

电子科技大学

课 设 报 告

一、实验项目名称：

自动组卷评卷考试系统

二、实验内容

用 Python 语言编程实现自动组卷评卷考试系统，软件主要功能包括：从题库中随机抽取试题自动组成试卷（满分 100 分）；实现考生考试答题操作界面；实现自动阅卷评分功能；等等。

三、实验要求

1. 题型包括单项选择题、填空题、判断题等等。
2. 题库可以采用文本文件、CSV 文件或数据库等来实现。
3. 要求在源程序中标注必要的注释。
4. 要求对程序的使用和运行方法进行必要说明。
5. 课程设计要提交程序源代码及附属的测试题库文档等（便于阅卷测试）。

四、实验器材（设备、元器件）

处理器：Intel® Core™ i5-8300H CPU @ 2.30GHz 2.30GHz

已安装的内存(RAM)：8GB

系统类型：64 位操作系统，基于 x64 的处理器

IDE：JetBrains PyCharm (Community Version) 2020.2.1

Python 解释器：Python 3.8

五、实验步骤

5.1 系统组成

系统总体上由前端、后端、防作弊演示程序、配置程序、文档结构树图生成程序和数据等部分组成。

5.2 框架设计

系统的总体框架与结构如图 1 所示。

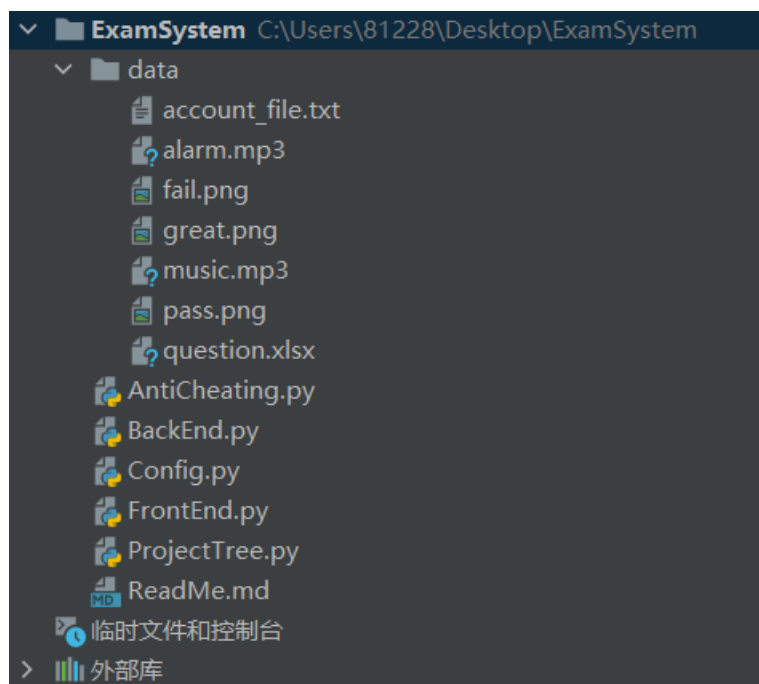


图 1 系统总体框架与结构示意图

下面分别介绍各个部分的作用及功能：

- FrontEnd.py，主要包括的是前端类，主要实现了注册和答题两个界面和数据调用等功能。
- BackEnd.py，主要包括的是后端类，主要实现了检验用户是否存在、账号密码是否正确、添加新用户、产生随机题目序号、获取题目并返回给前端等功能。
- AntiCheating.py，主要包括的是防作弊演示程序，主要实现了强行关闭主流文本编辑器和网页浏览器、强制清空系统剪切板等功能。
- ProjectTree.py，主要包括的是防作弊演示，主要实现了产生项目结构树图功能。
- Config.py，主要实现了获取当前路径，找寻用户信息表和题库等功能。
- data 文件夹里有题库 question.xlsx，考试结束自动播放的音乐 music.mp3，存储管理员账号和测试账号的文本 account_file.txt，反映考试成绩的图片 fail.png、pass.png、great.png。

5.3 具体实现

FrontEnd.py

```
1. # -*- coding: utf-8 -*-
2. # @Time : 2021/1/20 19:55
3. # @Author : UestcXiye
4. # @File : FrontEnd.py
5. # @Software: PyCharm
6.
7. import tkinter as tk
8. from tkinter import messagebox
9. from tkinter import scrolledtext
10. from threading import Timer
11. from playsound import playsound
12. from BackEnd import BackEnd, checkAccount, addUser
13. from Config import *
14. from pil import Image, ImageTk
15.
16. dataList = BackEnd() # 存储得到的考题
17.
18. for i in range(5):
19.     print(dataList.SingleList[i]['参考答案'])
20. im = [] # 读取文件
21. img = [] # 转换格式后
22.
23. for i in range(5): # 初始化读取的图片列表
24.     im.append(None)
25.     img.append(None)
26.
27.
28. class FrontEnd:
29.     """ 前端类，完成注册和答题两个界面和数据调用 """
30.
31.     def __init__(self):
32.         self.state = STATE_INIT # 有限状态机，完成题目衔接和变化
33.         self.count = 0 # 计数到第几道题了
34.         self.minute = 60
35.         self.second = 0 # 设定考试时间 60min
36.         self.ans = [] # 存放考生的结果，确认后判断
37.         self.score = 0 # 分数
38.         self.loginWindow = tk.Tk()
39.         self.initialLoginWindow(self.loginWindow)
40.
```

```

41.     def initialLoginWindow(self, loginWindow):
42.         """for login"""
43.         loginWindow['bg'] = 'skyblue' # background color
44.         loginWindow.title('考试系统登陆界面')
45.         loginWindow.resizable(width=True, height=True)
46.
47.         width = loginWindow.winfo_screenwidth()
48.         height = loginWindow.winfo_screenheight()
49.         loginWindow.geometry(
50.             "400x200+%d+%d" %
51.             (width / 2 - 200, height / 2 - 200))
52.
53.         self.varAccount = tk.StringVar()
54.         self.varAccount.set('')
55.         self.varKey = tk.StringVar()
56.         self.varKey.set('')
57.
58.         # 创建标签
59.         self.labelAccount = tk.Label(
60.             loginWindow,
61.             text='用户名:',
62.             justify=tk.RIGHT,
63.             width=80)
64.         self.labelKey = tk.Label(
65.             loginWindow,
66.             text='密 码:',
67.             justify=tk.RIGHT,
68.             width=80)
69.         self.labelRegister = tk.Label(
70.             loginWindow, text='注 册', justify=tk.RIGHT, width=80)
71.
72.         # 将标签放到窗口上
73.         self.labelAccount.place(x=20, y=10, width=160, height=40)
74.         self.labelKey.place(x=20, y=60, width=160, height=40)
75.
76.         # 创建账号文本框, 同时设置关联的变量
77.         self.account = tk.Entry(
78.             loginWindow,
79.             width=80,
80.             textvariable=self.varAccount)
81.         self.account.place(x=200, y=10, width=160, height=40)
82.         # 创建密码文本框
83.         self.key = tk.Entry(
84.             loginWindow,

```

```

85.         show='*',
86.         width=80,
87.         textvariable=self.varKey)
88.     self.key.place(x=200, y=60, width=160, height=40)
89.
90.     # 创建按钮组件, 同时设置按钮事件处理函数
91.     buttonOk = tk.Button(loginWindow, text='登录', command=self.login)
92.     buttonOk.place(x=20, y=140, width=100, height=40)
93.     buttonCancel = tk.Button(
94.         loginWindow,
95.         text='取消',
96.         command=self.cancelLogin)
97.     buttonCancel.place(x=140, y=140, width=100, height=40)
98.     buttonRegister = tk.Button(loginWindow, text='注册', command=self.regist)
99.     buttonRegister.place(x=260, y=140, width=100, height=40)
100.
101.     # make Esc exit the program
102.     loginWindow.bind('<Escape>', lambda e: loginWindow.destroy())
103.     # 启动消息循环
104.     loginWindow.mainloop()
105.
106.     def login(self):
107.         """ 获取用户名和密码 """
108.         name = self.account.get()
109.         passwd = self.key.get()
110.
111.         nameList, passwordList = checkAccount('account_file.txt')
112.         # for test
113.         for i in range(len(nameList)):
114.             if name == nameList[i]:
115.                 if passwd == passwordList[i]:
116.                     tk.messagebox.showinfo(title='提示', message='登录成功! ')
117.                     self.loginWindow.destroy()
118.                     self.mainWindow = tk.Tk()
119.                     self.initialMainWindow(self.mainWindow)
120.                     return
121.         tk.messagebox.showerror('Python tk', message='账号或密码错误! ')
122.
123.     def cancelLogin(self):
124.         """ 清空用户输入的用户名和密码 """
125.         self.varAccount.set('')
126.         self.varKey.set('')
127.
128.     def regist(self):

```

```

129.         name = self.account.get()
130.         passwd = self.key.get()
131.         userNameList, userPasswordList = checkAccount('account_file.txt')
132.         if not userNameList or not userPasswordList:
133.             addUser('account_file.txt', name, passwd)
134.             return
135.         for userName in userNameList:
136.             if name == userName:
137.                 tk.messagebox.showerror('Python tk', message='已有该用户名! ')
138.                 registerSuccessful = addUser('account_file.txt', name, passwd)
139.                 if registerSuccessful:
140.                     messagebox.showinfo('提示信息', message='注册成功! ')
141.
142.     def initialMainWindow(self, mainWindow):
143.         """ initialize window and the window settings"""
144.         self.width = mainWindow.winfo_screenwidth()
145.         self.height = mainWindow.winfo_screenheight()
146.
147.         print('[Function: initialMainWindow]')
148.         mainWindow.geometry("%dx%d" % (self.width, self.height))
149.         mainWindow['bg'] = 'skyblue' # background color
150.         mainWindow.title('考试系统考试界面')
151.         mainWindow.resizable(width=True, height=True)
152.
153.         mainWindow.protocol('WM_DELETE_WINDOW', self.closeMainWindow)
154.         self.setMenu(mainWindow)
155.         # make Esc exit the program
156.         mainWindow.bind('<Escape>', lambda e: mainWindow.destroy())
157.
158.         self.totalCount = dataList.Single.totalNum + \
159.                             dataList.Multi.totalNum + dataList.Judge.totalNum
160.
161.         self.showInitFsm()
162.         self.watchDog()
163.         mainWindow.mainloop()
164.
165.     def showInitFsm(self):
166.         nextState = STATE_SINGLE
167.         print('[Function: Init_fsm] startup')
168.
169.         self.varScore = tk.StringVar() # 已获得分数
170.         self.varScore.set(str(self.score) + '\100')
171.         self.showScoreName = tk.Label(self.mainWindow,
172.                                         text='已获得分数: ',

```

```

173.                                width=150, # 设置 label 的宽度: 30
174.                                height=50, # 设置 label 的高度: 10
175.                                justify='left', # 设置文本对齐方式: 左对齐
176.                                anchor='nw', # 设置文本在 label 的方位: 西北方
    位
177.                                font=('微软雅黑', 18), # 设置字体: 微软雅黑,
    字号: 18
178.                                fg='white', # 设置前景色: 白色
179.                                bg='grey', # 设置背景色: 灰色
180.                                )
181.        self.showScoreName.place(x=10, y=10, width=150, height=50)
182.        self.showScore = tk.Label(self.mainWindow, textvariable=self.varScore)
183.        self.showScore.place(x=10, y=70, width=150, height=50)
184.        self.varTimeLeft = tk.StringVar()
185.        # self.varTimeLeft.set(str(min) + '分' + str(sec) + '秒')
186.        self.timeLeft = tk.Label(self.mainWindow, textvariable=self.varTimeLeft)
187.        self.timeLeft.place(x=self.width - 200, y=70, width=150, height=50)
188.
189.        # 剩余时间见函数 watchDog
190.        # self.watchDog(10, 00) # 考试时间 10min
191.
192.        self.showTimeLeft = tk.Label(self.mainWindow, text='剩余时间', # 设置文本内
    容
193.                                width=150, # 设置 label 的宽度: 30
194.                                height=50, # 设置 label 的高度: 10
195.                                justify='left', # 设置文本对齐方式: 左对齐
196.                                anchor='ne', # 设置文本在 label 的方位: 西北方
    位
197.                                font=('微软雅黑', 18), # 设置字体: 微软雅黑, 字
    号: 18
198.                                fg='white', # 设置前景色: 白色
199.                                bg='grey', # 设置背景色: 灰色
200.                                padx=20, # 设置 x 方向内边距: 20
201.                                pady=10) # 设置 y 方向内边距: 10
202.        self.showTimeLeft.place(x=self.width - 200, y=10, width=150, height=60)
203.
204.        self.varButtonA = tk.StringVar()
205.        self.varButtonA.set(
206.            'A. ' + str(dataList.SingleList[self.count % 10]['A']))
207.        self.varButtonB = tk.StringVar()
208.        self.varButtonB.set(
209.            'B. ' + str(dataList.SingleList[self.count % 10]['B']))
210.        self.varButtonC = tk.StringVar()
211.        self.varButtonC.set(

```

```

212.         'C. ' + str(dataList.SingleList[self.count % 10]['C']))
213.     self.varButtonD = tk.StringVar()
214.     self.varButtonD.set(
215.         'D. ' + str(dataList.SingleList[self.count % 10]['D']))
216.     self.varButtonE = tk.StringVar()
217.     self.varButtonE.set('')
218.
219.     self.buttonA = tk.Button(self.mainWindow,
220.                             textvariable=self.varButtonA,
221.                             command=self.buttonAFsm)
222.     self.buttonB = tk.Button(self.mainWindow,
223.                             textvariable=self.varButtonB,
224.                             command=self.buttonBFsm)
225.     self.buttonC = tk.Button(self.mainWindow,
226.                             textvariable=self.varButtonC,
227.                             command=self.buttonCFsm)
228.     self.buttonD = tk.Button(self.mainWindow,
229.                             textvariable=self.varButtonD,
230.                             command=self.buttonDFsm)
231.     self.buttonOK = tk.Button(self.mainWindow,
232.                              text='确认',
233.                              command=self.buttonOKFsm) # 确认按钮, 确认不再更改
答案
234.     self.buttonA.place(x=100, y=400, width=750, height=50)
235.     self.buttonB.place(x=100, y=500, width=750, height=50)
236.     self.buttonC.place(x=100, y=600, width=750, height=50)
237.     self.buttonD.place(x=100, y=700, width=750, height=50)
238.     self.buttonOK.place(x=1000, y=400, width=300, height=50)
239.
240.     self.varChoice = tk.StringVar()
241.     self.varChoice.set(list2str(self.ans)) # 显示考生选择的选项
242.     self.showChoice = tk.Label(
243.         self.mainWindow, textvariable=self.varChoice)
244.     self.showChoice.place(x=1000, y=600, width=150, height=50)
245.     self.subject = scrolledtext.ScrolledText(
246.         self.mainWindow, relief="solid")
247.     self.subject.place(x=self.width / 3, y=10)
248.     self.subject.insert('end', str(self.count + 1) + '. ' +
249.                        dataList.SingleList[self.count]['题目内容'] + '\n')
250.
251.     self.count = 0
252.     print('[Function: Init_fsm] complicated')
253.     self.state = nextState
254.

```



```
255.     def buttonAFsm(self):
256.         print('      [Event: buttonA clicked]')
257.         if self.state == STATE_SINGLE: # 单选
258.             self.ans = []
259.             self.ans.append('A')
260.         elif self.state == STATE_MULTIPLE: # 多选
261.             if 'A' not in self.ans:
262.                 self.ans.append('A')
263.                 self.ans = sorted(self.ans)
264.             else:
265.                 self.ans.remove('A')
266.         else: # 判断题
267.             self.ans = []
268.             self.ans.append('对')
269.             self.varChoice.set(list2str(self.ans))
270.
271.     def buttonBFsm(self):
272.         print('      [Event: buttonB clicked]')
273.         if self.state == STATE_SINGLE: # 单选
274.             self.ans = []
275.             self.ans.append('B')
276.         elif self.state == STATE_MULTIPLE: # 多选
277.             if 'B' not in self.ans:
278.                 self.ans.append('B')
279.                 self.ans = sorted(self.ans)
280.             else:
281.                 self.ans.remove('B')
282.                 self.ans = sorted(self.ans)
283.         else:
284.             self.ans = []
285.             self.ans.append('对')
286.             self.varChoice.set(list2str(self.ans))
287.
288.     def buttonCFsm(self):
289.         print('      [Event: buttonC clicked]')
290.         if self.state == STATE_SINGLE: # 单选
291.             self.ans = []
292.             self.ans.append('C')
293.         elif self.state == STATE_MULTIPLE: # 多选
294.             if 'C' not in self.ans:
295.                 self.ans.append('C')
296.                 self.ans = sorted(self.ans)
297.             else:
298.                 self.ans.remove('C')
```

```

299.         sorted(self.ans)
300.     else: # 判断
301.         self.ans = []
302.         self.ans.append('错')
303.         self.varChoice.set(list2str(self.ans))
304.
305.     def buttonDFsm(self):
306.         print('      [Event: buttonD clicked]')
307.         if self.state == STATE_SINGLE: # 单选
308.             self.ans = []
309.             self.ans.append('D')
310.         elif self.state == STATE_MULTI: # 多选
311.             if 'D' not in self.ans:
312.                 self.ans.append('D')
313.                 self.ans = sorted(self.ans)
314.             else:
315.                 self.ans.remove('D')
316.                 self.ans = sorted(self.ans)
317.         else: # 判断
318.             self.ans = []
319.             self.ans.append('错')
320.             self.varChoice.set(list2str(self.ans))
321.
322.     def buttonEFsm(self):
323.         print('      [Event: buttonE clicked]')
324.         if self.state == STATE_SINGLE: # 单选
325.             self.ans = []
326.             self.ans.append('E')
327.         elif self.state == STATE_MULTI: # 多选
328.             if 'E' not in self.ans:
329.                 self.ans.append('E')
330.                 self.ans = sorted(self.ans)
331.             else:
332.                 self.ans.remove('E')
333.                 self.ans = sorted(self.ans)
334.         else: # 判断
335.             self.ans = []
336.             self.ans.append('错')
337.             self.varChoice.set(list2str(self.ans))
338.
339.     def buttonOKFsm(self):
340.         """ 确认按钮, 点击后进入下一状态 """
341.         print('      [Event: buttonOK clicked]')
342.

```

```

343.         self.score += self.checkAns()
344.         self.varScore.set(str(self.score) + '/100') # 显示得分
345.
346.         self.count = self.count + 1 # 下一题
347.         self.varChoice.set('') # 清空显示的考生选项，准备下一题
348.
349.         self.ans = [] # 清空内部存储的考生选项，准备下一题
350.         if self.state == STATE_SINGLE:
351.             self.showSingleFsm()
352.         elif self.state == STATE_MULTI:
353.             self.showMultiFsm()
354.         elif self.state == STATE_JUDGE:
355.             self.showJudgeFsm()
356.         else: # 结束，分数不再变动
357.             self.showDoneFsm()
358.
359.     def checkAns(self) -> int:
360.         """ 检查结果，返回本题得分 """
361.         if self.state == STATE_SINGLE:
362.             print('      [Debug: your choice:] ' + str(self.ans))
363.             if list2str(
364.                 self.ans) == dataList.SingleList[self.count % 10]['参考答案
365.             ']:
366.                 # self.score = self.score + 3 # 本题得分
367.                 return 3
368.             else:
369.                 return 0
370.         elif self.state == STATE_MULTI:
371.             print('      [Debug: your choice:] ' + str(self.ans))
372.             if list2str(
373.                 self.ans) == dataList.MultiList[self.count % 10]['参考答案']:
374.                 # self.score += 5 # 本题得分
375.                 return 5
376.             else:
377.                 return 0
378.         else:
379.             print('      [Debug: your choice:] ' + str(self.ans))
380.             if list2str(
381.                 self.ans) == dataList.JudgeList[self.count % 10]['参考答案']:
382.                 # self.score += 2 # 本题得分
383.                 return 2
384.             else:
385.                 return 0

```

```

386.     def updateSubject(self, listName):
387.         self.subject.delete(0, tk.END)
388.         self.subject.insert('end', str(self.count + 1) + '. ' +
389.                             listName[self.count % 10]['题目内容'] + '\n')
390.         self.varButtonA.set(
391.             'A. ' + str(listName[self.count % 10]['A']))
392.         self.varButtonB.set(
393.             'B. ' + str(listName[self.count % 10]['B']))
394.         self.varButtonC.set(
395.             'C. ' + str(listName[self.count % 10]['C']))
396.         self.varButtonD.set(
397.             'D. ' + str(listName[self.count % 10]['D']))
398.         if self.state == STATE_MULTI:
399.             self.varButtonE.set(
400.                 'E. ' + str(listName[self.count % 10]['E']))
401.
402.     def showSingleFsm(self):
403.         if self.count < self.totalCount / 3 - 1:
404.             nextState = STATE_SINGLE
405.         else:
406.             nextState = STATE_MULTI
407.             self.buttonE = tk.Button(self.mainWindow,
408.                                     textvariable=self.varButtonE,
409.                                     command=self.buttonEFsm)
410.             self.buttonA.place(x=100, y=400, width=750, height=50)
411.             self.buttonB.place(x=100, y=480, width=750, height=50)
412.             self.buttonC.place(x=100, y=560, width=750, height=50)
413.             self.buttonD.place(x=100, y=640, width=750, height=50)
414.             self.buttonE.place(x=100, y=720, width=750, height=50)
415.
416.             self.updateSubject(dataList.SingleList)
417.
418.             self.state = nextState
419.
420.     def showMultiFsm(self):
421.         if self.totalCount / 3 <= self.count < 2 * self.totalCount / 3:
422.             nextState = STATE_MULTI
423.         else:
424.             nextState = STATE_JUDGE
425.             self.buttonA.destroy()
426.             self.buttonB.destroy()
427.             self.buttonC.destroy()
428.             self.buttonD.destroy()
429.             self.buttonE.destroy()

```

```

430.         self.buttonTrue = tk.Button(self.mainWindow,
431.                                     text='对',
432.                                     command=self.buttonAFsm)
433.         self.buttonFalse = tk.Button(self.mainWindow,
434.                                     text='错',
435.                                     command=self.buttonEFsm)
436.         self.buttonTrue.place(x=100, y=400, width=750, height=50)
437.         self.buttonFalse.place(x=100, y=600, width=750, height=50)
438.
439.         self.updateSubject(dataList.MultiList) # 刷新题目和选项
440.
441.         self.state = nextState
442.
443.     def showJudgeFsm(self):
444.         print('total count: ', self.totalCount)
445.         if self.count < self.totalCount:
446.             nextState = STATE_JUDGE
447.         else:
448.             nextState = STATE_DONE
449.
450.         self.subject.delete(0.0, tk.END) # 清空上一题
451.         self.subject.insert('end', str(self.count + 1) + '. ' +
452.                             dataList.JudgeList[self.count % 10]['题目内容
453. ']) + '\n')
454.         self.state = nextState
455.
456.     def showDoneFsm(self):
457.         """ 结束状态 """
458.
459.         # 清除所有无用控件
460.         self.buttonTrue.destroy()
461.         self.buttonFalse.destroy()
462.         self.buttonOK.destroy()
463.         self.showChoice.destroy()
464.         self.subject.destroy()
465.
466.         # 播放音乐
467.         playsound(getCurrentPath() + DataPath + 'music.mp3', block=False)
468.
469.         # 计时结束, 清零
470.         self.timeCount.cancel()
471.         # self.varTimeLft.set('0:00')
472.         self.showScoreName = tk.Label(self.mainWindow,

```

```

473.                                     text='最终得分: ',
474.                                     width=150, # 设置 label 的宽度: 30
475.                                     height=50, # 设置 label 的高度: 10
476.                                     justify='left', # 设置文本对齐方式: 左对齐
477.                                     anchor='nw', # 设置文本在 label 的方位: 西北方
        位
478.                                     font=('微软雅黑', 18), # 设置字体: 微软雅黑,
        字号: 18
479.                                     fg='white', # 设置前景色: 白色
480.                                     bg='grey', # 设置背景色: 灰色
481.                                     )
482.     self.showScoreName.place(x=10, y=10, width=150, height=50)
483.     # 加载图像
484.     global im
485.     global img
486.
487.     if self.score < 60:
488.         im[0] = Image.open(getCurrentPath() + DataPath + "fail.png")
489.         img[0] = ImageTk.PhotoImage(im[0])
490.         imLabel = tk.Label(self.mainWindow, image=img[0]).pack()
491.     elif 60 <= self.score <= 85:
492.         im[1] = Image.open(getCurrentPath() + DataPath + "pass.png")
493.         img[1] = ImageTk.PhotoImage(im[1])
494.         imLabel = tk.Label(self.mainWindow, image=img[1]).pack()
495.     else:
496.         im[2] = Image.open(getCurrentPath() + DataPath + "great.png")
497.         img[2] = ImageTk.PhotoImage(im[2])
498.         imLabel = tk.Label(self.mainWindow, image=img[2]).pack()
499.
500.     def setMenu(self, window):
501.         """create a menu bar with Exit command and version info"""
502.         menubar = tk.Menu(window)
503.         filemenu = tk.Menu(menubar, tearoff=0)
504.         filemenu.add_command(label="Exit", command=window.destroy)
505.         infoMenu = tk.Menu(menubar, tearoff=0)
506.         infoMenu.add_command(label="Version Info", command=self.menuInfo)
507.         menubar.add_cascade(label="File", menu=filemenu)
508.         menubar.add_cascade(label="Info", menu=infoMenu)
509.         window.config(menu=menubar)
510.
511.     def menuInfo(self):
512.         messagebox.showinfo(
513.             'info',
514.             'Created By UestcXiye \n version 1.0')

```

```

515.
516.     def watchDog(self):
517.         """ 定时程序，考试时间最多一小时，结束终止答题，显示分数，播放音乐 """
518.         timeLeft = 60 * self.minute + self.second
519.         timeLeft -= 1
520.         self.second = self.second - 1
521.         if self.second < 0:
522.             self.minute = self.minute - 1
523.             self.second = 59
524.         if self.minute < 0 or timeLeft == 0:
525.             self.state = STATE_DONE
526.             playsound(
527.                 getCurrentPath() + DataPath + 'music.mp3',
528.                 block=False
529.             ) # 倒计时结束，播放提示音乐。
530.             self.showDoneFsm()
531.             self.varTimeLeft.set(str(self.minute) + ':' + str(self.second))
532.             self.timeCount = Timer(1, self.watchDog, ())
533.             self.timeCount.start() # 计时器启动
534.
535.     def closeMainWindow(self):
536.         """ to check if you really wanna exit """
537.         ans = messagebox.askyesno(title='Quit', message='要关闭窗口吗？您所做的修改不会保存')
538.         if ans:
539.             self.mainWindow.destroy()
540.         else:
541.             pass
542.
543.
544. if __name__ == '__main__':
545.     test = FrontEnd()

```

BackEnd.py

```

1. # -*- coding: utf-8 -*-
2. # @Time : 2021/1/20 20:12
3. # @Author : UestcXiye
4. # @File : BackEnd.py
5. # @Software: PyCharm
6.
7. import pandas as pd
8. import random
9. from Config import *

```

```

10.
11.
12. def checkAccount(filename) -> tuple:
13.     """ 检验用户是否存在, 账号密码是否正确 """
14.     path = getCurrentPath() + DataPath + filename
15.     fid = open(path, 'r+')
16.     accountList = []
17.     userNameList, userPasswordList = [], []
18.     line = fid.readlines()
19.     for child in line:
20.         # print('[Function checkAccount]: ' + child)
21.         # if not (line.startswith("@")): # 注释行开头为@
22.         accountList.append(child.strip("\n").split('\t'))
23.         # print(accountList)
24.     for name, password in accountList:
25.         userNameList.append(name)
26.         userPasswordList.append(password)
27.         # print(userNameList)
28.         # print(userPasswordList)
29.     fid.close()
30.     return userNameList, userPasswordList
31.
32.
33. def addUser(filename, userName: str, userPassword: str) -> int:
34.     """ 添加新用户, 在用户名不重读的情况下才会调用 """
35.     path = getCurrentPath() + DataPath + filename
36.     txtfile = open(path, 'a')
37.     data = '\n' + userName + '\t' + userPassword
38.     txtfile.write(data)
39.     txtfile.close()
40.     return 1
41.
42.
43. class SingleChoiceSubject:
44.
45.     def __init__(self):
46.         self.scorePer = 3 # 每道题的分值
47.         self.totalNum = 10 # 总共 10 道单选
48.         self.subjectList = {} # 存放所有题目信息
49.         self.path = getCurrentPath() + DataPath + 'question.xlsx'
50.         self.df = pd.read_excel(self.path, sheet_name='单选')
51.         self.tempList = [] # 存储一行信息
52.         self.randList = [] # 存储已经选用的题目, 防止随机题目
53.

```



```

54.     def generateRand(self):
55.         """ 产生随机题目序号 """
56.         count = 0
57.         while count < self.totalNum:
58.             randCount = random.randint(0, 519) # 共 520 道单选题
59.             if randCount not in self.randList:
60.                 self.randList.append(randCount)
61.                 count = count + 1
62.             else:
63.                 continue
64.
65.     def getData(self):
66.         """ 获取题目, 返回数据给前端 """
67.         self.generateRand()
68.         count = 0
69.         for randCount in self.randList:
70.             # 还有记得, 是不是要 canvas 上面分布这些按钮, 然后随着 canvas 销毁而消失
71.             self.subjectList[count] = {}
72.             self.subjectList[count]['题目内容'] = self.df['题目内容'][randCount]
73.             self.subjectList[count]['A'] = self.df['A'][randCount]
74.             self.subjectList[count]['B'] = self.df['B'][randCount]
75.             self.subjectList[count]['C'] = self.df['C'][randCount]
76.             self.subjectList[count]['D'] = self.df['D'][randCount]
77.             self.subjectList[count]['参考答案'] = self.df['参考答案'][randCount]
78.             count = count + 1
79.         return self.subjectList
80.
81.
82. class MultiChoiceSubject:
83.
84.     def __init__(self):
85.         self.scorePer = 5 # 每道题的分值
86.         self.totalNum = 10 # 总共 10 道单选
87.         self.subjectList = {} # 存放所有题目信息
88.         self.path = getCurrentPath() + DataPath + 'question.xlsx'
89.         self.df = pd.read_excel(self.path, sheet_name='多选')
90.         self.randList = []
91.
92.     def generateRand(self):
93.         """ 产生随机题目序号 """
94.         count = 0
95.         while count < self.totalNum:
96.             randCount = random.randint(0, 265) # 共 520 道单选题
97.             if randCount not in self.randList:

```

```

98.             self.randList.append(randCount)
99.             count = count + 1
100.         else:
101.             continue
102.
103.     def getData(self):
104.         """ 获取题目，返回数据给前端 """
105.         self.generateRand()
106.         count = 0
107.         for randCount in self.randList:
108.             # 还有记得，是不是要 canvas 上面分布这些按钮，然后随着 canvas 销毁而消失
109.             self.subjectList[count] = {}
110.             self.subjectList[count]['题目内容'] = self.df['题目内容'][randCount]
111.             self.subjectList[count]['A'] = self.df['A'][randCount]
112.             self.subjectList[count]['B'] = self.df['B'][randCount]
113.             self.subjectList[count]['C'] = self.df['C'][randCount]
114.             self.subjectList[count]['D'] = self.df['D'][randCount]
115.             self.subjectList[count]['E'] = self.df['E'][randCount]
116.             self.subjectList[count]['参考答案'] = self.df['参考答案'][randCount]
117.             count = count + 1
118.         return self.subjectList
119.
120.
121. class JudgeSubject:
122.
123.     def __init__(self):
124.         self.scorePer = 2 # 每道题的分值
125.         self.totalNum = 10 # 总共 10 道单选
126.         self.subjectList = {} # 存放所有题目信息
127.         self.path = getCurrentPath() + DataPath + 'question.xlsx'
128.         self.df = pd.read_excel(self.path, sheet_name='判断')
129.         self.randList = []
130.
131.     def generateRand(self):
132.         """ 产生随机题目序号 """
133.         count = 0
134.         while count < self.totalNum:
135.             randCount = random.randint(0, 362) # 共 520 道单选题
136.             if randCount not in self.randList:
137.                 self.randList.append(randCount)
138.                 count = count + 1
139.             else:
140.                 continue
141.

```

```

142.     def getData(self):
143.         """ 获取题目, 返回数据给前端 """
144.         self.generateRand()
145.         count = 0
146.         for randCount in self.randList:
147.             self.subjectList[count] = {}
148.             self.subjectList[count]['题目内容'] = self.df['题目内容'][randCount]
149.             self.subjectList[count]['参考答案'] = self.df['参考答案'][randCount]
150.             count = count + 1
151.         return self.subjectList
152.
153.
154. class BackEnd:
155.     """ 与前端的数据接口 """
156.
157.     def __init__(self):
158.         self.Single = SingleChoiceSubject()
159.         self.Multi = MultiChoiceSubject()
160.         self.Judge = JudgeSubject()
161.         self.SingleList = self.Single.getData()
162.         self.MultiList = self.Multi.getData()
163.         self.JudgeList = self.Judge.getData()
164.
165.     def test(self):
166.         print("SingleList:", self.SingleList)
167.         print("MultiList:", self.MultiList)
168.         print("JudgeList:", self.JudgeList)
169.
170.
171. if __name__ == '__main__':
172.     test = BackEnd()
173.     test.test()
174.     print(test.SingleList[0]['A'])
175.     print(test.MultiList[2]['参考答案'])
176.     print(test.JudgeList[9]['题目内容'])
177.     print(type(test.MultiList[2]['参考答案']))
178.     print(test.SingleList[2]['参考答案'])
179.     if test.SingleList[2]['参考答案'] == 'A':
180.         print('aaaa')
181.     if test.SingleList[2]['参考答案'] == 'B':
182.         print('bb')
183.     if test.SingleList[2]['参考答案'] == 'C':
184.         print('cc')
185.     if test.SingleList[2]['参考答案'] == 'D':

```

```
186.         print('dd')
```

AntiCheating.py

```
1. # -*- coding: utf-8 -*-
2. # @Time : 2021/1/20 18:34
3. # @Author : UestcXiye
4. # @File : AntiCheating.py
5. # @Software: PyCharm
6.
7. import os
8. import time
9. import tkinter
10. import threading
11. import ctypes
12. import psutil
13.
14. root = tkinter.Tk()
15.
16. root.title('防作弊演示')
17.
18. # 窗口初始大小和位置
19. root.geometry('250x80+300+100')
20.
21. # 不允许改变窗口大小
22. root.resizable(False, False)
23.
24. ban = tkinter.IntVar(root, 0)
25.
26.
27. def funcBan():
28.     while ban.get() == 1:
29.         # 强行关闭主流文本编辑器和网页浏览器
30.         for pid in psutil.pids():
31.             try:
32.                 p = psutil.Process(pid)
33.                 exeName = os.path.basename(p.exe()).lower()
34.                 if exeName in ('notepad.exe', 'winword.exe', 'wps.exe', 'wordpad.exe',
35.                                'iexplore.exe',
36.                                'chrome.exe', 'qqbrowser.exe', '360chrome.exe', '360se.exe',
37.                                'sougouexplorer.exe', 'firefox.exe', 'opera.exe', 'maxthon.exe',
```

```

37.                 'netscape.exe', 'baidubrowser.exe', '2345Explorer.exe
    '):
38.                 p.kill()
39.             except:
40.                 pass
41.
42.             # 清空系统剪切板
43.             ctypes.windll.user32.OpenClipboard(None)
44.             ctypes.windll.user32.EmptyClipboard()
45.             ctypes.windll.user32.CloseClipboard()
46.             time.sleep(1)
47.
48.
49. def start():
50.     ban.set(1)
51.     t = threading.Thread(target=funcBan)
52.     t.start()
53.
54.
55. buttonStart = tkinter.Button(root, text='开始考试', command=start)
56. buttonStart.place(x=20, y=10, width=100, height=20)
57.
58.
59. def stop():
60.     ban.set(0)
61.
62.
63. buttonStop = tkinter.Button(root, text='结束考试', command=stop)
64. buttonStop.place(x=130, y=10, width=100, height=20)
65. # 模拟用, 开启考试模式以后, 所有内容都不再允许复制
66. entryMessage = tkinter.Entry(root)
67. entryMessage.place(x=10, y=40, width=230, height=20)
68. root.mainloop()

```

ProjectTree.py

```

1. # -*- coding: utf-8 -*-
2. # @Time : 2021/1/20 21:01
3. # @Author : UestcXiyue
4. # @File : ProjectTree.py
5. # @Software: PyCharm
6.
7. from pathlib import Path
8.

```

```

9. tree_str = ''
10.
11.
12. def generate_tree(pathname, n=0):
13.     """ 产生项目结构树图 """
14.     global tree_str
15.     if pathname.is_file():
16.         tree_str += '    |' * n + '-' * 4 + pathname.name + '\n'
17.     elif pathname.is_dir():
18.         tree_str += '    |' * n + '-' * 4 + \
19.             str(pathname.relative_to(pathname.parent)) + '\\ ' + '\n'
20.         for cp in pathname.iterdir():
21.             generate_tree(cp, n + 1)
22.
23.
24. if __name__ == '__main__':
25.     generate_tree(Path.cwd())
26.     print(tree_str)

```

Config.py

```

1. # -*- coding: utf-8 -*-
2. # @Time : 2021/1/20 20:23
3. # @Author : UestcXiyue
4. # @File : Config.py
5. # @Software: PyCharm
6.
7. import os
8.
9. DataPath = '\\ ' + 'data' + '\\ '
10.
11. STATE_INIT = 1
12. STATE_SINGLE = 2
13. STATE_MULTI = 3
14. STATE_JUDGE = 4
15. STATE_DONE = 5
16.
17.
18. def getCurrentPath():
19.     """ 获取当前路径，找寻用户信息表和题库 """
20.     path = os.getcwd() # 当前工作路径
21.     # path = os.path.abspath('.') # 当前工作目录的父目录路径
22.     return path
23.

```

```

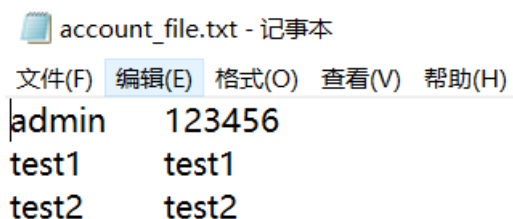
24.
25. def list2str(changList) -> str:
26.     """ 为tkinter 的 varString 显示处理准备, 可以显示考生选择的选项 """
27.     res = ''
28.     for index in range(len(changList)):
29.         res = res + str(changList[index])
30.     return res
31.
32.
33. if __name__ == '__main__':
34.     from pil import Image, ImageTk
35.
36.     pilImage = Image.open(getCurrentPath() + DataPath + "fail.png")
37.     img = pilImage.resize((600, 500), Image.ANTIALIAS)
38.     tkImage = ImageTk.PhotoImage(image=img)
39.     print(pilImage[0])

```

六、数据及项目演示

6.1 数据

account_file.txt 的内容如图 2 所示。



```

account_file.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
admin 123456
test1 test1
test2 test2

```

图 2 account_file.txt 的内容示意图

第一列为账号名, 第二列为账号密码。其中, 管理员账号 admin, 对应密码 123456;

测试账号 test1 和 test2, 密码和账号一样。

题库 question.xlsx 包含单选题、多选题和判断题, 它的内容如图 3 所示。

	A	B	C	D	E	F	G	H	I	J
1	序号	题型	难度	题目内容	A	B	C	D	E	参考答案
2	1	单选题	中	中小微企业和新兴互联网客户（聚焦科技园区、产业园区、创新园区等），提供标准化云服务，主推以下哪项业务？	公有云	私有云	混合云	桌面云		A
3	2	单选题	低	政务云主要面向各级政府定制，提供以（ ）为主的云计算服务。	PAAS	SAAS	IAAS	IDC		C
4	3	单选题	低	（ ）不属于运营商在运营的云。	天翼云	沃云	政务云	阿里云		D
5	4	单选题	低	在政务云市场中，（ ）不是运营商的优势。	网络带宽资源	机房资源	互联网大数据运营经验	运营维护经验		C
		单选题	低	失踪人口和区域强	小区广播	找你找我	警讯通	爱归来		D
		单选	多选	判断						

	A	B	C	D	E	F	G	H	I	J
1	序号	题型	难度	题目内容	A	B	C	D	E	参考答案
2	1	多选题	低	下面哪些情况不适合建设公有云？	云计算配置报价高于传统配置报价	用户软件需要GPU支持、特殊的芯片组、加密硬件或加密狗等	用户软件为简单的门户网站	用户网络环境为涉密网		AED
3	2	多选题	中	关于云计算数据中心IAAS层描述正确的有？	商业视角：云计算=信息电厂	技术角度：云计算=计算/存储的网络	云计算就是传统的数据中心DC	云计算包括两部分：云平台+云服务	桌面云是云计算的一种	ABCD
4	3	多选题	中	下列哪些项是大数据合作的基本准则？	原始数据不出门	个人信息校验需授权	信息输出最小化	数据二次加工需申请	可提供用户信息资料	ABC
5	4	多选题	低	能力开放平台中包含下面哪几个能力？	短信	位置	彩印（电子名片）	小号	GPS	ABCD
	5	多选题	低	中国移动的物联网应用优势是：以移动通信网络为传输载体，为集团客户提供设备、设备到人、人到设备	生产过程监控	指挥调度	远程数据采集	远程诊断	定位监控	ABCDE
		单选	多选	判断						

	A	B	C	D	E	F	G	H	I	J
1	序号	题型	难度	题目内容	A	B	C	D	E	参考答案
2	1	判断题	低	当前中国移动ONE- NET能力 开放平台 提供的大 数据服务 能力应用 包括视频 会议。						错
3	2	判断题	中	公司政企 业务数字 化转型的 平台是云 计算、大 数据。						对
4	3	判断题	中	“企业上 云”的三 大子工程 包括基础 上云、管 理上云、 流程上云 。						错
5	4	判断题	中	移动大数 据在涉及 用户个人 信息资料 验真时， 不需要取 得用户的 主动授权 。						错
		判断题	低	企业在移 动彩云由						错

图 3 题库 question.xlsx 的内容示意图

反映考试成绩的图片 fail.png、pass.png、great.png 分别如图 4、图 5、图 6 所示。

不及格！

图 4 fail.png

及格

图 5 pass.png

优秀!

图 6 great.png

6.2 项目演示

运行 FrontEnd.py, 首先进入考试系统的登陆界面, 输入用户名和密码, 点击登录, 如图 7 所示。



图 7 考试系统登录界面

提示登录成功后，进入考试系统答题界面，如图 8 所示。

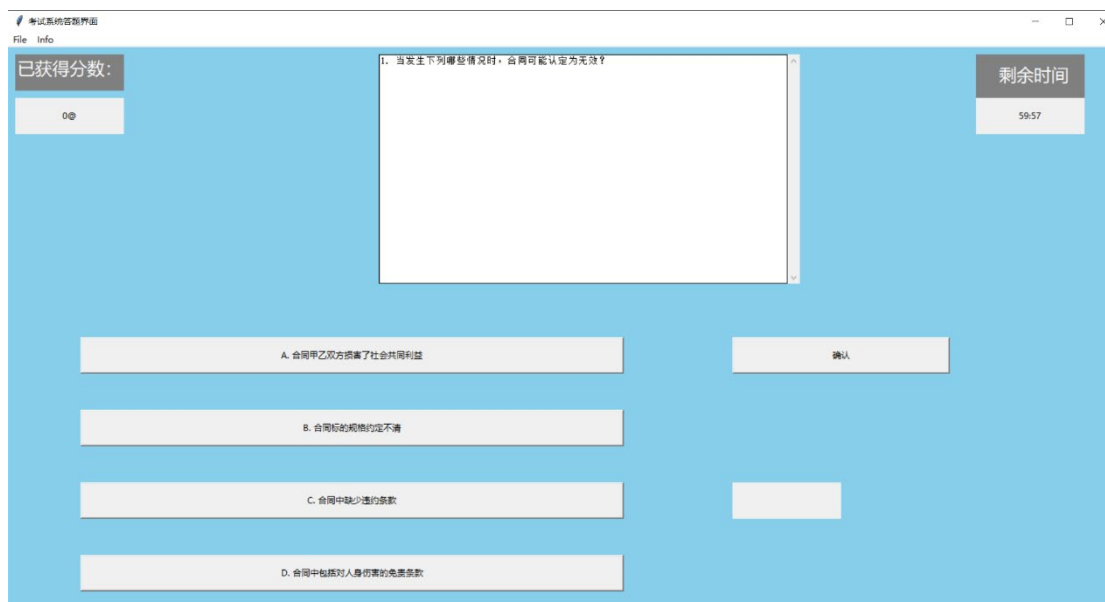


图 8 考试系统答题界面

答题完成后，显示成绩，播放音乐，如图 9 所示。



图 9 考试系统显示成绩界面

七、总结及心得体会：

本次课程设计完成了用 Python 语言编程实现自动组卷评卷考试系统，项目主要实现了从题库中随机抽取试题自动组成试卷、实现考生考试答题操作界面、自动阅卷评分、防作弊演示等功能，加强了 Python 语言的编程能力。

八、对本实验过程及方法、手段的改进建议：

1. 前端界面改用更高级的模块，如 graphic；
2. 采用遗传算法甚至粒子群算法进行自动组卷，使每次生成的试卷难度基本一致；
3. 利用数据库存储题库；
4. 更加完善代码注释，提高代码的可读性。