论功能，通过集成实时消息推送和讨论区，教师和学生能够即时互动，学生可以提问、讨论，教师可以发布通知和解答问题，从而提高课堂的互动性和学生的学习参与度；强大的视频播放与处理能力，平台能够支持多种格式的视频上传、转换与处理，确保不同网络环境下的流畅播放，同时提供视频进度记忆、倍速播放和视频笔记功能，帮助学生提高学习效率；便捷的用户管理与权限控制，通过角色管理，系统能够区分教师、学生和管理员的权限，确保每个用户能够根据权限访问相应的功能，用户注册、登录、课程报名、考试参加等功能都能够顺利进行；完善的数据统计与分析功能，为教师和管理员提供全面的课程学习进度、成绩分析、学习完成率等统计功能，支持课程的优化与调整，同时为平台运营提供决策支持。最终，项目将实现一个集课程管理、考试与作业、互动交流、视频播放、数据统计等功能于一体的在线教育平台，满足不同用户的需求，并为在线教育的发展提供技术支持。

## 相关工作计划安排

|  |  |
| --- | --- |
| 姓 名 | 角 色 |
| 许瑞钰 | 项目经理、需求分析员、程序员 |
| 朱子言 | 需求分析员、系统设计员、程序员 |
| 黄一鸣 | 需求分析员、系统设计员、程序员 |
| 邹圣琳 | 质量和产品保证、需求分析员、程序员、测试员 |
| 王博伟 | 需求分析员、配置管理员、程序员 |

甘特图如下：



项目组成员组成与互评

项目组成员组成、职责、互评表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 学号 | 姓名 | 职责 | 评价 | 互评打分 |
| 20211120067 | 许瑞钰 | 需求分析，项目整体规划 | 优秀 | 100 |
| 20211150019 | 朱子言 | 程序开发、布置后端架构 | 优秀 | 100 |
| 20211910103 | 黄一鸣 | 需求分析，前后端项目开发 | 优秀 | 100 |
| 20211060167 | 邹圣琳 | 产品与质量保证 | 优秀 | 100 |
| 20211120122 | 王博伟 | 前端页面开发，后端接口设计 | 优秀 | 100 |

# 本人负责模块的需求分析

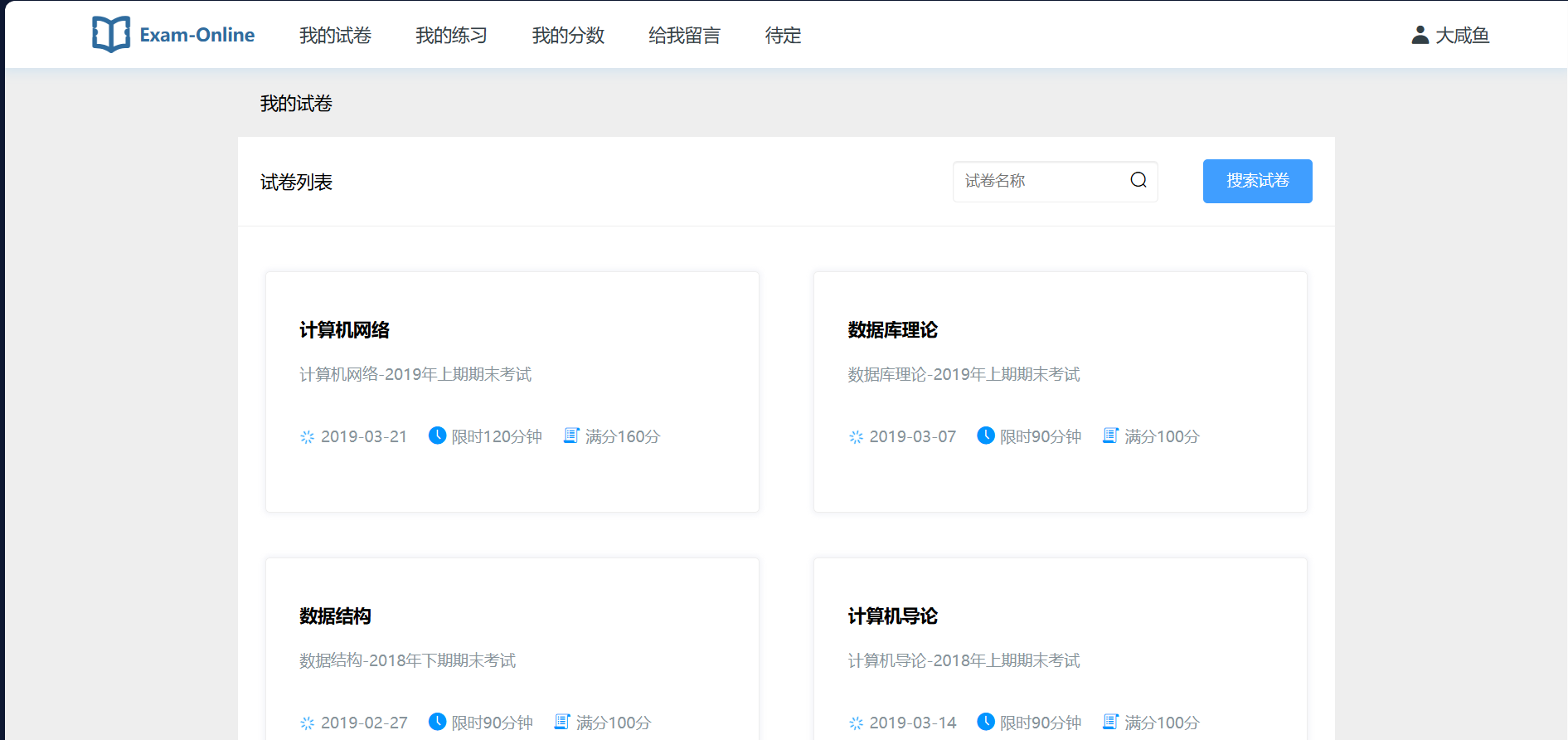
## 需求分析

### 功能介绍

本人负责的主要项目功能为对于学生端的试卷的设置与处理，试卷模块主要分为四大部分，分别是选择题、填空题、判断题和主观题。学生在其页面打开需要完成的试卷后，页面将会开始实现试卷的内容传递，同时答题倒计时也将会启动，用户在规定的时间内完成其试卷内容，同时提交试卷，之后，试卷的客观题部分（包括填空题）将会由系统自动完成分数审阅，而主观题部分将会被送到数据库中，在教师端开辟一个评阅端口，教师将会在该端口上进行主观题分数的评阅，最后提交分数，将主观题分数更新在该学生的系统自动生成的客观题分数上。最后，该学生的该门试卷分数将会呈现在该学生的成绩查询模块以及教师的学生成绩管理模块上。

### 基础信息管理

图形用户界面

描述已自动生成

上述四幅图分别显示了学生端口的学生对于自身登陆密码的修改，以及学生端口处的试卷信息管理。同时在教师端口处，教师可以查询学生的具体信息，以及查看对应学生的主观题内容，并且给出分数。图形用户界面, 应用程序

描述已自动生成图形用户界面

描述已自动生成

### 图形用户界面, 文本, 应用程序 描述已自动生成图形用户界面, 应用程序 描述已自动生成业务办理及信息查询

上述两幅图片分别显示了教师对于主观题的评阅，同时教师可以给出主观题评阅分数，而右图则显示为对应试卷的详情部分。

### 图形用户界面, 文本, 应用程序, 聊天或短信 描述已自动生成实时数据采集与显示

上图显示了整个试卷的内部信息，包括四个部分，选择题，填空题，判断题和主观题，同时左侧显示了题号，右侧为题目的具体内容，也是学生的作答区域。图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成

## 示例用例

### 图形用户界面, 文本, 应用程序, 网站 描述已自动生成图表, 折线图 描述已自动生成图表, 饼图 描述已自动生成更新实时信息

上述图片显示了学生在提交试卷作答后并且教师完成主观题的评阅之后，其该门试卷的整个成绩会在各个模块进行显示，学生可以自行检查，同时教师也可以管理成绩。

### 图形用户界面, 应用程序 描述已自动生成图形用户界面, 应用程序 描述已自动生成图形用户界面, 应用程序 描述已自动生成图形用户界面, 表格 描述已自动生成管理人员信息

上述图片显示了在管理员页面下其可以实现的具体功能，包括对于学生和教师的信息管理以及关于课程，考试的一些安排和处理。

# 本人负责模块的详细设计

## 模块概述

本模块主要实现试卷的设计，试卷部分由于时间与精力有限，只针对了一套试卷，试卷部分由多人分工完成，我的主要部分是主观题的部分，一方面是在已有的客观题基础上加上主观题的前端显示。这里先需要在数据库sql表中加上主观题的内容，再通过前端的设计将其传递到前端页面上，因此我主要实现将主观题的各个部分先显示在前端页面上，之后再通过JS文件等配置实现其动态调整。而在用户实现对于试卷的作答后，整套试卷的客观题部分会根据前端的逻辑直接进行评阅，而客观题成绩将会被写入对应的数据库表。而另外需要将主观题信息写入数据库，并且同时在教师端开辟一个新的端口，即试卷评阅端口，将主观题答案传递到该处。同时教师进行打分，并且将分数同步更新到该生对应的成绩数据库中。

## 功能设计

### 数据库扩展功能

为了实现试卷中主观题的完整展示与处理，需要对现有数据库结构进行扩展。具体而言，新增一个主观题表，用于存储主观题的基本信息，包括主观题编号、题干内容、参考答案、分值和题目类型等字段。同时，还需要设计一个关联表，用于记录学生的作答内容和对应的题目编号，从而实现主观题内容与学生答题记录的关联。这一设计确保了主观题部分的数据存储完整性，为后续的数据处理和交互提供了坚实的基础。

### 前端显示功能

主观题的前端展示功能是试卷设计模块的重要组成部分。通过调用后端接口获取主观题数据，将其动态加载到试卷页面中，实现与客观题部分一致的页面布局和视觉效果。为了提升用户体验，可以采用模板渲染引擎或现代前端框架（如Vue.js或React）进行开发，使主观题能够根据试卷结构进行动态调整。同时，设计良好的样式和布局，使主观题与客观题自然融合，确保试卷整体的美观性与逻辑性。

### 答题提交功能

学生在完成主观题作答后，需要将答案提交到后端进行存储。为此，开发了一个答题提交接口，支持逐题提交和多题批量提交两种模式，确保学生的作答能够灵活、高效地上传到服务器。前端通过AJAX请求或表单提交的方式传输学生的答案，后端接收并验证数据后，将答案写入数据库对应的表中。此功能设计中重点考虑了数据传输的安全性和高效性，以保证作答内容不会因网络问题而丢失。

### 评阅端口功能

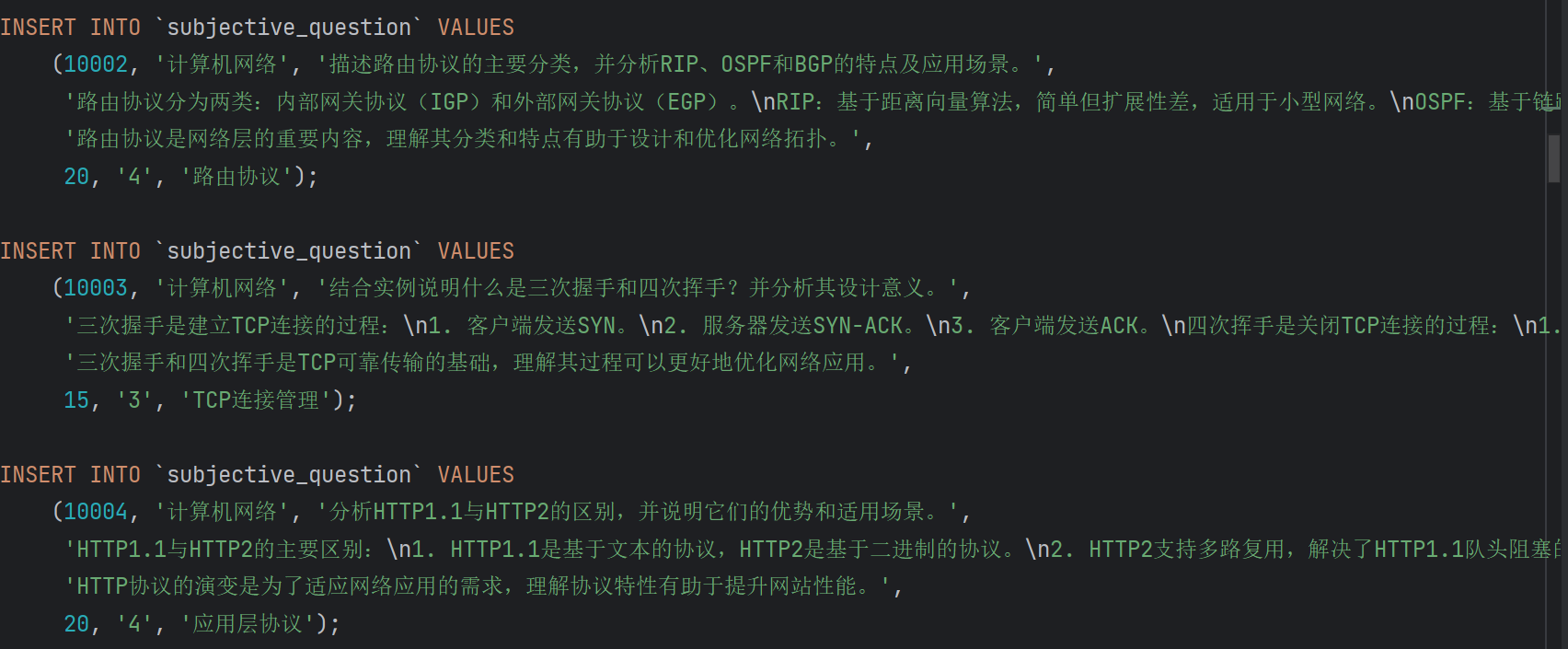
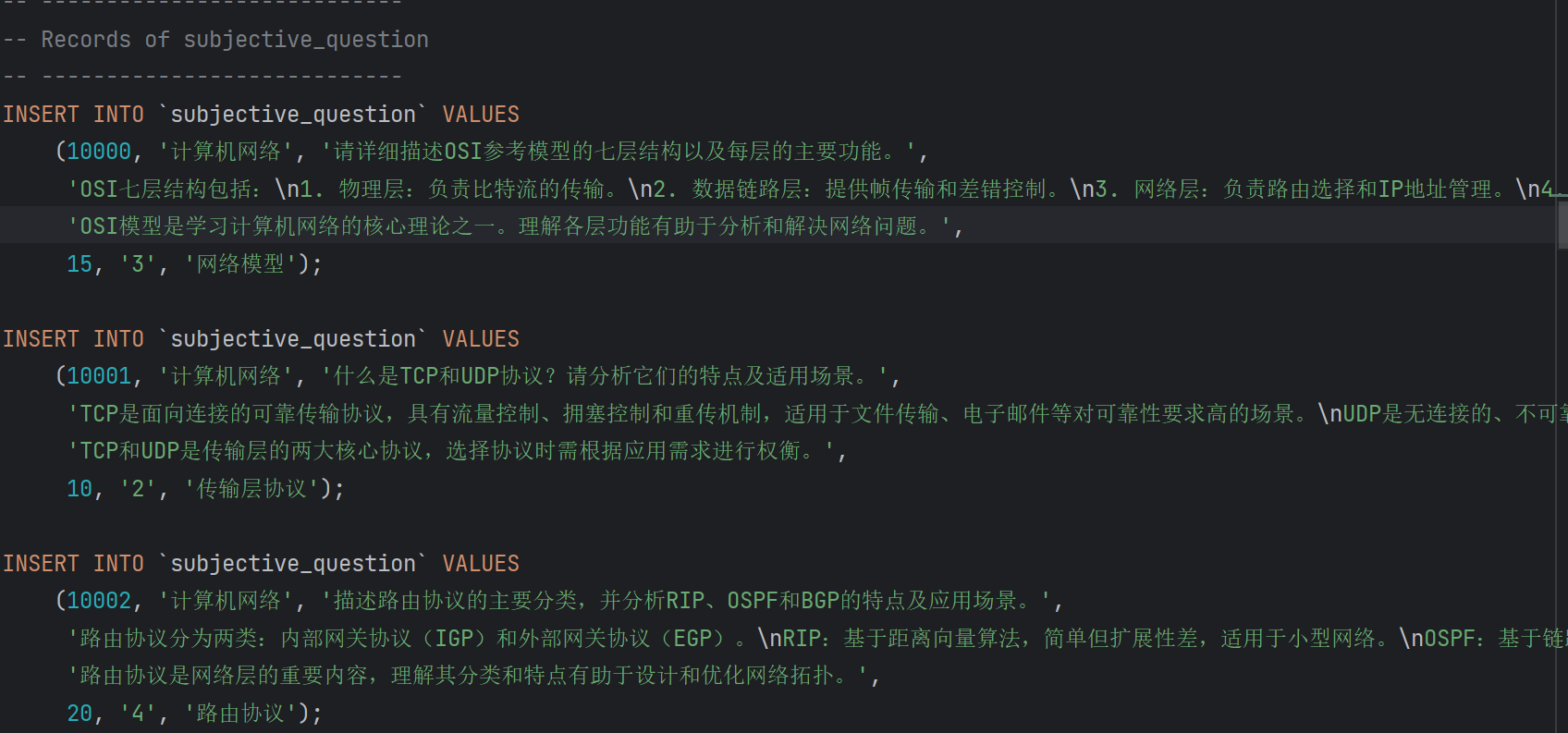
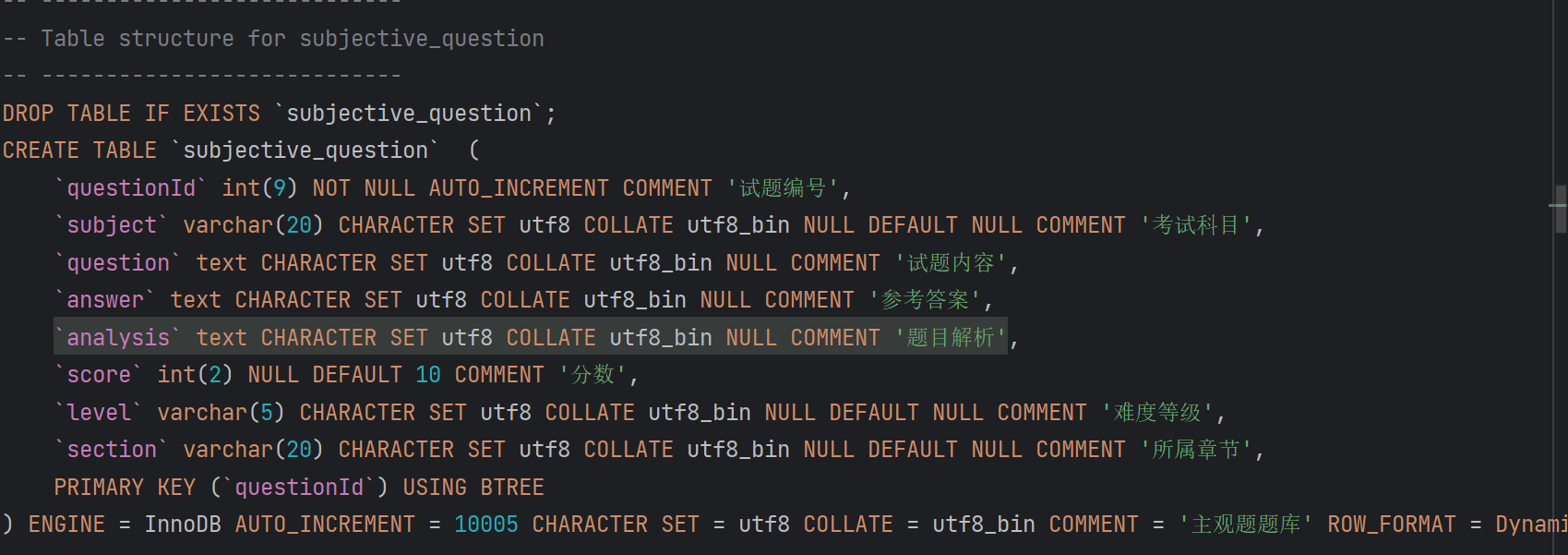
在教师端新设立了一个专门的评阅端口，用于主观题的批改和评分。该端口能够显示学生提交的主观题答案，并提供教师评阅界面，支持打分和批注功能，方便教师快速、高效地完成评分工作。评阅结果支持实时保存功能，避免评阅过程中因意外情况导致评分数据丢失。同时，通过一键提交功能，教师可以将所有评分结果同步到学生成绩数据库，保证了数据更新的及时性和准确性。

### 数据同步功能

试卷提交后的处理逻辑中，客观题部分会依据前端的自动评阅规则直接完成评分并写入数据库。对于主观题部分，评分完成后将分数和评语与学生的成绩记录进行同步更新。为了实现数据的准确传递，设计了可靠的同步机制，并对可能出现的异常情况（如网络中断或系统错误）加入了容错处理逻辑。这一功能确保了整套试卷评阅工作的完整性，保证学生成绩的准确性和可靠性。

## 数据库设计

### 主观题信息表

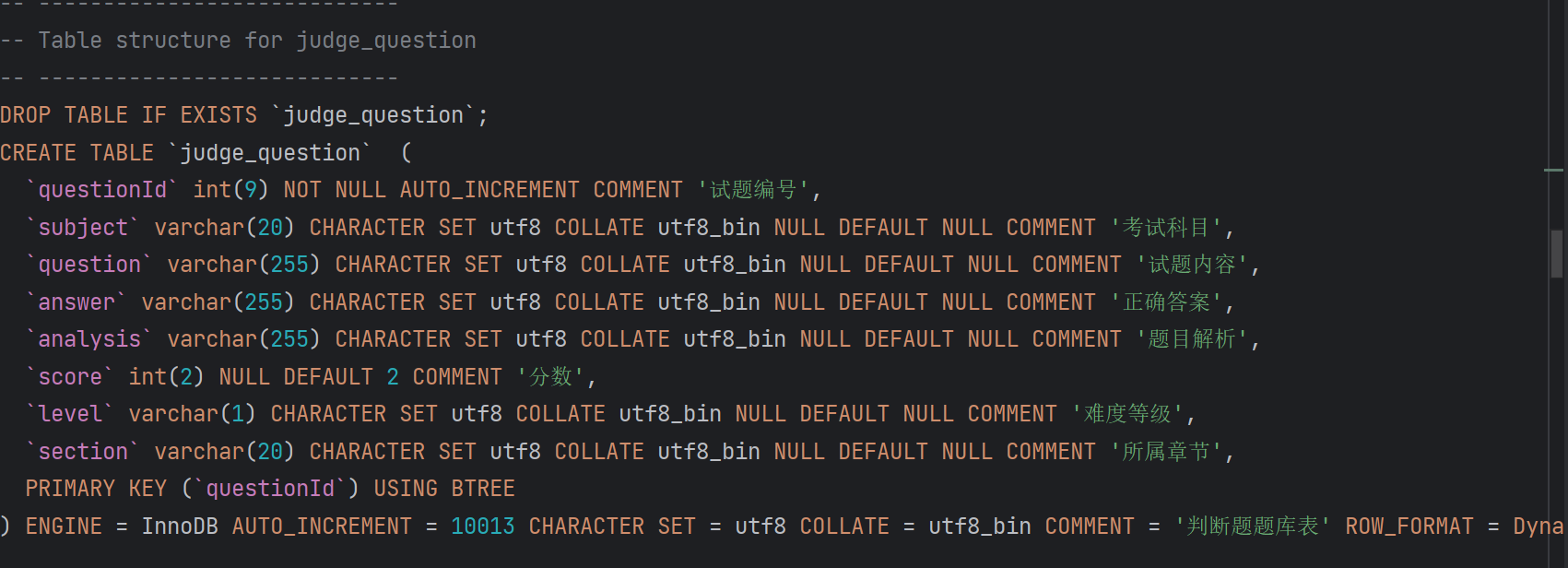
下图为主观题的数据库信息表，其中主要包含题目序号、题目内容、题目分析、题目答案等。

### 选择题信息表

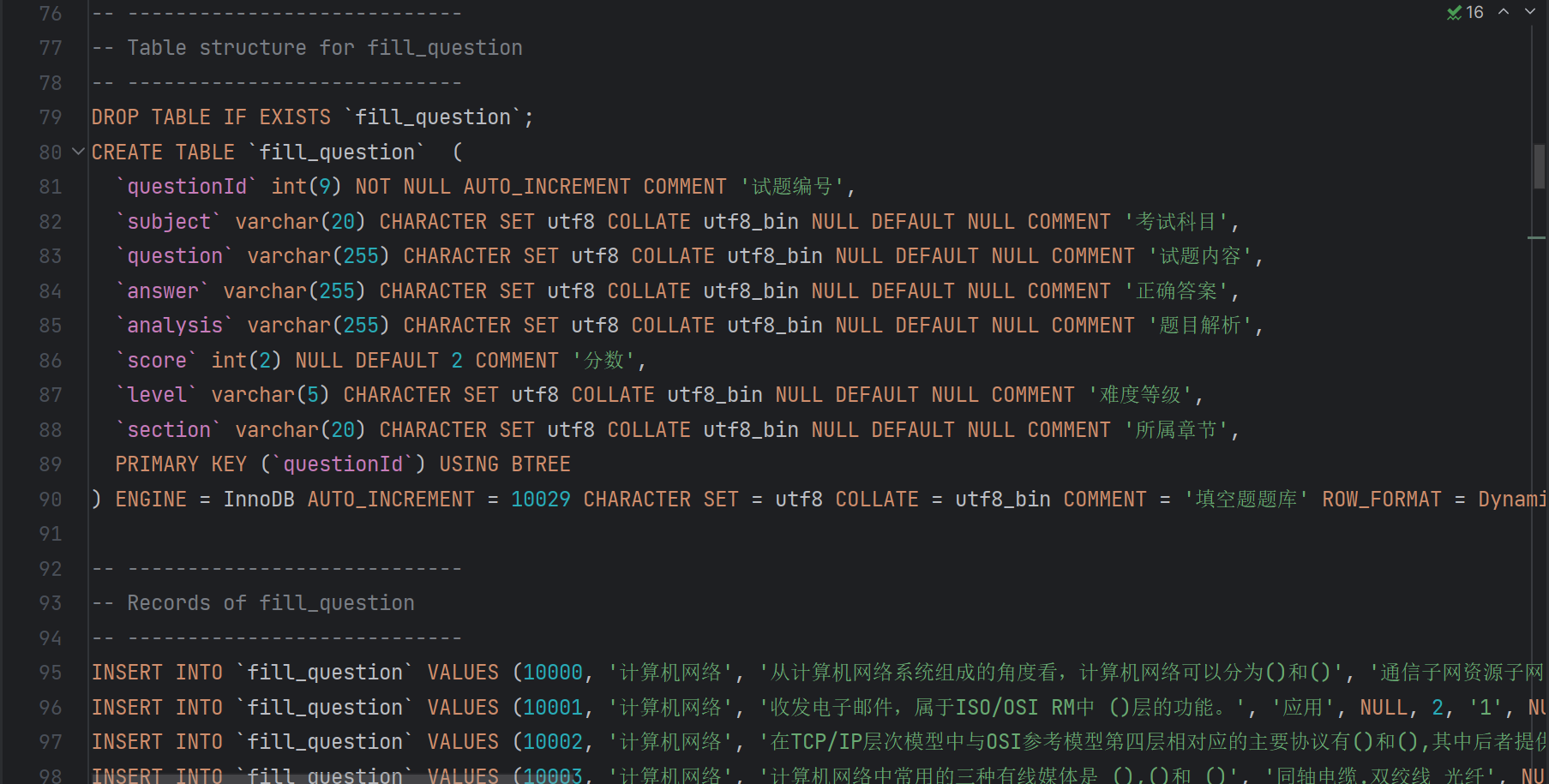
下述内容为选择题的显示内容，同样包含题目序号、试题属性、题目内容以及选项等。

### 图形用户界面, 文本 描述已自动生成判断题信息表

下述内容显示了判断题的基本信息设置，此处的判断题采用‘T’和‘F’来实现，同时也设置了对应的题目信息、作答选项、题目属性与序号等信息。



### 填空题信息表

下述图片显示了填空题的对应信息内容，一部分是对于表的创建的sql语句，而另一部分则是对于该表信息的填充，同样可以看出该部分也包含了题目的序号、属性、作答情况和正确答案等配置。



## 系统界面设计

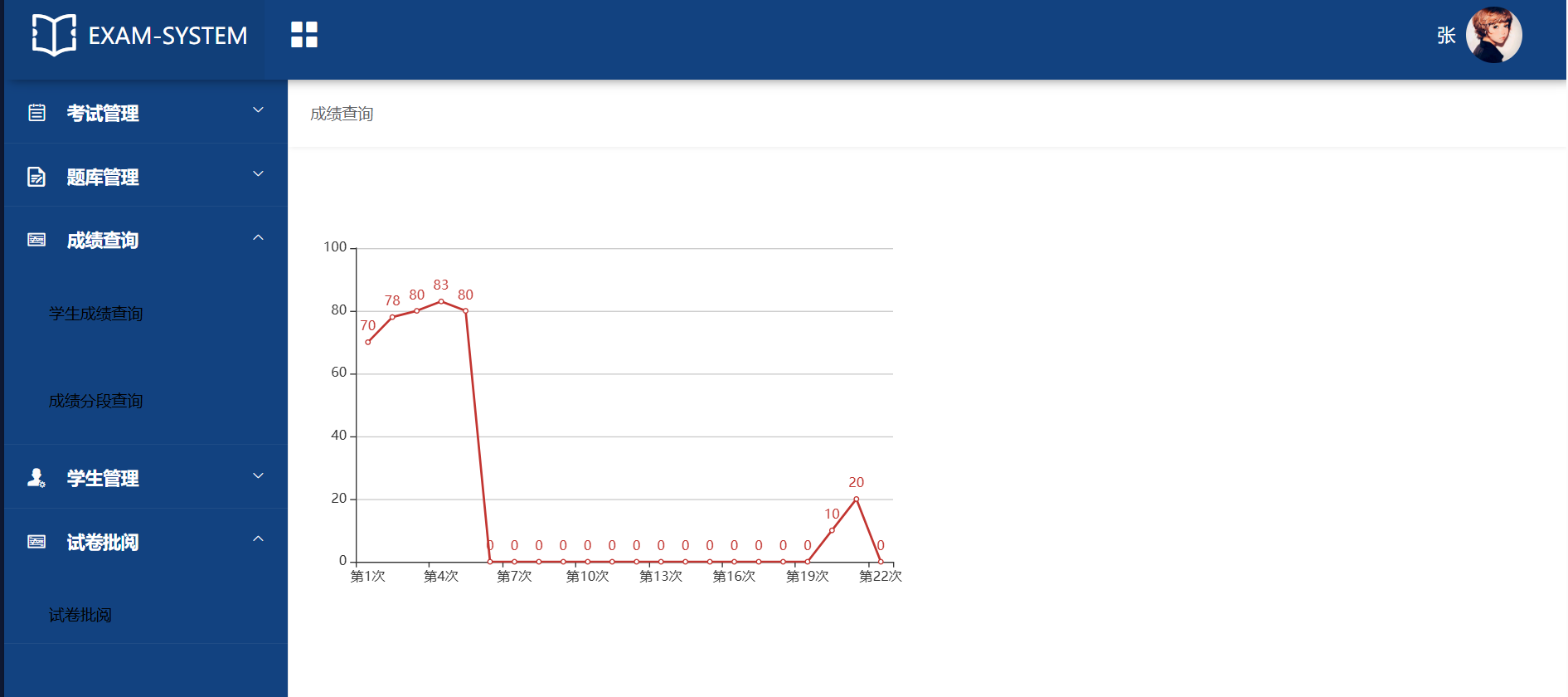
图形用户界面, 应用程序, Teams

描述已自动生成图形用户界面, 应用程序

描述已自动生成图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成

表格

描述已自动生成

# 本人负责模块的编程实现

## 屏幕的截图 描述已自动生成项目结构

上述两幅图片分别为项目前端与后端的目录结构，本项目为前后端分离式项目，因此目录结构比较分明，主要代码文件如前端的vue文件与后端的java类文件均在上述存储。

## 关键代码及说明

### 前端试题显示功能实现代码

// 点击试卷后的缩略信息  
<template>  
 <div id="msg">  
 <div class="title">  
 <span>试卷列表</span>  
 <span>/ {{examData.source}}</span>  
 </div>  
 <div class="wrapper">  
 <ul class="top">  
 <li class="example">{{examData.source}}</li>  
 <li><i class="iconfont icon-pen-"></i></li>  
 <li><i class="iconfont icon-share"></i></li>  
 <li class="right">  
 <div>  
 <span class="count">总分</span>  
 <span class="score">{{score[0]+score[1]+score[2]+score[3]}}</span>  
 </div>  
 </li>  
 </ul>  
 <ul class="bottom">  
 <li>更新于{{examData.examDate}}</li>  
 <li>来自 {{examData.institute}}</li>  
 <li class="btn">{{examData.type}}</li>  
 <li class="right"><el-button @click="toAnswer(examData.examCode)">开始答题</el-button></li>  
 </ul>  
 <ul class="info">  
 <li @click="dialogVisible = true"><a href="javascript:;"><i class="iconfont icon-info"></i>考生须知</a></li>  
 </ul>  
 </div>  
 <div class="content">  
 <el-collapse v-model="activeName" >  
 <el-collapse-item class="header" name="0">  
 <template slot="title" class="stitle" >  
 <div class="title">  
 <span>{{examData.source}}</span><i class="header-icon el-icon-info"></i>  
 <span class="time">{{examData.totalScore}}分 / {{examData.totalTime}}分钟</span>  
 <el-button type="primary" size="small">点击查看试题详情</el-button>  
 </div>  
 </template>  
 <el-collapse class="inner">  
 <el-collapse-item>  
 <template slot="title" name="1">  
 <div class="titlei">选择题 (共{{topicCount[0]}}题 共计{{score[0]}}分)</div>  
 </template>  
 <div class="contenti">  
 <ul class="question" v-for="(list, index) in topic[1]" :key="index">  
 <li>{{index+1}}. {{list.question}} {{list.score}}分</li>  
 </ul>  
 </div>  
 </el-collapse-item>  
 <el-collapse-item>  
 <template slot="title" name="2">  
 <div class="titlei">填空题 (共{{topicCount[1]}}题 共计{{score[1]}}分)</div>  
 </template>  
 <div class="contenti">  
 <ul class="question" v-for="(list, index) in topic[2]" :key="index">  
 <li>{{topicCount[0]+index+1}}.{{list.question}} {{list.score}}分</li>  
 </ul>  
 </div>  
 </el-collapse-item>  
 <el-collapse-item>  
 <template slot="title" name="3">  
 <div class="titlei">判断题 (共{{topicCount[2]}}题 共计{{score[2]}}分)</div>  
 </template>  
 <div class="contenti">  
 <ul class="question" v-for="(list, index) in topic[3]" :key="index">  
 <li>{{topicCount[0]+topicCount[1]+index+1}}. {{list.question}} {{list.score}}分</li>  
 </ul>  
 </div>  
 </el-collapse-item>  
 <el-collapse-item>  
 <template slot="title" name="4">  
 <div class="titlei">主观题 (共{{topicCount[3]}}题 共计{{score[3]}}分)</div>  
 </template>  
 <div class="contenti">  
 <ul class="question" v-for="(list, index) in topic[4]" :key="index">  
 <li>{{topicCount[0]+topicCount[1]+topicCount[2]+index+1}}. {{list.question}} {{list.score}}分</li>  
 </ul>  
 </div>  
 </el-collapse-item>  
 </el-collapse>  
 </el-collapse-item>  
  
 </el-collapse>  
 </div>  
 <!--考生须知对话框-->  
 <el-dialog  
 title="考生须知"  
 :visible.sync="dialogVisible"  
 width="30%">  
 <span>{{examData.tips}}</span>  
 <span slot="footer" class="dialog-footer">  
 <el-button @click="dialogVisible = false">知道了</el-button>  
 </span>  
 </el-dialog>  
 </div>  
</template>  
  
<script>  
export default {  
 data() {  
 return {  
 dialogVisible: false, //对话框属性  
 activeName: '0', //默认打开序号  
 topicCount: [],//每种类型题目的总数  
 score: [], //每种类型分数的总数  
 examData: { //考试信息  
 // source: null,  
 // totalScore: null,  
 },  
 topic: { //试卷信息  
  
 },  
 }  
 },  
 mounted() {  
 this.init()  
 },  
 methods: {  
 //初始化页面数据  
 init() {  
 let examCode = this.$route.query.examCode //获取路由传递过来的试卷编号  
 this.$axios(`/api/exam/${examCode}`).then(res => { //通过examCode请求试卷详细信息  
 res.data.data.examDate = res.data.data.examDate.substr(0,10)  
 this.examData = { ...res.data.data}  
 let paperId = this.examData.paperId  
 this.$axios(`/api/paper/${paperId}`).then(res => { //通过paperId获取试题题目信息  
 this.topic = {...res.data}  
 let keys = *Object*.keys(this.topic) //对象转数组  
 keys.forEach(e => {  
 let data = this.topic[e]  
 this.topicCount.push(data.length)  
 let currentScore = 0  
 for(let i = 0; i< data.length; i++) { //循环每种题型,计算出总分  
 currentScore += data[i].score  
 }  
 this.score.push(currentScore) //把每种题型总分存入score  
 })  
 })  
 })  
 },  
 toAnswer(id) {  
 this.$router.push({path:"/answer",query:{examCode: id}})  
 },  
 }  
}  
</script>

上述代码为试卷详情页，包含试卷的作答，左侧部分为试题序号，右侧部分为作答区域。

// 我的试卷页面  
<template>  
 <div id="myExam">  
 <div class="title">我的试卷</div>  
 <div class="wrapper">  
 <ul class="top">  
 <li class="order">试卷列表</li>  
 <li class="search-li"><div class="icon"><input type="text" placeholder="试卷名称" class="search" v-model="key"><i class="el-icon-search"></i></div></li>  
 <li><el-button type="primary" @click="search()">搜索试卷</el-button></li>  
 </ul>  
 <ul class="paper" v-loading="loading">  
 <li class="item" v-for="(item,index) in pagination.records" :key="index">  
 <h4 @click="toExamMsg(item.examCode)">{{item.source}}</h4>  
 <p class="name">{{item.source}}-{{item.description}}</p>  
 <div class="info">  
 <i class="el-icon-loading"></i><span>{{item.examDate.substr(0,10)}}</span>  
 <i class="iconfont icon-icon-time"></i><span v-if="item.totalTime != null">限时{{item.totalTime}}分钟</span>  
 <i class="iconfont icon-fenshu"></i><span>满分{{item.totalScore}}分</span>  
 </div>  
 </li>  
 </ul>  
 <div class="pagination">  
 <el-pagination  
 @size-change="handleSizeChange"  
 @current-change="handleCurrentChange"  
 :current-page="pagination.current"  
 :page-sizes="[6, 10, 20, 40]"  
 :page-size="pagination.size"  
 layout="total, sizes, prev, pager, next, jumper"  
 :total="pagination.total">  
 </el-pagination>  
 </div>  
 </div>  
 </div>  
</template>  
  
<script>  
export default {  
 // name: 'myExam'  
 data() {  
 return {  
 loading: false,  
 key: null, //搜索关键字  
 allExam: null, //所有考试信息  
 pagination: { //分页后的考试信息  
 current: 1, //当前页  
 total: null, //记录条数  
 size: 6 //每页条数  
 }  
 }  
 },  
 created() {  
 this.getExamInfo()  
 this.loading = true  
 },  
 // watch: {  
   
 // },  
 methods: {  
 //获取当前所有考试信息  
 getExamInfo() {  
 this.$axios(`/api/exams/${this.pagination.current}/${this.pagination.size}`).then(res => {  
 this.pagination = res.data.data  
 this.loading = false  
 *console*.log(this.pagination)  
 }).catch(error => {  
 *console*.log(error)  
 })  
 },  
 //改变当前记录条数  
 handleSizeChange(val) {  
 this.pagination.size = val  
 this.getExamInfo()  
 },  
 //改变当前页码，重新发送请求  
 handleCurrentChange(val) {  
 this.pagination.current = val  
 this.getExamInfo()  
 },  
 //搜索试卷  
 search() {  
 this.$axios('/api/exams').then(res => {  
 if(res.data.code == 200) {  
 let allExam = res.data.data  
 let newPage = allExam.filter(item => {  
 return item.source.includes(this.key)  
 })  
 this.pagination.records = newPage  
 }  
 })  
 },  
 //跳转到试卷详情页  
 toExamMsg(examCode) {  
 this.$router.push({path: '/examMsg', query: {examCode: examCode}})  
 *console*.log(examCode)  
 }  
 }  
}  
</script>

本页面为试卷初始页面，包括各科试卷的简述，学生可以点击标签进行细节跳转，实现具体试卷的打开与关闭。

### 教师端口主观题评阅功能实现代码

1. <template>  
    <div class="all">  
    <el-table :data="pagination.records" border>  
    <el-table-column fixed="left" prop="studentName" label="姓名" width="180"></el-table-column>  
    <el-table-column prop="institute" label="学院" width="200"></el-table-column>  
    <el-table-column prop="major" label="专业" width="200"></el-table-column>  
    <el-table-column prop="grade" label="年级" width="200"></el-table-column>  
    <el-table-column prop="clazz" label="班级" width="100"></el-table-column>  
    <el-table-column prop="sex" label="性别" width="120"></el-table-column>  
    <el-table-column prop="tel" label="联系方式" width="120"></el-table-column>  
    <el-table-column fixed="right" label="操作" width="150">  
    <template slot-scope="scope">  
    <el-button @click="checkGrade(scope.row.studentId)" type="primary" size="small">主观题批阅</el-button>  
    </template>  
    </el-table-column>  
    </el-table>  
     
    <el-pagination  
    @size-change="handleSizeChange"  
    @current-change="handleCurrentChange"  
    :current-page="pagination.current"  
    :page-sizes="[6, 10]"  
    :page-size="pagination.size"  
    layout="total, sizes, prev, pager, next, jumper"  
    :total="pagination.total"  
    class="page">  
    </el-pagination>  
     
    <!-- 编辑对话框-->  
    <el-dialog  
    title="主观题批阅"  
    :visible.sync="dialogVisible"  
    width="30%"  
    :before-close="handleClose">  
    <section class="update">  
    <el-form :model="form">  
    <!-- 遍历学生的所有答案条目 -->  
    <div v-for="(item, index) in form.answers" :key="index">  
    <el-form-item :label="'题目 ' + item.topicId">  
    <el-input v-model="item.answer" type="textarea" rows="4"></el-input>  
    <el-input v-model="item.score" type="number" min="0" max="100" placeholder="请输入分数" style="width: 120px; margin-top: 10px;"></el-input>  
    </el-form-item>  
    </div>  
    </el-form>  
    </section>  
    <span slot="footer" class="dialog-footer">  
    <el-button @click="dialogVisible = false">取 消</el-button>  
    <el-button type="primary" @click="submit()">确 定</el-button>  
    </span>  
    </el-dialog>  
    </div>  
   </template>  
     
   <script>  
   export default {  
    data() {  
    return {  
    pagination: {  
    current: 1,  
    total: null,  
    size: 6,  
    },  
    dialogVisible: false, // 弹框控制  
    form: {  
    studentName: '',  
    answers: []  
    }  
    };  
    },  
    created() {  
    this.getStudentInfo();  
    },  
    methods: {  
    getStudentInfo() {  
    this.$axios(`/api/students/${this.pagination.current}/${this.pagination.size}`).then(res => {  
    if (res.data && res.data.data) {  
    this.pagination.records = res.data.data.records || [];  
    this.pagination.total = res.data.data.total || 0;  
    }  
    }).catch(error => {  
    *console*.error("Error fetching student info", error);  
    });  
    },  
    handleSizeChange(val) {  
    this.pagination.size = val;  
    this.getStudentInfo();  
    },  
    handleCurrentChange(val) {  
    this.pagination.current = val;  
    this.getStudentInfo();  
    },  
    checkGrade(studentId) {  
    this.dialogVisible = true;  
    // 设置加载状态  
    this.loading = true;  
    // 清空之前的数据  
    this.form = {  
    studentId: studentId, // 保存学生学号  
    studentName: '',  
    answers: []  
    };  
    this.$axios(`http://localhost:8080/api/subjective-answers/student/${studentId}`).then(res => {  
    this.loading = false; // 请求完成，关闭加载状态  
    const answers = res.data;  
    if (*Array*.isArray(answers) && answers.length > 0) {  
    this.form = {  
    studentId: studentId, // 设置 studentId  
    studentName: answers[0].studentName, // 显示学生姓名  
    answers: answers.map(answer => ({  
    topicId: answer.topicId,  
    answer: answer.answer,  
    score: '' // 每道题目提供一个输入框来填写分数  
    }))  
    };  
    } else {  
    this.form = {  
    studentId: studentId, // 设置 studentId  
    studentName: '无数据',  
    answers: []  
    };  
    }  
    }).catch(err => {  
    this.loading = false;  
    *console*.error("Error fetching student answers", err);  
    this.form = {  
    studentId: studentId, // 设置 studentId  
    studentName: '无数据',  
    answers: []  
    };  
    });  
    },  
    submit() {  
    this.dialogVisible = false;  
     
    const totalScore = this.form.answers.reduce((sum, item) => sum + (parseInt(item.score, 10) || 0), 0);  
    const dataToSend = {  
    studentId: this.form.studentId,  
    etScore: totalScore // 主观题总分  
    };  
     
    *console*.log("Submitting data:", dataToSend); // 调试日志  
     
    this.$axios({  
    url: '/api/Subjective',  
    method: 'put',  
    data: dataToSend  
    }).then(res => {  
    *console*.log("Response received:", res.data); // 打印后端响应  
    if (res.data.code == 200) {  
    this.$message({  
    message: '更新成功',  
    type: 'success'  
    });  
    }  
    // this.getStudentInfo(); // 更新学生列表  
    }).catch(err => {  
    *console*.error("Error during submission:", err); // 捕获错误  
    this.$message({  
    message: '请求失败，请检查网络或后端服务',  
    type: 'error'  
    });  
    });  
    },  
    handleClose(done) {  
    this.$confirm('确认关闭？')  
    .then(\_ => {  
    done();  
    }).catch(\_ => {});  
    }  
    }  
   };  
   </script>

### 文本 描述已自动生成学生端试卷信息显示与答案处理功能实现代码

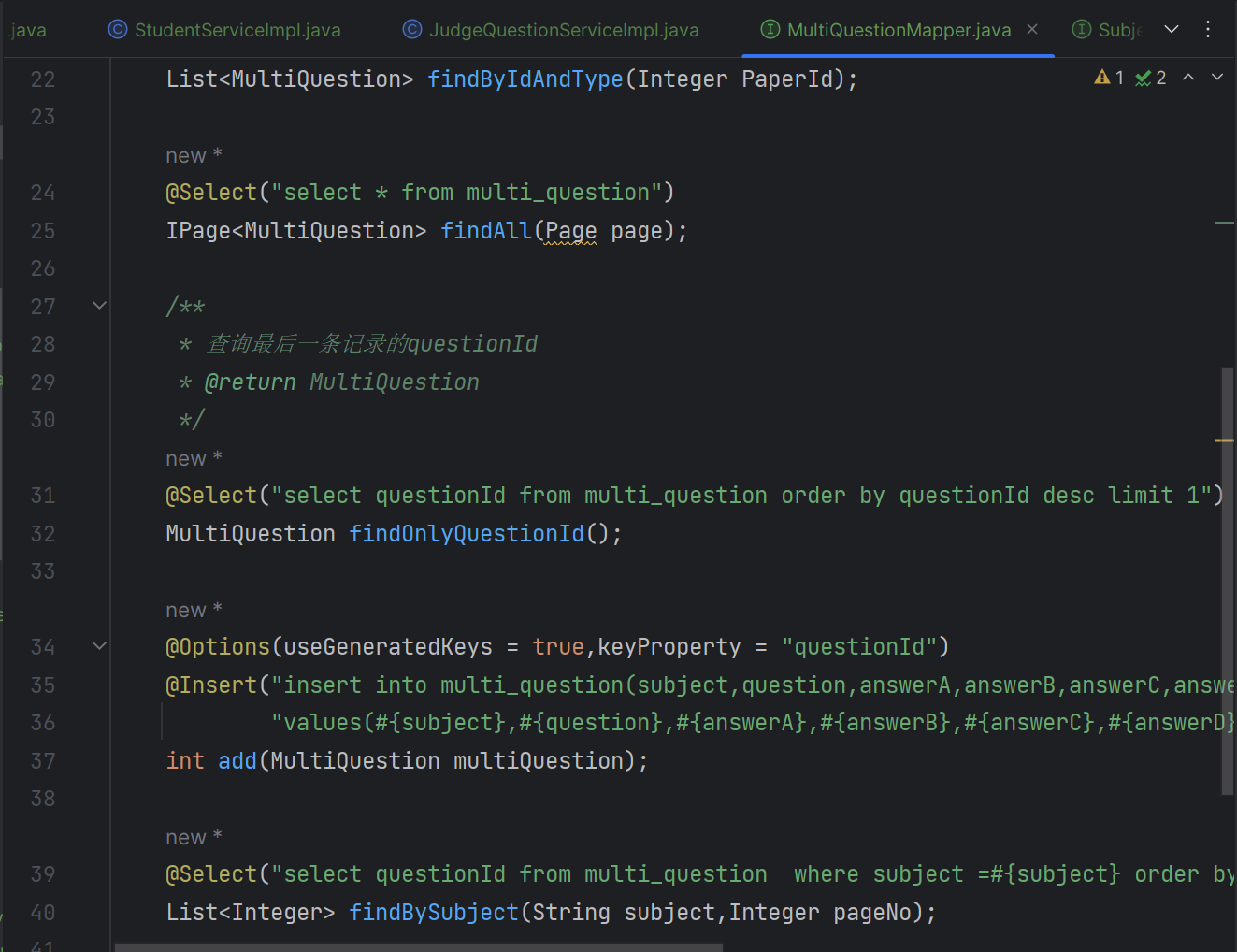
上述图片实现了学生在后端控制平台的信息处理，包括对于数据库中试卷信息的获取，对于学生作答情况的收集，以及将学生主观题作答情况放置在数据库之中，实现后续其他端口的调用。文本

描述已自动生成文本

描述已自动生成电脑屏幕截图

描述已自动生成

### 文本 描述已自动生成教师端主观题答案获取以及分数上传功能实现代码

上述图片实现了教师端口对于数据库中学生主观题答案的提取，以及给出分数后将其更新到学生的成绩显示模块中，实现数据流的传递。文本

描述已自动生成文本

描述已自动生成

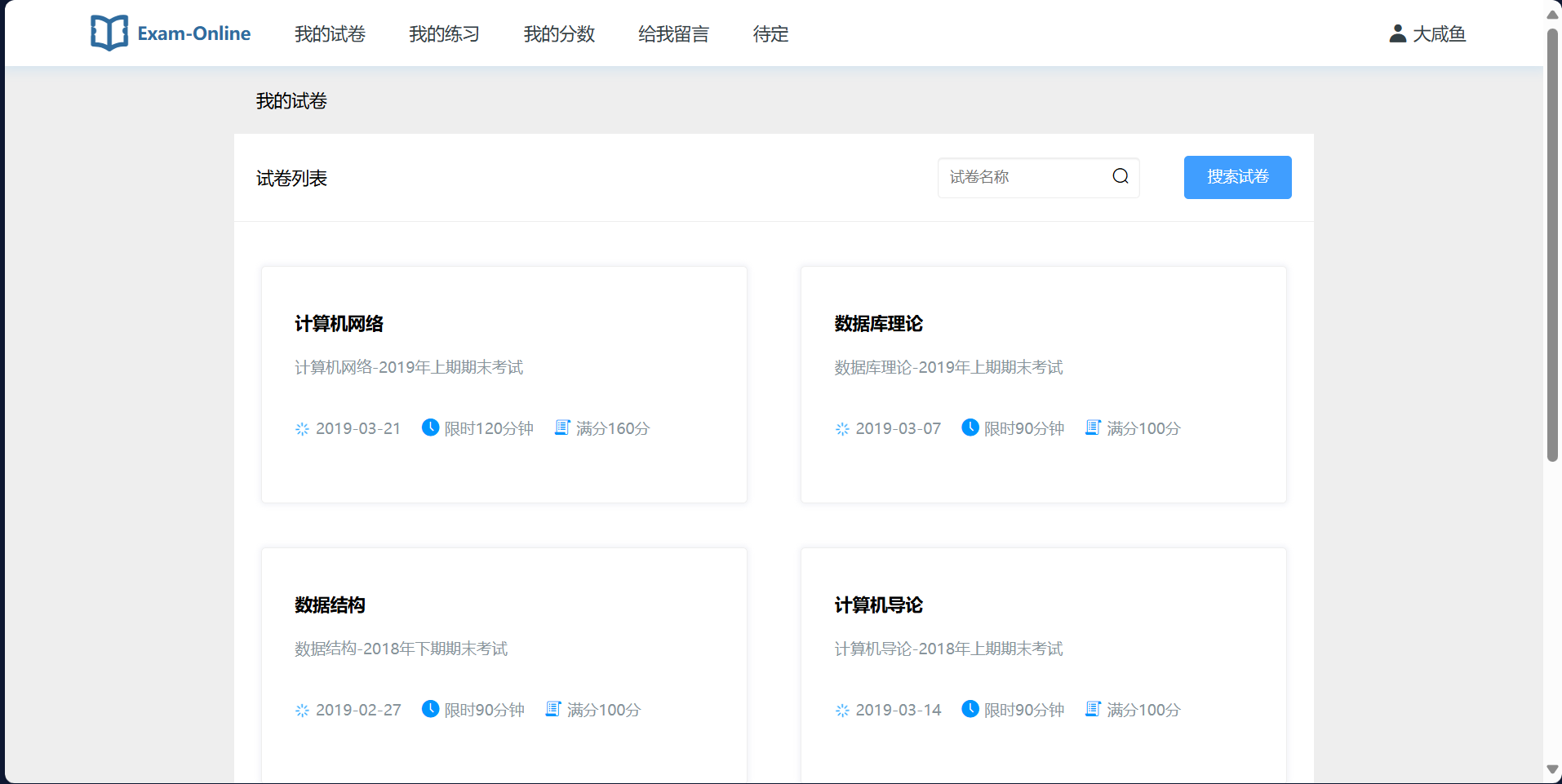
# 本人负责模块的测试与功能展示

## 测试

上述图片为IDEA终端显示的项目网站在进行试卷打开与作答时的后端数据流显示，它表示系统会自动根据当前登录的用户名来进行筛选，根据用户的ID号来查询其目前的试卷内容，并且当用户打开试卷详情页时，系统会根据前端接口所传递的路由来向后端接口发出通知，以此来查询数据库从而获取试卷页面信息。最终将试卷信息呈现在前端页面之上，使学生用户可以在该页面进行作答，以此实现试卷内容的呈递和后端数据的更新，最终实现教师端口的学生分数设置，从而实现分数的赋值。

## 功能展示

学生端详情页



打开具体试卷详情

图形用户界面, 文本, 应用程序

描述已自动生成进入试卷作答页面

图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成图形用户界面, 文本, 应用程序, 聊天或短信, 电子邮件

描述已自动生成进行四类题型的作答

图形用户界面, 文本, 应用程序, 聊天或短信

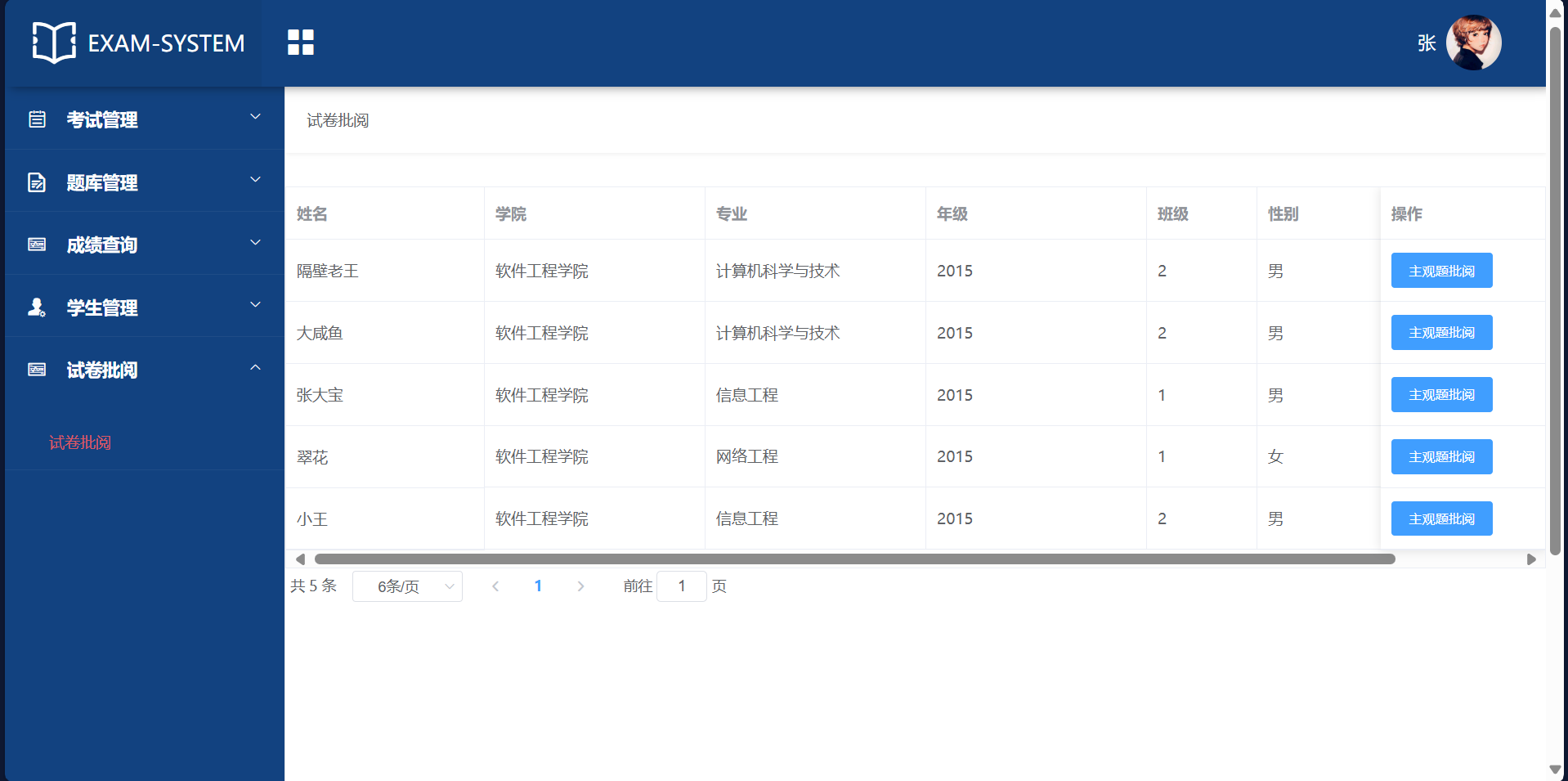
描述已自动生成

图形用户界面, 应用程序

描述已自动生成图形用户界面, 应用程序

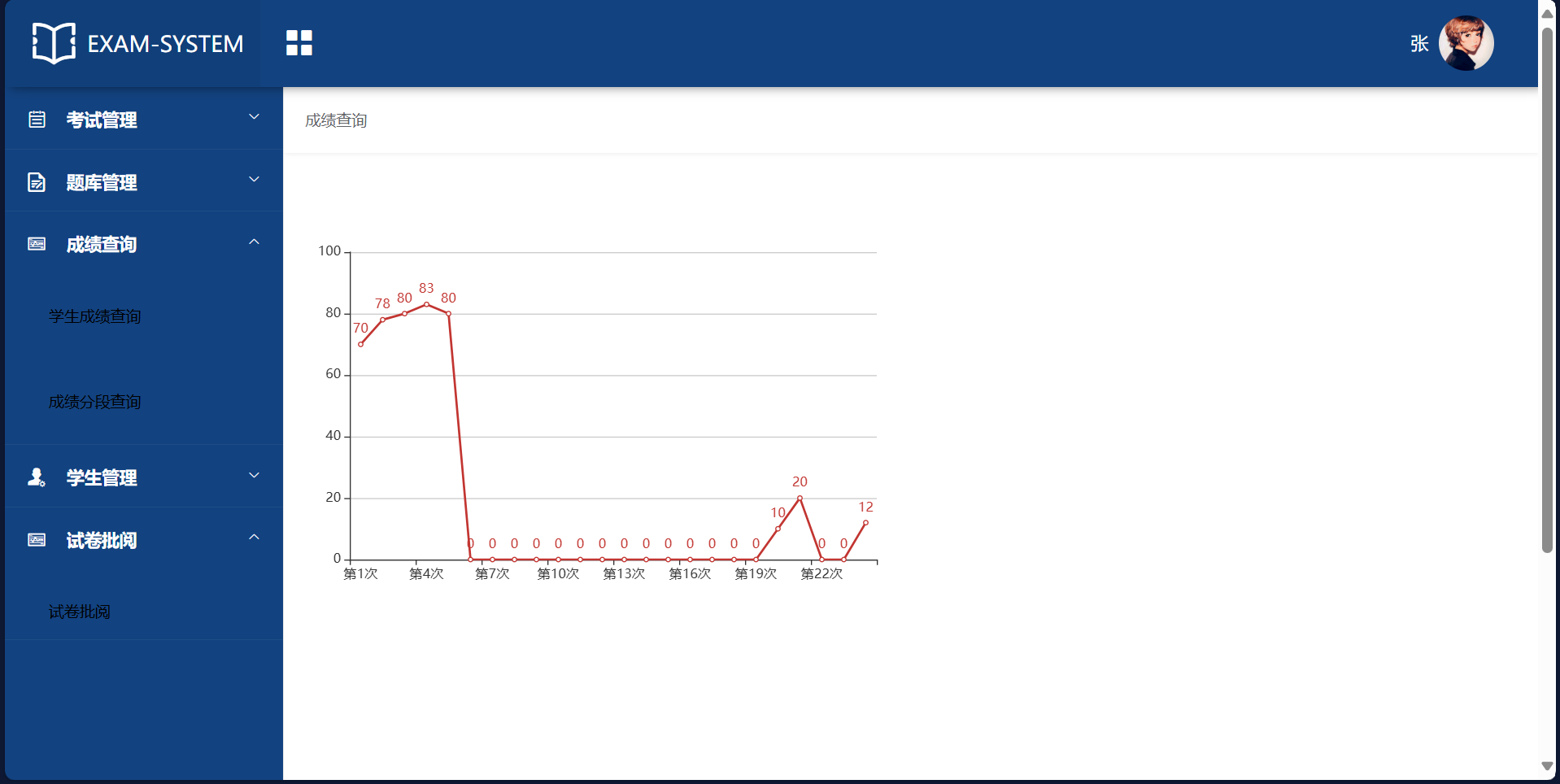
描述已自动生成试卷提交

教师端进行主观题批改



图形用户界面, 应用程序, Teams

描述已自动生成批改过程

分数呈现

图形用户界面, 表格

描述已自动生成

# 实训总结

在本次实训中，我主要负责在线考试系统中学生端试卷设置与处理模块的开发工作。试卷模块分为选择题、填空题、判断题和主观题四部分，功能覆盖了试卷加载、答题倒计时、试卷提交、评分机制及成绩管理等环节。这是一个完整的功能链条，也是考试系统的核心部分，对用户体验和系统效率的要求都很高。

在技术方面，最大的收获是对前后端交互的深入理解。在试卷加载和内容传递的实现过程中，我采用了异步请求技术（如AJAX）来确保页面能够快速响应，实现了试卷内容的动态渲染，同时优化了用户体验。此外，倒计时功能的实现涉及精确的时间管理与前端展示逻辑，通过定时器（如setInterval）与后台时间校验机制相结合，我确保了倒计时的精准性，避免了用户因网络延迟导致的时间误差问题。

评分模块是另一个技术难点。对于客观题部分，我采用了系统自动判分的方式，将标准答案存储于数据库中，通过逻辑匹配算法完成评分，尤其是填空题评分需要处理用户输入的灵活性（如同义词、大小写敏感等问题）。对于主观题部分，我设计了数据库存储结构，将学生提交的答案和教师评分结果进行分表存储，并为教师端提供了一个评阅端口。在评分完成后，系统会通过事务机制将主观题分数与客观题分数进行合并，以确保数据一致性和完整性。

数据库操作是开发过程中的一个重点。在开发中，我设计了清晰的表结构，包括学生信息表、试卷表、答案表和成绩表等，同时通过索引优化了常用查询的效率。在主观题答案提交和评分更新的过程中，我使用了事务处理来保证数据的一致性。例如，当教师提交评分时，系统会同时更新成绩表和答案表，如果其中一个步骤失败，整个事务将回滚，确保数据不会出现错误。

在开发过程中，我深入参与了功能的设计与实现，对整个系统的运行逻辑有了更清晰的认识。从试卷内容动态加载到倒计时的精准触发，再到客观题自动评分与主观题人工评阅，每一个环节都需要精细化的设计与开发。尤其是在主观题评分部分，为教师端设计了评阅端口，并实现了教师评分数据的实时更新与同步，将分数准确地合并到学生成绩中。这让我对系统交互和数据库操作有了更深的理解，也进一步锻炼了自己的全栈开发能力。

实训的过程并非一帆风顺。初期在需求分析时，如何平衡功能完整性与用户体验是一个重要的挑战。我不断调整设计方案，结合实际场景优化功能逻辑，最终达到了既能满足用户需求，又能保证系统性能的效果。此外，开发中的一些细节问题，比如倒计时的精准性、分数更新的事务处理，也让我认识到技术细节的重要性。在解决这些问题的过程中，我的代码调试能力和对系统架构的把控能力得到了很大提升。

本次实训让我不仅掌握了多种技术开发技能，还对项目管理、时间规划有了更多体会。通过这一项目，我深刻感受到团队协作的意义，大家在讨论中不断完善方案，在开发中相互支持，为项目的最终成功打下了坚实的基础。

同时，这次实训也让我对自己的专业能力有了更多信心。我从中学习到了如何更好地分析需求、设计功能和优化系统；也积累了丰富的实践经验，比如模块化开发的思路、前后端联调的技巧以及数据库事务处理的重要性。这些经验不仅提升了我的技术水平，也让我对未来的学习和工作充满了期待。

总的来说，本次实训是一次非常宝贵的经历，我不仅收获了知识，也积累了实际开发的经验，更深刻理解了作为开发者在实际项目中应具备的责任感和专业性。这段经历对我的成长意义非凡，我会将其作为继续前行的重要动力。

指导教师评语：

指导教师签字：

年 月 日