

**贵州大学计算机科学与信息学院**

**《Web系统开发》期末考查报告**

组 员 赵强、张帅、蒋日健、吴国豪、 刘杰

姓 名 赵强

学 号 1308060300

班 级 网络工程131班

完成 时间 2016-12-14

目 录

[1. 项目背景和意义 1](#_Toc23462)

[2. 需求分析 3](#_Toc16214)

[2.1 用户管理 3](#_Toc16570)

[2.2 功能模块 3](#_Toc1828)

[2.3 界面设计 3](#_Toc9300)

[3. 系统设计 4](#_Toc19180)

[3.1 用户注册模块： 4](#_Toc11866)

[3.2 用户登录模块： 5](#_Toc18782)

[3.3 系统主页模块： 5](#_Toc1944)

[3.4 个人言论模块： 6](#_Toc16777)

[3.5 发布帖子模块： 7](#_Toc3723)

[3.6 帖子详情模块： 8](#_Toc27894)

[4. 数据库结构设计 9](#_Toc8079)

[5. 系统模块界面设计 11](#_Toc932)

[5.1 登录页面： 11](#_Toc29558)

[5.2 注册页面： 11](#_Toc24769)

[5.3 系统主页： 12](#_Toc28620)

[5.4 帖子详情页： 13](#_Toc22104)

[5.5 个人言论页： 14](#_Toc24155)

[5.6 发布帖子页： 14](#_Toc28496)

[6. 功能实现 15](#_Toc28072)

[7. 总结 38](#_Toc6759)

## 项目背景和意义

随着互联网的兴起，Facebook在短时间内风靡全球，我国的腾讯QQ、微信、微博等更是在全国掀起热潮，社交网络（又称社交网络系统或社交网络服务）成为近年来最为流行的互联网应用之一，并被认为是继搜索和Web2.0之后互联网的又一次革命。由Boyd和Ellison提出的社交网络系统的定义是被学术界和工业界引用最多的定义之一。Boyd和Ellison认为：“社交网络系统是一种允许个人在有边界的系统中构建公开或者半空开的个人空间，阐明与自己有联系用户的列表，以及查看和遍历这一用户列表和列表中的用户发布的信息的网络服务”。社交网络系统强调用户的实名制和建立与用户之间真实社交关系相对应的网络联系；而且突出用户创造的内容，通过与真实社交关系相对应的网络联系的传播，以降低用户获取信息的成本，提高信息传播的有效性。

高校校园用户是互联网上最活跃的群体，是使社交网络系统得以发展和壮大的重要力量。国际和国内拥有众多独立的公共社交网络系统，例如Facebook,“人人网”等。虽然他们能够满足高校师生在人际沟通和群组沟通方面的需求，但是对在高校校园中的学习、工作和文化生活等方面的现实环境和真实活动紧密结合方面的支持却显得不足，也不能很好地支持不同学校在教学、科研和文化生活方面的个性化需求。而且，由于在定位和商业上的考虑，公共社交网络系统在关系模型和隐私保护等方面的设计也给用户带来了诸多使用上的不便和潜在的问题。并且在高校内部，专门提供给学生的社交平台相对缺失，对学生之间的信息交流和沟通带来了不便。因此，构建适合高校校园用户使用的、与校园的现实环境和真实活动紧密结合的、以学生用户体验为中心的校园社交网络系统是当前高校数字校园建设中一项重要的新内容。

基于flask-restful框架的校园社交平台为贵大学生提供了一个相对便捷的交流和信息共享平台，比常用的QQ、微信等更加公开和便利。其中的应用组件不是以游戏类的应用组件为主，而是直接服务于贵大师生的学习、工作和生活。该平台面向学生的常用需求设置了“校内信息”、“学习交流”、“吃喝玩乐”、“失物招领”四个模块，对信息进行分类，整合了相关信息资源，提高了学生对相关信息的查找效率，有利于服务高效师生以及高校校园中的学习、工作和文化交流等活动。与公共社交网络系统不同，基于flask-restful框架的校园社交平台不是独立存在的，它与其中已经成熟应用的主页系统、门户系统、账号管理与电子身份认证系统、电子校务系统等有着十分密切的联系。而且，高校校园社交网络系统中的同时，由于没有商业上的考虑，基于flask-restful框架的校园社交平台在关系模型的设计和隐私保护方面都能够为高校师生提供更为方便和放心的服务。

## 需求分析

需求分析是用对的、行的通、不可缺少等规则对系统做一个完整的功能分析说明。需求分析要求每个需求的功能必须描述清楚，确保在自用的知识能力和电脑配置条件下可以实现，并且每个需求，功能是否必须正确，是否可以以后实现，是否可以在减少成本的情况发生时删掉。

基于flask-restful框架的校园社交平台主要用于实现校内信息的交流和共享，主要以论坛发帖的形式实现信息的发布和交流。

#### 用户管理

基于flask-restful框架的校园社交平台包括两类用户：普通用户和系统管理员。普通用户可以实现帖子的查看、发布、评论和删除本人帖子的功能，管理员除可以实现帖子的查看、发布、评论和删除本人帖子的功能外，还可以删除他人帖子以及对帖子进行置顶。为保障系统的安全性，注册时只能实现普通用户的注册，管理员账号由后台数据库添加。

#### 功能模块

平台中的帖子信息包括校内信息、学习交流、吃喝玩乐、失物招领四类。每个帖子会显示发布者头像、用户名、帖子内容、图片和发布时间。平台支持帖子的发布者从本地上传图片、选择帖子信息分类，支持帖子的评论回复功能，以及帖子的查看、发布、评论和删除功能。

#### 界面设计

界面设计力求简约大方，突出消息内容，避免无用信息及其他界面设置的干扰，方便用户查看所有的网站信息和详细信息的内容，易于用户了解系统的功能，便于操作和使用。

## 系统设计

根据分析，该系统应该有两个角色：管理员、普通用户。系统需要赋予他们对应的功能和权限，具体如下：

管理员具有：查看帖子、发布帖子，删除帖子，置顶帖子，评论帖子的功能和权限。

普通用户具有：查看帖子，发布帖子，评论帖子的功能和权限。

根据系统的需求，该系统应该具有以下六个模块：用户注册模块，用户登录模块、系统主页模块、个人言论模块、发布帖子模块、帖子详情模块。各模块对应的基本功能如下：

#### 用户注册模块：

（1）用户输入用户名、密码、确认密码。

（2）系统检验用户名是否为空，是否可用，以及验证密码与确认密码是否一致，并给予用户相关消息提醒。

注册流程如图3-1所示：

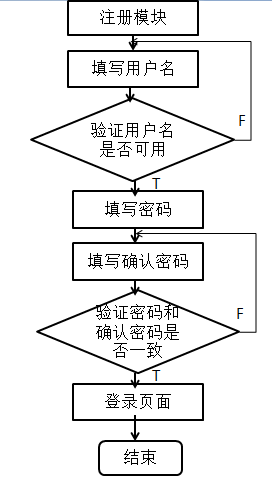


图3-1 注册模块

#### 用户登录模块：

（1）用户输入用户名和密码。

（2）系统检验用户名是否在数据库中，并检验用户名和密码是否一致，并给予用户相关消息提醒。

登录流程如图3-2所示：

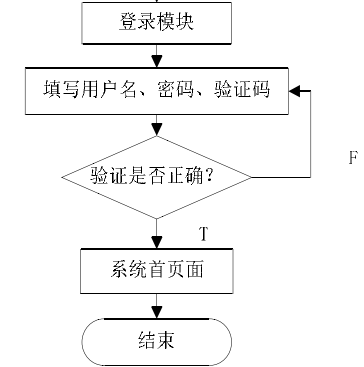


图3-2 登录模块

#### 系统主页模块：

（1） 按类别（“校内信息”、“学习交流”、“吃喝玩乐”、“失物招领”）显示帖子内容等信息。

1. 管理员可以对帖子进行置顶和删除。
2. 可点击帖子查看详情。
3. 可点击查看登录用户本人发表的帖子。
4. 可点击发表帖子。

系统主页模块如图3-3所示：

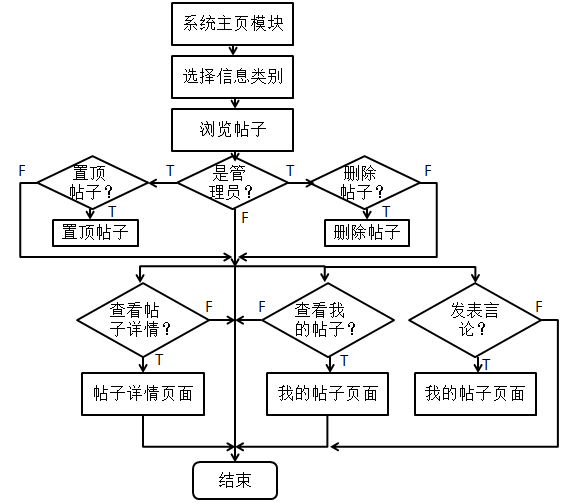


图3-3 系统主页模块

#### 个人言论模块：

1. 可以查看本人发表的所有言论。
2. 可对他人对自己帖子的评论进行回复。
3. 可以删除本人发表的帖子。

个人言论模块如图3-4所示：

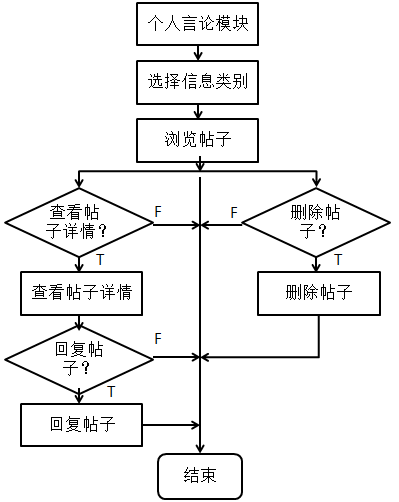


图3-4 个人言论模块

#### 发布帖子模块：

1. 可以输入帖子的主题和内容。
2. 可以从本地上传图片到帖子中。

（3） 可以从四个分类“校内信息”、“学习交流”、“吃喝玩乐”、“失物招领”中选择自己帖子的分类。

（4） 发表言论。

发布帖子模块如图3-5所示：

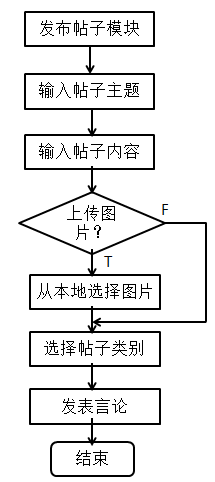


图3-5 发布帖子模块

#### 帖子详情模块：

1. 显示帖子发布者的头像、用户名和帖子内容、距发布日期的天数等。
2. 可以对帖子进行评论。

帖子详情模块如图3-6所示：

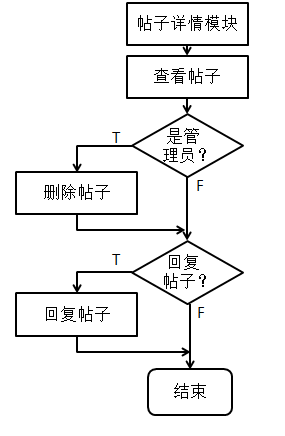


图3-6 帖子详情模块

## 数据库结构设计

数据库是整个系统的基础和核心,因此数据库设计的好坏将直接影响到整个系统完成的成败和好坏。制作数据库一开始要确定对象的属性和实体间的关联，根据关联制定数据表。系统的数据库表如以下表所示：

表4-1 用户表数据库设计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **英文列名** | **中文列名** | **类型** | **外键** | **备注** |
| user\_id | 用户编号 | Integer |  | 自增，主键 |
| user\_name | 用户名 | String(20) |  | 唯一性 |
| user\_password | 用户密码 | String(20) |  | 6到18位 |
| user\_privilege | 用户权限 | Integer |  | 0用户/1管理员 |

表4-2 用户认证表数据库设计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **英文列名** | **中文列名** | **类型** | **外键** | **备注** |
| usertoken\_str | 认证字符串 | String(100) |  | 主键 |
| usertoken\_userid | 用户编号 | Integer | user.user\_id |  |

表4-3 类别表数据库设计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **英文列名** | **中文列名** | **类型** | **外键** | **备注** |
| category\_id | 类别编号 | Integer |  | 自增，主键 |
| category\_name | 类别名称 | String(20) |  |  |

表4-4 帖子表数据库设计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **英文列名** | **中文列名** | **类型** | **外键** | **备注** |
| post\_id | 帖子编号 | Integer |  | 自增，主键 |
| post\_categoryid | 类别编号 | Integer | category.category\_id |  |
| post\_userid | 用户编号 | Integer | user.user\_id |  |
| post\_title | 帖子主题 | String(500) |  |  |
| post\_time | 帖子时间 | String(100) |  |  |
| post\_content | 帖子内容 | String(2000) |  |  |

表4-5 评论表数据库设计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **英文列名** | **中文列名** | **类型** | **外键** | **备注** |
| comment\_id | 评论编号 | Integer |  | 1. 自增,主键 |
| comment\_postid | 帖子编号 | Integer | post.post\_id |  |
| comment\_userid | 用户编号 | Integer | user.user\_id |  |
| comment\_text | 评论内容 | String(2000) |  |  |
| comment\_datetime | 评论日期 | String(100) |  |  |
| comment\_parentid | 父评论编号 | Integer | comment.comment\_id |  |

表4-6 图片表数据库设计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **英文列名** | **中文列名** | **类型** | **外键** | **备注** |
| picture\_id | 图片编号 | Integer |  | 自增,主键 |
| picture\_name | 图片名称 | String(200) |  |  |
| picture\_url | 图片位置 | String(200) |  |  |
| picture\_postid | 帖子编号 | Integer | post.post\_id |  |

表4-1 置顶帖子表数据库设计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **英文列名** | **中文列名** | **类型** | **外键** | **备注** |
| top\_id | 置顶编号 | Integer |  | 自增,主键 |
| top\_postid | 置顶帖子编号 | Integer | post.post\_id |  |
| top\_time | 置顶时间 | String(100) |  |  |
| top\_categoryid | 帖子类型编号 | Integer | category.category\_id |  |

## 系统模块界面设计

#### 登录页面：

无论是网站还是客 户端的系统，登录是最基本的功能模块，它能保证系统的稳定，系统数据的安全，也能够保证用户的权限，为用户分配不同的功能。登录功能的实现也很常规简单。 当用户在页面输入信息然后登陆后，系统会做出判断，将用户输入的信息和数据库表中的信息做比较，如果信息错误或不存在就不能够登录，如果正确就能顺利登录。登录模块如图5-1所示：

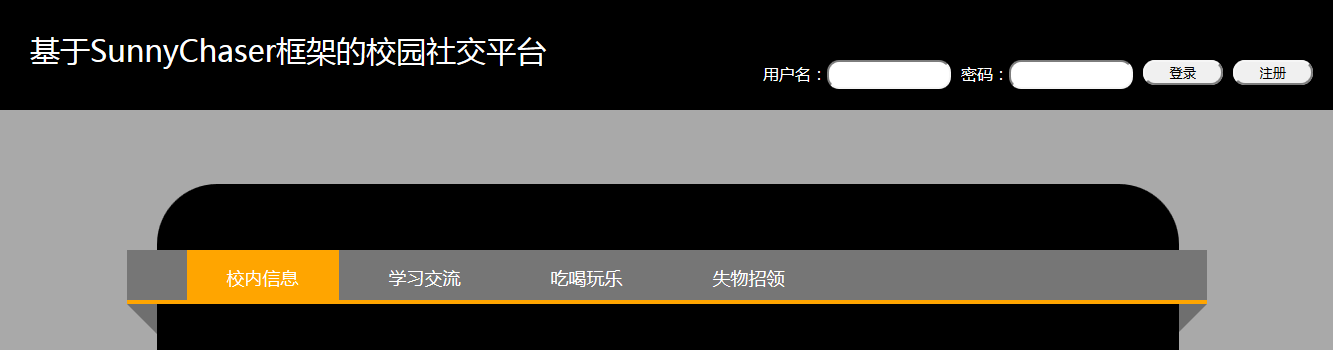


图5-1 登录页面

#### 注册页面：

此模块的功能是用户进行注册，用户添入相关的个人信息。用户输入信息后点击“注册”按钮，用户输入的信息就会从页面传到页面的后台去处理，其中的处理包括很多，为了保证用户唯一性，还要对传过来的数据进行判断，数据表中如果有相同的用户就不可以让他注册，否则会造成数据混乱。用户注册界面如图5-2所示：



图5-2 注册页面

#### 系统主页：

系统主页是主要部分，当用户输入正确的用户名、密码，并核对正确后，系统自动跳转到如下页面。在该页面中，用户可以查看帖子发布者的用户名、头像、发布日期和帖子的简要内容，对感兴趣的帖子可以点击进入查看详情。管理员用户可以对帖子进行置顶和删除。系统主页页面如图5-3所示：



图5-2 系统主页页面

#### 帖子详情页：

帖子详情页可以详细显示帖子内容，以及帖子作者用户名、头像、帖子图片，并可以在帖子下方的评论框输入评论内容发布，与作者进行互动。帖子详情页面如图5-4和图5-5所示：



图5-4 帖子详情页页面1



图5-5 帖子详情页页面2

#### 个人言论页：

个人言论页可以显示用户本人所发布的所有言论，用户可以查看帖子详情、删除帖子、对他人的评论进行回复等。个人言论页页面如图5-6所示：



图5-6 个人言论页页面

#### 发布帖子页：

用户可以在发布帖子页输入帖子的主题、内容，可以从本地上传图片，选择帖子内容类别，发布帖子。发布帖子页页面如图5-7所示：



图5-7 发布帖子页页面

## 功能实现

打开首页，可以看到首页显示的已发布的帖子内容（默认呈现“校内信息模块内容”），如图6-1所示：



图6-1 登录前的校内信息帖子列表

注册用户。输入用户名、密码、确认密码，系统验证后即可完成注册（以用户Mary为例），如图6-2所示：



图6-2 用户注册

def adduser(self,username, userpassword):

session = DBSession()

try:

if len(username) > 16:

return Info(False, '用户名不合法', None).tojson()

if session.query(User).filter\_by(user\_name=username).count() > 0:

session.close()

return Info(False, '用户名已被占用', None).tojson()

else:

session.add(User(user\_name = username,

user\_password = userpassword,

user\_privilege = 0,

user\_userlogo = "default.jpg",

))

session.commit()

session.close()

return Info(True, '注册成功',None).tojson()

except Exception as a:

print(a)

session.rollback()

session.close()

return Info(False, '数据库错误', None).tojson()

登录系统。输入用户名和密码，选择登录身份，系统验证正确后即可完成登录，如图6-3所示。登录后显示校内信息帖子列表，如图6-4所示：

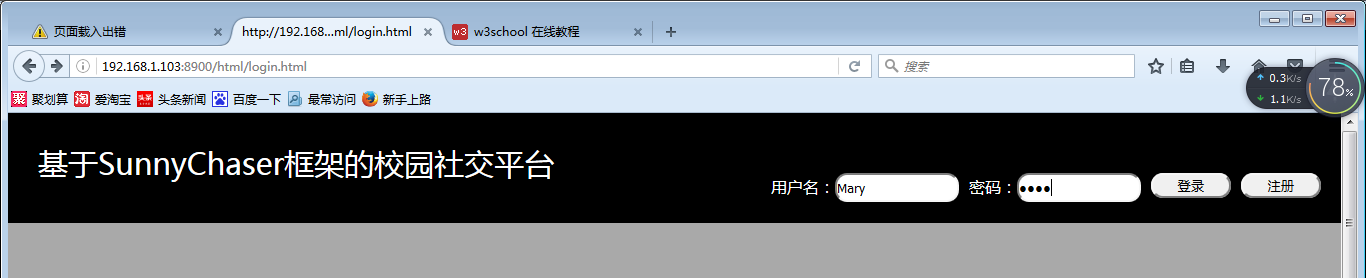


图6-3 用户登录

def checklogin(self, username, password):

session = DBSession()

try:

if session.query(User).filter\_by(user\_name=username, user\_password=password).count() > 0:

user = session.query(User.user\_id,User.user\_privilege).filter\_by(

user\_name=username).all()

userid = user[0][0]

userprivilege = user[0][1]

usertoken = str(uuid.uuid4())

userdict = {}

userdict['usertoken\_str'] = usertoken

userdict['user\_privilege'] = userprivilege

if session.query(Usertoken).filter\_by(usertoken\_userid=userid).count() > 0:

session.query(Usertoken).filter\_by(usertoken\_userid=userid).delete()

session.add(Usertoken(usertoken\_str=usertoken, usertoken\_userid=userid))

session.commit()

session.close()

else:

session.add(Usertoken(usertoken\_str=usertoken,usertoken\_userid=userid))

session.commit()

session.close()

return Info(True, '登录成功', userdict).tojson()

else:

session.close()

return Info(False, '用户名或密码错误', None).tojson()

except Exception as a:

print(a)

session.close()

return Info(False, '数据库错误', None).tojson()



图6-4 登录后的校内信息帖子列表

在首页上可以查看帖子简要内容，可以点击导航栏查看不同类别的消息，包括“校内信息”、“学习交流”、“吃喝玩乐”、“失物招领”，信息内容分别如图6-5,图6-6，图6-7，图6-8所示：



图6-5 校内信息帖子列表



图6-6 学习交流帖子列表



图6-7 吃喝玩乐帖子列表



图6-8 失物招领帖子列表

def postlist(self,startposi,pagesize,categoryname):

session = DBSession()

try:

category\_id = session.query(Category.category\_id).filter\_by(category\_name=categoryname).all()

categoryid = category\_id[0][0]

listdict = []

if int(startposi) > 0 :

postlist = session.query(Post.post\_title,Post.post\_time,Post.post\_content,Post.post\_userid,Post.post\_id).filter\_by(post\_categoryid=categoryid).order\_by(desc(Post.post\_time)).offset(startposi).limit(pagesize).all()

if len(postlist) == 0 :

session.close()

return Info(True,0,"已经没有帖子了").tojson()

for lista in postlist:

dicta = {}

dicta["post\_id"] = lista[4]

dicta["post\_title"] = lista[0]

dicta["post\_time"] = lista[1]

dicta["post\_content"] = lista[2]

user\_name = session.query(User.user\_name,User.user\_userlogo).filter\_by(user\_id=lista[3]).all()

username = user\_name[0][0]

dicta["user\_name"] = username

userlogo = user\_name[0][1]

userlogo = 'http://' + str(staticserver) + ":" + str(staticport) + '/logo/' + userlogo

dicta["post\_userhead"] = userlogo

if session.query(Toppost).filter\_by(top\_postid=lista[4]).count() == 0:

dicta["post\_topstatu"] = "0"

else:

dicta["post\_topstatu"] = "1"

listdict.append(dicta)

session.close()

return Info(True,0,listdict).tojson()

toplistid = session.query(Toppost.top\_postid).filter\_by(top\_categoryid=categoryid).order\_by(Toppost.top\_time).offset(startposi).limit(pagesize).all()

toplist\_id = []

for postid in toplistid:

toplist\_id.append(postid)

dict1 = {}

postdetail = session.query(Post.post\_title,Post.post\_time,Post.post\_content,Post.post\_userid,Post.post\_id).filter\_by(post\_id=postid[0]).all()

dict1["post\_id"] = postdetail[0][4]

dict1["post\_title"] = postdetail[0][0]

dict1["post\_time"] = postdetail[0][1]

dict1["post\_content"] = postdetail[0][2]

user\_name = session.query(User.user\_name,User.user\_userlogo).filter\_by(user\_id=postdetail[0][3]).all()

username = user\_name[0][0]

dict1["user\_name"] = username

userlogo = user\_name[0][1]

if not userlogo:

suserlogo = "default.jpg"

userlogo = 'http://' + str(staticserver) + ":" + str(staticport) + '/logo/' + userlogo

dict1["post\_userhead"] = userlogo

if session.query(Toppost).filter\_by(top\_postid=postdetail[0][4]).count() == 0:

dict1["post\_topstatu"] = "0"

else:

dict1["post\_topstatu"] = "1"

listdict.append(dict1)

listdictlen = len(listdict)

if len(listdict) != 0:

pagesizeafter = int(pagesize) - int(listdictlen)

else:

pagesizeafter = int(pagesize)

postlist = session.query(Post.post\_title,Post.post\_time,Post.post\_content,Post.post\_userid,Post.post\_id).filter\_by(post\_categoryid=categoryid).order\_by(desc(Post.post\_time)).offset(str(startposi)).limit(str(pagesizeafter)).all()

if len(postlist) == 0 :

session.close()

return Info(True,listdictlen,listdict).tojson()

for lista in postlist:

dicta = {}

dicta["post\_id"] = lista[4]

dicta["post\_title"] = lista[0]

dicta["post\_time"] = lista[1]

dicta["post\_content"] = lista[2]

user\_name = session.query(User.user\_name,User.user\_userlogo).filter\_by(user\_id=lista[3]).all()

username = user\_name[0][0]

dicta["user\_name"] = username

userlogo = user\_name[0][1]

userlogo = 'http://' + str(staticserver) + ":" + str(staticport) + '/logo/' + userlogo

dicta["post\_userhead"] = userlogo

if session.query(Toppost).filter\_by(top\_postid=lista[4]).count() == 0:

dicta["post\_topstatu"] = "0"

else:

dicta["post\_topstatu"] = "1"

listdict.append(dicta)

listdictafter = []

lengh = int(pagesize) - int(startposi)

for a in listdict:

if a not in listdictafter:

listdictafter.append(a)

listdict = listdictafter

if len(listdict) < lengh:

session.close()

return Info(True,listdictlen,listdict).tojson()

for a in listdict:

if a not in listdictafter:

listdictafter.append(a)

allpagesize = session.query(Post).filter\_by(post\_categoryid=categoryid).count()

while len(listdictafter) < int(pagesize) and len(listdictafter) < allpagesize:

b = int(startposi) + len(listdictafter)

c = int(pagesize) - len(listdictafter)

postlistb = session.query(Post.post\_title,Post.post\_time,Post.post\_content,Post.post\_userid,Post.post\_id).filter\_by(post\_categoryid=categoryid).order\_by(desc(Post.post\_time)).offset(str(b)).limit(str(c)).all()

listdict = listdictafter

for lista in postlistb:

dicta = {}

dicta["post\_id"] = lista[4]

dicta["post\_title"] = lista[0]

dicta["post\_time"] = lista[1]

dicta["post\_content"] = lista[2]

user\_name = session.query(User.user\_name,User.user\_userlogo).filter\_by(user\_id=lista[3]).all()

username = user\_name[0][0]

dicta["user\_name"] = username

userlogo = user\_name[0][1]

userlogo = 'http://' + str(staticserver) + ":" + str(staticport) + '/logo/' + userlogo

dicta["post\_userhead"] = userlogo

if session.query(Toppost).filter\_by(top\_postid=lista[4]).count() == 0:

dicta["post\_topstatu"] = "0"

else:

dicta["post\_topstatu"] = "1"

listdict.append(dicta)

listdictafter = []

for a in listdict:

if a not in listdictafter:

listdictafter.append(a)

session.close()

return Info(True,listdictlen,listdictafter).tojson()

except Exception as a:

print(a)

session.close()

return Info(False, '数据库错误', None).tojson()

管理员用户（以tom为例。账号为后台添加）登录时，页面会显示“置顶”和“删除”按钮，如图6-9所示。管理员可以对帖子进行删除或置顶（或取消置顶），如图6-10，图6-11所示：



图6-9 显示“置顶”和“删除”按钮



图6-10 置顶帖子

def posttop(self,usertoken,postid,categoryname):

session = DBSession()

try:

if session.query(Usertoken).filter\_by(usertoken\_str=usertoken).count() == 0:

return Info(False, '无效的用户，登陆断开连接', None).tojson()

user\_id = session.query(Usertoken.usertoken\_userid).filter\_by(usertoken\_str=usertoken).all()

userid =user\_id[0][0]

user\_privilege = session.query(User.user\_privilege).filter\_by(user\_id=userid).all()

userprivilege = user\_privilege[0][0]

if userprivilege == 0:

return Info(False, '用户没有操作权限', None).tojson()

category\_id = session.query(Category.category\_id).filter\_by(category\_name=categoryname).all()

categoryid = category\_id[0][0]

topposttime = time.strftime("%Y-%m-%d %H:%M:%S",time.localtime())

if session.query(Toppost).filter\_by(top\_postid=postid).count() > 0:

return Info(False, '该帖子已经置顶，不能重复置顶帖子', None).tojson()

session.add(Toppost(top\_postid=postid,

top\_time=topposttime,

top\_categoryid=categoryid))

session.commit()

session.close()

return Info(True,'置顶成功',None).tojson()

except Exception as a:

print(a)

session.rollback()

session.close()

return Info(False, '数据库错误', None).tojson()



图6-11 删除帖子

def postdelete(self,usertoken,postid):

session = DBSession()

try:

if session.query(Usertoken).filter\_by(usertoken\_str=usertoken).count() == 0:

return Info(False, '无效的用户，登陆断开连接', None).tojson()

user\_id = session.query(Usertoken.usertoken\_userid).filter\_by(usertoken\_str=usertoken).all()

userid1 = user\_id[0][0]

if session.query(Post).filter\_by(post\_id=postid).count() == 0:

return Info(False, '没有该帖子', None).tojson()

user\_id2 = session.query(Post.post\_userid).filter\_by(post\_id=postid).all()

userid2 = user\_id2[0][0]

user\_privilege = session.query(User.user\_privilege).filter\_by(user\_id=userid1).all()

userprivilege = user\_privilege[0][0]

if userid1 == userid2 or userprivilege == 1:

if session.query(Picture).filter\_by(picture\_postid=postid).count() > 0:

session.query(Picture).filter\_by(picture\_postid=postid).delete()

if session.query(Comment).filter\_by(comment\_postid=postid).count() > 0 :

session.query(Comment).filter\_by(comment\_postid=postid).delete()

if session.query(Toppost).filter\_by(top\_postid=postid).count() > 0:

session.query(Toppost).filter\_by(top\_postid=postid).delete()

session.query(Post).filter\_by(post\_id=postid).delete()

session.commit()

session.close()

return Info(True,'删除帖子成功',None).tojson()

else:

return Info(False, '没有操作的权限', None).tojson()

except Exception as a:

print(a)

session.rollback()

session.close

return Info(False, '数据库错误', None).tojson()

查看帖子详情：点击某帖子即可跳转到如下帖子详情页面，可以查看帖子详细信息（包括图片），如图6-12，图6-14所示，并在帖子下方发布评论内容，如图6-13所示：



图6-12 帖子详情



图6-13 评论帖子

def postiddetail(self,postid):

session = DBSession()

try:

postlist = session.query(Post.post\_userid,Post.post\_title,Post.post\_content,Post.post\_time).filter\_by(post\_id=postid).all();

postdetail = {}

postdetail["post\_id"] = postid

postdetail["post\_userid"] = postlist[0][0]

userdetail = session.query(User.user\_name,User.user\_userlogo).filter\_by(user\_id=postlist[0][0]).all()

postdetail["post\_username"] = userdetail[0][0]

userlogo = userdetail[0][1]

if not userlogo:

userlogo = "default.jpg"

userlogo = 'http://' + str(staticserver) + ":" + str(staticport) + '/logo/' + userlogo

postdetail["post\_userhead"] = userlogo

postdetail["post\_title"] = postlist[0][1]

postdetail["post\_content"] = postlist[0][2]

tt = Timetools()

postdetail["post\_time"] = tt.timetonow(postlist[0][3])

picturelist = []

picture = session.query(Picture.picture\_url).filter\_by(picture\_postid=postid).all()

for pic in picture:

pictureurl = 'http://' + str(staticserver) + ":" + str(staticport) + '/pics/' + pic[0]

picturelist.append(pictureurl)

postdetail["post\_pics"] = picturelist

print(postdetail["post\_pics"])

comment = session.query(Comment.comment\_id,Comment.comment\_userid,Comment.comment\_text,Comment.comment\_datetime).filter\_by(comment\_postid=postid).order\_by(desc(Comment.comment\_datetime)).all()

commentlist = []

for y in comment:

commentdict = {}

commentdict['comment\_id'] = y[0]

commentdict['comment\_userid'] = y[1]

username1 = session.query(User.user\_name).filter\_by(user\_id=y[1]).all()

username1 = username1[0][0]

commentdict['comment\_username'] = username1

userlogo = session.query(User.user\_userlogo).filter\_by(user\_id=y[1]).all()

userlogo = userlogo[0][0]

if not userlogo:

userlogo = "default.jpg"

userlogo = 'http://' + staticserver + ':' + staticport +'/logo/'+userlogo

commentdict['comment\_userhead'] = userlogo

commentdict['comment\_text'] = y[2]

commentdict['comment\_datetime'] = y[3]

commentlist.append(commentdict)

commentlist.sort(key = lambda x : x['comment\_datetime'],reverse=False)

for x in commentlist:

x['comment\_datetime'] = tt.timetonow(x['comment\_datetime'])

postdetail['post\_comments'] = commentlist

return Info(True,'返回成功',postdetail).tojson()

session.commit()

session.close()

except Exception as a:

print(a)

session.close()

return Info(False, '数据库错误', None).tojson()



图6-14 含图片的帖子详情

def postiddetail(self,postid):

session = DBSession()

try:

postlist = session.query(Post.post\_userid,Post.post\_title,Post.post\_content,Post.post\_time).filter\_by(post\_id=postid).all();

postdetail = {}

postdetail["post\_id"] = postid

postdetail["post\_userid"] = postlist[0][0]

userdetail = session.query(User.user\_name,User.user\_userlogo).filter\_by(user\_id=postlist[0][0]).all()

postdetail["post\_username"] = userdetail[0][0]

userlogo = userdetail[0][1]

if not userlogo:

userlogo = "default.jpg"

userlogo = 'http://' + str(staticserver) + ":" + str(staticport) + '/logo/' + userlogo

postdetail["post\_userhead"] = userlogo

postdetail["post\_title"] = postlist[0][1]

postdetail["post\_content"] = postlist[0][2]

tt = Timetools()

postdetail["post\_time"] = tt.timetonow(postlist[0][3])

picturelist = []

picture = session.query(Picture.picture\_url).filter\_by(picture\_postid=postid).all()

for pic in picture:

pictureurl = 'http://' + str(staticserver) + ":" + str(staticport) + '/pics/' + pic[0]

picturelist.append(pictureurl)

postdetail["post\_pics"] = picturelist

print(postdetail["post\_pics"])

comment = session.query(Comment.comment\_id,Comment.comment\_userid,Comment.comment\_text,Comment.comment\_datetime).filter\_by(comment\_postid=postid).order\_by(desc(Comment.comment\_datetime)).all()

commentlist = []

for y in comment:

commentdict = {}

commentdict['comment\_id'] = y[0]

commentdict['comment\_userid'] = y[1]

username1 = session.query(User.user\_name).filter\_by(user\_id=y[1]).all()

username1 = username1[0][0]

commentdict['comment\_username'] = username1

userlogo = session.query(User.user\_userlogo).filter\_by(user\_id=y[1]).all()

userlogo = userlogo[0][0]

if not userlogo:

userlogo = "default.jpg"

userlogo = 'http://' + staticserver + ':' + staticport +'/logo/'+userlogo

commentdict['comment\_userhead'] = userlogo

commentdict['comment\_text'] = y[2]

commentdict['comment\_datetime'] = y[3]

commentlist.append(commentdict)

commentlist.sort(key = lambda x : x['comment\_datetime'],reverse=False)

for x in commentlist:

x['comment\_datetime'] = tt.timetonow(x['comment\_datetime'])

postdetail['post\_comments'] = commentlist

return Info(True,'返回成功',postdetail).tojson()

session.commit()

session.close()

except Exception as a:

print(a)

session.close()

return Info(False, '数据库错误', None).tojson()

发表言论：点击主页右上方的“发表言论”，进入发表言论页面，输入相关内容，并选择帖子类别（以“学习交流”为例），如图6-15所示：

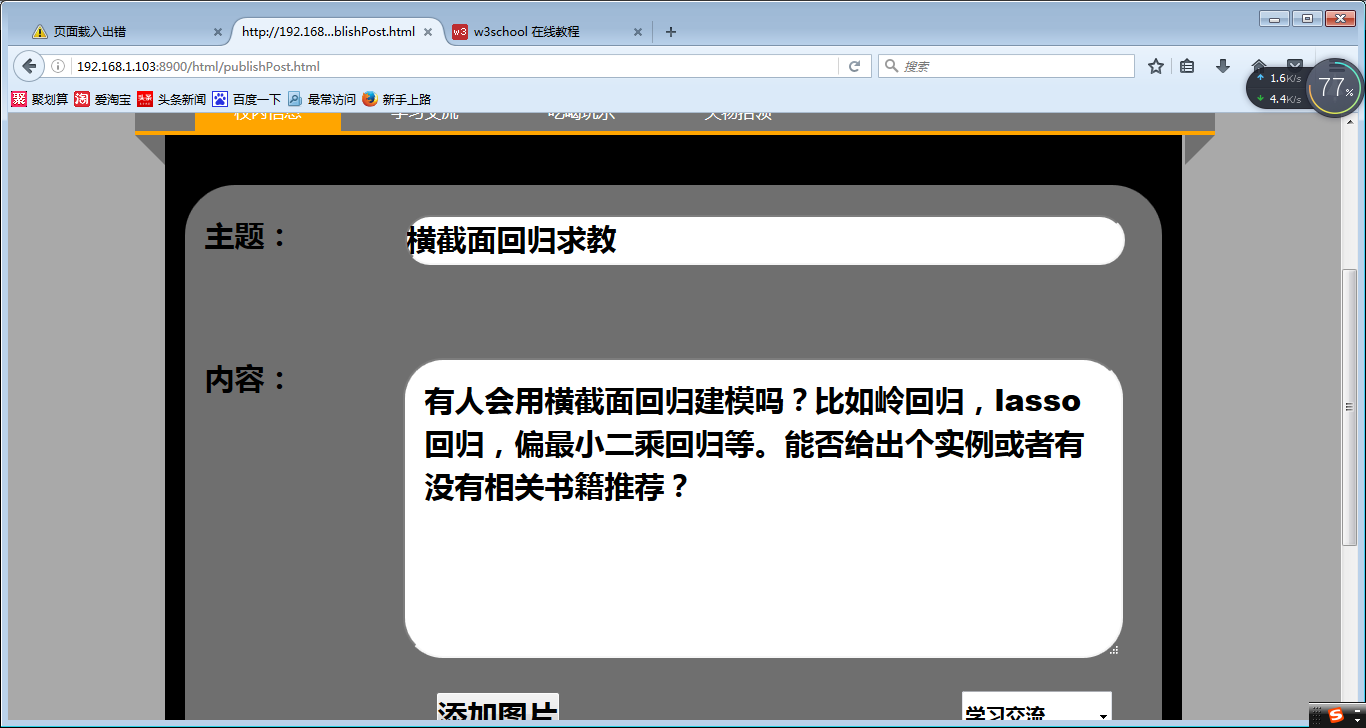


图6-15 发表言论

def postappend(self,usertoken,posttitle,postcontent,categoryname,postpics):

session = DBSession()

try:

if session.query(Usertoken).filter\_by(usertoken\_str=usertoken).count() == 0:

return Info(False, '无效的用户，登陆断开连接', None).tojson()

user\_id = session.query(Usertoken.usertoken\_userid).filter\_by(usertoken\_str=usertoken).all()

userid = user\_id[0][0]

category\_id = session.query(Category.category\_id).filter\_by(category\_name=categoryname).all()

categoryid = category\_id[0][0]

posttime = time.strftime("%Y-%m-%d %H:%M:%S",time.localtime())

session.add(Post(post\_categoryid=categoryid,

post\_userid=userid,

post\_title=posttitle,

post\_time=posttime,

post\_content=postcontent))

post\_id = session.query(Post.post\_id).filter\_by(post\_userid=userid,post\_title=posttitle,post\_time=posttime).all()

postid = post\_id[0][0]

postpics = demjson.decode(postpics)

for pics in postpics:

picslist = pics.split('/')

picsname = picslist[-1]

session.add(Picture(picture\_name=picsname,

picture\_postid=postid,

picture\_url=picsname))

session.commit()

session.close()

return Info(True,'发布成功',None).tojson()

except Exception as a:

print(a)

session.rollback()

session.close

return Info(False, '数据库错误', None).tojson()

点击发布即可成功发布消息。在“学习交流”中即可看到自己发表的言论，如图6-16所示：



图6-16 显示发表言论

查看用户本人发表的言论：点击首页右上方的“我的言论”按钮，即可跳转到个人言论页面，如图6-17所示，可以查看到本人发布的所有言论，对言论进行删除、回复等，如图6-18所示：



图6-17 我的言论

def mypostlist(self,startposi,pagesize,usertoken,categoryname):

session = DBSession()

try:

if session.query(Usertoken).filter\_by(usertoken\_str=usertoken).count() == 0:

return Info(False, '无效的用户，登陆断开连接', None).tojson()

user\_id = session.query(Usertoken.usertoken\_userid).filter\_by(usertoken\_str=usertoken).all()

userid = user\_id[0][0]

user\_logo = session.query(User.user\_userlogo).filter\_by(user\_id = userid).all()

userlogo = user\_logo[0][0]

if not userlogo:

userlogo = "default.jpg"

userlogo = 'http://' + str(staticserver) + ":" + str(staticport) + '/logo/' + userlogo

category\_id = session.query(Category.category\_id).filter\_by(category\_name=categoryname).all()

categoryid = category\_id[0][0]

postlist = session.query(Post.post\_title,Post.post\_time,Post.post\_content,Post.post\_categoryid,Post.post\_id).filter\_by(post\_userid=userid,post\_categoryid=categoryid).order\_by(desc(Post.post\_time)).offset(startposi).limit(pagesize).all()

listpost = []

for postdetail in postlist:

postdict = {}

postdict["post\_id"] = postdetail[4]

postdict["post\_title"] = postdetail[0]

postdict["post\_content"] = postdetail[2]

user\_name = session.query(User.user\_name).filter\_by(user\_id=userid).all()

username = user\_name[0][0]

postdict["user\_name"] = username

tt = Timetools()

postdict["post\_time"] = tt.timetonow(postdetail[1])

category\_name = session.query(Category.category\_name).filter\_by(category\_id=postdetail[3]).all()

categoryname = category\_name[0][0]

postdict["category\_name"] = categoryname

postdict["post\_userhead"] = userlogo

picturelist = []

picture = session.query(Picture.picture\_url).filter\_by(picture\_postid=postdetail[4]).all()

for pic in picture:

pictureurl = 'http://' + str(staticserver) + ":" + str(staticport) + '/pics/' + pic[0]

picturelist.append(pictureurl)

postdict["picture\_url"] = picturelist

listpost.append(postdict)

session.close()

return Info(True,'返回我发布的帖子列表成功',listpost).tojson()

except Exception as a:

print(a)

session.close()

return Info(False, '数据库错误', None).tojson()



图6-1 我的言论详情及权限

## 总结

基于flask-restful框架的校园社交平台基本实现了贵大师生对信息的交流和共享，对交流信息进行分类，使信息查找和交流更加便捷。将失物招领部分单独列出，为学生物品丢失和认领提供了信息交流的平台，便于学生寻找遗失物品及失主。与商业化的社交平台相比，基于flask-restful框架的校园社交平台相对保护了用户的隐私，较好地服务了校内师生。

在本次项目中，我参与了后端代码的开发和整体系统的分析与部署。在本次项目中我们使用的flask-restful框架是属于一种轻量级的Web开发框架，方便于灵活且快速的开发Web项目，总框架的使用上让我对常用Web框架有了一定的了解。作为团队里后端开发的队员，我在功能上使用了或多或少的算法，提高了我对算法的应用水平，使得我在往后的算法应用中多了一份从容与自信。在我们的框架中，前端用API与后端交互，后端用flask的库sqlalchemy与数据库交互，所以后端是整个框架的枢纽，也是根基，所以大部分框架的部署都是在后端上，这也使我对框架的使用最为熟练，也得到了最好的锻炼。通过本次项目的锻炼，让我在今后的Web开发中能够有更多的经验与兴趣，也让我在开发的过程中多了一份自信。