**合肥工业大学**

**《专业实习与实训》**

**报告**

**题 目： 竞赛报名系统**

**学 号： 2021218152**

**姓 名： 陈嘉乐**

**专业班级： 计科21-3班**

**时 间： 2024/12/19**

目录

[1 实训目的及内容简介 3](#_Toc185781503)

[2 项目题目简介 3](#_Toc185781504)

[3 项目需求分析与设计 3](#_Toc185781505)

[4 项目设计方案 6](#_Toc185781506)

[5 项目实现与验证 22](#_Toc185781507)

[6 实训总结 30](#_Toc185781508)

### 1 实训目的及内容简介

**实训目的：**使得学生能够设计针对计算机与信息类复杂工程问题的解决方案，设计满足特定需求的系统、单元（部件）或开发流程，并能在设计环节中体现创新意识，在设计开发解决方案的过程中，运用工程管理与经济决策方法。通过项目了解社会发展形势，能分析和评价计算机专业工程实践对社会、健康、安全、法律、文化的影响，以及这些制约因素对项目实施的影响。能够组织、协调和指挥团队成员开展研究或设计开发工作。

**实训内容：**选择相应的实训实习题目，并根据自己的认知，完善需求，按照软件开发流程进行设计、开发和测试。

### 2 项目题目简介

**项目名称**：竞赛报名系统

**项目简介**：

本项目的核心目标是开发一套便于各类竞赛报名和统计的系统，帮助管理者高效组织竞赛活动，并为参赛者提供便捷的报名渠道。该系统采用 Java 语言开发，包括管理后台和数据库两部分。管理后台主要负责竞赛信息的发布、报名数据的管理以及统计分析；数据库则用于存储竞赛相关信息和报名数据。项目通过直观友好的界面和高效的数据处理方式，实现竞赛报名流程的全面数字化。

系统具备以下特点：

1. **便捷性**：参赛者可以在线完成报名，管理者无需手动整理报名数据。
2. **高效性**：通过后台自动化管理竞赛信息和报名数据，提高工作效率。
3. **统计分析**：提供报名数据统计功能，生成多维度的统计报表，辅助管理者决策。
4. **可扩展性**：系统支持多种竞赛类型，并能根据需求进行功能扩展。

### 3 项目需求分析与设计

**需求分析**

**功能需求：**

1. **参赛者功能模块：**
   1. 在线注册：用户需先注册个人账户，提供基础信息（姓名、联系方式、学校等）。
   2. 报名竞赛：选择竞赛并填写报名信息。
   3. 报名查询：查看报名状态，获取竞赛相关信息。
2. **管理者功能模块：**
   1. 竞赛管理：发布、修改、删除竞赛信息。
   2. 报名管理：查看和审核参赛者的报名信息。
   3. 数据统计：生成报名人数、竞赛类别分布等统计报表。
3. **系统功能模块：**
   1. 登录与权限控制：区分管理者与普通用户权限。
   2. 数据存储：将用户、竞赛、报名信息存储在数据库中。
   3. 数据备份与恢复：定期备份数据库内容，支持系统恢复。

**非功能需求：**

1. **性能需求：**支持同时处理 1000+ 用户并发请求，响应时间在 2 秒以内。
2. **安全需求**：
   1. 用户密码采用加密存储。
   2. 管理后台需验证权限访问。
3. **兼容性需求：**系统支持主流浏览器（Chrome、Firefox、Edge）。

**系统设计**

**架构设计：** 系统采用典型的三层架构：

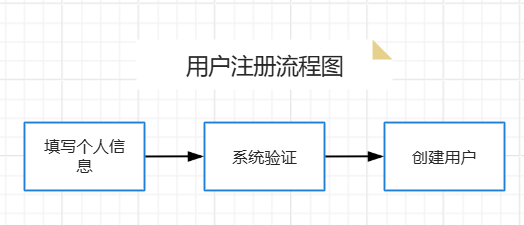
1. **表现层（Presentation Layer）**：
   1. 实现用户交互功能，如登录、报名、查询等。
   2. 使用 JSP 或前端框架实现动态页面。
2. **业务逻辑层（Business Logic Layer）**：
   1. 实现竞赛、报名等功能的逻辑处理。
   2. 提供统一的接口供表现层调用。
3. **数据访问层（Data Access Layer）**：
   1. 负责与数据库的交互，包括数据的增删改查。
   2. 使用 MyBatis实现持久化操作。

**数据库设计：** 数据库主要包括以下表：

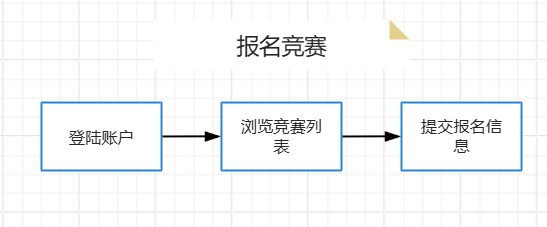
1. 用户表（t\_user）：存储用户基本信息，如用户名、密码、角色等。
2. 竞赛表（t\_competition）：存储竞赛信息，包括竞赛名称、时间、地点等。
3. 团队表（t\_team）：存储团队信息，包括团队名称、队长等。
4. 用户角色表（t\_user\_role）：存储用户与角色的映射关系。
5. 用户团队表（t\_user\_team）：存储用户与团队的关系信息。
6. 用户竞赛表（t\_user\_competition）：存储用户报名的竞赛信息。
7. 报名申请表（t\_apply）：存储用户提交的报名申请。

**系统功能流程图：**

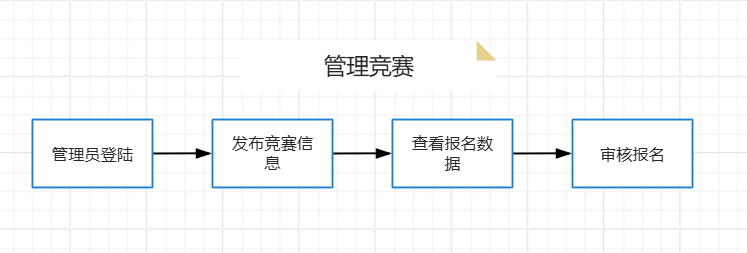
1. 用户注册：填写个人信息 → 系统验证 → 创建账户。



1. 报名竞赛：登录账户 → 浏览竞赛列表 → 提交报名信息。



1. 管理竞赛：管理员登录 → 发布竞赛信息 → 查看报名数据 → 审核报名。



通过以上分析和设计，本竞赛报名系统能够满足多样化的竞赛管理需求，显著提升报名效率和管理效果。

### 4 项目设计方案

**1. 系统总体设计**

本项目采用基于 Java 的后端和 Vue 前端的 B/S（Browser/Server）架构，划分为以下三层：

1. 表现层（前端）：基于 Vue 框架开发，负责实现用户界面交互和数据展示。
2. 业务逻辑层（后端）：基于 Spring Boot 开发，处理竞赛发布、报名管理和权限验证等核心业务逻辑。
3. 数据访问层（数据库）：使用 MySQL 存储数据，并通过 MyBatis 框架实现数据库访问。

**主要技术选型：**

* 后端技术： Java（JDK 17）、Spring Boot、MyBatis。
* 前端技术： Vue 3、Element-Plus。
* 数据库： MySQL。
* 缓存： Redis（用于加速报名统计查询）。
* 其他工具： Git（版本控制）、Postman（接口调试）。

**2. 模块设计**

1. **用户模块**

**功能描述：**

* 用户注册、登录和权限管理。
* 查询个人报名记录。
* 修改个人信息。

**实现细节：**

* 登录采用基于 Token 的身份认证（JWT）。
* 密码采用 MD5 加密存储。
* 用户可根据条件查询已报名竞赛信息。

@RequestMapping(value = "/updatePassword", method = RequestMethod.POST)

public @ResponseBody

ResponseMessage updatePassword(@RequestBody User u) {

try {

User user = (User) SecurityUtils.getSubject().getPrincipal();

//获取盐值

ByteSource credentialsSalt = ByteSource.Util.bytes(user.getUserName());

//加密密码

SimpleHash simpleHash = new SimpleHash("MD5", u.getPassword(), credentialsSalt, 1024);

user.setPassword(simpleHash.toString());

userService.updatePassword(user);

Subject subject = SecurityUtils.getSubject();

if (subject.isAuthenticated()) {

subject.logout();

}

return new ResponseMessage("1", "修改成功,请重新登录！");

} catch (Exception e) {

e.printStackTrace();

return new ResponseMessage("0", "修改失败");

}

}

@RequestMapping(value = "/login", method = RequestMethod.POST)

public @ResponseBody

ResponseMessage login(@RequestBody User user) {

//获取subject

Subject subject = SecurityUtils.getSubject();

if (!subject.isAuthenticated()) {

System.out.println("用户未登录");

//如果用户未登录，执行登录逻辑

try {

UsernamePasswordToken usernamePasswordToken =

new UsernamePasswordToken(user.getUserName(), user.getPassword());

// 执行认证提交

subject.login(usernamePasswordToken);

//返回登录成功信息

ResponseMessage responseMessage = new ResponseMessage(SuccessEnum.S\_LOGIN\_SUCCESS);

//获取用户的信息

User principal = (User) subject.getPrincipal();

for (Role role : principal.getRoles()) {

if (role.getRoleId().equals(user.getRoles().iterator().next().getRoleId())) {

//将角色返回

responseMessage.getData().put("roles",

principal.getRoles());

System.out.println(principal.getRoles());

return responseMessage;

}

}

if ("1".equals(user.getRoles().iterator().next().getRoleId())) {

//无该角色，登出（后同）

if (SecurityUtils.getSubject().isAuthenticated()) {

subject.logout();

}

return new ResponseMessage(ErrorEnum.E\_NOTADMIN);

} else if ("2".equals(user.getRoles().iterator().next().getRoleId())) {

if (SecurityUtils.getSubject().isAuthenticated()) {

subject.logout();

}

return new ResponseMessage(ErrorEnum.E\_NOTSTUDENT);

} else {

if (SecurityUtils.getSubject().isAuthenticated()) {

subject.logout();

}

return new ResponseMessage(ErrorEnum.E\_NOTROLE);

}

} catch (UnknownAccountException e) {

e.printStackTrace();

//返回用户不存在

return new ResponseMessage(ErrorEnum.E\_UNKNOWN\_ACCOUNT);

} catch (AuthenticationException e) {

e.printStackTrace();

//返回密码错误信息

return new ResponseMessage(ErrorEnum.E\_PASSWORD\_ERROR);

}

} else {

//如果用户已经登录则无需执行登录轮逻辑，直接返回登录成功

return new ResponseMessage(SuccessEnum.S\_LOGINED);

}

}

@Service

public class UserServiceImpl implements UserService {

//注意@Resource 与 @Autowired 区别

@Autowired

private UserDao userDao;

@Override

public User findByUsernameAndPassword(User user) {

User u = userDao.findByUsernameAndPassword(user);

return u;

}

@Override

public String findCollegeIdByUserName(String userName) {

String collegeId = userDao.findCollegeIdByUserName(userName);

return collegeId;

}

@Override

public User findUserByUserName(String userName) {

User user = userDao.findUserByUserName(userName);

return user;

}

@Override

public void updatePassword(User user) {

userDao.updatePassword(user);

}

@Override

public User findUserByUserId(String userId) {

User user = userDao.findUserByUserId(userId);

return user;

}

@Override

public List<User> getAllUser() {

List<User> users = userDao.getAllUser();

return users;

}

}

1. **竞赛模块**

**功能描述：**

* 管理员发布、修改和删除竞赛信息。
* 用户浏览竞赛列表并进行报名。
* 支持竞赛按状态分类（未开始、进行中、已结束）。

**实现细节：**

* 后端提供竞赛信息分页查询接口，支持筛选和排序功能。
* 前端使用 Vue 的组件化开发展示竞赛列表，配合 Element-Plus 实现表单输入和对话框操作。

/\*\*

\* 根据比赛id获得比赛

\*

\* @param competitionId

\* @return

\*/

@RequestMapping("/findCompetitionById")

public @ResponseBody

ResponseMessage findCompetitionById(String competitionId) {

try {

Competition competition = competitionService.findCompetitionById(competitionId);

ResponseMessage responseMessage = new ResponseMessage("1", "获取成功");

responseMessage.getData().put("competition", competition);

System.out.println(competition);

return responseMessage;

} catch (Exception e) {

e.printStackTrace();

return new ResponseMessage("0", "获取失败");

}

}

/\*\*

\* 获得该竞赛已报名的用户列表

\*

\* @param competitionId

\* @return

\*/

@RequestMapping("/findUserByCompetitionId")

public @ResponseBody

ResponseMessage findUserByCompetitionId(String competitionId) {

try {

List<User> users = competitionService.findUserByCompetitionId(competitionId);

ResponseMessage responseMessage = new ResponseMessage("1", "获取成功");

responseMessage.getData().put("users", users);

return responseMessage;

} catch (Exception e) {

e.printStackTrace();

return new ResponseMessage("0", "获取失败");

}

}

1. **报名模块**

**功能描述：**

* 用户选择竞赛进行报名。
* 管理员审核报名信息。
* 查看和导出报名数据。

**实现细节：**

* 报名信息存储在 t\_user\_competition 表中，关联用户和竞赛表。
* 管理员可通过前端的筛选条件查看特定竞赛的报名详情。
* 使用后端批量导出报名数据功能，支持生成 Excel 文件。

/\*\*

\* 个人赛报名

\*

\* @param competitionId

\* @return

\*/

@RequestMapping("/doApply")

public @ResponseBody

ResponseMessage doApply(String competitionId) {

//创建中间表对象，封装报名信息

UserCompetition userCompetition = new UserCompetition();

Competition competition = new Competition();

competition.setCompetitionId(competitionId);

userCompetition.setCompetition(competition);

User user = (User) SecurityUtils.getSubject().getPrincipal();

userCompetition.setUser(user);

//获取当前时间毫秒值,并存入封装对象

userCompetition.setDate(System.currentTimeMillis());

try {

applyService.insertApply(userCompetition);

return new ResponseMessage("1", "报名成功");

} catch (Exception e) {

e.printStackTrace();

return new ResponseMessage("0", "报名失败");

}

}

/\*\*

\* 组队赛报名

\*

\* @param teamId

\* @return

\*/

@RequestMapping("/doApplyByTeam")

public @ResponseBody

ResponseMessage doApplyByTeam(String teamId) {

//创建队伍对象，封装id和报名时间,队伍状态

Team team = new Team();

team.setTeamId(teamId);

//设置队伍状态为3,即已报名

team.setTeamState(3);

team.setApplyTime(System.currentTimeMillis());

System.out.println(team);

try {

teamService.updateTeam(team);

return new ResponseMessage("1", "报名成功");

} catch (Exception e) {

e.printStackTrace();

return new ResponseMessage("0", "报名失败");

}

}

/\*\*

\* 个人赛取消报名

\*

\* @param competitionId

\* @return

\*/

@RequestMapping("/cancelApply")

public @ResponseBody

ResponseMessage cancelApply(String competitionId) {

//创建中间表对象，封装报名信息

UserCompetition userCompetition = new UserCompetition();

Competition competition = new Competition();

competition.setCompetitionId(competitionId);

userCompetition.setCompetition(competition);

User user = (User) SecurityUtils.getSubject().getPrincipal();

userCompetition.setUser(user);

try {

applyService.deleteApply(userCompetition);

return new ResponseMessage("1", "取消报名成功");

} catch (Exception e) {

e.printStackTrace();

return new ResponseMessage("0", "取消报名失败");

}

}

/\*\*

\* 管理员取消用户报名

\*

\* @param userCompetition

\* @return

\*/

@RequestMapping(value = "/adminCancelApply", method = RequestMethod.POST)

public @ResponseBody

ResponseMessage adminCancelApply(@RequestBody UserCompetition userCompetition) {

System.out.println(userCompetition);

try {

applyService.deleteApply(userCompetition);

return new ResponseMessage("1", "取消报名成功");

} catch (Exception e) {

e.printStackTrace();

return new ResponseMessage("0", "取消报名失败");

}

}

/\*\*

\* 组队赛取消报名

\*

\* @param competitionId

\* @return

\*/

@RequestMapping("/cancelTeamApply")

public @ResponseBody

ResponseMessage cancelTeamApply(String competitionId) {

User user = (User) SecurityUtils.getSubject().getPrincipal();

try {

Team team = teamService.findTeamByCaptainIdAndCompetitionId(user.getUserId(), competitionId);

if (team != null && team.getApplyTime() != null) {

if (team.getCaptain().getUserId().equals(user.getUserId())) {

teamService.updateTeamCancelApply(team.getTeamId());

return new ResponseMessage("1", "取消报名成功");

} else {//非队长取消报名

return new ResponseMessage("1", "取消报名失败，队员无权取消报名，请联系队长");

}

} else {

return new ResponseMessage("0", "未报名，取消报名失败");

}

} catch (Exception e) {

e.printStackTrace();

return new ResponseMessage("0", "取消报名失败");

}

}

/\*\*

\* 管理员取消组队赛报名

\*

\* @param teamId

\* @return

\*/

@RequestMapping("/adminCancelTeamApply")

public @ResponseBody

ResponseMessage adminCancelTeamApply(String teamId) {

try {

Team team = new Team();

team.setTeamId(teamId);

team.setTeamState(2);

System.out.println(team);

teamService.updateTeam(team);

return new ResponseMessage("1", "取消报名成功");

} catch (Exception e) {

e.printStackTrace();

return new ResponseMessage("0", "取消报名失败");

}

}

1. **数据统计模块**

**功能描述：**

* 统计每个竞赛的报名人数。
* 按时间、地域等维度生成统计图表。

**实现细节：**

* 前端使用 ECharts 绘制柱状图、折线图等统计图表。
* Redis 缓存统计数据，减少数据库查询压力

/\*\*

\* 指定缓存失效时间

\*

\* @param key 键

\* @param time 时间(秒)

\* @return 操作结果 true成功 false失败

\*/

public boolean expire(String key, long time) {

try {

if (time > 0) {

redisTemplate.expire(key, time, TimeUnit.SECONDS);

}

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* 根据key 获取过期时间

\*

\* @param key 不能为空

\* @return 返回0代表永久有效

\*/

public long getExpire(String key) {

return redisTemplate.getExpire(key, TimeUnit.SECONDS);

}

/\*\*

\* 判断key是否存在

\*

\* @param key 键

\* @return true存在 false不存在

\*/

public boolean hasKey(String key) {

try {

return redisTemplate.hasKey(key);

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* 删除缓存

\*

\* @param key 可以传一个值或者多个

\*/

public void del(String... key) {

if (key != null && key.length > 0) {

if (key.length == 1) {

redisTemplate.delete(key[0]);

} else {

redisTemplate.delete(CollectionUtils.arrayToList(key));

}

}

}

1. **权限控制模块**

**功能描述：**

* 区分管理员和普通用户权限。
* 管理员拥有竞赛发布、报名审核等操作权限。

**实现细节：**

* 后端通过 Spring Security 实现基于角色的权限控制。
* 前端根据用户角色动态渲染不同的功能菜单。

**3. 数据库设计**

主要表结构：

1. 用户表（t\_user）：
   * 存储用户基本信息，如用户名、加密密码、角色等。
   * 关联角色表实现权限控制。

|  |  |
| --- | --- |
| Id | 主键，唯一标识用户 |
| Username | 用户名 |
| Password | 加密后的密码 |
| Role | 用户角色（admin/user） |

1. 竞赛表（t\_competition）：
   * 存储竞赛的基本信息，如名称、描述、时间、地点等。

|  |  |
| --- | --- |
| Id | 主键 |
| Time | 时间 |
| Place | 地点 |
| Describe | 描述 |
| Name | 名称 |

1. 用户竞赛表（t\_user\_competition）：
   * 存储用户报名的竞赛记录。

|  |  |
| --- | --- |
| Id | 主键 |
| User\_id | 用户ID |
| Competition | 竞赛ID |
| Status | 状态（已审核/已通过/已拒绝） |

1. 角色表（t\_user\_role）：
   * 存储用户与角色的关联关系。

|  |  |
| --- | --- |
| Id | 主键 |
| User\_id | 用户ID |
| Role | 角色名称 |

1. 团队表（t\_team）：
   * 存储团队信息及其与用户的关系。

|  |  |
| --- | --- |
| Id | 主键 |
| User\_id | 用户ID |
| Team\_id | 团队ID |

通过以上模块设计，该竞赛报名系统能够满足参赛者和管理员的核心需求，保证操作流畅性和系统稳定性。

### 5 项目实现与验证

介绍项目关键模块的实现，并附有相关截图进行验证。

**1. 系统前端模块实现**

**1.1 登录功能**

* **功能说明**：提供用户登录功能，通过 JWT 实现身份验证。
* **实现细节**：
  + 使用 Vue 和 Element-Plus 实现登录表单，支持表单校验。
  + 提交登录信息后，前端存储服务器返回的 Token，用于后续接口请求。



**1.2 竞赛信息展示功能**

* **功能说明**：用户可浏览竞赛列表，查看竞赛详情并进行报名。
* **实现细节**：
  + 使用 Axios 请求后端接口获取竞赛数据。
  + 前端使用 Vue 的动态组件渲染竞赛信息，支持按状态筛选。





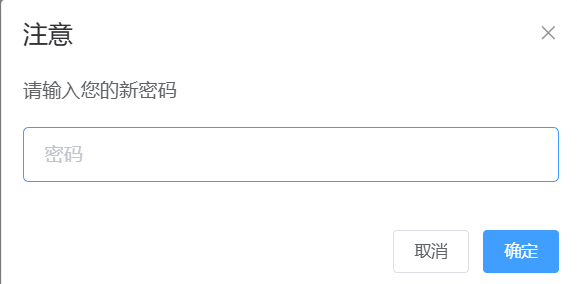


**2. 后端关键模块实现**

**2.1 用户权限管理**

* **功能说明**：实现用户登录、角色分配和权限验证以及密码修改。
* **实现细节**：
  + 使用 Spring Security 实现基于角色的权限控制。
  + 登录成功后生成 JWT Token，附加到每次请求的 Header 中。





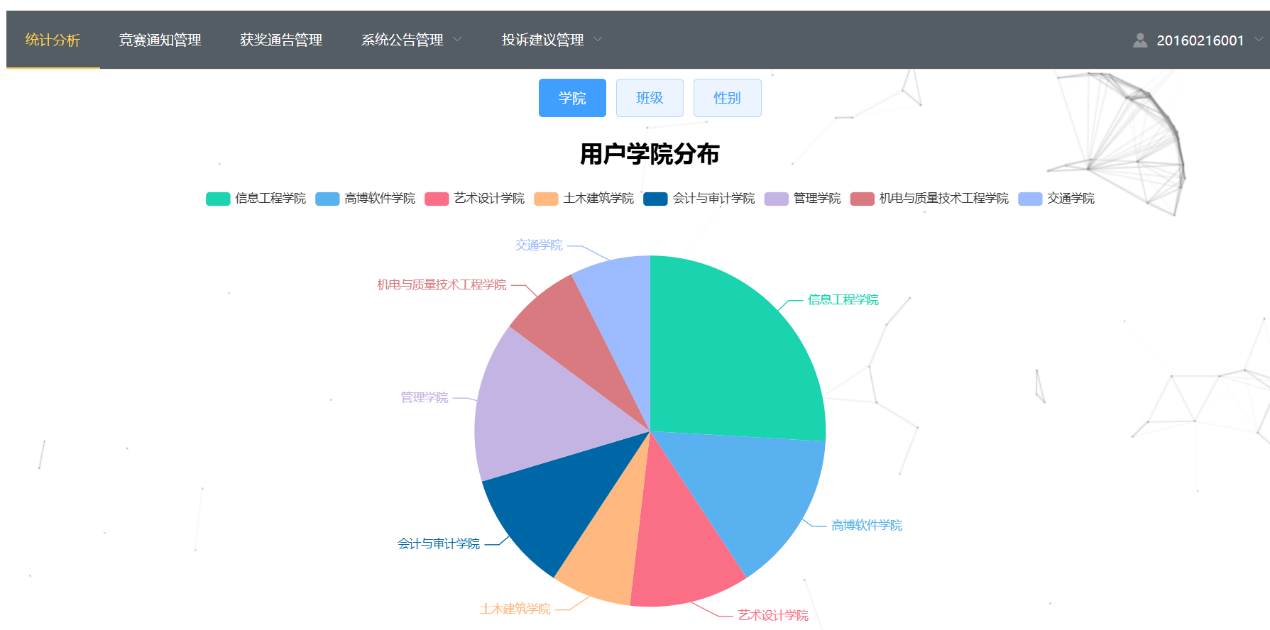


**2.2 报名功能实现**

* **功能说明**：用户选择竞赛报名，管理员审核报名信息。
* **实现细节**：
  + 用户提交报名数据后，后端存储在 t\_user\_competition 表中。
  + 管理员通过后台审核报名请求，并更新其状态（待审核→已通过/已拒绝）。

**2.3 数据统计功能实现**

* **功能说明**：根据报名数据生成多维度统计报表。
* **实现细节**：
  + 后端使用 Redis 缓存统计数据，减轻数据库压力。
  + 前端通过调用后端统计接口，使用 ECharts 绘制图表。

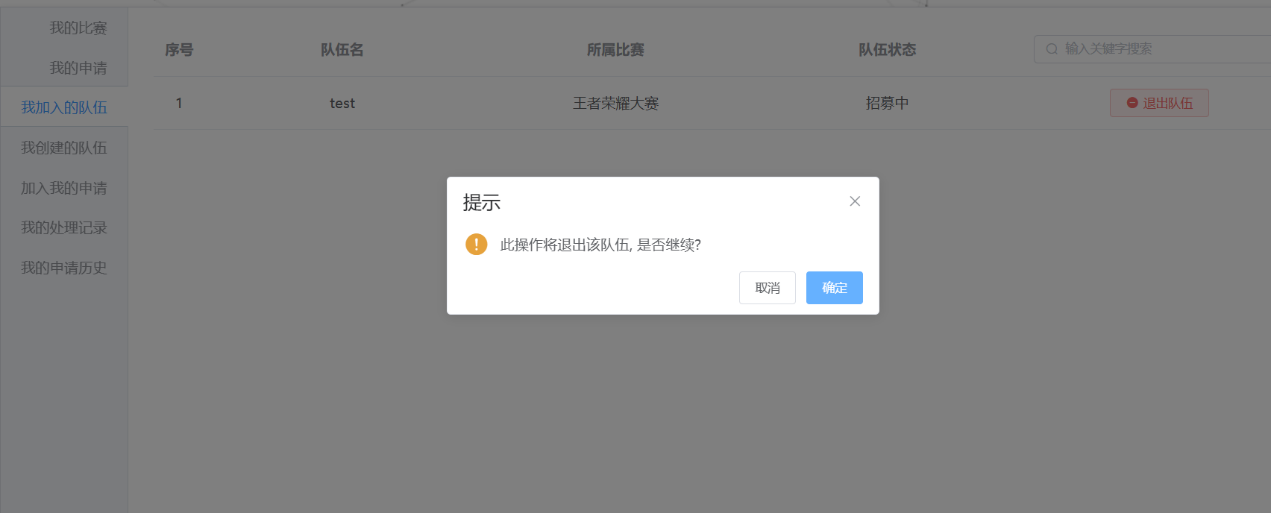
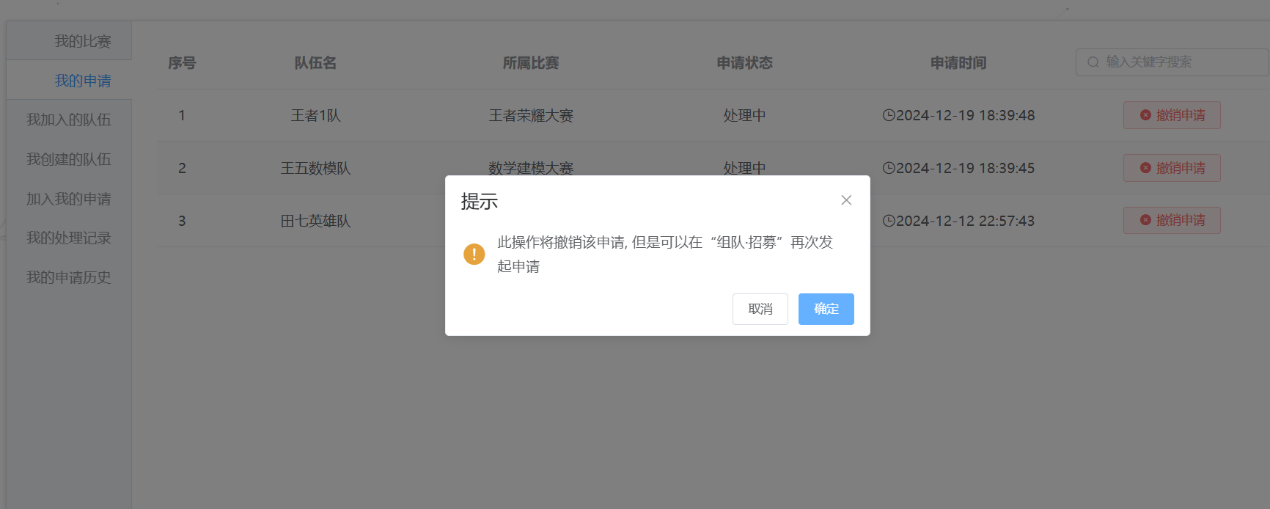


**3. 项目测试与验证**

**3.1 功能测试**

* 测试用户登录、竞赛报名、创建以及加入队伍等功能的完整流程。
* 验证管理员发布竞赛、审核报名数据的功能。

用户竞赛报名以及创建和加入队伍测试



管理员发布竞赛测试



**3.2 性能测试**

* 使用 Apache JMeter 测试并发用户数量，确保系统在 1000+ 并发下响应时间小于 2 秒。
* 验证 Redis 缓存对统计查询性能的提升。

通过对以上模块的实现与验证，本项目完成了竞赛报名系统的核心功能，并满足需求分析中的功能和非功能需求。

### 6 实训总结

在完成竞赛报名系统的开发过程中，我深刻体会到了将理论知识应用于实践的过程所带来的挑战与收获。从最初的需求分析到最终的功能实现，每个环节都需要充分的思考和细致的规划。这不仅提升了我的技术能力，也让我对项目开发的全流程有了更深刻的理解。

本项目采用了前后端分离的架构，前端使用 Vue 实现了用户友好的界面，后端则通过 Spring Boot 提供了强大的接口支持。通过两者的协同开发，我进一步掌握了前后端交互的实现方式，包括接口设计、状态管理和数据传输。尤其是在权限管理和数据统计功能的实现中，我切实感受到了如何平衡系统的安全性和性能优化。

此外，在项目过程中，面对复杂的业务逻辑和高并发性能需求，我意识到代码规范和模块化设计的重要性。一方面，良好的代码结构能显著提升后续的维护效率；另一方面，使用 Redis 缓存和 MyBatis 持久化框架优化了数据存储和访问效率，让系统能够应对更大的数据规模和更高的用户访问量。

这个项目不仅是技术的实践，也是团队协作能力的体现。在实际开发中，我需要与其他团队成员紧密沟通，明确功能需求，并在分工协作中学会妥善处理意见分歧。通过这种合作，我深刻认识到高效的沟通和明确的责任分配是项目成功的关键。

总的来说，本次实训让我积累了宝贵的项目经验，也提升了自身的技术能力和问题解决能力。这些收获不仅体现在竞赛报名系统的完成上，更让我在未来的开发工作中充满信心。

|  |  |  |
| --- | --- | --- |
| **专业实习与实训验收评分细则** | | |
| **成绩等级** | **具体表现** | **教师评分（百分制）** |
| **优秀[85-100]** | **1）能够在规定时间内完成项目，项目具有完整性，交互设计优秀；2）工作量充足；** |  |
| **良好[75-85)** | **1）能够在规定时间内完成项目，且项目具有较好的完整性，交互设计较为优秀；2）工作量较为充足；** |  |
| **中等[65-75)** | **1）能够在规定时间内完成项目，项目完整性较好，交互设计一般；2）工作量一般；** |  |
| **及格[60-65)** | **1）虽完成项目编码但未能在规定时间内完成的；** |  |
| **不及格（<60）** | **1）未进行验收的；** |  |
| **专业实习与实训答辩评分细则** | | |
| **成绩等级** | **具体表现** | **教师评分（百分制）** |
| **优秀[85-100]** | **1）能够在规定时间内完成答辩；2）能够清晰、正确、完美地回答答辩教师提问；3）项目难度与工作量较大；** |  |
| **良好[75-85)** | **1）能够在规定时间内完成答辩；2）能够正确回答答辩教师提问；3）项目具有一定难度与工作量；** |  |
| **中等[65-75)** | **1）能够在规定时间内完成答辩；2）能够较为正确地回答答辩教师提问；3）项目完成具有一定工作量；** |  |
| **及格[60-65)** | **1）不能够正确地回答答辩教师提问的；** |  |
| **不及格（<60）** | **1）未参与答辩的；** |  |
| **专业实习与实训报告评分细则** | | |
| **成绩等级** | **具体表现** | **教师评分（百分制）** |
| **优秀[85-100]** | **1）实习报告格式完美，充分采用图表来说明问题，章节设计优秀，工作量饱满；** |  |
| **良好[75-85)** | **1）实习报告格式良好，采用了图表来说明问题，章节设计良好，工作量较为饱满；** |  |
| **中等[65-75)** | **1）实习报告格式较好，章节设计较好，工作量一般；** |  |
| **及格[60-65)** | **1）虽提交报告但未能在规定时间内完成的；** |  |
| **不及格（<60）** | **1）未提交报告的；** |  |
| 评阅教师签名： 电子签名 | | |