1. **需求分析**
   1. **问题陈述**

作为在校大学生，平时除了紧张的学习生活外，大多数同学都会参与到社团活动中。许多社团会招收大量的会员，开展一系列丰富有趣的社团活动。由于招收会员人数众多，社团的管理人员在进行管理时所面临的问题也随之变得复杂。传统的社团管理方式往往存在一些弊端：

1. 社团会员注册一般采用纸质登记的方式，再由干事输入Excel表格；有些则利用问卷星等工具，可直接获得Excel表格。这种方式记录的会员信息在后期使用上灵活性较低，往往需要管理人员进行大量手动操作。
2. 会员活动考勤一般采用纸质登记的方式，再由干事输入Excel表格。这种方式存在一些漏洞，比如帮签。
3. 活动报名方面，大多数情况下，即便管理人员已经拥有会员信息，也需要会员另外再次填写报名信息，不但会员需要反复填写，干事也需要再次整理资料，效率很低。

为了解决以上问题，我们小组将开发一个会员管理系统，采用B/S结构。该系统在会员第一次登记的时候，需要填写基本信息，包括姓名、性别、学号、专业、电话、邮箱等，这些数据将被记录在服务器的数据库中，会员可利用用户名和密码登录系统。登陆后，会员可以看到所有已发布的活动、已报名的活动、个人信息及签到记录等，在活动详情中可以报名，报名信息将直接从数据可调用，而无需重复填写。若某个活动需要签到，会员可在活动详情中进行考勤；干事账号具备会员账号的全部功能，另外能够发布活动、编辑活动、发布考勤以及查看考勤名单等。

* 1. **用例析取**

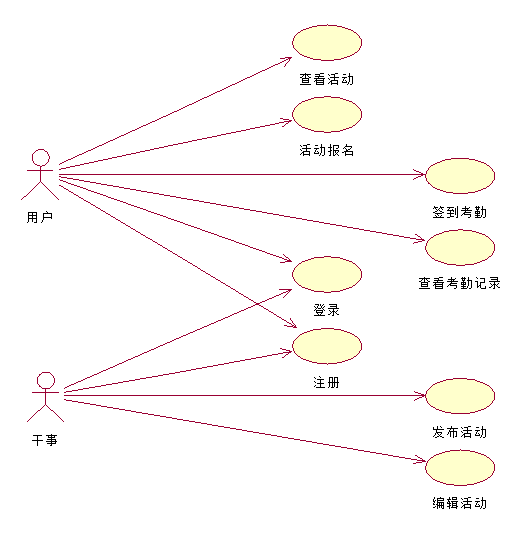


图1.1 用例图

* 1. **用例规约**
     1. **注册**

1. **简要描述**

本用例允许用户和干事在平台注册。

1. **参与者**

活动管理系统的用户及干事。

1. **场景描述**

用户需要先注册账号才能够正常使用系统，注册时需要根据提示填写姓名、学号、用户名、密码、联系方式等信息。

1. **前置条件**

使用者希望使用活动管理系统并申请成为注册用户。

1. **后置条件**

使用者注册成功后将成为系统的注册用户。

1. **事件流**
2. **基本事件流**
3. 使用者进入注册页。
4. 使用者根据系统提示框输入系统所需要的信息。
5. 前端成功验证使用者的输入。
6. 前端向后端提交用户输入。
7. 后端验证信息无误且无重复注册。
8. 后端向前端反馈成功信息，前端展示成功提示信息。
9. 注册成功后自动登录系统。
10. **异常事件流**
11. 使用者输入信息不合法，系统给出相应错误提示。
12. 使用者重复注册，后端返回错误信息，前端返回相应错误提示。
13. 系统异常，无法注册，前端给出相应错误提示。
    * 1. **签到考勤**
14. **简要描述**

到场参与活动的用户，扫描干事在现场发布的二维码即可进行签到。

1. **参与者**

活动管理系统的用户。

1. **场景描述**

用户需要先注册账号才能够正常使用系统，注册时需要根据提示填写姓名、学号、用户名、密码、联系方式等信息。

1. **前置条件**

干事发起签到并展示系统生成的二维码，用户可以使用微信扫一扫或其他扫码工具进行扫码。

1. **后置条件**

若本用例成功，系统将根据活动信息及用户信息在数据库中记录考勤信息。

1. **事件流**
2. **基本事件流**
3. 用户使用扫码功能，扫描干事展示的签到二维码。
4. 用户访问二维码中的URL，前端将用户信息及活动信息传至后端。
5. 后端根据所获得的信息，结合数据库信息，判断用户是否满足签到条件。
6. 满足条件将记录签到信息至数据库。
7. 后端返回成功信息至前端，前端展示成功签到提示。
8. **异常事件流**
9. 扫码后，前端检测到用户未登录，返回相应错误提示。
10. 扫码后，后端检测到用户并未报名当前活动，返回相应错误提示。
11. 其他错误，返回相应错误提示。

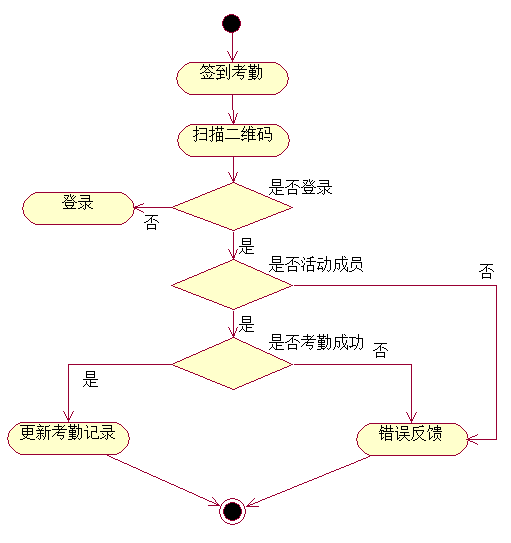


图1.2 签到考勤活动图

* + 1. **发布活动**

1. **简要描述**

干事发布活动至系统，吸引用户报名参加。

1. **参与者**

活动管理系统的干事。

1. **场景描述**

干事在举行活动前，将活动的介绍放到系统中，使得所有用户能够了解活动的大致情况，吸引用户报名参加。

1. **前置条件**

登录干事账号，点击添加活动按钮，编辑活动信息，信息无误即可发布至系统。

1. **后置条件**

若本用例成功，系统会将活动展示在首页，用户可以自由查看详情。

1. **事件流**
2. **基本事件流**
3. 登录干事账号。
4. 点击添加活动按钮，跳转至活动编辑页。
5. 填写相关信息并上传图片。
6. 信息无误且完整时，后端将数据存入数据库，并反馈添加成功信息。
7. 前端显示成功添加提示。
8. **异常事件流**
9. 若非干事账号，将无法看到添加活动按钮。
10. 非干事账号，直接通过URL无法访问活动编辑页面。
11. 其他错误，返回相应错误提示。

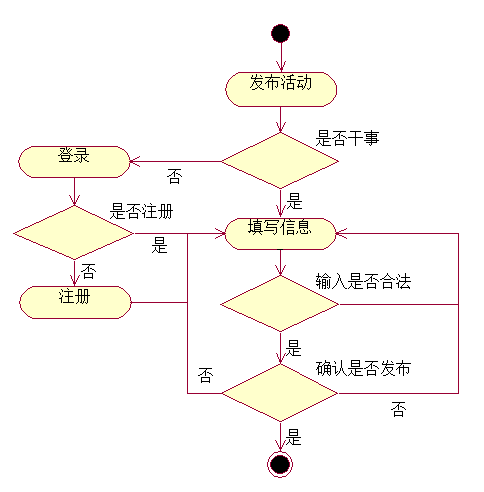


图1.3 活动发布活动图

* + 1. **活动报名**

1. **简要描述**

用户浏览到合心意的活动，可以报名参加。

1. **参与者**

活动管理系统的用户。

1. **场景描述**

用户浏览到合适的活动时，可以报名活动。报名后，干事能够获得报名名单，并且只有报名后的用户才可签到考勤。

1. **前置条件**

使用者注册成为系统用户后，可以报名任何有空位的公开活动，报名后可以考勤签到。

1. **后置条件**

若本用例成功，用户信息将会与活动信息绑定，记录在数据库中。

1. **事件流**
2. **基本事件流**
3. 浏览活动详情。
4. 点击报名按钮。
5. 检查登录状态。
6. 用户信息和活动信息将绑定记录在数据库中。
7. 报名成功后，前端页面进行反馈。
8. **异常事件流**
9. 点击报名，但用户没有登录，跳转登陆页面。
10. 点击报名，报名人数已满，返回相应错误提示。
11. 其他错误，返回相应错误提示。

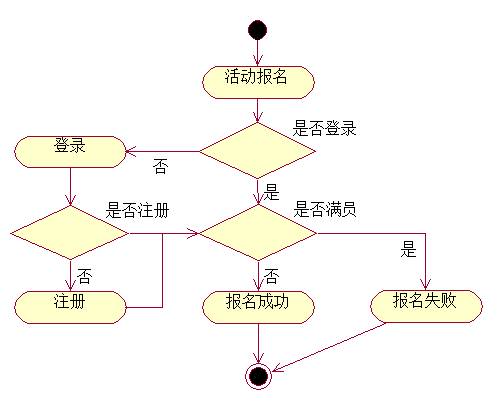


图1.4 活动报名活动图

* 1. **补充规约**
     1. **目标**

本文档的目的是定义活动管理系统的需求。本补充规约列出了不便于在用例模型的用例中获取的系统需求。补充规约和用例模型一起记录关于系统的一整套需求。

* + 1. **范围**

1. 本补充规约适用于活动管理系统，将要由学习面向对象软件分析与设计的同学开发。
2. 本规约除定义了许多用例中所共有的功能性需求以外，还定义了系统的非功能性需求，例如：可靠性、可用性、安全性等。
   * 1. **参考**

《UML和模式应用》一书

* + 1. **可行性**

本系统是Web应用，基于现在流行的JavaScript语言，采用成熟且被行业内广泛使用的Nodejs，并基于成熟稳定的前后端框架，支持大多主流的现代浏览器。

* + 1. **可靠性**

采用稳定成熟的Nodejs，拥有大量优秀、可靠地功能模块和第三方库，并且使用Meteor全栈框架，提供了包括用户登录注册、数据加密等高级功能的一站式解决方案。

* + 1. **安全性**

用户登录注册时，密码经过加密算法处理且登录时需要与后端进行token认证，保证了账户信息的安全。另外，数据采用严格的发布订阅模式，每个页面只能获取所需数据，对于其他数据无权访问。

* 1. **术语表**

|  |  |
| --- | --- |
| **名词术语** | **定义** |
| 前端 | 前端指的是浏览器中的页面，主要负责处理与用户的交互，承载着输入和输入的功能。在本系统中主要指的是Web App的界面。 |
| 后端 | 后端指的是运行在服务器中，负责处理数据存取调用的代码。在本系统中，主要处理对数据库的增删查改，发布数据给前端。 |
| 数据库 | 数据库指的是以一定方式储存在一起、能为多个用户共享、具有尽可能小的冗余度的特点、是与应用程序彼此独立的数据集合。本系统中采用MongoDB，管理Web App的所有数据。 |
| 用户 | 特指已注册本系统账号并登陆使用的使用者。 |
| 干事 | 在普通用户基础上，具有更多权限的使用者，权限包括：增加活动、编辑活动、发布考勤、删除活动等。 |
| 活动 | 特指在本系统中发布且可以被系统用户报名参加的事件，由干事发布、组织 |

表1 术语表