```python
"""
Definition of views.
"""

import json
from os import path
from datetime import datetime
from django.contrib.auth.decorators import login_required
from django.contrib.auth.forms import UserCreationForm
from django.urls import reverse, reverse_lazy
from django.http import HttpRequest, HttpResponseRedirect
from django.shortcuts import get_object_or_404, render, redirect
from django.utils import timezone
from django.views.generic import ListView, DetailView, CreateView
from app.models import ProxyUser, Group, User, Membership
from app.googledrive import GoogleDrive
from .forms import CreateGroupForm, AddUserForm
from django.http import HttpResponse
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes

class SignUp(CreateView):
    form_class = UserCreationForm
    success_url = reverse_lazy('login')
    template_name = 'app/signup.html'

def contact(request):
    """Renders the contact page."""
    assert isinstance(request, HttpRequest)
    return render(
        request,
        'app/contact.html',
        {
            'title':'Contact',
            'message':'Your contact page.',
            'year':datetime.now().year,
        }
    )

def home(request):
    assert isinstance(request, HttpRequest)
    return render(
        request,
        'app/about.html',
        {
            'title':'Securing the Cloud',
            'message':'Your contact page.',
            'year':datetime.now().year,
        }
    )

def about(request):
    """Renders the about page."""
    assert isinstance(request, HttpRequest)
    return render(
        request,
        'app/about.html',
        {
            'title':'About',
            'message':'Your application description page.',
            'year':datetime.now().year,
        }
```

```
    )

def vote(request, poll_id):
    """Handles voting. Validates input and updates the repository."""
    poll = get_object_or_404(Poll, pk=poll_id)
    try:
        selected_choice = poll.choice_set.get(pk=request.POST['choice'])
    except (KeyError, Choice.DoesNotExist):
        return render(request, 'app/details.html', {
            'title': 'Poll',
            'year': datetime.now().year,
            'poll': poll,
            'error_message': "Please make a selection.",
    })
    else:
        selected_choice.votes += 1
        selected_choice.save()
        return HttpResponseRedirect(reverse('app:results', args=(poll.id,)))

@login_required
def seed(request):
    """Seeds the database with sample polls."""
    samples_path = path.join(path.dirname(__file__), 'samples.json')
    with open(samples_path, 'r') as samples_file:
        samples_polls = json.load(samples_file)

    for sample_poll in samples_polls:
        poll = Poll()
        poll.text = sample_poll['text']
        poll.pub_date = timezone.now()
        poll.save()

        for sample_choice in sample_poll['choices']:
            choice = Choice()
            choice.poll = poll
            choice.text = sample_choice
            choice.votes = 0
            choice.save()

    return HttpResponseRedirect(reverse('app:home'))

@login_required
def drive(request):
    """Drive login."""
    assert isinstance(request, HttpRequest)
    drive = GoogleDrive()
    file_list = drive.getFiles()
    return render(
        request,
        'app/drive.html',
        {
            'title':'Drive',
            'file_list': file_list,
            'folderId': '1a_ZOqi75h6nTvsUEPDi8NGUrb9Tk-dkh',
        }
    )

@login_required
def driveFolder(request, folder, groupId):
    """Drive login."""
    assert isinstance(request, HttpRequest)
    drive = GoogleDrive()
```

```
125     file_list = drive.getFilesInFolder(folder)
126     group = Group.objects.get(id=groupId)
127     membersList =  []
128     for m in Membership.objects.filter(group=group):
129         membersList.append(m.user_id)
130     return render(
131         request,
132         'app/drive.html',
133         {
134             'title':group.name,
135             'file_list': file_list,
136             'folderId': folder,
137             'groupId': groupId,
138             'membersList': membersList,
139             'errorMessage': 'app/errors/403.html'
140         }
141     )
142 
143 @login_required
144 def driveUpload(request, folder, groupId):
145     assert isinstance(request, HttpRequest)
146     gDrive = GoogleDrive()
147     gDrive.uploadFile(folder, groupId)
148     return driveFolder(request, folder, groupId)
149 
150 @login_required
151 def driveDownload(request, id, title, folder, groupId):
152     assert isinstance(request, HttpRequest)
153     gDrive = GoogleDrive()
154     gDrive.downloadFile(id, title, groupId)
155     return driveFolder(request, folder, groupId)
156 
157 @login_required
158 def createGroup(request, username):
159     assert isinstance(request, HttpRequest)
160     if request.method == 'POST':
161         form = CreateGroupForm(request.POST)
162         if form.is_valid():
163             name = form.cleaned_data['name']
164             if Group.objects.filter(name=name).exists():
165                 return render(
166                     request,
167                     'app/createGroup.html',
168                     {
169                         'title': username,
170                         'form': CreateGroupForm(),
171                         'error':'Group already exists'
172                     }
173                 )
174             else:
175                 user = User.objects.get_by_natural_key(username)
176                 gDrive = GoogleDrive()
177                 folderId = gDrive.createGroup(name)
178 
179                 group = Group()
180                 group.name = name
181                 group.owner = user
182                 group.gdriveid = folderId
183                 group.key = get_random_bytes(16)
184                 group.save()
185 
186                 membership = Membership()
```

```
                membership.group = group
                membership.user = user
                membership.save()
                return redirect('/userpage/'+username)
    else:
        return render(
            request,
            'app/createGroup.html',
            {
                'title': username,
                'form': CreateGroupForm()
            }
        )

@login_required
def userpage(request, username):
    assert isinstance(request, HttpRequest)
    proxyuser = ProxyUser.objects.get_by_natural_key(username)
    return render(
            request,
            'app/userpage.html',
            {
                'title': 'Manage groups',
                'proxyuser': proxyuser
            }
        )

@login_required
def manageUsers(request, groupId):
    assert isinstance(request, HttpRequest)
    group = Group.objects.get(id=groupId)
    if request.method == 'POST':
        form = AddUserForm(request.POST)
        if form.is_valid():
            name = form.cleaned_data['name']
            if not User.objects.filter(username=name).exists():
                return render(
                    request,
                    'app/manageUsers.html',
                    {
                        'title': 'ManageUsers',
                        'form': AddUserForm(),
                        'error':'User doesn\'t exist',
                        'group': group
                    }
                )
            user = User.objects.get_by_natural_key(name)
            if Membership.objects.filter(user=user, group=group).exists():
                return render(
                    request,
                    'app/manageUsers.html',
                    {
                        'title': 'ManageUsers',
                        'form': AddUserForm(),
                        'error':'User is already a member',
                        'group': group
                    }
                )
            else:
                membership = Membership()
                membership.group = group
                membership.user = user
```

```
249                 membership.save()
250                 return render(
251                     request,
252                     'app/manageUsers.html',
253                     {
254                         'title': 'ManageUsers',
255                         'form': AddUserForm(),
256                         'group': group
257                     }
258                 )
259     else:
260         return render(
261                 request,
262                 'app/manageUsers.html',
263                 {
264                     'title': 'ManageUsers',
265                     'form': AddUserForm(),
266                     'group': group
267                 }
268             )
269 
270 @login_required
271 def removeUser(request, groupId, userId):
272     group = Group.objects.get(id=groupId)
273     user = User.objects.get(id=userId)
274 
275     if Membership.objects.filter(user=user, group=group).exists():
276         Membership.objects.filter(user=user, group=group).delete()
277 
278     return redirect('/manageUsers/'+groupId)
```