文章编号：2095-2163(2022)03-0080-07 中图分类号：TP311.1 文献标志码：A

基于前后端分离算法的 ACM 智能管家系统

高云泽，王莉莉，董文睿，冯紫君，胡祖容，赵中楠
（哈尔滨理工大学 计算机科学与技术学院，哈尔滨 150080）

摘 要：为了 ACM 国际大学生程序设计竞赛参赛者对算法进行系统学习并积累编程能力，本文基于前后端分离技术，设计了一个功能全面，架构稳定的智能辅助学习系统。通过该系统，参赛者可以对算法训练题目进行分析，模拟比赛，并进行赛后复盘和总结。系统后端采用 Django 框架，前端采用 Electron 和 Vue 框架，保证前后端的可扩展性，降低了系统的耦合性，并采用 WebSocket 完成用户之间的实时通信。系统提供了训练题库，借助文本分词来获取题目的类型标签，并使用该标签类型对题目进行划分，并指导用户数据的可视化结果。

关键词：文本分词；前后端分离；Django；Electron；Vue

ACM intelligent supporting system based on front and back end separation algorithm

GAO Yunze, WANG Lili, DONG Wenrui, FENG Zijun, HU Zurong, ZHAO Zhongnan
(School of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080, China)

【Abstract】 For the participants of ACM International College Students´ programming competition to systematically learn the algorithm and improve programming ability, an intelligent supporting learning system is designed with comprehensive functions and stable architecture based on the frontend and backend separation technology. Through this system, the participants can analyze the training algorithms, simulate the contests, repeat and summarize after the contests. Django framework is adopted at the back end of the system, and Electron and Vue framework are adopted at the front end to ensure the scalability and reduce the coupling of the system. Websocket is designed for the real-time communication between users. The system provides a training problem bank. The problem labels are obtained according to the text word segmentation, and the problems are divided into different types based on the question labels, which plays a guidance role on the visualization of user data.

【Key words】 text segmentation; front end and back end separation; Django; Electron; Vue

0 引 言

国际大学生程序设计竞赛(ACM/ICPC)是由国际计算机协会(ACM)主办的比赛，其宗旨是展示大学生的创新能力、团队精神，以及在压力环境下通过编写程序进行分析和解决问题的能力。

目前，在国内 ACM 赛事的训练过程中发现一些急需解决的问题。根据 1995～2018 年中国大陆高校入围 ACM-ICPC 全球总决赛统计数据可知，大部分高素质人才集中在名校，入围高校中只有 5 所是非 985/211。其次，有些高校的投入力度不够，很多学校 ACM 活动仅由社团、协会或团委以课余活动的形式开展，在资金、师资、设备、场地等方面受到一定的限制，影响了发展规模。还有一些学校起步比较晚，经验不足，只能通过去友校参观学习、摸着石头过河等方法去积累经验，需要很多拓荒者。在此过程中，会遇到许多困难，比如训练内容繁多复杂，短期内看不到成绩、感觉不到自己的进步等等，慢慢会感到枯燥、迷茫，甚至放弃。造成这些问题的另一个重要原因，是缺少一个好的学习辅助平台，以及系统的学术交流平台来提供指导和帮助。对于参加 ACM 竞赛的学生来说，训练过程中迫切地需要相应的科学指导；对于学校的 ACM 组织来说，也需要一个高效的团队管理系统。因此，开发一个集能力分析、训练指导、团队管理和学术交流平台于一体的智能管家系统尤为必要。

本文针对此问题研发了 ACM 智能管家系统，该系统通过爬虫技术获取题目信息，通过机器学习中的分词技术，归纳出题目的标签，建立智慧题库，让用户更方便地选择适合自己能力的题目；通过爬虫

基金项目：黑龙江省大学生创新创业训练计划项目(202010214103)；黑龙江省教育科学“十四五”规划课题(GJB1421061)资助。

作者简介：高云泽(2000-)，男，本科生，主要研究方向：计算机应用；王莉莉(1980-)，女，博士，教授，主要研究方向：探测与成像、计算机应用。
通讯作者：王莉莉 wanglili@hrbust.edu.cn

收稿日期：2021-11-15

技术抓取用户历史刷题记录，并将可视化展示给用户，让用户更好地在系统中感受到自己的成长。该系统还具有好友通信、复盘总结等功能来辅助用户更好地提升 ACM 竞赛能力。

1　系统分析

1.1　需求分析

系统用户共有两类，分别为普通用户和管理员用户。管理员爬取国内外 OJ 系统的题目信息，再通过关键词搜索，结合分词技术补全题目相应的算法标签，从而构建一个大而全的智慧题库，使用户可以更好地筛选题目进行专项练习；同时通过用户绑定的信息，可以获取用户在各大做题平台上的做题记录，然后再对所有做题记录进行聚合分析，并最终整合为图表展现给用户，从而使用户的成长记录可视化。

用户可发布团队或者个人定时训练赛，训练赛题目可由用户选择，或者规定范围由系统进行推送[1]。在用户训练过程中，系统会记录用户提交的记录以及浏览记录，并将这些信息转化为图表展现给用户，方便用户进行复盘总结。

日志主要用于满足用户即想即记的需求，并且采用多级分类结构，可以让日志的管理更为便捷。在通信方面，通过 WebSocket 建立长连接，满足用户间实时通信的需要。

系统用例如图 1 所示。

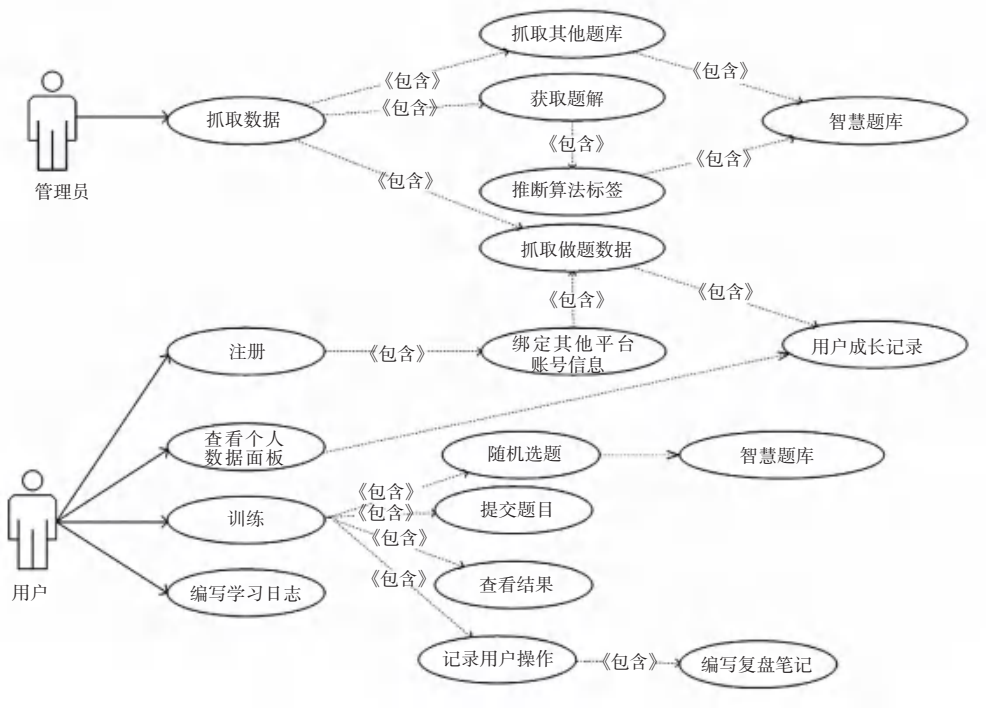

图 1　系统用例图

Fig. 1　Use case diagram of system

1.2　系统框架

该平台基于 B/S 的开发模式，利用前后端分离技术，实现系统的高内聚低耦合，减少后端服务器的并发/负载压力。前端使用 Electron 和 Vue 框架搭建，基于组件化思想来提高各个组件复用，降低耦合度[2]。后端使用 Django 框架搭建，利用各种数据库访问插件，大大提高了数据库访问的效率[3-4]。用户操作用户界面向后端发送对应事件，后端收到事件，通过方法查询对应数据并整理后送给前端，前端收到结果集后，将数据整理并渲染展示给用户。系统结构如图 2 所示。

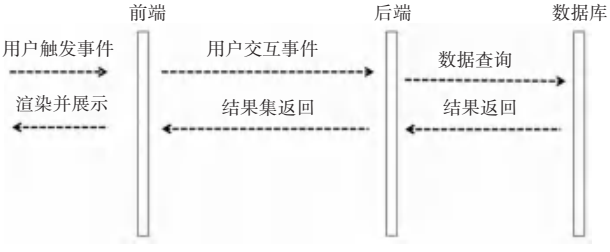

图 2　系统结构图

Fig. 2　Structure of system

1.3　系统基本策略

该系统研究的主要问题如下：

(1) 如何在抓取题解后使用合适的文本分词来

划分题目标签,并提高题目标签的准确性;

(2)系统需要随时执行抓取题目,获取用户做题信息,处理题目标签等任务,系统如何在保证高效的情况下执行这一系列操作,无论是用户信息或者题库信息都会存在变动,系统如何实时并同步更新这些信息;

(3)如何保证用户间实时通信和消息推送。

根据以上分析,本系统的基本策略如下:

(1)对目前的算法标签种类进行整理,获取一个全集计数表。对于一份题解进行分词处理后,提取所有题目标签,在全集表中对相应的题目标签进行计数。在查询获取一定题目的标签后,统计全集表中计数最大的题目标签,并以此标签作为该题目的标签;

(2)利用 Redis 作为消息队列来处理耗时的异步操作。主服务器向 Redis 中的对应任务队列分发任务;后台服务器通过队列的优先级别,由高到低不断获取队列中的任务并在主服务器中运行。同时主服务器可以向 Redis 中下发定时任务,后台服务器按照消息队列的任务优先级来调度处理。在处理完单个任务之后,按需将结果返回给主服务器。任务处理机制如图 3 所示;

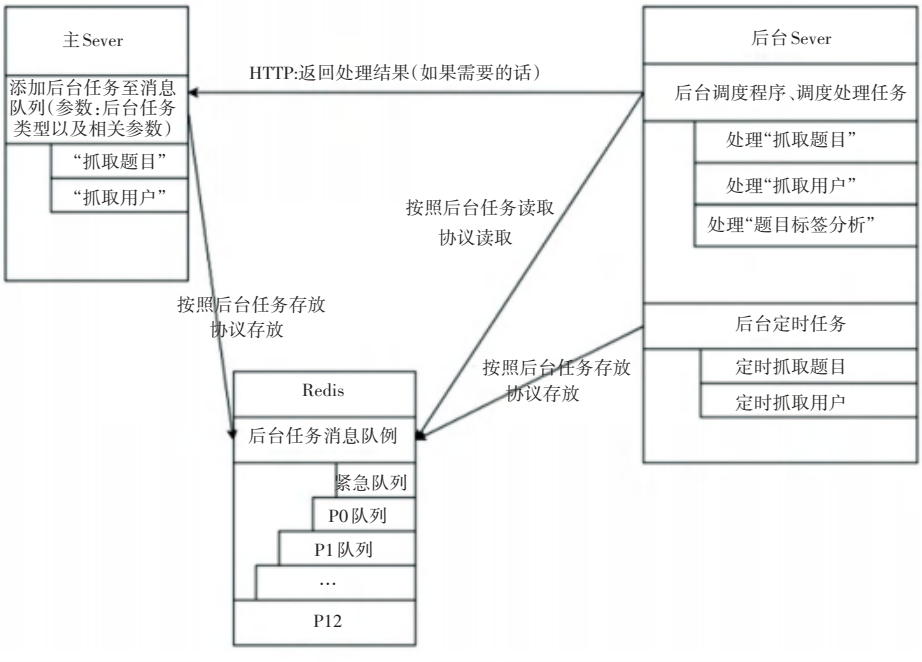

图 3 任务处理机制

Fig. 3 Task processing mechanism diagram

(3)对比常用实时通信方法,本文选择 WebSocket 协议。用户成功登录后,与后端服务器建立双向通信通道。当用户间进行实时通信时(如聊天、发信息等),系统会将消息即时推送给同时在线的用户,或者将消息缓存到数据库中,直到对方下次登录时再进行推送;此外服务器也可以在特定时机,主动将消息推送给用户。

1.4 系统数据组成

本系统数据主要由 3 个部分组成:

(1)用户在系统内的行为数据。其中包括用户的基础信息、团队信息、训练记录以及相关日志信息;

(2)用户在其他训练平台的历史做题记录,主要用于将用户的成长过程进行可视化展现;

(3)系统题库数据是系统的核心数据,这部分数据主要通过爬虫的方式,去获取主流做题训练平台的题库信息,再通过关键词搜索加分词技术补全题目相应的算法标签。

2 系统主要模块设计

2.1 智慧题库模块

智慧题库模块由爬虫获取的信息实现,爬虫能力占据该模块的主体。模块中所需要爬取的信息可分为两类:一类为题目信息,另一类为用户做题记录信息。系统将这两类信息的爬取任务交给后台主线程执行,主线程将任务分发到不同的任务栈中,对任务栈初始化后,由子进程来执行任务栈中的任务。在子进程处理任务期间,若遇到异常则及时抛出,当所有任务栈中的任务执行完毕后,后台主线程结束。爬虫框架如图 4 所示。

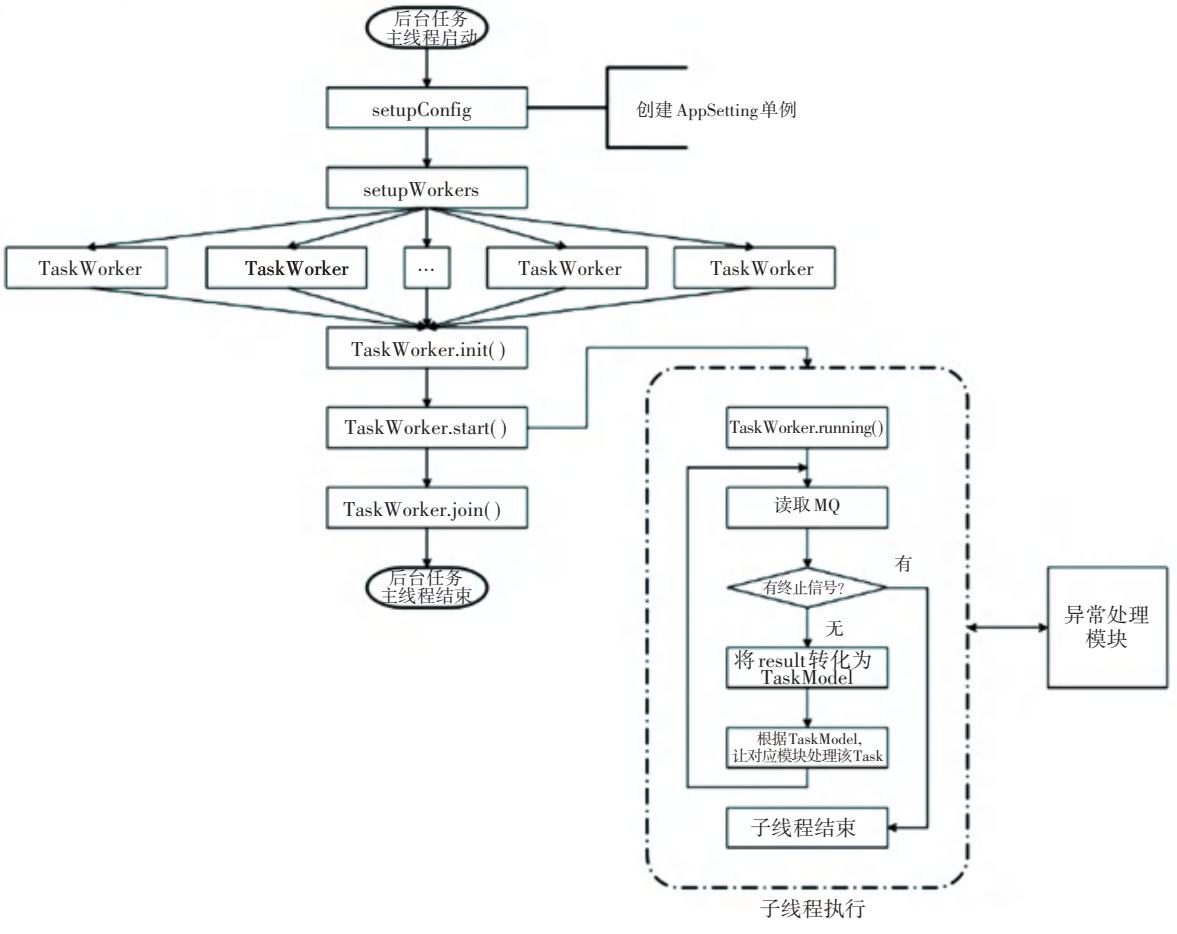

图 4　爬虫框架

Fig. 4　Spider framework

2.1.1　系统题库搭建

本系统主要获取国内外最主要的三大实体 OJ 题库信息，分别为 CodeForces、HDU，以及 POJ。题库中，除 Codeforces 可以直接通过爬虫抓取题目标签以外，其余题库题目均为从题目标签层面来对题目进行划分。因此，系统题库的实现首先必须满足以下条件：

(1)获取题库原始数据；

(2)对于原始题库中缺乏的数据进行补全。

为了搭建系统题库，首先通过爬虫来获取各个题库的题目信息，存入题库后再由后台分发对应的任务存入到 Redis 服务器中。当后台服务器处理到任务时，通过爬虫抓取对应题库的题目信息。对于已有标签的题目则直接存入系统题库，对于没有标签的题目再进行第二次爬虫。第二次爬虫基于题目信息来爬取题目的题解，在 html 页面获取题解内容后，通过文本分词划分高频标签，并进行计数统计，以计数最高的标签作为该题目的标签并存入题库[5]。题目处理流程如图 5 所示。

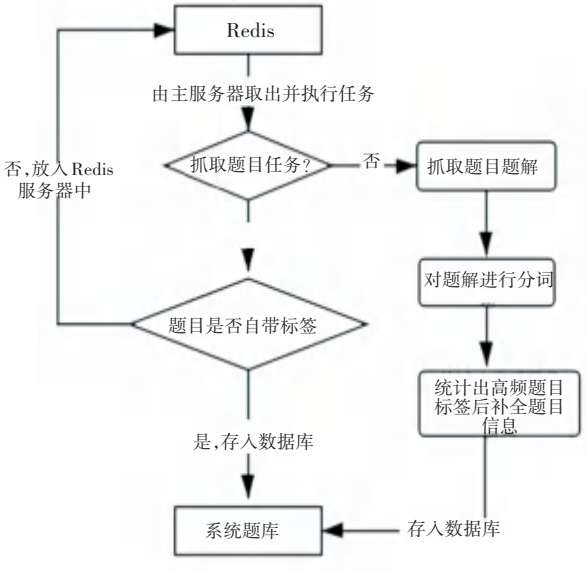

图 5　题目处理流程

Fig. 5　Flow chart of problem handling

2.1.2　用户成长记录模块

支持用户成长记录模块的数据，主要来自题库

内获取到的用户做题信息及题目信息。当用户使用该系统时,可以选择希望绑定的 OJ。系统在绑定 OJ 后,会对该用户的做题记录进行爬取,将爬取的题目映射到系统题库内的题目,并对这些题目进行时间、题库以及题目标签上的划分,使用 ECharts 来处理这些数据并整合为图表展示给用户,让用户对自己做题的数量、类型和周期有一个直观的了解。

(1)用户信息抓取:当用户登录管家系统后,系统会向用户申请绑定 OJ 账号,当账号绑定成功后,系统将利用爬虫模拟用户登录并获取用户提交过的题目 id。这个过程具有一定的时间间隔,该任务会被放到定时任务中,在特定的时刻对任务进行爬取并执行。在完成模拟登录以及获取题目 id 后,系统将此部分的题目 id 与系统题库内的 id 进行映射对比,获取此部分的题目信息集合,将该部分内容的时间、题目来源和题目标签等信息进行抽离,并形成图表所必须的信息集合;

(2)用户数据可视化:用户可视化需要使用到用户信息集合,将用户信息集合按照做题时间、题目来源、题目标签进行划分后,在 ECharts 的模型层,将对应的 JSON 数据注入到对应的图表信息中,并在视图层对图表样式进行配置。之后,在控制层对事件展示进行控制,最后将用户的做题信息以图表的形式展现给用户。

2.1.3 用户个性化题库模块

用户个性化题库模块负责将后端采集到的题目信息展现给用户。其中包括用户是否通过、题目名称、算法标签以及该题目的通过数,并提供筛选功能。用户可以通过输入题目进行模糊匹配,选择特定平台的题库或者根据特定的算法标签进行筛选。

鉴于题库中题量较大,且需要满足用户的精准查询,系统将需精准搜索的平台题库和标签做了一层缓存,将用户每次的查询都放入缓存里,从而提升用户使用的流畅度[6]。

2.2 训练与复盘模块

训练赛模块支持个人训练以及团队训练,用户可在本地完成代码,通过管家系统的接口进行提交。通过训练模块中的任意题目,用户均可打开提交界面,该界面会接受用户传输的源代码并将代码打包发送到对应 OJ 平台进行提交。

用户在选题时,可以手动输入题库平台和题目 ID 来添加题目,也可以选择一些过滤条件,通过随机方式自动添加一些所有参与者均没有通过的题目。

用户在结束训练赛后,可以进入当前比赛的“复盘模块”。当用户第一次进入这个模块时,系统自动给用户生成一个复盘文档的模板,用户可以回顾整个比赛的过程,并写下自己的收获。

2.3 日志模块

在日志模块中,用户可创建多级日志用于赛后复盘总结,每页日志最多可存储 2 Mb 的数据。当用户完成日志的构建后,后台会校验日志数据合格性,通过校验后存入数据库,同时将用户日志的缓存更新,保持数据一致性。日志编辑框支持 markdown 语法,用户可在日志中添加文字、图片、链接等一系列内容来完成赛后总结或刷题记录,并将日志以二进制文件的方式存入数据库中。在用户进入系统时由子进程查询日志目录结构,并将这些内容放入缓存中。当用户点击对应日志时,系统进行解析,交给前端渲染,同时在缓存中添加此内容。

(1)数据结构:日志模块中创建多层级的日志,创建的日志目录会直观的展示给用户,但是为了保持数据的稳定性以及查询效率,整体目录最多向下延伸 3 个层级。对于任意日志均存在两种指针:一种指向其父节点,用于子日志后回溯父日志;另一种指向子日志,用于快速展示目录。用户目录会在第一次加载后进行缓存,之后用户刷新界面都使用缓存中获取的目录,直到更新目录后台时再次获取新的日志目录,并更新用户界面,同时将新的用户目录传入缓存;

(2)日志内容:用户可使用 markdown 语法来完成日志的添加。文字和链接将以 html 文件的形式存入数据库,图片也保存至数据库。每次加载日志时,会从数据库汇总获取并解析,解析后的内容会存入缓存,下次进入页面时不再重新加载和渲染。当用户修改日志内容后,系统直接将用户修改后的 html 文件存入数据库,同时将缓存中的内容同步修改。

2.4 通信模块

当用户进行通信时,系统会在用户之间建立一个长链接,用来保证信息可以实时进行传输。使用长链接还可以在每次传输中复用 TCP 链接,节约了信息传输时间。

通信协议遵守三层数据的规范,保证整体协议的规范化。第一层包含具体对象、具体场景编号以及第二层数据;第二层包含好友编号和第三层数据;第三层包含具体要传输的数据。

当用户编辑完消息后点击发送键,后台会将用户编辑的信息进行序列化,并通过长链接将序列化后的数据传输给对应的好友。对于收到的好友消

息，后台将遵守三层数据规范对数据进行解析，前端将反序列化的数据展示给用户，至此完成了用户之间的数据传输。由于保持着长链接的存在，用户进行数据传输的过程中减少了链接的反复建立，可以很大程度上提高用户间通信的实时性。

3　系统测试

3.1　用户登录主页

用户登录后的个人页面展示了用户在各个 OJ 平台中的做题信息，可以直观的查看自己在特定时间内做题的数量、标签分类以及平台分布。图 6 是用户在 2021 年 10 月份的做题情况。

其中，雷达图用来展示不同标签的做题量；饼状图用来展示不同题库的做题量；柱状图用来展示当月每日的做题量。

图 6　用户界面

Fig. 6　User interface

3.2　题库运行情况

训练赛定制页面如图 7 所示。用户可选择团队或个人的训练方式，并且可以添加训练赛题目：

图 7　题库运行图

Fig. 7　Question bank

3.3　训练赛定制和赛后复盘

训练赛定制页面如图 8 所示。用户可选择团队或个人的训练方式，并且可以添加训练赛题目。随机生成题目的过程如下。

(1) 当用户点击“随机”的时候，会检验当前所有的比赛参与者，个人训练为 1 人，团队训练为 3 人，并向服务端传递所有的参与者信息；同时服务端会通过异步的方式，筛选出所有参与者未通过的题目集合写入缓存中；

(2) 当用户对过滤条件进行选择并点击“确定”时，会向服务端传递过滤条件，服务端根据过滤条件分别从缓存中提取满足条件的题目集合，将这些题目集合的交集返回给用户[7]。

图 8　创建训练

Fig. 8　Create training match

用户可以自定义个人/团队的训练赛，训练赛后可以复盘比赛。复盘模块的目的是，引导用户逐步建立起及时复盘的习惯。复盘页面如图 9 所示。

图 9 复盘界面

Fig. 9 Match reply

3.4 用户日志页面运行图

用户可主动创建以及打开并编辑日志,所有日志都保存在用户本地磁盘中。日志运行界面如图 10 所示。

图 10 日志页面

Fig. 10 Log page

4 结束语

本系统实现了题目分析、训练赛添加、信息可视化展示、好友实时通信、日志添加等功能,为用户提供了一个全渠道的 ACM 技能提升系统,目的在于提升用户 ACM 竞赛水平。本文对其功能实现和相关技巧进行了详细阐述。系统功能多样、实用,交互性强,经过各个模块的测试,系统稳定性也符合预期,是一款符合用户需求的软件。

参考文献

[1] 白雪丽. 浅析基于 Python 爬虫技术的特性及应用[J]. 山西科技, 2018, 33(2): 53-55.

[2] 方生. 基于"Vue.js"前端框架技术的研究[J]. 电脑知识与技术, 2021,17(19):59-60,64.

[3] 邱红丽,张舒雅. 基于 Django 框架的 web 项目开发研究[J]. 科学技术创新,2021(27):97-98.

[4] 雷晓薇. 基于 Django 框架的教学管理系统的研究与实现[J]. 电子设计工程, 2018, 26(18):45-49,54.

[5] 于游,付钰,吴晓平. 中文文本分类方法综述[J]. 网络与信息安全学报,2019,5(5):1-8.

[6] 陈荟慧,熊杨帆,蒋滔滔,等. 基于在线测评系统的编程题目难度研究[J]. 现代计算机(专业版),2018(13):26-30,34.

[7] 李德光,李晓辉,张庆熙,等. 面向 Online Judge 提交日志的用户编程行为可视分析[J]. 计算机辅助设计与图形学学报,2020, 32(11):1731-1741.