

```

1  #导入与网页有关
2  from django.shortcuts import render
3  from django.http import HttpResponse,HttpResponseRedirect,HttpResponseForbidden
4  from django.urls import reverse,reverse_lazy
5
6  #导入与认证有关
7  from django.contrib.auth.hashers import make_password
8  from django.contrib.auth import authenticate,login,logout
9  from django.contrib.auth.models import User, Group
10 from django.contrib.auth.decorators import login_required
11 from django.db.utils import IntegrityError
12 from django.views.generic.edit import CreateView, DeleteView, FormView, UpdateView
13
14 #导入与数据库有关
15 from it_exam.models import Exam
16 from it_exam.models import Assurance, UserHistory, UserGrade, UserSession,
CountUserStopTime
17 from django.contrib.sessions.models import Session
18 from django.db.models import Q
19
20 #导入与表单有关
21 from django.views.decorators.csrf import csrf_protect
22 from django.views import defaults
23 from .forms import Logon, Login, ForgetPassword, EnterNewPassword, UserData,
GroupManage, Classes
24
25 #导入与时区有关
26 from django.utils.timezone import now
27
28 #导入与邮箱登陆有关
29 from django.core.mail import BadHeaderError,send_mail
30 from exam_web import settings
31 from smtplib import SMTPRecipientsRefused
32 from django.core.exceptions import ObjectDoesNotExist
33 from django.db.models import Case, When
34
35 #导入与文件有关
36 from zipfile import ZipFile
37 import os
38
39 #导入与Ajax有关
40 import json
41 from django.http import JsonResponse
42
43 #导入与到导出有关
44 import xlwt
45
46 #导入其他
47 import random
48 # Create your views here.
49 from django.utils.safestring import mark_safe
50
51
52
53 #-----
54 #调试函数
55 #-----
56
57 #or (('docx' not in single[1].split('|')[1]) and ('.py' not in
single[1].split('|')[1]))
58 #常用函数
59 class function():
60
61     #显示主页相关内容
62     def mainpage_show(request,username):
63
64         if request.method != 'POST':
65             ready_history = []
66             past = UserHistory.objects.all().filter(user__username = username)
67
68             #整理用户做题历史，是格式为form要求的格式

```

```

68     for each in past:
69         ready_history.append((str(each.history + ',' + str(each.date)),{
            '做题时间:' + str(each.date).replace('+08:00','')[:19]:
70
71             ['正确题数:' + str(each.correct),
72
73             '错误题数:' + str(each.wrong),
74
75             '正确率:' + each.accuracy_percent]}))
76
77 form = UserData(ready_history)
78
79 data = UserGrade.objects.get(user__username = username)
80 all_data = UserGrade.objects.all().exclude(user__username = username)
81 number_of_all_data = UserGrade.objects.count() - 1
82 surpass_number = 0
83
84 for i in all_data:
85     if float(i.accuracy_percent.replace('%','')) < float(data.
86         accuracy_percent.replace('%','')):
87         surpass_number += 1
88
89 surpass_rate = str(round((surpass_number * 100) / number_of_all_data,2))
90 + '%'
91
92 return render(request, 'it_exam/mainpage_examinee.html', {
93     'form':form,
94     'username':username,
95     'accuracy_percent':str(
96         data.accuracy_percent),
97     'correct':str(data.
98         correct),
99     'wrong':str(data.wrong),
100     'cover':data.cover_rate,
101     'how_many': surpass_rate,
102 })
103
104 else:
105
106     #为用户显示此次做题的情况
107     form = request.POST
108
109     #数据类型: '253:wrong|no_reply|A, 294:wrong|no_reply|B,.....
110     21:wrong|no_reply|C, 526:wrong|no_reply|D'
111
112     try:
113         single_history = form['history'].split(',')
114     except KeyError:
115         return HttpResponseRedirect(reverse('mainpage'))
116     datetime = single_history[-1]
117     single_history.pop()
118
119     return function.show_history(single_history,username,datetime)
120
121 #判断是否登录
122 def is_login(request):
123
124     if request.session.get("username", "False") == "False":
125         return HttpResponseRedirect('/itexam/')
126
127     username = request.session.get("username")
128
129     if not UserSession.objects.get(user__username = username).is_authenticated:
130         return HttpResponse(''
131             <!DOCTYPE html>
132             <html>
133                 <meta name="viewport" content="width=device-width,
134                 initial-scale=1.0">
135                 <meta charset="UTF-8">
136                 <head>
137                     <link rel="stylesheet"
138                     href="/static/it_exam/transition.css">

```

```

129         <title>学考练习系统</title>
130     </head>
131     <body>
132         <h1>
133             %s已登出!
134         </h1>
135         <a href = "/itexam/">重新登陆</a>
136     </body>
137 </html>''' % username)
138
139     return username
140
141
142 #显示做题历史
143 def show_history(single_history,user,datetime):
144
145     all_question = ''
146     filename = UserHistory.objects.get(user__username = user, date = str(datetime)
147 ).replace('%20',' ').filename
148     if filename != 'NoFile':
149         file_download = filename.split('#')[0]
150         which_file = filename.split('#')[1].split(',')
151         which_file.remove('')
152         file_dict = {}
153
154         for each_file in which_file:
155             file_dict[int(each_file.split('|')[0])] = each_file.split('|')[1]
156
157     else:
158         file_dict = {}
159         file_download = ''
160         which_file = ['', '', '', '', '', '']
161
162     for j in range(46):
163
164         #数据类型: '253:wrong|no_reply|A, 294:wrong|no_reply|B,.....
165         #21:wrong|no_reply|C, 526:wrong|no_reply|D'
166
167         try:
168             single = single_history[j].split(':')
169         except IndexError:
170             continue
171         single[1] = single[1].replace('○○○○○○○○(株)⊕☆☆', ':').replace('S№☆☆○○○○',
172 ',')
173         #获得这个题的相关数据
174         key = single[0]
175
176         try:
177             user_question = Exam.objects.get(pk = int(key))
178         except ValueError:
179             continue
180
181         name = user_question.question[user_question.question.find('>',20) + 1:
182 user_question.question.find('<',20)]
183         mode = 'PAT' if len(single_history) == 6 else ''
184
185         #为用户展示做题记录
186
187         #为用户标记正确选项与被选选项
188         index = ['A','B','C','D']
189         color = ['transparent','transparent','transparent','transparent']
190         if user_question.pattern in ['1','0']:
191             if single[1].split('|')[1] == single[1].split('|')[2]:
192                 if single[1].split('|')[1] == 'A':
193                     color[0] = 'palegreen'
194
195                 elif single[1].split('|')[1] == 'B':
196                     color[1] = 'palegreen'
197
198                 elif single[1].split('|')[1] == 'C':
199                     color[2] = 'palegreen'

```

```

197         elif single[1].split('|')[1] == 'D':
198             color[3] = 'palegreen'
199
200     else:
201         if single[1].split('|')[1] == 'A':
202             color[0] = 'lightpink'
203             color[index.index(single[1].split('|')[2])] = 'palegreen'
204
205         elif single[1].split('|')[1] == 'B':
206             color[1] = 'lightpink'
207             color[index.index(single[1].split('|')[2])] = 'palegreen'
208
209         elif single[1].split('|')[1] == 'C':
210             color[2] = 'lightpink'
211             color[index.index(single[1].split('|')[2])] = 'palegreen'
212
213         elif single[1].split('|')[1] == 'D':
214             color[3] = 'lightpink'
215             color[index.index(single[1].split('|')[2])] = 'palegreen'
216
217
218 #获取所有选项的答题情况
219 all_question += (
220
221     '<div style="padding-bottom: 2em;">' +
222
223     '<div class="Question">' +
224     str(j + 1) + '. ' + user_question.question +
225     '</div>' +
226
227     '%s' %
228     (
229
230     #显示正确还是错误还是未作答
231     '<p style="background-color: %s;border-radius: 10px;padding: 10px;
232     display: inline-block; color: %s">%s</p>' %
233     ('hsl(120, 70%, 60%)' if single[1].split('|')[1] == single[1].split(
234     '|')[2] else 'hsl(350, 70%, 60%)',
235     'black' if single[1].split('|')[1] == single[1].split('|')[2] else
236     'white',
237     '正确' if single[1].split('|')[1] == single[1].split('|')[2] else
238     '错误' if single[1].split('|')[1] != 'no_reply' else '未作答') +
239
240     '<label class="Option" style="background-color: %s;">' % (color[0]) +
241     'A.' + user_question.answer_a +
242     '</label><br>' +
243
244     '<label class="Option" style="background-color: %s;">' % (color[1]) +
245     'B.' + user_question.answer_b +
246     '</label><br>' +
247     if j < (40 if not len(single_history) == 6 else 0) else '' +
248
249     '%s' % (
250
251     '<label class="Option" style="background-color: %s;">' % (color[2]) +
252     'C.' + user_question.answer_c +
253     '</label><br>' +
254
255     '<label class="Option" style="background-color: %s;">' % (color[3]) +
256     'D.' + user_question.answer_d +
257     '</label>' +
258     if user_question.pattern == '1'
259     else ''
260     ) +
261
262     '%s' % (
263
264     '<label class="description" id = "%s" onclick="describe(id)">' % (
265     user_question.id) +
266     '解析: ' +
267     '</label>'

```

```

264         if not (user_question.pattern == '3' or user_question.pattern == '4'
265               or user_question.pattern == '5' or user_question.pattern == '6' or
266               user_question.pattern == '8')
267         else:
268             ) +
269             '<h1>' +
270             '%s' % ('未作答<br>' if (single[1].split('|')[1] == 'no_reply') or (
271             single[1].split('|')[1] == '') else '你的答案是:' + single[1].split(
272             '|')[1] + '<br>' if j < (41 if not len(single_history) == 6 else 0)
273             else '你的答案是:<a href="%s" download="%s">作答文件</a>' % (
274             file_download + '/' + file_dict[j - 41], file_dict[j - 41]) + '<br>'
275             if single[1].split('|')[1] != '文件类型不正确,综合题必须是word文档'
276             and single[1].split('|')[1] !=
277             '文件类型不正确,编程/海龟画图题必须是python文件' else single[1].split
278             ('|')[1] + '<br>') +
279             '正确答案是:' + (single[1].split('|')[2] if (mode == '' and (j < 42
280             or j > 44)) or (mode == 'PAT' and (j < 2 or j > 4)) else '<a
281             href="%s" download="%s">参考答案</a>' % (
282             'http://192.144.180.87:8000/static/it_exam/编程题答案/' + name +
283             '.py', name)) + '<br>' +
284             '</h1>'
285             '</div>'
286         )
287
288     #编写html相关内容,装饰
289     return HttpResponse('''
290         <head>
291             <script
292                 src="https://code.jquery.com/jquery-3.6.0.min.js"></sc
293                 ript>
294             <meta name="viewport" content="width=device-width,
295             initial-scale=1.0">
296             <title>学考练习系统</title>
297             <link rel="stylesheet"
298             href="/static/it_exam/response.css">
299             </head>''' +
300
301             '<body>' +
302
303             '''
304             <div class="Header" style="position: sticky; top: 0">'''
305             +
306             '<h1 class = "welcome">欢迎使用此信息练习系统</h1>' +
307             '<h1 class="TimeCounter" style="font-size: 20px;
308             line-height: 30px;" id="counter">做题时间:' +
309             str(UserHistory.objects.get(user__username = user,
310             date = str(datetime).replace('%20', ' ')).date)[:19] +
311             '<br>' +
312             '正确率:' + UserHistory.objects.get(user__username =
313             user, date = str(datetime).replace('%20', ' ')).
314             accuracy_percent + '&nbsp;' +
315             '正确数:' + str(UserHistory.objects.get(
316             user__username = user, date = str(datetime).replace(
317             '%20', ' ')).correct) + '&nbsp;' +
318             '错误数:' + str(UserHistory.objects.get(
319             user__username = user, date = str(datetime).replace(
320             '%20', ' ')).wrong) + '&nbsp;' +
321             '</h1>' +
322             '<a href="/itexam/%s/">' % ('mainpage_show' if User.
323             objects.get(username = user).is_superuser else
324             'mainpage') +
325             '<button style = "width: 220px; margin-top:
326             -74px; height: 60; font-size: 30" class =
327             "butn">返回主页</button>' +
328             '</a>' +
329             '</div>' +
330
331             '<div style = "margin-top: 150px; margin-left:
332             12%;margin-right: 12%;height: fit-content;">' +
333             all_question +
334             '</div>' +

```

```

303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
'''
<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 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(nextElement);
                    }
                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 i = 0
367 n = 0
368 num = 1
369 exam_paper_ready = [[], ]
370 number_picked = []
371 QuestionChoiced = ''
372
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)

```

```

431
432     #如果错过则多抽两次提高被抽几率
433     if qes.id in question_wrong:
434         random_number = random.randint(1,exam_count)
435
436         if random_number == 1 or random_number == 2:
437             number_picked.append(j)
438             QuestionChocied += str(qes.id) + ','
439
440             if qes.question.find('jpg') != -1:
441                 Question = Question[0:Question.find('<')]
442                 Prepare = [0, str(num) + '.' + Question, qes.answer_a,
443                     qes.answer_b, qes.answer_c, qes.answer_d]
444
445                 if qes.question[-1] == '>':
446                     ImgPath = qes.question[qes.question.find('/'):qes.
447                         question.find('jpg')+3]
448                     Prepare.append(ImgPath)
449                     Prepare.append(1)
450
451                 i += 1
452                 num += 1
453                 exam_paper_ready.append(Prepare)
454
455     else:
456         break
457
458     while i < number_choice:
459
460         #抽取下标
461         number = random.randint(0,len(exam_paper) - 1)
462         qes = exam_paper[number]
463         #防止重选
464         if number in number_picked:
465             continue
466
467         #过滤通用技术
468         elif len(qes.pattern) == 4 and qes.pattern[1] == '1':
469             continue
470
471         number_picked.append(number)
472         QuestionChocied += str(qes.id) + ','
473
474         Question = qes.question
475
476         if qes.question.find('jpg') != -1:
477             Question = Question[0:Question.find('<')]
478             Prepare = [0, str(num) + '.' + Question, qes.answer_a, qes.answer_b,
479                 qes.answer_c, qes.answer_d]
480
481             if qes.question[-1] == '>':
482                 ImgPath = qes.question[qes.question.find('/'):qes.question.find(
483                     'jpg')+3]
484                 Prepare.append(ImgPath)
485                 Prepare.append(1)
486                 exam_paper_ready.append(Prepare)
487
488             i += 1
489             num += 1
490
491     #判断题
492     i = 0
493     if recent_pick == '':
494         exam_paper = Exam.objects.all().filter(pattern = '0')
495
496     #如果已经在做题，抽取正在做的题
497     else:
498         exam_paper = Exam.objects.all().filter(pattern = '0',id__in = recent_pick
499             ).order_by(case_expression)

```





```

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
587         ).order_by(case_expression)
588
589         for j in range(len(exam_paper)):
590             qes = exam_paper[j]
591             exam_paper_ready.append([4, str(num) + '.' + qes.question])
592             num += 1
593
594
595     if recent_pick == '':
596         exam_count = exam_paper.count()
597
598         for j in range(len(exam_paper)):
599             if i < number_brief:
600
601                 qes = exam_paper[j]
602                 #如果没做过则多抽一次提高被抽几率
603                 if qes.id not in question:
604                     if random.randint(1,exam_count) == 1:
605                         number_picked.append(j)
606                         QuestionChoiced += str(qes.id) + ','
607                         exam_paper_ready.append([4, str(num) + '.' + qes.question
608                         ])
609
610                         i += 1
611                         num += 1
612
613                 #如果错过则多抽两次提高被抽几率
614                 if qes.id in question_wrong:
615                     random_number = random.randint(1,exam_count)
616
617                     if random_number == 1 or random_number == 2:
618                         number_picked.append(j)
619                         QuestionChoiced += str(qes.id) + ','
620                         exam_paper_ready.append([4, str(num) + '.' + qes.question
621                         ])
622
623                         i += 1
624                         num += 1
625
626                 else:
627                     break
628
629     while i < number_brief:
630
631         #抽取下标
632         number = random.randint(0,len(exam_paper) - 1)
633         qes = exam_paper[number]
634         #防止重选
635         if number in number_picked:
636             continue
637         #过滤通用技术

```

```

637         elif len(qes.pattern) == 4 and qes.pattern[1] == '1':
638             continue
639         number_picked.append(number)
640         QuestionChoiced += str(qes.id) + ','
641         #加入出题表
642         exam_paper_ready.append([4, str(num) + '.' + qes.question])
643
644         num += 1
645         break
646
647     #操作题
648     i = 0
649     if recent_pick == '':
650         exam_paper = Exam.objects.all().filter(pattern = '3')
651
652     #如果已经在做题，抽取正在做的题
653     else:
654         exam_paper = Exam.objects.all().filter(pattern = '3', id__in = recent_pick
655         ).order_by(case_expression)
656
657         for j in range(len(exam_paper)):
658             qes = exam_paper[j]
659             exam_paper_ready.append([4, str(num) + '.' + qes.question +
660             '只能提交word文档'])
661             num += 1
662
663     if recent_pick == '':
664         exam_count = exam_paper.count()
665
666         for j in range(len(exam_paper)):
667             if i < number_integrate:
668
669                 qes = exam_paper[j]
670                 #如果没做过则多抽一次提高被抽几率
671                 if qes.id not in question:
672                     if random.randint(1, exam_count) == 1:
673                         number_picked.append(j)
674                         QuestionChoiced += str(qes.id) + ','
675                         exam_paper_ready.append([4, str(num) + '.' + qes.question
676                         + '只能提交word文档'])
677
678                         i += 1
679                         num += 1
680
681                 #如果错过则多抽两次提高被抽几率
682                 if qes.id in question_wrong:
683                     random_number = random.randint(1, exam_count)
684
685                     if random_number == 1 or random_number == 2:
686                         number_picked.append(j)
687                         QuestionChoiced += str(qes.id) + ','
688                         exam_paper_ready.append([4, str(num) + '.' + qes.question
689                         + '只能提交word文档'])
690
691                         i += 1
692                         num += 1
693
694             else:
695                 break
696
697     while i < number_integrate:
698         #抽取下标
699         number = random.randint(0, len(exam_paper) - 1)
700         qes = exam_paper[number]
701         #防止重选
702         if number in number_picked:
703             continue
704         #过滤通用技术
705         elif len(qes.pattern) == 4 and qes.pattern[1] == '1':
706             continue

```

```

705         number_picked.append(number)
706         QuestionChoiced += str(qes.id) + ','
707         #加入出题表
708         exam_paper_ready.append([4, str(num) + '.' + qes.question +
        '只能提交word文档'])
709
710         num += 1
711         break
712
713
714     #编程题的测试数据
715     testA = ''
716     testB = ''
717     testC = ''
718
719     #抽编程题a
720     i = 0
721     if recent_pick == '':
722         exam_paper = Exam.objects.all().filter(pattern = '4')
723
724     #如果已经在做题，抽取正在做的题
725     else:
726         exam_paper = Exam.objects.all().filter(pattern = '4',id__in = recent_pick
        ).order_by(case_expression)
727
728         for j in range(len(exam_paper)):
729             qes = exam_paper[j]
730             exam_paper_ready.append([3, str(num) + '.' + qes.question +
        '只能提交python文件'])
731             num += 1
732
733     if recent_pick == '':
734         exam_count = exam_paper.count()
735
736         for j in range(len(exam_paper)):
737             if i < number_program_a:
738
739                 qes = exam_paper[j]
740                 #如果没做过则多抽一次提高被抽几率
741                 if qes.id not in question:
742                     if random.randint(1,exam_count) == 1:
743                         number_picked.append(j)
744                         QuestionChoiced += str(qes.id) + ','
745                         exam_paper_ready.append([3, str(num) + '.' + qes.question
        + '只能提交python文件'])
746
747                         n += 1
748                         i += 1
749                         num += 1
750
751
752                 #如果错过则多抽两次提高被抽几率
753                 if qes.id in question_wrong:
754                     random_number = random.randint(1,exam_count)
755
756                     if random_number == 1 or random_number == 2:
757                         number_picked.append(j)
758                         QuestionChoiced += str(qes.id) + ','
759                         exam_paper_ready.append([3, str(num) + '.' + qes.question
        + '只能提交python文件'])
760
761                         n += 1
762                         i += 1
763                         num += 1
764
765
766         else:
767             break
768
769
770     #抽编程题b
771     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
778         ).order_by(case_expression)
779
780         for j in range(len(exam_paper)):
781             qes = exam_paper[j]
782             exam_paper_ready.append([3, str(num) + '.' + qes.question +
783             '只能提交python文件'])
784             num += 1
785
786     if recent_pick == '':
787         exam_count = exam_paper.count()
788
789         for j in range(len(exam_paper)):
790             if i < number_program_b:
791                 qes = exam_paper[j]
792                 #如果没做过则多抽一次提高被抽几率
793                 if qes.id not in question:
794                     if random.randint(1,exam_count) == 1:
795                         number_picked.append(j)
796                         QuestionChoiced += str(qes.id) + ','
797                         exam_paper_ready.append([3, str(num) + '.' + qes.question
798                         + '只能提交python文件'])
799
800                         n += 1
801                         i += 1
802                         num += 1
803
804                 #如果错过则多抽两次提高被抽几率
805                 if qes.id in question_wrong:
806                     random_number = random.randint(1,exam_count)
807
808                     if random_number == 1 or random_number == 2:
809                         number_picked.append(j)
810                         QuestionChoiced += str(qes.id) + ','
811                         exam_paper_ready.append([3, str(num) + '.' + qes.question
812                         + '只能提交python文件'])
813
814                         n += 1
815                         i += 1
816                         num += 1
817
818             else:
819                 break
820
821     #抽编程题c
822     i = 0
823     if recent_pick == '':
824         exam_paper = Exam.objects.all().filter(pattern = '6')
825
826     #如果已经在做题，抽取正在做的题
827     else:
828         exam_paper = Exam.objects.all().filter(pattern = '6',id__in = recent_pick
829         ).order_by(case_expression)
830
831         for j in range(len(exam_paper)):
832             qes = exam_paper[j]
833             exam_paper_ready.append([3, str(num) + '.' + qes.question +
834             '只能提交python文件'])
835             num += 1
836
837     if recent_pick == '':
838         exam_count = exam_paper.count()
839
840         for j in range(len(exam_paper)):

```

```

838         if i < number_program_c:
839
840             qes = exam_paper[j]
841             #如果没做过则多抽一次提高被抽几率
842             if qes.id not in question:
843                 if random.randint(1,exam_count) == 1:
844                     number_picked.append(j)
845                     QuestionChoiced += str(qes.id) + ','
846                     exam_paper_ready.append([3, str(num) + '.' + qes.question
847                                             + '只能提交python文件'])
848
849                     n += 1
850                     i += 1
851                     num += 1
852
853             #如果错过则多抽两次提高被抽几率
854             if qes.id in question_wrong:
855                 random_number = random.randint(1,exam_count)
856
857                 if random_number == 1 or random_number == 2:
858                     number_picked.append(j)
859                     QuestionChoiced += str(qes.id) + ','
860                     exam_paper_ready.append([3, str(num) + '.' + qes.question
861                                             + '只能提交python文件'])
862
863                     n += 1
864                     i += 1
865                     num += 1
866
867             else:
868                 break
869
870     #编程题
871     while n < number_total_program:
872
873         if 0 <= n and n < number_program_a:
874             exam_paper = Exam.objects.all().filter(pattern = '4')
875
876         if number_program_a <= n and n < number_program_b + number_program_a:
877             exam_paper = Exam.objects.all().filter(pattern = '5')
878
879         if number_program_b + number_program_a <= n and n < number_program_b
880         + number_program_a + number_program_c:
881             exam_paper = Exam.objects.all().filter(pattern = '6')
882
883         #抽取下标
884         number = random.randint(0,len(exam_paper) - 1)
885         qes = exam_paper[number]
886         #防止重选
887         if number in number_picked:
888             continue
889         #过滤通用技术
890         elif len(qes.pattern) == 4 and qes.pattern[1] == '1':
891             continue
892         number_picked.append(number)
893         QuestionChoiced += str(qes.id) + ','
894         #加入出题表
895         exam_paper_ready.append([3, str(num) + '.' + qes.question +
896                                 '只能提交python文件'])
897         testA = qes.question.find('')
898
899         n += 1
900         num += 1
901
902     #海龟画图题
903     i = 0
904     if recent_pick == '':
905         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
910     ).order_by(case_expression)
911
912     for j in range(len(exam_paper)):
913         qes = exam_paper[j]
914         exam_paper_ready.append([5, str(num) + '.' + qes.question +
915         '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)'])
916         num += 1
917
918 if recent_pick == '':
919     exam_count = exam_paper.count()
920
921     for j in range(len(exam_paper)):
922         if i < number_turtle:
923
924             qes = exam_paper[j]
925             #如果没做过则多抽一次提高被抽几率
926             if qes.id not in question:
927                 if random.randint(1,exam_count) == 1:
928                     number_picked.append(j)
929                     QuestionChoiced += str(qes.id) + ','
930                     exam_paper_ready.append([5, str(num) + '.' + qes.question
931                     +
932                     '只能提交python文件(下载"Turtle模块的相关函数"不会导致做题
933                     记录丢失)'])
934
935                     i += 1
936                     num += 1
937
938             #如果错过则多抽两次提高被抽几率
939             if qes.id in question_wrong:
940                 random_number = random.randint(1,exam_count)
941
942                 if random_number == 1 or random_number == 2:
943                     number_picked.append(j)
944                     QuestionChoiced += str(qes.id) + ','
945                     exam_paper_ready.append([5, str(num) + '.' + qes.question
946                     +
947                     '只能提交python文件(下载"Turtle模块的相关函数"不会导致做题
948                     记录丢失)'])
949
950                     i += 1
951                     num += 1
952
953         else:
954             break
955
956 while i < number_turtle:
957     #抽取下标
958     number = random.randint(0,len(exam_paper) - 1)
959     qes = exam_paper[number]
960     #防止重选
961     if number in number_picked:
962         continue
963     #过滤通用技术
964     elif len(qes.pattern) == 4 and qes.pattern[1] == '1':
965         continue
966     number_picked.append(number)
967     QuestionChoiced += str(qes.id) + ','
968     #加入出题表
969     exam_paper_ready.append([5, str(num) + '.' + qes.question +
970     '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)'])
971     break
972
973 user_name = User.objects.get(email = user_email)
974 user_name.userhistory_set.create(history = user_email if mode == 'All'
975 else 'PAT',

```

```

968         date = now(),
969         accuracy_percent = '',
970         correct = 0,
971         wrong = 0,
972         question = QuestionChoiced.removesuffix(
973             ','),
974         filename = 'NoFile')
975
976     return exam_paper_ready
977
978 #读取并设置UserGrade的人名
979 def name():
980     with open('Files_Used_In_Itexam/在校生0365邮箱-230508.txt', encoding='utf-8')
981         as file:
982             i = 0
983             lines = file.readlines()
984             students = {}
985             for line in lines:
986                 if i == 0:
987                     i += 1
988                     continue
989                 line = line.replace('\n', '').split(',')
990                 students[line[-1]] = line[0]
991
992     for i in students:
993         try:
994             student = UserGrade.objects.get(user__email = i)
995         except ObjectDoesNotExist:
996             continue
997         student.name = students[i]
998         student.save()
999
1000
1001 def get_email_by_number(number):
1002     with open('Files_Used_In_Itexam/在校生0365邮箱-230508.txt', encoding='utf-8')
1003         as file:
1004             i = 0
1005             lines = file.readlines()
1006             for line in lines:
1007                 if i == 0:
1008                     i += 1
1009                     continue
1010                 line = line.replace('\n', '').split(',')
1011                 if line[1] == number:
1012                     return line[-1]
1013             return False
1014
1015 def thetest(request):
1016     return render(request, 'it_exam/thetest.html', {'request': "abc"})
1017
1018
1019 #导出用户做题历史
1020 def loadfile(request):
1021
1022     username = function.is_login(request)
1023
1024     if type(username) != type('string'):
1025         return username
1026
1027     if not User.objects.get(username = username).is_superuser:
1028         return HttpResponseRedirect('/itexam/mainpage/')
1029
1030     # 指定数据类型
1031     response = HttpResponse(content_type='application/ms-excel')
1032     # 设置文件名称
1033     response['Content-Disposition'] = 'attachment; filename="result.csv"'
1034     # 创建工作簿
1035     wb = xlwt.Workbook(encoding='utf-8')
1036     # 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         )
1078         Session.objects.get(session_key = request.session.session_key).delete()
1079         #系统数据库清理
1080         session.is_authenticated = 0
1081         session.is_pick = 0
1082         session.save()
1083         logout(request) #系统登出
1084     except (KeyError):
1085         pass
1086
1087     return HttpResponseRedirect('/itexam/')
1088
1089 #javascript 获取数据库数据
1090 def get_data(request,user):
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 #管理员演示
1111 def mainpage_show(request):
1112
1113     username = function.is_login(request)
1114
1115     if type(username) != type('string'):
1116         return username
1117
1118     mail = User.objects.get(username = username).email
1119     UserHistory.objects.all().filter(Q(user__username = username) & (Q(history = mail
1120 ) | 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 #获取本班级用户
1134 def get_user(request,name):
1135
1136     username = function.is_login(request)
1137
1138     if type(username) != type('string'):
1139         return username
1140
1141     if request.method != 'POST':
1142         return HttpResponseRedirect(reverse('mainpage'))
1143
1144     all_user = []
1145     try:
1146         group = Group.objects.get(name = name)
1147     except ObjectDoesNotExist:
1148         return HttpResponseRedirect(reverse('managegroup'))
1149     users = group.user_set.all()
1150     for i in users:
1151         all_user.append([i.username,UserGrade.objects.get(user__username = i.username
1152 ).name])
1153
1154     return JsonResponse(all_user, safe=False)
1155
1156 #管理用户分组
1157 def manage_group(request):
1158
1159     username = function.is_login(request)
1160
1161     if type(username) != type('string'):
1162         return username
1163
1164
1165     if User.objects.get(username = username).is_superuser:
1166
1167         msg = ''
1168         all_name = []
1169
1170         def create_form(query):
1171             has = False
1172             multiple = False
1173             #逗号分隔搜法
1174             if query != 'all':
1175                 if ',' in query:

```

```

1177         multiple = True
1178         query = query.split(',')
1179         try:
1180             query.remove('')
1181         except ValueError: #list.remove(x): x not in list
1182             pass
1183         for i in range(len(query)):
1184             #去除空格
1185             query[i] = query[i].strip()
1186
1187         #去除空格
1188         if not multiple:
1189             query = query.strip()
1190         #按照关键字查找
1191         all_user = User.objects.all().filter(is_superuser = 0)
1192         for i in all_user:
1193             user_name = UserGrade.objects.get(user__username = i.username).
1194             name
1195             for student in query:
1196                 if (student if multiple else query) in user_name:
1197                     name = user_name
1198                     all_name.append((i.username,name))
1199                     has = True
1200                     if not multiple:
1201                         break
1202
1203             if has:
1204                 form = GroupManage(all_name)
1205                 edit = ''
1206                 groups = Group.objects.all()
1207
1208                 for i in groups:
1209                     edit += (''
1210                             <div style = "border-top: 0.01em solid grey;
1211                             padding: 0.3em;">
1212                             <label for = "%s">选择学生从班级内删除</label>
1213                             <a id = %s style = "position: relative; top: 1em;"
1214                             onclick = "manage(id)">%s</a>
1215                             <br>
1216                             <div id = "%s" style = "margin-top: 2em;">
1217                             <label for = "%s"><input id = "%s" name =
1218                             "delete_name" type="checkbox">
1219                             value="%s">删除班级</label><br>
1220                             </div></div>
1221                             <br>''
1222                             % (i.name,i.name,i.name,i.name + '-div',i.name,i.name
1223                             ,i.name))
1224
1225                 return form,edit,''
1226
1227         msg = mark_safe("<h3>没有找到对应的学生</h3>")
1228         form,edit = create_form('all')
1229         return form,edit,msg
1230
1231     else:
1232         all_user = User.objects.all().filter(is_superuser = 0)
1233
1234         for i in all_user:
1235             try:
1236                 name = UserGrade.objects.get(user__username = i.username,
1237                 user__is_superuser = 0).name
1238             except ObjectDoesNotExist:
1239                 continue
1240
1241             all_name.append((i.username,name))
1242
1243         form = GroupManage(all_name)
1244         edit = ''
1245         groups = Group.objects.all()
1246
1247         for i in groups:

```

```

1242         edit += (''
1243                 <div style = "border-top: 0.01em solid grey; padding:
1244                 0.3em;">
1245                 <label for = "%s">选择学生从班级内删除</label>
1246                 <a id = %s style = "position: relative; top: 1em;" onclick
1247                 = "manage(id)">%s</a>
1248                 <br>
1249                 <div id = "%s" style = "margin-top: 2em;">
1250                 <label for = "%s"><input id = "%s" name = "delete_name"
1251                 type="checkbox" value="%s">删除班级</label><br>
1252                 </div></div>
1253                 <br>''
1254                 % (i.name,i.name,i.name,i.name + '-div',i.name,i.name,i.name
1255                 ))
1256
1257     return form,edit
1258
1259 if request.method != 'POST':
1260     form,edit = create_form('all')
1261     return render(request, 'it_exam/manage_group.html', {'form': form, 'edit'
1262     : mark_safe(edit), 'htmlmsg': msg})
1263
1264 form = dict(request.POST)
1265 try:
1266     query = form['q'][0]
1267 except KeyError:
1268     try:
1269         name_of_group = form['name'][0]
1270
1271     except KeyError:
1272         try:
1273             for name_of_group in form['delete_name']:
1274                 Group.objects.get(name = name_of_group).delete()
1275             return HttpResponseRedirect(reverse('managegroup'))
1276         except KeyError:
1277             try:
1278                 for i in form['group']:
1279                     i = i.split('|')
1280                     user = User.objects.get(username = i[1])
1281                     group = Group.objects.get(name = i[0])
1282                     group.user_set.remove(user)
1283             except KeyError:
1284                 pass
1285             return HttpResponseRedirect(reverse('managegroup'))
1286
1287 if name_of_group.isalpha() or name_of_group.isdecimal():
1288     pass
1289 else:
1290     form,edit = create_form('all')
1291     form.errors['name'] = ['班级名称只能有数字、字母和中文']
1292     return render(request, 'it_exam/manage_group.html', {'form': form,
1293     'edit': mark_safe(edit), 'htmlmsg': msg})
1294
1295 try:
1296     test = form['group']
1297 except KeyError:
1298     form,edit = create_form('all')
1299     form.errors['group'] = ['没有选择学生']
1300     return render(request, 'it_exam/manage_group.html', {'form': form,
1301     'edit': mark_safe(edit), 'htmlmsg': msg})
1302
1303 group = Group(name = name_of_group + ':' + username)
1304
1305 try:
1306     group.save()
1307 except IntegrityError:
1308     pass
1309
1310 for i in form['group']:
1311     user = User.objects.get(username = i)

```

```

1307         group = Group.objects.get(name = name_of_group + ':' + username)
1308         group.user_set.add(user)
1309
1310         return HttpResponseRedirect(reverse('managegroup'))
1311
1312
1313     form,edit,msg = create_form(query)
1314     return render(request, 'it_exam/manage_group.html', {'form': form, 'edit':
mark_safe(edit), 'htmlmsg': msg})
1315
1316     return HttpResponseRedirect(reverse('mainpage'))
1317
1318
1319 #用户主页设计
1320 def mainpage(request):
1321
1322     username = function.is_login(request)
1323
1324     if type(username) != type('string'):
1325         return username
1326
1327
1328     mail = User.objects.get(username = username).email
1329     UserHistory.objects.all().filter(Q(user__username = username) & (Q(history = mail
) | Q(history = 'CAJ') | Q(history = 'PAT'))).delete()
1330     CountUserStopTime.objects.all().filter(username = username).delete()
1331     session = UserSession.objects.get(user__username = username)
1332     session.is_pick = 0
1333     session.save()
1334
1335 #管理员
1336 if User.objects.get(username = username).is_superuser:
1337
1338     if request.method != 'POST':
1339
1340         #获取用户搜索的值,若没有查询值则返回空字符串
1341         q_value = request.GET.get('q','')
1342         class_value = request.GET.get('class','')
1343
1344         user = ''
1345         user_list = []
1346         searched = False
1347         has = False
1348
1349         #没有提供搜索参数的情况
1350         if q_value == '' and class_value == '':
1351             msg = mark_safe("<h3>没有提供搜索参数</h3>")
1352             return render(request, 'it_exam/show_result.html', {'username':
username, 'user': '', 'htmlmsg':msg, 'script': ''})
1353
1354         #所有学生和班级的情况
1355         elif q_value == 'all':
1356             all_user = User.objects.all().filter(is_superuser = 0)
1357
1358         #指定学生或班级的情况
1359         else:
1360             searched = True
1361
1362             #判断是否输入班级
1363             if class_value != '':
1364                 try:
1365                     group = Group.objects.get(name = class_value)
1366                 except ObjectDoesNotExist:
1367                     msg = mark_safe("<h3>没有找到对应的班级</h3>")
1368                     return render(request, 'it_exam/show_result.html', {
'username':username, 'user': '', 'htmlmsg':msg, 'script': ''
})
1369
1370
1371         #只写了班级没写学生
1372         if q_value == '':
1373

```

```

1374         all_user = User.objects.all().filter(Q(is_superuser = 0) & Q(
1375             groups__name = class_value))
1376         has = True
1377         searched = False
1378
1379     else:
1380         #如果有班级，在班级内查询
1381         if class_value != '':
1382             all_user = User.objects.all().filter(Q(is_superuser = 0) & Q(
1383                 groups__name = class_value))
1384             #如果没有，在全部学生中查询
1385         else:
1386             all_user = User.objects.all().filter(is_superuser = 0)
1387
1388         for i in all_user:
1389             if q_value in UserGrade.objects.get(user__username = i.
1390                 username).name:
1391                 usergrade = UserGrade.objects.get(user__username = i.
1392                     username)
1393                 user_list.append(usergrade.name + ':' +
1394                     '正确率' + ':' + usergrade.accuracy_percent +
1395                     '正确题数' + ':' + str(usergrade.correct) +
1396                     '错误题数' + ':' + str(usergrade.wrong)
1397                 )
1398                 has = True
1399
1400         if not has:
1401             msg = mark_safe("<h3>没有找到对应的学生</h3>")
1402             return render(request, 'it_exam/show_result.html', {'username':
1403                 username, 'user': '', 'htmlmsg':msg, 'script': ''})
1404
1405     #读取每个<button>显示的内容
1406     if not searched:
1407         for i in all_user:
1408
1409             usergrade = UserGrade.objects.get(user__username = i.username)
1410
1411             user_list.append(usergrade.name + ':' +
1412                 '正确率' + ':' + usergrade.accuracy_percent +
1413                 '正确题数' + ':' + str(usergrade.correct) +
1414                 '错误题数' + ':' + str(usergrade.wrong) +
1415                 '覆盖率' + ':' + str(usergrade.cover_rate)
1416             )
1417
1418     #动态生成HTML
1419     id = 0
1420     for i in range(len(user_list)):
1421         user += '<p><button id="' + str(id) + '" onclick=insert(' + str(id) +
1422             ', "' + all_user[i].username + '")>' + user_list[i] + '</button></p>'
1423         id += 1
1424
1425     return render(request, 'it_exam/show_result.html', {'username': username,
1426         'htmlmsg': '', 'user': mark_safe(user), 'script': mark_safe(''<script>
1427
1428         function getCookie(name) {
1429
1430             let cookieValue = null;
1431
1432             if (document.cookie && document.cookie !== '') {
1433
1434                 const cookies = document.cookie.split(';');
1435
1436                 for (let i = 0; i < cookies.length; i++) {
1437
1438                     const cookie = cookies[i].trim();
1439
1440                     // Does this cookie string begin with the
1441                     name we want?

```

```

1431                                     if (cookie.substring(0, name.length + 1)
=== (name + '=')) {
1432                                     cookieValue =
decodeURIComponent(cookie.substring(name.length + 1));
1433                                     break;
1434                                     }
1435                                     }
1436                                     }
1437                                     return cookieValue;
1438                                     }
1439
1440     function insert(id,username){
1441         const target = document.getElementById(id);
1442
1443         if (target.classList.contains("clicked")) {
1444             var nextElement = target.nextSibling;
1445             nextElement.parentNode.removeChild(nextElement);
1446             target.classList.remove('clicked');
1447         }
1448         else {
1449             var body = document.querySelector('body');
1450             body.style.setProperty("pointer-events", "none");
1451             target.setAttribute("class", "clicked");
1452             let csrftoken = getCookie('csrftoken');
1453             $(document).ready(function() {
1454                 $.ajax({
1455                     url: '/itexam/getdata/' + username + '/',
1456                     type: 'POST',
1457                     headers: {'X-CSRFToken': csrftoken},
1458                     mode: 'same-origin',
1459                     success: function(response) {
1460                         let user = '';
1461                         for(let i = 0; i <
response.length; i++){
1462                             user += '<option value = "' +
response[i]['history'] + ', ' + response[i]['date'].replace('T','%20') + ', ' +
response[i]['user_id'] + '><div>做题时间:' + response[i]['date'].replace('T','
').slice(0,19) +
1463

```

```

1464         ' 正确率:' + response[i]['accuracy_percent'] +
1465
1466         ' 正确题数:' + response[i]['correct'] +
1467
1468         ' 错误题数:' + response[i]['wrong'] +
1469
1470         '</option>';
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483     else:
1484
1485         #为用户显示此次做题的情况
1486         form = request.POST
1487
1488         #数据类型: '253:wrong|no_reply|A, 294:wrong|no_reply|B,.....
1489         21:wrong|no_reply|C, 526:wrong|no_reply|D'
1490
1491         single_history = form['history'].split(',')
1492         datatime = single_history[-2]
1493         user = User.objects.get(id = single_history[-1]).username
1494         single_history.pop()
1495         single_history.pop()
1496         return function.show_history(single_history,user,datatime)
1497
1498
1499         #学生
1500     else:
1501         return function.mainpage_show(request,username)
1502
1503
1504     #用户错题集
1505     def wrongproblem(request):
1506
1507         #判断是否登录

```



```

1508 username = function.is_login(request)
1509
1510 if type(username) != type('string'):
1511     return username
1512
1513 all_question = ''
1514 all_choice_question = ''
1515 all_fill_question = ''
1516 all_judge_question = ''
1517 all_do_question = ''
1518 all_program_question = ''
1519 all_turtle_question = ''
1520
1521 if UserGrade.objects.get(user__username = username).wrong_problems.split(',') == ['']:
1522     return HttpResponse(''
1523         <head>
1524             <meta name="viewport" content="width=device-width, initial-scale=1.0">
1525             <title>学考练习系统</title>
1526             <link rel="stylesheet" href="/static/it_exam/response.css">
1527         </head>' +
1528
1529         '<body>' +
1530
1531         '''
1532         <div class="Header">''' +
1533             '<h1 style="background-color: transparent; text-align: center; font-size: 40px; line-height: 100px;">错题集' +
1534             '</h1>' +
1535             '</div>' +
1536
1537             '<div style = "margin-top: 150px; margin-left: 12%;margin-right: 12%;height: fit-content;">' +
1538                 all_question +
1539             '</div>' +
1540
1541             '</body>'
1542     )
1543
1544 for i in UserGrade.objects.get(user__username = username).wrong_problems.split(','):
1545     try:
1546         user_question = Exam.objects.get(id = i)
1547     except ObjectDoesNotExist:
1548         continue
1549     index = ['A', 'B', 'C', 'D']
1550     color = ['transparent', 'transparent', 'transparent', 'transparent']
1551     if user_question.pattern in ['1', '0']:
1552         if user_question.answer == 'A':
1553             color[0] = 'palegreen'
1554
1555         elif user_question.answer == 'B':
1556             color[1] = 'palegreen'
1557
1558         elif user_question.answer == 'C':
1559             color[2] = 'palegreen'
1560
1561         elif user_question.answer == 'D':
1562             color[3] = 'palegreen'
1563
1564     if user_question.pattern == '1':
1565         all_choice_question += ('<div class="Question">' +
1566             user_question.question +
1567             '</div><br>' +
1568
1569             '<label class="Option" style="background-color: %s;">' % color[0] +
1570             'A.' + user_question.answer_a +
1571             '</label><br>' +
1572
1573             '<label class="Option" style="background-color: %s;">' % color[1] +

```

```

1574         'B.' + user_question.answer_b +
1575         '</label><br>' +
1576
1577         '<label class="Option" style="background-color:
1578         %s;">' % color[2] +
1579         'C.' + user_question.answer_c +
1580         '</label><br>' +
1581
1582         '<label class="Option" style="background-color:
1583         %s;">' % color[3] +
1584         'D.' + user_question.answer_d +
1585         '</label><br><br>')
1586
1587 if user_question.pattern == '0':
1588     all_judge_question += ('<div class="Question">' +
1589     user_question.question +
1590     '</div><br>' +
1591
1592     '<label class="Option" style="background-color:
1593     %s;">' % color[0] +
1594     'A.' + user_question.answer_a +
1595     '</label><br>' +
1596
1597     '<label class="Option" style="background-color:
1598     %s;">' % color[1] +
1599     'B.' + user_question.answer_b +
1600     '</label><br><br>')
1601
1602 if user_question.pattern == '9':
1603     all_fill_question += ('<div class="Question">' +
1604     user_question.question +
1605     '</div><br><br>')
1606
1607 if user_question.pattern == '3':
1608     all_do_question += ('<div class="Question">' +
1609     user_question.question +
1610     '</div><br><br>')
1611
1612 if user_question.pattern == '4' or user_question.pattern == '5' or
1613 user_question.pattern == '6':
1614     name = user_question.question[user_question.question.find('>',20) + 1:
1615     user_question.question.find('<',20)]
1616     all_program_question += ('<div class="Question">' +
1617     user_question.question + '<br>' +
1618     '正确答案是:' + ('<a href="%s"
1619     download="%s">参考答案</a>' % (
1620     'http://192.144.180.87:8000/static/it_exam/编程题
1621     答案/' + name + '.py',name)) + '<br>'
1622     '</div><br><br>')
1623
1624 if user_question.pattern == '8':
1625     all_turtle_question += ('<div class="Question">' +
1626     user_question.question +
1627     '</div><br><br>')
1628
1629 all_question = all_choice_question + all_judge_question + all_fill_question +
1630 all_do_question + all_program_question + all_turtle_question
1631
1632 return HttpResponse(''
1633     <head>
1634     <meta name="viewport" content="width=device-width, initial-scale=1.0">
1635     <title>学考练习系统</title>
1636     <link rel="stylesheet" href="/static/it_exam/response.css">
1637 </head>' +
1638
1639     '<body>' +
1640
1641     ''
1642     <div class="Header">' +
1643     '<h1 style="background-color: transparent; text-align: center;
1644     font-size: 40px; line-height: 100px;" id="counter">错题集' +
1645     '</h1>' +

```

```

1635         '</div>' +
1636
1637         '<div style = "margin-top: 150px; margin-left: 12%;margin-right:
1638         12%;height: fit-content;">' +
1639         all_question +
1640         '</div>' +
1641
1642         '</body>'
1643     )
1644
1645     #账号登陆
1646     def log_in(request):
1647         user_name = ''
1648         user_password = ''
1649
1650         #接收用户输入的账号与密码
1651         if request.method != 'POST':
1652             form = Login()
1653             return render(request, 'it_exam/login.html', {'form': form})
1654
1655         form = Login(request.POST)
1656         if form.is_valid():
1657             user_name = form.cleaned_data['username'].replace(' ', '')
1658             user_password = form.cleaned_data['password']
1659
1660             #验证用户
1661             user = authenticate(request, username = user_name, password = user_password)
1662
1663             #验证通过,
1664             if user is not None:
1665
1666                 #未登录状态
1667                 try:
1668                     test = request.session['username']
1669                 except KeyError:
1670                     login(request, user)
1671                     request.session['username'] = user_name
1672                     session = UserSession.objects.get(user__username = user_name)
1673                     session.is_authenticated = 1
1674                     session.session_key = request.session.session_key
1675                     session.save()
1676                     return HttpResponseRedirect(reverse('mainpage'))
1677
1678                 #如果本设备有登陆的账号,拒绝登陆
1679                 login(request, user)
1680                 request.session['username'] = user_name
1681                 return HttpResponse('''
1682                 <!DOCTYPE html>
1683                 <html>
1684                     <meta name="viewport" content="width=device-width, initial-scale=1.0">
1685                     <meta charset="UTF-8">
1686                     <head>
1687                         <link rel="stylesheet" href="/static/it_exam/transition.css">
1688                         <title>学考练习系统</title>
1689                     </head>
1690                     <body>
1691                         <h1>
1692                         该设备已有账号登陆或有账号未登出!
1693                         </h1>
1694                         <a href = "/itexam/">重新登陆</a>
1695                         <p>无法登出? <a href = "/itexam/logout/">强制登出</a></p>
1696                     </body>
1697                 </html>''')
1698
1699             else:
1700                 form.errors['password'] = ['账号或密码错误']
1701                 return render(request, 'it_exam/login.html', {'form': form})
1702         else:
1703             form.errors['password'] = ['账号或密码错误']
1704             return render(request, 'it_exam/login.html', {'form': form})
1705

```

```

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 #账号注册
1718 def log_on(request):
1719     if request.method != 'POST':
1720         form = Logon()
1721         return render(request, 'it_exam/logon.html', {'form': form, 'email':
1722             '@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
1742                 , 'error':''})
1743
1744             form = Logon()
1745             return render(request, 'it_exam/logon.html', {'form': form, 'email':
1746                 '@i.pkuschool.edu.cn', 'error':mark_safe('<ul
1747                 class="errorlist"><li>找不到此学号,请确认输入的是正确的学号</li></ul>')})
1748
1749
1750     form = Logon(request.POST)
1751
1752     if form.is_valid():
1753         if not form.cleaned_data['email'].endswith('@i.pkuschool.edu.cn') :
1754             form.errors['email'] = ['请输入地址为"@i.pkuschool.edu.cn"的邮箱']
1755             return render(request, 'it_exam/logon.html', {'form': form, 'email':
1756                 '@i.pkuschool.edu.cn', 'error':''})
1757
1758         user_name = form.cleaned_data['username']
1759         user_password = form.cleaned_data['password']
1760         mail = form.cleaned_data['email']
1761
1762         for i in user_password:
1763             if i.isalpha() or i.isdecimal():
1764                 pass
1765             else:
1766                 form.errors['password'] = ['密码只能有数字和字母']
1767                 return render(request, 'it_exam/logon.html', {'form': form, 'email':
1768                     '@i.pkuschool.edu.cn', 'error':''})
1769
1770         for i in user_name:
1771             if i.isalpha() or i.isdecimal() or '_' in i:
1772                 pass
1773             else:
1774                 form.errors['username'] = ['用户名只能有数字、字母、下划线和中文']
1775                 return render(request, 'it_exam/logon.html', {'form': form, 'email':

```

```

        '@i.pkuschool.edu.cn', 'error': ''}))
1772
1773     if mail in all_mail:
1774         form.errors['email'] = ['该邮箱已注册']
1775         return render(request, 'it_exam/logon.html', {'form': form, 'email':
        '@i.pkuschool.edu.cn', 'error': ''}))
1776
1777     if user_name in all_username:
1778         form.errors['username'] = ['该用户名已被使用']
1779         return render(request, 'it_exam/logon.html', {'form': form, 'email':
        '@i.pkuschool.edu.cn', 'error': ''}))
1780
1781     is_superuser = 0
1782     #如果是maohuajun@i.pkuschool.edu.cn,设置为管理员
1783     if mail == 'maohuajun@i.pkuschool.edu.cn':
1784         is_superuser = 1
1785
1786     #发送邮件
1787
1788     #生成确认码
1789     ensurekey = ''
1790     logon_time = now()
1791
1792     for i in range(50):
1793         ensurekey += str(random.randint(0,9))
1794
1795     try:
1796         send_mail(
1797             subject = '练习网站注册',
1798             message = '',
1799             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
        )
1809     except (BadHeaderError, ValueError, SMTPRecipientsRefused):
1810         return HttpResponse('发送失败,请重新操作!')
1811
1812     Assurance.objects.create(key = ensurekey, date = logon_time,
1813                             email = mail, username = user_name,
1814                             password = make_password(user_password),
1815                             task = 'logon', is_superuser = is_superuser)
1816
1817     return HttpResponse(''
1818     <!DOCTYPE html>
1819     <html>
1820         <meta name="viewport" content="width=device-width, initial-scale=1.0">
1821         <meta charset="UTF-8">
1822         <head>
1823             <link rel="stylesheet" href="/static/it_exam/transition.css">
1824             <title>学考练习系统</title>
1825         </head>
1826         <body>
1827             <h1>
1828                 成功发送至%s,请到此邮箱打开激活网址!
1829             </h1>
1830             <a href = "/itexam/">登陆</a>
1831             <a href = "https://outlook.office.com/">点击进入outlook邮箱</a>
1832         </body>
1833     </html>'' % mail)
1834
1835 else:
1836     form.errors['email'] = ['请输入有效的邮箱']
1837     return render(request, 'it_exam/logon.html', {'form': form, 'email':

```

```

1838     '@i.pkuschool.edu.cn', 'error': ''})
1839
1840
1841 #注册或找回密码确认
1842 def logon_ensure_or_enter_new_password(request, number, time, task):
1843     if task == 'logon':
1844         try:
1845             user = Assurance.objects.get(Q(key = number) & Q(date = str(time).replace
1846                 ('%20', ' ')) & Q(task = 'logon'))
1847         except ObjectDoesNotExist:
1848             return HttpResponse('<link rel="stylesheet"
1849                 href="/static/it_exam/transition.css"><h1>确认网址不正确<h1>')
1850             User.objects.create_user(username = user.username, password = 'creating',
1851                 email = user.email, is_superuser = user.is_superuser, is_staff = user.
1852                 is_superuser)
1853             logon_user = User.objects.get(username = user.username)
1854             logon_user.password = user.password
1855             logon_user.save()
1856
1857             #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
1858             user_account = User.objects.get(email = user.email)
1859             user_account.usergrade_set.create(total_exam = 0, correct = 0, wrong = 0,
1860                 accuracy_percent = '0%', cover_rate = '0%')
1861             function.name()
1862             user_account.usersession_set.create(is_authenticated = 0, is_pick = 0,
1863                 session_key = '')
1864
1865             Assurance.objects.get(key = number).delete()
1866             return HttpResponse('<link rel="stylesheet"
1867                 href="/static/it_exam/transition.css"><h1>注册成功</h1><a href =
1868                 "/itexam/">登陆</a>')
1869
1870 elif task == 'forgetpassword':
1871     try:
1872         user = Assurance.objects.get(Q(key = number) & Q(date = str(time).replace
1873             ('%20', ' ')) & Q(task = 'forgetpassword') & Q(password = 'finding'))
1874     except ObjectDoesNotExist:
1875         return HttpResponse('<link rel="stylesheet"
1876             href="/static/it_exam/transition.css"><h1>确认网址不正确<h1>')
1877
1878     #确保表单网址正确
1879     url = user.key + '/' + str(user.date) + '/' + user.task
1880     if request.method != 'POST':
1881         form = EnterNewPassword()
1882         username = User.objects.get(email = user.email).username
1883         return render(request, 'it_exam/enternewpassword.html', {'form': form,
1884             'url': url, 'username': username})
1885     form = EnterNewPassword(request.POST)
1886
1887     #重设密码
1888     if form.is_valid():
1889         try:
1890             user = User.objects.get(email = user.email)
1891         except ObjectDoesNotExist:
1892             return HttpResponse('<link rel="stylesheet"
1893                 href="/static/it_exam/transition.css"><h1>用户不存在<h1><br><a
1894                 href="/itexam/">登陆</a>')
1895             user.set_password(form.cleaned_data['newpassword'])
1896             user.save()
1897             Assurance.objects.get(key = number).delete()
1898             return HttpResponse(''
1899                 <!DOCTYPE html>
1900                 <html>
1901                     <meta name="viewport" content="width=device-width, initial-scale=1.0">
1902                     <meta charset="UTF-8">
1903                     <head>
1904                         <link rel="stylesheet" href="/static/it_exam/transition.css">
1905                         <title>学考练习系统</title>
1906                     </head>
1907                     <body>
1908                         <h1>

```

```

1896                 更改成功!
1897                 </h1>
1898                 <a href = "/itexam/">登陆</a>
1899             </body>
1900         </html>''')
1901
1902
1903 #找回密码
1904 def forget_password(request):
1905     all_mail = []
1906     all_user = User.objects.all()
1907     for i in all_user:
1908         all_mail.append(i.email)
1909
1910     if request.method != 'POST':
1911         form = ForgetPassword()
1912         return render(request, 'it_exam/forgetpassword.html', {'form': form, 'email':
            '@i.pkuschool.edu.cn', 'error':''})
1913
1914
1915 form = request.POST
1916 if 'number' in form.keys():
1917
1918     #如果选择自动填充, 查找并填充
1919     email = function.get_email_by_number(form['number'])
1920     if email:
1921         form = ForgetPassword()
1922         return render(request, 'it_exam/forgetpassword.html', {'form': form,
            'email':email, 'error':''})
1923
1924     form = ForgetPassword()
1925     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>')})
1926
1927
1928 form = ForgetPassword(request.POST)
1929
1930 if form.is_valid():
1931     mail = form.cleaned_data['email']
1932
1933     if mail not in all_mail:
1934         return HttpResponse('''
1935             <!DOCTYPE html>
1936             <html>
1937                 <meta name="viewport" content="width=device-width, initial-scale=1.0">
1938                 <meta charset="UTF-8">
1939                 <head>
1940                     <link rel="stylesheet" href="/static/it_exam/transition.css">
1941                     <title>学考练习系统</title>
1942                 </head>
1943                 <body>
1944                     <h1>
1945                         %s, 该邮箱没有账号!
1946                     </h1>
1947                     <a href = "/itexam/">登陆</a>
1948                 </body>
1949             </html>''' % mail)
1950
1951 #生成确认码
1952 ensurekey = ''
1953 forgetpassword_time = now()
1954
1955 for i in range(50):
1956     ensurekey += str(random.randint(0,9))
1957
1958 try:
1959     send_mail(
1960         subject = '练习网站密码找回',
1961         message = '',
1962         html_message = '<h3>请点击<a
            href="%s">此链接</a>以继续账号或密码找回</h3>'

```

```

1963
1964         % ('http://192.144.180.87:8000' + reverse('ensure',args = [ensurekey,str(
forgetpassword_time),'forgetpassword'])),
1965
1966         from_email = 'webmaster163mail <%s>' % (settings.EMAIL_HOST_USER),
1967         recipient_list = [mail],
1968         fail_silently=False
1969     )
1970 except (BadHeaderError,ValueError,SMTPRecipientsRefused) as reason:
1971     return HttpResponse('发送失败,原因为%s请重新操作!' % reason)
1972
1973 Assurance.objects.create(key = ensurekey, date = forgetpassword_time,
1974                           email = mail, username = 'finding',
1975                           password = 'finding', task = 'forgetpassword',
is_superuser = User.objects.get(email = mail).
is_superuser)
1976
1977     return HttpResponse(''
1978 <!DOCTYPE html>
1979 <html>
1980     <meta name="viewport" content="width=device-width, initial-scale=1.0">
1981     <meta charset="UTF-8">
1982     <head>
1983         <link rel="stylesheet" href="/static/it_exam/transition.css">
1984         <title>学考练习系统</title>
1985     </head>
1986     <body>
1987         <h1>
1988             成功将找回链接发送至%s!
1989         </h1>
1990         <a href = "/itexam/">登陆</a>
1991         <a href = "https://outlook.office.com/">点击进入outlook邮箱</a>
1992     </body>
1993 </html>''' % mail)
1994
1995 else:
1996     form.errors['email'] = ['请输入有效的邮箱']
1997     return render(request, 'it_exam/forgetpassword.html', {'form': form, 'email':
'i.pkuschool.edu.cn', 'error':''})
1998
1999
2000 #抽取题目,共46题
2001 def pick(request,mode):
2002
2003     #判断是否登录
2004     username = function.is_login(request)
2005
2006     if type(username) != type('string'):
2007         return username
2008
2009     user_email = User.objects.get(username = username).email
2010     user = username
2011
2012     session = UserSession.objects.get(user__email = user_email)
2013
2014     if mode != 'ChoiceAndJudge' and mode != 'All' and mode != 'ProgramAndTackle':
2015         return HttpResponseRedirect(reverse('mainpage'))
2016
2017
2018     if User.objects.get(email = user_email).is_superuser:
2019         show = 'mainpage_show'
2020
2021     else:
2022         show = 'mainpage'
2023
2024
2025     if session.is_pick == 0:
2026
2027         if mode == 'ChoiceAndJudge':
2028             number_of_question = 40
2029
2030         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

```

```

2097         </h1>
2098         <a href = "/itexam/mainpage/">主页</a>
2099     </body>
2100 </html>''' )
2101
2102
2103 #判断题目正误
2104 def answer(request):
2105
2106     #判断是否登陆
2107     username = function.is_login(request)
2108
2109     if type(username) != type('string'):
2110         return username
2111
2112     user = username
2113
2114     session = UserSession.objects.get(user__username = user)
2115     session.is_pick = 0
2116     session.save()
2117
2118     try:
2119         CountUserStopTime.objects.get(username = user).delete()
2120     except ObjectDoesNotExist:
2121         pass
2122
2123     #读取答案
2124     if request.method == 'POST':
2125
2126         #读取题目
2127         try:
2128             history_for_question = UserHistory.objects.get(Q(user__username = user) &
2129                 Q(accuracy_percent = ''))
2130             question = history_for_question.question
2131         except ObjectDoesNotExist:
2132             return HttpResponseRedirect(reverse('have_submit_or_mainpage'))
2133
2134         #读取用户基本信息
2135         user_ans = []
2136         datetime = now()
2137         wrong = 0
2138         correct = 0
2139         have_file = False
2140         name_of_file = ''
2141
2142         ans = request.POST
2143
2144         index = 1
2145         while index < 47:
2146
2147             if history_for_question.history == 'PAT':
2148                 mode = 'PAT'
2149                 #只处理文件
2150                 if index > 6:
2151                     break
2152
2153                 try:
2154                     file = request.FILES['p' + str(index)]
2155
2156                     if not file.name.endswith('.docx') and index == 2:
2157                         user_ans.append('文件类型不正确$N$★$N$●$N$◇综合题必须是word文档')
2158                     elif not file.name.endswith('.py') and index != 2:
2159                         user_ans.append(
2160                             '文件类型不正确$N$★$N$●$N$◇编程/海龟画图题必须是python文件')
2161                     else:
2162                         have_file = True
2163                         user_ans.append(file.name.replace('|', '').replace(':',
2164                             '〇●$N$◇$N$◇(株)($N$★$N$●$N$◇)'))
2165                         information_of_file = str(history_for_question.date)[0:19].
2166                             replace('%20', '').replace(':', '.') + history_for_question.
2167                             user.username
2168                         os.makedirs('it_exam/storage/' + information_of_file,

```

```

2164         exist_ok = True)
2165         #创建压缩文件
2166         with ZipFile(file = 'it_exam/storage/' + information_of_file
2167                     + '/' + file.name + '.zip', mode = 'w') as f:
2168             f.writestr(file.name,file.read(),compresslevel = 9)
2169             name_of_file += (str(index - 42) + '|' + file.name + '.zip').
2170             replace('#','😊😊😊😊😊😊😊😊') + ','
2171
2172     except KeyError:
2173         user_ans.append('no_reply')
2174
2175     index += 1
2176     continue
2177
2178 try:
2179     #处理文件
2180     if index >= 42:
2181         try:
2182             file = request.FILES['p' + str(index)]
2183
2184             if not file.name.endswith('.docx') and index == 42:
2185                 user_ans.append(
2186                     '文件类型不正确$N$★$○$◎$◇综合题必须是word文档')
2187             elif not file.name.endswith('.py') and index != 42:
2188                 user_ans.append(
2189                     '文件类型不正确$N$★$○$◎$◇编程/海龟画图题必须是python文件')
2190             else:
2191                 have_file = True
2192                 user_ans.append(file.name.replace('|','').replace(':',
2193                     '😊😊😊😊😊😊😊😊($N$★$○$◎$◇)').replace(',',', '$N$★$○$◎$◇'))
2194                 information_of_file = str(history_for_question.date)[0:19
2195                     ].replace('%20','').replace(':','.') +
2196                 history_for_question.user.username
2197                 os.makedirs('it_exam/storage/' + information_of_file,
2198                     exist_ok = True)
2199
2200                 #创建压缩文件
2201                 with ZipFile(file = 'it_exam/storage/' +
2202                             information_of_file + '/' + file.name + '.zip', mode =
2203                             'w') as f:
2204                     f.writestr(file.name,file.read(),compresslevel = 9)
2205                     name_of_file += (str(index - 42) + '|' + file.name +
2206                     '.zip').replace('#','😊😊😊😊😊😊😊😊') + ','
2207
2208         except KeyError:
2209             user_ans.append('no_reply')
2210
2211     index += 1
2212     continue
2213
2214 user_answer = ans['p' + str(index)]
2215
2216 #四十一题特殊情况
2217 if user_answer == '' and index == 41:
2218     user_ans.append('no_reply')
2219     index += 1
2220     continue
2221
2222 #保存用户答案
2223 user_ans.append(user_answer.replace('|','').replace(':',
2224     '😊😊😊😊😊😊😊😊($N$★$○$◎$◇)').replace(',',', '$N$★$○$◎$◇'))
2225
2226 except KeyError:
2227
2228     #若没有作答，则保存no_reply
2229     user_ans.append('no_reply')
2230     index += 1
2231     continue
2232
2233 index += 1
2234
2235 #有文件则存储其地址
2236 if have_file:

```

```

2223         history_for_question.filename = 'http://192.144.180.87:8000/storage/' +
2224         information_of_file + '#' + name_of_file
2225         history_for_question.save()
2226
2227     #个人账户数据修改
2228     account = UserGrade.objects.get(user__username = user)
2229     aggregations = set(account.wrong_problems.split(','))
2230     aggregations_question = set(account.question.split(','))
2231
2232     #去除split所带来的影响（使空字符串长度为1）
2233     if len(aggregations) == 1:
2234         aggregations.pop()
2235
2236     if len(aggregations_question) == 1:
2237         aggregations_question.pop()
2238
2239     #判断正误
2240     question_list = question.split(',')
2241     for i in range(46):
2242
2243         #判断是否为编程题
2244         #是
2245         '''if i == 42 or i == 43 or i == 44:
2246
2247             #创建并给虚拟环境的全局变量赋值
2248             safe_globals['user_answer'] = '未收到用户赋值'
2249
2250             try:
2251                 compiled_code = compile_restricted(user_ans[i], '<string>',
2252                 'exec')
2253                 exec(compiled_code,safe_globals)
2254             except Exception as reason:
2255                 print(reason)
2256                 # 处理错误，例如不允许的操作或访问限制
2257                 return HttpResponse('<h1>输入错误</h1>')
2258             user_answer = safe_globals['user_answer']
2259
2260             if correct_answer.answer == str(user_answer):
2261
2262                 #本次做题数据修改
2263                 question.correct += 1
2264                 question_list[i] = question_list[i] + ':correct|' +
2265                 str(user_answer) + '|' + correct_answer.answer
2266
2267                 #个人账户数据修改
2268                 account.correct += 1
2269
2270                 #题目库数据修改
2271                 correct_answer.correct += 1
2272                 correct_answer.correct_possibility = str((correct_answer.correct
2273                 * 100) / (correct_answer.correct + correct_answer.wrong)) + '%'
2274                 correct_answer.save()
2275
2276             else:
2277
2278                 #本次做题数据修改
2279                 question.wrong += 1
2280                 question_list[i] = question_list[i] + ':wrong|' +
2281                 str(user_answer) + '|' + correct_answer.answer
2282
2283                 #个人账户数据修改
2284                 account.wrong += 1
2285
2286                 #以集合的形式防止重复录入错题
2287                 aggregations.add(question.history.split(',')[i])
2288
2289                 #题目库数据修改
2290                 correct_answer.wrong += 1
2291                 correct_answer.correct_possibility = str((correct_answer.correct

```

```

2290         * 100) / (correct_answer.correct + correct_answer.wrong)) + '%'
2291         correct_answer.save()'''
2292
2293     #否
2294     #else:
2295     try:
2296         correct_answer = Exam.objects.get(id = question_list[i])
2297     except (IndexError, ObjectDoesNotExist):
2298         continue
2299     correct_answer.total += 1
2300     account.total_exam += 1
2301
2302     #录入做过的题
2303     aggregations_question.add(question.split(',')[i])
2304
2305     #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
2306     if correct_answer.answer == user_ans[i]:
2307
2308         #本次做题数据修改
2309         correct += 1
2310         question_list[i] = question_list[i] + ':correct|' + user_ans[i] + '|' +
2311             + correct_answer.answer
2312
2313         #个人账户数据修改
2314         if question_list[i].split(':')[0] in account.wrong_problems.split(','):
2315             aggregations_discard(question_list[i].split(':')[0])
2316             account.correct += 1
2317
2318         #题目库数据修改
2319         correct_answer.correct += 1
2320         correct_answer.correct_possibility = str(round((correct_answer.correct
2321             * 100) / (correct_answer.correct + correct_answer.wrong), 2))
2322         + '%'
2323         correct_answer.save()
2324
2325     else:
2326
2327         #本次做题数据修改
2328         wrong += 1
2329         question_list[i] = question_list[i] + ':wrong|' + user_ans[i] + '|' +
2330             + correct_answer.answer
2331         #个人账户数据修改
2332         account.wrong += 1
2333
2334         #以集合的形式防止重复录入错题
2335         aggregations.add(question.split(',')[i])
2336
2337         #题目库数据修改
2338         correct_answer.wrong += 1
2339         correct_answer.correct_possibility = str(round((correct_answer.correct
2340             * 100) / (correct_answer.correct + correct_answer.wrong), 2))
2341         + '%'
2342         correct_answer.save()
2343
2344     #个人账户数据修改
2345
2346     #检测账户做题模式
2347     userhistory = UserHistory.objects.get(Q(user__username = user) & Q(
2348         accuracy_percent = ''))
2349     index = Q(pattern = '0') | Q(pattern = '1') | Q(pattern = '3') | Q(pattern =
2350         '4') | Q(pattern = '5') | Q(pattern = '6') | Q(pattern = '8') | Q(pattern = '9')
2351
2352     account.accuracy_percent = str(round((account.correct * 100) / account.
2353         total_exam, 2)) + '%'
2354     account.wrong_problems = str(aggregations).replace('{', ',').replace('}', ',').
2355         replace(' ', ',').replace(' ', ',')
2356     account.question = str(aggregations_question).replace('{', ',').replace('}', ',')
2357         .replace(' ', ',').replace(' ', ',')
2358     account.cover_rate = str(round((len(account.question.split(',')) * 100) /

```

```

2349 Exam.objects.filter(index).count(),2)) + '%'
2350 account.save()
2351
2352 #本次做题数据修改
2353 userhistory.history = str(question_list).replace('[','').replace(']', '').
2354 replace("'",'')
2355 userhistory.date = datetime
2356 userhistory.accuracy_percent = str(round((correct * 100) / (wrong + correct),
2357 2)) + '%'
2358 userhistory.correct = correct
2359 userhistory.wrong = wrong
2360 userhistory.save()
2361
2362 #获取每个题的做题记录
2363 single_history = UserHistory.objects.get(user__username = user, date = str(
2364 datetime).replace('%20',' ')).history.split(',')
2365
2366 return function.show_history(single_history,user,datetime)
2367
2368 else:
2369     return HttpResponseRedirect('/itexam/mainpage/')
2370
2371 #设置时间
2372 def set_time(request):
2373
2374     #判断是否登陆
2375     username = function.is_login(request)
2376
2377     if type(username) != type('string'):
2378         return username
2379
2380     user = username
2381
2382     if request.method != 'POST':
2383         return HttpResponseRedirect(reverse('mainpage'))
2384
2385     user_time = CountUserStopTime.objects.get(username = user)
2386     if user_time.is_stop:
2387         user_time.all_stop_time = (now() - user_time.stop_time).total_seconds() +
2388         user_time.all_stop_time
2389     else:
2390         user_time.stop_time = now()
2391         user_time.is_stop = (False if user_time.is_stop else True)
2392         user_time.save()
2393
2394     return JsonResponse([1], safe=False)
2395
2396 #获取解析
2397 def get_description(request, id):
2398
2399     #判断是否登陆
2400     username = function.is_login(request)
2401
2402     if type(username) != type('string'):
2403         return username
2404
2405     if request.method != 'POST':
2406         return HttpResponseRedirect(reverse('mainpage'))
2407
2408     description = Exam.objects.get(id = id).description
2409
2410     return JsonResponse([description], safe = False)
2411
2412 #删除没有做完的做题历史
2413 def delete_history(request):
2414
2415     #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
2428 ) | Q(history = 'CAJ') | Q(history = 'PAT'))).delete()
2429 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)
```