

班 级：计科 1907
学 号：2019040504



北京化工大学

毕业设计(论文)

题 目 ACM 竞赛训练数据分析系统的设计与开发

专 业 计算机科学与技术

学 生 蒋恒

指导教师 刘勇(副教授)

2022 年 6 月 3 日

诚信声明

本人声明：

本人所呈交的毕业设计（论文），是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，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结果由本人承担。

作者签名：  日期： 2022年6月3日

毕业设计(论文)任务书

设计(论文)题目: ACM 竞赛训练数据分析系统的设计与开发

学院: 信息科学与技术学院 专业 计算机科学与技术 班级: 计科 1907

学生: 蒋恒 指导教师: 刘勇(副教授) 专业负责人: 韩永明

1. 设计(论文)的主要任务及目标

本毕业设计的主要任务是使用现代 Web 技术设计并开发一套 ACM 竞赛训练数据分析系统, 实现自动数据收集、数据处理、数据展示和智能化分析功能, 用于提升大学生程序设计竞赛训练过程的质量和效果。

2. 设计(论文)的主要内容及要求

(1) 面向 codeforce 和 atcoder 两个流行的算法竞赛平台, 研发数据自动爬取系统;

(2) 基于前后端分离技术, 开发面向 acm 竞赛的数据管理系统, 展示竞赛队员在算法竞赛平台的参赛、做题、补题等数据;

(3) 基于企业微信, 研发信息通知机器人, 自动推送比赛预告和结果信息, 更好的督促竞赛队员参加训练;

(4) 根据竞赛队员的训练数据, 题目分类等信息, 智能化分析竞赛队员的水平, 查漏补缺, 给出针对性的训练建议。

3. 主要参考文献

[1] 周晓琳,解静,刘勇,尤枫,吴佳伟.面向程序设计竞赛的人才培养模式实践与探索[J].大学教育,2020(07):144-146.

[2]Hou Y. The design and implementation of the framework for Spring+SpringMVC+MyBatis in the development of Web application[C]//Institute of Management Science and Industrial Engineering.Proceedings of 2019 4th International Industrial Informatics and Computer Engineering Conference(IIICEC 2019).Francis Academic Press,2019:369-374.DOI:10.26914/c.cnkihy.2019.039191.

4. 进度安排

	设计(论文)各阶段名称	起 止 日 期
1	需求收集	2022.11.29--2022.12.15
2	功能设计	2022.12.15--2023.1.1
3	系统设计	2023.1.1--2023.2.1
4	代码编写及调试	2023.2.1--2023.4.1
5	撰写报告	2023.4.1--2023.5.1

ACM 竞赛训练数据分析系统的设计与开发

摘 要

正文：（四号宋体，段落按照“首行缩进，2 字符”格式，标点符号占一格）**建议 150 字左右，不要分段、不要设序号。**

关键词：算法竞赛，数据收集，爬虫，系统开发

（分页符）题目：三号 Times New Roman”字体，全部采用大写字母，可分成 1-3 行居中打印。每行左右两边至少留五个字符空格。段后 3 行

ABSTRACT

（小三“Times New Roman”字体，段后 2 行打印英文摘要内容，英文摘要与中文摘要相对应）

正文：摘要内容每段开头留四个字符空格，四号“Times New Roman”字体，段后 2 行，22 磅

KEY WORDS:（四号“Times New Roman”字体关键字段前 2 行，其后关键词小写，每个单词首字母大写，最少三个，不超过 4 个）

英文一律采用“Times New Roman”字体

目录

前 言	1
第 1 章 绪论	2
1.1 研究背景及意义	2
1.2 国内外研究现状	4
1.3 本文研究内容	7
1.4 本文组织结构	7
第 2 章 相关理论及技术综述	9
2.1 B/S 系统架构	9
2.2 前端技术框架	10
2.2.1 MVVM 模式	10
2.2.2 Vue.js 简介	11
2.2.4 ajax 与 axios 工具	11
2.2.5 ElementUI	12
2.3 后端技术框架	12
2.3.1 SpringBoot 框架	12
2.3.2 Spring Data JPA	13
2.4 数据库技术	13
2.4.1 MySQL 数据库	13
2.4.2 Redis 数据库	14
2.5 本章小结	14
第 3 章 系统需求分析	16
3.1 北京化工大学 ACM 竞赛训练现状	16
3.1.1 训练平台与方式	16
3.1.2 学生竞赛能力的量化与评估	16
3.1.3 教练工作的繁琐性	16
3.2 系统功能需求分析	16
3.2.1 系统数据管理	17
3.2.2 学生数据管理	20
3.2.3 外部数据收集	26
3.2.4 数据展示	27
3.2.5 系统智能提示	29
3.2.6 智能训练	29
3.3 非功能需求分析	32
3.3.1 性能需求	32
3.3.2 自动运维需求	33
3.4 本章小结	34
第 4 章 系统设计	35
4.1 系统结构设计	35

4.1.1 系统总体架构设计	35
4.1.2 系统流程设计	36
4.2 数据库设计	36
4.2.1 数据库 E-R 模型设计	36
4.2.2 数据表设计	38
4.3 数据获取模块	42
4.3.1 数据获取模块架构设计	42
4.3.2 爬虫终端设计	42
4.3.3 爬虫调度端架构设计	44
4.3.4 爬虫接收端架构设计	45
4.4 业务端(数据管理模块与数据展示模块)设计	46
4.4.1 基础架构设计	46
4.4.2 系统管理功能设计	48
4.4.3 学生数据管理功能模块设计	49
4.4.4 系统智能提示功能设计	49
4.4.5 数据展示功能设计	50
4.4.6 智能训练功能设计	51
第 5 章 系统实现与测试	53
第 6 章 总结与展望	58
6.1 总结	58
6.2 展望	58

前 言

本篇论文将探讨程序设计竞赛（**Programming Contest**）辅助工具的设计与开发。

随着互联网的普及以及计算机学科的发展，越来越多的本科生开始参与到程序设计竞赛中。而要在程序设计竞赛中获得好成绩，需要进行大量且合理的训练。在此过程中，竞赛教练的指导是必不可少的。然而，随着学生的增多，竞赛教练需要管理的人数越来越多，教练迫切需要一些辅助工具来协助其日常管理。

基于此前提，本毕业设计开发了一个面向竞赛教练的竞赛数据分析平台，旨在辅助竞赛教练进行日常管理。该平台为竞赛教练提供了一些便捷的辅助工具，以协助他们更有效地管理参赛学生和日常训练任务。同时，该平台也面向学生提供一些辅助工具，以帮助他们更好地进行训练。

第 1 章 绪论

1.1 研究背景及意义

国际大学生程序设计竞赛（英语：International Collegiate Programming Contest, ICPC）是一项旨在展示大学生的创新能力、团队精神以及在压力下编写程序、分析和解决问题能力的赛事。作为全球最具影响力的大学生计算机竞赛之一，自 1997 年中国大陆地区开始举办区域赛和参加世界总决赛以来，ICPC 已成为我国高校科技活动的一个热点，同时也是展示计算机教育成果和优秀人才综合素质的重要活动。

一般的高校学生入学时没有 NOI 经验，有关 ICPC 所需要的知识基础较薄弱，而 ICPC 题目一般难度较大且强调算法的高效性，不仅要解决一个指定的命题，而且必须以最佳的方式在最短的时间内解决指定的命题，要求选手具有一定的理论知识和实践能力，以及编程速度和正确性；更重要是对选手的实践能力和洞察力的要求，这一点相对于理论和技术而言更难培养。

在这大背景下，相关的辅助系统也应运而生，最早是以在线评测系统这一类系统形式出现的。^[4] 在线评测系统 (Online Judge, 简称 OJ) 是一种用于自动评测算法代码正确性的系统。在 OJ 上，用户可以提交自己编写的程序代码，经过系统测试后获得相应的得分和反馈，以及运行时间和内存使用情况等信息。^[5]

OJ 系统不仅能用于竞赛培训，还可以用于实践教学。在教学方面，各大高校将该系统直接应用于程序设计类课程的上机实验教学中。任课教师根据程序设计类课程的实验内容和具体要求，合理布置实验任务，新增题目或者从题库中选择合适的题目供学生上机练习。^[6]

除了部署在校内的 OJ 之外，在互联网上也有许多著名的 OJ，比如美国弗吉尼亚大学的 UVA OJ 系统^[7]、俄罗斯萨拉托夫国立大学的 SGU OJ 系统、俄罗斯乌拉尔国立大学的 URAL OJ 系统。此外，还有一些国外公司开发的知名网站，包括 Topcoder 公司的 Topcoder 网站、Directi 公司的 CodeChef 网站、Codeforces 公司的 Codeforces 网站等。

在以上的互联网 OJ 平台中，广为人知且获得众参赛队员接受的平台是 Codeforces 与 Atcoder，各个学校也倾向于使用这两个平台进行日常的训练。

Codeforces 是一个举办编程竞赛的网站。它由 Mikhail Mirzayanov 领导的 ITMO 大学的一群竞赛程序员维护。自 2013 年以来，Codeforces 号称在活跃参赛者方面超

过 Topcoder。截至 2018 年，它已有超过 600,000 名注册用户。Codeforces 每周会不定期举办 1-2 场编程竞赛，赛制类似于 ICPC，但是可以单人参赛。

Atcoder 是一家以举办程序设计竞赛为主要业务的日本公司。目前其每周六会举办一场常规的适宜初学者的竞赛，以及不定期举办适宜高阶编程竞赛人员的竞赛。

在早期的竞赛训练中，由于主要参与人员的人数较少，日常的训练可由教练人工制定标准并前往各个训练平台进行。但随着人员的增加，教练继续人工前往各个网站查看各项训练数据变得困难。

如图 1-1 所示，以北京化工大学的训练参与人数为例，2015 级仅仅只有 3 人参与日常的训练，而到了 2021 级则有 150 人参与训练，教练的工作激增。

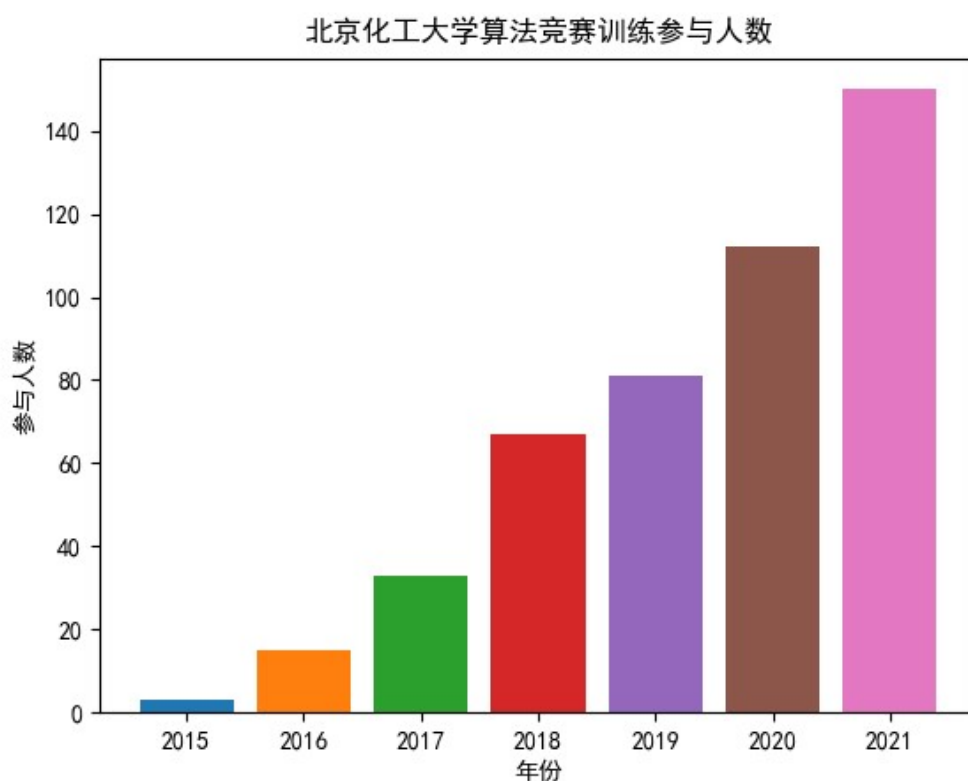


图 1 -1 北京化工大学算法竞赛训练参与人数变化图

如图 1-2 与图 1-3 所示，北京化工大学 ACM 队，近十场 Codeforces 比赛人数最大值达 30 人，最小值有 12 人，平均每场比赛有 18 人参与；近十场 Atcoder 比赛人数最大值达 26 人，最小值有 4 人，平均每场比赛有 16 人参与。这就意味着如果教练使用手工方式去核对比赛人数，至少需要半小时时间去收集和记录下这些数据，考虑到算法竞赛教练一般而言只是兼职，这个时间代价显然是不能接受的。

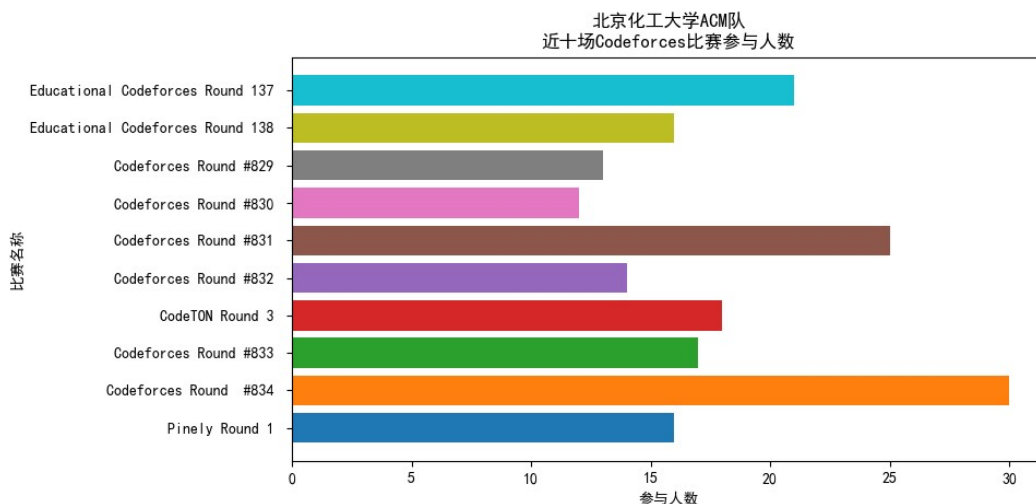


图 1-2 北京化工大学 ACM 队近十场 Codeforces 比赛参与人数

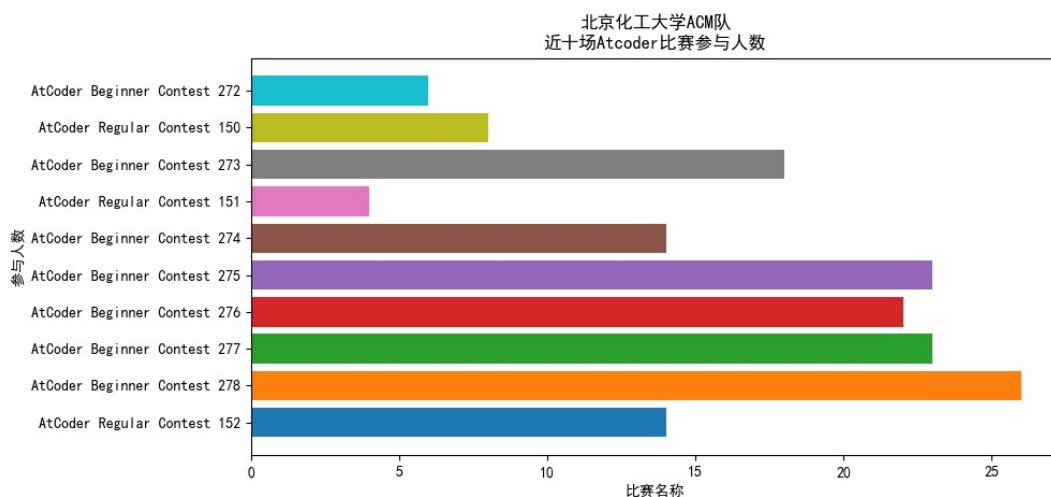


图 1-3 北京化工大学 ACM 队 Atcoder 比赛参与人数

在这个现实背景下，竞赛训练迫切需要一个数据分析系统来辅助竞赛教练进行竞赛训练的日常管理。因此，此次毕业设计为推出了一个专门的算法竞赛数据分析平台，它能够从各个 OJ 网站获取数据，并将其汇总、展示，同时提供可视化数据分析服务。

1.2 国内外研究现状

ACM 竞赛训练在各个高校和公司中的普及程度在近年来呈现不断增加的趋势。随着计算机科学专业的普及和学生对象算法竞赛的兴趣不断增加，越来越多的高校和公司开始重视 ACM 竞赛训练，以培养学生的编程技能和解题能力。

在国内高校中，越来越多的院系开始将 ACM 竞赛训练纳入教学计划，以帮助学生提升编程技能和算法分析能力。同时，一些高校也会在竞赛训练中加入创新创业等元素，以培养学生的创新能力。此外，很多高校也会组建 ACM 竞赛队伍参加国内外各种竞赛，并且取得了不俗的成绩，这也促进了 ACM 竞赛训练在高校中的普及。

在企业中，越来越多的 IT 公司开始注重对编程能力和算法分析能力的考察，特别是一些在算法和数据科学方面有需求的公司。这些公司通常会要求应聘者参加编程竞赛，例如 ACM/ICPC 等，并将这些竞赛结果作为招聘的重要参考。因此，ACM 竞赛训练在企业中也越来越受重视。

目前，ACM 竞赛训练数据的大多数来源都是在线评测（online judge, OJ）系统。这种系统起源于 ACM 国际大学生程序设计竞赛。最早的在线评测系统是由西班牙 Valladolid 大学的西里亚科·加西亚·德·塞利斯于 1995 年开发的，当时用于该校参加 ACM/ICPC 西南欧区域赛选拔队员。经过 ACM 竞赛近 30 年的发展，各大高校纷纷开发了属于自己的在线评测系统。其中，国外知名高校的在线评测系统有美国弗吉尼亚大学的 UVA OJ 系统、俄罗斯萨拉托夫国立大学的 SGU OJ 系统、俄罗斯乌拉尔国立大学的 URAL OJ 系统。此外，还有一些国外公司开发的知名网站，包括 Topcoder 公司的 Topcoder 网站、Directi 公司的 CodeChef 网站、Codeforces 公司的 Codeforces 网站等。

ICPC 已经在国内发展多年，国内现在已经有各种基于在线评测系统的竞赛训练管理平台。在这些国内知名的在线评测系统中，最早开发的在线评测系统是浙江大学的 ZOJ，而提交次数最高的 OJ 系统是北京大学的 POJ。本校（北京化工大学）也拥有一个用于校内训练与比赛的 OJ 系统（BUCTOJ）。然而，这些系统大多都是在提供一个评测系统给学生进行训练的基础上，对现有的系统进行二次开发提供对学生在该训练系统上的数据的分析。

由于各个 OJ 的数据并不共通，且各个 OJ 上都具有一定的优质题目，因此在这种情况下，Xu Han 提出了一种基于爬虫的类 Online Judge 系统：Virtual Judge 系统。Virtual Judge 系统与其他 OJ 平台一样，主要致力于为用户提供一个高效的竞赛训练和测试环境，以提升其编程技能和解题能力。然而，与其它 OJ 平台不同的是，Virtual Judge 系统采用了一种基于爬虫技术的创新模式，可以将各大 OJ 平台的题目缓存在其系统中，并运行用户在自己的系统上提交的代码，通过爬虫技术再次提交到其他 OJ 平台进行判题。这样一来，用户可以在 Virtual Judge 系统中使用多个 OJ 平台的数据进行训练，极大地提高了训练的有效性和灵活性。

需要注意的是，尽管 Virtual Judge 系统在为用户提供训练和比赛环境方面具有明显的优势，但其并不提供数据分析功能，这一点与其他 OJ 平台相同。现有的这些系统基本都专注于学生在系统自身上进行训练的数据，并不注重竞赛教练对于训练或比赛结果的分析，忽视了算法竞赛教练的需求。

OJ 系统的数据不共通是目前现有系统中存在的一个主要问题。不同的 OJ 系统拥有自己的题库，这就导致了数据不共通，使得各个系统之间的数据交流和汇总变得十分困难，也使得数据分析受到限制。这种情况下，ACM 竞赛训练数据分析系统的设计与开发就显得尤为重要。

由于数据不共通，竞赛教练需要花费大量的时间和精力在各个 OJ 系统上进行查找和筛选，以便找到合适的题目进行训练。这不仅浪费了大量的人力资源，还使得训练的效率受到了很大的限制。同时，由于各个 OJ 系统的数据不共通，不同系统上的题目难度和类型也会存在差异，这使得训练的公平性和有效性受到了影响。

另外，也是由于数据不共通的原因，现有系统中的数据分析功能十分有限。大多数 OJ 系统主要提供评测和做题功能，数据分析功能较少，这使得竞赛教练很难对学生在训练过程中的表现进行深入的分析和挖掘。例如，竞赛教练很难获得学生在多个 OJ 系统上的训练数据，并进行全面的统计和分析，以便更好地评估学生的编程技能和算法分析能力。这使得现有系统对于竞赛教练的需求很难得到满足。OJ 系统的数据分析功能相对不足：大多数 OJ 系统主要提供评测和做题功能，数据分析功能较少，难以满足对竞赛数据的深入分析和挖掘需求。

在过去的十几年间，也有部分院校针对 ACM 竞赛训练的流程做出了一些针对性的辅助系统。如山东理工大学曾开发过一个 ACM 竞赛训练计划管理系统，用以管理 ACM 队训练的日常工作，集宁师范学院在 2014 年时曾设计并实现过一个 ACM/ICPC 培训管理系统，但上述系统均针对的是 ACM 竞赛训练的流程，而并没有将竞赛数据进行分析。

竞赛训练管理系统的出现一定程度上简化了竞赛训练的管理和组织工作，然而，现有系统往往只关注竞赛训练的管理和组织，忽视了竞赛训练的数据的利用，无法利用竞赛数据更好地优化竞赛训练的流程，这也是现有系统存在的不足之处。

现有系统中往往只提供了基本的管理和组织功能，例如，学生信息管理、竞赛日程安排等，这些功能虽然有助于竞赛训练的组织和管理，但并不能充分利用竞赛数据优化竞赛训练的流程。例如，现有系统往往无法提供全面的竞赛数据分析功能，竞赛教练无法对学生在训练过程中的表现进行深入的分析和挖掘。

1.3 本文研究内容

本论文对 ACM 竞赛数据分析系统的设计与开发做了详细论述，首先分析了 ACM 竞赛训练的发展情况，查阅相关技术基础与理论，调研了算法竞赛日常训练的流程，确定了数据分析系统的业务需求，然后使用前后端分离的开发流程，使用 Vue3、SpringBoot 等技术框架完成系统的代码编写工作，最后进行测试，最终实现了一个为 ACM 队日常训练服务的数据分析系统。具体的研究内容为以下方面：

（1）竞赛训练流程分析与具体技术资料收集整理：针对高校 ACM 竞赛训练流程与教练相关需求进行分析，并对实现 ACM 竞赛训练管理系统的相关技术进行调研整理，主要包括 Vue、Axios、MySQL、Mysql、Redis 等。

（2）系统需求分析与概要设计：根据 ACM 竞赛训练流程的情况进行了详细的研究，并设计了系统的总架构，根据各项需求划分具体的功能模块，明确了每个系统子模块的功能点，对系统的功能进行需求分析工作。

（3）前后端设计与实现：根据 ACM 竞赛数据分析系统的需求完成了软件的详细设计并进行编码实现，设计 Web 前端页面与后端服务交互，划分系统功能模块，并在类图、时序图的帮助下，对系统的功能模块进行了详细介绍，完成相关软件开发工作。

（4）系统测试：根据系统的实际运行环境对系统进行了测试工作，编写测试用例并进行测试，验证系统相关功能是否满足需求。

1.4 本文组织结构

本文描述了 ACM 竞赛训练数据分析系统从系统需求分析到测试各个阶段的内容，具体内容一共分六章。

第 1 章. 绪论，本章主要分析本文研究背景、国内外研究现状等基础内容，并对本文的结构与内容进行了基本阐述。

第 2 章. 相关理论与基础概述。本章针对系统设计中用到相关技术与理论进行了基本的描述，包括 B/S 架构、前后端分离、Vue.js 框架、SpringBoot 框架、MySQL 和 Redis 数据库等。

第 3 章. 系统需求分析。根据具体的 ACM 竞赛训练流程，提取系统的功能需求，完成系统的需求分析。主要包括数据收集与整合、数据分析与汇总、自动提醒功能、个性化竞赛辅助等方面。

第 4 章. 系统设计。根据系统的需求建立起合理的系统架构，确定每一模块的实现

第 5 章. 系统实现与测试。本章根据系统的设计进行具体的功能实现，详细介绍各个模块中的各个功能的具体实现以及其实现效果。包括前端界面设计、后端接口实现等。并对系统进行功能测试与性能测试，验证系统是否满足预期目标

第 6 章. 总结与展望。本章总结本文的研究内容，分析目前系统的不足之处，并展望系统的优化与发展。针对系统可能存在的问题，提出改进措施和未来发展方向。

第 2 章 相关理论及技术综述

2.1 B/S 系统架构

B/S 结构是一种由 C/S 结构改进而来的架构，区别于传统的 C/S 结构，B/S 结构使用浏览器(Browser)作为客户端，这种模式使得其随着网络普及逐渐发展。系统功能主要由服务器实现，而开发人员则负责设计前端页面和系统代码。用户可以通过浏览器访问服务器，从而减轻了客户端开发的负担，简化了系统的开发、维护和使用。

如图 2-1 所示，传统的 C/S 架构客户端程序直接依赖于操作系统，不易使用和移植，而 B/S 架构中应用依赖于浏览器，而浏览器在大多数操作系统中均有实现，可以在多数地方使用

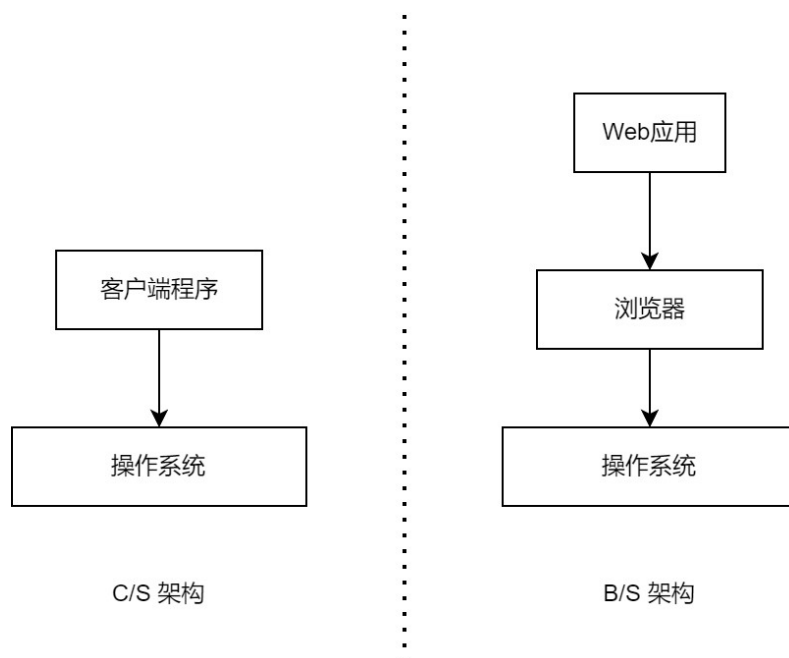


图 2-1 C/S 架构与 B/S 架构对比

B/S 模式的优点是易于使用，成本低，这也是为什么它在现代化的应用开发中被广泛采用的原因之一。此外，开发人员可以从不同的地点和方式访问和操作项目，这样就增加了灵活性和效率。在安全性方面，B/S 结构能够有效地保护平台的访问权限和数据库，从而确保了数据的安全性。

但它也有一定的缺点。例如，服务器需要维护与管理，这会增加成本和风险。此外，数据压力大，当服务器崩溃时，修复时间较长，这会影响系统的正常运行。

2.2 前端技术框架

随着 Web 应用的发展，前端的功能需求不断增强，传统的 Web 前端框架包含 HTML、CSS 和 JavaScript 已经难以满足开发者的需求。在这种情况下，新的前端框架开始出现，旨在解决这些问题。早期的 Web 前端框架主要负责页面的结构、样式和交互等方面，但是在实现复杂功能时，前端需要调用后端接口来完成，这限制了前端的发展空间。

为了解决这些局限性，MVVM（Model-View-ViewModel）模式出现并逐渐流行。在这种模式下，前端框架通过数据绑定实现了前后端分离，前端负责页面的呈现和交互逻辑，而后端则负责数据的处理和存储。这种方式可以大大提高前端的开发效率和代码的可维护性。

在国外，前端框架的发展较快，率先推出了多个优秀的前端框架。目前，主流的前端框架包括 Vue.js、React 和 AngularJS 等。这些框架在不同的领域有着各自的优势，如 Vue.js 注重轻量级、易上手、性能高、生态丰富等方面，React 则强调组件化开发和虚拟 DOM 等技术，AngularJS 则更适合大型项目的开发。

在本次毕业设计中，由于我们同时需要完成前端与后端的开发，为了节省开发时间，所以我们选用了 Vue.js 这一前端框架。

2.2.1 MVVM 模式

传统 Web 项目采用服务器端渲染，即在后端代码中嵌入前端代码，实现数据的加载和页面的生成。传统页面相对简单，交互性弱，数据处理和呈现方式也比较单一。然而，随着 Web 应用规模的扩大和技术的不断发展，传统的 Web 开发模式已经无法满足大型 Web 应用的需求，需要将前端和后端分开开发。

随着 JavaScript 技术的发展，浏览器端分层架构的 MVVM 模式开始出现，这种模式通常被应用于特定的组件或单个页面，其核心思想是将视图展示层和视图模型层分开，通过双向数据绑定实现状态的自动传递。

在 MVVM 模式中，视图展示层负责页面的展示，视图模型层则负责处理数据和逻辑，并提供与视图展示层交互的接口。视图展示层和视图模型层之间通过双向数据绑定建立联系，使得视图展示层中的数据和视图模型层中的数据保持同步。当视图模型层的状态发生改变时，这些改变可以自动传递到视图展示层，从而实现动态的数据更新。

双向数据绑定是 MVVM 模式的重要特点，它使得视图展示层和视图模型层之间

的数据同步变得更加方便和高效。通过双向数据绑定，开发人员可以更加专注于数据和业务逻辑的处理，而无需过多考虑数据和页面的同步问题，从而提高开发效率。

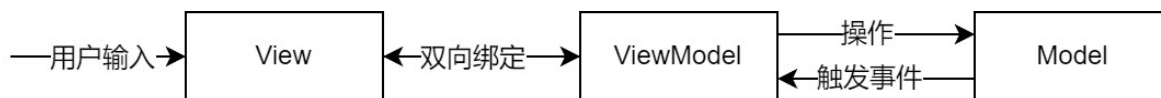


图 2-2 MVVM 模式架构图

2.2.2 Vue.js 简介

Vue 是一款渐进式 JavaScript 框架，于 2014 年发布。它采用自底向上增量开发方法，使得开发者可以逐渐引入 Vue.js 的特性，而无需一开始就投入大量的时间和精力。

Vue.js 采用 MVVM 架构，将视图展示层和视图模型层分离。相对于传统的 MVC 模式，MVVM 模式更加易于上手，特别是针对视图模型层。在 Vue.js 中，开发人员只需要简单地绑定数据和 DOM 元素，就可以实现自动的数据更新，DOM 和 Data 保持同步，无需手动操作，从而简化了代码。

Vue.js 还支持组件化开发，通过组件化封装，实现了代码的复用，同时提高了系统的可扩展性和开发速度。通过封装不同的组件，可以将页面分成若干个小块，便于维护和管理。这种组件化的开发方式也非常适合团队协作开发，开发人员可以根据自己的职责和技能，分别负责不同的组件，最后将组件整合到一个完整的系统中。

2.2.3 VueRouter 路由管理

Vue router 是 Vue.js 的官方路由插件，负责管理页面之间的转换指向。在过去，页面转换通常由后端控制器控制，通过超链接实现。然而，随着 Web 应用规模的扩大和前端技术的发展，使用 Vue router 可以实现前端控制页面跳转过程。

使用 Vue router 后，页面跳转过程由前端实现，通过路径切换相应组件。Vue router 在 Web 端完成 HTML 渲染，将路径映射到相应的组件，并在组件间切换。这种方式使得页面之间的跳转更加快速、平滑，并提高了用户体验。

Vue router 的基本作用是将路径映射到相应组件，并在组件间进行切换。它可以让开发者更加方便地组织代码，实现模块化开发。同时，Vue router 也提供了丰富的路由功能，如路由参数、路由组件传参、路由导航守卫等，可以满足不同的业务需求。

2.2.4 ajax 与 axios 工具

Ajax（Asynchronous JavaScript and XML）是一种在 Web 应用程序中使用的技术，它允许在不刷新整个页面的情况下异步地与服务器交换数据。Ajax 背后的基本思想

是使用 JavaScript 发送异步 HTTP 请求到服务器，从而可以在后台处理数据并返回结果，而不需要重新加载整个页面。这使得 Web 应用程序更加灵活和动态。

在早期的 Ajax 应用程序中，通常使用 XML（可扩展标记语言）格式来传输数据。但是，随着时间的推移，开发人员发现使用 XML 传输数据存在一些问题，比如 XML 语法过于复杂、解析和处理速度较慢等等。因此，随着技术的发展，Json（JavaScript 对象表示法）成为了更受欢迎的数据格式。

axios 则是一款基于 Promise 的 JavaScript HTTP 客户端库，用于在浏览器和 Node.js 中发送 HTTP 请求。通过它可以实现 ajax 技术发送或接收 json 数据。

Axios 提供了许多功能，包括：支持 Promise API：Axios 的请求和响应都是基于 Promise 实现的，这使得在处理异步请求时非常方便。支持请求拦截和响应拦截：可以在请求发送和响应返回的过程中进行拦截，进行统一的处理。自动转换 JSON 数据：Axios 默认会将请求和响应的数据转换为 JSON 格式，可以方便地进行传输和处理。

2.2.5 ElementUI

ElementUI 是为前端开发者量身定做的组件库，它包含了大量的 UI 组件，如折叠菜单、时间控件、轮播图、各种列表等。这些组件灵活多样，能够满足各种 Web 应用的需求。

使用 ElementUI 可以节省编写样式的时间，提高编程速度，适合轻型管理系统开发。开发者可以快速地构建具有吸引力和响应性的用户界面，并且可以方便地自定义组件的样式和行为。同时，ElementUI 也提供了丰富的组件文档和示例，方便开发者查阅和学习使用。

在实际开发中，ElementUI 可以极大地提高开发效率和用户体验，让开发者更加专注于业务逻辑和数据处理。ElementUI 的灵活性和丰富性，也为开发者提供了更多的选择和可能性。

2.3 后端技术框架

2.3.1 SpringBoot 框架

SpringBoot 是由 Pivotal 团队基于 Spring 设计的全新框架，于 2013 年首次发布。它继承了 Spring 框架的优点，同时简化了开发过程，使得开发者可以更加专注于业务逻辑和功能实现，而不是过多地考虑框架的配置和细节。

SpringBoot 的目标是避免重新进行 XML 配置，而不是解决问题。它通过提供默认的配置选项和自动配置机制，使得开发者可以更快地搭建应用程序，同时也可以通过个性化配置来满足特定的需求。

本质上，SpringBoot 是一些库的集合，它为开发者提供了各种常用的库和工具，如 Tomcat、JPA、Jackson 等。这些库和工具已经被预先配置和集成到 SpringBoot 中，开发者可以直接使用，而不需要手动引入和配置。

SpringBoot 极大地简化了 Java 应用程序的开发和部署过程，同时也提高了应用程序的可维护性和可扩展性。它已经成为了 Java 开发领域中非常受欢迎的框架之一。

2.3.2 Spring Data JPA

Spring Data JPA 是一个用于简化 JPA 开发的 Spring 框架模块，它在 JPA 的基础上进行了封装和增强，使得使用 JPA 更加方便和简单。JPA 是 Java Persistence API 的缩写，它提供了一种标准的方式来管理 Java 对象与关系型数据库之间的映射。

Spring Data JPA 提供了一些常用的功能，包括自动生成 CRUD 操作、自定义查询、分页和排序、事务支持等。其中，自动生成 CRUD 操作是 Spring Data JPA 的一大特点，通过定义接口的方式，可以自动生成基本的增删改查操作，大大简化了开发人员的工作。自定义查询也是另一个常用的功能，开发人员可以在接口中定义自己的查询方法，而不必编写 SQL 语句，使得查询更加方便和简单。另外地，Spring Data JPA 还提供了对事务的支持，使得开发人员可以方便地进行事务控制。

2.4 数据库技术

2.4.1 MySQL 数据库

MySQL 是一种开源的关系型数据库管理系统（RDBMS），被广泛用于 Web 应用程序的开发中。它使用 SQL（结构化查询语言）作为操作语言，提供了高效、可靠、安全的数据存储和访问。

MySQL 的特点包括：

- （1）开源：MySQL 是一个完全开源的数据库管理系统，可以免费使用和修改。
- （2）可扩展性：MySQL 支持多种扩展方式，可以根据实际需求进行灵活扩展。
- （3）高性能：MySQL 在大量并发访问时也能够提供快速响应，可以处理大规模数据。
- （4）可靠性：MySQL 提供了多种备份和恢复机制，可以有效地保护数据的安全。

全性和完整性。

（5）安全性：MySQL 提供了多种安全机制，可以控制用户的访问权限，保护数据的安全性。

（6）跨平台性：MySQL 支持多种操作系统，包括 Windows、Linux、Mac 等。

（7）兼容性：MySQL 遵循 ANSI SQL 标准，与其他数据库管理系统（比如 Oracle、Microsoft SQL Server 等）之间也可以进行兼容性操作。

2.4.2 Redis 数据库

Redis 是一款开源的内存型键值存储数据库，也可以用作缓存和消息中间件。Redis 具有高性能、高可靠性、易扩展等特点，被广泛应用于互联网、电商、游戏等领域。

Redis 的主要特点包括：

（1）高性能：Redis 采用内存存储，可以提供高速的读写速度。同时，Redis 还采用多线程和异步 I/O 的方式，可以处理大量并发请求。

（2）支持多种数据结构：Redis 支持多种数据结构，包括字符串、列表、集合、散列、有序集合等，可以满足不同的应用需求。

（3）持久化支持：Redis 支持多种持久化方式，包括快照和 AOF（Append-Only File）两种方式，可以在服务器重启后恢复数据。

（4）高可靠性：Redis 提供了主从复制和 Sentinel 两种高可用方案，可以保证数据的可靠性和高可用性。

（5）事务支持：Redis 支持事务，可以在一次请求中执行多个命令，保证数据的原子性。

（6）Pub/Sub 支持：Redis 支持发布订阅模式，可以实现消息的异步传输。

（7）简单易用：Redis 的命令非常简单，易于学习和使用。

2.5 本章小结

本章对本次毕业设计所采用的主要技术进行了详细的阐述。首先，从 B/S 架构的角度出发，阐述了采用前后端分离式开发的优点，并解释了这种开发模式能够提高项目的可维护性、可扩展性及提高开发效率。在这一基础上，对前端和后端所使用的技术框架进行了详细的介绍。

在前端技术方面，本章介绍了 Vue.js 框架，分析了它的响应式设计、组件化开发、易上手性等优点。此外，还探讨了前后端交互的方式，通过使用 axios 库实现了

基于 ajax 技术的数据交互。这种交互方式有利于提高数据传输的效率，减少了前后端耦合度，使得前后端各自独立地进行开发与维护。

在后端技术方面，本文深入介绍了 SpringBoot 框架，包括其核心理念、自动配置、模块化开发等特点。同时，也分析了 SpringBoot 在实际项目中所展现的高度集成性、灵活性以及便捷性，以证明它在当前后端技术领域中的重要地位。

在数据库技术方面，本文介绍了两种不同类型的数据库：关系型数据库 MySQL 和键值型数据库 Redis。分析了它们在系统中的作用，以及它们各自的优缺点。MySQL 作为关系型数据库，在数据存储和查询方面具有较强的灵活性；而 Redis 作为键值型数据库，在缓存和高并发访问场景下展现出优异的性能。

通过本章的阐述，为后续章节的详细设计与实现奠定了扎实的技术基础。

第 3 章 系统需求分析

3.1 北京化工大学 ACM 竞赛训练现状

ACM 竞赛训练作为一项提高选手在算法、数据结构、编程能力和团队合作方面技能的活动，对于选拔高水平的计算机专业人才具有重要意义。本节将分析北京化工大学 ACM 竞赛训练的现状，以及存在的问题和改进方向。

3.1.1 训练平台与方式

目前，北京化工大学 ACM 竞赛训练主要依赖于校内自有在线评测系统(BUCTOJ)以及外部平台（如 Codeforces、Atcoder 等）进行。学校内的 BUCTOJ 系统为学生提供了一个便捷的在线评测环境，而外部平台则提供了丰富的题库资源和多样化的比赛形式。竞赛教练通常会安排学生在这些平台上进行解题训练或参加相关比赛，以提高选手的解题能力和实战经验。

3.1.2 学生竞赛能力的量化与评估

为了有效地量化学生的竞赛能力和训练情况，竞赛教练会定期收集学生在各个训练平台上的比赛积分和解题数量。通过分析这些数据，教练可以了解学生的训练进度和存在的问题，从而制定针对性的训练计划。学生在相关平台上的表现直接影响教练的判断，进而决定其是否有资格参加线下正式比赛。

3.1.3 教练工作的繁琐性

对于竞赛教练来说，目前的训练管理方式存在一定的繁琐性。教练需要日常前往各个平台收集学生的训练数据，并在此基础上制定训练计划和监督学生的训练过程。这种方式不仅耗费教练大量的时间和精力，还可能导致数据收集不全面、分析不准确的问题。

3.2 系统功能需求分析

针对当前 ACM 竞赛训练的现状，本次毕业设计旨在构建一个数据管理分析平台，以协助竞赛教练进行教学。该平台能够自动访问相关网站，收集并分析数据，从而为教练提供综合信息。此外，该平台还能够替代教练执行学生提醒任务，并利用历史数据为学生提供竞赛辅助。例如，通过分析学生的历史训练数据，为其推荐适合完成的特定题目，从而提供更高质量的训练。

根据上述目标，我们将系统划分为六个主要需求：系统数据管理、学生数据管理、外部数据收集、数据展示、系统智能提示和智能训练。每个主要需求都被视为一个独立的模块，接下来将对每个主要需求进行详细划分和整理，形成独立的子需求。

3.2.1 系统数据管理

系统数据管理需求，作为系统的关键组成部分，扮演着至关重要的角色，其核心目标在于建立一个健全的基础信息控制功能。此需求的实现对整个系统的运行效率和数据准确性有着直接的影响。在满足此需求的过程中，我们主要考虑了两种角色：系统自身和管理员。系统需要具有自我管理的能力，以维持其正常运行，而管理员则需要具有对系统进行管理的权限，以保证系统的有效运行。相应的功能用例图如图 3-1 所示：

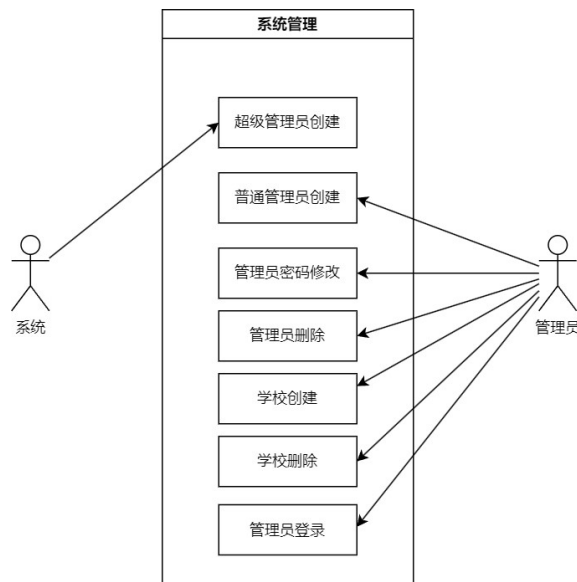


图 3-1 系统管理模块用例图

管理员相关的功能需求包括以下几点：

1. 超级管理员创建：系统需要一个初始管理员登入系统进行后续数据操作。

表 3-1 超级管理员创建用例表

用例名称	超级管理员创建
参与者	系统
前置条件	1. 系统正常启动 2. 数据库连接
后置条件	数据库中置入超级管理员
主要事件流	1. 系统启动 2. 系统连接数据库 3. 系统读取预置账户密码 4. 系统向数据库中创建超级管理员

次要事件流	1.1 系统启动异常 1.2 取消创建，系统关闭 2.1 数据库没有连接 2.2 取消创建，报错
-------	---

2. 普通管理员创建：除超级管理员外，还需要若干普通管理员提供给多个教练登入系统进行管理

表 3-2 普通管理员创建用例表

用例名称	普通管理员创建
参与者	管理员
前置条件	1. 管理员已经登录到系统中
后置条件	新的管理员账户成功创建，并系统更新相应的管理员列表
主要事件流	1. 管理员打开管理员创建页面。 2. 系统显示管理员创建页面，包含必填字段和可选字段。 3. 管理员填写新管理员的详细信息，如用户名、密码、姓名等。 4. 管理员选择创建类型与所属学校。 5. 管理员点击提交按钮。
次要事件流	1.1 用户名已存在 1.2 系统显示错误消息，指示普通管理员选择其他用户名。 1.3 普通管理员修改用户名并重新提交。 2.1 密码不符合要求： 2.2 系统显示错误消息，指示普通管理员输入符合要求的密码。 2.3 普通管理员修改密码并重新提交。

3. 管理员密码修改：防止管理员密码泄露或忘记密码，需要对管理员密码提供修改功能

表 3-3 管理员密码修改用例表

用例名称	管理员密码修改
参与者	管理员
前置条件	管理员已登录到系统
后置条件	管理员的密码成功修改
主要事件流	1. 管理员打开密码修改页面。 2. 管理员输入新密码和确认新密码。 3. 管理员点击提交按钮。 4. 系统验证所填信息是否有效。 5. 如果所填信息有效，系统更新管理员账户的密码为新密码。 6. 系统显示成功消息，提示密码修改成功。
次要事件流	1. 新密码不符合要求 2. 系统显示错误消息，指示管理员输入符合要求的新密码。

4. 管理员删除：当有不使用的管理员账号时，需要提供管理员账号删除功能，及时删除管理员账号，防止账号被恶意使用

表 3-4 管理员删除修改用例表

用例名称	管理员删除
参与者	管理员
前置条件	管理员已登录到系统
后置条件	管理员成功删除账号
主要事件流	<ol style="list-style-type: none"> 1. 管理员打开删除页面或选择要删除的内容。 2. 系统显示删除确认提示框，包含以下信息：要删除的内容的名称或标识与相关警告或提示信息 3. 管理员确认删除操作。
次要事件流	<ol style="list-style-type: none"> 1. 如果管理员取消删除操作： 2. 系统不执行删除操作。 3. 系统返回到原始状态，不做任何更改。

5. 学校创建：需要有学校数据来标识每一个用户与管理员的信息，且该数据必需为管理员创建的数据

表 3-5 学校创建用例表

用例名称	学校创建
参与者	管理员
前置条件	管理员已登录到系统
后置条件	管理员成功创建学校
主要事件流	<ol style="list-style-type: none"> 1. 管理员打开学校创建页面。 2. 管理员填写学校的必填字段：学校名称 3. 管理员点击提交按钮。 4. 系统验证所填信息是否有效。 5. 如果所填信息有效，系统创建新的学校并保存填写的信息。 6. 系统显示成功消息，提示学校创建成功。
次要事件流	<ol style="list-style-type: none"> 1. 学校名称已存在 2. 系统使用错误消息指示管理员选择其他学校名称。

6. 学校删除：当某个学校退出系统时，需要有学校删除功能来清理相关数据

表 3-6 学校删除用例表

用例名称	学校删除
参与者	管理员
前置条件	管理员已登录到系统
后置条件	管理员成功删除学校
主要事件流	<ol style="list-style-type: none"> 1. 管理员选择要删除的学校。 2. 管理员确认删除操作。 3. 系统验证管理员权限。 4. 如果管理员具有足够的权限，系统执行删除操作，将指定的学校从系统中删除。 5. 系统显示删除成功的消息，提示学校删除成功。

次要事件流	1. 管理员取消删除操作 2. 系统不执行删除操作。
-------	-------------------------------

7. 管理员登录：管理员需要可以登录到系统中进行其他数据管理

表 3-7 管理员登录用例表

用例名称	管理员登录
参与者	管理员
前置条件	管理员已经打开系统登录页面
后置条件	管理员成功登录账号
主要事件流	1. 管理员打开系统登录页面。 2. 管理员输入用户名和密码。 3. 管理员点击登录按钮。 4. 系统验证管理员的用户名和密码。 5. 如果验证成功，系统将授予管理员权限，并将管理员登录到系统。 6. 系统显示登录成功的消息，提示管理员登录成功，并跳转到管理员控制台页面。 7. 管理员可以开始执行管理员权限下的操作。
次要事件流	1.1 如果管理员输入的用户名或密码为空 1.2 系统显示错误消息，要求管理员重新输入。 2.1 如果管理员输入的用户名或密码与系统中存储的凭据不匹配 2.2 系统显示错误消息，要求管理员重新输入有效的用户名和密码。

3.2.2 学生数据管理

学生数据管理需求作为系统核心需求的重要组成部分，在整个系统中起到了至关重要的作用，确保了系统的正常运行和整体功能的实现。此需求的主要职能在于，为学生提供便捷的数据登记通道，同时也赋予管理员对数据的管理权限，这两方面的功能构成了整个系统后续操作的基础。

一方面，通过学生数据管理需求，学生能够将他们的训练进度和成果以数据的形式进行记录和跟踪，这对于精确掌握每个学生的训练状态和表现至关重要。另一方面，管理员可以借助该需求对学生的训练数据进行有序管理和有效监控，从而为数据分析和决策支持等后续功能提供准确和及时的数据输入。

在满足该需求的过程中，我们主要考虑了两种角色：管理员和普通用户（学生）。针对这两类用户，我们设计了相应的功能模块，以满足他们各自的特定需求。这些功能的需求用例如图 3-2 所示：

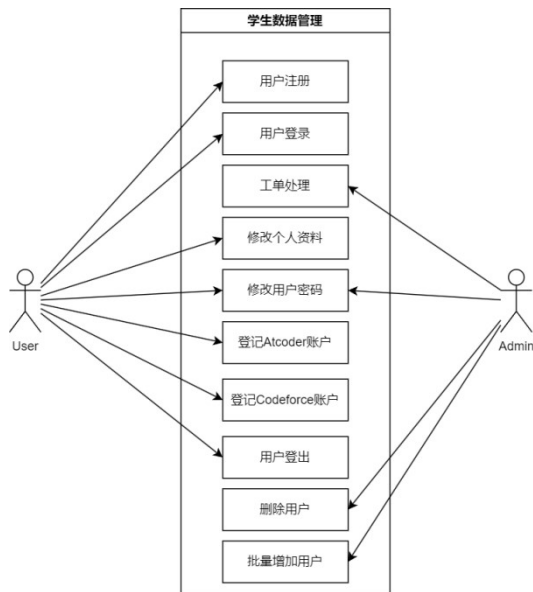


图 3-2 学生数据管理模块用例表

学生管理相关的功能需求包括以下几点：

1. 普通用户注册：普通用户为学生所使用的用户，学生可以在其中管理自己的训练平台账户等信息，为系统爬取竞赛数据提供支持。

表 3-8 普通用户注册用例表

用例名称	用户注册
参与者	用户
前置条件	用户访问注册页面
后置条件	提交工单至工单系统
主要事件流	<div>1. 用户打开系统注册页面。</div> <div>2. 用户填写必填字段：用户名、密码、学校、班级、学号等信息。</div> <div>3. 用户点击提交按钮。</div> <div>4. 系统验证所填信息是否有效。</div> <div>5. 如果所填信息有效，系统创建新的用户账户。</div> <div>6. 系统生成工单并将填写的信息提交到工单表中。</div> <div>7. 系统显示成功消息，提示用户注册成功并等待管理员审核工单。</div>
次要事件流	<div>1.1 如果用户名已存在</div> <div>1.2 系统显示错误消息</div> <div>1.3 指示用户选择其他用户名。</div> <div>2.1 如果密码不符合要求</div> <div>2.2 系统显示错误消息</div> <div>2.3 指示用户输入符合要求的密码。</div>

2. 普通用户登录：普通用户需要为其提供一个登入系统的功能。

表 3-9 普通用户登录用例表

用例名称	用户登录
参与者	用户
前置条件	用户已经打开系统登录页面
后置条件	用户成功登录账号
主要事件流	<ol style="list-style-type: none"> 1. 用户打开系统登录页面。 2. 用户输入用户名和密码。 3. 用户点击登录按钮。 4. 系统验证用户的用户名和密码。 5. 验证成功，系统显示登录成功的消息，提示管理员登录成功，并跳转到首页。 6. 用户可以开始执行操作。
次要事件流	<ol style="list-style-type: none"> 1.1 如果用户输入的用户名或密码为空 1.2 系统显示错误消息，要求用户重新输入。 2.1 如果用户输入的用户名或密码与系统中存储的凭据不匹配 2.2 系统显示错误消息，要求用户重新输入有效的用户名和密码。

3. 工单处理：用户提交的用户注册与对系统数据的修改均需要经过用户员的审核，需要提供一个工单处理需求来审核用户的请求。

表 3-10 工单处理用例表

用例名称	工单处理
参与者	管理员
前置条件	管理员已登录到系统
后置条件	工单状态更新，并根据处理结果进行相应的操作
主要事件流	<ol style="list-style-type: none"> 1. 管理员打开工单列表页面或工单详情页面。 2. 管理员查看待处理的工单列表或选择具体的工单进行处理。 3. 管理员阅读工单详细信息，包括工单内容、优先级、提交时间等。 4. 系统根据工单内容和要求，执行相应的处理操作。 5. 系统保存工单处理结果和备注信息，并更新工单状态。
次要事件流	<ol style="list-style-type: none"> 1. 工单处理异常 2. 系统退出处理 3. 报错通知管理员

4. 修改个人资料：用户在使用过程中可能伴随着个人资料改变的情况，需要提供一个修改个人资料的功能供用户使用。

表 3-11 修改个人资料用例表

用例名称	修改个人资料
参与者	用户
前置条件	用户已登录到系统
后置条件	个人资料成功修改，并系统更新相应的信息
主要事件流	<ol style="list-style-type: none"> 1. 用户打开个人资料页面或点击编辑个人资料按钮。 2. 系统显示用户的当前个人资料信息。 3. 用户修改需要更新的个人资料字段 4. 用户点击保存按钮。 5. 系统验证修改后的个人资料是否有效。 6. 如果个人资料有效，系统更新用户的个人资料信息。 7. 系统显示成功消息，提示个人资料修改成功。
次要事件流	<ol style="list-style-type: none"> 1. 用户输入的数据不符合要求 2. 系统显示错误消息 3. 指示用户修改

5. 修改用户密码：防止用户密码泄露或忘记密码，需要对管理员密码提供修改功能。

表 3-12 用户密码修改用例表

用例名称	用户密码修改
参与者	用户或管理员
前置条件	用户或管理员已登录到系统
后置条件	用户或管理员的密码成功修改
主要事件流	<ol style="list-style-type: none"> 1. 用户或管理员打开用户密码修改页面。 2. 用户或管理员输入新密码和确认新密码。 3. 用户或管理员点击提交按钮。 4. 系统验证所填信息是否有效。 5. 如果所填信息有效，系统更新用户账户的密码为新密码。 6. 系统显示成功消息，提示密码修改成功。
次要事件流	<ol style="list-style-type: none"> 2. 新密码不符合要求 2. 系统显示错误消息，指示用户或管理员输入符合要求的新密码。

6. 登记 Atcoder 账户：系统需要 Atcoder 账户的数据进行进一步的处理，需要用户登记自己的 Atcoder 账户数据。

表 3-13 用户登记 Atcoder 账户

用例名称	用户登记 Atcoder 账户
参与者	用户
前置条件	用户已登录到系统
后置条件	用户或管理员的密码成功修改
主要事件流	<ol style="list-style-type: none"> 1. 用户打开 Atcoder 账户页面。 2. 用户填写 Atcoder 账户字段的内容。 3. 用户点击提交按钮。 4. 系统验证填写的 Atcoder 账户请求是否有效。 5. 如果 Atcoder 账户请求有效，系统生成工单并将 Atcoder 账户请求信息提交至工单系统。 6. 系统显示成功消息，提示 Atcoder 账户请求已提交成功。
次要事件流	<ol style="list-style-type: none"> 1. 如果用户填写的 Atcoder 账户请求信息不完整或格式不正确 2. 系统显示错误消息 3. 指示用户修正相应字段的内容。

7. 登记 Codeforces 账户：系统需要 Codeforces 账户的数据进行进一步的处理，需要用户登记自己的 Codeforces 账户数据。

表 3-14 用户登记 Codeforces 账户

用例名称	用户登记 Codeforces 账户
参与者	用户
前置条件	用户已登录到系统
后置条件	用户或管理员的密码成功修改
主要事件流	<ol style="list-style-type: none"> 7. 用户打开 Codeforces 账户页面。 8. 用户填写 Codeforces 账户字段的内容。 9. 用户点击提交按钮。 10. 系统验证填写的 Codeforces 账户请求是否有效。 11. 如果 Codeforces 账户请求有效，系统生成工单并将 Codeforces 账户请求信息提交至工单系统。 12. 系统显示成功消息，提示 Codeforces 账户请求已提交成功。
次要事件流	<ol style="list-style-type: none"> 4. 如果用户填写的 Codeforces 账户请求信息不完整或格式不正确 5. 系统显示错误消息 6. 指示用户修正相应字段的内容。

8. 用户登出：有时候会遇到一个网页多人使用的情况，为用户增加一个登出界面，便于使用

表 3-15 用户登出用例表

用例名称	用户登出
参与者	用户
前置条件	用户已登录到系统
后置条件	用户成功登出系统
主要事件流	<ol style="list-style-type: none"> 1. 用户点击退出或登出按钮。 2. 系统执行登出操作，清除用户登录状态和相关信息。 3. 系统显示成功消息，提示用户登出成功。 4. 用户被重定向到系统的登录页面或其他适当的页面。
次要事件流	<ol style="list-style-type: none"> 1. 如果用户在登出前有未保存的工作或数据 2. 系统可以显示确认提示 3. 提醒用户保存工作或数据。

9. 删除用户：当用户退出竞赛训练时，为了节省系统资源，需要删除该用户

表 3-16 删除用户用例表

用例名称	删除用户
参与者	管理员
前置条件	管理员已登录到系统
后置条件	用户账户成功删除，用户无法再访问系统资源
主要事件流	<ol style="list-style-type: none"> 1. 管理员打开用户管理页面或用户列表页面。 2. 管理员选择要删除的用户账户。 3. 管理员点击删除按钮或选择删除操作。 4. 系统显示删除确认提示框，包含相关警告信息。 5. 管理员确认删除操作。 6. 系统验证管理员是否具有足够的权限执行删除操作。 7. 如果管理员具有足够的权限，系统执行删除操作，将选定的用户账户从系统中删除。 8. 系统显示成功消息，提示用户账户删除成功。
次要事件流	<ol style="list-style-type: none"> 1. 管理员取消删除操作 2. 系统不执行删除操作。

10. 批量创建用户：管理员可以通过预先收集学生数据，为学生提前创建好账户，等待后续使用。

表 3-17 批量创建用户用例表

用例名称	批量创建用户
参与者	管理员
前置条件	管理员已登录到系统
后置条件	用户账户成功删除，用户无法再访问系统资源
主要事件流	<ol style="list-style-type: none"> 1. 管理员打开批量创建用户页面或选择批量创建用户操作。 2. 管理员准备一个符合预定格式的 CSV 文件，包含用户的相关信息，如用户名、密码、邮箱等。 3. 管理员上传 CSV 文件到系统。 4. 系统验证管理员是否具有足够的权限执行批量创建用户操作，以及 CSV 文件的格式是否符合要求。 5. 如果验证通过，系统解析 CSV 文件，提取用户信息。 6. 系统逐行读取 CSV 文件中的用户信息，并根据每行的数据创建对应的用户账户。 7. 系统在创建用户账户之前，检查用户名是否冲突。 8. 如果用户名冲突，系统跳过该用户并记录冲突信息，继续创建其他用户账户。 9. 系统将成功创建的用户账户信息保存到数据库中。 10. 系统显示成功消息，提示用户账户批量创建成功。
次要事件流	<ol style="list-style-type: none"> 1.1 如果 CSV 文件格式不正确 1.2 系统显示错误消息 1.3 指示管理员修正 CSV 文件的格式。 2.1 如果存在用户名冲突 2.2 系统记录冲突信息并继续创建其他用户账户。

3.2.3 外部数据收集

数据收集是 ACM 竞赛训练数据分析系统的基础环节，对于整体系统的运作和功能的实现具有重要的意义。

首先，数据收集是数据分析的前提。它是信息的来源，保证了系统有充足、准确的原始数据进行处理和分析。无论是训练的过程数据，还是竞赛的结果数据，都需要通过数据收集功能获取并储存，才能供后续的数据处理和分析使用。

其次，数据收集是提供个性化服务的关键。每个用户的训练情况和需求都可能不同，通过收集各种数据，系统可以理解每个用户的特点和需求，从而提供更加精准、个性化的服务，比如定制化的训练建议。

该模块的需求主要侧重于 Atcoder 和 Codeforces 两个网站，它主要收集学生在网站上解题的相关数据，以及网站上的相关比赛信息。

目前收集数据的需求如下表所列：

表 3-18 外部数据需求表

数据平台	数据类型
Codeforces	比赛数据
	题目数据
	学生解题数据
	学生提交数据
	学生提交代码数据
	学生参赛数据
Atcoder	学生解题数据
	学生提交数据
	学生提交代码数据
	学生参赛数据
	比赛数据

3.2.4 数据展示

数据展示作为 ACM 竞赛训练数据分析系统的一个重要需求，对整体系统的功能性和实用性产生了深远影响。首先，它提供了一种方式，使得收集到的数据可以被转化为具有实际意义和操作性的信息。通过数据展示，教练与学生能够更深入地理解和解读数据，获得关于 ACM 竞赛训练的有用见解。

其次，数据展示为用户提供了决策支持。无论是教练在分配训练资源，还是学生在规划学习策略，都可以依赖于展示的数据来进行更为有效的决策，这对于提升竞赛表现具有重要价值。同时，数据展示作为反馈机制，让用户可以直观地看到他们的训练进度和效果，以此来了解他们是否在正确的路径上，或是否需要调整策略。

此外，一个优秀的展示设计，能够显著提升用户体验。通过易于理解和操作的图表、表格和图像，系统的使用将变得更加直观和友好，用户可以更便捷地获取和理解信息。

该部分需求所需要展示的数据在表 3-19 展示数据列表中列出：

表 3-19 展示数据列表

类型	数据名	数据描述
用户数据	用户列表	系统中全部学生用户的情况
	月度积分	系统中全部学生用户的训练情况，根据某个给定规则形成的量化表
	年级学生对比	系统中各年级学生人数的对比
Codeforces 数据	比赛列表	Codeforces 的比赛列表
	学生参赛情况	学生参与 Codeforces 比赛的情况
	学生解题情况	学生在 Codeforces 里解题的情况
	学生提交情况	学生在 Codeforces 比赛里的提交记录
	网站题目列表	Codeforces 网站中的题目
	学生账户竞赛列表	学生 Codeforces 网站中的账户信息
	Codeforces 积分年份变化	各个 Codeforces 账号在各月的积分情况
	Codeforces 积分人数对比	各个用户 Codeforces 账号在各年的积分人数对比
	Codeforces 解题对比	用户 Codeforces 解题类型对比
Atcoder 数据	比赛列表	Atcoder 的比赛列表
	学生参赛情况	学生参与 Atcoder 比赛的情况
	学生解题情况	学生在 Atcoder 里解题的情况
	学生提交情况	学生在 Atcoder 比赛里的提交记录
	学生账户竞赛列表	学生 Atcoder 网站中的账户信息
	Atcoder 积分年份变化	各个 Atcoder 账号在各月的积分情况
用户分组数据	分组 Codeforces 积分对比	分组内用户的 Codeforces 积分对比情况
	分组 Atcoder 积分对比	分组内用户的 Atcoder 积分对比情况

	分组 Codeforces 提交情况对比	分组内用户的 Codeforces 提交情况对比
	分组 Atcoder 提交情况对比	分组内用户的 Atcoder 提交情况对比
	分组 Codeforces 解题类型对比	分组内用户的 Codeforces 解题类型对比

3.2.5 系统智能提示

系统智能提示需求作为系统核心功能的一部分，它充分利用已收集的数据，并通过调用外部接口的方式，向其他平台推送信息，实现对训练的智能提醒。这一需求的实现对整个系统的效能和用户体验起到了重要作用。

系统智能提示需求的实现可以提高用户的使用便利性和满意度。通过将训练数据推送到用户常用的其他平台，用户可以及时了解自己的训练状态和成果，以及选择接下来应该进行的训练。

目前系统智能提示的需求有以下两种：

1. 比赛预告智能提醒

系统应能够根据收集的数据，智能提醒队员即将到来的 Atcoder, Codeforces 比赛。包括但不限于比赛日期、时间等信息。

2. 比赛结果智能提醒

系统应能够根据收集的数据，智能提醒队员已经完成的 Atcode 或, Codeforces 比赛结果。包括但不限于比赛后积分变化等信息。

3.2.6 智能训练

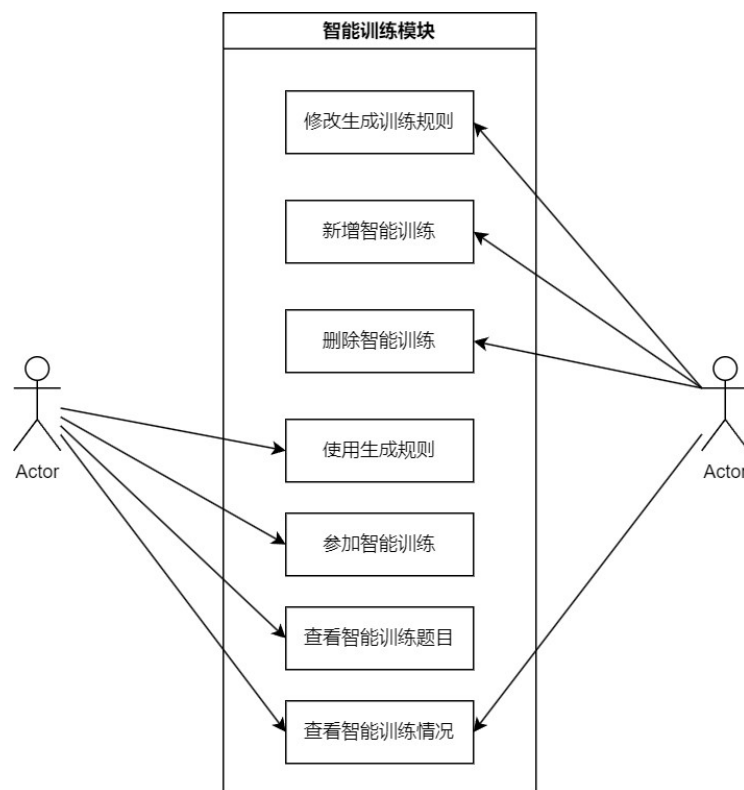
智能训练是 ACM 竞赛训练数据分析系统的一项扩展功能。这一模块充分挖掘和利用用户的历史解题数据，通过分析用户的解题行为，识别出他们的技能水平、解题习惯以及潜在的短板。在此基础上，系统将结合一定的算法规则，为用户提供高度个性化的题目推荐，从而实现对用户进行精准、针对性的训练。

智能训练功能的主要目标是帮助用户更有效地提高他们的编程和问题解决能力。通过为用户推荐和生成符合他们当前能力水平和进步路径的训练题目，系统可以大大提高训练的效率和效果，同时也能够激发用户的学习兴趣和动力，使他们在竞赛

训练过程中获得更好的体验和成果。

智能训练模块的用例图如图 3-3 所示：

图 3-3 智能训练模块用例图



智能训练的相关需求有以下几个：

1. 修改生成训练规则

管理员可以控制一场训练中相关题目的生成规则，用例描述如下：

表 3-20 修改生成训练规则用例表

用例名称	修改生成训练规则
参与者	管理员
前置条件	管理员已登录到系统
后置条件	训练规则被成功修改并保存到系统
主要事件流	1. 用户选择需要修改的训练规则 2. 系统显示选定的训练规则详细信息 3. 用户修改规则的具体参数并提交修改 4. 系统验证修改是否合法 5. 如果修改合法，系统保存修改并返回成功修改的消息
次要事件流	1.1 在步骤 4 中，如果修改无效或非法 1.2 系统显示错误消息，用户需要重新修改规则 2.1 如果规则不存在， 2.2 系统显示错误消息，强制回到规则列表

2. 新增智能训练

管理员可以新增一场智能训练供学生参加，用例描述如下：

表 3-21 新增智能训练用例表

用例名称	新增智能训练
参与者	管理员
前置条件	管理员已登录到系统
后置条件	新的智能训练计划已经成功创建并保存到系统
主要事件流	<ol style="list-style-type: none"> 1. 用户点击“创建新的智能训练计划”按钮 2. 系统显示新的智能训练计划创建界面 3. 管理员输入新的智能训练计划的相关信息 4. 管理员点击“保存”按钮提交新的智能训练计划 5. 系统验证新的智能训练计划的信息的有效性 6. 如果信息有效，系统保存新的智能训练计划并返回成功创建的消息
次要事件流	<ol style="list-style-type: none"> 1. 在步骤 5 中，如果信息无效 2. 系统显示错误消息 3. 管理员需要重新输入有效的智能训练计划信息

3. 删除智能训练

管理员可以删除已有的智能训练，用例描述如下：

表 3-21 删除智能训练用例表

用例名称	删除智能训练
参与者	管理员
前置条件	管理员已登录到系统
后置条件	指定的智能训练计划已经从系统中删除
主要事件流	<ol style="list-style-type: none"> 1. 管理员在系统中找到需要删除的智能训练计划 2. 管理员点击“删除”按钮 3. 系统询问管理员是否确定删除 4. 管理员确认删除 5. 系统删除指定的智能训练计划并返回成功删除的消息
次要事件流	<ol style="list-style-type: none"> 1. 在步骤 3 中，用户选择取消删除操作 2. 系统返回到智能训练计划列表界面

4. 使用生成规则

用户可以主动调用生成规则，生成智能训练任务

表 3-21 使用生成规则用例表

用例名称	删除智能训练
参与者	用户
前置条件	用户已登录到系统
后置条件	用户获得生成的题目编号
主要事件流	<ol style="list-style-type: none"> 1. 用户在系统中找到需要使用的生成规则 2. 用户点击“使用规则”按钮 3. 系统根据所选规则生成训练任务 4. 系统保存生成的训练任务并返回成功消息
次要事件流	<ol style="list-style-type: none"> 1. 在步骤 3 中，用户选择取消生成操作 2. 系统返回到智能训练计划列表界面

5. 参加智能训练

用户可以参加智能训练，在该场智能训练中生成相应题目

表 3-21 5 参加智能训练用例表

用例名称	参加智能训练
参与者	用户
前置条件	用户已登录系统 存在至少一个可用的智能训练计划
后置条件	用户已成功加入选定的智能训练计划
主要事件流	1. 用户在系统中浏览可用的智能训练计划 2. 用户选择一个智能训练计划并点击“参加训练”按钮 3. 系统根据该场智能训练的规则生成训练任务 4. 系统保存生成的训练任务并返回成功消息
次要事件流	1. 在步骤 3 中，系统生成计划失败 2. 系统报错，并返回到智能训练计划列表界面

6. 查看智能训练题目

用户可以查看参加的智能训练题目

表 3-21 查看智能训练题目用例表

用例名称	查看智能训练题目
参与者	用户
前置条件	用户已登录系统 用户已参加对应的智能训练计划
后置条件	系统显示该用户在该场智能训练的题目
主要事件流	1. 用户在系统中浏览他参加的智能训练计划 2. 用户选择一个智能训练计划并点击“查看题目”按钮 3. 系统显示选定的智能训练计划的所有题目列表
次要事件流	无

7. 查看智能训练情况

用户或管理员可以查看某场已经开始的智能训练

表 3-21 查看智能训练情况用例表

用例名称	查看智能训练情况
参与者	用户或管理员
前置条件	该场智能训练已经开始
后置条件	系统显示该场智能训练情况
主要事件流	1. 用户在系统中浏览可用的智能训练计划 2. 用户选择一个智能训练计划并点击“查看训练情况”按钮 3. 系统显示选定的智能训练计划的详细训练情况
次要事件流	无

3.3 非功能需求分析

3.3.1 性能需求

为了确保系统具有高效的功能和用户体验，以下量化标准将用于衡量系统的响

应速度性能：

响应时间：系统应在用户发出请求后的特定时间范围内做出响应。目标是使常见请求的响应时间小于 1 秒。

并发处理能力：系统应支持同时处理多个请求，量化标准为能够处理 100 个以上的并发请求。

系统稳定性：系统应能够在长时间运行的情况下保持稳定的性能。要求系统连续运行 24 小时以上而无需重启或出现严重的性能下降。

系统资源利用率：系统应合理利用硬件资源，如 CPU、内存和磁盘。目标是将 CPU 利用率保持在 70% 以下。

故障恢复时间：在系统出现故障时，系统应能够快速恢复正常运行。要求系统在 10 分钟内从故障中恢复。

以上量化标准将作为评估系统性能的指标，以确保系统具有高效的响应速度。在设计和开发过程中，将对系统进行测试和优化，以确保量化标准满足项目的实际需求，并为用户提供快速响应的用户体验。

3.3.2 自动运维需求

系统的自动运维需求是确保系统能够以可靠和高效的方式进行运行、监控和维护。以下是本系统的自动运维需求：

自动化部署和配置：系统应具备自动化部署和配置的能力，以简化系统的部署和设置过程。自动化工具和脚本可以用来快速部署系统，并自动完成必要的配置步骤，从而减少人工操作和人为错误的风险。

日志和监控：系统应能够自动记录关键操作和事件的日志，并提供监控和警报机制。通过自动化的日志记录和监控系统，可以及时发现系统的异常行为和潜在问题，并快速采取相应的措施进行调整和修复。

自动化备份和恢复：系统应具备自动化备份和恢复的功能，以保护数据和系统的完整性。自动备份可以定期创建数据和配置的备份副本，并将其存储在可靠的位置。在系统出现故障或数据丢失时，自动恢复机制可以使用备份数据进行快速恢复。

3.4 本章小结

本章节首先深入分析了 ACM 竞赛的训练流程，基于此流程，识别并定义了六大主要功能需求，每个主需求下进一步细化为多个子需求。为满足用户交互需求，我们提供了用例图和用例表。在处理数据爬取和展示的部分，我们定义了相应的数据类型。最后，我们阐述了系统的非功能性需求，明确了对系统性能和自动化运维的需求。

第 4 章 系统设计

4.1 系统结构设计

4.1.1 系统总体架构设计

ACM 竞赛训练数据分析系统的总体设计以模块化的思路进行，系统被划分为三个主要的功能模块，分别是数据获取模块、数据展示模块和数据管理模块。每个模块都有特定的功能和责任，但同时也需要与其他模块共享和交换信息，以实现整个系统的协同运作。

数据获取模块的主要职责是从各类来源收集用户的解题数据，包括但不限于解题时间、解题策略、题目难度等信息。这些数据是系统进行后续分析的基础，因此其质量和完整性至关重要。

数据展示模块则负责将收集和分析的数据以直观的形式展示给用户，如数据图表、进度报告等。通过用户友好的界面和可视化工具，用户可以更好地理解他们的训练情况和进步路径，从而更有针对性地进行训练。

数据管理模块则是面向管理员，提供系统及用户数据的管理，包括用户数据的存储、查询、更新和删除，智能训练系统的控制，智能提示系统的设置等操作。

三个模块间的数据统一使用 MySQL 进行管理，确保了数据的一致性和可靠性。各模块间在功能上相对独立，模块间交互通过访问和操作数据库中的数据完成。

三个模块间的架构如图 4-1 所示

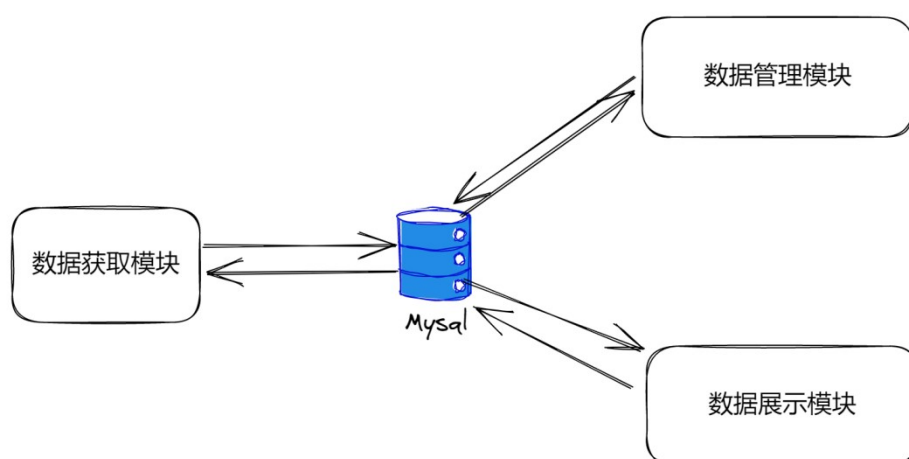


图 4-1 整体架构图

4.1.2 系统流程设计

ACM 竞赛训练数据分析系统三个模块的主要工作流程涉及学生用户、系统管理员以及系统之间的相互交互和协同工作，学生用户主要使用的模块是数据展示模块，系统管理员主要使用的模块是数据管理模块。此外，数据获取模块是一个后台模块，它独立运行并且不提供图形化的操作。

首先，学生在数据展示模块中进行注册。在注册过程中，学生需要提供基本的个人资料，例如姓名、学号、联系方式等，以及他们在各竞赛网站上的账户信息。这些信息将用于系统识别并追踪学生的解题数据。

注册信息提交后，系统管理员在数据管理模块中对学生的注册信息进行审查。管理员需要核实学生的身份以及所提供信息的准确性。审核通过后，学生的账户将被激活，从而可以开始利用系统提供的各种功能。

一旦学生账户被激活，数据获取模块将根据学生在注册时提供的竞赛网站账户信息，从相应网站获取学生的解题数据。这些数据，包括解题记录、解题用时、题目难度等，将被系统自动收集并存入数据库。

当数据成功存储到数据库后，系统管理员和学生用户都可以在数据展示模块查看这些数据。数据展示模块将以图表、报告等形式展示数据，从而帮助用户更深入地理解和分析自己的训练状况。

另外，系统还具备自动通知功能。在某些指定事件，例如新的比赛发布或比赛结果公布时，数据管理模块将根据数据库中的数据，自动向学生发送通知。这将帮助学生及时获取比赛的最新信息，提升他们的参与度和竞争力。

最终，系统还提供智能训练功能。管理员可以通过数据管理模块管理智能训练相关信息，如设定训练题目，调整生成算法等。学生则可以在数据展示模块参与智能训练，通过系统提供的个性化题目推荐进行针对性的训练。

以上流程设计旨在使系统流程分配，模块角色明确，提高系统的使用效率和用户体验，同时降低系统的管理和维护成本。

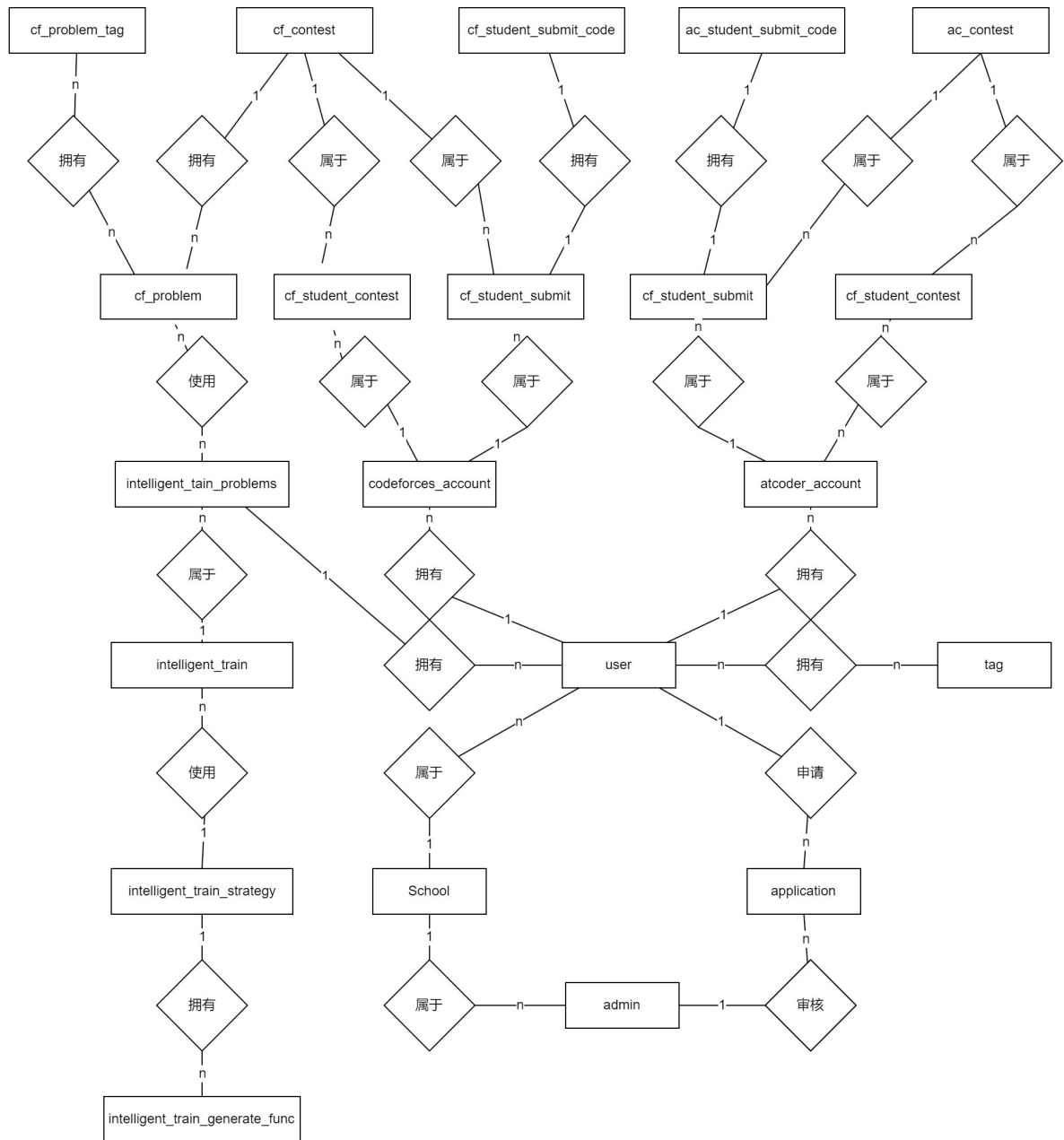
4.2 数据库设计

4.2.1 数据库 E-R 模型设计

本小节对系统的数据库进行设计，分析实体之间的关系，使用 E-R 图对具体的逻辑结构进行说明。由于系统的功能众多，根据系统的主要需求可以得出系统包含：

学校、管理员、Codeforces 比赛、Atcoder 比赛、学生参赛记录、智能训练记录等多个实体对象，在 E-R 图中标注了实体间的关系，比如一个学校拥有多个用户，它们之间的联系是一对多，每个 Codeforces 问题有多个标签，多个标签又同时被多个问题拥有，它们之间的联系是多对多。系统数据库的 E-R 图如图 4-2 所示，实体的属性过多，为了防止 E-R 图过于复杂混乱，图中仅展示实体之间的联系，没有标出实体的属性。

图 4-2 数据库 E-R 图



4.2.2 数据表设计

系统根据功能需求设计了多张数据库表，本节主要以模块划分来展示部分核心功能的数据表。主要展示系统及用户模块的数据表，Codeforces 模块的数据表，以及智能训练的数据表，由于 Atcoder 数据相关表与 Codeforces 模块数据相关表类型，故此节仅提供 Codeforces 部分的数据表的描述。

1. 系统及用户模块

(1) 管理员用户表

表 4-1 管理员用户表

字段名	字段类型	字段含义	是否必须
id	bigint	主键 ID	是
username	varchar(255)	用户名	是
password	varchar(255)	用户密码	是
school	bigint	学校主键 ID:外键字段	是
isSuper	tinyint	是否为超级管理员	是

(2) 普通用户表

表 4-2 普通用户表

字段名	字段类型	字段含义	是否必须
id	bigint	主键 ID	是
username	varchar(255)	用户名	是
password	varchar(255)	用户密码	是
school	bigint	学校主键 ID:外键字段	是
classname	varchar(255)	班级	否
stuNO	varchar(255)	学号	是
year	int	年级	否
status	tinyint	账户状态	是

(3) 工单表

表 4-3 工单表

字段名	字段类型	字段含义	是否必须
id	bigint	主键 ID	是
uid	bigint	用户 ID	是
time	bigint	申请时间	是
operation	string	申请操作	是
parameter	json	参数	是
status	tinyint	状态	是
aid	bigint	操作人 ID	是
operationTime	bigint	操作时间	是

2. Codeforces 模块

(1) Codeforces 比赛表

表 4-4 Codeforces 比赛表

字段名	字段类型	字段含义	是否必须
cid	bigint	主键 ID:codeforces 比赛 ID	是
name	varchar(255)	比赛名称	是
type	varchar(255)	比赛类型	否
level	int	比赛等级	否
startTimeStamp	bigint	开始时间戳	是
endTimeStamp	bigint	结束时间戳	是
duration	bigint	持续时间	是
isNormal	tinyint	状态是否正常	是

(2) Codeforces 用户比赛表

表 4-5 Codeforces 比赛用户表

字段名	字段类型	字段含义	是否必须
id	bigint	条目 ID	是
cid	bigint	主键 ID:codeforces 比赛 ID	是
cfid	bigint	账户 ID	是
rank	int	当场比赛排名	是
solve	int	解题数	是
diff	int	积分变化情况	是
rating	int	比赛后积分	是

(3) Codeforces 提交情况表

表 4-6 Codeforces 提交情况表

字段名	字段类型	字段含义	是否必须
sid	bigint	条目 ID	是
cid	bigint	主键 ID:codeforces 比赛 ID	是
uid	bigint	账户 ID	是
index	varchar(255)	比赛 index	是
ctime	bigint	提交时间	是
status	varchar(255)	提交状态	是
isAfter	bigint	是否为补题	是
language	varchar(255)	使用语言	是

3. 智能训练模块

(1) 智能训练表

表 4-7 智能训练表

字段名	字段类型	字段含义	是否必须
id	bigint	主键 ID: 智能训练比赛 ID	是
name	varchar(255)	比赛名称	是

字段名	字段类型	字段含义	是否必须
strategy_id	int	题目生成策略 ID	是
start_time_stamp	bigint	开始时间戳	是
end_time_stamp	bigint	结束时间戳	是

(2) 智能训练生成策略表

表 4-8 智能训练生成策略表

字段名	字段类型	字段含义	是否必须
id	int	主键 ID: 智能训练生成策略 ID	是
strategy_name	varchar(255)	生成策略名称	是
use_function	json	该生成策略中的生成函数配比，以 jsonArray 形式存储	是

(3) 智能训练比赛题目表

表 4-9 智能训练比赛题目表

字段名	字段类型	字段含义	是否必须
uid	bigint	主键 ID: 智能训练生成策略 ID	是
tid	int	智能训练 ID	是
problems	json	该智能训练中用户的题目，以 jsonArray 形式存储	是

(4) 智能训练生成函数表

表 4-10 智能训练生成函数表

字段名	字段类型	字段含义	是否必须
id	bigint	主键 ID: 智能训练生成策略 ID	是
name	varchar(255)	生成函数名称	是
description	varchar(255)	该生成函数描述	是

4.3 数据获取模块

4.3.1 数据获取模块架构设计

数据获取模块的核心需求是采集数据，其实质为一个网络爬虫系统。当用户给定适当的输入时，系统将前往各网站收集数据，然后将数据存入数据库。在传统的实现方式中，任务发布、数据获取以及数据持久化处理通常在同一程序中完成。然而，这种模式有其固有的缺点，例如性能低下，链条中断可能导致整个系统出错等。因此，本设计在这部分考虑实现分布式爬虫系统。

为了充分利用分布式集群的性能，我们将数据管理模块划分为三个子模块，每个子模块负责不同的任务。爬虫终端作为接收爬取指令的服务器，负责从互联网获取相关数据。爬虫接收端负责对已获取的数据进行持久化处理，并将其存储到数据库。爬虫调度端根据定时任务或主动触发，向爬虫终端派发任务。这种架构的优势在于可以最大限度地利用多台服务器的性能。实际上，我们可以将爬虫终端部署在不同的网络环境中，只需确保三个模块都能稳定地访问 redis 服务器，即可保证爬虫流程的正常运行。

数据获取模块架构图如图 4-3 所示：

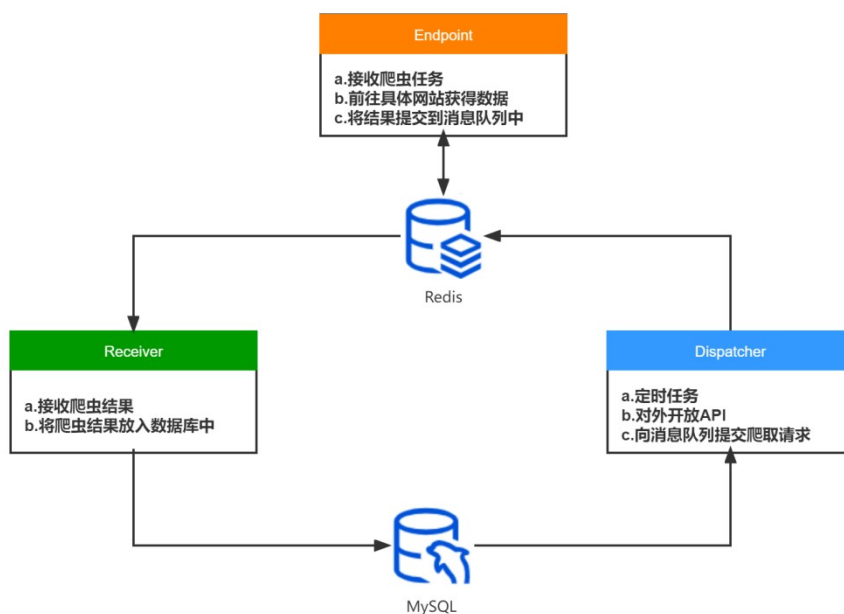


图 4-3 数据获取模块架构图

4.3.2 爬虫终端设计

爬虫终端的主要功能包括接收爬虫任务、前往特定网站获取数据，以及将结果

提交到消息队列。具体流程如下：爬虫终端内部设有一个监听器，其任务是监听 redis 中的爬取请求。一旦接收到爬取请求，监听器会激活 URL 生成器，将请求中的数据转化为一个具体的爬取任务，然后将此任务添加到爬虫队列。爬虫会持续从爬虫队列中获取任务，前往特定的网站进行数据爬取。爬取完成后，爬虫处理数据并调用结果上传器，由结果上传器决定后续的数据处理流程。

爬虫终端流程大致如图 4-4:



图 4-4 爬虫终端流程图

在爬虫终端的设计中，我们选择了 SpringBoot 作为基础框架，并将各个爬虫组件作为 Bean 嵌入到基于 SpringBoot 的系统中。其整体架构借鉴了已有的 Java 爬虫框架 WebMagic 的设计思路。

爬虫终端架构图如图 4-5 所示:

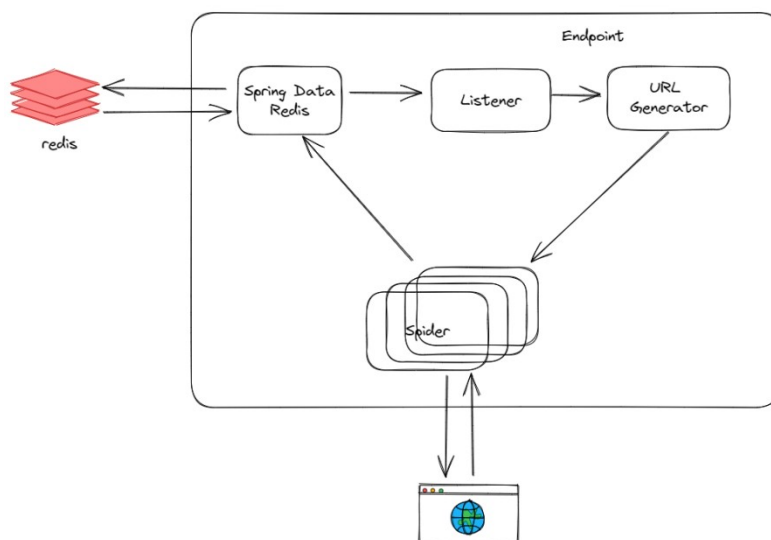


图 4-5 爬虫终端架构图

在爬虫终端的设计中，每一个 Spider 都被视为一个独立的 Java Bean，它拥有一个主工作线程和多个爬取线程。程序启动后，这些 Bean 将随之实例化，并由 SpringBoot 进行管理。Spider 内部主要由四个组件构成：Processor、Scheduler、Pipeline 和 Downloader。

其中，Downloader 负责页面请求以及获取数据，Processor 负责对页面数据进行处理，Scheduler 负责缓存并分发任务到对应的 Spider，而 Pipeline 负责处理爬取的结果。

爬虫内部结构图如图 4-6:

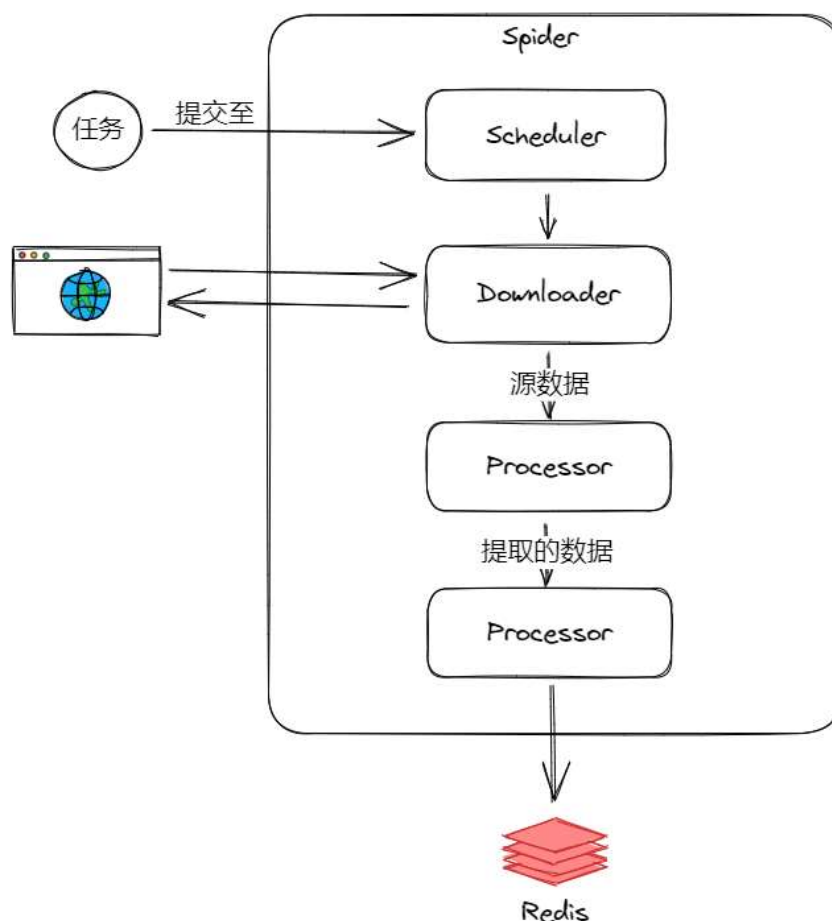


图 4-6 爬虫内部结构图

4.3.3 爬虫调度端架构设计

爬虫调度端主要负责生成爬虫请求，这些请求的产生主要通过两种方式：主动触发和定时任务，统称为请求生成。主动触发通常在用户或管理员需要立即获取特定数据时发起，而定时任务则是预先设定在特定时间自动执行的任务，这两种方式保证了爬虫请求的灵活性和及时性。

当调度端需要生成请求时，它会查询数据库中的相关数据，这些数据可能包括需要爬取的网站、页面 URL、数据类型等信息。调度端根据这些信息组装并生成一个具体的爬虫请求。

生成的爬虫请求会被放入 redis 中，这是一个高性能的内存数据存储系统，可以用作数据库、缓存和消息代理。将爬虫请求存放在 redis 中，可以实现快速的数据交换，并且能够有效地处理大量并发请求。

一旦爬虫请求被存入 redis，爬虫终端就会获取这些请求并开始执行爬虫任务。这种设计方式使得爬虫任务的发布和执行可以在不同的系统组件中完成，提高了系统的模块化程度，也提升了整个系统的效率和稳定性。

具体流程如图 4-7:

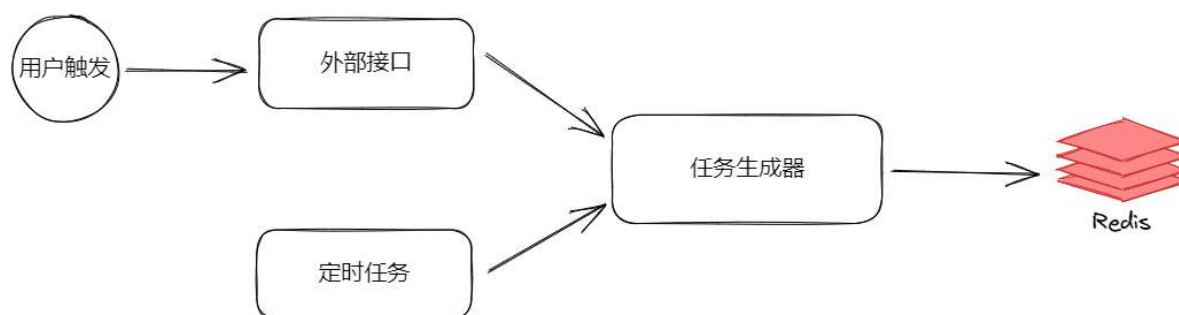


图 4-7 爬虫调度端设计

4.3.4 爬虫接收端架构设计

爬虫接收端的主要职责是接收爬虫终端处理后的数据，并进行持久化处理，即将数据保存到数据库中。当爬虫终端完成数据的爬取和初步处理后，这些结果会被发送到爬虫接收端。爬虫接收端接收到数据后，会进行必要的进一步处理（如数据清洗、格式转换等），然后将处理后的数据存入数据库，以保证数据的持久化。这种设计可以有效地将数据爬取和数据处理两个环节分开，提高了系统的稳定性和易维护性。

具体流程如图 4-8:



图 4-8 爬虫接收端流程图

4.4 业务端(数据管理模块与数据展示模块)设计

业务端是对数据管理模块和数据展示模块的总体概括，它们均作为系统内的 Web 子服务存在。这两个模块的基础架构设计具有相似性，都是基于 SpringBoot 和 Vue 进行开发。它们的主要区别在于服务的对象不同：数据管理模块主要服务于管理员，而数据展示模块主要服务于普通用户。

在需求分析的过程中，我们确定了许多功能点，这些功能点常常需要在这两个模块中都得到实现。因此，本节的设计将以功能模块为分节进行，而不是以架构模块进行，这样可以更清晰地展示每个功能模块在数据管理和数据展示两个模块中的具体设计，有助于更全面、深入地理解系统的工作原理和使用方式。

4.4.1 基础架构设计

在业务端部分，其顶层设计遵循经典的前后端分离架构，每个系统都拥有自身的前端和后端。后端与数据库相连接，提供 API 接口供前端调用；前端则利用 Ajax 技术调用这些后端接口，从而获取数据并展示给用户。

其大致结构如图 4-9：

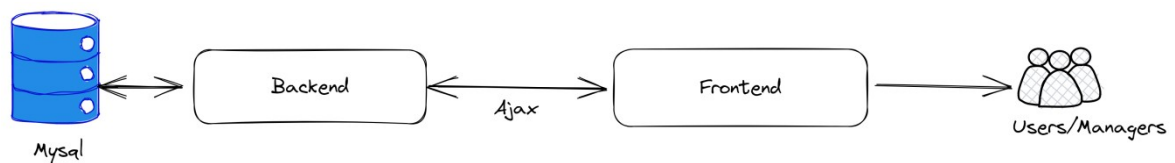


图 4-9 业务端顶层架构图

接下来我们对顶层架构图进行细化，在初步设计中，后端基于 SpringBoot 进行开发，内部可以被细化为六个主要层级。按照从用户到数据库的顺序，从上到下分别为：

1. 用户界面层：这是用户直接交互的界面，包括各种用户操作的响应以及数据展示。
2. 前端框架层：该层使用 Vue.js 等前端框架构建，负责处理用户输入，以及根据后端数据动态更新界面。
3. 数据交互层：负责前后端数据的传输，通常使用 Ajax 技术，将用户的请求发送到后端，并接收后端返回的数据。
4. 后端接口层：负责处理前端发送的请求，调用相应的业务逻辑，然后将结果返回给前端。

5. 后端业务层：这里包含了核心的业务逻辑处理，如数据验证、数据处理、业务规则的实现等。

6. 数据库层：负责数据的持久化存储，通过 SQL 语句与数据库进行交互，实现数据的查询、插入、更新和删除。

这种分层架构的设计，有助于分清楚各个功能模块的职责，使得系统更加结构清晰、代码易于管理和维护，同时也便于进行模块化和组件化的开发。

具体架构如图 4-10 所示

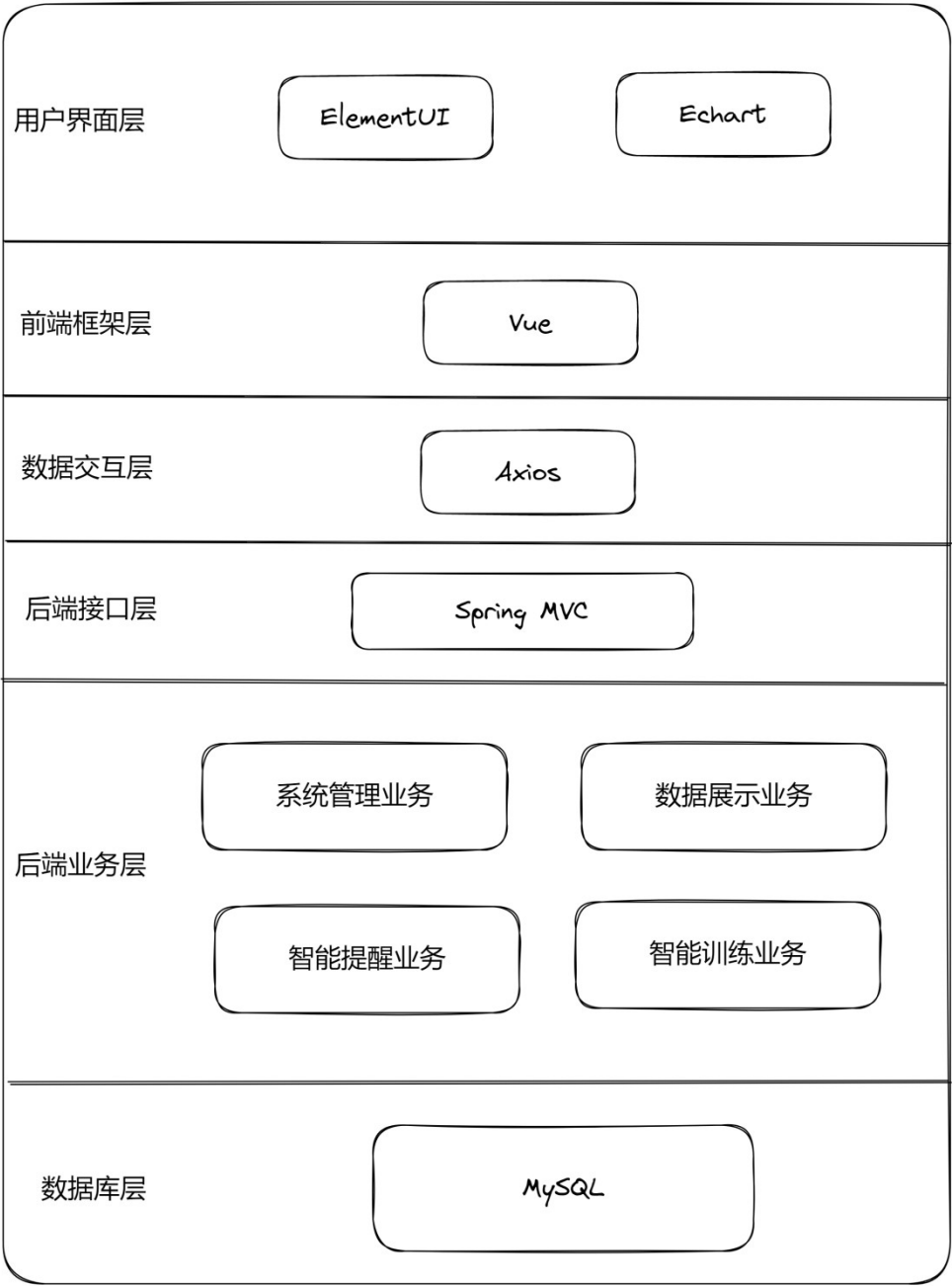


图 4-10 业务端分层架构图

4.4.2 系统管理功能设计

根据需求分析，系统管理功能模块功能主要实现在数据管理模块中，包含超级管理员创建、普通管员创建，普通管理员创建，普通管理员密码修改，管理员删除、学校创建、学校删除、管理员登录等六个功能，它们在数据管理模块中的设计如下：

（1）超级管理员创建

超级管理员创建不存在界面，它是系统内置的在系统开启时会自动调用的一个功能，在系统启动并且连接到数据库后，它会自动检测系统中是否存在一个超级管理超级管理员账户，若不存在，它会根据预留的数据，在数据库中创建一个超级管理员账户

（2）管理员创建

管理员创建功能存在于管理员列表页面中，点击相关管理员创建的按钮会弹出一个管理员创建的表单，管理员填写该表单后会创建一个新的管理员账户

（3）管理员密码修改

管理员密码修改功能存在于管理员列表页面中，点击相关管理员密码修改的按钮会弹出修改密码的表单，管理员可以在其中对该管理员的账号进行修改

（4）管理员删除

管理员删除功能存在于管理员账户列表中，点击按钮会弹出确认提示框，确认后即可删除管理员账户。

（5）学校创建

学校创建功能存在于学校列表页面中，点击学校创建的按钮会弹出一个学校创建的表单，管理员填写该表单后会创建一个新的学校记录

（6）学校删除

学校删除功能存在于管理员账户中，点击按钮会弹出确认提示框，确认后即可删除学校记录。

（7）管理员登录

管理员登录是一个单独存在的页面，提供用户名及密码的输入框，管理员尝试进入系统时会跳转到该页面，需要登录后才能执行后续操作。

4.4.3 学生数据管理功能模块设计

学生数据管理功能模块横跨数据管理模块与数据展示模块。以两结构模块划分，其功能图如图 4-11：

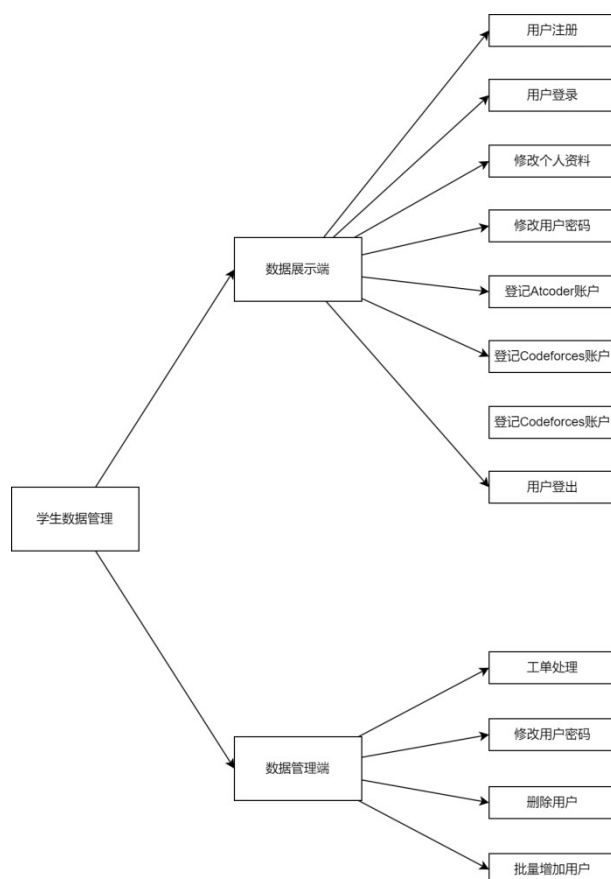


图 4-11 学生数据管理功能图

4.4.4 系统智能提示功能设计

系统的智能提示功能作为定时任务存在于数据管理模块，该功能根据预设的时间点触发相应的函数，执行特定的提示任务。目前，需要进行提示的数据主要分为两类：比赛预告和比赛结果，根据这两类数据的特性，设计了不同的处理流程。

对于比赛预告数据，系统会主动访问数据库，检索未来七天内尚未进行的比赛信息，然后将这些信息构造为一条通知消息。随后，系统会调用外部平台的 API 接口，将这条通知消息推送给用户。

对于比赛结果数据，系统会持续监听现有的比赛消息，一旦发现新的比赛结果信息，系统会自动收集这些信息，并在下一次预设的推送时间点到来时，将这些信息构造为一条通知消息。同样，系统会调用外部 API 接口，将这条通知消息推送给用户。

该模块流程如图 4-12 所示

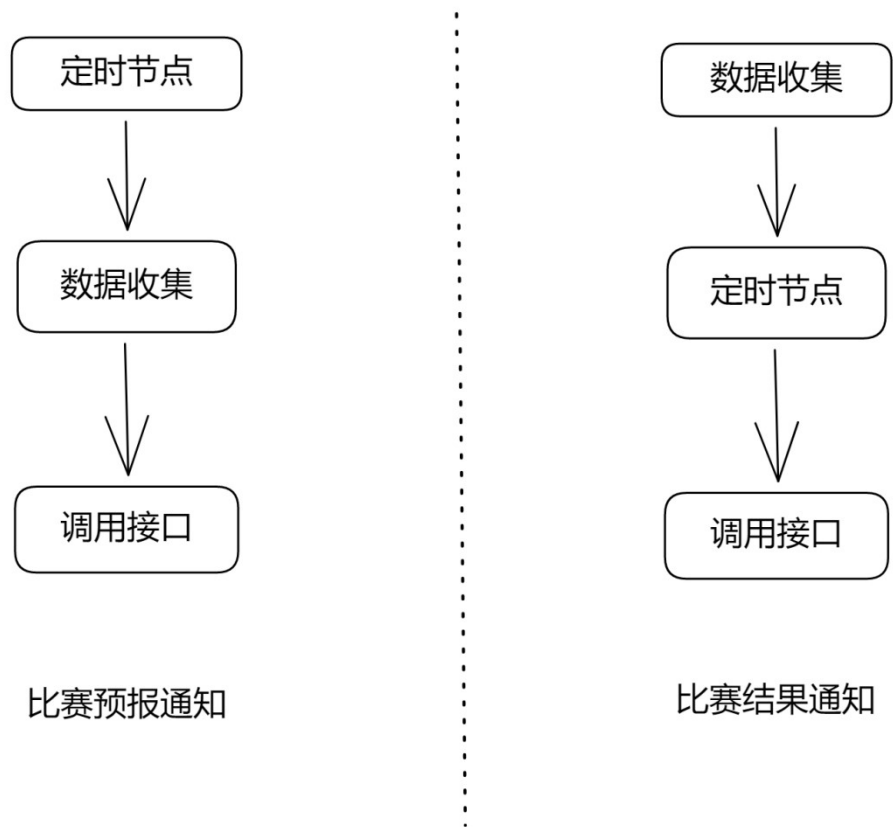


图 4-12 系统智能提示功能流程图

4.4.5 数据展示功能设计

数据展示功能主要是负责在系统中展示数据，它被实现在数据展示端中，它主要使用两种形式展示数据：表格和可视化图表。

表格：这种形式通常用于显示大量详细的数据。例如，我们可以通过表格形式展示学生的比赛成绩、完成题目的时间、使用的策略等信息。表格形式的展示使得用户可以轻松地查看和比较数据。

可视化图表：这种形式通常用于显示数据的总体趋势或模式。例如，我们可以通过折线图展示学生的成绩随时间的变化情况，或者通过饼图展示各个比赛题目类型的分布情况。图表形式的展示使得用户可以直观地理解数据，并快速获取重要信息。

4.4.6 智能训练功能设计

智能训练是系统的附加功能，它依赖于现有的数据，为学生和教练提供一些训练辅助。然而，面向学生和面向教练的需求并不完全相同。为了提高这部分功能的易用性和可扩展性，我们决定采用接口分离的设计模式，将业务流程和算法实现分开。

具体来说，我们将题目推荐设计成一类函数调用，这些函数都实现了同一接口。这个接口只需要接收一个用户 ID 作为输入，然后通过这个用户 ID 获取所有其他需要的数据，最后返回一系列推荐的题目。

这种设计的基本架构如图 4-13 所示。这样的设计旨在提高系统的灵活性和可扩展性，使得我们可以更容易地根据不同的需求调整和优化智能训练功能。

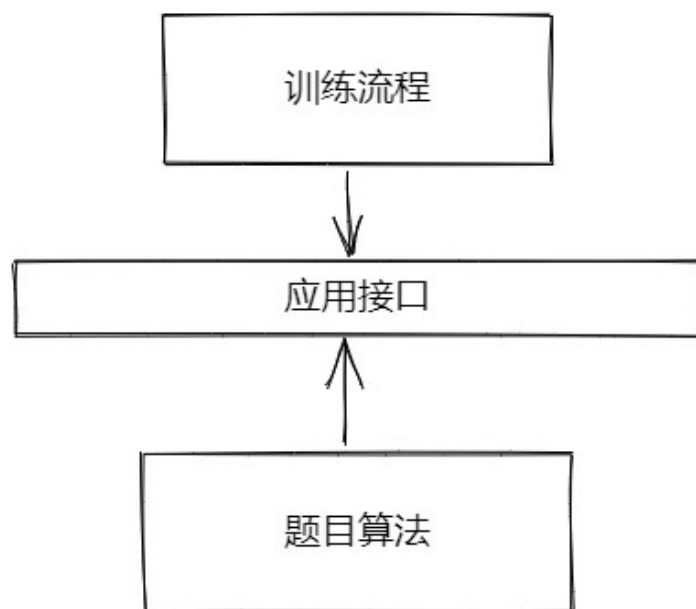


图 4-13 智能训练基本架构

对于教练的需求，我们将其视为独立的训练场次，每个场次都可以设定其生成策略。每个生成策略由若干个生成函数组成，这样，教练可以复用生成策略，简化创建训练的流程。复用已有的策略，避免了每次都需要配置生成函数比例的麻烦。这样，教练可以根据每个训练场次的具体需求，灵活地设定和调整生成策略，以实现个性化训练。

对于学生的需求，我们提供直接生成推荐题目的功能。学生可以直接调用生成函数，按照指定的题目数量生成题目，而无需在训练场次中使用该系统。这样，学

生可以根据自己的学习进度和需求，随时获取推荐的题目，进行自主学习和练习。

以上设计架构图如图 4-14 所示，这种设计不仅可以满足教练和学生的不同需求，还可以提高系统的灵活性和易用性，使得教练和学生都能更有效地利用系统进行训练和学习。

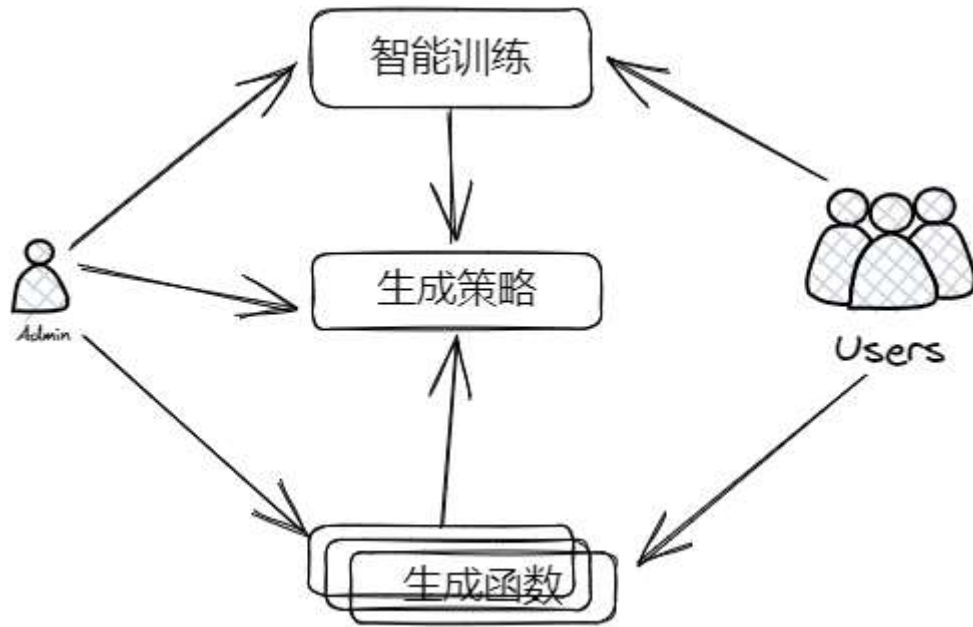


图 4-14 系统具体架构图

第 5 章 系统实现与测试

5.1 数据获取模块实现

5.1.1 数据终端实现

在爬虫终端的实现中，我们严格按照第四章的设计进行。在这一部分的实现中，我们将 URL 生成器实现为一系列的 Handler 类，这些类共同继承自 IHandler 类，负责生成 URL。此外，我们还有一个 NormalListener 类，负责调用 Handler 类，并将任务提交给 Spider。每个 Spider 都具有四个组件：Scheduler、Downloader、Processor 和 Pipelines。以下是这四个组件的详细实现：

1. Scheduler：我们基于 Java 的 BlockingQueue 实现了一个可以支持多线程且无冲突的 NormalSchedule，其队列大小为 10000，每个队列可以供多个工作线程读写。

2. Downloader：我们基于 SpringBoot 的 RestTemplate 实现了一个基本的下载器：RestTemplateDownloader。这个下载器可以接收固定的爬虫任务，并根据任务的参数（如 URL、HTTPMethod、header 等）访问网络以获取数据。

3. Processor：每个爬虫都会根据它们所爬取的网站实现具体的解析器。虽然每个爬虫的解析器都不尽相同，但大体上可以分为两种类型：JSON 解析器（负责解析 JSON 格式的请求结果）和 HTML 解析器（负责解析 HTML 格式的请求结果）。解析完成后的结果会被放入请求任务的结果中，以供下一步处理。

4. Pipeline：每个爬虫针对其结果可以有多种处理方式，如输出到控制台或上传到消息队列中。Pipeline 负责确定数据的去向。目前我们实现的是 RedisSuccessPipeline 和 LogSuccessPipeline，它们分别将数据提交到 Redis 和控制台中。

除了以上四个组件，爬虫内部还维护一个调度线程和若干个工作线程。调度线程负责从 Scheduler 中取出任务，并启动一个工作线程来执行该任务。工作线程使用的是 SpringBoot 的线程池，其中的任务规则是一套固定的规则。工作线程在获取任务后，会按照 Downloader -> Processor -> Pipelines 的流程调用组件，完成该次爬取。

数据终端的具体时序图如图 5-1 所示：

5.1.2 爬虫调度端实现

在爬虫调度端的实现上，我们严格遵循了第四章的设计。这个端主要包含四种关键的类：Dispatcher、Entity、Service 和 Interface。以下是这些类的实现介绍：

1. Dispatcher: Dispatcher 类负责组装爬取请求并将其提交到 Redis 中。在这个类中，我们实现了一个抽象基类 AbstractDispatcher。AbstractDispatcher 有一个具体的函数 dispatch，在这个抽象基类的函数中，我们实现了将请求发送到 Redis 的流程。当我们需要实现具体的 Dispatcher 时，只需要继承这个类，然后重写相关的数据，指定使用的爬虫和处理器。在具体使用时，调用这个类的 dispatch 函数，即可完成提交爬虫数据的过程。

2. Entity: Entity 是 SpringBoot JPA 必要的组件，一个 Entity 类对应数据库中的一个表。

3. Service: Service 是 Interface 的具体业务支持，它通过从数据库中取出数据，并调用 Dispatcher 向 Redis 中提交请求。

4. Interface: Interface 提供了对外的 HTTP 触发接口。外部用户可以通过 HTTP 请求调用这个类中的函数来发送爬取请求。

爬虫调度端的时序图如图 5-2 所示

5.1.3 爬虫接收端实现

爬虫接收端的实现可以被视为爬虫调度端的一个逆过程：它主要负责从 Redis 中取出数据并将其存储在 MySQL 数据库中。爬虫接收端同样包含四个关键的类：Listener、Handler、Service 和 Entity。以下是这些组件的详细实现：

Listener: Listener 主要负责监听 Redis 中的消息。当 Redis 中有结果消息时，Listener 会将其取出并调用 Handler 进行下一步处理。

Handler: Handler 负责处理具体的 Redis 消息。根据消息的内容，Handler 会调用对应的一个或多个 Service 进行处理。

Service: Service 负责具体结果的处理。每个 Service 负责处理一类特定的数据，接收数据并调用 SpringBoot JPA 的接口将结果存入数据库中。

Entity: Entity 是 SpringBoot JPA 必要的组件，一个 Entity 类对应数据库中的一个表。

5.2 数据管理模块实现

5.2.1 超级管理员创建


在超级管理员创建的实现中，我们使用了 SpringBoot 的 `@PostConstruct` 注解。具有此注解的方法会在 SpringBoot 创建对象时自动运行，从而实现在系统启动时运行的效果。

为了确保在数据库连接建立之后才运行该函数，我们可以让这个类依赖于 Spring JPA 相关组件。这样，我们可以确保在数据库连接成功后才执行该方法。

在该方法运行时，系统会读取 `application.properties` 中的相关配置，并检查预设的账户名在数据库中是否已存在。如果不存在，则会向 MySQL 数据库中创建一个新的记录。

如图 5-3 所示是超级管理员创建的时序图：

5.2.2 管理员登录



The image shows a simple login interface. It consists of a light gray rounded rectangle containing two text input fields. The first field is labeled 'username' and the second is labeled 'password'. Below these fields is a button with the text '登录' (Login) in Chinese.

图 5-4 数据管理模块登录界面

管理员登录功能是数据管理模块的第一个用户界面，它要求用户首先登录才能进行任何相关操作。登录后，普通管理员可以进行除系统管理以外的所有操作，而超级管理员可以进行所有操作。

用户会从前端发起登录请求，后端接收到请求后，会去数据库中验证账户和密码。这个过程涉及到几个关键的类，如图 5-5 管理员登录相关类类图所示：

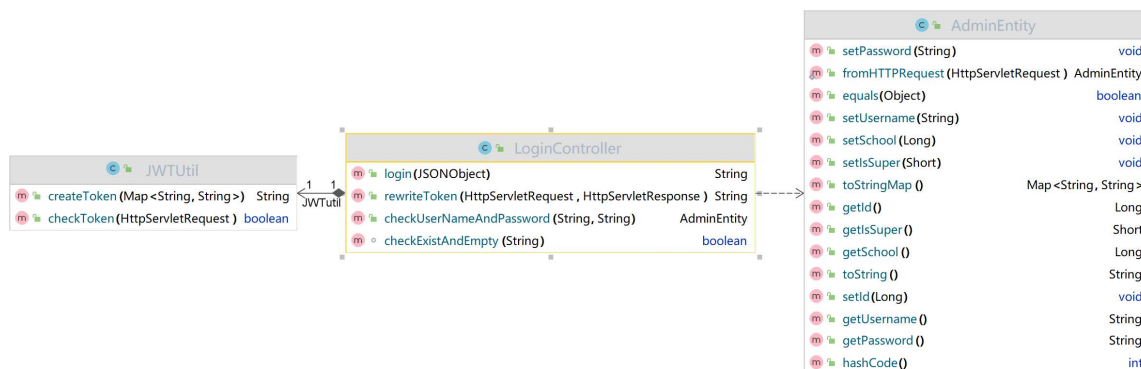


图 5-5 管理员登录相关类类图

管理员登录时会使用到 JWT (JSON Web Token) 相关功能，用以存储登录状态，前端会将其保存在浏览器的 Storage 中，再之后的请求中会附着到 HTTP Header 的 Token 字段中供后端验证。

实现整个登录逻辑的函数是 LoginController 类中的 login 函数，HTTP 请求到来时会调用该函数，该函数会完成数据库查询、JWT 形成两个工作，最后返回结果到前端。

5.2.3 管理员管理

id	用户名	权限	新增管理员
13	jhas	超级管理员	修改密码 删除
14	manager	超级管理员	修改密码 删除
15	liuyong	北京化工大学	修改密码 删除
16	zhouxiaolin	北京化工大学	修改密码 删除
17	liuxiangrui	北京化工大学	修改密码 删除
18	root	超级管理员	修改密码 删除

图 5-6 管理员管理页面

如图 5-6 所示，管理员管理为一个列表页面，展示各个管理员信息，点击修改密码按钮可以修改该管理员密码，点击删除按钮可以删除该管理员，在表头有新增管理员按钮，填写相关信息后可新增管理员账户。

管理员管理后端实现中涉及到一个主要的 Controller 类，类中实现了四种函数，对应四种接口：

getManager: 负责获取全部管理员数据

addManager：负责新增管理员
deleteManager：负责删除管理员
modifyPassword：负责修改管理员
具体如图 5-7 管理员管理相关类图所示

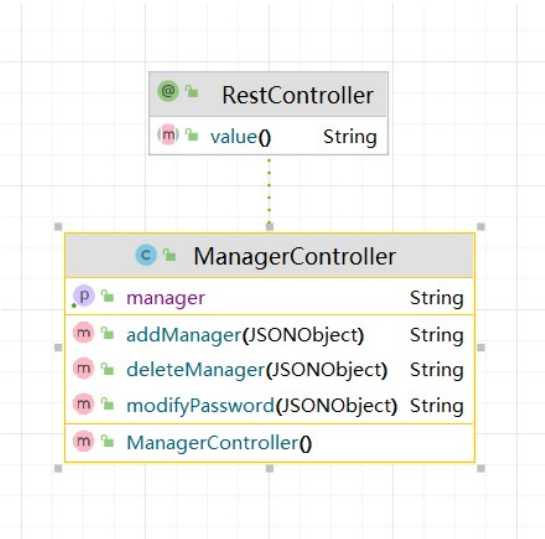


图 5-7 管理员管理相关类图

5. 2. 4 学校管理

Id	Name	
1	北京化工大学	<div>增加</div> <div>删除</div>
2	南通大学	<div>删除</div>
3	云南大学	<div>删除</div>

图 5-8 学校管理页面

如图 5-8 所示，学校管理为一个列表页面，展示学校管理员信息，点击增加按钮并在弹出的表单中填写相关信息后即可新增学校信息。左侧对应记录的删除按钮点击后可以删除该学校，并且会导致各个依赖于该学校的管理员与学生也会被级联删除。

学校管理后端实现中同样涉及到一个主要的 Controller 类，类中实现了四种函数，对应

- 5. 2. 5 学生管理
- 5. 2. 6 智能训练管理
- 5. 2. 7 定时任务管理

5.3 数据展示模块实现

5.3.1 首页

5.3.2 学生登录

5.3.3 学生注册

5.3.4 学生列表展示

5.3.5 训练数据展示

5.3.6 可视化图表展示

5.3.7 智能训练

5.4 系统性能测试

5.5 系统兼容性测试

第 6 章 总结与展望

6.1 总结

6.2 展望