目录

[面向对象设计说明 2](#_Toc420529632)

[人机交互设计 2](#_Toc420529633)

[需求分析 2](#_Toc420529634)

[面向对象设计 6](#_Toc420529635)

[控制驱动设计 8](#_Toc420529636)

[根据类图的账户类 9](#_Toc420529637)

[根据类图的用户类 9](#_Toc420529638)

[根据类图的投票类 12](#_Toc420529639)

[根据类图的管理员类 12](#_Toc420529640)

[补充说明 13](#_Toc420529641)

[数据管理设计 13](#_Toc420529642)

# 面向对象设计说明

面向对象设计分为以下三个部分：

## 人机交互设计

### 需求分析

1. **人员活动类**

考虑到此产品适用于社会各行各业人群，用户职业构成多样，对计算机和移动设备熟悉度不同，产品界面要具有启发性和容错性，即能够启发和引导用户正确、有效地进行界面操作，并对用户的误操作有容忍能力或补救措施。另一方面，预计用户会以30岁以下的年轻人为主，并包括部分中年人群，所以产品界面风格要有艺术性和趣味性，简单美观，方便操作。

1. **从User Case中提取人机交互描述**

|  |
| --- |
| 浏览内容（人机交互） |
| 显示投票标题和图片  If 点击标题 then  显示投票情况  If 点击投票 then  输出提示信息“需要登录”；  End if；  End if； |

|  |
| --- |
| 发布投票（人机交互） |
| 用户在个人主页点击“发布投票”  输入投票标题，投票内容，选择个数  For 用户每个选择 do  输入描述；  上传照片；  End for；  If 按“提交”按钮 then  输出提示信息“发布成功”；  显示主页；  End if；  Else if 按“退出”按钮 then  输出提示信息“放弃发布投票”；  显示主页；  End if；  End else； |

|  |
| --- |
| 关闭投票（人机交互） |
| 用户点击“查看历史->查看发布的投票”；  点击“关闭投票”；  输出确认信息“是否确认关闭？”  If 按“是”按钮 then  输出提示信息“投票关闭”；  显示“查看发布的投票”页面；  End if；  Else if 按“否”按钮 then  输出提示信息“放弃关闭投票”；  显示“查看发布的投票”页面；  End if；  End else； |

|  |
| --- |
| 投票（人机交互） |
| 用户点击投票标题；  显示图片和当前投票情况  If 用户点击“投票” then  If 用户状态为未参与 then  用户点击一个选项；  显示用户对此投票状态变为已参与；  End if  Else  输出提示“已参与该投票”；  End else  End if |

|  |
| --- |
| 用户评价（人机交互） |
| If 用户点击“评价” then  If 用户状态为未参与 then  输出提示信息“需要参与投票后才可评价”；  End if  Else  用户输入评价内容；  输出提示信息“评价成功”；  End else  End if |

|  |
| --- |
| 关注投票（人机交互） |
| 用户点击投票标题；  显示图片和当前投票情况；  If 用户点击“关注” then  输出提示信息“关注投票成功”；  End if |

|  |
| --- |
| 取消关注投票（人机交互） |
| 用户点击投票标题；  显示图片和当前投票情况  If 用户点击“取消关注” then  输出提示信息“取消关注投票成功”；  End if |
| 后置条件：更新个人历史中关注的投票 |

|  |
| --- |
| 查看历史（人机交互） |
| 用户点击“查看历史”  If用户点击“查看发布的投票” then  显示“发布的投票”页面；  End if  Else if用户点击“查看发布的投票” then  显示“发布的投票”页面；  End else if  Else  显示“发布的投票”页面；  End else |

### 面向对象设计

1. **窗口Window类作为基本类**

|  |  |
| --- | --- |
| Window | |
| 属性 | |
| width: float | 宽度 |
| height: float | 高度 |
| locateX: float | 位置横坐标 |
| locateY: float | 位置纵坐标 |
| 操作 | |

1. **窗口的部件为窗口对象类**

窗口内的一切部件均继承自Component类：

|  |  |
| --- | --- |
| Component | |
| 属性 | |
| width: float | 宽度 |
| height: float | 高度 |
| locateX: float | 位置横坐标 |
| locateY: float | 位置纵坐标 |
| color: int | 颜色 |
| compoID: int | 部件ID |
| 操作 | |
| click() | 单击 |
| slide() | 滑动 |

窗口部件包括菜单栏MenuBar类、边栏SideBar类、搜索栏SearchBar类和投票展示区VoteDisplay类，与窗口类形成聚合关系，而VoteItem类与VoteDisplay是聚合关系。

关联图：



1. **窗口中主要有以下各界面元素，也都继承自Component类**

|  |  |
| --- | --- |
| TextArea(文字) | |
| 属性 | |
| font: int | 字体 |
| fontSize: int | 字体大小 |
| fontColor: int | 字体颜色 |

|  |  |
| --- | --- |
| Button(按钮) | |
| 属性 | |
| picSource: string | 图片地址 |
| text: string | 按钮显示文字 |

|  |  |
| --- | --- |
| Picture(图片) | |
| 属性 | |
| source: string | 图片地址 |

|  |  |
| --- | --- |
| InputField(输入框) | |
| 属性 | |
| text: string | 输入提示信息 |

1. **这些界面元素与窗口对象类形成聚合关系**









## 控制驱动设计

系统驱动部分根据类图的主动类得出主要有三类图，在进入系统之前的**注册和登录**操作

进入系统之后的**用户的使用操作**以及管理员拥有的**删除操作**

### 根据类图的账户类

注册登录分别是两个进程

以注册操作为例：



在注册的过程中如果验证信息不符合或者用户选择注册按钮则进入到注册进程，否则调用显示投票列表的进程，也就是开始界面。

### 根据类图的用户类

用户操作主要有发布投票，参与投票，关注投票，关闭投票以及查看历史

1. 发布投票和关闭投票是用户主动操作的过程，所以应该作为独立的驱动流



在接受到用户其投票的请求之后应该修改投票类中的**所有投票**属性，也就是修改对应的数据库内容，然后再返回投票结果

关闭投票则应该在所有投票中找到对应投票，然后修改该实例的**关闭状态**

1. 参与投票有两种形式，分别是投票和评论

投票和评论都是参与的形式，在获取到用户的请求之后应该根据用户选定的投票找到相应的实例，前者应该修改其**已投人数**等属性，而后者应该修改投票参与记录类的实例的**评论**属性，

同时应该更新用户的历史信息以便于查询

1. 关注投票也是用户的主动操作，作为对类图的补充，增加取消关注的功能。



关注投票先要获取请求，然后找到对应的投票类的实例，然后讲对应投票类实例的ID信息加到用户以关注投票列表中，取消关注则是讲选定的投票ID移除关注列表

1. 查看历史需要区分用户选择的历史信息类型(比如参与历史，投票历史，或者评论历史等)



查看历史首先需要区分不同的查询信息，然后再根据不同的查询信息获取列表中对应投票实例，并以列表的形式返回。

1. 作为对关注功能补充，应该增加单独显示以关注投票的列表操作



### 根据类图的投票类

显示投票列表是主动操作，显示投票详情也是主动操作，而且在显示投票列表操作中应该包含可选的显示选定投票详情的操作。所以前者应该有调用后者的操作



显示投票列表应该作为显示主用户界面的过程发生，在用户选择查看详情之后应该调用显示投票详情操作。这两个过程都是返回数据库中记录的投票实例。

### 根据类图的管理员类

删除用户操作，删除投票操作以及删除投票参与记录的操作都应该是管理员的主动操作。



通过管理员主动发起删除操作，根据删除的用户的ID可以删除指定用户

同样也可以指定删除投票的ID，以及同时指定某个具体投票记录ID删除某个具体投票记录

### 补充说明

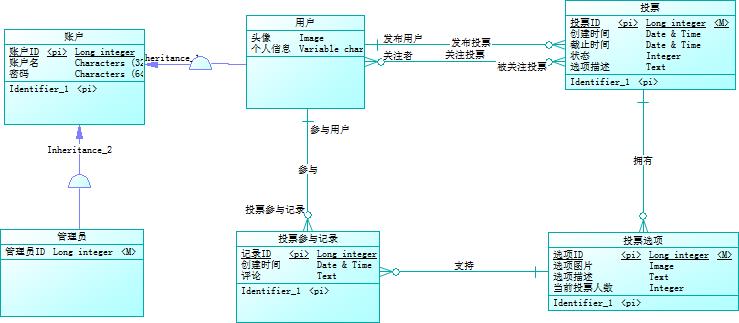
助教说如果是移动端开发的话应该使用线程，但是我们主要开发的是网页应用，另一方面，为了简单，会尽量减少并发。

不过用户类以及投票类的操作我都改成线程的了。因为考虑到对于用户而言，这些操作应该放在后台进程来做比较好，另一个显示进程应该保证在这个过程隐藏后台的操作，并在返回结果后显示结果。但是网页端开发应该不用这么做。

## 数据管理设计

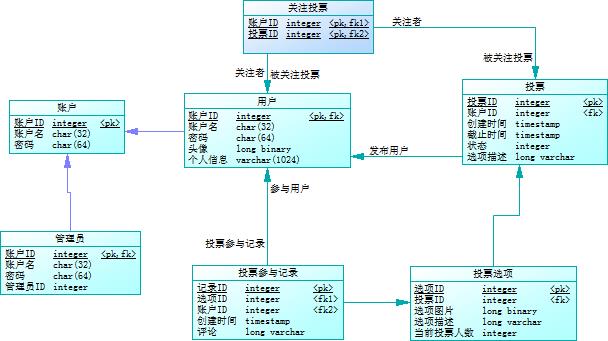
我们使用powerdesigner来进行数据管理设计。

其中coolchooser.cdm是整个系统的概念模型，基本依据类图实现的。着重表现每一个类以及其关系，反应了OOA的结果。



可以看到，这里的每一个对象都是对应着类图里的对象，类图所描述的每一个属性都在这里得到了体现。比如记录有ID，创建时间，评论等。我按照实际情况，添加了每个类的标识属性，也就是主码；同样按照实际情况添加它的数据类型。

将它转换到物理模型之后是coolchooser.pdm，这一步就是OOD的步骤——依照对象之间的关系，按照3NF建立合适的数据库表来存储它。图中有7个数据库表，其中每一个数据库表都对应了一个类或者关系。



要着重注意：

（1）为了简单化依赖关系，让每一个类的主码都是唯一标识符ID，这样表关系就会简单很多。

（2）像投票-投票选项，投票选项-投票记录，创建用户-投票，用户-投票记录，都是一对多关系且前者非空，所以在设计中后者的数据库中有前者的主码作为外键。

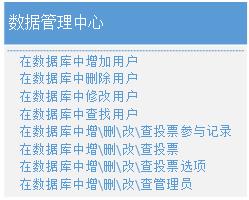
（3）关注关系是多对多关系，所以新建了一张表，让关联的两个类的主键作为外键，且一起成为这个表的主键。

（4）后来发现账户这个类是一个需要被虚继承的类，因为账户要么是用户要么是管理员。将其实现是冗余的。所以由一定理由不将其在作为数据库的一张表实现。

要注意到所有的类都对应着数据库中的一个表，所以增删查改都只要找到同名的对应表即可。

之后，将物理模型coolchooser.pdm导出至SQL，就得到了数据管理1.sql以及数据管理。因为后来指出这不是OOD的步骤，所以就不详细展示。

最后，建立一个类“”，用于统一管理整个数据库，想要增删改查每个类都可以通过这个类。这个类没有属性，只有很多个操作。见下：



具体内容见数据管理系统.vsdx