学校代码： 11059

学 号： 20302092033



**毕业设计（论文）**

**BACHELOR DISSERTATION**



论文题目:基于决策树和回归分析的在线评测系统OJ的开发和优化

学位类别: 学士学位

学科专业: 软件工程专业

作者姓名: 王星力

导师姓名: 胡春玲

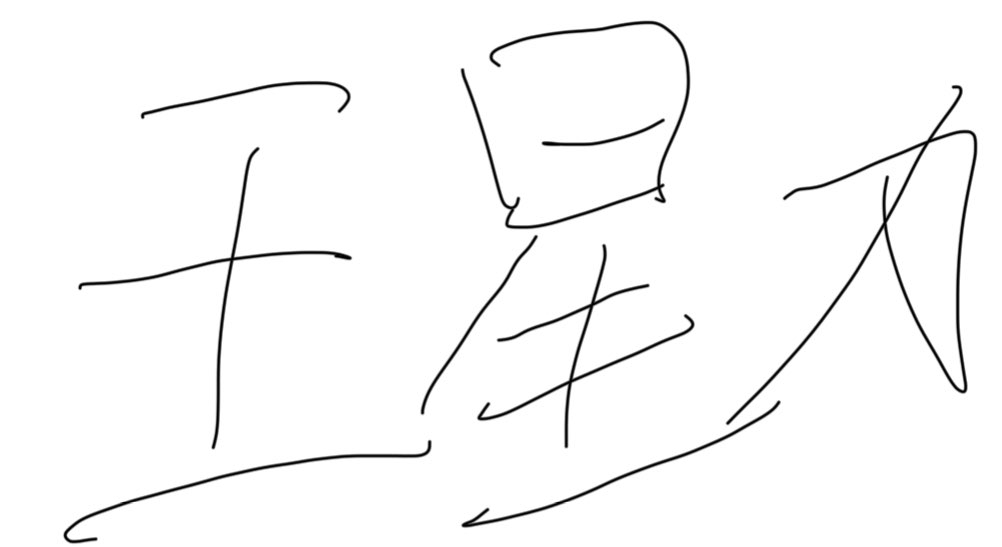
完成时间: 2024.05

合肥大学学位论文原创性声明

郑重声明：本人所呈交的学位论文

基于决策树和回归分析的在线评测系统OJ的开发和优化

是在导师的指导下，独立进行研究取得的成果。除文中已经注明引用的内容外，本论文不包含个人或集体已经发表或撰写过的作品成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律后果，并承诺因本声明而产生的法律结果由本人承担。

学位论文作者：

日期： 2024 年 5月 30日

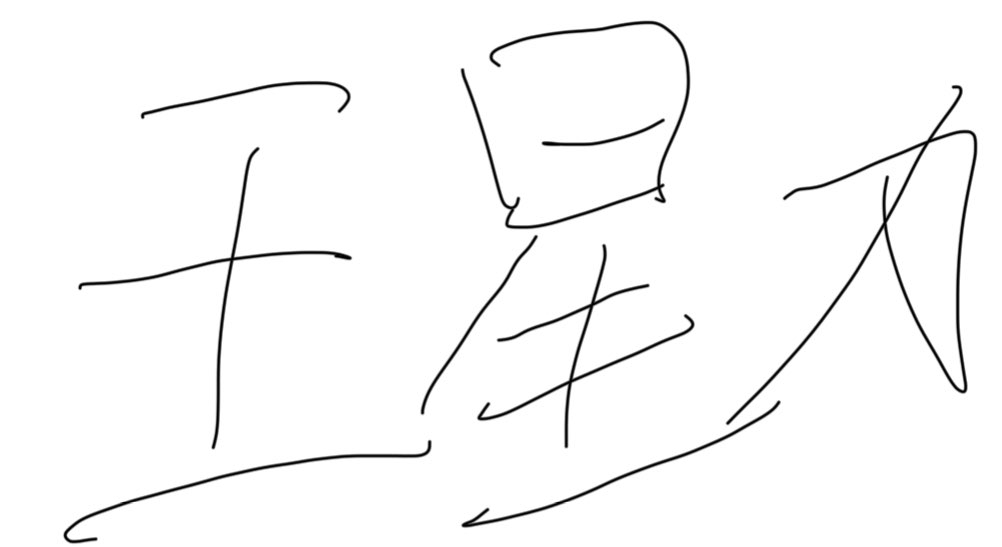
合肥大学学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权**合肥大学**将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于

保 密🞎，在\_\_\_\_年解密后适用本授权书。

不保密🗹。

学位论文作者：  指导教师：胡春玲

日期：2024 年 5 月30 日 日期： 2024 年 5 月 30 日

**基于决策树和回归分析的在线评测系统OJ的开发和优化**

**中 文 摘 要**

程序设计的在线评测系统（OJ）能够提供即时的反馈和评测结果，帮助计算机相关专业学生提高编程能力。在线评测系统可以模拟真实竞赛的环境，让算法和程序设计类竞赛选手在类似真实竞赛的环境中练习，提高竞赛水平。随着互联网技术的成熟，建立在线评测平台变得容易，同时云计算和分布式计算技术使得大规模的在线评测成为可能。

本系统的总体架构为前端使用Vue，后端使用Django，并用数据库模型存储用户信息、题目数据、提交记录等，同时设计前后端交互的 API 接口。在前端开发中，使用 Vue CLI 创建 Vue 项目，开发前端页面，如：登录/注册、题目列表、题目详情、提交代码、查看评测结果等。在后端开发中，使用 Django 的命令创建项目和应用，根据设计阶段的数据库模型，使用 Django 的 ORM 创建数据库表。编写视图函数处理前端请求，如用户认证、题目管理、提交评测等。使用 Django REST framework 快速搭建 RESTful API，配置 Django 的用户系统、权限系统和 admin 管理界面。

本系统首先使用决策树对评测机的判题逻辑进行了优化，部分明显会超时的代码不用跑满时限就能被判超时。其次，系统还使用了回归分析算法，来计算出用户使用评测机的并发量，并选择合适的服务器。最后系统进行全方位的测试，包括功能测试，验证系统的兼容性等等，确保系统能正常运行。

**关键词：在线评测系统；ACM竞赛；决策树；回归分析；系统优化****Development and Optimization of an Online Judging System (OJ) Based on Decision Trees and Regression Analysis**

**ABSTRACT**

An online system refers to a type of system that can provide online services to its users, or specifically, an online evaluation system. Through this system, program codes are compiled and executed. The specific process involves the user logging into a browser, submitting their program code, and the system backend compiles and executes the program. Afterwards, the system uses pre-set inputs and outputs for comparison to check the correctness of the program. The technical theoretical basis for this is black-box testing in software engineering. Users submit their programs online through a browser, and considering system security factors, certain restrictions need to be placed on the program code's compilation execution time and memory usage. First, the system backend compiles the unknown program upon receipt of the user's submission. Compilation commands can use CCC, which is called in a manner similar to calling Shell commands. Programs that pass compilation then undergo correctness testing. The method of testing program correctness is black-box testing, which uses test data or test cases designed according to the program for input and output comparison to determine program correctness.This paper deeply studies the design and implementation of the Online Judge system based on the tag intelligent recommendation algorithm. The system is divided into two parts. The Web backend implements tag-based topic intelligent recommendation, user management module, announcement management module, question bank management module and The competition management module and the question judgment module realize the load balancing design of multiple evaluation machines. Students can easily log in to the system to conduct programming training and participate in competitions online. Analyzed the requirements of OJ system and researched the latest development technology, discussed the system architecture design and specific function development as well as some problems and challenges encountered, and realized a feature-rich OJ system, including tag-based intelligent Recommendation algorithm, Web application development based on Python and Django, multi-evaluation machine load balancing and Docker-based deployment operation and maintenance, etc.

This research initially analyzed the workflow and existing problems of traditional online evaluation systems, and then introduced decision tree algorithms to optimize the process of determining problem difficulty, enabling the system to predict the likelihood of a user solving a specific problem based on their submission history. Furthermore, through regression analysis techniques, this paper also optimized the system's scoring model, allowing it to more accurately reflect the performance of the user's program. In terms of system implementation, this study adopted a modular design concept, separating functions such as problem recommendation, code evaluation, and score feedback, which not only improved the system's processing efficiency but also enhanced its maintainability and scalability. In experimental evaluations, compared to traditional online evaluation systems, the online evaluation system based on decision trees and regression analysis showed significant improvements in evaluation efficiency and accuracy. By introducing decision tree and regression analysis techniques, this paper successfully developed and optimized the online evaluation system, not only enhancing the efficiency and accuracy of evaluations but also improving user experience. This study has significant theoretical and practical significance for advancing the development of online evaluation systems.

**KEY WORDS: Online Judging System;Decision Trees;ACM Competition;Regression Analysis; System Optimization**

**目 录**

[中 文 摘 要 I](#_Toc167814372)

[ABSTRACT II](#_Toc167814373)

[目 录 III](#_Toc167814374)

[第一章 绪论 1](#_Toc5217)

[1.1 课题的来源及意义 1](#_Toc30424)

[1.2 系统设计的目标及基本思路 1](#_Toc14316)

[1.2.1 设计目标 1](#_Toc16372)

[1.2.2 基本思路 2](#_Toc30723)

[第二章 关键技术介绍 3](#_Toc2455)

[2.1框架介绍 3](#_Toc26884)

[2.1.1 Django框架 3](#_Toc2541)

[2.1.2 Vue框架 3](#_Toc13975)

[2.2 算法介绍 4](#_Toc23449)

[2.2.1决策树算法 4](#_Toc8188)

[2.2.1 回归分析算法 4](#_Toc23369)

[2.3 功能模块 4](#_Toc26611)

[第三章 OJ前端实现与优化 6](#_Toc15989)

[3.1 OJ主页面设计 6](#_Toc23776)

[3.1.1首页设计 6](#_Toc5496)

[3.1.2问题界面设计 6](#_Toc793)

[3.1.3比赛界面设计 7](#_Toc15150)

[3.1.4 Rank界面设计 7](#_Toc31350)

[3.1.5其他界面设计 8](#_Toc1717)

[3.2 用户管理页面设计 8](#_Toc16803)

[3.2.1 用户登录界面设计 8](#_Toc30727)

[3.2.2 用户管理界面 10](#_Toc19651)

[第四章 OJ 后端实现与优化 12](#_Toc7809)

[4.1 数据库的实现 13](#_Toc21092)

[4.2 部分模块功能实现 14](#_Toc12441)

[4.2.1 登录模块的操作： 14](#_Toc2257)

[4.2.2 题目权限设置： 15](#_Toc18000)

[4.3 判题功能实现： 16](#_Toc8012)

[4.3.1 判题机 16](#_Toc29730)

[4.3.2 评测技术 17](#_Toc22960)

[4.4. 基于决策树的评测技术的优化 18](#_Toc12930)

[4.4.1决策树算法 18](#_Toc10892)

[4.4.2 决策树算法的实现 19](#_Toc7712)

[4.4.3 后端实现 21](#_Toc18107)

[4.5 测评服务器与测评机 24](#_Toc16276)

[4.6基于回归分析的服务器选择优化 25](#_Toc18171)

[4.6.1 回归分析算法 25](#_Toc15647)

[4.6.2回归分析算法的实现 26](#_Toc20252)

[4.6.3 后端实现 27](#_Toc12184)

[4.3.7 Special judge 28](#_Toc12216)

[第五章 系统测试 31](#_Toc23503)

[5.1 测试目的 31](#_Toc17169)

[5.2 测试用例 31](#_Toc24665)

[参考文献 36](#_Toc32627)

[致 谢 37](#_Toc12037)

**第一章 绪论**

**1.1 课题的来源及意义**

在线评测系统（Online Judging, OJ）作为评估编程技能和算法知识的重要工具，在教育和编程竞赛中扮演着核心角色。随着计算机科学教育的普及和在线学习平台的发展，对于这类系统的需求日益增加，尤其是在提供准确、高效评测服务方面的需求。然而，随着用户基数的扩大和评测任务的复杂化，传统的OJ系统面临着效率低下和准确性不足的挑战。为了解决这些问题，研究人员开始探索利用先进的数据分析技术来优化OJ系统[1]。

在这种背景下，基于决策树和回归分析的在线评测系统的开发和优化成为了一个重要的研究方向。决策树作为一种有效的数据挖掘技术，能够基于历史数据预测出用户的行为模式，从而帮助系统更准确地判断用户提交的代码质量和可能存在的问题。同时，回归分析可以对系统的评测服务器的选择进行优化，通过分析历史评测数据，找出评测结果与实际代码质量之间的关系，进一步提升评测服务器的运行效率[2]。

随着人工智能技术的日益成熟，我们可以将其融入OJ系统中，不仅能够优化当前的评测流程，更能通过个性化的学习建议和问题推荐，为用户带来前所未有的体验提升。这种融合不仅对于提高系统性能至关重要，更重要的是，它为计算机科学教育的革新与发展注入了新的活力。特别是基于决策树和回归分析等先进技术打造的在线评测系统，其开发不仅标志着系统性能的新飞跃，更预示着教育领域将迎来一场由技术驱动的深刻变革。

**1.2 系统设计的目标及基本思路**

1.2.1 设计目标

本系统旨在打造一个高效、准确、实用的在线评测系统。该系统的主要设计目标包括提高评测的效率，通过利用回归分析来选择服务器，提升资源利用率，并使用决策树模型预测用户提交的代码中可能存在的错误模式，提升评测机器的运行效率。为了支持系统的长期发展，设计目标还包括提升系统的可扩展性和可维护性，确保新功能的添加或用户规模的扩展不会影响现有系统的稳定性。此外，保障用户数据的安全性和隐私性也是设计的重要考虑，系统将实施严格的数据保护措施，以符合相关法律法规的要求。最后，系统将提供实时的反馈和辅导机制，帮助用户理解评测结果，提供错误代码的可能原因，进一步提升用户体验。通过这些设计目标的实现，该系统旨在为用户提供一个全面、高效、安全的在线编程学习和评测环境。

1.2.2 基本思路

在开发和优化基于决策树和回归分析的在线评测系统(OJ)中，选择Django作为后端框架和Vue.js作为前端框架提供了一个强大且灵活的技术基础。Django， 作为一个成熟的Python Web框架，将负责处理复杂的后端逻辑，包括用户认证、数据库操作、与机器学习模型的集成等，利用其ORM和安全特性来高效管理数据并确保系统安全。Vue.js的引入则旨在提供一个响应式且用户友好的界面，通过与Django后端的紧密集成，实现一个流畅的单页应用体验。这种前后端分离的架构不仅提高了开发和维护的效率，也使得系统更加可扩展。决策树和回归分析算法的应用进一步增强了平台的核心功能，即提供准确的编程题目评测和反馈。通过这样的设计思路，该系统旨在为用户提供一个安全、高效且易于使用的在线编程学习和评测环境，同时通过精准的技术反馈促进用户学习效率的提升。

**第二章 关键技术介绍**

**2.1框架介绍**

2.1.1 Django框架

Django是一个常见的Python Web框架，框架内部有大量的内置功能，例如：用户登录的一些API，表单的处理等等，这些功能可以通过可重用的组件实现，使得开发者能够专注于项目的独特部分而不是重新发明轮子。Django的另一个亮点是其优秀的安全性能，它自带了防御SQL注入、跨站脚本、跨站请求伪造等网络攻击的机制。此外，它的ORM（对象关系映射）系统提供了一个强大且直观的数据库操作方式，支持多种数据库后端，并能够进行复杂的数据查询和管理。Django的可扩展性和多功能性使其成为开发复杂Web应用的首选框架之一[3]。

在开发Online Judge（OJ）系统方面，Django可以发挥重要作用。OJ系统的核心功能之一是代码的提交和评估，这涉及到严格的安全性要求和高效的数据处理能力，Django的安全特性可以帮助防御潜在的安全威胁，而其ORM和数据库支持可以高效地处理大量提交记录和测试数据。Django的模型-视图-模板（MVT）架构允许开发者快速构建OJ的用户界面，同时保持代码的清晰和易于维护。Django的强大缓存框架和中间件支持能够帮助OJ系统优化性能和处理高并发的用户请求。最终，Django的模块化特性使得OJ系统可以方便地扩展新功能，如编程语言支持、竞赛管理、排行榜等，满足不断增长的用户需求和功能迭代。

2.1.2 Vue框架

Vue.js是一个渐进式JavaScript框架，适用于构建用户界面和单页应用程序（SPA）。它以其高度的灵活性著称，核心库专注于视图层，便于与其他库或现有项目无缝集成。Vue.js的数据绑定和组件系统简化了复杂应用的构建过程，同时保证了代码的可读性和可维护性。其双向数据绑定功能使得交互式表单输入和界面反馈的创建变得轻而易举，而组件化架构则支持大型应用的开发，通过将应用拆分为可复用的小组件。Vue.js还提供了虚拟DOM渲染和性能优化机制，确保即使处理大量数据也能保持出色的性能。

对于开发Online Judge系统，Vue.js可以作为前端开发的主力框架，提供丰富的用户交互和现代的界面体验。在OJ平台中，用户需要一个清晰、响应式的界面来编写、提交代码，以及查看结果和反馈。Vue可以帮助开发者构建这样的界面，其组件可以用来表示代码编辑器、提交历史记录、问题列表等UI元素，而且可以轻松地与后端API进行交互，以异步方式获取数据和提交用户代码。Vue的快速渲染性能对于需要频繁更新数据的OJ系统尤为关键，如实时排行榜、实时反馈等。总而言之，Vue.js为Online Judge系统提供了一个高效、灵活且易于维护的前端开发解决方案。

**2.2 算法介绍**

2.2.1决策树算法

决策树算法通过模拟人的决策过程来预测或分类问题，是一种常见的数据挖掘方法。它由节点和分支组成，内部节点代表属性测试，分支显示测试结果，而叶节点则代表分类标签或连续值。构建决策树时，需要挑选最佳属性进行分裂，以在每个节点上最大化类别的区分度。这种选择常基于信息增益、增益率或基尼不纯度等统计标准。决策树模型直观易懂，无需复杂数据预处理，且能处理不同类型的数据。但需注意，过深的树可能导致过拟合问题[4]。

在Online Judge（OJ）系统中，决策树算法可以被用来优化优化判题机的判题效率。在我们比赛过程中，参赛选手经常会写一些超时的代码，这种代码会长时间霸占评测机器的资源导致评测队列里面的提交数量累增，大大影响评测效率已经参赛选手的参赛体验。我们用决策树算法可以通过选手的代码的一些外在条件来预测出他的代码是否是超时代码，如果是不用跑满评测时间，就能返回评测结果，可以大大提高评测效率。

2.2.1 回归分析算法

回归分析是统计学中广泛应用的算法，旨在探索自变量（解释变量）与因变量（响应变量）之间的关系。它寻求最佳拟合模型以预测或估计实际数值。简单线性回归关注两个变量间的直线关系，而多元回归则涉及多个自变量。模型性能常通过决定系数（R²）评估，量化模型对数据变异的解释能力。回归分析不仅用于预测，还可用作因果关系推断（需额外统计检验和数据理解）。此算法广泛应用于经济学、生物学、工程学和社会科学等领域。

在Online Judge（OJ）系统中，我们可以使用多元线性回归来预测用户的代码需要运行的时间和所占有的内存，来帮助我们为比赛选择适合的服务器。如果并发量高，我们可以选择性能更加优秀的服务器，提升参赛选手的体验。如果并发量低，可以选择性能一般的服务器额，节省系统的资源利用。

**2.3 功能模块**

本系统主要分为8个模块，具体见图2-1，OJ模块化。

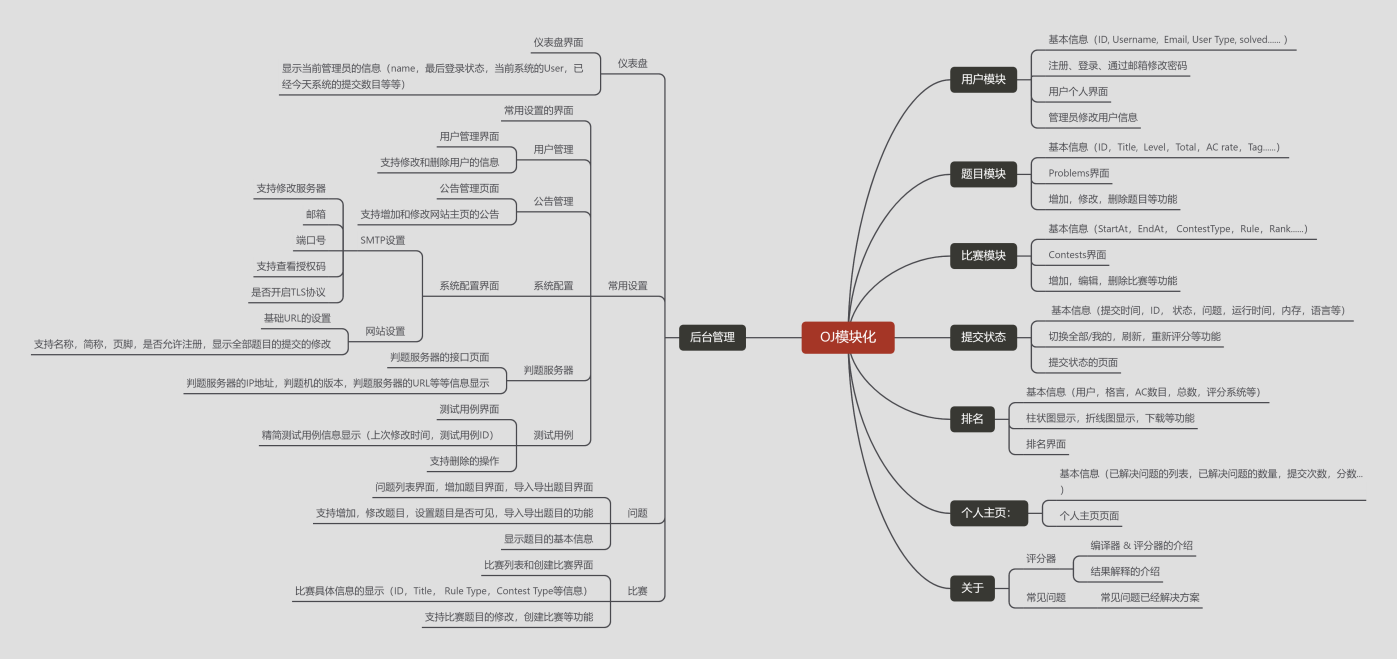


图2-1 OJ模块化

**第三章 OJ前端实现与优化**

OJ系统的前端主要基于Vue框架，利用MVVM（Model-View-ViewModel）设计模式。在MVVM中，Model层主要负责存储数据，View层用于数据的展示，而ViewModel层主要是实现数据在Model和View之间的双向绑定。与其他框架相比，Vue通过其底层的观察者模式有效地处理了数据绑定的中间消息传递，从而让我们能够更专注于View和Model层的开发。在本系统中，页面开发主要基于组件化思想，每个页面都被视为独立的组件，组件之间保持独立，互不干扰。页面间的导航通过Vue Router实现。在组件内部，数据通常在Created生命周期钩子中通过Axios库与后端API进行交互，获取的数据随后绑定到组件的data属性中，Vue的自动数据绑定机制会确保这些数据能够实时、自动地展示给用户。为了提升开发效率和用户体验，前端的页面布局和元素使用了Element UI库进行开发。这种设计使得开发者能够更快速地构建出功能丰富、交互友好的用户界面[5]。

**3.1 OJ主页面设计**

3.1.1首页设计

首页界面提供了系统的概览，包括最新动态、热门题目、即将到来的比赛等信息。我们使用Vue的组件系统创建可复用的信息卡片，用于展示动态信息和题目摘要。Bootstrap的导航栏组件实现了顶部导航菜单，包括问题、比赛、排名等导航项。

 图3-1 OJ主页面

3.1.2问题界面设计

问题界面列出了所有可供解决的编程题目，用户可以按照难度、类别、通过率等进行筛选和排序。每个题目包含题干、输入输出示例、时间内存限制等详细信息，并集成了代码编辑器供用户在线编写代码。使用Vue的表单处理和状态管理，用户可以提交代码并实时查看评测结果。

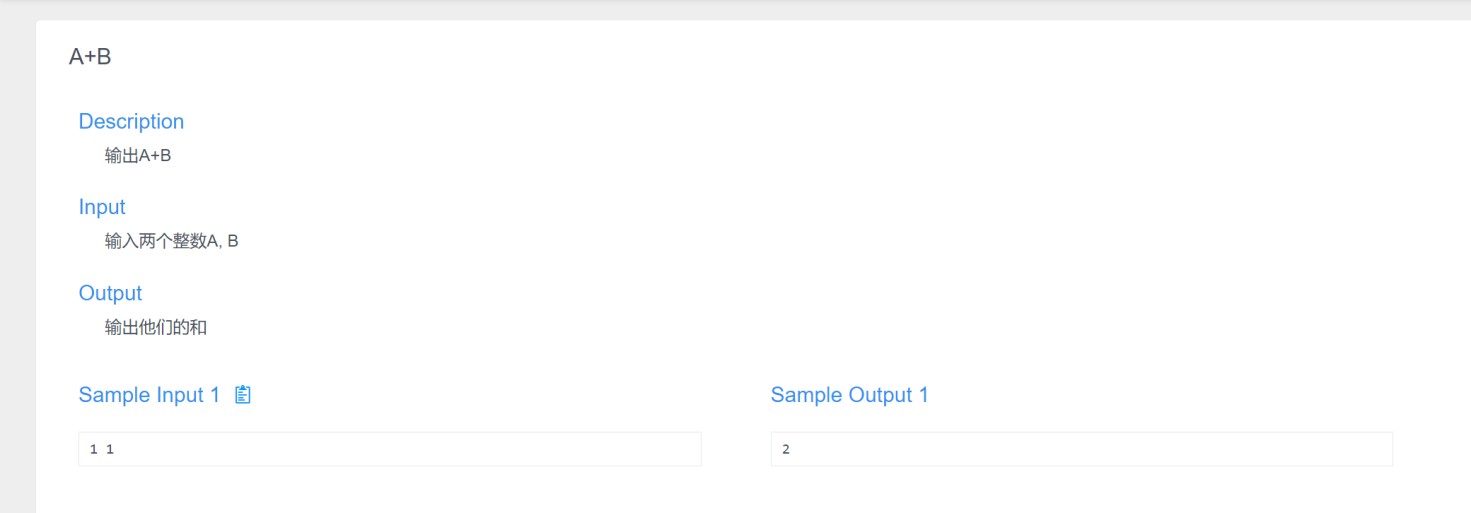


图3-2 问题页面

3.1.3比赛界面设计

比赛界面展示了当前和即将到来的编程比赛，用户可以查看比赛详情、注册参加。比赛进行时，界面会显示倒计时、实时排名和题目状态，使用户能够随时掌握比赛进展。Vue的动态数据绑定和组件更新机制确保了比赛的实时性。

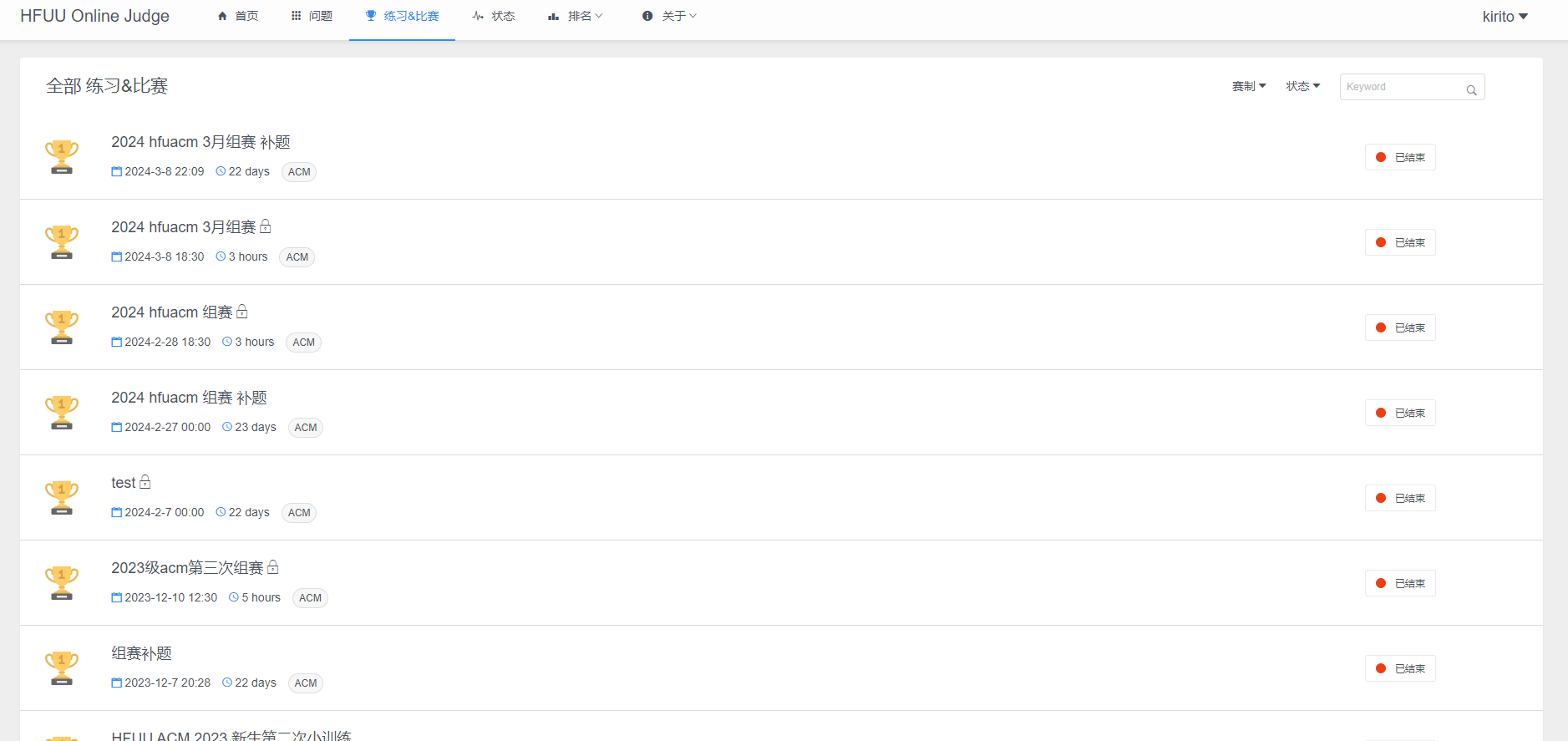


图3-3 比赛页面

3.1.4 Rank界面设计

Rank界面利用Python的matplotlib和seaborn可视化库，实时展示所有用户的数据。为了实现数据的实时更新，系统集成了WebSocket技术，确保前端页面与后端服务器之间能够实时通信，从而迅速反映数据的变化。用户还可以借助筛选、排序、搜索等多种交互功能，更加灵活地查看和分析排名数据。

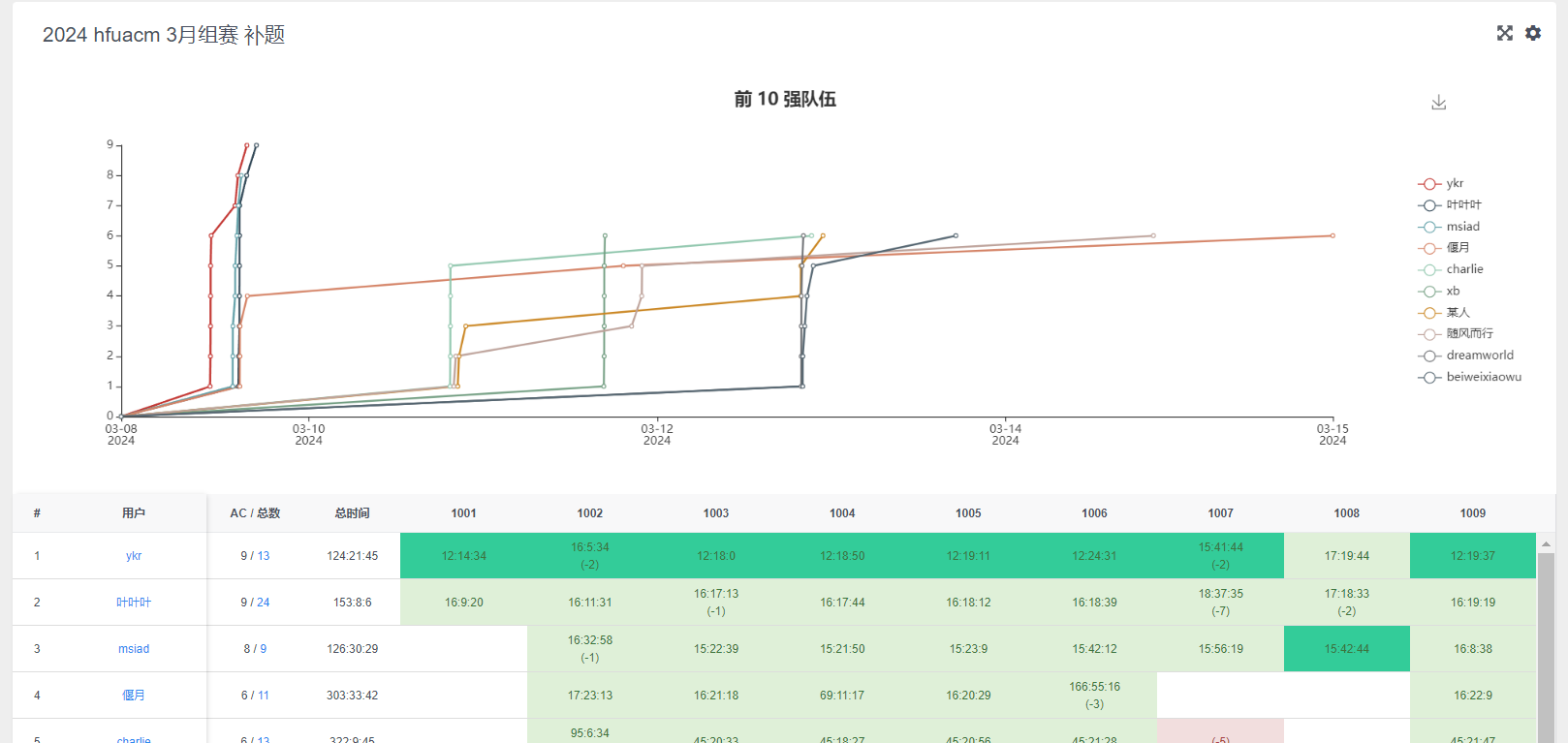


图3-4 排名页面

3.1.5其他界面设计

提交状态界面显示了用户的所有提交记录，包括提交的题目、语言、结果、耗时等信息。排名界面展示了用户的解题排名，使用Vue的计算属性，动态计算用户的排名和积分。

该界面为用户提供了评分器的详细介绍和常见问题的解答，旨在帮助用户更深入地了解系统并顺畅使用。这些信息通过Bootstrap的卡片组件进行展示，确保了界面的清晰性和易读性。个人界面位于页面的显著位置（如右上角），用户只需点击自己的用户名即可轻松进入。在个人界面中，用户可以方便地查看和修改个人信息、密码，以及快速查阅自己的提交记录和排名情况。

**3.2 用户管理页面设计**

3.2.1 用户登录界面设计

在在线评测系统（Online Judge, OJ）中，用户登录界面是用户访问系统并进行交互的第一步。界面设计注重简洁与明了，布局合理且直观，主要组成部分包括便于用户登录的表单以及必要的辅助元素。表单元素包括用户名和密码输入框，以及登录按钮。辅助元素有忘记密码链接和注册新账号链接等等。界面与OJ的整体设计风格保持一致，使用户在登录时感受到整体的统一性。

在功能实现方面，系统对用户输入进行了严格的验证，确保用户名和密码的有效性，从而保障了用户账号的安全性。数据传输过程中，系统采用了HTTPS协议进行加密，保证了用户的账号安全。一旦用户登录成功，系统能够妥善管理用户的会话状态，例如通过cookie或session机制来存储登录信息，确保用户会话的连续性和安全性。若登录失败，系统会提供明确的错误消息，指导用户重新输入或尝试其他操作，从而提升了用户体验和系统的易用性。

用户体验方面，登录界面能快速响应，减少用户的等待时间。界面直观易用，减少用户的操作复杂性。对于已注册用户，登录界面能提供个性化的欢迎信息或提示，登录界面如图3-5所示。

部分代码：

def register(request):

if request.method == "POST":

register\_form = UserCreationForm(request.POST)

if register\_form.is\_valid():

register\_form.save();

user = authenticate(username = register\_form.cleaned\_data['username'], password = register\_form.cleaned\_data['password1'])

login(request, user)

return redirect("/index/")

else:

register\_form = UserCreationForm()

content = {'register\_form':register\_form}

return render(request, "register.html",content)

实现效果：



图3-5 用户登录页面

3.2.2 用户管理界面

用户管理界面是系统管理员的核心工作平台，用于执行各种用户相关的操作。该界面采用模块化设计，将不同管理任务划分为多个部分，以提升管理效率。

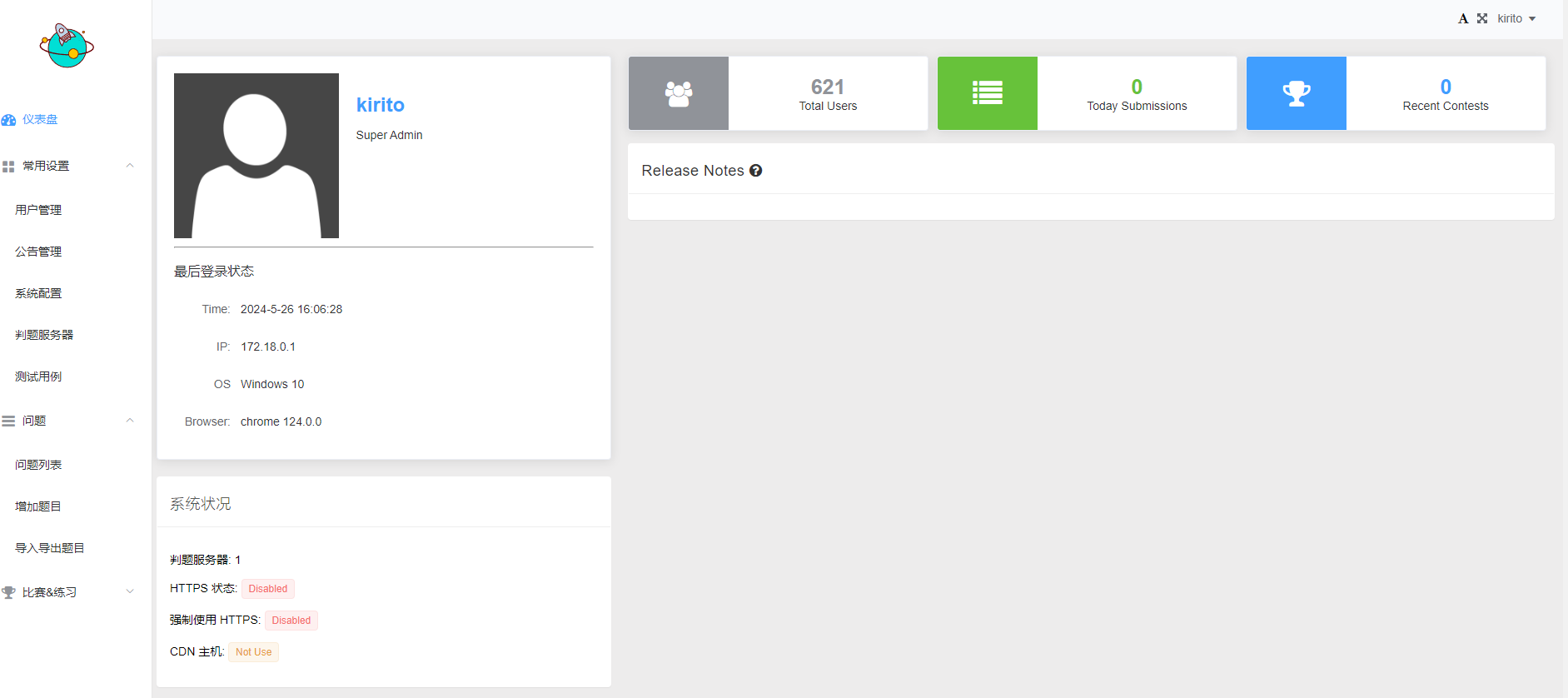
首先，仪表盘作为用户管理界面的中心，聚合了多个关键管理模块的概览信息。其中，用户管理模块允许管理员浏览所有用户的基本详情，如用户名、注册时间、所属权限组等，并赋予管理员禁用账户、调整用户权限等操作权限。

公告管理模块则是管理员发布和管理系统公告的窗口。管理员可以通过此模块创建新公告，对已有公告进行编辑或删除，同时控制公告的发布时间、显示范围等属性。

在常见设置模块中，管理员可以配置系统的基本参数。系统设置子模块允许管理员设定时间格式、题目分类方式、系统支持的语言等。而权限管理子模块则用于定义不同用户组（如普通用户、比赛管理员等）的访问和操作权限，确保系统安全有序运行。

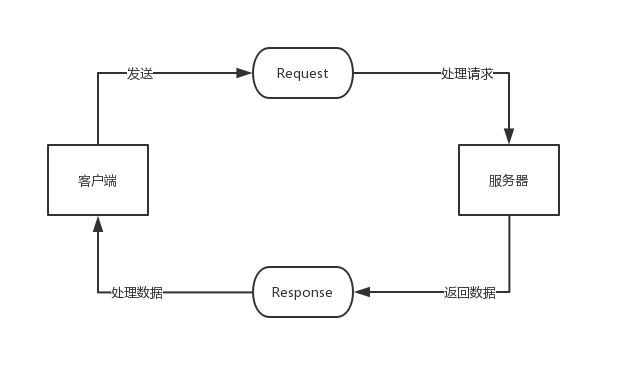
问题管理模块专注于题目和测试数据的维护。题目管理子模块支持管理员添加、修改或删除题目，设定题目的难度级别、答题时间限制、内存占用限制等属性。测试数据管理子模块则提供了测试数据的上传和维护功能，确保评测系统的准确性和公正性。

最后，比赛管理模块为管理员提供了完整的比赛管理流程。在比赛设置子模块中，管理员可以创建新比赛，设定比赛名称、开始和结束时间、题目列表等关键信息。而比赛状态管理子模块则让管理员能够实时监控比赛状态（如进行中、已结束等），并执行相关操作，如发布比赛结果、重置比赛状态等。

图3-6 用户管理页面

**第四章 OJ 后端实现与优化**

后端的开发相较于前端有其特定的优势，主要在于可以专注于数据的处理与逻辑，而无需深入关心具体的显示逻辑。在众多后端框架中，我选择了Python语言的Django框架，这是因为Django开发成本相对较低，且Python语言与我们的判题程序高度兼容，从而构成了一个理想的组合。Django框架采用的是MVT（Model-View-Template）的设计模式，为了实现我们主要关注的数据的交付，我们还会使用REST框架，它基于HTTP协议，具有极高的通用性和灵活性，任何支持HTTP的客户端都能与之交互。REST框架主要由三个核心部分组成：Model、Serializer和View。Model层负责定义数据在数据库中的结构；Serializer层作为中间层，负责数据的序列化和反序列化，以及定义各种数据库操作；而View层则决定哪些数据应该呈现给用户，以及以何种方式呈现。所以，我们编写API时，只需要考虑三个层面的功能。REST框架的引入极大地简化了这一过程，使得数据API的开发变得更为高效和便捷[5]。

 图4-1后端设计主要逻辑

**4.1 数据库的实现**

在开发在线评测系统（OJ）时，为了提高开发效率并简化数据库中表的一些复杂繁琐的操作，我们使用ORM框架来处理与数据库的交互。该框架通过将编程语言中的对象与数据库中的表和字段进行映射，提供了一种更为直观和高级的方式来操作数据库，无需直接编写繁琐的SQL语句。使用ORM框架可以极大地简化OJ系统的数据库操作过程。通过定义模型类，ORM框架能够自动生成对应的数据库表结构，这不仅省去了手动编写SQL语句的麻烦，还确保了数据模型与数据库结构的一致性。此外，ORM框架还提供了丰富的查询语言和查询构建器，使得构建复杂的查询语句变得轻而易举，从而更好地满足OJ系统对于数据查询的多样化需求。因此，ORM框架在OJ系统的开发中起到了至关重要的作用。例如用户的数据库代码：

class UserInfo(models.Model):#继承models

username = models.OneToOneField(User, on\_delete=models.CASCADE)

password = models.CharField(max\_length = 64)

solved = models.TextField()

unsolved = models.TextField()

rating = models.IntegerField();

contests = models.TextField();

usertype = models.CharField(max\_length=30)

submissions = models.TextField()

在本在线评测系统（OJ）中，数据库的角色不仅仅局限于数据存储，它还兼任了队列管理的功能。由于MySQL具备出色的事务管理功能，即使在面对高并发的查询请求时，系统也能游刃有余地处理。在这种架构中，判题机器作为消费者不断地从数据库中拉取待判题的记录，而用户则作为生产者持续地向数据库中插入新的待判题记录。这种生产者与消费者的模式，使得数据库成为了一个高效的中间件，协调着判题流程中的各个环节。通过这种方式，系统能够确保判题过程的顺畅进行，并有效地处理大量的判题请求。如图4-2：

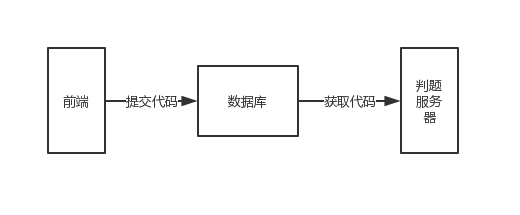


图4-2数据库作为中间件的角色

**4.2 部分模块功能实现**

4.2.1 登录模块的操作：

创建一个自定义的用户模型以满足OJ系统的特定需求，例如添加注册时间、用户类型等字段。接着，创建一个登录表单，包含用户名和密码字段，然后用登录视图来处理用户登录的逻辑，主要是验证用户名和密码是否匹配。同时，会设置cookie，下次重新进入系统的时候，就不用重复登录。如果用户登录的账号密码不正确，就会提示用户，让他重新输入。测试登录逻辑以确保其按预期工作[6]。

class SMTPAPI(APIView):

@super\_admin\_required

def get(self, request):

smtp = SysOptions.smtp\_config

if not smtp:

return self.success(None)

smtp.pop("password")

return self.success(smtp)

@super\_admin\_required

@validate\_serializer(CreateSMTPConfigSerializer)

def post(self, request):

SysOptions.smtp\_config = request.data

return self.success()

@super\_admin\_required

@validate\_serializer(EditSMTPConfigSerializer)

def put(self, request):

smtp = SysOptions.smtp\_config

data = request.data

for item in ["server", "port", "email", "tls"]:

smtp[item] = data[item]

if "password" in data:

smtp["password"] = data["password"]

SysOptions.smtp\_config = smtp

return self.success()

4.2.2 题目权限设置：

本系统在Django框架中，实现OJ（Online Judge）系统的题目权限设置，通过定义题目模型和用户模型之间的关系来实现。首先，创建一个自定义的题目模型，包含题目名称、难度、描述、输入输出格式等信息，并添加一个字段用于存储与题目相关的用户权限信息。在题目模型中，为每个题目添加一个权限字段，可以存储一个或多个用户模型实例，从而为每个题目关联多个用户并设置相应的权限。在Django的视图中，根据用户请求的题目和用户的权限信息，判断用户是否有权访问该题目，并相应地处理权限验证。在模板中，根据用户的权限信息，决定是否显示某个题目，以保证用户能够根据其权限访问和操作题目。通过以上步骤，使用Django创建一个OJ系统的题目权限设置，并确保用户能够根据其权限访问和操作题目，从而更好地管理和保护题目资源，并提供一个公平和安全的编程环境[7]。

def check\_user\_permission(self, user, check\_share=True):

if self.user\_id == user.id or user.is\_super\_admin() or user.can\_mgmt\_all\_problem() or self.problem.created\_by\_id == user.id:

return True

if check\_share:

If self.contest and self.contest.status != ContestStatus.CONTEST\_ENDED:

return False

if self.problem.share\_submission or self.shared:

return True

return False

class Meta:

db\_table = "submission"

ordering = ("-create\_time",)

def \_\_str\_\_(self):

return self.id

**4.3 判题功能实现：**

4.3.1 判题机

在在线评测系统（OJ）中，测评机扮演着核心角色，专门负责进行代码提交的判题工作。当判题服务器接收到来自用户的判题请求后，它会将相应的提交数据转发给测评机。测评机接收到这些数据后，会在一个隔离的环境中（通常称为“沙盒”）对用户的代码进行评测。具体过程如下图所示：

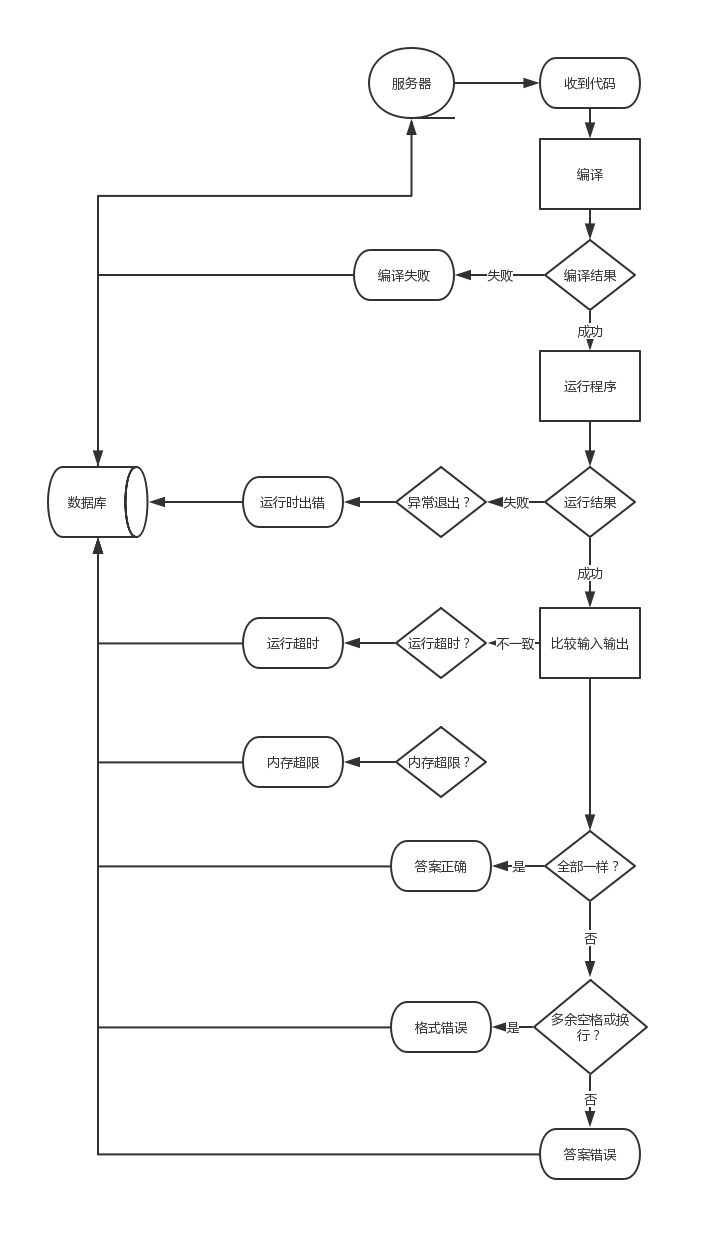


图4-3判题机判题流程

4.3.2 评测技术

在追求系统稳定性的过程中，开发一个完全自主的测评程序可能会面临诸多挑战，包括技术难题、时间成本和潜在的错误。因此，集成已经经过验证的、稳定的测评技术是一个明智的选择。在开源社区中，由青岛大学开发的一款测评模块因其出色的稳定性和实用性而备受关注。这款测评模块在青岛大学内部得到了广泛应用，并且因其出色的性能和稳定性而获得了良好的口碑。该测评技术专注于其核心功能，即运行程序并捕获输出，使得其设计简洁而高效。通过集成这款测评模块到我们的系统中，我们可以充分利用其稳定性和可靠性，确保用户提交的代码能够在一个受控、安全的环境中得到准确的评测。经过测试和改进后，这款测评模块已经与我们的系统兼容，并可以无缝集成到我们的评测流程中。这不仅提高了我们系统的稳定性和可靠性，还大大减少了开发时间和成本。此外，通过集成这款开源的测评模块，我们还可以享受到来自开源社区的持续支持和更新，从而确保我们的系统始终保持最新的技术水平和最佳的性能表现[8]。函数结构如下：

Function (Maximum CPU Time, Maximum Runtime, Maximum Allowed Memory, Maximum Output Size, Maximum Stack Size, Path to Execute Program, Path to Input File, Path to Output File, Path to Error Output)

在调用特定的函数并传入必要的参数后，该函数负责创建一个安全的隔离环境（通常被称为“沙盒”），随后在沙盒内部执行指定的程序。程序执行完毕后，该函数会返回一个结构体，该结构体封装了程序的运行结果。这样的设计确保了程序在受控的环境中运行，同时提供了对运行结果的便捷访问,具体如下：

{'cpu\_time', 'signal':, 'memory', 'exit\_code', 'result', 'error', 'real\_time'}

分别对应着，程序运行使用CPU得时间，程序发出的信号，程序占用的内存，程序退出时的返回值，程序运行的结果，程序运行的时间（包括了系统调度的时间）其中result有如下6个枚举值

SUCCESS = 0 (此结果仅代表程序成功运行完毕并正确退出)

CPU\_TIME\_LIMIT\_EXCEEDED = 1

REAL\_TIME\_LIMIT\_EXCEEDED = 2

MEMORY\_LIMIT\_EXCEEDED = 3

RUNTIME\_ERROR = 4

SYSTEM\_ERROR = 5

当程序正常退出时，我们可以捕获其输出内容，并将其与预设的正确答案进行比对。通过这一比较过程，我们可以判断程序的输出是否正确，从而得出是答案错误还是通过的结果。这种比对机制确保了评测的准确性，并为用户提供了关于其程序性能的直接反馈。。

4.4. 基于决策树的评测技术的优化

4.4.1决策树算法

决策树在OJ开发中可以通过对用户提交代码的智能分析，实现提速的作用，本系统采用的是ID3算法。首先，决策树可以根据代码的多种特征，如代码长度、算法复杂度等，预测代码的评测结果，快速识别出错的提交，从而节省资源并加速整体评测流程。其次，决策树可以根据问题的历史评测数据，如平均运行时间、内存使用情况，预测新提交的代码可能需要的资源量，并据此合理分配评测资源，提高评测效率。随着用户的数量越来越多，算法的不确定性（熵）会减少，减少的越多越好，即信息增益越大越好。

我们假设通过n个条件，D个提交次数来预测新提交的代码的需要资源量,P为选择该分类的概率，K为类别那么：

普通熵的计算方法为：

经验熵的计算方法为：

4.4.2 决策树算法的实现

先从数据库中获取获取提交代码的数据，形成数据集

#将mnist.idx-ubyte文件转换成.csv文件（mnist原始数据集为字符格式），代码如下

def convert(imgf, labelf, outf, n):

f = open(imgf, "rb")

o = open(outf, "w")

l = open(labelf, "rb")

f.read(16)

l.read(8)

images = []

for i in range(n):

image = [ord(l.read(1))]

for j in range(28 \* 28):

image.append(ord(f.read(1)))

images.append(image)

for image in images:

o.write(",".join(str(pix) for pix in image) + "\n")

f.close()

o.close()

l.close()

if \_\_name\_\_ == '\_\_main\_\_':

convert(".\Mnist\\t10k-images.idx3-ubyte", ".\Mnist\\t10k-labels.idx1-ubyte",

".\Mnist\\mnist\_test.csv", 10000)

convert(".\Mnist\\train-images.idx3-ubyte", ".\Mnist\\train-labels.idx1-ubyte",

".\Mnist\mnist\_train.csv", 60000)

# 算法模块的实现

#计算数据集D的经验熵，参考公式经验熵的计算。

def Cal\_HD(trainLabelArr):

HD = 0

# 统计该分支的标签情况

# set()删除重复数据

trainLabelSet = set([label for label in trainLabelArr])

# 遍历每一个出现过的标签

for i in trainLabelSet:

p = trainLabelArr[trainLabelArr == i].size / trainLabelArr.size

# 对经验熵的每一项进行累加求和

HD += -1 \* p \* np.log2(p)

# 返回经验熵

return HD

#经验熵的计算

def Cal\_HD(trainLabelArr):

HD = 0

# 统计该分支的标签情况

# set()删除重复数据

trainLabelSet = set([label for label in trainLabelArr])

# 遍历每一个出现过的标签

for i in trainLabelSet:

p = trainLabelArr[trainLabelArr == i].size / trainLabelArr.size

# 对经验熵的每一项进行累加求和

HD += -1 \* p \* np.log2(p)

# 返回经验熵

return HD

4.4.3 后端实现

# 处理每个判题机的逻辑

def deal\_client(newSocket: socket, addr):

global mutex, queue

statue = False

cursor = db.cursor()

falsetime = 0

while True:

sleep(2) # 每隔两秒取两次

if mutex.acquire(): # 获取队列锁

try:

if statue == True and queue.empty() is not True:

id = queue.get() # 如果可以判题，那就发送判题命令

cursor.execute(

"SELECT language from judgestatus\_judgestatus where id = '%d'" % (id))

data = cursor.fetchall()

# print(data[0][0])

newSocket.send(("judge|%d" % id).encode("utf-8"))

statue = False

else:

newSocket.send("getstatue".encode("utf-8"))

data = newSocket.recv(1024)

recv\_data = data.decode('utf-8')

if recv\_data == "ok":

falsetime = 0

statue = True

else:

falsetime = falsetime + 1

statue = False

if falsetime >= 3600: # 计算一下未准备好的时间，如果超过120s，发送销毁重启命令

newSocket.send("timeout".encode("utf-8"))

# print(addr, "timeout!")

newSocket.close()

mutex.release()

return

# print(addr, statue)

except socket.error:

newSocket.close()

mutex.release()

return

except:

print("error!")

mutex.release()

return

mutex.release()

server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)

server.bind(("", judgerjson["server\_port"]))

server.listen(20)

print("server is running!")

t = threading.Thread(target=getSubmition, args=()) # 用一个线程去跑

t.setDaemon(True)

t.start()

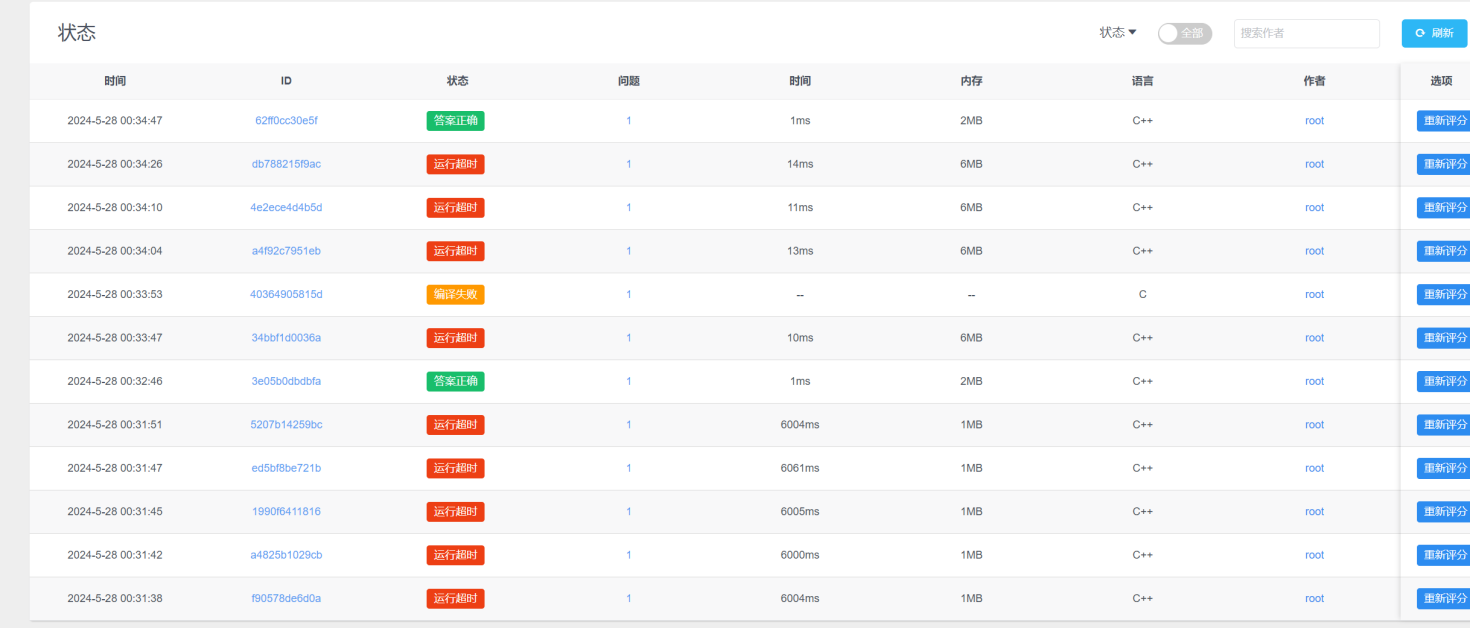


图4-4 用户提交列表

通过时间比对，可以发现，有些会超时的代码，不用跑到极限时限，可以直接被设置为运行超时的状态，可以大大提升判题机和系统的判题效率。

4.5 测评服务器与测评机

尽管测评模块确保了程序在安全稳定的环境中运行，但它本身并不包含判断程序是否通过的逻辑。因此，我们需要额外实现一系列的逻辑处理来评估程序的表现。本系统的测评模块主要分为两个：评测机和评测服务器。评测服务器主要的作用是能为提交记录选择且分配合适的评测机，评测机器的作用是来判断用户的代码是否正确。

在判题的过程中，我们还要考虑到资源的消耗和利用率。如果只是用一个判题程序来处理所有任务，这样的判题队列会很长，会导致队列尾部的提交得不到响应。如果我们可以选择多个判题程序同时运行，有可能会因为资源使用过度导致前端和后端的系统不能正常运行。

为了解决上述问题，我们需要一个判题系统，能在多台机器上协同合作。这一系统应具备均衡负载功能，能够智能地分配判题任务给不同的机器，确保每台机器都能充分利用其资源，同时避免资源过度集中导致的性能瓶颈。通过这样的设计，我们能够在保证系统稳定性的同时，实现高效的判题效率，为用户提供更加优质的测评体验。因此本OJ测评模块示意图如下：

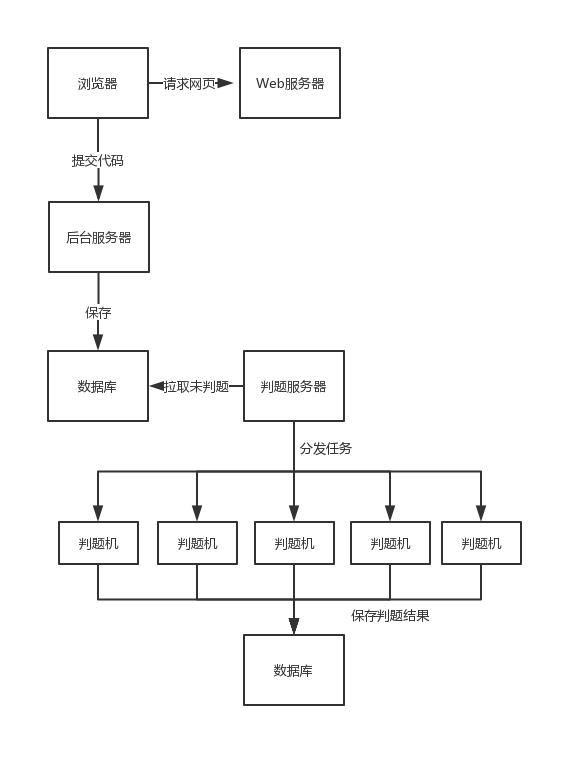


图4-5 OJ测评模块

4.6基于回归分析的服务器选择优化

4.6.1 回归分析算法

在OJ开发中，回归分析可以基于历史评测数据，如题目的平均运行时间、内存使用情况等，建立资源需求预测模型，本系统采用的是多元线性回归。通过这个模型，可以预测新提交的代码可能需要的资源量，从而实现智能的资源分配。例如，对于预测需要较多资源的提交，可以将其分配到性能更强的评测机上，或者优先处理；而对于资源需求较小的提交，可以进行并行处理或分配到资源较少的评测机上。这样可以避免资源的浪费，提高评测效率，加快评测速度，优化用户体验。同时，通过不断收集新的评测数据和反馈，回归分析模型可以持续更新和优化，以适应不断变化的OJ系统和用户需求[9]。

我们假设代码平均运行时间、内存使用情况、代码长度等条件为 x1, x2,x3...,参数为w1，偏置为b，从用户数据去获取数据，并形成向量后，可得预测函数：

并我们可以使用最小二乘法或者梯度下降法求解。

4.6.2回归分析算法的实现

假设AW执行：

def work(limit):

contidions = ['x1', 'x2', 'x3', 'x5', 'x6']

for i in range(len(contidions)):

d= data[contidions]

x = sm.add\_constant(data1) #生成自变量

y = data['y'] #生成因变量

model = sm.OLS(y, x) #生成模型

result = model.fit() #模型拟合

pervalues= result.pervalues#得到结果中所有P值

pvalues.drop('const',inplace=True) #把const取得

permax= max(pvalues) #选出最大的P值

if permax>limit:

ind = pvalues.idxmax() #找出最大P值的index

contidions.remove(ind) #把这个index从cols中删除

else:

return result

result = work(0.05)

result.summary()

得到的多元线性模型为y =c0 + c2 x1 + c3 x2 + c4 x3 + c5 x5，(c为计算出来的参数)这个就是我们最终要用到的有效的多元线性模型。我们可以从数据库中获取影响判题因素的信息，做类似的运算，跑出模型。通过这个模型，可以预测新提交的代码可能需要的资源量，从而实现智能的资源分配[10]。

4.6.3 后端实现

utex = threading.Lock() # queue mutex

queue = Queue() # 全局判题列表

myjsonfile = open("./setting.json", 'r')

judgerjson = json.loads(myjsonfile.read())

if os.environ.get("DB\_USER"):

judgerjson["db\_ip"] = os.environ.get("DB\_HOST")

judgerjson["db\_pass"] = os.environ.get("DB\_PASSWORD")

judgerjson["db\_user"] = os.environ.get("DB\_USER")

judgerjson["db\_port"] = os.environ.get("DB\_PORT")

try:

db = MySQLdb.connect(judgerjson["db\_ip"], judgerjson["db\_user"], judgerjson["db\_pass"],

judgerjson["db\_database"], int(judgerjson["db\_port"]), charset='utf8')

except Exception as e:

print(e)

exit(1)

# 获取未判题列表，放入到全局队列中

def getSubmition():

global queue, mutex, db

cursor = db.cursor()

while True:

sleep(1)

if mutex.acquire():

cursor.execute(

"SELECT \* from judgestatus\_judgestatus where result = '-1'")

data = cursor.fetchall()

try:

for d in data:

queue.put(d[0])

cursor.execute(

"UPDATE judgestatus\_judgestatus SET result = '-6' WHERE id = '%d'" % d[0])

db.commit()

except:

db.rollback()

mutex.release()

db.close()

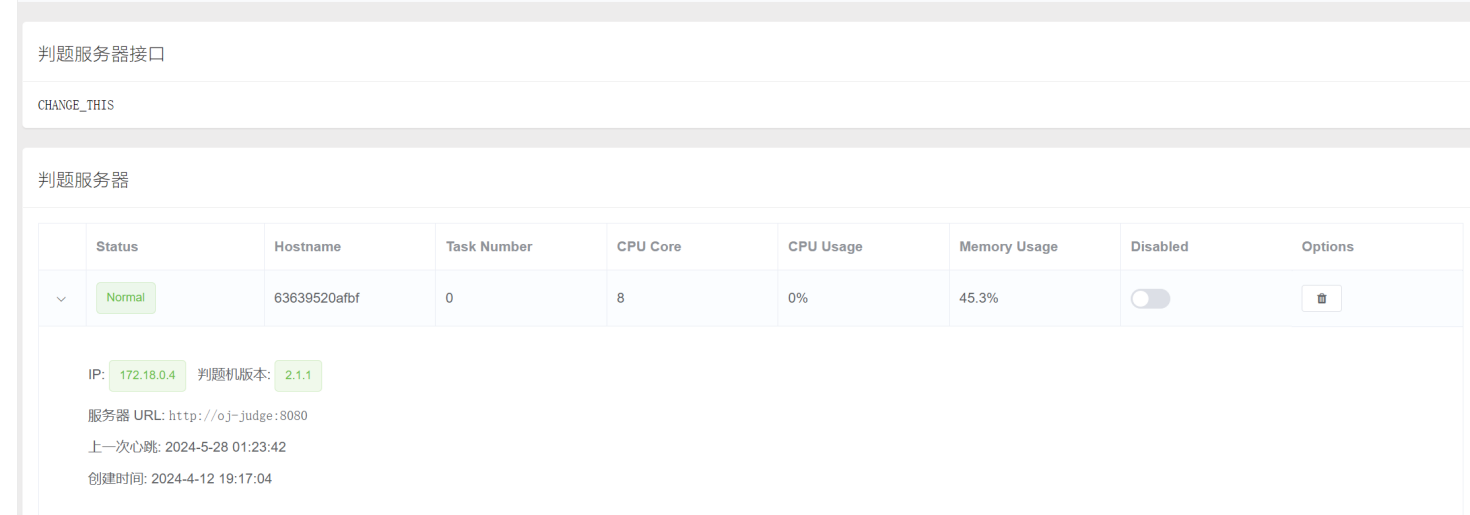


图4-6 服务器选择版本

4.3.7 Special judge

在处理可以接受多种正确答案的题目时，为了确保准确的评判，通常会采用Special Judge（简称SPJ）机制。SPJ程序是一种特殊的评判程序，它利用输入数据和其他相关信息来评估参赛者程序的输出，并返回相应的评判结果。当管理员打算使用SPJ时，要确保在提交题目数据时，将特判程序（通常是命名为spj.cpp的文件）一并上传。这样做是因为，如果缺少特判程序，判题系统将默认将题目视为一般题目，采用标准的评判规则。重要的是要注意，即使管理员在添加题目时没有明确设置SPJ标记，但只要上传了名为spj.cpp的特判程序，判题系统也会自动识别该题目为需要特殊评判的题目，并调用相应的SPJ程序进行评判[10]。

使用SPJ的数据包应该有这些文件：1.in, 1.out, 2.in, 2.out....spj.cpp

然后spj.cpp的写法为：

#include<stdio.h>

using namespace std;

int main(int argc, char\* argv[]) {

FILE \* f\_in=fopen(argv[1],"r"); //测试输入

FILE \* f\_out=fopen(argv[2],"r"); //测试输出

FILE \* f\_user=fopen(argv[3],"r"); //用户输出

int ret=0; //AC=0, WA=1, 其他均为 System Error

/\*\*\*\*\*spj代码区域\*\*\*\*\*\*\*/

// 以下是一个a+b的例子

int a,b,c,d;

fscanf(f\_in,"%d %d",&a,&b);

fscanf(f\_out,"%d",&c);

fscanf(f\_user,"%d",&d);

if(c==d)

ret=0;

else

ret = 1;

/\*\*\*\*\*spj-end\*\*\*\*\*\*\*\*/

fclose(f\_in);

fclose(f\_out);

fclose(f\_user);

//如果要想输出更多的信息，我们可以在相同的目录下输出一个叫做 spjmsg.txt 的文件，如果返回1时，系统会读取spjmsg.txt中的内容，显示在判题信息中。

#include <fstream>

ofstream out("spjmsg.txt");

if (out.is\_open())

{

out << "This is a line.\n";

out << "This is another line.\n";

out.close();

}

return ret;}

**第五章 系统测试**

**5.1 测试目的**

系统测试的目的是确保软件系统在满足业务需求和用户期望的同时，具备高质量、稳定性和可靠性。具体来说，系统测试主要有以下几个目标：

1. 验证系统的功能：系统测试旨在检查软件系统是否按照需求规格说明书中所定义的功能工作。测试人员将执行各种测试用例，包括正常操作、异常情况和边界条件，确保系统在各种情况下都能正确运行。
2. 发现和修复缺陷：系统测试是为了发现系统中的缺陷和问题。通过模拟真实用户的使用场景，并执行各种操作和输入，测试人员可以有效地发现系统中的错误和异常行为。这些缺陷将被记录、报告并优先修复，以提高系统的质量和可用性。首先，我们将验证系统的基本功能是否能够正常运行，包括代码提交、自动评测和结果显示等功能。接着，我们会评估系统的性能，检查其响应时间和处理能力是否符合我们的预期。我们还将对系统的稳定性进行测试，以确保其在长时间运行和高负载情况下仍能保持稳定。安全性测试也是我们的重点之一，我们将确保系统能够保护用户数据和自身安全，防止恶意代码执行和数据泄露。此外，我们还将检查系统的兼容性，确保其能够在不同的操作系统、浏览器和网络环境下正常工作。我们关注用户体验，将评估系统的界面设计和操作流程是否符合用户的使用习惯。错误处理能力也是我们需要测试的重点，我们将检查系统对错误输入和异常情况的处理能力。我们还将测试系统的可靠性，确保同一份代码在不同时间提交得到的评测结果相同。最后，我们将评估系统的可扩展性和维护性，以确保其能够方便地增加新的功能和题目，同时便于后续的维护和升级。
3. 评估系统的性能：系统测试还涉及对软件系统的性能进行评估。通过模拟并压力测试系统，测试人员可以确定系统在不同负载和压力情况下的性能表现，并找出潜在的性能瓶颈和资源利用不足的问题。
4. 验证系统的兼容性：系统测试还包括验证系统在不同硬件、软件和操作系统环境下的兼容性。测试人员将测试系统在各种配置和组合中的功能和性能，以确保系统能够在广泛的环境中正常工作。

通过系统测试，可以提供给开发团队和用户一个准确的软件质量评估，帮助找出并解决潜在的问题和风险，确保软件系统能够稳定、可靠地运行，并满足用户的期望和需求。

**5.2 测试用例**

以下是对《基于决策树和回归分析的在线评测系统的开发与优化》的部分功能的测试。具体测试用例如下。

表5-1 用户登录测试表

|  |  |  |  |
| --- | --- | --- | --- |
| 功能名称 | 用户登录 | 测试序号 | 01 |
| 测试时间 | 2024年5月20日 | 测试人员 | 开发者 |
| 测试目的 | 测试系统登录功能是否正常、稳定 | | |
| 测试步骤 | 1. 用户要登录本系统 2. 输入用户名“root”，密码“rootroot”点击登录按钮   （3）输入用户名“root”，密码“123456”点击登录按钮 | | |
| 预测结果 | （1）通过后台服务器与数据库的信息匹配正确后，提示“登录成功”，并进入主界面，用户root可以正常登录系统。  （2）经过客户端代码对输入的数据进行格式规范校验后，提示“用户名或密码不对，请重新输入”。 | | |
| 实际结果 | （1）通过后台服务器与数据库信息匹配正确后，提示“登录成功”，并进入主界面，用户root可以正常登录系统。  （2）经过客户端代码对输入的数据进行格式规范校验后，提示“用户名或密码不对，请重新输入” | | |
| 功能名称 | 用户登录 | 测试序号 | 01 |
| 测试时间 | 2024年5月20日 | 测试人员 | 开发者 |

表5-2 判题机测试表

|  |  |  |  |
| --- | --- | --- | --- |
| 功能名称 | 判题机 | 测试序号 | 02 |
| 测试时间 | 2024年5月20日 | 测试人员 | 开发者 |
| 测试目的 | 测试判题机能否正确判断，并返回正确的提交结果 | | |
| 测试步骤 | 1. 进入题目1 2. 提交能正确运行代码 3. 提交含有语法错误的代码 4. 提交数组越界代码 5. 提交死循环代码 | | |
| 预测结果 | 代码提交以后通过判题机的运行与数据库的答案进行比对以后，不同的测试步骤应有不同的结果：   1. 提交状态为：答案正确 2. 提交状态为：编译失败 3. 提交状态为：运行时错误 4. 提交状态为：运行超时 | | |
| 实际结果 | 代码提交以后通过判题机的运行与数据库的答案进行比对以后，不同的测试步骤有不同的结果：图6-1  （1）提交状态为：答案正确  （2）提交状态为：编译失败  （3）提交状态为：运行时错误  （4）提交状态为：运行超时 | | |

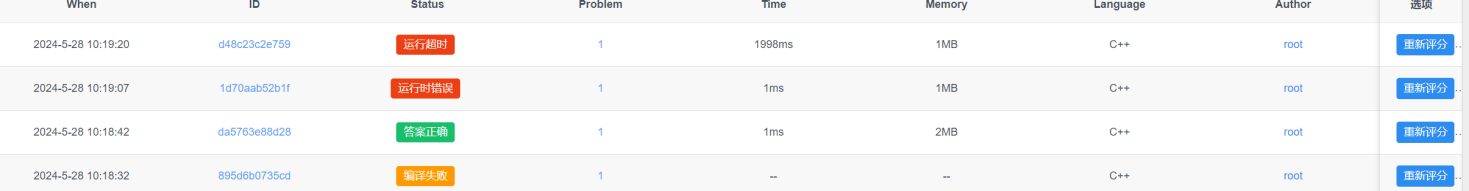


图5-1 测试结果

表5-3 比赛管理测试表

|  |  |  |  |
| --- | --- | --- | --- |
| 功能名称 | 比赛管理 | 测试序号 | 03 |
| 测试时间 | 2024年5月20日 | 测试人员 | 开发者 |
| 测试目的 | 测试管理员能添加比赛的功能 | | |
| 测试步骤 | 1. 进入后台管理的比赛管理页面 2. 点击创建比赛 3. 依次输入比赛标题、比赛描述等信息 4. 进入比赛问题列表界面 5. 添加比赛用题 | | |
| 预测结果 | 从普通用户的角度来看，应该能进行以下操作   1. 能看到比赛列表找到比赛信息 2. 点开比赛以后能看到具体描述 3. 点开题目，能在规定时间内交题 4. 能看到比赛可视化榜单 | | |
| 实际结果 | 以上功能都能正确实现:图6-2， 6-3 | | |



图5-2比赛界面



图5-3比赛结果可视化

**参考文献**

1. 李洪昌,周磊.基于Python Django框架的多媒体发布系统[J].物联网技术,2018,8(2):39-41.
2. 何迎生,罗强. Online Judge评判内核的设计与实现[J]. 吉首大学学报(自然科学版),2010,06:37-39.
3. 莫文水.Web前端中MVVM框架的应用研究[J].网络安全技术与应用,2017(4):64-64.
4. Yildiz, O.T.,Alpaydin, E.Omnivariate decision trees. IEEE Transactions on Neural Networks . 2001.
5. 游俊慧.MVC、MVP、MVVM三种架构模式的对比[J].办公自动化,2020(22):11-12.
6. 程亚维,李攀.基于J2EE的Web应用程序开发研究[J].数字通信世界,2021(9):88-89.
7. 刘梅娇,曹炳元.Fuzzy线性规划最优解的新探及扩展.中国系统工程学会模糊数学与模糊系统委员会第十一届年会论文选集[C].2002.
8. 吴春梅,蒋林利,余荣川.基于Python和Django框架的二级学院资料室图书管理系统设计与实现[J].无线互联科技,2020,17(16):67-70.
9. 刘梅娇,曹炳元.Fuzzy线性规划最优解的新探及扩展.中国系统工程学会模糊数学与模糊系统委员会第十一届年会论文选集[C].2002.
10. 任雪松,于秀林.多元统计分析[M].中国统计出版社:201012.393.

**致 谢**

首先，我要感谢合肥大学，感谢你为我提供了一个良好的学习环境和广阔的发展空间。在这里，我接受了全面的素质教育，拓宽了视野，丰富了知识，也结交了许多优秀的同学和朋友。学校严谨的学术氛围和丰富的课外活动，让我在追求学术的同时，也锻炼了自身的能力和素质。感谢你，合肥大学，是你为我打开了通往未来的大门。

胡老师，您的教诲和关怀，是我大学生活中最宝贵的财富。您不仅教会了我专业知识，更教会了我做人的道理。您的严谨治学、和蔼可亲，让我明白了教师这个职业的伟大。您的鼓励和支持，让我在困难面前坚定了信心，勇往直前。感谢您，胡老师，您的辛勤付出，让我收获了成长和成功。

家人，你们是我最坚实的后盾和最温暖的港湾。在我求学的道路上，你们一直默默地支持我、鼓励我，给予我无尽的爱和关怀。你们的付出和牺牲，让我有了追求梦想的勇气和力量。感谢你们，亲爱的家人，是你们陪伴我走过了这段不平凡的旅程。

最后由衷感谢ACM实验室，感谢ICPC的所有承办学校、企业，能让我以赛促学，一直保持对算法、对编程的热爱，打下扎实的基础。

在这个毕业的时刻，我要向你们深深地鞠一躬，表达我最真诚的感谢。未来的路还很长，我会带着你们的期望和祝福，继续前行，为实现自己的梦想而努力。再次感谢你们，愿我们都能在人生的道路上，收获满满的幸福和成功。