```python
#导入与网页有关
from django.shortcuts import render
from django.http import HttpResponse,HttpResponseRedirect,HttpResponseForbidden
from django.urls import reverse,reverse_lazy

#导入与认证有关
from django.contrib.auth.hashers import make_password
from django.contrib.auth import authenticate,login,logout
from django.contrib.auth.models import User, Group
from django.contrib.auth.decorators import login_required
from django.db.utils import IntegrityError
from django.views.generic.edit import CreateView, DeleteView, FormView, UpdateView

#导入与数据库有关
from it_exam.models import Exam
from it_exam.models import Assurance, UserHistory, UserGrade, UserSession,
CountUserStopTime
from django.contrib.sessions.models import Session
from django.db.models import Q

#导入与表单有关
from django.views.decorators.csrf import csrf_protect
from django.views import defaults
from .forms import Logon, Login, ForgetPassword, EnterNewPassword, UserData,
GroupManage, Classes

#导入与时区有关
from django.utils.timezone import now

#导入与邮箱登陆有关
from django.core.mail import BadHeaderError,send_mail
from exam_web import settings
from smtplib import SMTPRecipientsRefused
from django.core.exceptions import ObjectDoesNotExist
from django.db.models import Case, When

#导入与文件有关
from zipfile import ZipFile
import os

#导入与Ajax有关
import json
from django.http import JsonResponse

#导入与到导出有关
import xlwt

#导入其他
import random
# Create your views here.
from django.utils.safestring import mark_safe



#-------------------------------------------------------------------------------
-------------------------------------------------------------
#调试函数
#-------------------------------------------------------------------------------
-------------------------------------------------------------
#or (('.docx' not in single[1].split('|')[1]) and ('.py' not in
single[1].split('|')[1]))
#常用函数
class function():

    #显示主页相关内容
    def mainpage_show(request,username):

        if request.method != 'POST':
            ready_history = []
            past = UserHistory.objects.all().filter(user__username = username)

            #整理用户做题历史，是格式为form要求的格式
```

```
            for each in past:
                ready_history.append((str(each.history + ',' + str(each.date)),{
                '做题时间:' + str(each.date).replace('+08:00','')[:19]:
                    ['正确题数:' + str(each.correct),
                    '错误题数:' + str(each.wrong),
                    '正确率:' + each.accuracy_percent]}))

            form = UserData(ready_history)

            data = UserGrade.objects.get(user__username = username)
            all_data = UserGrade.objects.all().exclude(user__username = username)
            number_of_all_data = UserGrade.objects.count() - 1
            surpass_number = 0

            for i in all_data:
                if float(i.accuracy_percent.replace('%','')) < float(data.
                accuracy_percent.replace('%','')):
                    surpass_number += 1
            surpass_rate = str(round((surpass_number * 100) / number_of_all_data,2))
            + '%'

            return render(request, 'it_exam/mainpage_examinee.html', {
                                                    'form':form,
                                                    'username':username,
                                                    'accuracy_percent':str(
                                                    data.accuracy_percent),
                                                    'correct':str(data.
                                                    correct),
                                                    'wrong':str(data.wrong),
                                                    'cover':data.cover_rate,
                                                    'how_many': surpass_rate,
                                                    })

        else:

            #为用户显示此次做题的情况
            form = request.POST

            #数据类型: '253:wrong|no_reply|A, 294:wrong|no_reply|B,......
            21:wrong|no_reply|C, 526:wrong|no_reply|D'

            try:
                single_history = form['history'].split(',')
            except KeyError:
                return HttpResponseRedirect(reverse('mainpage'))
            datatime = single_history[-1]
            single_history.pop()

            return function.show_history(single_history,username,datatime)


    #判断是否登录
    def is_login(request):

        if request.session.get("username", "False") == "False":
            return HttpResponseRedirect('/itexam/')

        username = request.session.get("username")

        if not UserSession.objects.get(user__username = username).is_authenticated:
            return HttpResponse('''
                    <!DOCTYPE html>
                    <html>
                        <meta name="viewport" content="width=device-width,
                        initial-scale=1.0">
                        <meta charset="UTF-8">
                        <head>
                            <link rel="stylesheet"
                            href="/static/it_exam/transition.css">
```

```
                            <title>学考练习系统</title>
                        </head>
                        <body>
                            <h1>
                            %s已登出!
                            </h1>
                            <a href = "/itexam/">重新登陆</a>
                        </body>
                    </html>''' % username)

        return username


    #显示做题历史
    def show_history(single_history,user,datatime):

        all_question = ''
        filename = UserHistory.objects.get(user__username = user, date = str(datatime
        ).replace('%20',' ')).filename
        if filename != 'NoFile':
            file_download = filename.split('#')[0]
            which_file = filename.split('#')[1].split(',')
            which_file.remove('')
            file_dict = {}

            for each_file in which_file:
                file_dict[int(each_file.split('|')[0])] = each_file.split('|')[1]

        else:
            file_dict = {}
            file_download = ''
            which_file = ['','','','','','']

        for j in range(46):

            #数据类型: '253:wrong|no_reply|A, 294:wrong|no_reply|B,......
            21:wrong|no_reply|C, 526:wrong|no_reply|D'

            try:
                single = single_history[j].split(':')
            except IndexError:
                continue
            single[1] = single[1].replace('○●○⊕◎Θ○¤㈱㊣★☆',':').replace('§№☆★○●◎◇',
            ',')
            #获得这个题的相关数据
            key = single[0]

            try:
                user_question = Exam.objects.get(pk = int(key))
            except ValueError:
                continue

            name = user_question.question[user_question.question.find('>',20) + 1:
            user_question.question.find('<',20)]
            mode = 'PAT' if len(single_history) == 6 else ''

            #为用户展示做题记录

            #为用户标记正确选项与被选选项
            index = ['A','B','C','D']
            color = ['transparent','transparent','transparent','transparent']
            if user_question.pattern in ['1','0']:
                if single[1].split('|')[1] == single[1].split('|')[2]:
                    if single[1].split('|')[1] == 'A':
                        color[0] = 'palegreen'

                    elif single[1].split('|')[1] == 'B':
                        color[1] = 'palegreen'

                    elif single[1].split('|')[1] == 'C':
                        color[2] = 'palegreen'

```

```
                    elif single[1].split('|')[1] == 'D':
                        color[3] = 'palegreen'

                else:
                    if single[1].split('|')[1] == 'A':
                        color[0] = 'lightpink'
                        color[index.index(single[1].split('|')[2])] = 'palegreen'

                    elif single[1].split('|')[1] == 'B':
                        color[1] = 'lightpink'
                        color[index.index(single[1].split('|')[2])] = 'palegreen'

                    elif single[1].split('|')[1] == 'C':
                        color[2] = 'lightpink'
                        color[index.index(single[1].split('|')[2])] = 'palegreen'

                    elif single[1].split('|')[1] == 'D':
                        color[3] = 'lightpink'
                        color[index.index(single[1].split('|')[2])] = 'palegreen'


            #获取所有选项的答题情况
            all_question += (

                '<div style="padding-bottom: 2em;">' +

                '<div class="Question">' +
                str(j + 1) + '.' + user_question.question +
                '</div>' +

                '%s' %
                (

                #显示正确还是错误还是未作答
                '<p style="background-color: %s;border-radius: 10px;padding: 10px;
                display: inline-block; color: %s">%s</p>' %
                ('hsl(120, 70%, 60%)' if single[1].split('|')[1] == single[1].split(
                '|')[2] else 'hsl(350, 70%, 60%)',
                 'black' if single[1].split('|')[1] == single[1].split('|')[2] else
                 'white',
                 '正确' if single[1].split('|')[1] == single[1].split('|')[2] else
                 '错误' if single[1].split('|')[1] != 'no_reply' else '未作答') +

                '<label class="Option" style="background-color: %s;">' % (color[0]) +
                'A.' + user_question.answer_a +
                '</label><br>' +

                '<label class="Option" style="background-color: %s;">' % (color[1]) +
                'B.' + user_question.answer_b +
                '</label><br>'
                if j < (40 if not len(single_history) == 6 else 0) else '') +

                '%s' % (

                '<label class="Option" style="background-color: %s;">' % (color[2]) +
                'C.' + user_question.answer_c +
                '</label><br>' +

                '<label class="Option" style="background-color: %s;">' % (color[3]) +
                'D.' + user_question.answer_d +
                '</label>'
                if user_question.pattern == '1'
                else ''
                ) +

                '%s' % (

                '<label class="description" id = "%s" onclick="describe(id)">' % (
                user_question.id) +
                '解析: ' +
                '</label>'
```

```
                if not (user_question.pattern == '3' or user_question.pattern == '4'
                or user_question.pattern == '5' or  user_question.pattern == '6' or
                user_question.pattern == '8')
                else ''
                ) +

                '<h1>' +
                '%s' % ('未作答<br>' if (single[1].split('|')[1] == 'no_reply') or (
                single[1].split('|')[1] == '') else '你的答案是:' + single[1].split(
                '|')[1] + '<br>' if j < (41 if not len(single_history) == 6 else 0)
                else '你的答案是:<a href="%s" download="%s">作答文件</a>' % (
                file_download + '/' + file_dict[j - 41],file_dict[j - 41]) + '<br>'
                if single[1].split('|')[1] != '文件类型不正确,综合题必须是word文档'
                and single[1].split('|')[1] !=
                '文件类型不正确,编程/海龟画图题必须是python文件' else single[1].split
                ('|')[1] + '<br>') +
                '正确答案是:' + (single[1].split('|')[2] if (mode == '' and (j < 42
                or j > 44)) or (mode == 'PAT' and (j < 2 or j > 4)) else '<a
                href="%s" download="%s">参考答案</a>' % (
                'http://192.144.180.87:8000/static/it_exam/编程题答案/' + name +
                '.py',name)) + '<br>' +
                '</h1>'
                '</div>'
            )

        #编写html相关内容，装饰
        return HttpResponse('''
                            <head>
                                <script
                                src="https://code.jquery.com/jquery-3.6.0.min.js"></sc
                                ript>
                                <meta name="viewport" content="width=device-width,
                                initial-scale=1.0">
                                <title>学考练习系统</title>
                                <link rel="stylesheet"
                                href="/static/it_exam/response.css">
                            </head>''' +

                            '<body>' +

                            '''
                            <div class="Header" style="position: sticky; top: 0">'''
                            +
                                '<h1 class = "welcome">欢迎使用此信息练习系统</h1>' +
                                '<h1 class="TimeCounter" style="font-size: 20px;
                                line-height: 30px;" id="counter">做题时间:' +
                                str(UserHistory.objects.get(user__username = user,
                                date = str(datatime).replace('%20',' ')).date)[:19] +
                                 '<br>' +
                                '正确率:' + UserHistory.objects.get(user__username =
                                user, date = str(datatime).replace('%20',' ')).
                                accuracy_percent + ' ' +
                                '正确数:' + str(UserHistory.objects.get(
                                user__username = user, date = str(datatime).replace(
                                '%20',' ')).correct) + ' ' +
                                '错误数:' + str(UserHistory.objects.get(
                                user__username = user, date = str(datatime).replace(
                                '%20',' ')).wrong) + ' ' +
                                '</h1>' +
                                '<a href="/itexam/%s/">' % ('mainpage_show' if User.
                                objects.get(username = user).is_superuser else
                                'mainpage') +
                                    '<button style = "width: 220px; margin-top:
                                    -74px; height: 60; font-size: 30" class =
                                    "butn">返回主页</button>' +
                                '</a>' +
                            '</div>' +

                            '<div style = "margin-top: 150px; margin-left:
                            12%;margin-right: 12%;height: fit-content;">' +
                            all_question +
                            '</div>' +
```

```
                                '''
                                <script>

                                    function getCookie(name) {
                                        let cookieValue = null;
                                        if (document.cookie && document.cookie !== '') {
                                            const cookies = document.cookie.split(';');
                                            for (let i = 0; i < cookies.length; i++) {
                                                const cookie = cookies[i].trim();
                                                // Does this cookie string begin with
                                                the name we want?
                                                if (cookie.substring(0, name.length + 1)
                                                === (name + '=')) {
                                                    cookieValue =
                                                    decodeURIComponent(cookie.substring(na
                                                    me.length + 1));
                                                    break;
                                                }
                                            }
                                        }
                                        return cookieValue;
                                    }

                                    function describe(id){
                                        let csrftoken = getCookie('csrftoken');
                                        $.ajax({
                                        url: '/itexam/getdescription/'+ id + '/',
                                        type: 'POST',
                                        headers: {'X-CSRFToken': csrftoken},
                                        mode: 'same-origin',
                                        success: function(response) {
                                                var element = document.getElementById(id);
                                                if
                                                (element.classList.contains('clicked')){
                                                    element.classList.remove('clicked');
                                                    var nextElement = element.nextSibling;

                                                    nextElement.parentNode.removeChild(nex
                                                    tElement);
                                                }
                                                else{

                                                    element.insertAdjacentHTML('afterend',
                                                    '<p>' + response + '</p>');
                                                    element.classList.add('clicked');
                                                }
                                            }
                                        })
                                    }
                                </script>
                                ''' +

                                '</body>'
                                )


    #抽取题目
    def pick(user_email,mode,recent_pick):


        #规定选择题，判断题，简答题，综合题，编程题，海龟画图题
        number_choice = 30
        number_judge = 10
        number_brief = 1
        number_integrate = 1
        number_program_a = 1
        number_program_b = 1
        number_program_c = 1
        number_total_program = number_program_a + number_program_b + number_program_c
        number_turtle = 1

```

```
366
367             i = 0
368             n = 0
369             num = 1
370             exam_paper_ready = [[], ]
371             number_picked = []
372             QuestionChoiced = ''
373
374
375             question = UserGrade.objects.get(user__email = user_email).question.split(',')
376             question_wrong = UserGrade.objects.get(user__email = user_email).
                wrong_problems.split(',')
377             case_expression = Case(*[When(id=id_val, then=pos) for pos, id_val in
                enumerate(recent_pick)], default=-1)
378
379
380             #选择题
381             i = 0
382             if recent_pick == '':
383                 exam_paper = Exam.objects.all().filter(pattern = '1')
384
385             #如果已经在做题，抽取正在做的题
386             else:
387                 exam_paper = Exam.objects.all().filter(pattern = '1',id__in = recent_pick
                    ).order_by(case_expression)
388                 for j in range(len(exam_paper)):
389                     qes = exam_paper[j]
390                     Question = qes.question
391                     if qes.question.find('jpg') != -1:
392                         Question = Question[0:Question.find('<')]
393                     Prepare = [0, str(num) + '.' + Question, qes.answer_a, qes.answer_b,
                        qes.answer_c, qes.answer_d]
394
395                     if qes.question[-1] == '>':
396                         ImgPath = qes.question[qes.question.find('/'):qes.question.find(
                            'jpg')+3]
397                         Prepare.append(ImgPath)
398                         Prepare.append(1)
399                     exam_paper_ready.append(Prepare)
400                     num += 1
401
402
403             if recent_pick == '':
404                 exam_count = exam_paper.count()
405
406                 for j in range(len(exam_paper)):
407                     if i < number_choice:
408
409                         #如果没做过则多抽一次提高被抽几率
410                         qes = exam_paper[j]
411
412                         if qes.id not in question:
413
414                             if random.randint(1,exam_count) == 1:
415                                 Question = qes.question
416                                 number_picked.append(j)
417                                 QuestionChoiced += str(qes.id) + ','
418
419                                 if qes.question.find('jpg') != -1:
420                                     Question = Question[0:Question.find('<')]
421                                 Prepare = [0, str(num) + '.' + Question, qes.answer_a,
                                    qes.answer_b, qes.answer_c, qes.answer_d]
422
423                                 if qes.question[-1] == '>':
424                                     ImgPath = qes.question[qes.question.find('/'):qes.
                                        question.find('jpg')+3]
425                                     Prepare.append(ImgPath)
426                                     Prepare.append(1)
427
428                                 i += 1
429                                 num += 1
430                                 exam_paper_ready.append(Prepare)
```

```
                    #如果错过则多抽两次提高被抽几率
                    if qes.id in question_wrong:
                        random_number = random.randint(1,exam_count)

                        if random_number == 1 or random_number == 2:
                            number_picked.append(j)
                            QuestionChoiced += str(qes.id) + ','

                            if qes.question.find('jpg') != -1:
                                Question = Question[0:Question.find('<')]
                            Prepare = [0, str(num) + '.' + Question, qes.answer_a,
                            qes.answer_b, qes.answer_c, qes.answer_d]

                            if qes.question[-1] == '>':
                                ImgPath = qes.question[qes.question.find('/'):qes.
                                question.find('jpg')+3]
                                Prepare.append(ImgPath)
                                Prepare.append(1)

                            i += 1
                            num += 1
                            exam_paper_ready.append(Prepare)

                else:
                    break

            while i < number_choice:

                #抽取下标
                number = random.randint(0,len(exam_paper) - 1)
                qes = exam_paper[number]
                #防止重选
                if number in number_picked:
                    continue

                #过滤通用技术
                elif len(qes.pattern) == 4 and qes.pattern[1] == '1':
                    continue


                number_picked.append(number)
                QuestionChoiced += str(qes.id) + ','

                Question = qes.question

                if qes.question.find('jpg') != -1:
                    Question = Question[0:Question.find('<')]
                Prepare = [0, str(num) + '.' + Question, qes.answer_a, qes.answer_b,
                qes.answer_c, qes.answer_d]

                if qes.question[-1] == '>':
                    ImgPath = qes.question[qes.question.find('/'):qes.question.find(
                    'jpg')+3]
                    Prepare.append(ImgPath)
                    Prepare.append(1)
                exam_paper_ready.append(Prepare)


                i += 1
                num += 1


        #判断题
        i = 0
        if recent_pick == '':
            exam_paper = Exam.objects.all().filter(pattern = '0')

        #如果已经在做题，抽取正在做的题
        else:
            exam_paper = Exam.objects.all().filter(pattern = '0',id__in = recent_pick
            ).order_by(case_expression)
```

```
498
499             for j in range(len(exam_paper)):
500                 qes = exam_paper[j]
501                 exam_paper_ready.append([2, str(num) + '.' +qes.question])
502                 num += 1
503
504         if recent_pick == '':
505             exam_count = exam_paper.count()
506
507             for j in range(len(exam_paper)):
508                 if i < number_judge:
509
510                     qes = exam_paper[j]
511                     #如果没做过则多抽一次提高被抽几率
512                     if qes.id not in question:
513                         if random.randint(1,exam_count) == 1:
514                             number_picked.append(j)
515                             QuestionChoiced += str(qes.id) + ','
516                             exam_paper_ready.append([2, str(num) + '.' +qes.question])
517
518                             i += 1
519                             num += 1
520
521
522                     #如果错过则多抽两次提高被抽几率
523                     if qes.id in question_wrong:
524                         random_number = random.randint(1,exam_count)
525
526                         if random_number == 1 or random_number == 2:
527                             number_picked.append(j)
528                             QuestionChoiced += str(qes.id) + ','
529                             exam_paper_ready.append([2, str(num) + '.' + qes.question
                            ])
530
531                             i += 1
532                             num += 1
533
534
535                 else:
536                     break
537
538             while i < number_judge:
539
540                 #抽取下标
541                 number = random.randint(0,len(exam_paper) - 1)
542                 qes = exam_paper[number]
543                 #防止重选
544                 if number in number_picked:
545                     continue
546
547                 #过滤通用技术
548                 elif len(qes.pattern) == 4 and qes.pattern[1] == '1':
549                     continue
550                 number_picked.append(number)
551                 QuestionChoiced += str(qes.id) + ','
552
553                 exam_paper_ready.append([2, str(num) + '.' + qes.question])
554                 i += 1
555                 num += 1
556
557
558         #若只做选择填空题则跳过下方代码
559         if mode == 'ChoiceAndJudge' and recent_pick == '':
560             user_name = User.objects.get(email = user_email)
561             user_name.userhistory_set.create(history = 'CAJ',
562                                              date = now(),
563                                              accuracy_percent = '',
564                                              correct = 0,
565                                              wrong = 0,
566                                              question = QuestionChoiced.removesuffix(
                                                ','),
567                                              filename = 'NoFile')
```

```
568             return exam_paper_ready
569
570
571
572         elif mode == 'ProgramAndTackle' and recent_pick == '':
573             num = 1
574             number_picked = []
575             QuestionChoiced = ''
576             exam_paper_ready = [[],]
577
578
579         #简答题
580         i = 0
581         if recent_pick == '':
582             exam_paper = Exam.objects.all().filter(pattern = '9')
583
584         #如果已经在做题，抽取正在做的题
585         else:
586             exam_paper = Exam.objects.all().filter(pattern = '9',id__in = recent_pick
                ).order_by(case_expression)
587
588             for j in range(len(exam_paper)):
589                 qes = exam_paper[j]
590                 exam_paper_ready.append([4, str(num) + '.' + qes.question])
591                 num += 1
592
593
594         if recent_pick == '':
595             exam_count = exam_paper.count()
596
597             for j in range(len(exam_paper)):
598                 if i < number_brief:
599
600                     qes = exam_paper[j]
601                     #如果没做过则多抽一次提高被抽几率
602                     if qes.id not in question:
603                         if random.randint(1,exam_count) == 1:
604                             number_picked.append(j)
605                             QuestionChoiced += str(qes.id) + ','
606                             exam_paper_ready.append([4, str(num) + '.' + qes.question
                                ])
607
608                             i += 1
609                             num += 1
610
611
612                     #如果错过则多抽两次提高被抽几率
613                     if qes.id in question_wrong:
614                         random_number = random.randint(1,exam_count)
615
616                         if random_number == 1 or random_number == 2:
617                             number_picked.append(j)
618                             QuestionChoiced += str(qes.id) + ','
619                             exam_paper_ready.append([4, str(num) + '.' + qes.question
                                ])
620
621                             i += 1
622                             num += 1
623
624
625                 else:
626                     break
627
628             while i < number_brief:
629
630                 #抽取下标
631                 number = random.randint(0,len(exam_paper) - 1)
632                 qes = exam_paper[number]
633                 #防止重选
634                 if number in number_picked:
635                     continue
636                 #过滤通用技术
```

```
                elif len(qes.pattern) == 4 and qes.pattern[1] == '1':
                    continue
                number_picked.append(number)
                QuestionChoiced += str(qes.id) + ','
                #加入出题表
                exam_paper_ready.append([4, str(num) + '.' + qes.question])

                num += 1
                break

        #操作题
        i = 0
        if recent_pick == '':
            exam_paper = Exam.objects.all().filter(pattern = '3')

        #如果已经在做题，抽取正在做的题
        else:
            exam_paper = Exam.objects.all().filter(pattern = '3',id__in = recent_pick
            ).order_by(case_expression)

            for j in range(len(exam_paper)):
                qes = exam_paper[j]
                exam_paper_ready.append([4, str(num) + '.' + qes.question +
                '只能提交word文档'])
                num += 1

        if recent_pick == '':
            exam_count = exam_paper.count()

            for j in range(len(exam_paper)):
                if i < number_integrate:

                    qes = exam_paper[j]
                    #如果没做过则多抽一次提高被抽几率
                    if qes.id not in question:
                        if random.randint(1,exam_count) == 1:
                            number_picked.append(j)
                            QuestionChoiced += str(qes.id) + ','
                            exam_paper_ready.append([4, str(num) + '.' + qes.question
                             + '只能提交word文档'])

                            i += 1
                            num += 1


                    #如果错过则多抽两次提高被抽几率
                    if qes.id in question_wrong:
                        random_number = random.randint(1,exam_count)

                        if random_number == 1 or random_number == 2:
                            number_picked.append(j)
                            QuestionChoiced += str(qes.id) + ','
                            exam_paper_ready.append([4, str(num) + '.' + qes.question
                             + '只能提交word文档'])

                            i += 1
                            num += 1


                else:
                    break

            while i < number_integrate:
                #抽取下标
                number = random.randint(0,len(exam_paper) - 1)
                qes = exam_paper[number]
                #防止重选
                if number in number_picked:
                    continue
                #过滤通用技术
                elif len(qes.pattern) == 4 and qes.pattern[1] == '1':
                    continue
```

```
                number_picked.append(number)
                QuestionChoiced += str(qes.id) + ','
                #加入出题表
                exam_paper_ready.append([4, str(num) + '.' + qes.question +
                '只能提交word文档'])

                num += 1
                break


        #编程题的测试数据
        testA = ''
        testB = ''
        testC = ''

        #抽编程题a
        i = 0
        if recent_pick == '':
            exam_paper = Exam.objects.all().filter(pattern = '4')

        #如果已经在做题，抽取正在做的题
        else:
            exam_paper = Exam.objects.all().filter(pattern = '4',id__in = recent_pick
            ).order_by(case_expression)

            for j in range(len(exam_paper)):
                qes = exam_paper[j]
                exam_paper_ready.append([3, str(num) + '.' + qes.question +
                '只能提交python文件'])
                num += 1

        if recent_pick == '':
            exam_count = exam_paper.count()

            for j in range(len(exam_paper)):
                if i < number_program_a:

                    qes = exam_paper[j]
                    #如果没做过则多抽一次提高被抽几率
                    if qes.id not in question:
                        if random.randint(1,exam_count) == 1:
                            number_picked.append(j)
                            QuestionChoiced += str(qes.id) + ','
                            exam_paper_ready.append([3, str(num) + '.' + qes.question
                             + '只能提交python文件'])

                            n += 1
                            i += 1
                            num += 1


                    #如果错过则多抽两次提高被抽几率
                    if qes.id in question_wrong:
                        random_number = random.randint(1,exam_count)

                        if random_number == 1 or random_number == 2:
                            number_picked.append(j)
                            QuestionChoiced += str(qes.id) + ','
                            exam_paper_ready.append([3, str(num) + '.' + qes.question
                             + '只能提交python文件'])

                            n += 1
                            i += 1
                            num += 1


                else:
                    break


        #抽编程题b
        i = 0
```

```
772             if recent_pick == '':
773                 exam_paper = Exam.objects.all().filter(pattern = '5')
774 
775             #如果已经在做题，抽取正在做的题
776             else:
777                 exam_paper = Exam.objects.all().filter(pattern = '5',id__in = recent_pick
                    ).order_by(case_expression)
778 
779                 for j in range(len(exam_paper)):
780                     qes = exam_paper[j]
781                     exam_paper_ready.append([3, str(num) + '.' + qes.question +
                        '只能提交python文件'])
782                     num += 1
783 
784             if recent_pick == '':
785                 exam_count = exam_paper.count()
786 
787                 for j in range(len(exam_paper)):
788                     if i < number_program_b:
789 
790                         qes = exam_paper[j]
791                         #如果没做过则多抽一次提高被抽几率
792                         if qes.id not in question:
793                             if random.randint(1,exam_count) == 1:
794                                 number_picked.append(j)
795                                 QuestionChoiced += str(qes.id) + ','
796                                 exam_paper_ready.append([3, str(num) + '.' + qes.question
                                    + '只能提交python文件'])
797 
798                                 n += 1
799                                 i += 1
800                                 num += 1
801 
802 
803                         #如果错过则多抽两次提高被抽几率
804                         if qes.id in question_wrong:
805                             random_number = random.randint(1,exam_count)
806 
807                             if random_number == 1 or random_number == 2:
808                                 number_picked.append(j)
809                                 QuestionChoiced += str(qes.id) + ','
810                                 exam_paper_ready.append([3, str(num) + '.' + qes.question
                                    + '只能提交python文件'])
811 
812                                 n += 1
813                                 i += 1
814                                 num += 1
815 
816 
817                     else:
818                         break
819 
820             #抽编程题c
821             i = 0
822             if recent_pick == '':
823                 exam_paper = Exam.objects.all().filter(pattern = '6')
824 
825             #如果已经在做题，抽取正在做的题
826             else:
827                 exam_paper = Exam.objects.all().filter(pattern = '6',id__in = recent_pick
                    ).order_by(case_expression)
828 
829                 for j in range(len(exam_paper)):
830                     qes = exam_paper[j]
831                     exam_paper_ready.append([3, str(num) + '.' + qes.question +
                        '只能提交python文件'])
832                     num += 1
833 
834             if recent_pick == '':
835                 exam_count = exam_paper.count()
836 
837                 for j in range(len(exam_paper)):
```

```
                if i < number_program_c:

                    qes = exam_paper[j]
                    #如果没做过则多抽一次提高被抽几率
                    if qes.id not in question:
                        if random.randint(1,exam_count) == 1:
                            number_picked.append(j)
                            QuestionChoiced += str(qes.id) + ','
                            exam_paper_ready.append([3, str(num) + '.' + qes.question
                             + '只能提交python文件'])

                            n += 1
                            i += 1
                            num += 1


                    #如果错过则多抽两次提高被抽几率
                    if qes.id in question_wrong:
                        random_number = random.randint(1,exam_count)

                        if random_number == 1 or random_number == 2:
                            number_picked.append(j)
                            QuestionChoiced += str(qes.id) + ','
                            exam_paper_ready.append([3, str(num) + '.' + qes.question
                             + '只能提交python文件'])

                            n += 1
                            i += 1
                            num += 1


                else:
                    break

            #编程题
            while n < number_total_program:

                if 0 <= n and n < number_program_a:
                    exam_paper = Exam.objects.all().filter(pattern = '4')

                if number_program_a <= n and n < number_program_b + number_program_a:
                    exam_paper = Exam.objects.all().filter(pattern = '5')

                if number_program_b + number_program_a <= n and n < number_program_b
                + number_program_a + number_program_c:
                    exam_paper = Exam.objects.all().filter(pattern = '6')

                #抽取下标
                number = random.randint(0,len(exam_paper) - 1)
                qes = exam_paper[number]
                #防止重选
                if number in number_picked:
                    continue
                #过滤通用技术
                elif len(qes.pattern) == 4 and qes.pattern[1] == '1':
                    continue
                number_picked.append(number)
                QuestionChoiced += str(qes.id) + ','
                #加入出题表
                exam_paper_ready.append([3, str(num) + '.' + qes.question +
                '只能提交python文件'])
                testA = qes.question.find('')

                n += 1
                num += 1



        #海龟画图题
        i = 0
        if recent_pick == '':
            exam_paper = Exam.objects.all().filter(pattern = '8')
```

```
906
907             #如果已经在做题，抽取正在做的题
908             else:
909                 exam_paper = Exam.objects.all().filter(pattern = '8',id__in = recent_pick
                    ).order_by(case_expression)
910
911                 for j in range(len(exam_paper)):
912                     qes = exam_paper[j]
913                     exam_paper_ready.append([5, str(num) + '.' + qes.question +
                        '只能提交python文件(下载"Turtle模块的相关函数"不会导致做题记录丢失)'])
914                     num += 1
915
916             if recent_pick == '':
917                 exam_count = exam_paper.count()
918
919                 for j in range(len(exam_paper)):
920                     if i < number_turtle:
921
922                         qes = exam_paper[j]
923                         #如果没做过则多抽一次提高被抽几率
924                         if qes.id not in question:
925                             if random.randint(1,exam_count) == 1:
926                                 number_picked.append(j)
927                                 QuestionChoiced += str(qes.id) + ','
928                                 exam_paper_ready.append([5, str(num) + '.' + qes.question
                                    +
                                    '只能提交python文件(下载"Turtle模块的相关函数"不会导致做题
                                    记录丢失)'])
929
930                                 i += 1
931                                 num += 1
932
933
934                         #如果错过则多抽两次提高被抽几率
935                         if qes.id in question_wrong:
936                             random_number = random.randint(1,exam_count)
937
938                             if random_number == 1 or random_number == 2:
939                                 number_picked.append(j)
940                                 QuestionChoiced += str(qes.id) + ','
941                                 exam_paper_ready.append([5, str(num) + '.' + qes.question
                                    +
                                    '只能提交python文件(下载"Turtle模块的相关函数"不会导致做题
                                    记录丢失)'])
942
943                                 i += 1
944                                 num += 1
945
946
947                     else:
948                         break
949
950                 while i < number_turtle:
951                     #抽取下标
952                     number = random.randint(0,len(exam_paper) - 1)
953                     qes = exam_paper[number]
954                     #防止重选
955                     if number in number_picked:
956                         continue
957                     #过滤通用技术
958                     elif len(qes.pattern) == 4 and qes.pattern[1] == '1':
959                         continue
960                     number_picked.append(number)
961                     QuestionChoiced += str(qes.id) + ','
962                     #加入出题表
963                     exam_paper_ready.append([5, str(num) + '.' + qes.question +
                        '只能提交python文件(下载"Turtle模块的相关函数"不会导致做题记录丢失)'])
964                     break
965
966                 user_name = User.objects.get(email = user_email)
967                 user_name.userhistory_set.create(history = user_email if mode == 'All'
                    else 'PAT',
```

```
                                                date = now(),
                                                accuracy_percent = '',
                                                correct = 0,
                                                wrong = 0,
                                                question = QuestionChoiced.removesuffix(
                                                ','),
                                                filename = 'NoFile')

        return exam_paper_ready


    #读取并设置UserGrade的人名
    def name():
        with open('Files_Used_In_Itexam/在校生O365邮箱-230508.txt', encoding='utf-8')
         as file:
            i = 0
            lines = file.readlines()
            students = {}
            for line in lines:
                if i == 0:
                    i += 1
                    continue
                line = line.replace('\n','').split(',')
                students[line[-1]] = line[0]


        for i in students:
            try:
                student = UserGrade.objects.get(user__email = i)
            except ObjectDoesNotExist:
                continue
            student.name = students[i]
            student.save()


    def get_email_by_number(number):
        with open('Files_Used_In_Itexam/在校生O365邮箱-230508.txt', encoding='utf-8')
         as file:
            i = 0
            lines = file.readlines()
            for line in lines:
                if i == 0:
                    i += 1
                    continue
                line = line.replace('\n','').split(',')
                if line[1] == number:
                    return line[-1]
            return False


def thetest(request):
    return render(request, 'it_exam/thetest.html', {'request': "abc"})


#导出用户做题历史
def loadfile(request):

    username = function.is_login(request)

    if type(username) != type('string'):
        return username

    if not User.objects.get(username = username).is_superuser:
        return HttpResponseRedirect('/itexam/mainpage/')

    # 指定数据类型
    response = HttpResponse(content_type='application/ms-excel')
    # 设置文件名称
    response['Content-Disposition'] = 'attachment; filename="result.csv"'
    # 创建工作簿
    wb = xlwt.Workbook(encoding='utf-8')
    # 创建表
```

```
1037        ws = wb.add_sheet('Menu')
1038        row_num = 0
1039        font_style = xlwt.XFStyle()
1040        # 二进制
1041        font_style.font.bold = True
1042        # 表头内容
1043        columns = ['姓名','做题情况']
1044        # 写进表头内容
1045        for col_num in range(len(columns)):
1046            ws.write(row_num, col_num, columns[col_num], font_style)
1047        # Sheet body, remaining rows
1048        font_style = xlwt.XFStyle()
1049        # 获取数据库数据
1050        rows = []
1051        students = UserGrade.objects.all().filter(user__is_superuser = 0)
1052        for i in students:
1053            rows.append([i.name,i.accuracy_percent])
1054        # 遍历提取出来的内容
1055        for row in rows:
1056            row_num += 1
1057            # 逐行写入Excel
1058
1059            for col_num in range(len(row)):
1060                ws.write(row_num, col_num, row[col_num], font_style)
1061
1062        wb.save(response)
1063        return response
1064
1065
1066    #在非全屏时提醒用户
1067    def not_full_screen(request):
1068        return HttpResponse('<h1>请勿退出全屏</h1>')
1069
1070
1071    #用户登出
1072    def log_out(request):
1073
1074        try:
1075            #自定义数据库的清理
1076            session = UserSession.objects.get(user__username = request.session["username"
                ])
1077            Session.objects.get(session_key = request.session.session_key).delete()
                #系统数据库清理
1078            session.is_authenticated = 0
1079            session.is_pick = 0
1080            session.save()
1081            logout(request)  #系统登出
1082        except (KeyError):
1083            pass
1084
1085        return HttpResponseRedirect('/itexam/')
1086
1087
1088    #javascript 获取数据库数据
1089    def get_data(request,user):
1090
1091
1092        if request.method == 'POST':
1093            #判断是否登录
1094            username = function.is_login(request)
1095
1096            if type(username) != type('string'):
1097                return username
1098
1099            data = UserHistory.objects.all().filter(user__username = user).values()
1100
1101            #避免JS解析datetime时间与python不一致，转换为字符串
1102            for j in range(len(data)):
1103                data[j]['date'] = str(data[j]['date'])
1104
1105            return JsonResponse(list(data), safe=False)
1106
```

```
1107        else:
1108            return HttpResponseRedirect(reverse('mainpage'))
1109
1110
1111    #管理员演示
1112    def mainpage_show(request):
1113
1114        username = function.is_login(request)
1115
1116        if type(username) != type('string'):
1117            return username
1118
1119        mail = User.objects.get(username = username).email
1120        UserHistory.objects.all().filter(Q(user__username = username) & (Q(history = mail
            ) | Q(history = 'CAJ'))).delete()
1121        session = UserSession.objects.get(user__username = username)
1122        session.is_pick = 0
1123        session.save()
1124
1125        #判断是否为管理员
1126        if User.objects.get(username = username).is_superuser:
1127
1128            return function.mainpage_show(request,username)
1129
1130        else:
1131            return HttpResponseRedirect(reverse('mainpage'))
1132
1133
1134    #获取本班级用户
1135    def get_user(request,name):
1136
1137        username = function.is_login(request)
1138
1139        if type(username) != type('string'):
1140            return username
1141
1142        if request.method != 'POST':
1143            return HttpResponseRedirect(reverse('mainpage'))
1144
1145        all_user = []
1146        try:
1147            group = Group.objects.get(name = name)
1148        except ObjectDoesNotExist:
1149            return HttpResponseRedirect(reverse('managegroup'))
1150        users = group.user_set.all()
1151        for i in users:
1152            all_user.append([i.username,UserGrade.objects.get(user__username = i.username
                ).name])
1153
1154        return JsonResponse(all_user, safe=False)
1155
1156
1157    #管理用户分组
1158    def manage_group(request):
1159
1160        username = function.is_login(request)
1161
1162        if type(username) != type('string'):
1163            return username
1164
1165
1166        if User.objects.get(username = username).is_superuser:
1167
1168            msg = ''
1169            all_name = []
1170
1171            def create_form(query):
1172                has = False
1173                multiple = False
1174                #逗号分隔搜法
1175                if query != 'all':
1176                    if ',' in query:
```

```
                    multiple = True
                    query = query.split(',')
                    try:
                        query.remove('')
                    except ValueError: #list.remove(x): x not in list
                        pass
                    for i in range(len(query)):
                        #去除空格
                        query[i] = query[i].strip()

                #去除空格
                if not multiple:
                    query = query.strip()
                #按照关键字查找
                all_user = User.objects.all().filter(is_superuser = 0)
                for i in all_user:
                    user_name = UserGrade.objects.get(user__username = i.username).
                    name
                    for student in query:
                        if (student if multiple else query) in user_name:
                            name = user_name
                            all_name.append((i.username,name))
                            has = True
                            if not multiple:
                                break

                if has:
                    form = GroupManage(all_name)
                    edit = ''
                    groups = Group.objects.all()

                    for i in groups:
                        edit += ('''
                                <div style = "border-top: 0.01em solid grey;
                                padding: 0.3em;">
                                <label for = "%s">选择学生从班级内删除</label>
                                <a id = %s style = "position: relative; top: 1em;"
                                onclick = "manage(id)">%s</a>
                                <br>
                                <div id = "%s" style = "margin-top: 2em;">
                                <label for = "%s"><input id = "%s" name =
                                "delete_name" type="checkbox"
                                value="%s">删除班级</label><br>
                                </div></div>
                                <br>'''
                                % (i.name,i.name,i.name,i.name + '-div',i.name,i.name
                                ,i.name))

                    return form,edit,''

                msg = mark_safe("<h3>没有找到对应的学生</h3>")
                form,edit = create_form('all')
                return form,edit,msg

            else:
                all_user = User.objects.all().filter(is_superuser = 0)


            for i in all_user:
                try:
                    name = UserGrade.objects.get(user__username = i.username,
                    user__is_superuser = 0).name
                except ObjectDoesNotExist:
                    continue

                all_name.append((i.username,name))

            form = GroupManage(all_name)
            edit = ''
            groups = Group.objects.all()

            for i in groups:
```

```
                edit += ('''
                        <div style = "border-top: 0.01em solid grey; padding:
                        0.3em;">
                        <label for = "%s">选择学生从班级内删除</label>
                        <a id = %s style = "position: relative; top: 1em;" onclick
                        = "manage(id)">%s</a>
                        <br>
                        <div id = "%s" style = "margin-top: 2em;">
                        <label for = "%s"><input id = "%s" name = "delete_name"
                        type="checkbox" value="%s">删除班级</label><br>
                        </div></div>
                        <br>'''
                        % (i.name,i.name,i.name,i.name + '-div',i.name,i.name,i.name
                        ))

            return form,edit


        if request.method != 'POST':
            form,edit = create_form('all')
            return render(request, 'it_exam/manage_group.html', {'form': form, 'edit'
            : mark_safe(edit), 'htmlmsg': msg})


        form = dict(request.POST)
        try:
            query = form['q'][0]
        except KeyError:
            try:
                name_of_group = form['name'][0]

            except KeyError:
                try:
                    for name_of_group in form['delete_name']:
                        Group.objects.get(name = name_of_group).delete()
                    return HttpResponseRedirect(reverse('managegroup'))
                except KeyError:
                    try:
                        for i in form['group']:
                            i = i.split('|')
                            user = User.objects.get(username = i[1])
                            group = Group.objects.get(name = i[0])
                            group.user_set.remove(user)
                    except KeyError:
                        pass
                    return HttpResponseRedirect(reverse('managegroup'))

            if name_of_group.isalpha() or name_of_group.isdecimal():
                pass
            else:
                form,edit = create_form('all')
                form.errors['name'] = ['班级名称只能有数字、字母和中文']
                return render(request, 'it_exam/manage_group.html', {'form': form,
                'edit': mark_safe(edit), 'htmlmsg': msg})

            try:
                test = form['group']
            except KeyError:
                form,edit = create_form('all')
                form.errors['group'] = ['没有选择学生']
                return render(request, 'it_exam/manage_group.html', {'form': form,
                'edit': mark_safe(edit), 'htmlmsg': msg})

            group = Group(name = name_of_group + ':' + username)

            try:
                group.save()
            except IntegrityError:
                pass

            for i in form['group']:
                user = User.objects.get(username = i)
```

```
                group = Group.objects.get(name = name_of_group + ':' + username)
                group.user_set.add(user)

            return HttpResponseRedirect(reverse('managegroup'))


        form,edit,msg = create_form(query)
        return render(request, 'it_exam/manage_group.html', {'form': form, 'edit':
        mark_safe(edit), 'htmlmsg': msg})

    return HttpResponseRedirect(reverse('mainpage'))


#用户主页设计
def mainpage(request):

    username = function.is_login(request)

    if type(username) != type('string'):
        return username


    mail = User.objects.get(username = username).email
    UserHistory.objects.all().filter(Q(user__username = username) & (Q(history = mail
    ) | Q(history = 'CAJ') | Q(history = 'PAT'))).delete()
    CountUserStopTime.objects.all().filter(username = username).delete()
    session = UserSession.objects.get(user__username = username)
    session.is_pick = 0
    session.save()

    #管理员
    if User.objects.get(username = username).is_superuser:

        if request.method != 'POST':

            #获取用户搜索的值,若没有查询值则返回空字符串
            q_value = request.GET.get('q','')
            class_value = request.GET.get('class','')

            user = ''
            user_list = []
            searched = False
            has = False

            #没有提供搜索参数的情况
            if q_value == '' and class_value == '':
                msg = mark_safe("<h3>没有提供搜索参数</h3>")
                return render(request, 'it_exam/show_result.html', {'username':
                username, 'user': '', 'htmlmsg':msg, 'script': ''})

            #所有学生和班级的情况
            elif q_value == 'all':
                all_user = User.objects.all().filter(is_superuser = 0)


            #指定学生或班级的情况
            else:
                searched = True

                #判断是否输入班级
                if class_value != '':
                    try:
                        group = Group.objects.get(name = class_value)
                    except ObjectDoesNotExist:
                        msg = mark_safe("<h3>没有找到对应的班级</h3>")
                        return render(request, 'it_exam/show_result.html', {
                        'username':username, 'user': '', 'htmlmsg':msg, 'script': ''
                        })


                #只写了班级没写学生
                if q_value == '':
```

```
                    all_user = User.objects.all().filter(Q(is_superuser = 0) & Q(
                    groups__name = class_value))
                    has = True
                    searched = False


                else:
                    #如果有班级，在班级内查询
                    if class_value != '':
                        all_user = User.objects.all().filter(Q(is_superuser = 0) & Q(
                        groups__name = class_value))
                    #如果没有，在全部学生中查询
                    else:
                        all_user = User.objects.all().filter(is_superuser = 0)

                    for i in all_user:
                        if q_value in UserGrade.objects.get(user__username = i.
                        username).name:
                            usergrade = UserGrade.objects.get(user__username = i.
                            username)
                            user_list.append(usergrade.name + ':' +
                                    '正确率' + ':' + usergrade.accuracy_percent +
                                    '正确题数' + ':' + str(usergrade.correct) +
                                    '错误题数' + ':' + str(usergrade.wrong)
                                    )
                            has = True


                if not has:
                    msg = mark_safe("<h3>没有找到对应的学生</h3>")
                    return render(request, 'it_exam/show_result.html', {'username':
                    username, 'user': '', 'htmlmsg':msg, 'script': ''})


            #读取每个<button>显示的内容
            if not searched:
                for i in all_user:

                    usergrade = UserGrade.objects.get(user__username = i.username)

                    user_list.append(usergrade.name + ':' +
                                    '正确率' + ':' + usergrade.accuracy_percent +
                                    '正确题数' + ':' + str(usergrade.correct) +
                                    '错误题数' + ':' + str(usergrade.wrong) +
                                    '覆盖率' + ':' + str(usergrade.cover_rate)
                                    )


            #动态生成HTML
            id = 0
            for i in range(len(user_list)):
                user += '<p><button id="' + str(id) + '" onclick=insert(' + str(id) +
                 ',"' + all_user[i].username + '")>' + user_list[i] + '</button></p>'
                id += 1

            return render(request ,'it_exam/show_result.html',{'username': username,
            'htmlmsg':'', 'user':mark_safe(user), 'script': mark_safe('''<script>

                                    function getCookie(name) {

                                    let cookieValue = null;

                                    if (document.cookie && document.cookie !== '') {

                                        const cookies = document.cookie.split(';');

                                        for (let i = 0; i < cookies.length; i++) {

                                            const cookie = cookies[i].trim();

                                            // Does this cookie string begin with the
    name we want?
```

```
                                        if (cookie.substring(0, name.length + 1) === (name + '=')) {
                                            cookieValue = decodeURIComponent(cookie.substring(name.length + 1));
                                            break;
                                        }
                                    }
                                }
                                return cookieValue;
                            }

                            function insert(id,username){
                                const target = document.getElementById(id);

                                if (target.classList.contains("clicked")) {
                                    var nextElement = target.nextSibling;
                                    nextElement.parentNode.removeChild(nextElement);
                                    target.classList.remove('clicked');
                                }
                                else {
                                    var body = document.querySelector('body');
                                    body.style.setProperty("pointer-events", "none")
                                    target.setAttribute("class", "clicked");
                                    let csrftoken = getCookie('csrftoken');
                                    $(document).ready(function() {
                                        $.ajax({
                                        url: '/itexam/getdata/' + username + '/',
                                        type: 'POST',
                                        headers: {'X-CSRFToken': csrftoken},
                                        mode: 'same-origin',
                                        success: function(response) {
                                                let user = '';
                                                for(let i = 0; i < response.length; i++){
                                                    user += '<option value = "' + response[i]['history'] + ',' + response[i]['date'].replace('T','%20') + ',' + response[i]['user_id'] +'">做题时间:' + response[i]['date'].replace('T',' ').slice(0,19) +
```

```
' 正确率:' + response[i]['accuracy_percent'] +
' 正确题数:' + response[i]['correct'] +
' 错误题数:' + response[i]['wrong'] +
'</option>';
                                                    };


target.insertAdjacentHTML('afterend', '<form action = "/itexam/mainpage/"
method="post" class="MainGet" style = "display:flex; max-width:100%" align="center"
id="UserP"><input type="hidden" name="csrfmiddlewaretoken" value=' + csrftoken +
'><tr><th><label for="id_history">做题历史记录:</label></th><td><select
name="history" id="id_history">' + user + '</select></td></tr><input style = "width:
100%; max-width: 90%;" type="submit" value="查看">  </form>');

body.style.setProperty("pointer-events", "auto")
                                                }
                                            });
                                        });
                                    }

                                }
                                </script>
                            ''')})


        else:

            #为用户显示此次做题的情况
            form = request.POST

            #数据类型: '253:wrong|no_reply|A, 294:wrong|no_reply|B,.....
            21:wrong|no_reply|C, 526:wrong|no_reply|D'

            single_history = form['history'].split(',')
            datatime = single_history[-2]
            user = User.objects.get(id = single_history[-1]).username
            single_history.pop()
            single_history.pop()
            return function.show_history(single_history,user,datatime)


    #学生
    else:
        return function.mainpage_show(request,username)


#用户错题集
def wrongproblem(request):

    #判断是否登录
```

```
    username = function.is_login(request)

    if type(username) != type('string'):
        return username

    all_question = ''
    all_choice_question = ''
    all_fill_question = ''
    all_judge_question = ''
    all_do_question = ''
    all_program_question = ''
    all_turtle_question = ''

    if UserGrade.objects.get(user__username = username).wrong_problems.split(',') ==
    ['']:
        return HttpResponse('''
        <head>
            <meta name="viewport" content="width=device-width, initial-scale=1.0">
            <title>学考练习系统</title>
            <link rel="stylesheet" href="/static/it_exam/response.css">
        </head>''' +

        '<body>' +

            '''
            <div class="Header">''' +
                '<h1 style="background-color: transparent; text-align: center;
                font-size: 40px; line-height: 100px;">错题集' +
                '</h1>' +
            '</div>' +

            '<div style = "margin-top: 150px; margin-left: 12%;margin-right:
            12%;height: fit-content;">' +
            all_question +
            '</div>' +

            '</body>'
        )

    for i in UserGrade.objects.get(user__username = username).wrong_problems.split(
    ','):
        try:
            user_question = Exam.objects.get(id = i)
        except ObjectDoesNotExist:
            continue
        index = ['A','B','C','D']
        color = ['transparent','transparent','transparent','transparent']
        if user_question.pattern in ['1','0']:
            if user_question.answer == 'A':
                color[0] = 'palegreen'

            elif user_question.answer == 'B':
                color[1] = 'palegreen'

            elif user_question.answer == 'C':
                color[2] = 'palegreen'

            elif user_question.answer == 'D':
                color[3] = 'palegreen'

        if user_question.pattern == '1':
            all_choice_question += ('<div class="Question">' +
                                    user_question.question +
                                    '</div><br>' +

                                    '<label class="Option" style="background-color:
                                    %s;">' % color[0] +
                                    'A.' + user_question.answer_a +
                                    '</label><br>' +

                                    '<label class="Option" style="background-color:
                                    %s;">' % color[1] +
```

```
                                    'B.' + user_question.answer_b +
                                    '</label><br>' +

                                    '<label class="Option" style="background-color:
                                    %s;">' % color[2] +
                                    'C.' + user_question.answer_c +
                                    '</label><br>' +

                                    '<label class="Option" style="background-color:
                                    %s;">' % color[3] +
                                    'D.' + user_question.answer_d +
                                    '</label><br><br>')

        if user_question.pattern == '0':
            all_judge_question += ('<div class="Question">' +
                                    user_question.question +
                                    '</div><br>' +

                                    '<label class="Option" style="background-color:
                                    %s;">' % color[0] +
                                    'A.' + user_question.answer_a +
                                    '</label><br>' +

                                    '<label class="Option" style="background-color:
                                    %s;">' % color[1] +
                                    'B.' + user_question.answer_b +
                                    '</label><br><br>')

        if user_question.pattern == '9':
            all_fill_question += ('<div class="Question">' +
                                    user_question.question +
                                    '</div><br><br>')

        if user_question.pattern == '3':
            all_do_question += ('<div class="Question">' +
                                    user_question.question +
                                    '</div><br><br>')

        if user_question.pattern == '4' or user_question.pattern == '5' or
        user_question.pattern == '6':
            name = user_question.question[user_question.question.find('>',20) + 1:
            user_question.question.find('<',20)]
            all_program_question += ('<div class="Question">' +
                                    user_question.question + '<br>' +
                                    '正确答案是:' + ('<a href="%s"
                                    download="%s">参考答案</a>' % (
                                    'http://192.144.180.87:8000/static/it_exam/编程题
                                    答案/' + name + '.py',name)) + '<br>'
                                    '</div><br><br>')

        if user_question.pattern == '8':
            all_turtle_question += ('<div class="Question">' +
                                    user_question.question +
                                    '</div><br><br>')

    all_question = all_choice_question + all_judge_question + all_fill_question +
    all_do_question + all_program_question + all_turtle_question

    return HttpResponse('''
        <head>
            <meta name="viewport" content="width=device-width, initial-scale=1.0">
            <title>学考练习系统</title>
            <link rel="stylesheet" href="/static/it_exam/response.css">
        </head>''' +

        '<body>' +

            '''
            <div class="Header">''' +
                '<h1 style="background-color: transparent; text-align: center;
                font-size: 40px; line-height: 100px;" id="counter">错题集' +
                '</h1>' +
```

```
            '</div>' +

            '<div style = "margin-top: 150px; margin-left: 12%;margin-right:
            12%;height: fit-content;">' +
            all_question +
            '</div>' +

            '</body>'
            )


#账号登陆
def log_in(request):
    user_name = ''
    user_password = ''

    #接收用户输入的账号与密码
    if request.method != 'POST':
        form = Login()
        return render(request, 'it_exam/login.html', {'form': form})

    form = Login(request.POST)
    if form.is_valid():
        user_name = form.cleaned_data['username'].replace(' ','')
        user_password = form.cleaned_data['password']

        #验证用户
        user = authenticate(request, username = user_name, password = user_password)

        #验证通过,
        if user is not None:

            #未登录状态
            try:
                test = request.session['username']
            except KeyError:
                login(request,user)
                request.session['username'] = user_name
                session = UserSession.objects.get(user__username = user_name)
                session.is_authenticated = 1
                session.session_key = request.session.session_key
                session.save()
                return HttpResponseRedirect(reverse('mainpage'))

            #如果本设备有登陆的账号,拒绝登陆
            login(request,user)
            request.session['username'] = user_name
            return HttpResponse('''
            <!DOCTYPE html>
            <html>
                <meta name="viewport" content="width=device-width, initial-scale=1.0">
                <meta charset="UTF-8">
                <head>
                    <link rel="stylesheet" href="/static/it_exam/transition.css">
                    <title>学考练习系统</title>
                </head>
                <body>
                    <h1>
                    该设备已有账号登陆或有账号未登出!
                    </h1>
                    <a href = "/itexam/">重新登陆</a>
                    <p>无法登出? <a href = "/itexam/logout/">强制登出</a></p>
                </body>
            </html>''')

        else:
            form.errors['password'] = ['账号或密码错误']
            return render(request, 'it_exam/login.html', {'form': form})
    else:
        form.errors['password'] = ['账号或密码错误']
        return render(request, 'it_exam/login.html', {'form': form})

```

```
1706
1707    #403错误
1708    def permission_denied(request, reason=""):
1709        return render(request, 'it_exam/403.html')
1710
1711
1712    #404错误
1713    def page_not_found(request, exception):
1714        return render(request, 'it_exam/404.html')
1715
1716
1717
1718    #账号注册
1719    def log_on(request):
1720        if request.method != 'POST':
1721            form = Logon()
1722            return render(request, 'it_exam/logon.html', {'form': form, 'email':
                '@i.pkuschool.edu.cn', 'error':''})
1723
1724        #接收用户输入的邮箱
1725        all_mail = []
1726        all_username = []
1727        all_user = User.objects.all()
1728        for i in all_user:
1729            all_mail.append(i.email)
1730        for i in all_user:
1731            all_username.append(i.username)
1732
1733
1734        form = request.POST
1735        if 'number' in form.keys():
1736
1737            #如果选择自动填充，查找并填充
1738            email = function.get_email_by_number(form['number'])
1739            if email:
1740                form = Logon()
1741                return render(request, 'it_exam/logon.html', {'form': form, 'email':email
                    , 'error':''})
1742
1743            form = Logon()
1744            return render(request, 'it_exam/logon.html', {'form': form, 'email':
                '@i.pkuschool.edu.cn', 'error':mark_safe('<ul
                class="errorlist"><li>找不到此学号,请确认输入的是正确的学号</li></ul>')})
1745
1746
1747        form = Logon(request.POST)
1748
1749        if form.is_valid():
1750            if not form.cleaned_data['email'].endswith('i.pkuschool.edu.cn') :
1751                form.errors['email'] = ['请输入地址为"i.pkuschool.edu.cn"的邮箱']
1752                return render(request, 'it_exam/logon.html', {'form': form, 'email':
                    '@i.pkuschool.edu.cn', 'error':''})
1753
1754            user_name = form.cleaned_data['username']
1755            user_password = form.cleaned_data['password']
1756            mail = form.cleaned_data['email']
1757
1758
1759            for i in user_password:
1760                if i.isalpha() or i.isdecimal():
1761                    pass
1762                else:
1763                    form.errors['password'] = ['密码只能有数字和字母']
1764                    return render(request, 'it_exam/logon.html', {'form': form, 'email':
                        '@i.pkuschool.edu.cn', 'error':''})
1765
1766            for i in user_name:
1767                if i.isalpha() or i.isdecimal() or '_' in i:
1768                    pass
1769                else:
1770                    form.errors['username'] = ['用户名只能有数字、字母、下划线和中文']
1771                    return render(request, 'it_exam/logon.html', {'form': form, 'email':
```

```
                '@i.pkuschool.edu.cn', 'error':''})

        if mail in all_mail:
            form.errors['email'] = ['该邮箱已注册']
            return render(request, 'it_exam/logon.html', {'form': form, 'email':
            '@i.pkuschool.edu.cn', 'error':''})

        if user_name in all_username:
            form.errors['username'] = ['该用户名已被使用']
            return render(request, 'it_exam/logon.html', {'form': form, 'email':
            '@i.pkuschool.edu.cn', 'error':''})

        is_superuser = 0
        #如果是maohuajun@i.pkuschool.edu.cn,设置为管理员
        if mail == 'maohuajun@i.pkuschool.edu.cn':
            is_superuser = 1

        #发送邮件

        #生成确认码
        ensurekey = ''
        logon_time = now()

        for i in range(50):
            ensurekey += str(random.randint(0,9))

        try:
            send_mail(
            subject = '练习网站注册',
            message = '',
            html_message =
            '<h3>注意:如非本人操作,请忽略此邮件%s。</h3><br><h4>点击%s以完成认证</h4>'

            % ('<br>检测到您为管理员,可通过<a href =
            "http://192.144.180.87:8000/adminsite/">此链接</a>管理用户'
               if is_superuser else '','<a href = "http://192.144.180.87:8000/' +
               reverse('ensure',args = [ensurekey,str(logon_time),'logon']) +
               '">认证链接</a>'),

            from_email = 'webmaster163mail <%s>' % (settings.EMAIL_HOST_USER),
            recipient_list = [mail],
            fail_silently=False
            )
        except (BadHeaderError,ValueError,SMTPRecipientsRefused):
            return HttpResponse('发送失败,请重新操作!')

        Assurance.objects.create(key = ensurekey, date = logon_time,
                                 email = mail, username = user_name,
                                 password = make_password(user_password),
                                 task = 'logon', is_superuser = is_superuser)

        return HttpResponse('''
        <!DOCTYPE html>
        <html>
            <meta name="viewport" content="width=device-width, initial-scale=1.0">
            <meta charset="UTF-8">
            <head>
                <link rel="stylesheet" href="/static/it_exam/transition.css">
                <title>学考练习系统</title>
            </head>
            <body>
                <h1>
                成功发送至%s,请到此邮箱打开激活网址!
                </h1>
                <a href = "/itexam/">登陆</a>
                <a href = "https://outlook.office.com/">点击进入outlook邮箱</a>
            </body>
        </html>'''  % mail)

    else:
        form.errors['email'] = ['请输入有效的邮箱']
        return render(request, 'it_exam/logon.html', {'form': form, 'email':
```

```
            '@i.pkuschool.edu.cn', 'error':''})


#注册或找回密码确认
def logon_ensure_or_enter_new_password(request,number,time,task):
    if task == 'logon':
        try:
            user = Assurance.objects.get(Q(key = number) & Q(date = str(time).replace
            ('%20',' ')) & Q(task = 'logon'))
        except ObjectDoesNotExist:
            return HttpResponse('<link rel="stylesheet"
            href="/static/it_exam/transition.css"><h1>确认网址不正确<h1\>')
        User.objects.create_user(username = user.username, password = 'creating',
        email = user.email, is_superuser = user.is_superuser, is_staff = user.
        is_superuser)
        logon_user = User.objects.get(username = user.username)
        logon_user.password = user.password
        logon_user.save()

        #为用户添加对应的记录， 1的对象.多的模型_set，获得1对多的数据集
        user_account = User.objects.get(email = user.email)
        user_account.usergrade_set.create(total_exam = 0, correct = 0, wrong = 0,
        accuracy_percent = '0%', cover_rate = '0%')
        function.name()
        user_account.usersession_set.create(is_authenticated = 0, is_pick = 0,
        session_key = '')

        Assurance.objects.get(key = number).delete()
        return HttpResponse('<link rel="stylesheet"
        href="/static/it_exam/transition.css"><h1>注册成功</h1><a href =
        "/itexam/">登陆</a>')

    elif task == 'forgetpassword':
        try:
            user = Assurance.objects.get(Q(key = number) & Q(date = str(time).replace
            ('%20',' ')) & Q(task = 'forgetpassword') & Q(password = 'finding'))
        except ObjectDoesNotExist:
            return HttpResponse('<link rel="stylesheet"
            href="/static/it_exam/transition.css"><h1>确认网址不正确<h1\>')

        #确保表单网址正确
        url = user.key + '/' + str(user.date) + '/' + user.task
        if request.method != 'POST':
            form = EnterNewPassword()
            username = User.objects.get(email = user.email).username
            return render(request, 'it_exam/enternewpassword.html', {'form': form,
            'url': url, 'username': username})
        form = EnterNewPassword(request.POST)

        #重设密码
        if form.is_valid():
            try:
                user = User.objects.get(email = user.email)
            except ObjectDoesNotExist:
                return HttpResponse('<link rel="stylesheet"
                href="/static/it_exam/transition.css"><h1>用户不存在<h1\><br><a
                href="/itexam/">登陆</a>')
            user.set_password(form.cleaned_data['newpassword'])
            user.save()
            Assurance.objects.get(key = number).delete()
            return HttpResponse('''
            <!DOCTYPE html>
            <html>
                <meta name="viewport" content="width=device-width, initial-scale=1.0">
                <meta charset="UTF-8">
                <head>
                    <link rel="stylesheet" href="/static/it_exam/transition.css">
                    <title>学考练习系统</title>
                </head>
                <body>
                    <h1>
```

```
                    更改成功!
                    </h1>
                    <a href = "/itexam/">登陆</a>
                </body>
            </html>''')


#找回密码
def forget_password(request):
    all_mail = []
    all_user = User.objects.all()
    for i in all_user:
        all_mail.append(i.email)

    if request.method != 'POST':
        form = ForgetPassword()
        return render(request, 'it_exam/forgetpassword.html', {'form': form, 'email':
        '@i.pkuschool.edu.cn', 'error':''})


    form = request.POST
    if 'number' in form.keys():

        #如果选择自动填充，查找并填充
        email = function.get_email_by_number(form['number'])
        if email:
            form = ForgetPassword()
            return render(request, 'it_exam/forgetpassword.html', {'form': form,
            'email':email, 'error':''})

        form = ForgetPassword()
        return render(request, 'it_exam/forgetpassword.html', {'form': form, 'email':
        '@i.pkuschool.edu.cn', 'error':mark_safe('<ul
        class="errorlist"><li>找不到此学号,请确认输入的是正确的学号</li></ul>')})


    form = ForgetPassword(request.POST)

    if form.is_valid():
        mail = form.cleaned_data['email']

        if mail not in all_mail:
            return HttpResponse('''
            <!DOCTYPE html>
            <html>
                <meta name="viewport" content="width=device-width, initial-scale=1.0">
                <meta charset="UTF-8">
                <head>
                    <link rel="stylesheet" href="/static/it_exam/transition.css">
                    <title>学考练习系统</title>
                </head>
                <body>
                    <h1>
                    %s,该邮箱没有账号!
                    </h1>
                    <a href = "/itexam/">登陆</a>
                </body>
            </html>'''  % mail)

        #生成确认码
        ensurekey = ''
        forgetpassword_time = now()

        for i in range(50):
            ensurekey += str(random.randint(0,9))

        try:
            send_mail(
            subject = '练习网站密码找回',
            message = '',
            html_message = '<h3>请点击<a
            href="%s">此链接</a>以继续账号或密码找回</h3>'
```

```
                % ('http://192.144.180.87:8000' + reverse('ensure',args = [ensurekey,str(
                forgetpassword_time),'forgetpassword'])),

                from_email = 'webmaster163mail <%s>' % (settings.EMAIL_HOST_USER),
                recipient_list = [mail],
                fail_silently=False
                )
            except (BadHeaderError,ValueError,SMTPRecipientsRefused) as reason:
                return HttpResponse('发送失败,原因为%s请重新操作!' % reason)

            Assurance.objects.create(key = ensurekey, date = forgetpassword_time,
                                     email = mail, username = 'finding',
                                     password = 'finding', task = 'forgetpassword',
                                     is_superuser = User.objects.get(email = mail).
                                     is_superuser)

            return HttpResponse('''
            <!DOCTYPE html>
            <html>
                <meta name="viewport" content="width=device-width, initial-scale=1.0">
                <meta charset="UTF-8">
                <head>
                    <link rel="stylesheet" href="/static/it_exam/transition.css">
                    <title>学考练习系统</title>
                </head>
                <body>
                    <h1>
                    成功将找回链接发送至%s!
                    </h1>
                    <a href = "/itexam/">登陆</a>
                    <a href = "https://outlook.office.com/">点击进入outlook邮箱</a>
                </body>
            </html>'''  % mail)

        else:
            form.errors['email'] = ['请输入有效的邮箱']
            return render(request, 'it_exam/forgetpassword.html', {'form': form, 'email':
            '@i.pkuschool.edu.cn', 'error':''})


    #抽取题目,共46题
    def pick(request,mode):

        #判断是否登录
        username = function.is_login(request)

        if type(username) != type('string'):
            return username

        user_email = User.objects.get(username = username).email
        user = username

        session = UserSession.objects.get(user__email = user_email)

        if mode != 'ChoiceAndJudge' and mode != 'All' and mode != 'ProgramAndTackle':
            return HttpResponseRedirect(reverse('mainpage'))


        if User.objects.get(email = user_email).is_superuser:
            show = 'mainpage_show'

        else:
            show = 'mainpage'


        if session.is_pick == 0:

            if mode == 'ChoiceAndJudge':
                number_of_question = 40

            elif mode == 'All':
```

```
2031                number_of_question = 46
2032
2033            else:
2034                number_of_question = 6
2035
2036            #标记为已经抽取过题目
2037            session.is_pick = 1
2038            session.save()
2039
2040        else:
2041            history = UserHistory.objects.get(user__email = user_email,accuracy_percent =
                 '')
2042
2043            amount = len(history.question.split(','))
2044            if amount == 46:
2045                number_of_question = 46
2046
2047            elif amount == 40:
2048                number_of_question = 40
2049
2050            else:
2051                number_of_question = 6
2052
2053            recent_pick = history.question.split(',')
2054            mode = history.history
2055
2056            #计算时间确保一致
2057            try:
2058                user_time = CountUserStopTime.objects.get(username = user)
2059                user_time.is_stop = 0
2060            except ObjectDoesNotExist:
2061                pass
2062            time = history.date
2063            time_now = now()
2064            minute = 90 if mode == user_email else 40
2065            second = (time_now - time).seconds - user_time.all_stop_time
2066            minute = minute - (second // 60 + 1) if second % 60 != 0 else second // 60
2067            sec = 60 - (second % 60)
2068            user_time.save()
2069            return render(request, "it_exam/学考练习系统.html", {"Timian": function.pick(
                user_email,mode,recent_pick),'number_of_questions':number_of_question, 'show'
                :show, 'min':minute, 'sec':sec})
2070
2071        CountUserStopTime.objects.create(username = user,stop_time = now(),is_stop =
            False)
2072        exam_paper_ready = function.pick(user_email,mode,'')
2073        return render(request, "it_exam/学考练习系统.html", {"Timian": exam_paper_ready,
            'number_of_questions':number_of_question, 'show':show, 'min':90 if mode == 'All'
            else 40, 'sec':0})
2074
2075
2076    # 当题目已经提交,调用此函数
2077    def have_submit_or_mainpage(request):
2078
2079        #判断是否登陆
2080        username = function.is_login(request)
2081
2082        if type(username) != type('string'):
2083            return username
2084
2085        return HttpResponse('''
2086            <!DOCTYPE html>
2087            <html>
2088                <meta name="viewport" content="width=device-width, initial-scale=1.0">
2089                <meta charset="UTF-8">
2090                <head>
2091                    <link rel="stylesheet" href="/static/it_exam/transition.css">
2092                    <title>学考练习系统</title>
2093                </head>
2094                <body>
2095                    <h1>
2096                        该套题目已经提交或已被放弃提交
```

```
                </h1>
                <a href = "/itexam/mainpage/">主页</a>
            </body>
        </html>''')


#判断题目正误
def answer(request):

    #判断是否登陆
    username = function.is_login(request)

    if type(username) != type('string'):
        return username

    user = username

    session = UserSession.objects.get(user__username = user)
    session.is_pick = 0
    session.save()

    try:
        CountUserStopTime.objects.get(username = user).delete()
    except ObjectDoesNotExist:
        pass

    #读取答案
    if request.method == 'POST':

        #读取题目
        try:
            history_for_question = UserHistory.objects.get(Q(user__username = user) &
             Q(accuracy_percent = ''))
            question = history_for_question.question
        except ObjectDoesNotExist:
            return HttpResponseRedirect(reverse('have_submit_or_mainpage'))

        #读取用户基本信息
        user_ans = []
        datatime = now()
        wrong = 0
        correct = 0
        have_file = False
        name_of_file = ''

        ans = request.POST

        index = 1
        while index < 47:

            if history_for_question.history == 'PAT':
                mode = 'PAT'
                #只处理文件
                if index > 6:
                    break

                try:
                    file = request.FILES['p' + str(index)]

                    if not file.name.endswith('.docx') and index == 2:
                        user_ans.append('文件类型不正确§№☆★○●◎◇综合题必须是word文档')
                    elif not file.name.endswith('.py') and index != 2:
                        user_ans.append(
                        '文件类型不正确§№☆★○●◎◇编程/海龟画图题必须是python文件')
                    else:
                        have_file = True
                        user_ans.append(file.name.replace('|','').replace(':',
                        '○●○⊕◎⊖○¤㈱㊣★☆').replace(',','§№☆★○●◎◇'))
                        information_of_file = str(history_for_question.date)[0:19].
                        replace('%20','').replace(':','.') + history_for_question.
                        user.username
                        os.makedirs('it_exam/storage/' + information_of_file,
```

```
                        exist_ok = True)
2164
2165                    #创建压缩文件
2166                    with ZipFile(file = 'it_exam/storage/' + information_of_file
                        + '/' + file.name + '.zip', mode = 'w') as f:
2167                        f.writestr(file.name,file.read(),compresslevel = 9)
2168                    name_of_file += (str(index - 42) + '|' + file.name + '.zip').
                        replace('#','□□囍□□☑✔□⊠') + ','
2169
2170                except KeyError:
2171                    user_ans.append('no_reply')
2172
2173                index += 1
2174                continue
2175
2176            try:
2177                #处理文件
2178                if index >= 42:
2179                    try:
2180                        file = request.FILES['p' + str(index)]
2181
2182                        if not file.name.endswith('.docx') and index == 42:
2183                            user_ans.append(
                            '文件类型不正确§№☆★○●◎◇综合题必须是word文档')
2184                        elif not file.name.endswith('.py') and index != 42:
2185                            user_ans.append(
                            '文件类型不正确§№☆★○●◎◇编程/海龟画图题必须是python文件')
2186                        else:
2187                            have_file = True
2188                            user_ans.append(file.name.replace('|','').replace(':',
                            '⊙●○⊕◎Θ○¤㈱㊣★☆').replace(',','§№☆★○●◎◇'))
2189                            information_of_file = str(history_for_question.date)[0:19
                            ].replace('%20','').replace(':','.') +
                            history_for_question.user.username
2190                            os.makedirs('it_exam/storage/' + information_of_file,
                            exist_ok = True)
2191
2192                            #创建压缩文件
2193                            with ZipFile(file = 'it_exam/storage/' +
                            information_of_file + '/' + file.name + '.zip', mode =
                            'w') as f:
2194                                f.writestr(file.name,file.read(),compresslevel = 9)
2195                            name_of_file += (str(index - 42) + '|' + file.name +
                            '.zip').replace('#','□□囍□□☑✔□⊠') + ','
2196
2197                    except KeyError:
2198                        user_ans.append('no_reply')
2199
2200                    index += 1
2201                    continue
2202
2203                user_answer = ans['p' + str(index)]
2204
2205                #四十一题特殊情况
2206                if user_answer == '' and index == 41:
2207                    user_ans.append('no_reply')
2208                    index += 1
2209                    continue
2210
2211                #保存用户答案
2212                user_ans.append(user_answer.replace('|','').replace(':',
                    '⊙●○⊕◎Θ○¤㈱㊣★☆').replace(',','§№☆★○●◎◇'))
2213            except KeyError:
2214
2215                #若没有作答，则保存no_reply
2216                user_ans.append('no_reply')
2217                index += 1
2218                continue
2219            index += 1
2220
2221        #有文件则存储其地址
2222        if have_file:
```

```
            history_for_question.filename = 'http://192.144.180.87:8000/storage/' +
            information_of_file + '#' + name_of_file
            history_for_question.save()

        #个人账户数据修改
        account = UserGrade.objects.get(user__username = user)
        aggregations = set(account.wrong_problems.split(','))
        aggregations_question = set(account.question.split(','))

        #去除split所带来的影响（使空字符串长度为1）
        if len(aggregations) == 1:
            aggregations.pop()

        if len(aggregations_question) == 1:
            aggregations_question.pop()


        #判断正误
        question_list = question.split(',')
        for i in range(46):

            #判断是否为编程题
            #是
            '''if i == 42 or i == 43 or i == 44:

                #创建并给虚拟环境的全局变量赋值
                safe_globals['user_answer'] = '未收到用户赋值'

                try:
                    compiled_code = compile_restricted(user_ans[i], '<string>',
                    'exec')
                    exec(compiled_code,safe_globals)
                except Exception as reason:
                    print(reason)
                    # 处理错误，例如不允许的操作或访问限制
                    return HttpResponse('<h1>输入错误</h1>')
                user_answer = safe_globals['user_answer']


                if correct_answer.answer == str(user_answer):

                    #本次做题数据修改
                    question.correct += 1
                    question_list[i] = question_list[i] + ':correct|' +
                    str(user_answer) + '|' + correct_answer.answer

                    #个人账户数据修改
                    account.correct += 1

                    #题目库数据修改
                    correct_answer.correct += 1
                    correct_answer.correct_possibility = str((correct_answer.correct
                    * 100) / (correct_answer.correct + correct_answer.wrong)) + '%'
                    correct_answer.save()

                else:

                    #本次做题数据修改
                    question.wrong += 1
                    question_list[i] = question_list[i] + ':wrong|' +
                    str(user_answer) + '|' + correct_answer.answer

                    #个人账户数据修改
                    account.wrong += 1

                    #以集合的形式防止重复录入错题
                    aggregations.add(question.history.split(',')[i])


                    #题目库数据修改
                    correct_answer.wrong += 1
                    correct_answer.correct_possibility = str((correct_answer.correct
```

```
                    * 100) / (correct_answer.correct + correct_answer.wrong)) + '%'
2290                    correct_answer.save()'''
2291
2292                #否
2293                #else:
2294                try:
2295                    correct_answer = Exam.objects.get(id = question_list[i])
2296                except (IndexError, ObjectDoesNotExist):
2297                    continue
2298                correct_answer.total += 1
2299                account.total_exam += 1
2300
2301                #录入做过的题
2302                aggregations_question.add(question.split(',')[i])
2303
2304                #记录每道题被作数，正确数和错误数以及个人的做题数正确数和错误数
2305                if correct_answer.answer == user_ans[i]:
2306
2307                    #本次做题数据修改
2308                    correct += 1
2309                    question_list[i] = question_list[i] + ':correct|' + user_ans[i] + '|'
                     + correct_answer.answer
2310
2311                    #个人账户数据修改
2312                    if question_list[i].split(':')[0] in account.wrong_problems.split(','
                    ):
2313                        aggregations.discard(question_list[i].split(':')[0])
2314                    account.correct += 1
2315
2316                    #题目库数据修改
2317                    correct_answer.correct += 1
2318                    correct_answer.correct_possibility = str(round((correct_answer.
                    correct * 100) / (correct_answer.correct + correct_answer.wrong),2))
                    + '%'
2319                    correct_answer.save()
2320
2321                else:
2322
2323                    #本次做题数据修改
2324                    wrong += 1
2325                    question_list[i] = question_list[i] + ':wrong|' + user_ans[i] + '|' +
                     correct_answer.answer
2326                    #个人账户数据修改
2327                    account.wrong += 1
2328
2329                    #以集合的形式防止重复录入错题
2330                    aggregations.add(question.split(',')[i])
2331
2332
2333                    #题目库数据修改
2334                    correct_answer.wrong += 1
2335                    correct_answer.correct_possibility = str(round((correct_answer.
                    correct * 100) / (correct_answer.correct + correct_answer.wrong),2))
                    + '%'
2336                    correct_answer.save()
2337
2338
2339            #个人账户数据修改
2340
2341            #检测账户做题模式
2342            userhistory = UserHistory.objects.get(Q(user__username = user) & Q(
                accuracy_percent = ''))
2343            index = Q(pattern = '0') | Q(pattern = '1') | Q(pattern = '3') | Q(pattern =
                '4')| Q(pattern = '5') | Q(pattern = '6')| Q(pattern = '8') | Q(pattern = '9')
2344
2345            account.accuracy_percent = str(round((account.correct * 100) / account.
                total_exam,2)) + '%'
2346            account.wrong_problems = str(aggregations).replace('{','').replace('}','').
                replace("'",'').replace(' ','')
2347            account.question = str(aggregations_question).replace('{','').replace('}',''
                ).replace("'",'').replace(' ','')
2348            account.cover_rate = str(round((len(account.question.split(',')) * 100) /
```

```
            Exam.objects.filter(index).count(),2)) + '%'
            account.save()

            #本次做题数据修改
            userhistory.history = str(question_list).replace('[','').replace(']','').
            replace("'",'')
            userhistory.date = datatime
            userhistory.accuracy_percent = str(round((correct * 100) / (wrong + correct),
            2)) + '%'
            userhistory.correct = correct
            userhistory.wrong = wrong
            userhistory.save()


            #获取每个题的做题记录
            single_history = UserHistory.objects.get(user__username = user, date = str(
            datatime).replace('%20',' ')).history.split(',')

            return function.show_history(single_history,user,datatime)

        else:
            return HttpResponseRedirect('/itexam/mainpage/')


    #设置时间
    def set_time(request):

        #判断是否登陆
        username = function.is_login(request)

        if type(username) != type('string'):
            return username

        user = username

        if request.method != 'POST':
            return HttpResponseRedirect(reverse('mainpage'))

        user_time = CountUserStopTime.objects.get(username = user)
        if user_time.is_stop:
            user_time.all_stop_time = (now() - user_time.stop_time).total_seconds() +
            user_time.all_stop_time
        else:
            user_time.stop_time = now()
        user_time.is_stop = (False if user_time.is_stop else True)
        user_time.save()

        return JsonResponse([1], safe=False)


    #获取解析
    def get_description(request, id):

        #判断是否登陆
        username = function.is_login(request)

        if type(username) != type('string'):
            return username

        if request.method != 'POST':
            return HttpResponseRedirect(reverse('mainpage'))

        description = Exam.objects.get(id = id).description

        return JsonResponse([description], safe = False)



    #删除没有做完的做题历史
    def delete_history(request):

        #判断是否登陆
```

```
2416        username = function.is_login(request)
2417
2418        if type(username) != type('string'):
2419            return username
2420
2421        if request.method != 'POST':
2422            return HttpResponseRedirect(reverse('mainpage'))
2423
2424        mail = User.objects.get(username = username).email
2425        form = request.POST
2426
2427        UserHistory.objects.all().filter(Q(user__username = username) & (Q(history = mail
            ) | Q(history = 'CAJ') | Q(history = 'PAT'))).delete()
2428        CountUserStopTime.objects.all().filter(username = username).delete()
2429        session = UserSession.objects.get(user__username = username)
2430        session.is_pick = 0
2431        session.save()
2432
2433        return JsonResponse([], safe = False)
```