1. 鹌鹑现在注册方式之一是邮箱注册 (http://www.ermiao.com/accounts/register) 这种情况注册后用户暂时没有头像 为了提供更好的体验 我们决定使用gravatar (https://cn.gravatar.com/) 请参照邮箱注册页面 模拟一个注册流程并从gravatar中取头像;

用户模型如下
User
id | avatar | email | password | username | created_date
len(password) >= 6

解决方案:
使用django

```
from base.models import User                                        //从base这个app中导入User数据模型（我一般会单独创建一个base app来存放model）
from django.template.loader import get_template         //导入get_template函数，用来获取html模板
from django.shortcuts import render_to_response         //导入render_to_response函数，用来渲染模板并作为http请求的返回值
from hashlib import md5                                                     //导入md5函数，用来计算用户的gravatar头像url
from datetime import *                                                      //导入datetime模块，用来计算用户注册日期

gravatar_url = "http://www.gravatar.com/avatar/"        //存储gravatar的固定url部分，用于生成完整的url

def register(request):                                                      //注：需要在urls.py中导入register函数并添加一行(r'^/accounts/register/$',register), 保证注册页面被路由至register函数处理
        new_username = request.POST['username']
        new_userid = request.POST['userid']
        new_avatar = md5(request.POST['avatar']).hexdigest()    //这里只存储了用户gravatar邮箱的md5值，考虑到如果存一个完整的url会占用很多不必要的数据库空间，并且gravatar的url应该不会随意更改，所以将url里md5前面的部分赋值成变量，使用时与gravatar_url变量连接得到完整url
        new_email = request.POST['email']                                   //这里其实可以考虑和avatar部分合并，一般人常用邮箱是一样的。更好的办法或许是在前端通过已经填写的email自动补全avatar输入框，如果用户有需要再进行修改。
        new_password = md5(request.POST['password'])                //不直接存储用户密码，采用md5加密，保护隐私
        new_created_date = date.today() //用户注册日期
        if len(password)>=6:
                newuser = User.objects.create(username = new_username,
                                                                                userid = new_userid,
                                                                                avatar = new_avatar,
                                                                                email = new_email,
                                                                                password = new_password,

created_date = new_created_date
                                                                                )
                                        //将用户数据存入数据库
        else:
                return render_to_response('wrongpwd.html',locals())             //跳转到密码错误页面（或者返回一些错误提示）
        return render_to_response('afterregister.html',locals())        //跳转到afterregister页面
```

注：
html中使用avatar的方法：
1、html模板中添加：
```
<img src="{{ avatar_url }}" />
```
2、views.py中对应函数添加：

```
..省略前后代码..
avatar_url = gravatar_url + someuser.avatar
..省略前后代码..
return render_to_response('someone.html',locals())
```

2. 有童鞋反映鹌鹑未登录时首页 (http://www.ermiao.com/) 展示的都是猫猫 都没有其他宠物 为了展现我们的大爱 我们决定修改首页展示数据内容 筛选出最近90天内用户发布的所有相片 其中40%的宠物属于猫，40%的宠物不属于猫，20%的内容属于5天之内喜欢数最多的

首页对象模型
```
Photo
id | image_uri | pet | user | text | like_count | created_data
Pet
id | name | avatar_uri | user | type | created_data
pet.type in ['cat', 'dog', 'other']
```

解决方案：

```
from base.models import Photo,Pet
from datetime import *

...省略前后代码...
the_number = 100                                    //首页显示的数量
before_90_days = date.today()-timedelta(90)
                                                        //90天之前的datetime
before_5_days = date.today()-timedelta(5)
all_in_last90days = Photo.objects.filter(created_data__gte=today_date)
                //获取90天内所有photo
cats_in_last90days = all_in_last90days.filter(pet__type="cat").order_by('?')            //
获取最近90天的猫的photo，因为发现首页的图片是固定数量的，所以获取的数据不可能全部显示，使用随机排序来使每次展示的内容不同。
others_in_last90days = all_in_last90days.filter(pet__type__ne="cat").order_by('?')      //
获取最近90天的不是猫的photo，因为发现首页的图片是固定数量的，所以获取的数据不可能全部显示，使用随机排序来使每次展示的内容不同。
most_like_count = all_in_last90days.filter(created_data__gte=before_5_days).order_by('-
like_count') //获取最近5天的photo并按like_count降序排列
cats_in_last90days = cats_in_last90days[0:int(the_number*0.4)-1]                         //
40%猫
others_in_last90days = others_in_last90days[0:int(the_number*0.4)-1]            //40%不是猫
most_like_count = most_like_count[0:int(the_number*0.2)-1]
        //20%5天内喜欢数最多
final_result = cats_in_last90days+others_in_last90days+most_like_count          //最终结果
...省略前后代码...
```

说明：
1、因为看到鹌鹑首页图片是固定数量的，所以采用了这种方式。
2、首页也可以考虑采用瀑布流的方式，滚动到底端自动加载。不过如果采用瀑布流就不能使用这种筛选方法了。应该先判断三种情况哪种相对数量最小，然后以此为基准算出另两种的数量，然后再截取。这样可以保证4:4:2的比例。
3、之所以先获取90天所有数据再筛选猫、非猫、5天内喜欢数最多，是为了优化性能。如果直接从数据库获取数据，需要访问三次数据库，但是优化后只需要访问一次。

ps:
由于没有实际环境所以没办法测试，可能有一些写错的地方。。。

或者，由于我理解问题有误所以写得不满足要求，如果是这样的话请指出，我可以修改