



ACM 训练管理平台 —— 刘建东



数据库应用系统设计

实验报告

—— ACM 训练管理平台

计算机科学与技术学院 17 级菁英班

姓名：刘建东 学号：201700130011

任课教师：彭朝晖

助教：许晓康



目录

一、系统开发平台.....	4
1.1 开发语言：	4
1.2 开发工具：	4
1.3 操作系统：	4
二、数据库规划.....	5
2.1 任务陈述.....	5
2.2 任务目标.....	6
三、系统定义.....	8
3.1 系统边界.....	8
3.2 用户视图.....	10
·普通用户视图.....	10
·管理员用户视图 (后台查看)	10
四、需求分析.....	12
4.1 用户需求说明.....	12
· 4.1.1 数据需求.....	12
· 4.1.2 事务需求.....	13



五、逻辑设计	15
5.1 ER 图：	15
5.2 数据字典.....	16
六、数据库物理设计	18
6.1 数据库初始信息的录入.....	18
6.2 安全机制.....	20
七、应用程序设计	21
7.1 功能模块.....	21
7.2 界面设计.....	24
7.3 事务设计.....	46
八、测试和运行	65
九、总结	77
9.1 系统优点.....	77
9.2 系统不足.....	77
9.3 系统改进.....	77
9.4 经验与收获.....	77



一、系统开发平台

1.1 开发语言：

Python、Javascript

Python 利用 tornado 架构搭建后台服务端，Js 结合 html、css 写前端页面

1.2 开发工具：

MySQL 5.7.23、Navicat Premium for MySQL、Pycharm、WebStorm

1.3 操作系统：

MacOS Mojave 10.14.1



二、数据库规划

2.1 任务陈述

ACM 训练管理平台是对 ACM 选手的训练进行记录、统计和管理。其中不仅只是对单个选手进行记录，还可以对一支 ACM 三人队伍、一个 ACM 实验室、一个学校的所有 ACM 成员的训练进行管理。该平台包括个人训练情况查询和更新、队伍训练情况查询和更新、题库内各种类型题目的查询、各类型题目中难易题目的查询。其中训练情况的查询和更新包括训练题目的数量查询和更新、训练题目算法类别的查询和更新、训练题目难度的查询和更新、比赛情况的查询和更新。



2.2 任务目标

模块	功能名称	具体描述
选手注册	选手账号、基础信息	选手账号用来唯一标识一个选手，基本信息中包含姓名、性别、专业、年级、班级、以及账号密码。
	队伍信息	选手所在的队伍的名称、其他两个队友的信息。选手队伍可以为空，表示还没有组队。
	ACM 实验室名称	选手所在 ACM 实验室的名称，用来唯一标识一个实验室以及后续统计实验室成员训练信息，不能为空。
	学校名称	选手所在学校的名称，不能为空。
题库	题目基本信息显示	题目的基本信息有题目的标题以及题目的 ID，即题目编号，用来唯一标识一个题目。
	题目来源	包括题目所在 OJ 名称，OJ 中的题号以及该题目可提交平台的网址。
	题目算法分类	每个题目有任意个算法标签，表示该题目用到了哪些算法。
	题目难易程度	题目难易程度分为 3 档，分别是入门、进阶、大师。
	比赛	记录各场比赛的情况，比赛的榜单以及比赛的题目。
比赛	比赛基本信息显示	比赛编号、比赛名称、比赛开始时间和结束时间、比赛类型以及比赛所包含的题目。
	创建校内组队/个人赛	系统为用户提供创建比赛的功能，每个用户都可以自行创建比赛，设置比赛的基础信息，添加比赛的题目，以及设定比赛的类型。如校内组队赛、校内个人赛、CCPC、ICPC、省赛等。
	创建个人/队伍训练计划	只需要在创建比赛时对比赛类型进行设定即可。依然提供自由添加题目的功能。
	对比赛中的题目进行添加、删除操作	进入每个比赛的单独界面之后就可以自行添加或者删除题目。
个人训练管理	训练题目记录 ⁶	所有写过的题目都会被记录，支持添



		加、删除题目训练记录。每个题目的基本信息，包括难度、算法分类以及题目的具体提交网址都会显示出来。
	参加的比赛记录	支持添加、删除个人用户参加过的比赛，每个比赛的基础信息如个人排名、完成的题数、总罚时都会显示，包括比赛所包含的题目信息。
	加入过的队伍记录	用户加入过哪些队伍，队伍组建的时间以及队员，队伍参加过的比赛都会显示。
队伍训练管理	队伍基础信息的管理	记录队伍名字，队伍成员信息。
	参加比赛的记录	以队为单位参加过得所有比赛的记录，包括本场比赛完成的题数、队伍排名以及总罚时，也包括每个比赛的具体内容。
实验室比赛/训练管理	发布比赛或训练题	以实验室为单位创建实验室账号，再用实验室的账号创建比赛，比赛类型为个人赛/队伍赛均可。最后在比赛的具体页面添加比赛/训练题目即可。
学校比赛/训练管理	发布校内比赛或训练题	以学校账号创建比赛，比赛类型为校内个人赛/校内组队赛。然后在比赛的具体页面添加题目即可。

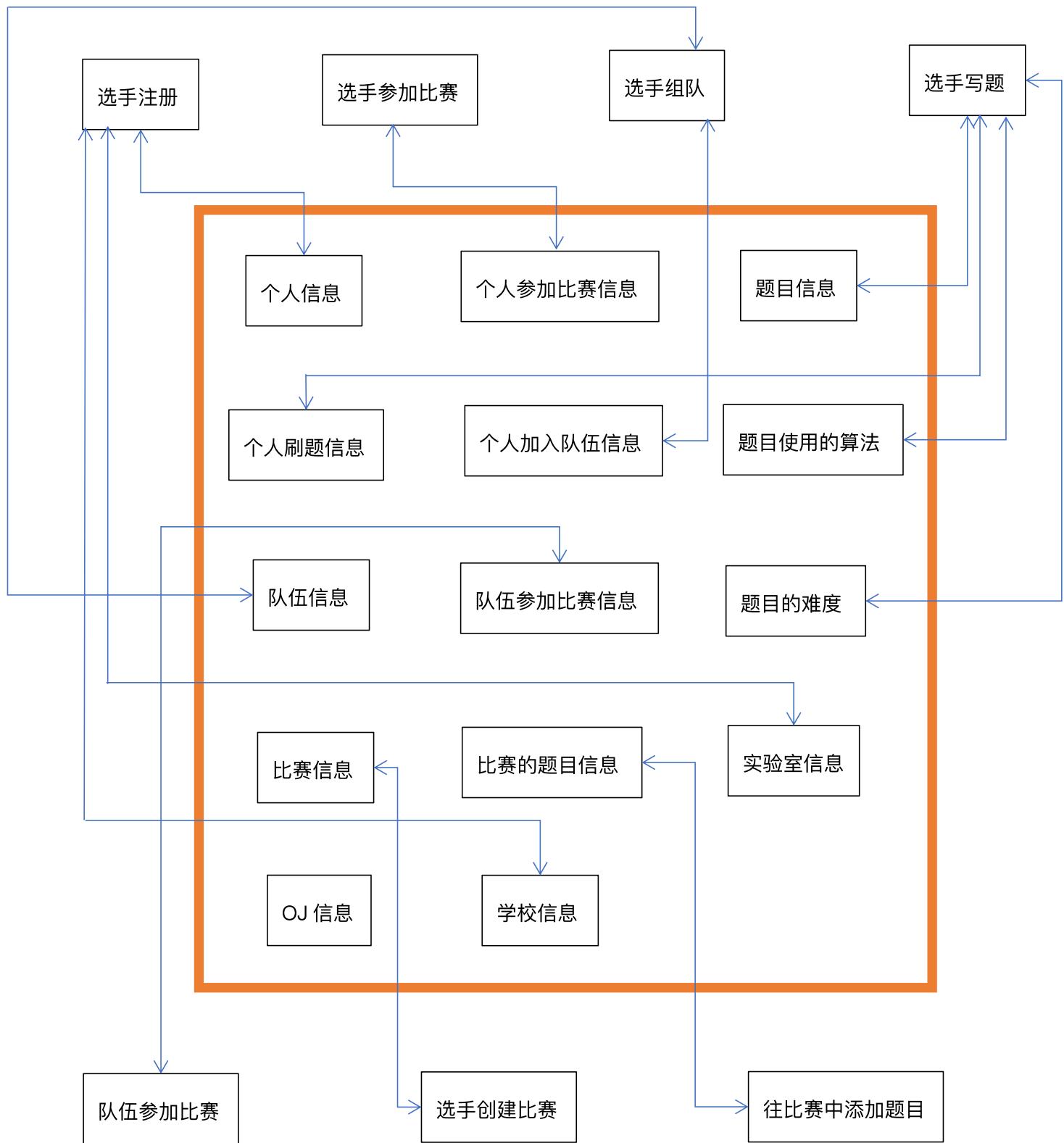


三、系统定义

3.1 系统边界

系统边界，即是系统所包含的功能与系统不包含的功能之间的界限。在本系统中主要是以个人、队伍、实验室、学校为单位的各项训练信息的记录以及支撑整个系统中所有数据的题库所包含的题目信息。

外部的信息如题目完整代码、题目在 OJ 上的完整交题记录以及数据获得的途径等一系列中间过程是本系统所不包含的，需要与内部信息做好区分。





3.2 用户视图

·普通用户视图

① 个人信息管理：

查看、修改个人信息。

② 个人做题记录管理：

查看、添加、删除个人做题记录。

③ 队伍信息管理：

查看、修改队伍信息，加入、退出队伍。

④ 队伍参加比赛记录管理：

记录队伍参加了哪些比赛，以及该场比赛的排名、罚时、做的题数等具体信息。

⑤ 用户创建的比赛：

查看用户创建的所有比赛，支持用户自己创建比赛，添加比赛的题目。

⑥ 题库信息：

查询题库中所有的题目具体信息。

·管理员用户视图（后台查看）

① 所有用户的基础信息：

添加、删除、修改用户信息，普通用户只能对自己的信息进行操作

② 所有比赛的信息：

添加、修改、删除比赛信息，普通用户不得删除比赛

③ 所有队伍的信息：



添加、删除、修改每个队伍参加的比赛，每个队伍的成员以及队伍的基础信息

④ 题库的信息：

添加、删除、修改题库中每个题目的基础信息，普通用户只能查看题库，不得

修改



四、需求分析

4.1 用户需求说明

· 4.1.1 数据需求

① 个人基本信息：

选手姓名、学号、大学、所属实验室、参加过的队伍、班级、年级、专业

② 个人训练记录：

所有训练过的题目在题库中的 ID、题目的名字、题目来源 OJ 与题号、题目算法分类、题目难度分类、个人参加过的比赛、比赛的名字、比赛 ID、比赛的题目、比赛的排名、比赛的罚时、比赛中完成的题数

③ 队伍基本信息：

队伍名称、队内成员、队伍组建时间

④ 队伍训练记录：

队伍一起参加过的比赛、队伍参加比赛的排名、队伍参加比赛的罚时、队伍参加比赛完成的题数

⑤ 题库内题目的基本信息：

题目来源、题目名称以及 ID、题目所涉及的算法类别、题目难度

⑥ 所有比赛的信息：

比赛名称、比赛的所有题目、比赛场的 ID、比赛的类型



· 4.1.2 事务需求

1. 数据录入：

① 录入选手个人的基本信息：

选手姓名、学号、大学、所属实验室、参加过的队伍、班级、年级、专业

② 录入选手训练的信息：

选手写过哪些题目、题目的算法、题目的难度、题目的来源

③ 录入选手参加过的比赛：

选手参加过哪些比赛、选手在比赛中的排名、选手在比赛中完成的题数、罚时

④ 录入队伍的基本信息：

队伍名称、队内成员、队伍组建时间

⑤ 录入队伍的训练信息：

队伍参加过哪些比赛、队伍参加比赛的排名、罚时、完成的题数

⑥ 录入题目：

题目标题、题目算法、题目难度、题目来源、题目链接

⑦ 录入比赛：

比赛名称、比赛开始时间、比赛结束时间、比赛中包含的题目、比赛的类型

2. 数据删除/更新：

① 选手基本信息的更新

② 选手完成过的题目的删除

③ 选手参加过的比赛的删除

④ 选手参加过的队伍的更新、删除



- ⑤ 队伍完成过的比赛的更新、删除
- ⑥ 比赛中所包含的题目的更新、删除

3. 数据查看：

- ① 查看选手的个人基本信息：

查看选手姓名、学号、大学、所属实验室、参加过的队伍、班级、年级、专业

- ② 查看选手训练的信息：

查看选手写过哪些题目、题目的算法、题目的难度、题目的来源

- ③ 查看选手参加比赛的信息：

查看选手参加的比赛的 ID、比赛中的排名、比赛的罚时、比赛中完成的题数

- ④ 查看选手参加过的队伍：

队伍中的成员个数、队伍中成员 ID、队伍组建时间

- ⑤ 查看队伍的训练信息：

查看队伍参加过哪些比赛、比赛中的排名、比赛的罚时、比赛中完成的题数

- ⑥ 查看题库中题目的基本信息：

题目的标题、题目的来源 OJ、OJ 中的题号、题目在 OJ 中的链接、题目的难

度、题目的算法

- ② 查看比赛的基本信息：

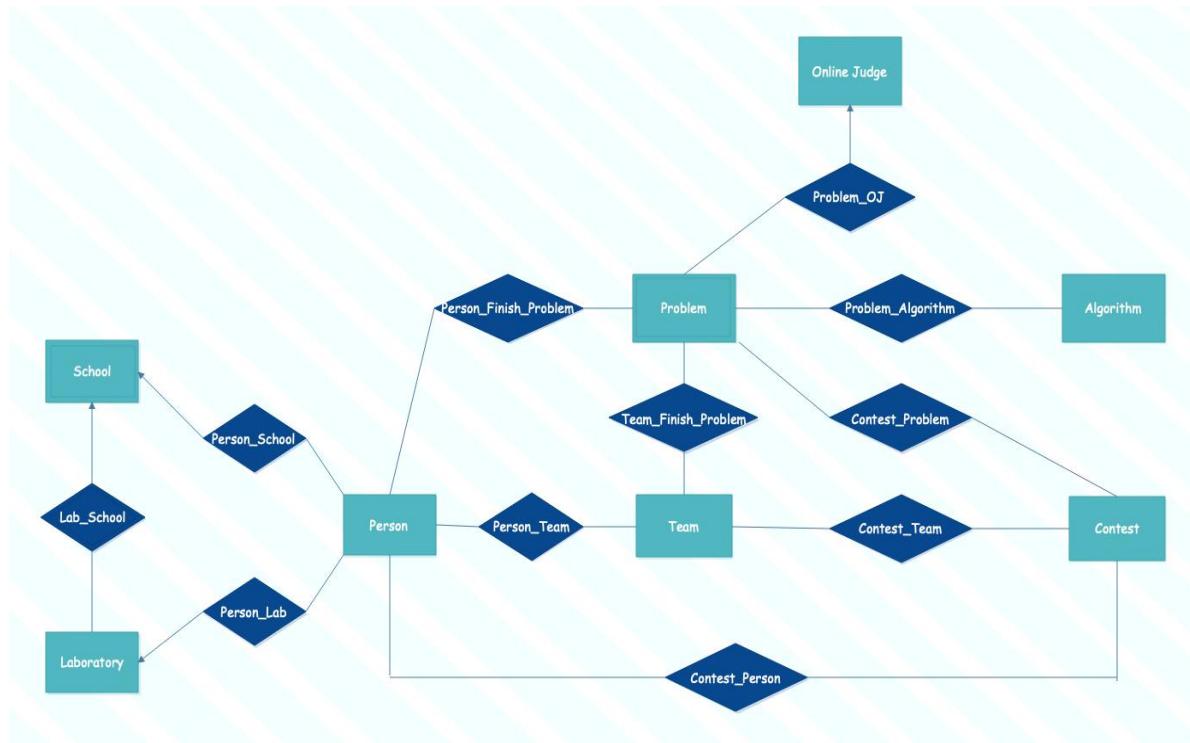
比赛的标题、比赛的开始时间、比赛的结束时间、比赛的类型、比赛中包含的

题目 ID、比赛中题目的算法、比赛中题目的难度、比赛中题目的标题



五、逻辑设计

5.1 ER 图：



该 ER 图共包含了 School、Laboratory、Person、Team、Problem、Contest、Algorithm、OnlineJudge 共 8 个实体以及 Lab_School、Person_Lab、Person_School、Person_Team、Person_Finish_Problem、Team_Finish_Problem、Contest_Person、Contest_Team、Contest_Problem、Problem_Algorithm、Problem_OJ 共 11 个联系。



5.2 数据字典

· 5.2.1 从数据字典中抽取出来的系统实体描述

Person:

Name	Type	Length	Decimals	Not Null	Virtual	Key
Person_id	varchar	30	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Name	varchar	40	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Gender	varchar	10	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Major	varchar	50	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Grade	int	30	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Class	varchar	30	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
School_id	varchar	20	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Lab_id	varchar	20	0	<input type="checkbox"/>	<input type="checkbox"/>	
Serect	varchar	255	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

School:

Name	Type	Length	Decimals	Not Null	Virtual	Key
School_id	varchar	20	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Name	varchar	30	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Country	varchar	20	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Province	varchar	30	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

Laboratory:

Name	Type	Length	Decimals	Not Null	Virtual	Key
Lab_id	varchar	20	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Name	varchar	40	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
School_id	varchar	20	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

Team:

Name	Type	Length	Decimals	Not Null	Virtual	Key
Team_id	varchar	20	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Name	varchar	30	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Lab_id	varchar	30	0	<input type="checkbox"/>	<input type="checkbox"/>	

Problem:

Name	Type	Length	Decimals	Not Null	Virtual	Key
Problem_id	varchar	30	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Problem_name	varchar	50	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
OJ_id	varchar	20	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
OJ_Pnumber	varchar	20	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Difficulty_level	varchar	20	0	<input type="checkbox"/>	<input type="checkbox"/>	
Link	varchar	255	0	<input type="checkbox"/>	<input type="checkbox"/>	



Algorithm:

Name	Type	Length	Decimals	Not Null	Virtual	Key
Algorithm_id	varchar	20	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Name	varchar	30	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Upper_id	varchar	20	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Online_Judge:

Name	Type	Length	Decimals	Not Null	Virtual	Key
OJ_id	varchar	20	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Name	varchar	20	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Link	varchar	255	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Contest:

Name	Type	Length	Decimals	Not Null	Virtual	Key
Contest_id	varchar	20	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Name	varchar	100	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Begin_time	varchar	100	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
End_time	varchar	100	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Contest_type	varchar	40	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

• 5.2.2 从数据字典中抽取出来的联系的描述:

实体	多样性	联系	多样性	实体	附加属性
Person	n	Person_School	1	School	NULL
Person	n	Person_Lab	1	Laboratory	NULL
Person	n	Person_Team	n	Team	Begin_time End_time
Laboratory	n	Lab_School	1	School	NULL
Problem	n	Person_Finish_Proble m	m	Person	Submit_time
Problem	n	Team_Finish_Proble m	n	Team	Submit_time
Problem	n	Contest_Problem	n	Contest	NULL
Problem	n	Problem_Algorithm	n	Algorithm	NULL
Problem	n	Problem_OJ	1	Online Judge	NULL
Contest	n	Contest_Team	n	Team	Rank Finished_number Time_total Honor_title
Contest	n	Contest_Person	n	Person	Rank Finished_number Time_total Honor_title



六、数据库物理设计

6.1 数据库初始信息的录入

6.1.1 题库信息的录入

① 题库信息是此平台中最重要的信息之一，是用户添加训练题，比赛添加题目的基础数据。因此此平台的数据都是使用爬虫从各大 OJ 上进行爬取的数据，爬取题目名字、题目链接、题目来源等信息。共爬取 1144 道题目。

Problem	1,144	176.00 KB	InnoDB	2018-12-21 18:00:58	utf8_general_ci
Problem_id	Problem_name	OJ_id	OJ_Pnumber	Difficulty_level	Link
0000001	Robberies	00003	2955	入门	http://acm.hdu.edu.cn/showproblem.php?pid=2955
0000002	最大报销额	00003	1864	大师	http://acm.hdu.edu.cn/showproblem.php?pid=1864
0000003	Bone Collector	00003	2602	入门	http://acm.hdu.edu.cn/showproblem.php?pid=2602
0000004	Coins	00003	2844	大师	http://acm.hdu.edu.cn/showproblem.php?pid=2844
0000005	FATE	00003	2159	进阶	http://acm.hdu.edu.cn/showproblem.php?pid=2159
0000008	Shopping Offers	00001	1170	进阶	http://ooi.ora/problem?id=1170

6.1.2 比赛信息的录入

① 比赛信息也是平台重要信息之一，是用户添加比赛、队伍添加比赛的基础数据。因此比赛从 vjudge 上进行爬取。共爬取 2079 项数据。

Contest	2,079	272.00 KB	InnoDB	2018-12-15 01:51:37	utf8_general_ci
Contest_id	Name	Begin_time	End_time	Contest_type	
000001	北京2015	2018-11-07 18:55:00	2018-11-07 23:55:00	校内组队赛	
000002	测试(非定级)赛 [Cloned]	2018-11-07 18:46:00	2018-11-07 19:46:00	校内个人赛	
000003	2018-2019 ACM-ICPC, NEERC, Southern	2018-11-07 18:45:00	2018-11-07 22:45:00	校内组队赛	
000004	Greater NY 2017 [Cloned]	2018-11-07 18:45:00	2018-11-07 23:45:00	校内组队赛	
000005	The 2016 ACM-ICPC Asia Beijing Region:	2018-11-07 18:42:00	2018-12-07 18:42:00	校内个人赛	
000006	Group 7 Beijing 2017 ICPC [Cloned]	2018-11-07 18:40:00	2018-11-07 23:40:00	校内个人赛	



6.1.3 用户信息的录入

① 用户信息主要从 ICPC 官网上进行爬取, ICPC 对于每场区域赛的参加队伍的队伍成员信息都会予以公布, 共爬取 544 项信息。

Person		544		80.00 KB InnoDB		2018-12-18 11:41:56		utf8_general_ci	
Person_id	Name	Gender	Major	Grade	Class	School_id	Lab_id	Serect	
000001	楼天城	男	计算机科学与技术学院	2004	姚班	00003	000004	rootroot	
000002	陈立杰	男	计算机科学与技术学院	2013	姚班	00003	000004	nsaTJwD5	
000003	杜瑜皓	男	计算机科学与技术学院	2016	姚班	00003	000004	cGC9eqR5	
000004	金策	男	计算机科学与技术学院	2016	姚班	00003	000004	ujWEbD2Q	
000005	吉如一	男	计算机科学与技术学院	2016	一班	00002	000003	CT3g1OUm	
000006	洪华敦	男	计算机科学与技术学院	2017	一班	00002	000003	VvC0AD3r	
000007	丁力煌	男	数学科学学院	2017	二班	00002	000003	NFJewUCY	

6.1.4 队伍信息的录入

① 队伍信息也是从 ICPC 官网上进行的爬取, 是爬下来了每个队伍的信息以及队伍中所包含的成员信息, 成员信息构成了 Person 这个表格, 队伍信息构成了 Team 这个表格, Person 与 Team 之间的关系构成了 Person_Team 这个表格。共爬取了 179 项队伍信息。

Team		179		16.00 KB InnoDB		2018-12-24 00:51:18		utf8_general_ci	
Person_Team		537		64.00 KB InnoDB		2018-12-07 18:16:48		utf8_general_ci	
Team_id	Name			Lab_id					
000001	墙角猫			000083					
000002	韦天魔术棒			000023					
000003	常山赵子龙			000023					
000004	小红帽烤冷面			000023					
000005	1977的回忆			000084					
000006	一颗柠檬			000084					
Person_id	Team_id	Begin_time	End_time						
000008	000001	2016-09-28	NULL						
000009	000001	2016-09-28	NULL						
000010	000001	2016-09-28	NULL						
000011	000002	2017-04-08	2018-05-03						
000012	000002	2017-04-08	2018-05-03						
000013	000002	2017-04-08	2018-05-03						
000014	000003	2017-02-25	2018-06-14						



6.2 安全机制

6.2.1 用户权限安全

- ① 登录界面需要输入选手的账号、密码，未登录之前只能看到登录界面。
- ② 在应用界面中普通用户对于题库中的题目以及所有的比赛，都不能执行删除操作，避免数据被恶意清空。题库中的题目以及比赛的删除只能通过管理员在后台进行删除。

6.2.2 添加数据安全

- ① 用户注册时需要输入用户所在学校、实验室名称、性别、入学年份，此平台统一采用了下拉菜单的方式进行实现，一是避免用户错误输入数据格式，二是防止用户恶意输入破坏数据库中的数据。
- ② 在用户个人信息修改界面、密码修改界面、添加比赛、添加比赛中的题目、删除比赛中的题目、加入队伍、退出退伍、添加个人刷题记录、删除个人刷题记录、添加个人参加比赛记录、删除个人参加比赛记录、添加队伍参加比赛记录、修改队伍的基本信息等几乎所有事务中，都对于可能发生的错误信息输入进行了规避，并且会根据不同的错误信息输入页面返回不同的错误提示来引导用户，同时如果信息输入正确，系统会显示操作成功的信息来告知用户。

在信息填取栏中，大量使用下拉菜单的方式来规避用户的错误输入，下拉菜单的内容都是通过查询数据库中的信息进行显示。比如在删除和添加用户参加过的比赛的页面，下拉菜单提供用户想要插入的比赛 ID，而下拉菜单中不包括用户已经参加过的比赛，因此避免了用户重复添加比赛的操作。

同上类似的大量信息输入规范设置都在页面设计中有所体现。



七、应用程序设计

7.1 功能模块

此 ACM 训练管理平台共有个人基础信息、题库信息、比赛信息、用户训练管理信息、队伍信息这五大模块，功能模块具体实现如下：

7.1.1 个人基础信息：

- ① 修改、查看用户姓名
- ② 修改、查看用户性别
- ③ 修改、查看用户所在学校
- ④ 修改、查看用户入学年级
- ⑤ 修改、查看用户所在的大学专业
- ⑥ 修改、查看用户所在专业的班级
- ⑦ 修改、查看用户所在学校的实验室
- ⑧ 用户的账号密码修改

7.1.2 题库信息：

- ① 查看题目 ID，用于唯一标识题目
- ② 查看题目所属的 OJ、题目所属的 OJ 的题号
- ③ 查看题目的标题
- ④ 查看题目所在网页的链接，可以到题目所在网页进行交题
- ⑤ 查看题目的难度



- ⑥ 查看题目包含的算法
- ⑦ 查看整个题库的算法分类、算法集合
- ⑧ 可以按 OJ、算法、难度对题目进行排序和检索
- ⑨ 可以将整个题库导出 excel 或者打印

7.13 比赛信息：

- ① 查看比赛 ID
- ② 查看比赛标题
- ③ 查看比赛开始和结束时间
- ④ 查看比赛类型
- ⑤ 可以按照比赛 ID、标题、开始时间、结束时间、比赛类型进行排序和检索
- ⑥ 可以查看每个比赛具体包含了哪些题目
- ⑦ 可以往单个比赛中添加题目
- ⑧ 可以添加比赛

7.14 用户训练管理信息：

- ① 查看用户写过哪些题目
- ② 查看用户写过题目的具体信息
- ③ 添加用户写过的题目
- ④ 删除用户写过的题目
- ⑤ 查看用户参加过的比赛
- ⑥ 查看用户参加的比赛的具体信息，如比赛中的排名、比赛中完成的题数、比赛的总罚时



-
- ⑦ 可以添加、删除用户参加过的比赛
 - ⑧ 查看用户参加过的队伍、队伍名、队伍组建的时间
 - ⑨ 创建一个队伍
 - ⑩ 退出一个队伍

7.15 队伍信息：

- ① 修改、查看队伍名
- ② 修改、查看队伍的其它两个队员
- ③ 查看队伍参加过的比赛、比赛 ID、比赛中包含的题目
- ④ 查看队伍参加比赛的具体情况，比赛的排名、比赛完成的题数、比赛的总罚时
- ⑤ 添加队伍参加过的比赛
- ⑥ 删除队伍参加过的比赛



7.2 界面设计

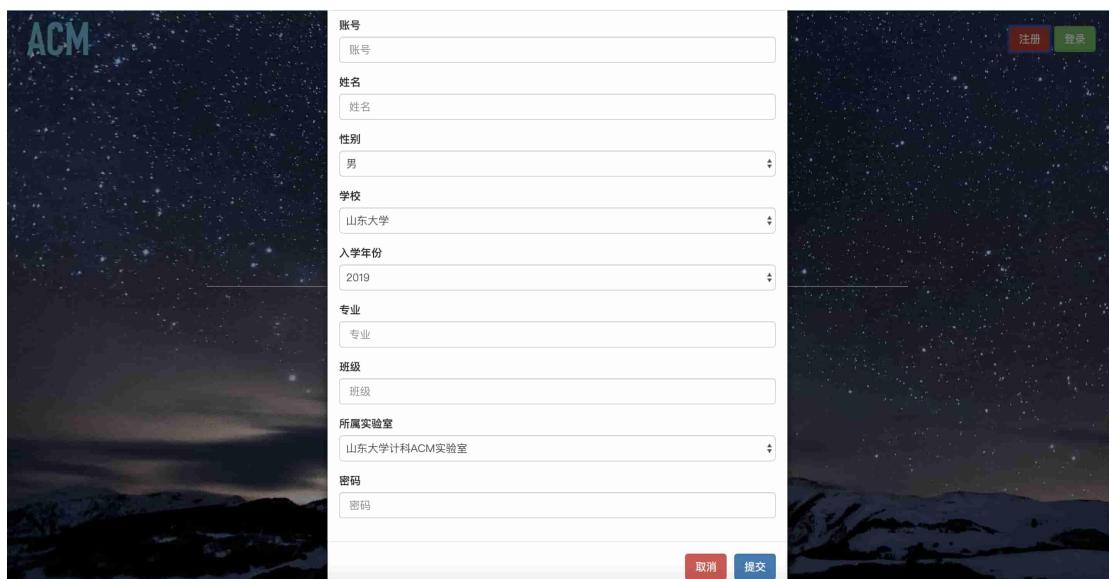
7.2.1 开始界面

用户进入开始界面后可以进行注册，也可以进行登录，登录成功后即可进入平台主界面。



7.2.2 注册界面

用户点击开始界面中的注册按钮，就会弹出注册框，按照说明填写注册信息即可完成注册，完成注册后，页面会显示注册成功。



学校和实验室都是系统内事先导入的数据，因此是下拉菜单进行选择，不支持直接输入。



学校

- ✓ 山东大学
- 北京大学
- 清华大学
- 浙江大学
- 上海交通大学**
- 复旦大学
- 杭州电子科技大学
- 武汉大学
- 北京航空航天大学
- 中国人大大学
- 北京交通大学
- 北京工业大学
- 北京理工大学
- 北京科技大学
- 北京化工大学
- 北京邮电大学
- 中国农业大学

性别

- 男
- ✓ 山东大学计科ACM实验室
- 山东大学软工ACM实验室
- 北京大学计科ACM实验室
- 清华大学计科ACM实验室
- 浙江大学计科ACM实验室
- 浙江大学软工ACM实验室
- 上海交通大学计科ACM实验室
- 复旦大学计科ACM实验室
- 复旦大学软工ACM实验室
- 杭州电子科技大学计科ACM实验室
- 武汉大学计科ACM实验室
- 北京航空航天大学计科ACM实验室
- 中国人大大学计科ACM实验室
- 中国人大大学软工ACM实验室
- 北京交通大学计科ACM实验室
- 北京交通大学软工ACM实验室
- 北京工业大学计科ACM实验室**
- 北京工业大学软工ACM实验室
- 北京理工大学计科ACM实验室
- 北京理工大学软工ACM实验室
- 北京科技大学计科ACM实验室
- 北京科技大学软工ACM实验室
- 北京化工大学计科ACM实验室
- 北京邮电大学计科ACM实验室

Objects Contest@Gene (Gene) School@Gene (Gene)

School_Id	Name	Country	Province
00001	山东大学	中国	山东省
00002	北京大学	中国	北京市
00003	清华大学	中国	北京市
00004	浙江大学	中国	浙江省
00005	上海交通大学	中国	上海市
00006	复旦大学	中国	上海市
00007	杭州电子科技大学	中国	浙江省
00008	武汉大学	中国	湖北省
00009	北京航空航天大学	中国	北京市
00010	中国人大大学	中国	北京市
00011	北京交通大学	中国	北京市
00012	北京工业大学	中国	北京市
00013	北京理工大学	中国	北京市
00014	北京科技大学	中国	北京市
00015	北京化工大学	中国	北京市
00016	北京邮电大学	中国	北京市
00017	中国农业大学	中国	北京市
00018	北京林业大学	中国	北京市
00019	中国传媒大学	中国	北京市
00020	中央民族大学	中国	北京市
00021	北京师范大学	中国	北京市
00022	中国石油大学	中国	北京市
00023	华东师范大学	中国	上海市
00024	上海外国语大学	中国	上海市

Objects Laboratory@Gene (Gene)

Lab_Id	Name	School_id
000001	山东大学计科ACM实验室	00001
000002	山东大学软工ACM实验室	00001
000003	北京大学计科ACM实验室	00002
000004	清华大学计科ACM实验室	00003
000005	浙江大学计科ACM实验室	00004
000006	浙江大学软工ACM实验室	00004
000007	上海交通大学计科ACM实验室	00005
000008	复旦大学计科ACM实验室	00006
000009	复旦大学软工ACM实验室	00006
000010	杭州电子科技大学计科ACM实验室	00007
000011	武汉大学计科ACM实验室	00008
000012	北京航空航天大学计科ACM实验室	00009
000013	中国人大大学计科ACM实验室	00010
000014	中国人大大学软工ACM实验室	00010
000015	北京交通大学计科ACM实验室	00011
000016	北京交通大学软工ACM实验室	00011
000017	北京工业大学计科ACM实验室	00012
000018	北京工业大学软工ACM实验室	00012
000019	北京理工大学计科ACM实验室	00013
000020	北京理工大学软工ACM实验室	00013
000021	北京科技大学计科ACM实验室	00014

并且注册时要求密码和账号不能为空，如果密码和账号为空，页面会显示相关错误信息，

直到注册信息按要求填入之后，页面才会显示注册成功。

账号

账号不能为空

密码

密码不能为空



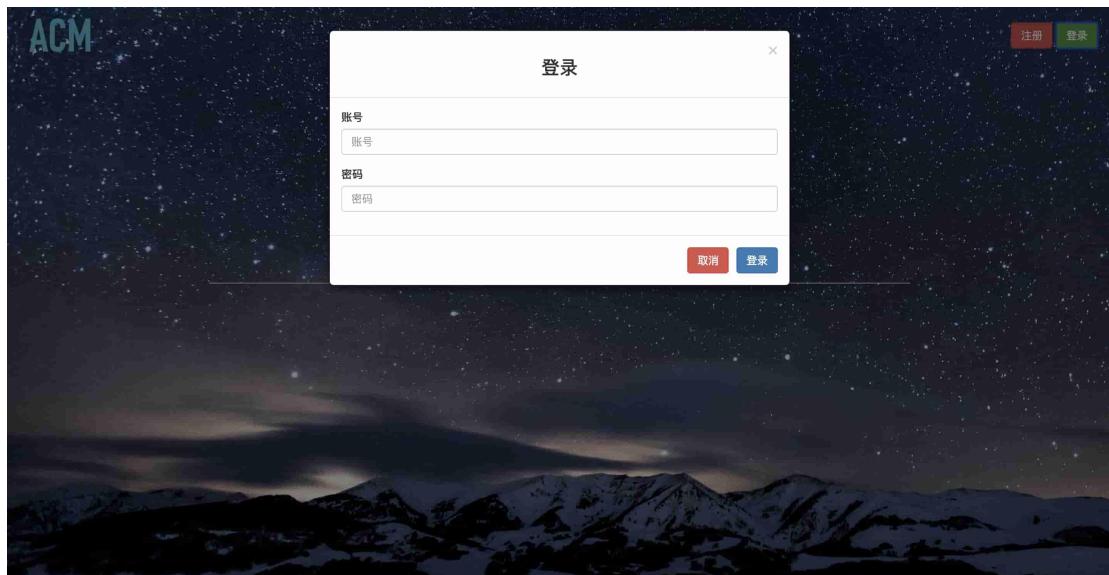
注册成功

账号

账号

7.2.3 登录界面

用户点击开始界面中的登录按钮，就会弹出登录框，正确输入账号和密码即可进入。



如果账号密码不匹配，页面会显示“账号或密码错误”来提示用户，如果密码和账号匹配，则会直接进入平台页面。

账号

账号或密码错误

账号

7.2.4 平台主页面-首页

平台首页用一个幻灯片进行展示，是一些关于 ACM-ICPC 图片的展示。



ACM 训练管理平台 —— 刘建东

ACM

首页 题库 比赛 用户

Gene_Liu



近期题目 推荐

BZOJ 2955

Robberies [入门]

网络流

往下翻分别是近期题目推荐和近期比赛推荐的信息。页面右上角显示的是登入的账号名。

ACM

首页 题库 比赛 用户

Gene_Liu

POJ 2828

BuyTickets [入门]

动态规划、数据结构

HDU 1698

Just a Hook [大师]

网络流

BZOJ 1041

[HAOI2008]圆上的整点 [进阶]

数学、计算几何

近期比赛

07

2018-11

ACM-ICPC北京赛区2017 [ICPC]

16:20-21:20

07

2018-11

绯色月下 风雨中做蛐蛐 [校内个人赛]

14:12-19:12

06

7.2.5 平台主页面-我的资料

点击右上角，会有退出和我的资料两个选项，如果选择退出，则会回到一开始的平台开始页面。如果点击我的资料，则会进入我的资料页面。



近期题目 推荐

我的资料页面分为两部分内容，一部分内容是个人信息的查看、修改、保存，个人信息包括姓名、性别、学校、年级、专业、班级以及所属的实验室。

我的资料页面中学校、实验室、年级依然是用下拉菜单实现，与注册页面的选项相同。未修改之前显示的信息就是自己的个人信息，修改之后点击保存，个人信息就修改成功了。

另一部分是修改密码，三个输入框，分别是原密码、新密码以及再次确认，只要原密码正确，并且新密码和再次确认的密码相同，即可修改成功，否则系统会显示修改错误的信息。



Chase Dreams!

目标的坚定是性格中最必要的力量源泉之一，也是成功的利器之一。没有它，天才会在矛盾无定的迷径中徒劳无功。

个人信息

修改密码

原密码

原密码

新密码

新密码

确认

再次确认

返回

修改

如果原密码不正确，页面会显示原密码错误。

原密码

新密码

确认

原密码错误

返回

修改

如果新密码和确认密码不一致系统会显示两次密码输入不一致，如果新密码为空，系统会显示新密码不得为空。

确认

再次确认

新密码不得为空

返回

修改

确认

再次确认

新密码两次输入不一致

7.2.5 平台主页面-题库

点击题库按钮，即可进入题库界面，题库界面也由两部分组成，第一部分是题目集合，会将数据库中所有的题目显示出来，包括题目 ID、标题、所属 OJ、OJ 中的题目编号、题



目的难度、题目的算法分类。题目的标题是一个链接，点击标题即可跳转到所在的 OJ 页面，用户可以提交。用户也可以直接进行搜索，支持模糊搜索。支持页面一次展示 10 行、20 行、50 行、100 行、200 行，也支持页面表格导出 excel 和打印页面表格。

The screenshot shows a web-based ACM training management system. At the top, there is a navigation bar with links for '首页' (Home), '题库' (Problem库), '比赛' (Contest), and '用户' (User). A user profile 'Gene_Liu' is shown on the right. The main content area features a dark banner with the text 'Chase Dreams!' and a quote: '目标的坚定是性格中最必要的力量源泉之一，也是成功的利器之一。没有它，天才会在矛盾无定的迷径中徒劳无功。'. Below the banner is a table of problems:

ID	OJ & Number	标题	难度	算法
0000001	HDU 2955	Robberies	入门	背包
0000002	HDU 1864	最大报销额	大师	背包
0000003	HDU 2602	Bone Collector	入门	背包
0000004	HDU 2844	Coins	大师	背包
0000005	HDU 2159	FATE	进阶	背包
0000008	POJ 1170	Shopping Offers	进阶	背包

Below the table are buttons for 'Show 20 rows', '导出Excel', and '打印表格'. A dropdown menu is open, showing options for 'Show 20 rows': 10, 20 (selected), 50, 100, 200, and All. The table also includes sorting icons for 'OJ & Number' and '标题'.

我们可以对于题目 ID、OJ、难度、算法进行排序，只需点击表格项右边的上升和下降选项即可。



ID	OJ & Number	标题	难度	算法
0000239	HDU 5008	Boring String Problem	入门	后缀数组结构
0000236	HDU 4691	Front compression	入门	后缀数组结构
0000235	HDU 3948	The Number of Palindromes	入门	后缀数组结构
0000232	HDU 2896	病毒侵袭	入门	AC自动机
0000227	HDU 2222	Keywords Search	入门	AC自动机
0000224	HDU 4763	Theme Section	入门	KMP

也可以在搜索框中进行搜索，比如搜索 HDU 的入门线段树的题目。

Show 20 rows		导出Excel	打印表格	搜索: HDU 入门 线段树
ID	OJ & Number	标题	难度	算法
0000173	HDU 3950	Parking Log	入门	线段树
0000167	HDU 2795	Billboard	入门	线段树
0000164	HDU 1166	敌兵布阵	入门	线段树

另一部分是算法分类，将数据库中算法表格中的内容展示出来，用户可以根据算法分类的内容到题目集合中进行搜索。

The screenshot shows the 'Algorithm Categories' section of the platform. It includes sections for Dynamic Programming, Search, Computational Geometry, and Trees. Each section lists various sub-topics or algorithms.

动态规划					
LCS	LIS	背包	单调性DP	环形DP	树形DP
状态压缩DP					

搜索					
枚举	搜索与剪枝	启发式搜索	DLX	双向搜索	折半搜索
记忆化搜索	模拟退火				

计算几何					
半平面交	凸包	几何图形的交与并	旋转卡壳	点定位	坐标变换
离散化与扫描	反演	Voronoi图	平面图的对偶图	三角剖分	梯形剖分
几何知识					

树结构					
最近公共祖先	生成树	DFS序列	树上倍增	树的分治	树链剖分
Link-Cut-Tree					

7.2.6 平台主页面-比赛

点击比赛按钮，即可进入比赛界面，比赛界面也由两部分组成，第一部分是比赛集合，会将数据库中所有的比赛显示出来，包括比赛 ID、标题、比赛开始时间和结束时间、比赛类型。也可以直接进行搜索，支持模糊搜索。也支持所有比赛信息的导出与打印。



The screenshot shows a dark-themed web interface for managing ACM contests. At the top, there's a navigation bar with 'ACM' and links for '首页', '题库', '比赛', and '用户'. A user profile 'Gene_Liu' is also at the top right. Below the header is a large banner with the text 'Chase Dreams!' and a quote: '目标的坚定是性格中最必要的力量源泉之一，也是成功的利器之一。没有它，天才会在矛盾无定的迷径中徒劳无功。' The main content area displays a table of contests:

ID	Name	开始时间	结束时间	比赛类型
000001	北京2015	2018-11-07 18:55:00	2018-11-07 23:55:00	校内组队赛
000002	测试(非定级)赛 [Cloned]	2018-11-07 18:46:00	2018-11-07 19:46:00	校内个人赛
000003	2018–2019 ACM–ICPC, NEERC, Southern Subregional Contest, Qualification Stage [Cloned]	2018-11-07 18:45:00	2018-11-07 22:45:00	校内组队赛
000004	Greater NY 2017 [Cloned]	2018-11-07 18:45:00	2018-11-07 23:45:00	校内组队赛
000005	The 2016 ACM–ICPC Asia Beijing Regional Contest [Cloned]	2018-11-07 18:42:00	2018-12-07 18:42:00	校内个人赛
000006	Group 7 Beijing 2017 ICPC [Cloned]	2018-11-07 18:40:00	2018-11-07 23:40:00	校内个人赛

表格显示方式、导出方式、搜索方式、排序方式与题库中的表格一致，比如搜索所有的校内组队赛。

The screenshot shows the same interface after performing a search for '校内组队赛' (Intra-team competition). The search results table is identical to the one above, showing the same six contests.

比赛标题是一个链接，点击比赛标题，即可跳转到每个比赛所属的具体页面。在该页面可以查看该比赛所包含的所有题目，包括每个题目的具体信息。也支持往比赛中添加题目。题目添加成功后，页面就会自动显示先添加的题目。



Chase Dreams!

目标的坚定是性格中最必要的力量源泉之一，也是成功的利器之一。没有它，天才会在矛盾无定的迷径中徒劳无功。

比赛标题

测试(非定级)赛 [Cloned]

ID: 000002

添加题目

ID	OJ & Number	标题	难度	算法
0000477	BZOJ 1228	[SDOI2009]E&D	进阶	
0001077	BZOJ 1828	[ZJOI2012]数列(sequence)	进阶	
0001083	BZOJ 1834	[BeiJing wc2012]冻结	大师	
0001111	BZOJ 1862	[HEOI2012]旅行问题	进阶	
0001128	BZOJ 1879	[JLOI2011]飞行路线	进阶	

ACM

首页 题库 比赛 用户

Gene_Liu

Chase Dreams!

目标的坚定是性格中最必要的力量源泉之一，也是成功的利器之一。没有它，天才会在矛盾无定的迷径中徒劳无功。

取消

添加

比赛标题

测试(非定级)赛 [Cloned]

ID: 000002

添加题目

ID	OJ & Number	标题	难度	算法
0000477	BZOJ 1228	[SDOI2009]E&D	进阶	
0001077	BZOJ 1828	[ZJOI2012]数列(sequence)	进阶	
0001083	BZOJ 1834	[BeiJing wc2012]冻结	大师	
0001111	BZOJ 1862	[HEOI2012]旅行问题	进阶	
0001128	BZOJ 1879	[JLOI2011]飞行路线	进阶	

比赛主界面的另一部分是添加比赛，可以根据平台给出的信息自主对比赛进行添加。

ACM

首页 题库 比赛 用户

Gene_Liu

Chase Dreams!

目标的坚定是性格中最必要的力量源泉之一，也是成功的利器之一。没有它，天才会在矛盾无定的迷径中徒劳无功。

比赛集合

添加比赛

比赛标题

比赛标题

持续时间

1

时间单位

秒

比赛类型

校内个人赛

返回

添加



比赛信息输入正确之后，页面就会显示添加成功的字样，这个比赛也就成功地添加到了数据库中。可以在前面所有比赛的页面中进行查找。

The screenshot shows a search results page for competitions. At the top, there is a search bar with the text "期末考核". Below it is a table with columns: ID, Name, Start Time, End Time, and Competition Type. One row is visible, showing ID 002084, Name 期末考核, Start Time 2019-01-05 10:32:26, End Time 2019-01-09 14:32:26, and Competition Type 校内个人赛. The table has a header row and a data row.

7.2.7 平台主页面-用户

点击用户按钮，即可进入用户界面，用户界面由三部分组成，第一部分是用户写过的题目的展示，会将所有用户写过的题目都显示出来，包括题目 ID、标题、题目所属 OJ、题目链接、题目算法以及题目的提交时间。在页面的右上角提供添加题目和删除题目的按钮，支持用户自行添加题目和删除题目。

The screenshot shows a user profile page with a dark background. At the top, there is a banner with the text "Chase Dreams!" and a quote: "目标的坚定是性格中最必要的力量源泉之一，也是成功的利器之一。没有它，天才会在矛盾无定的迷径中徒劳无功。". Below the banner, there are tabs: "写过的题目" (selected), "个人参加的比赛", and "队伍信息". On the right, there are buttons for "添加题目" (green) and "删除题目" (orange). A table below lists solved problems with columns: Problem_ID, OJ & Number, 标题, 难度, 算法, 提交时间. The table contains several rows of problem data.

点击添加题目的按钮，就会弹出添加题目的页面。页面中只需选择要添加的题目 ID 即可，如果添加的题目是之前没做过的题目，页面就会显示添加成功。



如果添加了之前做过的题目，页面就会显示此题已经做过，无法重复添加。

点击删除题目的按钮，就会弹出删除题目的界面。



Problem_ID	OJ & Number	Title	Difficulty	Algorithm	Submission Time
0000001	HDU 2955	Robberies	Easy	Backpack	2018-12-23 17:14:26
0000002	HDU 1864	Maximum Reward	Medium	Backpack	2018-12-23 17:14:31
0000014	HDU 1087	Super Jumping!	Advanced	LIS	2018-12-23 17:14:33

删除题目的界面中提供一个下拉菜单进行选择，其中全部都是该用户做过的题目 ID。

Problem_ID	OJ & Number	Title	Difficulty	Algorithm	Submission Time
0000001	HDU 2955	Robberies	Easy	Backpack	2018-12-23 17:14:26
0000002	HDU 1864	Maximum Reward	Medium	Backpack	2018-12-23 17:14:31

点击删除按钮之后，如果删除成功，系统就会显示删除成功的字样。

删除成功

第二部分是用户参加过的比赛。包括用户参加比赛的具体信息，比如比赛 ID、用户在这次比赛中的排名、完成的题数以及用户的总罚时。比赛 ID 是一个链接，点击链接即可跳转到比赛的具体页面中。页面右上角提供了添加比赛和删除比赛两个按钮，支持用户自行添



加或者删除比赛。

The screenshot shows the 'Contests' section of the ACM training management platform. At the top, there is a navigation bar with links for '首页', '题库', '比赛', and '用户'. A user profile icon for 'Gene_Liu' is also present. Below the navigation bar, a large banner displays the text 'Chase Dreams!' and a quote: '目标的坚定是性格中最必要的力量源泉之一，也是成功的利器之一。没有它，天才会在矛盾无定的迷径中徒劳无功。' Below the banner, there are three tabs: '写过的题目' (Written Problems), '个人参加的比赛' (Contests Participated In), and '队伍信息' (Team Information). The '个人参加的比赛' tab is selected. On the right side of the page, there are two buttons: '添加比赛' (Add Contest) and '删除比赛' (Delete Contest). Below these buttons, a table lists three contests with their details:

Contest_ID	排名	完成题数	总罚时
000018	1	9	13:19:00
000040	2	8	2 days, 23:13:00
000057	3	7	3 days, 6:42:00

点击添加比赛的按钮，就会弹出添加比赛的页面，需要填写的信息分别是比赛 ID、完成题数、比赛排名、罚时(时)、罚时(分)这几个信息，信息填取全部采用下拉菜单选择的方式，避免用户不按规则填入信息。

The screenshot shows a modal dialog titled '添加比赛' (Add Contest). The dialog contains four input fields with dropdown menus: '比赛ID' (Contest ID) with value '000001', '完成题数' (Completed Problems) with value '1', '排名' (Rank) with value '1', and '罚时 (时)' (Penalty Time (Hours)) with value '00'. At the bottom of the dialog, there are two buttons: '取消' (Cancel) and '添加' (Add).

并且与之前添加题目不同，此处添加比赛 ID 的下拉菜单中不显示已经做过的题目。



The screenshot shows a dropdown menu titled "比赛ID" (Competition ID) with a list of IDs from 000001 to 000025. The ID 000017 is highlighted with a red box. The background shows a user interface with tabs for "写的题目" (Written Programs), "个人参加的比赛" (Personal Competitions), and "队伍信息" (Team Information). There are also buttons for "添加比赛" (Add Competition) and "删除比赛" (Delete Competition).

点击删除比赛的按钮，就会弹出删除比赛的界面。

The screenshot shows a modal dialog box titled "删除题目" (Delete Competition) with a single input field for "比赛ID" (Competition ID) containing the value 000018. Below the input field are two buttons: "取消" (Cancel) and "删除" (Delete). The background shows the same user interface as the previous screenshot, with the "个人参加的比赛" tab selected.

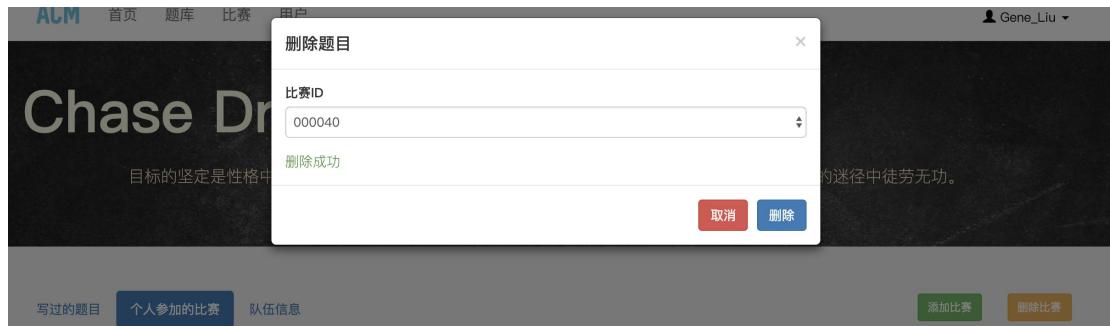
与删除题目的界面一样，删除比赛的界面提供一个下拉菜单用来选择比赛 ID，并且其中的比赛 ID 都是该用户已经参加过的比赛 ID。

The screenshot shows a modal dialog box titled "删除题目" (Delete Competition) with a dropdown menu for "比赛ID" (Competition ID) containing three options: 000018, 000040, and 000057. The option 000018 is selected. Below the dropdown are two buttons: "取消" (Cancel) and "删除" (Delete). The background shows the user interface with the "个人参加的比赛" tab selected, displaying a table of competition results.

Contest_ID	排名	完成题数	总罚时
000018	1	9	13:19:00
000040	2	8	2 days, 23:13:00
000057	3	7	3 days, 6:42:00



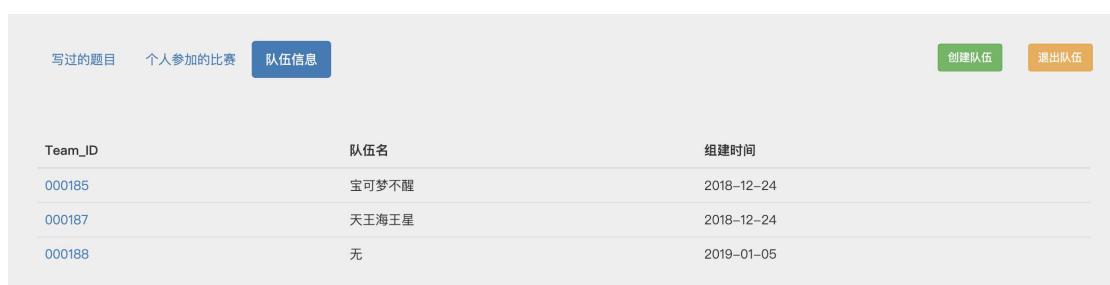
点击删除按钮，如果删除成功，系统就会显示删除成功的信息。



第三部分是用户的队伍信息，显示用户加入过的队伍，分别显示队伍 ID、队伍名以及队伍的组建时间。可以看到队伍 ID 是一个链接，点击队伍 ID 即可跳转到队伍的管理界面。



并且支持用户自行创建队伍，点击右上角的创建队伍按钮，系统就会自动创建一个队伍，而队伍的具体信息需要到队伍界面中进行修改。



也支持用户退出队伍，点击右上角的退出队伍按钮，就会弹出退出队伍的界面。



The screenshot shows the user interface for managing teams. At the top, there's a navigation bar with links for '首页', '题库', '比赛', and '用户'. On the right, it shows the user's name 'Gene_Liu'. Below the navigation, there are three tabs: '写过的题目', '个人参加的比赛', and '队伍信息', with '队伍信息' being the active tab. A modal dialog box titled '退出队伍' (Exit Team) is open. Inside the dialog, there's a dropdown menu labeled '队伍ID' (Team ID) containing the options '000185', '000187', and '000188'. At the bottom of the dialog are two buttons: '取消' (Cancel) and '删除' (Delete). In the background, there's a table listing teams with columns for 'Team_ID', '队伍名' (Team Name), and '组建时间' (Establishment Time). The table contains three rows:

Team_ID	队伍名	组建时间
000185	宝可梦不醒	2018-12-24
000187	天王海王星	2018-12-24
000188	无	2019-01-05

退出队伍的界面中可以选择想要退出队伍的队伍 ID，依然采用下拉菜单的方式，其中显示该用户目前已经加入队伍的所有队伍 ID。

This screenshot shows the 'Exit Team' dialog box again. The '队伍ID' dropdown now has a checkmark next to '000185'. The other two options, '000187' and '000188', are still listed below it. The '取消' (Cancel) and '删除' (Delete) buttons are at the bottom.

如果删除成功，页面就会显示退出成功的信息。

This screenshot shows the platform's main page after a successful exit operation. The '队伍信息' tab is still active. The 'Exit Team' dialog box is open, showing the message '退出成功' (Exit successful) in green text. The '队伍ID' dropdown still contains '000185'. The '取消' (Cancel) and '删除' (Delete) buttons are at the bottom.

7.2.8 平台主页面-队伍

点击用户界面-队伍信息中的队伍 ID，即可进入每个队伍的信息界面。



The screenshot shows a user interface for managing team information. At the top, there is a navigation bar with the ACM logo and links for Home, Problem Set, Competition, and User. A user profile is shown on the right. The main content area has a dark background with the text "Chase Dreams!" in large white letters. Below it is a quote: "目标的坚定是性格中最必要的力量源泉之一，也是成功的利器之一。没有它，天才会在矛盾无定的迷径中徒劳无功。". There are two tabs: "队伍信息" (selected) and "队伍参加的比赛". The "队伍信息" tab contains fields for Team_id (000185), Team Name (宝可梦不醒), Member 1 (Gene_Liu), Member 2 (root), and Member 3 (无). At the bottom right are "返回" (Return) and "修改" (Modify) buttons.

该界面分为两部分，第一部分是队伍信息，可以进行队伍信息的修改，主要是修改两个队员的信息以及队伍名的信息。

在修改队员成员时，要求正确输入队员的账号，如果系统中没有这个用户，页面就会显示查无此人，并且修改失败。

The first screenshot shows a successful update of the team members. The Member 2 field now contains "123". The second screenshot shows an attempt to update Member 2 to "root", which fails because the user "root" does not exist in the system. The Member 2 field is highlighted in red with the error message "查无此人". Both screenshots include the same "返回" and "修改" buttons at the bottom right.

并且队员 1、2、3 不能出现相同的用户 ID，如果出现相同的用户 ID，页面依然会显示



查无此人，并且修改失败。

Team_id	000185
队伍名	宝可梦不醒
队员1	Gene_Liu
队员2	root
队员3	Gene_Liu

返回 修改

Team_id	000185
队伍名	宝可梦不醒
队员1	Gene_Liu
队员2	root
查无此人	
队员3	无

返回 修改

如果输入正确的用户 ID，系统就会显示修改成功的信息。此时其它两个用户就自动的加入了这个队伍。

Team_id	000185
队伍名	宝可梦不醒
队员1	Gene_Liu
队员2	yjq
队员3	zjm
修改成功	

返回 修改

登入其它两个用户的账号，可以看到他们都加入了这个队伍，并且队伍成员与刚才显示的队伍成员相同。而且用户自身全部成为了队员 1，因为队员 1 是不能进行修改的。



ACM 首页 题库 比赛 用户

bjy

Chase Dreams!

目标的坚定是性格中最必要的力量源泉之一，也是成功的利器之一。没有它，天才会在矛盾无定的迷径中徒劳无功。

队伍信息 队伍参加的比赛

Team_id	000185
队伍名	宝可梦不醒
队员1	yjq
队员2	Gene_Liu
队员3	zjm

返回 修改

ACM 首页 题库 比赛 用户

bj zjm

Chase Dreams!

目标的坚定是性格中最必要的力量源泉之一，也是成功的利器之一。没有它，天才会在矛盾无定的迷径中徒劳无功。

写过的题目 个人参加的比赛 队伍信息

创建队伍 退出队伍

Team_ID	队伍名	组建时间
000185	宝可梦不醒	2019-01-05

队伍界面的另一部分就是队伍参加过的比赛的信息。显示这个队伍参加过的所有比赛的比赛 ID、比赛排名、完成题数以及比赛中的总罚时。

点击比赛 ID，即可进入比赛的具体界面，可以查看比赛的具体信息，比如这个比赛包括了哪些题目。



The screenshot shows the 'Contestant Information' section of the platform. At the top, there is a navigation bar with 'ACM' and links for '首页', '题库', '比赛', and '用户'. A user profile 'Gene_Liu' is also visible. Below the navigation is a large banner with the text 'Chase Dreams!' and a quote: '目标的坚定是性格中最必要的力量源泉之一，也是成功的利器之一。没有它，天才会在矛盾无定的迷径中徒劳无功。' The main content area displays a table of contestant information:

Contest_ID	排名	完成题数	总罚时
000020	7	9	3 days, 10:21:00
000076	1	1	0:00:00

At the bottom right of the table are two buttons: '添加比赛' (Add Competition) and '删除比赛' (Delete Competition).

点击添加比赛的按钮，就会弹出添加比赛的界面。需要填写的信息分别是比赛 ID、完成题数、比赛中的排名、比赛的总罚时这些信息。全部采取下拉菜单的方式进行填写，避免用户不按规则输入信息。

并且此处的比赛 ID 的下拉框中不包含已经参加过的比赛 ID，避免添加已经参加过的比赛。

The screenshot shows the 'Add Competition' modal dialog. It contains fields for '比赛ID' (Competition ID) set to '000001', '完成题数' (Completed Questions) set to '1', '排名' (Rank) set to '1', '罚时 (时)' (Penalty Time (Hours)) set to '00', and '罚时 (分)' (Penalty Time (Minutes)) set to '00'. At the bottom right of the dialog are '取消' (Cancel) and '添加' (Add) buttons.

如果信息填写正确，页面就会显示添加比赛成功的信息。



比赛ID
000002

完成题数
1

排名
1

罚时 (时)
00

罚时 (分)
00

添加成功

取消 添加

点击删除比赛的按钮，就会弹出删除比赛的页面。

删除题目

比赛ID
000001

取消 删除

Contest_ID	排名	完成题数	总罚时
000001	1	1	0:00:00
000020	7	9	3 days, 10:21:00
000076	1	1	0:00:00

页面中提供一个比赛 ID 的下拉菜单进行选择，下拉菜单中只包含用户参加过的比赛的比赛 ID，避免用户删除自己没有参加过的比赛造成系统错误。



The screenshot shows a modal dialog titled "Delete Question". Inside the dialog, there is a dropdown menu labeled "Competition ID" containing three items: "000001", "000020", and "000076". The item "000001" has a checked mark next to it. At the bottom right of the dialog are two buttons: "Cancel" (red) and "Delete" (blue). The background of the main page shows a competition named "Chase Dr" with a brief description: "目标的坚定是性格中的一条铁律，但毅力并不一定能在生活的迷径中徒劳无功。". Below the dialog, there is a table with columns: Contest_ID, 排名 (Rank), 完成题数 (Completed Questions), and 总罚时 (Total Penalties). The table contains three rows of data.

Contest_ID	排名	完成题数	总罚时
000001	1	1	0:00:00
000020	7	9	3 days, 10:21:00
000076	1	1	0:00:00

点击删除按钮，页面就会显示删除成功的信息。

The screenshot shows the same modal dialog as before, but now the "Delete" button has been clicked. A confirmation message "删除成功" (Delete successful) is displayed at the bottom of the dialog. The background page remains the same, showing the competition details and the table of completed questions.

7.3 事务设计

该系统中查询事务都用 select 实现，更新/修改事务都用 update 实现，添加事务都用 insert 语句实现，删除事务都用 delete 实现。

7.3.1 用户注册事务

在该界面主要是用户信息的注册，主要是 insert 语句来实现，实现往数据库中插入新用户这一事务。

在用户注册界面还需要显示系统中的所有学校信息以及实验室信息，从数据库中提取这两部分信息用 select 语句实现。

```
class RegisterHandler(RequestHandler):  
    def post(self, *args, **kwargs):
```



```
stus1 = self.application.db.get_all_obj("select Name from School", "School",
                                         "Name")

stus2 = self.application.db.get_all_obj("select Name from Laboratory", "Laboratory",
                                         "Name")

Person_id = self.get_body_argument("Person_id")

# print(Person_id)

if Person_id == '':
    self.render('html/RgError.html', jud=1, School=stus1, Lab=stus2)
    return

Person_id = "" + Person_id + ""

stus = self.application.db.get_all_obj("select Person_id from Person where Person_id = " + Person_id, "Person",
                                         "Person_id")

print(len(stus))

if len(stus) != 0:
    self.render('html/RgError.html', jud=2, School=stus1, Lab=stus2)
    return

Name = self.get_body_argument("Name")
Gender = self.get_body_argument("Gender")
Major = self.get_body_argument("Major")
Grade = self.get_body_argument("Grade")
Class = self.get_body_argument("Class")
School = self.get_body_argument("School")

School = "" + School + ""

stus = self.application.db.get_all_obj("select School_id from School where Name = " + School, "School",
                                         "School_id")

School_id = stus[0]['School_id']

Lab = self.get_body_argument("Lab")
Lab = "" + Lab + ""

stus = self.application.db.get_all_obj("select Lab_id from Laboratory where Name = " + Lab, "Laboratory",
                                         "Lab_id")

Lab_id = stus[0]['Lab_id']

Serect = self.get_body_argument("Serect")

if Serect == '':
    self.render('html/RgError.html', jud=3, School=stus1, Lab=stus2)
    return

Name = "" + Name + ""
Major = "" + Major + ""
Grade = "" + Grade + ""
Gender = "" + Gender + ""
Class = "" + Class + ""
School_id = "" + School_id + ""
Lab_id = "" + Lab_id + ""
Serect = "" + Serect + ""
```



```
sql = "insert into Person(Person_id,Name,Gender,Major,Grade,Class,School_id,Lab_id,Serect) values (" + Person_id + "," + Name + "," + Gender + "," + Major + "," + Grade + "," + Class + "," + School_id + "," + Lab_id + "," + Serect + ")"
# print(sql)
self.application.db.update(sql)
self.render('html/RgError.html', jud=4, School=stus1, Lab=stus2)
```

7.3.2 用户登录事务

用户登录界面需要将用户的账号和密码进行比对，来判断用户输入的密码是否正确。

```
class LoginHandler(RequestHandler):
    def post(self, *args, **kwargs):
        stus1 = self.application.db.get_all_obj("select Name from School", "School",
                                               "Name")
        stus2 = self.application.db.get_all_obj("select Name from Laboratory", "Laboratory",
                                               "Name")
        Person_id = self.get_body_argument("InputAccount")
        Serect = self.get_body_argument("InputPassword")
        tmp = Person_id
        Person_id = "''' + Person_id + '''"
        stus = self.application.db.get_all_obj("select Serect from Person where Person_id = " + Person_id, "Person",
                                              "Serect")
        # print(type(stus))
        # print(stus)
        if len(stus) == 0:
            self.render('html/LogError.html', School=stus1, Lab=stus2)
        elif stus[0]['Serect'] == Serect:
            self.render('html/HomePage.html', Person_id=tmp)
        else:
            self.render('html/LogError.html', School=stus1, Lab=stus2)
```

7.3.3 题库中题目显示事务

题库界面需要将所有的题目显示出来，这里主要用 select 语句实现。此处先是对多个表格中的数据进行查询，然后替换，之后再一起显示出来。

```
stus = db.get_all_obj("select * from Problem", "Problem")
for line in stus:
    line['Algorithm'] = ""
    OJ_id = "''' + line['OJ_id'] + '''"
    now = db.get_all_obj("select Name from Online_Judge where OJ_id = " + OJ_id,
                         "Online_Judge", "Name")
    line['OJ_id'] = now[0]['Name']
    Problem_id = "''' + line['Problem_id'] + '''"
```



```
now = db.get_all_obj()  
  
    "select Algorithm_id from Problem_Algorithm where Problem_id = " + Problem_id, "Problem_Algorithm",  
    "Algorithm_id")  
  
for hp in now:  
  
    Algorithm_id = hp['Algorithm_id']  
  
    tp = db.get_all_obj("select Name from Algorithm where Algorithm_id = " + Algorithm_id,  
                        "Algorithm", "Name")  
  
    hp['Algorithm_name'] = tp[0]['Name']  
  
cnt = 0  
  
for hp in now:  
  
    if cnt != 0:  
  
        line['Algorithm'] = line['Algorithm'] + ","  
  
    line['Algorithm'] = line['Algorithm'] + hp['Algorithm_id']  
  
    cnt += 1
```

7.3.4 比赛界面中比赛显示事务

比赛界面需要将数据库中所有的比赛信息显示出来，这里采用 select 语句进行实现。

```
class ContestHandler(RequestHandler):  
  
    def get(self, *args, **kwargs):  
  
        stus = self.application.db.get_all_obj("select * from Contest", "Contest")  
  
        self.render('html/Contest.html', Person_id=self.get_argument('Person_id'), Contest=stus)
```

7.3.5 比赛界面中添加比赛的事务

添加比赛的界面向数据库中添加新的比赛信息，这里采用 insert 语句进行实现。并且此处需要对于添加比赛的一些规范进行检查，对于不符合比赛信息规范的添加数据，系统返回错误信息，不予添加。

```
class AddConErrorHandler(RequestHandler):  
  
    def get(self, *args, **kwargs):  
  
        pass  
  
  
    def post(self, *args, **kwargs):  
  
        Name = self.get_argument('Name')  
  
        if Name == "":  
  
            self.render('html/AddContest.html', Person_id=self.get_argument('Person_id'), jud='1')  
  
            return  
  
        Time = self.get_argument('Time')  
  
        if Time == "":  
  
            self.render('html/AddContest.html', Person_id=self.get_argument('Person_id'), jud='2')  
  
            return
```



```
Unit = self.get_argument('Unit')
Type = self.get_argument('Type')

num = self.application.db.get_all_obj("select count(*) from Contest", "Contest", "Num")

Contest_id = "%06d" % (num[0]['Num'] + 1)

Contest_id = "''' + str(Contest_id) + "''"

a = datetime.now()

Hour = 0

Min = 0

Sec = 0

if Unit == '秒':
    Sec = int(Time)

elif Unit == '分':
    Min = int(Time)

elif Unit == '时':
    Hour = int(Time)

b = timedelta(hours=Hour, minutes=Min, seconds=Sec)

End_time = (a + b).strftime("%Y-%m-%d %H:%M:%S")

Begin_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

Begin_time = "''' + str(Begin_time) + "''"

End_time = "''' + str(End_time) + "''"

# print(Begin_time)

# print(End_time)

Type = "''' + Type + "''"

Name = "''' + Name + "''"

sql = "insert into Contest values (" + Contest_id + "," + Name + "," + Begin_time + "," + End_time + "," + Type + ")"

# print(sql)

self.application.db.update(sql)

self.render('html/AddContest.html', Person_id=self.get_argument('Person_id'), jud='3')
```

7.3.6 单个比赛界面比赛信息的显示与添加

单个比赛界面中比赛信息的显示，主要是使用 select 对于多个表格进行数据查询。需要查询比赛自身带有的信息，还需要查询比赛中所包括的题目的信息。

比赛信息的添加主要指向比赛中添加题目信息，增加比赛中的题目。

```
class ConInformationHandler(RequestHandler):

    def get(self, *args, **kwargs):
        tmp = "''' + self.get_argument('Contest_id') + "''"

        sql = "select * from Contest where Contest_id = " + tmp

        stus = self.application.db.get_all_obj(sql, "Contest")

        now = "''' + stus[0]['Contest_id'] + "'''"

        tmp = self.application.db.get_all_obj("select Problem_id from Contest_Problem where Contest_id = " + now,
```



```
"Contest_Problem", "Problem_id")\n\nPro = []\n\nfor line in tmp:\n    Pro.append(line['Problem_id'])\n\ncontent = []\n\nwith open('views/data/ans.txt', 'r') as f:\n    for line in f:\n        content.append(line.split('\\t'))\n\nProblem = []\n\nfor line in content:\n    if line[0] not in Pro:\n        continue\n\n    tmp = {}\n\n    tmp['Problem_id'] = line[0]\n    tmp['Problem_name'] = line[1]\n    tmp['OJ'] = line[2]\n    tmp['OJ_Pnumber'] = line[3]\n    tmp['Difficulty'] = line[4]\n    tmp['Link'] = line[5]\n    tmp['Algorithm'] = line[6].strip('\\n')\n\n    Problem.append(tmp)\n\n# print("Problem", Problem)\n\nLength = 15 * (15 - len(Problem))\n\nif Length < 0:\n    Length = 0\n\nself.render('html/ConInformation.html', Person_id=self.get_argument('Person_id'), stus=stus, Problem=Problem,\n           Length=Length, ProLen=len(content), jud='0')\n\n\n\ndef post(self, *args, **kwargs):\n    tmp = """ + self.get_argument('Contest_id') + """\n    sql = "select * from Contest where Contest_id = " + tmp\n    stus = self.application.db.get_all_obj(sql, \"Contest\")\n    now = """ + stus[0]['Contest_id'] + """\n    tmp = self.application.db.get_all_obj(\"select Problem_id from Contest_Problem where Contest_id = \" + now,\n                                         \"Contest_Problem\", \"Problem_id\")\n    AddProblem_id = self.get_body_argument('ConProblemID')\n    thp2 = """ + AddProblem_id + """\n    jud = '2'\n\n    for line in tmp:\n        if AddProblem_id == line['Problem_id']:\n            jud = '1'\n            break\n\n        if jud == '2':\n            self.application.db.get_all_obj(\"update Contest_Problem set Problem_id = \" + thp2 + \" where Problem_id = \" + line['Problem_id'],\n                                         \"Contest_Problem\", \"Problem_id\")\n\n    self.render('html/ConInformation.html', Person_id=self.get_argument('Person_id'), stus=stus, Problem=Problem,\n               Length=Length, ProLen=len(content), jud='1')
```



```
thp1 = """ + self.get_argument('Contest_id') + """
sql = "insert into Contest_Problem values (" + thp1 + "," + thp2 + ")"
self.application.db.update(sql)

tmp = self.application.db.get_all_obj("select Problem_id from Contest_Problem where Contest_id = " + now,
                                         "Contest_Problem", "Problem_id")

Pro = []
for line in tmp:
    Pro.append(line['Problem_id'])

content = []
with open('views/data/ans.txt', 'r') as f:
    for line in f:
        content.append(line.split('\t'))

Problem = []
for line in content:
    if line[0] not in Pro:
        continue
    tmp = {}
    tmp['Problem_id'] = line[0]
    tmp['Problem_name'] = line[1]
    tmp['OJ'] = line[2]
    tmp['OJ_Pnumber'] = line[3]
    tmp['Difficulty'] = line[4]
    tmp['Link'] = line[5]
    tmp['Algorithm'] = line[6].strip('\n')
    Problem.append(tmp)

Length = 15 * (15 - len(Problem))

if Length < 0:
    Length = 0

self.render('html/ConInformation.html', Person_id=self.get_argument('Person_id'), stus=stus, Problem=Problem,
           Length=Length, ProLen=len(content), jud=jud)
```

7.3.7 用户个人界面个人信息的显示与修改

用户个人界面的个人信息主要是基本信息和密码，基本信息和密码都支持修改与更新，主要采用 update 语句进行实现。此处基本信息的修改还会出现很多种不规范数据输入的情况，系统都需要对这些不规范情况进行识别，返回错误信息。

个人信息的修改包括对于学校以及实验室的选择，此时需要查询数据库中所包含的学校、实验室信息用于页面显示。



```
class PerInformationHandler(RequestHandler):

    def get(self, *args, **kwargs):
        stus1 = self.application.db.get_all_obj("select Name from School", "School",
                                                "Name")

        stus2 = self.application.db.get_all_obj("select Name from Laboratory", "Laboratory",
                                                "Name")

        tmp = self.get_argument('Person_id')

        tmp = "" + tmp + ""

        stus3 = self.application.db.get_all_obj("select * from Person where Person_id = " + tmp, "Person")

        for line in stus3:
            School_id = "" + line['School_id'] + ""
            now = self.application.db.get_all_obj("select Name from School where School_id = " + School_id, "School",
                                                "Name")
            line['School_id'] = now[0]['Name']

            Lab_id = "" + line['Lab_id'] + ""
            now = self.application.db.get_all_obj("select Name from Laboratory where Lab_id = " + Lab_id, "Laboratory",
                                                "Name")
            line['Lab_id'] = now[0]['Name']

        self.render('html/PerInformation.html', Person_id=self.get_argument('Person_id'), School=stus1, Lab=stus2,
                   Person=stus3[0])

    def post(self, *args, **kwargs):
        Person_id = self.get_argument('Person_id')

        tmp = "" + Person_id + ""

        stus = self.application.db.get_all_obj("select * from Person where Person_id = " + tmp, "Person")

        Name = self.get_body_argument("Name")
        Gender = self.get_body_argument("Gender")
        Major = self.get_body_argument("Major")
        School = self.get_body_argument("School")
        Grade = self.get_body_argument("Grade")
        Class = self.get_body_argument("Class")
        Lab = self.get_body_argument("Lab")

        if Name == "":
            Name = stus[0]['Name']

        if Gender == "":
            Gender = stus[0]['Gender']

        if School == "":
            School = stus[0]['School']

        if Major == "":
            Major = stus[0]['Major']

        if Grade == "":
            Grade = stus[0]['Grade']

        if Class == "":
            Class = stus[0]['Class']
```



```
Class = stus[0]['Class']

if Lab == "":
    Lab = stus[0]['Lab']

School = "" + School + ""

now = self.application.db.get_all_obj("select School_id from School where Name = " + School, "School",
                                         "School_id")

School_id = now[0]['School_id']

Lab = "" + Lab + ""

now = self.application.db.get_all_obj("select Lab_id from Laboratory where Name = " + Lab, "Laboratory",
                                         "Lab_id")

Lab_id = now[0]['Lab_id']

Person_id = "" + Person_id + ""

Name = "" + Name + ""

Gender = "" + Gender + ""

Major = "" + Major + ""

Grade = "" + Grade + ""

Class = "" + Class + ""

School_id = "" + School_id + ""

Lab_id = "" + Lab_id + ""

sql = "update Person set Name = " + Name + ",Gender = " + Gender + ",Major = " + Major + ",Grade = " + Grade + ",Class = "
      + Class + ",School_id = " + School_id + ",Lab_id = " + Lab_id + " where Person_id = " + Person_id

self.application.db.update(sql)

stus1 = self.application.db.get_all_obj("select Name from School", "School",
                                         "Name")

stus2 = self.application.db.get_all_obj("select Name from Laboratory", "Laboratory",
                                         "Name")

tmp = self.get_argument('Person_id')

tmp = "" + tmp + ""

stus3 = self.application.db.get_all_obj("select * from Person where Person_id = " + tmp, "Person")

for line in stus3:

    School_id = "" + line['School_id'] + ""

    now = self.application.db.get_all_obj("select Name from School where School_id = " + School_id, "School",
                                         "Name")

    line['School_id'] = now[0]['Name']

    Lab_id = "" + line['Lab_id'] + ""

    now = self.application.db.get_all_obj("select Name from Laboratory where Lab_id = " + Lab_id, "Laboratory",
                                         "Name")

    line['Lab_id'] = now[0]['Name']

self.render('html/PerInformation.html', Person_id=self.get_argument('Person_id'), School=stus1, Lab=stus2,
           Person=stus3[0])
```

用户密码的修改与基本信息的修改用两个类进行了分开处理，因为二者从前端传递的参



数不同，需要进行分开处理。

```
class ChangeSerectHandler(RequestHandler):

    def get(self, *args, **kwargs):
        pass

    def post(self, *args, **kwargs):
        Person_id = self.get_argument('Person_id')
        tmp = "'' + Person_id + """
        stus = self.application.db.get_all_obj("select * from Person where Person_id = " + tmp, "Person")
        OldSerect = self.get_body_argument('OldSerect')
        NewSerect = self.get_body_argument('NewSerect')
        ReNewSerect = self.get_body_argument('ReNewSerect')
        jud = '1'
        if OldSerect != stus[0]['Serect']:
            jud = '1'
        elif NewSerect == "":
            jud = '2'
        elif NewSerect != ReNewSerect:
            jud = '3'
        else:
            self.application.db.update("update Person set Serect = "+""+NewSerect+"+" where Person_id = "+ tmp)
            jud = '4'
        stus1 = self.application.db.get_all_obj("select Name from School", "School",
                                              "Name")
        stus2 = self.application.db.get_all_obj("select Name from Laboratory", "Laboratory",
                                              "Name")
        tmp = self.get_argument('Person_id')
        tmp = "'' + tmp + """
        stus3 = self.application.db.get_all_obj("select * from Person where Person_id = " + tmp, "Person")
        for line in stus3:
            School_id = "'' + line['School_id'] + """
            now = self.application.db.get_all_obj("select Name from School where School_id = " + School_id, "School",
                                                 "Name")
            line['School_id'] = now[0]['Name']
            Lab_id = "'' + line['Lab_id'] + """
            now = self.application.db.get_all_obj("select Name from Laboratory where Lab_id = " + Lab_id, "Laboratory",
                                                 "Name")
            line['Lab_id'] = now[0]['Name']
        self.render('html/SerectError.html', Person_id=self.get_argument('Person_id'), School=stus1, Lab=stus2,
                   Person=stus3[0], jud=jud)
```

7.3.8 用户个人训练记录的信息显示、添加与删除



用户个人的训练记录主要包括刷题记录、参加比赛记录以及加入队伍的记录。

用户个人刷题记录的页面主要是题目的显示，以及添加用户完成过的题目和删除用户添加过的题目信息。显示题目用 select 语句实现，添加新的题目信息用 update 语句进行实现，删除已有的题目信息用 delete 语句进行实现。

用户个人参加比赛记录的页面主要是比赛的显示，以及添加用户参加过的比赛信息以及删除用户曾经参加过的比赛的信息。显示比赛信息用 select 语句实现，添加用户参加过的信息用 insert 语句进行实现，删除用户曾经参加过的比赛的信息用 delete 语句进行实现。

用户队伍信息的页面主要是用户参加过的队伍的信息显示，以及创建一个新的队伍和退出用户之前参加过的队伍。显示用户之前参加过的队伍的信息主要用 select 语句对多个表进行查询实现，创建一个新的队伍使用 insert 语句实现，退出队伍使用 delete 语句实现。

这里多个事务全部使用同一个类进行实现，采用分支语句的方式对不同的事务进行不同的处理。

```
class UserHandler(RequestHandler):
    def get(self, *args, **kwargs):
        jud = '-1'
        tmp = "'' + self.get_argument('Person_id') + ''
        Type = self.get_argument('Type')
        if Type == '-2':
            jud = '-2'
        if Type == '4':
            tmpTeam = self.application.db.get_all_obj("select count(*) from Team", "Team", "Num")
            TeamNum = "%06d" % int(tmpTeam[0]['Num']) + 1
            TeamNum1 = "'' + str(TeamNum) + ''
            sql = "insert into Team(Team_id, Name) values (" + TeamNum1 + ", " + '无' + ")"
            self.application.db.update(sql)
            print("sql1", sql)
            Today = datetime.now().strftime("%Y-%m-%d")
            TodayTime = "'' + str(Today) + ''
            sql = "insert into Person_Team values (" + tmp + ", " + TeamNum1 + ", " + TodayTime + ", " + '无' + ")"
            self.application.db.update(sql)
            print("sql2", sql)
```



```
print("sql2", sql)
self.application.db.update(sql)
jud = '3'

sql = "select * from Person_Finish_Problem where Person_id = " + tmp
stus = self.application.db.get_all_obj(sql, "Person_Finish_Problem")
ProLen = 0

with open('views/data/ans.txt', 'r') as f:
    for line in f:
        ProLen += 1
        line = line.split('\t')
        for i in stus:
            if line[0] == i['Problem_id']:
                i['Problem_name'] = line[1]
                i['OJ'] = line[2]
                i['OJ_Pnumber'] = line[3]
                i['Difficulty'] = line[4]
                i['Link'] = line[5]
                i['Algorithm'] = line[6].strip('\n')
                break
sql = "select * from Contest_Person where Person_id = " + tmp
Contest = self.application.db.get_all_obj(sql, "Contest_Person")
ConLen = self.application.db.get_all_obj("select count(*) from Contest", "Contest", "Len")

sql = "select * from Person_Team where Person_id = " + tmp
Team = self.application.db.get_all_obj(sql, "Person_Team")

for line in Team:
    thp = "" + line['Team_id'] + ""
    tmpName = self.application.db.get_all_obj("select Name from Team where Team_id = " + thp, "Team", "Name")
    line['Name'] = tmpName[0]['Name']

self.render('html/User.html', Person_id=self.get_argument('Person_id'), Problem=stus,
           Length=(15 * (15 - len(stus))), ProLen=ProLen, jud=jud, Contest=Contest,
           Length2=(15 * (15 - len(Contest))), ConLen=ConLen[0]['Len'], Team=Team,
           Length3=(15 * (15 - len(Team)))))

def post(self, *args, **kwargs):
    Type = self.get_argument('Type')

    if Type == '1':
        Problem_id = self.get_body_argument('UserAddProblemID')
        jud = '0'
        tmp = "" + self.get_argument('Person_id') + ""
        sql = "select * from Person_Finish_Problem where Person_id = " + tmp
        stus = self.application.db.get_all_obj(sql, "Person_Finish_Problem")
        for i in stus:
```



```
if Problem_id == i['Problem_id']:
    jud = '1'

if jud == '0':
    time = "" + str(datetime.now().strftime("%Y-%m-%d %H:%M:%S")) + ""
    a1 = "" + Problem_id + ""

    sql = "insert into Person_Finish_Problem values (" + tmp + "," + a1 + "," + time + ")"
    self.application.db.update(sql)

    sql = "select * from Person_Finish_Problem where Person_id = " + tmp
    stus = self.application.db.get_all_obj(sql, "Person_Finish_Problem")

ProLen = 0
with open('views/data/ans.txt', 'r') as f:
    for line in f:
        ProLen += 1
        line = line.split('\t')
        for i in stus:
            if line[0] == i['Problem_id']:
                i['Problem_name'] = line[1]
                i['OJ'] = line[2]
                i['OJ_Pnumber'] = line[3]
                i['Difficulty'] = line[4]
                i['Link'] = line[5]
                i['Algorithm'] = line[6].strip('\n')
                break
        sql = "select * from Contest_Person where Person_id = " + tmp
        Contest = self.application.db.get_all_obj(sql, "Contest_Person")
        ConLen = self.application.db.get_all_obj("select count(*) from Contest", "Contest", "Len")

        sql = "select * from Person_Team where Person_id = " + tmp
        Team = self.application.db.get_all_obj(sql, "Person_Team")
        for line in Team:
            thp = "" + line['Team_id'] + ""
            tmpName = self.application.db.get_all_obj("select Name from Team where Team_id = " + thp, "Team",
                                                       "Name")
            line['Name'] = tmpName[0]['Name']
            self.render('html/User.html', Person_id=self.get_argument('Person_id'), Problem=stus,
                       Length=(15 * (15 - len(stus))), ProLen=ProLen, jud=jud, Contest=Contest,
                       Length2=(15 * (15 - len(Contest))), ConLen=ConLen[0]['Len'], Team=Team,
                       Length3=(15 * (15 - len(Team))))
elif Type == '11':
    a1 = "" + self.get_argument('Person_id') + ""
    a2 = "" + self.get_body_argument('UserDeleteProblemID') + ""

    sql = "delete from Person_Finish_Problem where Person_id = " + a1 + " and Problem_id = " + a2
```



```
self.application.db.delete(sql)

tmp = "" + self.get_argument('Person_id') + ""

sql = "select * from Person_Finish_Problem where Person_id = " + tmp

stus = self.application.db.get_all_obj(sql, "Person_Finish_Problem")

ProLen = 0

with open('views/data/ans.txt', 'r') as f:

    for line in f:

        ProLen += 1

        line = line.split('\t')

        for i in stus:

            if line[0] == i['Problem_id']:

                i['Problem_name'] = line[1]

                i['OJ'] = line[2]

                i['OJ_Pnumber'] = line[3]

                i['Difficulty'] = line[4]

                i['Link'] = line[5]

                i['Algorithm'] = line[6].strip('\n')

                break

sql = "select * from Contest_Person where Person_id = " + tmp

Contest = self.application.db.get_all_obj(sql, "Contest_Person")

ConLen = self.application.db.get_all_obj("select count(*) from Contest", "Contest", "Len")



sql = "select * from Person_Team where Person_id = " + tmp

Team = self.application.db.get_all_obj(sql, "Person_Team")

for line in Team:

    thp = "" + line['Team_id'] + ""

    tmpName = self.application.db.get_all_obj("select Name from Team where Team_id = " + thp, "Team",

                                              "Name")

    line['Name'] = tmpName[0]['Name']

    self.render('html/User.html', Person_id=self.get_argument('Person_id'), Problem=stus,

               Length=(15 * (15 - len(stus))), ProLen=ProLen, jud='11', Contest=Contest,

               Length2=(15 * (15 - len(Contest))), ConLen=ConLen[0]['Len'], Team=Team,

               Length3=(15 * (15 - len(Team)))))

elif Type == '2' or Type == '21':

    tmp = "" + self.get_argument('Person_id') + ""

    jud = '-1'

    if Type == '2':

        Contest_id = "" + self.get_body_argument('UserAddConID') + ""

        Rank = "" + self.get_body_argument('UserAddConRank') + ""

        Time_total = "" + self.get_body_argument('UserAddConTimeH') + ":" + self.get_body_argument(

            'UserAddConTimeM') + ":00"

        Finished_Num = "" + self.get_body_argument('UserAddConNum') + ""

        sql = "insert into Contest_Person values (" + Contest_id + "," + tmp + "," + Rank + "," + Finished_Num + "," + Time_total +")"
```



```
+ "," + "'NULL'" + ")"

    # print(sql)
    jud = '2'
    self.application.db.update(sql)

elif Type == '21':

    ConID = "" + self.get_body_argument('UserDeleteConID') + ""

    sql = "delete from Contest_Person where Person_id = " + tmp + " and Contest_id = " + ConID
    self.application.db.delete(sql)
    jud = '21'

    sql = "select * from Person_Finish_Problem where Person_id = " + tmp
    stus = self.application.db.get_all_obj(sql, "Person_Finish_Problem")

    ProLen = 0

    with open('views/data/ans.txt', 'r') as f:

        for line in f:
            ProLen += 1
            line = line.split('\t')
            for i in stus:
                if line[0] == i['Problem_id']:
                    i['Problem_name'] = line[1]
                    i['OJ'] = line[2]
                    i['OJ_Pnumber'] = line[3]
                    i['Difficulty'] = line[4]
                    i['Link'] = line[5]
                    i['Algorithm'] = line[6].strip('\n')
                    break
            sql = "select * from Contest_Person where Person_id = " + tmp
            Contest = self.application.db.get_all_obj(sql, "Contest_Person")
            ConLen = self.application.db.get_all_obj("select count(*) from Contest", "Contest", "Len")

            sql = "select * from Person_Team where Person_id = " + tmp
            Team = self.application.db.get_all_obj(sql, "Person_Team")

            for line in Team:
                thp = "" + line['Team_id'] + ""
                tmpName = self.application.db.get_all_obj("select Name from Team where Team_id = " + thp, "Team",
                                                          "Name")
                line['Name'] = tmpName[0]['Name']
                self.render('html/User.html', Person_id=self.get_argument('Person_id'), Problem=stus,
                           Length=(15 * (15 - len(stus))), ProLen=ProLen, jud=jud, Contest=Contest,
                           Length2=(15 * (15 - len(Contest))), ConLen=ConLen[0]['Len'], Team=Team,
                           Length3=(15 * (15 - len(Team))))
    elif Type == '41':
        tmp = "" + self.get_argument('Person_id') + ""
        TeamID = "" + self.get_body_argument('UserDeleteTeamID') + ""
```



```
sql = "delete from Person_Team where Person_id = " + tmp + " and Team_id = " + TeamID
self.application.db.delete(sql)
jud = '31'

sql = "select * from Person_Finish_Problem where Person_id = " + tmp
stus = self.application.db.get_all_obj(sql, "Person_Finish_Problem")
ProLen = 0
with open('views/data/ans.txt', 'r') as f:
    for line in f:
        ProLen += 1
        line = line.split('\t')
        for i in stus:
            if line[0] == i['Problem_id']:
                i['Problem_name'] = line[1]
                i['OJ'] = line[2]
                i['OJ_Pnumber'] = line[3]
                i['Difficulty'] = line[4]
                i['Link'] = line[5]
                i['Algorithm'] = line[6].strip('\n')
                break
sql = "select * from Contest_Person where Person_id = " + tmp
Contest = self.application.db.get_all_obj(sql, "Contest_Person")
ConLen = self.application.db.get_all_obj("select count(*) from Contest", "Contest", "Len")

sql = "select * from Person_Team where Person_id = " + tmp
Team = self.application.db.get_all_obj(sql, "Person_Team")
for line in Team:
    thp = "" + line['Team_id'] + ""
    tmpName = self.application.db.get_all_obj("select Name from Team where Team_id = " + thp, "Team",
                                              "Name")
    line['Name'] = tmpName[0]['Name']
self.render('html/User.html', Person_id=self.get_argument('Person_id'), Problem=stus,
           Length=(15 * (15 - len(stus))), ProLen=ProLen, jud=jud, Contest=Contest,
           Length2=(15 * (15 - len(Contest))), ConLen=ConLen[0]['Len'], Team=Team,
           Length3=(15 * (15 - len(Team))))
```

7.3.9 队伍基本信息与训练信息的显示、修改、添加与删除

队伍界面主要包括队伍的基本信息的显示和队伍训练信息的显示两个部分。在队伍基本信息显示的界面中支持对于队伍的基本信息，比如队伍名、队伍中的成员等信息进行修改，显示使用 select 语句实现，修改队伍名使用 update 语句实现，修改队伍成员使用 insert



语句和 delete 语句共同实现，因为队伍成员的变化意味着有人退出了队伍与有人加入了队伍。而且此处还需要对于输入数据的合法性进行验证，比如新增的队伍成员是否存在，即该队员信息是否存在于数据库中，此处主要用 select 语句进行验证，如果存在则进入下一个进程，如果不存在，则页面返回错误信息。

队伍训练信息的页面主要包括队伍参加过的所有比赛的信息显示、添加队伍曾经参加过比赛、删除队伍曾经参加过的比赛这三个事务。显示队伍曾经参加过的比赛这一事务使用 select 语句进行实现，添加队伍曾经参加过的比赛使用 insert 语句进行实现，删除队伍曾经参加过的比赛使用 delete 语句进行实现。

```
class TeamInformationHandler(RequestHandler):
    def get(self, *args, **kwargs):
        Person_id = self.get_argument('Person_id')
        Team_id = "''"+self.get_argument('Team_id')+"''"
        Team = self.application.db.get_all_obj("select * from Team where Team_id = "+Team_id, "Team")
        Person_Team = self.application.db.get_all_obj("select * from Person_Team where Team_id = "+Team_id, "Person_Team")
        Person = []
        Person.append(Person_id)
        for line in Person_Team:
            if line['Person_id'] not in Person:
                Person.append(line['Person_id'])
        Person.append("无")
        Person.append("无")
        Contest = self.application.db.get_all_obj("select * from Contest_Team where Team_id = "+Team_id, "Contest_Team")
        ConLen = self.application.db.get_all_obj("select count(*) from Contest", "Contest", "Len")
        self.render('html/TeamInformation.html', Person_id=Person_id, Team=Team, Person=Person, jud='1', Contest=Contest,
        ConLen=ConLen[0]['Len'])

    def post(self, *args, **kwargs):
        Person_id = self.get_argument('Person_id')
        Team_id = "''" + self.get_argument('Team_id') + "''"
        Type = self.get_argument('Type')
        Today = datetime.now().strftime("%Y-%m-%d")
        TodayTime = "''" + str(Today) + "''"
        jud = '-1'
        if Type == '3':
            ConID = "''" + self.get_body_argument('TeamDeleteConID') + "''"
```



```
sql = "delete from Contest_Team where Team_id = " + Team_id + " and Contest_id = " + ConID
self.application.db.delete(sql)
jud = '5'

elif Type == '2':
    Contest_id1 = "" + self.get_body_argument('TeamAddConID') + ""
    Rank = "" + self.get_body_argument('TeamAddConRank') + ""
    Time_total = "" + self.get_body_argument('TeamAddConTimeH') + ":" + self.get_body_argument(
        'TeamAddConTimeM') + ":00"
    Finished_Num = "" + self.get_body_argument('TeamAddConNum') + ""

    sql = "insert into Contest_Team values (" + Contest_id1 + "," + Team_id + "," + Rank + "," + Finished_Num + "," + Time_total +
+ "," + "NULL" + ")"
    # print(sql)
    jud = '4'
    self.application.db.update(sql)

elif Type == '1':
    TeamName = self.get_body_argument('TeamName')
    TeamOldPerson2 = self.get_argument('Person2')
    TeamOldPerson3 = self.get_argument('Person3')
    TeamPerson2 = self.get_body_argument('TeamPerson2')
    TeamPerson3 = self.get_body_argument('TeamPerson3')

    if TeamName != "":
        TeamName = ""+TeamName+"""

        self.application.db.update("update Team set Name = "+TeamName + "where Team_id = "+Team_id)

    tmpPerson = self.application.db.get_all_obj("select Person_id from Person", "Person", "Person_id")
    print("tmpPerson", tmpPerson)

    if TeamPerson2 == "" and TeamPerson3 != "":
        if TeamPerson3 == self.get_argument('Person_id') or TeamPerson3 == TeamOldPerson2 or TeamPerson3 == TeamOldPerson3:
            if TeamPerson3 not in [h['Person_id'] for h in tmpPerson]:
                jud = '2'
            else:
                TeamOldPerson3 = ""+TeamOldPerson3+"""

                TeamPerson3 = ""+TeamPerson3+"""

                self.application.db.delete("delete from Person_Team where Person_id = " + TeamOldPerson3 + " and Team_id = " +
Team_id)

                self.application.db.update("insert into Person_Team values( " + TeamPerson3 + "," + Team_id + "," + TodayTime +
+ "," + "无"+")")
                jud = '3'
        elif TeamPerson3 == "" and TeamPerson2 != "":
            if TeamPerson2 == self.get_argument('Person_id') or TeamPerson2 == TeamOldPerson2 or TeamPerson2 == TeamOldPerson3:
                if TeamPerson2 not in [h['Person_id'] for h in tmpPerson]:
                    jud = '1'
                else:
                    TeamOldPerson2 = ""+TeamOldPerson2+"""

                    TeamPerson2 = ""+TeamPerson2+"""

                    self.application.db.delete("delete from Person_Team where Person_id = " + TeamOldPerson2 + " and Team_id = " +
Team_id)

                    self.application.db.update("insert into Person_Team values( " + TeamPerson2 + "," + Team_id + "," + TodayTime +
+ "," + "无"+")")
                    jud = '2'
            else:
                TeamOldPerson2 = ""+TeamOldPerson2+"""

                TeamPerson2 = ""+TeamPerson2+"""

                self.application.db.delete("delete from Person_Team where Person_id = " + TeamOldPerson2 + " and Team_id = " +
Team_id)

                self.application.db.update("insert into Person_Team values( " + TeamPerson2 + "," + Team_id + "," + TodayTime +
+ "," + "无"+")")
                jud = '3'
```



```
TeamPerson2 = "''"+TeamPerson2+"''"

self.application.db.delete("delete from Person_Team where Person_id = " + TeamOldPerson2 + " and Team_id = " + Team_id)

self.application.db.update("insert into Person_Team values( " + TeamPerson2 + "," + Team_id + "," + TodayTime +
+ "," + "无"+")")

jud = '3'

elif TeamPerson3 != "" and TeamPerson2 != "":
    if TeamPerson2 == self.get_argument('Person_id') or TeamPerson2 == TeamOldPerson2 or TeamPerson2 == TeamOldPerson3 or (TeamPerson2 not in [h['Person_id'] for h in tmpPerson]):
        jud = '1'
    elif TeamPerson3 == self.get_argument('Person_id') or TeamPerson3 == TeamOldPerson2 or TeamPerson3 == TeamOldPerson3 or (TeamPerson3 not in [h['Person_id'] for h in tmpPerson]):
        jud = '2'
    else:
        TeamOldPerson3 = "''"+TeamOldPerson3+"''"
        TeamPerson3 = "''"+TeamPerson3+"''"
        self.application.db.delete(
            "delete from Person_Team where Person_id = " + TeamOldPerson3 + " and Team_id = " + Team_id)
        self.application.db.update(
            "insert into Person_Team values( " + TeamPerson3 + "," + Team_id + "," + TodayTime + "," + "无"+")")
        TeamOldPerson2 = "''"+TeamOldPerson2+"''"
        TeamPerson2 = "''"+TeamPerson2+"''"
        self.application.db.delete(
            "delete from Person_Team where Person_id = " + TeamOldPerson2 + " and Team_id = " + Team_id)
        self.application.db.update(
            "insert into Person_Team values( " + TeamPerson2 + "," + Team_id + "," + TodayTime + "," + "无"+")")
        jud = '3'

Team = self.application.db.get_all_obj("select * from Team where Team_id = " + Team_id, "Team")
Person_Team = self.application.db.get_all_obj("select * from Person_Team where Team_id = " + Team_id,
                                             "Person_Team")

Person = []
Person.append(Person_id)
for line in Person_Team:
    if line['Person_id'] not in Person:
        Person.append(line['Person_id'])
Person.append("无")
Person.append("无")

Contest = self.application.db.get_all_obj("select * from Contest_Team where Team_id = "+ Team_id, "Contest_Team")
ConLen = self.application.db.get_all_obj("select count(*) from Contest", "Contest", "Len")
self.render('html/TeamInformation.html', Person_id=Person_id, Team=Team, Person=Person, jud=jud, Contest=Contest,
           ConLen=ConLen[0]['Len'])
```



八、测试和运行

1. 初始界面

进入平台的初始界面，可以进行注册和登录。



我们注册一个临时账号，具体内容如下。

注册

账号	临时账号v
姓名	临时
性别	男
学校	山东大学
入学年份	2018

如果我们不输入账号名和密码的话，系统会返回错误。或者账号名已经被注册过的话，系统会显示账号已经被注册。



账号

账号不能为空

账号

账号已被注册

密码

密码不能为空

正确输入注册信息之后，系统界面显示注册成功的信息。

注册

注册成功

账号

姓名

性别

学校

进入登录界面，登录我们刚才注册的账号。



如果账号和密码不匹配，系统会显示账号或密码错误的信息。如果密码输入正确，那我



们就进入到了系统的主页面。



2. 主页面-首页

进入平台的主页面，看到 ICPC 图片以及系统推荐的题目和比赛，以及右上角可以看到登入的用户账号。测试没有问题。

3. 主页面-用户基本信息

点击右上角登入账号，即可进入我的资料进行查看。



进入个人信息界面，可以进行个人信息的修改，基本信息与注册界面一致，按照要求修改即可。

The screenshot shows the ACM training management platform's personal information modification page. At the top, there is a navigation bar with 'ACM' and links for '首页', '题库', '比赛', and '用户'. A dropdown menu for '临时账号' is also present. The main content area features a dark background with the text 'Chase Dreams!' and a quote: '目标的坚定是性格中最必要的力量源泉之一，也是成功的利器之一。没有它，天才会在矛盾无定的迷径中徒劳无功。'. Below this, there are two tabs: '个人信息' (selected) and '修改密码'. The '个人信息' section contains fields for '姓名' (Name), '性别' (Gender), '学校' (School), '年级' (Grade), and '专业' (Major). Each field has a placeholder value: '临时' (Temporary) for Name, '男' (Male) for Gender, '山东大学' (Shandong University) for School, '2018' for Grade, and '计算机科学与技术' (Computer Science and Technology) for Major.

进入修改密码界面。此处需要输入原密码和新密码以及确认密码。

The screenshot shows the password modification interface. It has two tabs: '个人信息' (selected) and '修改密码'. The '修改密码' section contains three input fields: '原密码' (Old Password), '新密码' (New Password), and '确认' (Confirm). Each field has a placeholder value: '原密码' for Old Password, '新密码' for New Password, and '再次确认' (Re-enter) for Confirmation. At the bottom right are '返回' (Back) and '修改' (Modify) buttons.

我们将原密码错误输入，系统会返回原密码错误的信息。除此之外还有新密码不得为空、新密码和确认密码不一致等错误，经测试均能正常显示，在页面设计中已经展示过了，此处不再赘述。

The screenshot shows the password modification interface with an error message. The '原密码' (Old Password) field contains the placeholder '原密码'. The '新密码' (New Password) and '确认' (Confirm) fields are empty. Below the fields, a red error message '原密码错误' (Incorrect Old Password) is displayed. At the bottom right are '返回' (Back) and '修改' (Modify) buttons.



4. 主页面-题库

进入题库界面。

The screenshot shows the ACM Problem库 page. At the top, there is a navigation bar with links for 首页 (Home), 题库 (Problem库), 比赛 (Contest), and 用户 (User). A user account icon with the text "临时账号" is also present. The main content area features a large banner with the text "Chase Dreams!" and a quote: "目标的坚定是性格中最必要的力量源泉之一，也是成功的利器之一。没有它，天才会在矛盾无定的迷径中徒劳无功。". Below the banner, there are two tabs: "题目集合" (Collection of Problems) and "算法分类" (Algorithm Categories). There are three buttons at the bottom of this section: "Show 20 rows", "导出Excel", and "打印表格". A search input field labeled "搜索:" is also present. The main table displays problem details with columns: ID, OJ & Number, 标题 (Title), 难度 (Difficulty), and 算法 (Algorithm). The data in the table is as follows:

ID	OJ & Number	标题	难度	算法
0000001	HDU 2955	Robberies	入门	背包
0000002	HDU 1864	最大报销额	大师	背包
0000003	HDU 2802	Bone Collector	入门	背包

可以对题目进行搜索，比如查询 POJ 入门 线段树的信息。

The screenshot shows the same ACM Problem库 page after a search. The search input field now contains "POJ 入门 线段树". The results table shows problems related to this search query, with the following data:

ID	OJ & Number	标题	难度	算法
0000176	POJ 3468	A SimpleProblem with Integers	入门	线段树
0000177	POJ 2528	Mayor'sposters	入门	线段树
0000178	POJ 1436Horizontally	Visible Segments	入门	线段树
0000193	POJ 1177	Picture	入门	线段树

具体细节在页面设计中都展示过了，此处不再赘述，主要是测试系统的稳定性。

5. 主页面-比赛

进入比赛界面。

The screenshot shows the ACM 比赛库 page. The main content area features a large banner with the text "Chase Dreams!" and a quote: "目标的坚定是性格中最必要的力量源泉之一，也是成功的利器之一。没有它，天才会在矛盾无定的迷径中徒劳无功。". Below the banner, there are two tabs: "比赛集合" (Collection of Contests) and "添加比赛" (Add Contest). There are three buttons at the bottom of this section: "Show 20 rows", "导出Excel", and "打印表格". A search input field labeled "搜索:" is also present. The main table displays contest details with columns: ID, Name, 开始时间 (Start Time), 结束时间 (End Time), and 比赛类型 (Contest Type). The data in the table is as follows:

ID	Name	开始时间	结束时间	比赛类型
000001	北京2015	2018-11-07 18:55:00	2018-11-07 23:55:00	校内组队赛
000002	测试(非定级)赛 [Cloned]	2018-11-07 18:46:00	2018-11-07 19:46:00	校内个人赛
000003	2018–2019 ACM-ICPC, NEERC, Southern Subregional Contest, Qualification Stage [Cloned]	2018-11-07 18:45:00	2018-11-07 22:45:00	校内组队赛
000004	Greater NY 2017 [Cloned]	2018-11-07 18:45:00	2018-11-07 23:45:00	校内组队赛
000005	The 2016 ACM-ICPC Asia Beijing Regional Contest [Cloned]	2018-11-07 18:42:00	2018-12-07 18:42:00	校内个人赛



查询校内个人赛，查询结果如下，查询正确。

ID	Name	开始时间	结束时间	比赛类型
000002	测试(非定级)赛 [Cloned]	2018-11-07 18:46:00	2018-11-07 19:46:00	校内个人赛
000005	The 2016 ACM-ICPC Asia Beijing Regional Contest [Cloned]	2018-11-07 18:42:00	2018-12-07 18:42:00	校内个人赛
000006	Group 7 Beijing 2017 ICPC [Cloned]	2018-11-07 18:40:00	2018-11-07 23:40:00	校内个人赛
000007	2016-2017 ACM-ICPC Northeastern European Regional Contest (NEERC 16) [Cloned]	2018-11-07 18:40:00	2018-11-07 23:40:00	校内个人赛
000009	[tai]Java大数测试水	2018-11-07 18:30:00	2018-11-11 23:18:00	校内个人赛

添加比赛，添加期末测验的比赛。并且持续时间这个输入框用 js 对输入内容进行了限制，只能输入整数，保证了安全性与稳定性。

比赛标题: 期末测验
持续时间: 5
时间单位: 时
比赛类型: 校内组队赛

返回 添加

添加成功之后，我们到所有比赛的页面查看我们添加的比赛。可以找到我们添加的数据，可以看到开始时间就是我们刚才添加比赛的时间。

ID	Name	开始时间	结束时间	比赛类型
002085	期末测验	2019-01-05 12:51:23	2019-01-05 17:51:23	校内组队赛

显示第 1 至 1 项结果，共 1 项 (由 2,085 项结果过滤)

上页 1 下页

6. 主页面-用户-训练题

进入用户训练题的界面。由于是新创建的账号，初始为空，然后我们往里面添加题目。



ACM 首页 题库 比赛 用户 临时账号v ▾

Chase Dreams!

目标的坚定是性格中最必要的力量源泉之一，也是成功的利器之一。没有它，天才会在矛盾无定的迷径中徒劳无功。

写过的题目 个人参加的比赛 队伍信息 添加题目 删除题目

Problem_ID	OJ & Number	标题	难度	算法	提交时间
0000001	HDU 2955	Robberies	入门	背包	2019-01-05 12:53:15

添加成功。我们多添加几道题目，发现均无异常。

ACM 自贡 题库 比赛 临时账号v ▾

Chase Dreams!

目标的坚定是性格中最必要的力量源泉之一，也是成功的利器之一。没有它，天才会在矛盾无定的迷径中徒劳无功。

添加题目

题目ID
0000001

添加成功

取消 添加

写过的题目 个人参加的比赛 队伍信息 添加题目 删除题目

Problem_ID	OJ & Number	标题	难度	算法	提交时间
0000001	HDU 2955	Robberies	入门	背包	2019-01-05 12:53:15
0000002	POJ 1080	Human Gene Functions	进阶	LCS	2019-01-05 12:53:49
0000003	POJ 1141	Brackets Sequence	进阶	区间DP	2019-01-05 12:53:51

我们来删除题目，均可删除成功。

ACM 首页 题库 比赛 用户 临时账号v ▾

Chase Dreams!

目标的坚定是性格中最必要的力量源泉之一，也是成功的利器之一。没有它，天才会在矛盾无定的迷径中徒劳无功。

删除题目

题目ID
0000001
0000021
0000022

取消 删除

写过的题目 个人参加的比赛 队伍信息 添加题目 删除题目

Problem_ID	OJ & Number	标题	难度	算法	提交时间
0000001	HDU 2955	Robberies	入门	背包	2019-01-05 12:53:15
0000021	POJ 1080	Human Gene Functions	进阶	LCS	2019-01-05 12:53:49
0000022	POJ 1141	Brackets Sequence	进阶	区间DP	2019-01-05 12:53:51



7. 主页面-用户-比赛

进入用户参加比赛的界面，由于是刚创建的账号，没有数据。

The screenshot shows the user interface for managing competitions. At the top, there is a navigation bar with tabs for '首页' (Home), '题库' (Problem Set), '比赛' (Competition), and '用户' (User). A dropdown menu for '临时账号' (Temporary Account) is visible. Below the navigation, a large banner displays the text 'Chase Dreams!' and a quote: '目标的坚定是性格中最必要的力量源泉之一，也是成功的利器之一。没有它，天才会在矛盾无定的迷径中徒劳无功。'. Underneath the banner, there are three buttons: '写过的题目' (Written Problems), '个人参加的比赛' (Contests Participated In), and '队伍信息' (Team Information). The '个人参加的比赛' button is highlighted. At the bottom right of the main area, there are two buttons: '添加比赛' (Add Competition) and '删除比赛' (Delete Competition). The main content area is currently empty, showing a table header with columns: 'Contest_ID', '排名' (Ranking), '完成题数' (Completed Problems), and '总罚时' (Total Penalties).

我们来添加几个比赛。可以看到我们添加成功了。

This screenshot shows the '添加比赛' (Add Competition) dialog box. It contains fields for '比赛ID' (Competition ID) set to '000001', '完成题数' (Completed Problems) set to '1', '排名' (Ranking) set to '1', '罚时 (时)' (Penalty Time (Hours)) set to '00', and '罚时 (分)' (Penalty Time (Minutes)) set to '00'. Below the form, a green message says '添加成功' (Added successfully). At the bottom of the dialog are '取消' (Cancel) and '添加' (Add) buttons.

再来删除比赛，测试也可正常删除。

This screenshot shows the '删除题目' (Delete Problem) dialog box. It has a field for '比赛ID' (Competition ID) which is empty. Below the form, a green message says '删除成功' (Deleted successfully). At the bottom of the dialog are '取消' (Cancel) and '删除' (Delete) buttons.



8. 主页面-用户-队伍

进入用户加入的队伍界面。账号是刚创建的，因此该用户目前没有加入任何队伍。

Team_ID	队伍名	组建时间
000189	无	2019-01-05
000190	无	2019-01-05

因此我们来创建几个队伍。可以看到，创建队伍成功。

Team_ID	队伍名	组建时间
000189	无	2019-01-05
000190	无	2019-01-05

我们再来退出一个队伍。经测试，退出队伍也成功了。

Team_ID	队伍名	组建时间
000190	无	2019-01-05



9. 主页面-队伍

进入队伍页面进行查看，由于是刚创建的队伍，因此没有队员，进行队员添加，此处添加的队员必须是数据库中已有的用户，在页面设计部分已经展示过了，此处不再赘述，直接进行用户添加。

The screenshot shows a form titled '队伍信息' (Team Information). It contains fields for Team_id (000190), Team_name (empty), Member1 (临时账号v), Member2 (empty), and Member3 (empty). At the bottom are '返回' (Return) and '修改' (Modify) buttons.

输入队员名为数据库中已经有的账号，我们即可看到系统显示添加成功。

The screenshot shows the same form after adding members. The Team_name field now contains 'amazing'. The Member1 field contains '临时账号v', Member2 contains 'Gene_Liu', and Member3 contains 'yiq'. A green success message '修改成功' (Modification successful) is displayed above the buttons. The buttons remain the same: '返回' and '修改'.

此时，我们登录其他队员的账号，可以查看其他队员也已经自动加入了这个队伍。

The screenshot shows the main ACM homepage. At the top, there's a dark banner with the text 'Chase Dreams!' and a quote: '目标的坚定是性格中最必要的力量源泉之一，也是成功的利器之一。没有它，天才会在矛盾无定的迷径中徒劳无功。' Below the banner, the user profile 'Gene_Liu' is shown. The main content area has tabs for '写过的题目' (Written Programs), '个人参加的比赛' (Personal Contests), and '队伍信息' (Team Information). The '队伍信息' tab is active. A table lists three teams: '000185' (宝可梦不醒), '000187' (天王海王星), and '000190' (amazing). At the bottom right are '创建队伍' (Create Team) and '退出队伍' (Exit Team) buttons.



再查看队伍参加的比赛。刚创建的队伍没有参加过比赛，因此没有比赛显示。

Contestant_ID	排名	完成题数	总罚时
000004			
000006			

我们进行比赛添加。可以看到，比赛添加成功。

添加成功

再进行比赛删除。经测试可知，删除成功。

删除成功



10. 结论说明

经过测试，可以看到页面中的所有功能均可正常使用，并且正常运行，包括对于很多错误的输入都进行了有效地避免，保证数据库的安全性和稳定性。

由于页面设计中已经对于大量细节进行了展示，因此此处对于大量细节不再赘述，本章的目的只是对于正在运行的数据库稳定性和安全性进行一下测试。

由于页面中所有数据的展示都是从数据库 select 出来的，因此本章也没有到数据库中去寻找我们新添的数据信息。因为只要页面上能够显示出来就已经说明数据插入成功了。否则页面将不会显示我们新插入的数据。



九、总结

9.1 系统优点

- ① 系统设置了题库、比赛、用户、队伍这四大模块的信息展示与处理，支持四大模块之间相互建立联系，比如比赛中包含题库中的题目，用户训练题库中的题目，队伍参加比赛等，体现了系统设置的合理性与完备性。
- ② 系统页面美观简洁，采用统一 UI 风格进行展示，使所有页面形成了一个整体。
- ③ 操作简洁，没有复杂的操作指示和使用说明，用户可以迅速上手。
- ④ 对所有可能出现的异常都进行了考虑，对于一些无法避免的错误输入，采用错误信息提示的方式来告知用户，对于一些可以避免的错误输入，则在用户输入信息的时候就通过对输入信息格式的限制来避免此类错误输入。

9.2 系统不足

- ① 系统在大规模进程并发处理的时候表现不好，有待提升。
- ② 没有引入评测系统，如果能够在系统上实现评测，则表现更好。

9.3 系统改进

- ① 对于系统事务并行操作的处理可以加强。
- ② 可以搭建一个属于这个平台自己的评测系统，那就成为了一个真正自给自足的在线评测系统。

9.4 经验与收获



在这次数据库课设的实践中，完成了自己从头到尾，每一行代码亲力亲为的一个工程。从一开始的数据库设计，再到为了往数据库中导入信息学了一整天爬虫最后成功爬下数据，再到做前端学习 html、css、js，然后选择适合的前端框架，一开始选择了 wordpress 之后放弃重新选择 bootstrap。然后查找 bootstrap 视频，看了三天开始搭建自己的前端。再然后寻找搭建后端的方法，一开始打算使用 php，因为简单。然后放弃 php，使用了 python 中的 tornado 架构，又看了 3 天视频，然后终于了解了大概之后，又哼哧哼哧地写了两周。最大的收获就是知道了搭建这些应用的学习方式，如 app、小程序、web、ios 应用程序等，初期搭建这些应用所需要的只是一些操作知识，讲究会用，甚至不需要深刻地理解，这和我们在大学中学习的高数、概率是有本质区别的，因此高数这些基础学科才成为了其他学科的基石。

也自这次课设实践后改变了之前的不会的观念。以前别人问“你会做 app 吗”，只会回答“不会”，现在会回答“我可以学！而且很快能学会！”这种观念的改变对于我来说是这次课设实践最大的收获！