```
------------------------------------------------------------------------
Interface.py
------------------------------------------------------------------------
from tkinter import *
from tkinter import messagebox
from tkinter import ttk
from generalFileFunctions import*
from adminFunctions import*
from publishTest import*
import time
import os
#from publishTest import*

masterKey = "letmein"
CurrUser = ""
mainCount = 0
qList = []
newqList = []
testNameGlobe = ""
count_terminate = 0

class TestGUI_Interface_loginScr:#Login Screen
    def __init__(self, master):
        self.master = masterKey                     #Master root thread
        master.title("ECI TELECOM Test Interface")  #Window title
        master.geometry("800x600")                  #Window dimensions

        widgetF = Frame(master)                     #Widget Frame

        self.Header = Label(widgetF, text ="ECI TELECOM Login Portal", fg = "#0b34ba",font=("Helvetica", 20))#Header Label
        self.Header.grid(row=0,column=2)#Label pack to frame

        self.ULabel = Label(widgetF, text="Username")   #Text Label
        self.ULabel.grid(row=1,column=1)                #Label pack to frame
        self.UsernameBox = Entry(widgetF)           #Data entry box
        self.UsernameBox.grid(row=1,column=2)       #Box pack

        self.checkMaster = ttk.Checkbutton(widgetF,text="Supervisor Login",command=self.MasterBox)#Checkbox
        self.checkMaster.grid(row=1,column=3)#Checkbox pack

        self.PLabel = Label(widgetF, text="Password")   #Text Label
        self.PLabel.grid(row=2,column=1)                #Label pack to frame
        self.PasswrdBox = Entry(widgetF, show="*")      #Data entry box for password
        self.PasswrdBox.grid(row=2,column=2)            #Data box pack to frame

        self.submitCredential = Button(widgetF, text="Submit", command=self.submit) #Button
        self.submitCredential.grid(row=3,column=2)#Button pack

        self.close_button = Button(widgetF, text="Close", command=root.destroy)#Close program button
        self.close_button.grid(row=4,column=2)#Button pack

        self.unlockmsg1 = Label(widgetF, text="success",fg="blue")#Success message
        self.unlockmsg2 = Label(widgetF, text="error wrong username or password",fg="red")#Error message

        self.MasterBoxE = Entry(widgetF,show="*")#Master pass box


        widgetF.pack()#WidgetFrame pack to window root

    def MasterBox(self):
        self.MasterBoxE.grid(row=2,column=3)#Reveals hidden entry box


    def submit(self):#pack this in the controller class
        #return to database
        usernameArgs = self.UsernameBox.get()       #gets username data
        passwordArgs = self.PasswrdBox.get()        #gets password data
        if (check_User(usernameArgs,passwordArgs)): #Authentication
            #print(usernameArgs, passwordArgs)
            self.unlockmsg1.grid(row=5,column=2)    #Unlock success for employee
            #print(self.checkMaster.state()[0])DEBUGGER
            global masterKey
            #print(masterKey)
            #print(self.MasterBoxE.get())
            if(self.checkMaster.state()[0]=='selected'):#Supervisor unlock
                if(self.MasterBoxE.get() == masterKey):
                    global CurrUser#Global current user store
                    CurrUser = usernameArgs     #Alias
                    root = Tk()                 #New root process spawn
                    GUI1 = TestGUI_Interface_SuperInterface(root)       #Object creation
                    root.mainloop()             #New window spin-off
                else:
                    messagebox.showerror("Title", "Incorrect Root Username or Password")
            else:
                CurrUser = usernameArgs
                root = Tk()         #New root process
                GUI1 = TestGUI_Interface_SubInterface_ViewTests(root)#object creation
                root.mainloop()     #New window spin-off

        else:
            self.unlockmsg2.grid(row=5,column=2)#Unlock error msg
            messagebox.showerror("Title", "Incorrect Username or Password")

class TestGUI_Interface_SuperInterface:
    def __init__(self,master):
        self.master = master
        master.title("ECI TELECOM Test Interface")
        master.geometry("800x600")

        widgetF = Frame(master)
        self.Header = Label(widgetF, text ="Welcome to ECI TELECOM", fg = "#0b34ba",font=("Helvetica", 20),pady = 50)
        self.Header.grid(row=0, column=1)

        self.AdminFunction = Menubutton(widgetF, text="Admin Functions", fg="blue",width=40,height=2)
        self.AdminFunction.grid(row=4, column=1,pady=5)
        self.AdminFunction.menu = Menu(self.AdminFunction, tearoff = 0)
        self.AdminFunction["menu"] = self.AdminFunction.menu
        self.AdminFunction.menu.add_command(label="Add User", command=self.addRemove_Submit)
        self.AdminFunction.menu.add_command(label="Remove User",command=self.addRemove_Submit)

```

```
        self.QuestionBankOptions = Menubutton(widgetF, text="QuestionBank Options", fg="blue",width=40,height=2)
        self.QuestionBankOptions.grid(row=5, column=1,pady=5)
        self.QuestionBankOptions.menu = Menu(self.QuestionBankOptions, tearoff = 0)
        self.QuestionBankOptions["menu"] = self.QuestionBankOptions.menu
        self.QuestionBankOptions.menu.add_command(label="Add Question" , command=self.nextWindow)
        self.QuestionBankOptions.menu.add_command(label="Remove Question", command=self.nextWindow)
        self.QuestionBankOptions.menu.add_command(label="Add new Questionbank", command=self.nextWindow)
        self.QuestionBankOptions.menu.add_command(label="Remove Questionbank", command=self.nextWindow)

        self.PublishTest = Button(widgetF, text="Test Publish", fg="blue", width=40,height=2,command=self.Publish)
        self.PublishTest.grid(row=6, column=1,pady=5)

        self.EmployeeOverview = Button(widgetF, text="Employee Overview", width=40, height=2, command=self.Employee)
        self.EmployeeOverview.grid(row=7, column=1,pady=5)

        widgetF.pack()

    def nextWindow(self):
        root = Tk()
        GUI1 = TestGUI_Interface_SuperInterface_QuestionManager(root)
        root.mainloop()

        widgetF.pack()
    def addRemove_Submit(self):
        root = Tk()
        GUI1 = TestGUI_Interface_SuperInterface_AdF(root)
        root.mainloop()

    def Publish(self):
        root = Tk()
        GUI1 = TestGUI_Interface_SuperInterface_PublishTest(root)
        root.mainloop()

    def Employee(self):
        root = Tk()
        GUI1 = TestGUI_Interface_SuperInterface_Overview(root)
        root.mainloop()

class TestGUI_Interface_SuperInterface_AdF:
    def __init__(self,master):
        self.master = master#Master root
        master.title("ECI TELECOM Test Interface")#Master window
        master.geometry("900x600")#Master window dimension

        widgetM = Frame(master)#Master frame 1
        self.Header = Label(widgetM, text ="Administrative Functions", fg = "#0b34ba",font=("Helvetica", 20), padx = 10, pady = 50)
        self.Header.pack()#Label pack

        widgetF = Frame(master)#Master frame 2
        self.ULabel0 = Label(widgetF, text="Username")  #Text label
        self.ULabel0.grid(row=0,column=2)               #label pack
        self.ULabel0 = Label(widgetF, text="Password")  #Text label
        self.ULabel0.grid(row=0,column=3)               #label pack
        self.ULabel0 = Label(widgetF, text="Password")  #Text label
        self.ULabel0.grid(row=0,column=4)               #label pack
        self.ULabel0 = Label(widgetF, text="E-mail")    #Text label
        self.ULabel0.grid(row=0,column=5)               #label pack

        self.ULabel = Label(widgetF, text="Add User: ") #Text label
        self.ULabel.grid(row=1,column=1)                #label pack
        self.UsernameBox = Entry(widgetF)               #Entry box
        self.UsernameBox.grid(row=1,column=2)           #Entry box pack
        self.PasswrdBox = Entry(widgetF, show ='*')     #Entry box
        self.PasswrdBox.grid(row=1,column=3)            #Entry box pack
        self.PasswrdBox01 = Entry(widgetF, show ='*')   #Entry box
        self.PasswrdBox01.grid(row=1,column=4)          #Entry box pack
        self.EmailBox = Entry(widgetF)                  #Entry box
        self.EmailBox.grid(row=1,column=5)              #Entry box pack
        self.submitCredential = Button(widgetF, text="Submit", command=self.submit) #Action Button
        self.submitCredential.grid(row=2,column=2)                                   #Action button pack


        self.PLabel = Label(widgetF, text="Remove User: ")  #Text label
        self.PLabel.grid(row=3,column=1)                    #label pack
        self.UsernameBox1 = Entry(widgetF)                  #Entry box
        self.UsernameBox1.grid(row=3,column=2)              #Entry box pack
        self.PasswrdBox1 = Entry(widgetF, show ='*')        #Entry box
        self.PasswrdBox1.grid(row=3,column=3)               #Entry box pack

        self.submitCredential1 = Button(widgetF, text="Submit", command=self.submit2)
        self.submitCredential1.grid(row=5,column=2)

        self.unlockmsg1 = Label(widgetF, text="success",fg="blue")                                  #Success message
        self.unlockmsg2 = Label(widgetF, text="error user exist or password too short",fg="red")    #Error message
        self.unlockmsg3 = Label(widgetF, text="User does not exist or incorrect credentials",fg="red")#Error message

        widgetM.pack()#WidgetFrame Pack
        widgetF.pack()#WidgetFrame Pack

    def submit(self):
        #print("lol")
        usernameArgs = self.UsernameBox.get()#checks username to prevent duplicates
        passwordArgs = self.PasswrdBox.get()
        passwordArgs1 = self.PasswrdBox01.get()
        EmailArgs = self.EmailBox.get()
        #print(usernameArgs)


        if (check_User(usernameArgs) != True and passwordArgs != "" and passwordArgs == passwordArgs1):
            self.unlockmsg2.grid_forget()
            self.unlockmsg1.grid(row=6,column=2)
            add_User(usernameArgs, passwordArgs, passwordArgs1, EmailArgs)
            return True
        elif(check_User(usernameArgs) == True):
            messagebox.showerror("Error", "Username Taken")


        else:
            self.unlockmsg2.grid(row=6,column=2)
    def submit2(self):
        usernameArgs = self.UsernameBox1.get()#checks username to prevent duplicates
```

```
        passwordArgs = self.PasswrdBox1.get()
        #print(usernameArgs, passwordArgs)
        if (check_User(usernameArgs,passwordArgs)):
            #print(check_User(usernameArgs,passwordArgs))
            self.unlockmsg3.grid_forget()
            self.unlockmsg1.grid(row=6,column=2)
            rm_User(usernameArgs, passwordArgs)

            return True
        else:
            self.unlockmsg3.grid(row=6,column=2)
            messagebox.showerror("Error", "No such User exists")

class TestGUI_Interface_SuperInterface_QuestionManager:
    SessionBankNameVariable = ""#Stores current session Bank being worked on
    def __init__(self, master):
        self.master = master
        master.title("ECI TELECOM Test Interface")
        master.geometry("800x600")


        widgetF = Frame(master)
        dirlist = os.listdir("%sFileSystem/SuperV/Test_sets/tBank"%(univFilePath))
        #print(dirlist)

        self.label1 = Label(widgetF, text="TestBank")
        self.label1.grid(row=1,column=1)
        self.label2 = Label(widgetF, text="Test Questions")
        self.label2.grid(row=1,column=2,padx=100)
        self.listbox1 = Listbox(widgetF)
        count = 1;
        for f in dirlist:
            self.listbox1.insert(count, f)
            count +=1
        self.listbox1.grid(row=2, column=1)
        self.listbox2 = Listbox(widgetF)
        self.listbox2.grid(row=2, column=2)

        self.submitTBankSelection_Button = Button(widgetF, text="submit", command=self.submitTBankSelection_Function)
        self.submitTBankSelection_Button.grid(row=3,column=1)
        self.removeQuestion_Button = Button(widgetF, text="remove question", command=self.removeQuestion_Function)
        self.removeQuestion_Button.grid(row=2, column=3)

        widgetF.pack()

        widgetG = Frame(master)

        self.label3 = Label(widgetG, text="Add question")
        self.label3.grid(row=1, column=2)

        self.label4 = Label(widgetG, text="Question:")
        self.label4.grid(row=2, column=1)
        self.Box4e = Entry(widgetG)
        self.Box4e.grid(row=2, column=2)

        self.label5 = Label(widgetG, text="Option A:")
        self.label5.grid(row=3, column=1)
        self.Box5e = Entry(widgetG)
        self.Box5e.grid(row=3, column=2)

        self.label6 = Label(widgetG, text="Option B:")
        self.label6.grid(row=4, column=1)
        self.Box6e = Entry(widgetG)
        self.Box6e.grid(row=4, column=2,padx=30)

        self.label7 = Label(widgetG, text="Option C:")
        self.label7.grid(row=5, column=1)
        self.Box7e = Entry(widgetG)
        self.Box7e.grid(row=5, column=2)

        self.label8 = Label(widgetG, text="Option D:")
        self.label8.grid(row=6, column=1)
        self.Box8e = Entry(widgetG)
        self.Box8e.grid(row=6, column=2)

        self.label9 = Label(widgetG, text="Answer:")
        self.label9.grid(row=7, column=1)
        self.Box9e = Entry(widgetG)
        self.Box9e.grid(row=7, column=2)

        self.addQuestion_Button = Button(widgetG,text="add question",command=self.submitAddQuestion)
        self.addQuestion_Button.grid(row=4,column=3,padx=30)


        widgetG.pack()

        widgetH = Frame(master)

        self.label10 = Label(widgetH, text="TestBank")
        self.label10.grid(row=1, column=1)
        self.Box10e = Entry(widgetH)
        self.Box10e.grid(row=2, column=1)

        self.label11 = Label(widgetH, text="Add or Remove TestBank?")
        self.label11.grid(row=1, column=2)
        self.Box11e = Entry(widgetH)
        self.Box11e.grid(row=2, column=2)

        self.testBankButton = Button(widgetH, text="submit", command=self.polyTestSet)
        self.testBankButton.grid(row=2,column=3,padx=30)

        self.PolySuccess = Label(widgetH, text='Success!',fg = "#0b34ba")
        self.PolySuccess2 = Label(widgetH, text='Success!',fg = "red")
        self.PolySuccess3 = Label(widgetH, text='Error',fg = "red")

        widgetH.pack()

    def submitTBankSelection_Function(self):#Modular function allows reuse of button
        TBank_choice = self.listbox1.get(ACTIVE)
        TestGUI_Interface_SuperInterface_QuestionManager.SessionBankNameVariable = TBank_choice#Because variable is not accessisble outside
        #print(TBank_choice)
```

```
        #print(TestGUI_Interface_SuperInterface_QuestionManager.SessionBankNameVariable)
        count = 1;
        for f in questionbank_preview(TBank_choice):
            self.listbox2.insert(count, f)
            count +=1

    def removeQuestion_Function(self):#Modular function allows reuse of button
        #print(TestGUI_Interface_SuperInterface_QuestionManager.SessionBankNameVariable)
        removeQuestion_selection = self.listbox2.get(ACTIVE)
        questionbank_remove(removeQuestion_selection[0], TestGUI_Interface_SuperInterface_QuestionManager.SessionBankNameVariable)
        #print(removeQuestion_selection)

    def submitAddQuestion(self):#Modular function allows reuse of button
        questionbank_add(self.Box4e.get(),self.Box5e.get(),self.Box6e.get(),self.Box7e.get(),self.Box8e.get(),self.Box9e.get(),TestGUI_Interface_Supe
        messagebox.showerror("Success", "Question successfully added")

    def polyTestSet(self):
        BankName = self.Box10e.get()
        BankAction = self.Box11e.get()
        #print(BankName, BankAction)DEBUGGER
        if BankAction == 'add' or BankAction == 'Add':
            setBank(BankName)
            self.PolySuccess.grid(row=3,column=3)
            messagebox.showerror("Success", "Test Bank successfully created")
        elif BankAction == 'remove' or BankAction == 'Remove':
            if(setBank(BankName, BankAction) == 1):
                messagebox.showerror("Error", "Test Bank remove failed")

            else:
                setBank(BankName, BankAction)
                messagebox.showerror("Success", "Test Bank successfully removed")
                self.PolySuccess2.grid(row=4,column=3)
        else:
            self.PolySuccess3.grid(row=5,column=3)
            messagebox.showerror("Error", "Test Bank remove failed")

class TestGUI_Interface_SuperInterface_PublishTest:
    SessionBankNameVariable = ""#Stores current session Bank being worked on
    countGlobal = 1
    listGlobal = []
    def __init__(self, master):
        self.master = master#Master root
        master.title("ECI TELECOM Test Interface")#Window Title
        master.geometry("800x600")#Window dimensions

        dirlist = os.listdir("%sFileSystem/SuperV/Test_sets/tBank"%(univFilePath))#Gets directory contents
        widgetF = Frame(master)#Master frame

        self.label1 = Label(widgetF,text="QuestionBank")#Text label
        self.label1.grid(row=1,column=1)                #label pack

        self.label2 = Label(widgetF,text="Questions")   #Text label
        self.label2.grid(row=1,column=2)                #label pack

        self.label3 = Label(widgetF,text="Questions to Publish")#Text label
        self.label3.grid(row=1,column=3)                        #label pack

        self.listbox1 = Listbox(widgetF)#Listbox scrollable view
        self.listbox1.grid(row=2,column=1)#listbox pack

        self.listbox2 = Listbox(widgetF)#Listbox scrollable view
        self.listbox2.grid(row=2,column=2)#listbox pack

        self.listbox3 = Listbox(widgetF)#Listbox scrollable view
        self.listbox3.grid(row=2,column=3)#listbox pack

        self.listbox1Button = Button(widgetF, text="submit", command=self.TBankSelect)#Button
        self.listbox1Button.grid(row=3,column=1)#Button pack

        self.listbox2Button = Button(widgetF, text=">>", command=self.TBankQuestionSelect)#Button
        self.listbox2Button.grid(row=3,column=2)#Button pack

        self.DoneButton = Button(widgetF, text="Done", command=self.Done)#Button
        self.DoneButton.grid(row=3,column=3)#Button pack


        count=1
        for f in dirlist:
            self.listbox1.insert(count, f)
            count+=1

        widgetF.pack()
    def TBankSelect(self):#Select questionbank
        count2 =1
        TBank_choice = self.listbox1.get(ACTIVE)#Get active listbox selection
        TestGUI_Interface_SuperInterface_QuestionManager.SessionBankNameVariable = TBank_choice#Because variable is not accessisble outside
        for f in questionbank_preview(TBank_choice):#Iteration loop through live preview array
            self.listbox2.insert(count2, f)#insert new listbox entries
            count2+=1
    def TBankQuestionSelect(self):#Select questions
        TBank_choice = self.listbox2.get(ACTIVE)#Get active listbox selection
        self.listbox3.insert(TestGUI_Interface_SuperInterface_PublishTest.countGlobal, TBank_choice)#insert new listbox entry
        TestGUI_Interface_SuperInterface_PublishTest.countGlobal+=1#Global count variable increment
        TestGUI_Interface_SuperInterface_PublishTest.listGlobal.append(TBank_choice)#Global list append new entry
        #print(TestGUI_Interface_SuperInterface_PublishTest.listGlobal) DEBUGGER

    def Done(self):#Publish Exevute
        if(setBankPublish(TestGUI_Interface_SuperInterface_QuestionManager.SessionBankNameVariable)==1):
            messagebox.showerror("Error", "Test Bank duplicate exists")
        else:
            setBankPublish(TestGUI_Interface_SuperInterface_QuestionManager.SessionBankNameVariable)
            for f in TestGUI_Interface_SuperInterface_PublishTest.listGlobal:
                #print(f) DEBUGGER
                questionbank_add_Publish(f[1],f[2],f[3],f[4],f[5],TestGUI_Interface_SuperInterface_QuestionManager.SessionBankNameVariable,f[0])
                messagebox.showerror("Success", "Test Bank successfully Published")

class TestGUI_Interface_SubInterface_ViewTests:
    def __init__(self, master):
        #print('lol') DEBUGGER
        self.master = master#Root thread
        master.title("ECI TELECOM Test Interface")#Window title
```

```
        master.geometry("800x600")      #Window sizing

        dirlist = os.listdir("%sFileSystem/SuperV/Test_sets/tPublish"%(univFilePath))          #Get directory contents

        widgetF = Frame(master)         #Master Frame

        self.label = Label(widgetF, text="Available Tests", fg="#0b34ba", font=('Helvetica',20))#Text Label
        self.label.pack()               #Label pack

        self.listbox1 = Listbox(widgetF)#Scrollable list view
        self.listbox1.pack()            #Listbox pack

        self.taketest = Button(widgetF, text="Take Test", command=self.TestForward)#Submit choice button
        self.taketest.pack()            #Button pack

        count = 1
        for f in dirlist:#Directory content iteration
            self.listbox1.insert(count, f)#Add item to list
            count+=1

        widgetF.pack()#WidgetFrame pack

    def TestForward(self):
        SessionBankNameVariable = self.listbox1.get(ACTIVE)#Which bank
        #print(SessionBankNameVariable)
        global qList
        qList = questionbank_preview_publish(SessionBankNameVariable)#stores session preview
        #print(qList)
        global count_terminate
        for f in qList:
            count_terminate+=1

        root = Tk()     #New root
        GUI1 = TestGUI_Interface_SubInterface_TestTake(root, SessionBankNameVariable)#Object creation
        root.mainloop() #New window

class TestGUI_Interface_SubInterface_TestTake:
    def __init__(self, master, TestName):
        #print(TestName) DEBUGGER
        global testNameGlobe#Global variable reference
        testNameGlobe = TestName#Global update
        self.master = master#Master root
        master.title("ECI TELECOM Test Interface")#Window title
        master.geometry("800x600")#Window sizing

        widgetF =  Frame(master)#Master Frame
        #print(qList) DEBUGGER
        varQues = qList[mainCount][1]#Question variable
        varA = qList[mainCount][2]#Option1 variable
        varB = qList[mainCount][3]#Option2 variable
        varC = qList[mainCount][4]#Option3 variable
        varD = qList[mainCount][5]#Option4 variable

        #print("lol")DEBUGGER
        self.labelQuestion = Label(widgetF, text="Q. "+varQues, font=('Helvetics',16),fg="#0b34ba")#Question Label
        self.labelQuestion.pack()#Label pack


        self.checkA = Label(widgetF, text="A."+varA)#Option1 label
        self.checkA.pack(anchor = W)
        self.checkB = Label(widgetF, text="B."+varB)#Option2 label
        self.checkB.pack(anchor = W)
        self.checkC = Label(widgetF, text="C."+varC)#Option3 label
        self.checkC.pack(anchor = W)
        self.checkD = Label(widgetF, text="D."+varD)#Option4 label
        self.checkD.pack(anchor = W)

        self.Ans = Entry(widgetF)
        self.Ans.pack()


        self.submitButton = Button(widgetF, text="submit", command=self.Submit)#Button
        self.submitButton.pack()#Button pack

        widgetF.pack()#WidgetFrame pack

    def Submit(self):
        answerSub = self.Ans.get()
        answer = ""
        global qList,mainCount#Global reference
        print(qList)
        if answerSub == 'A' or answerSub == 'a':
            answer = qList[mainCount][2]
        if answerSub == 'B' or answerSub == 'b':
            answer = qList[mainCount][3]
        if answerSub == 'C' or answerSub == 'c':
            answer = qList[mainCount][4]
        if answerSub == 'D' or answerSub == 'd':
            answer = qList[mainCount][5]

        questionbank_add_Answer(qList[mainCount][6],qList[mainCount][1],qList[mainCount][2],qList[mainCount][3],qList[mainCount][4],qList[mainCount][
        mainCount+=1
        print(mainCount,count_terminate)
        if mainCount < count_terminate:
            root = Tk()#New root
            GUI1 = TestGUI_Interface_SubInterface_TestTake(root,testNameGlobe)#Object creation
            root.mainloop()#New window
        else:
            root = Tk()#New root
            GUI1 = TestGUI_Interface_SubInterface_TestComplete(root)#Object creation
            root.mainloop()#New window
class TestGUI_Interface_SubInterface_TestComplete:
    def __init__(self, master):
        self.master = master
        master.title("ECI TELECOM Test Interface")
        master.geometry("800x600")

        widgetF = Frame(master)

        self.label = Label(widgetF, text="COMPLETE", font=('Helvetica',50), fg="blue")
        self.label.pack()
```

```
555
556         widgetF.pack()
557
558 class TestGUI_Interface_SuperInterface_Overview:
559     person = ""
560     bank = ""
561     def __init__(self, master):
562         self.master = master
563         master.title("ECI TELECOM Test Interface")
564         master.geometry("800x600")
565
566         widgetF = Frame(master)
567
568         self.label = Label(widgetF, text="Employee Overview", font=('Helvetics',20))
569         self.label.grid(row=1,column=1)
570
571         self.EmployeeList = Listbox(widgetF)
572         self.EmployeeList.grid(row=2,column=1)
573
574         self.TestList = Listbox(widgetF)
575         self.TestList.grid(row=2,column=2)
576
577
578         dirlist = open("%sFileSystem/SuperV/userBase.csv"%(univFilePath))
579
580
581         #print(dirlist)
582
583         count = 1
584
585         for f in dirlist:
586             thing = [word.replace("\n", "") for word in f.split(",")]
587             self.EmployeeList.insert(count, thing[0])
588             #print(thing[0])
589             count+=1
590         self.button1 = Button(widgetF,text="select",command=self.select)
591         self.button1.grid(row=3,column=1)
592
593         self.button2 = Button(widgetF,text="select",command=self.submit)
594         self.button2.grid(row=3,column=2)
595
596
597         widgetF.pack()
598
599         widgetG = Frame(master)
600
601
602     def select(self):
603         TestGUI_Interface_SuperInterface_Overview.person = self.EmployeeList.get(ACTIVE)
604         dirlist = os.listdir("%sFileSystem/SuperV/User/%s"%(univFilePath,TestGUI_Interface_SuperInterface_Overview.person))
605         count = 1
606         for f in dirlist:
607             self.TestList.insert(count,f)
608             count+=1
609
610     def submit(self):
611         #widgetG = Frame(master)
612         choice = self.TestList.get(ACTIVE)
613         TestGUI_Interface_SuperInterface_Overview.bank = choice
614         data = open("%sFileSystem/SuperV/User/%s/%s"%(univFilePath,TestGUI_Interface_SuperInterface_Overview.person,choice))
615         #print(data)
616         testOut = []
617
618         count = 1
619         for i in data:
620             newOut = []
621             thing = [word.replace("\n", "") for word in i.split(",")]
622             for j in range(1,8):
623                 newOut.append(thing[j])
624             testOut.append(newOut)
625         print(testOut)
626         count+=1
627
628         root = Tk()
629         GUI1 = Out(root,testOut,TestGUI_Interface_SuperInterface_Overview.bank)
630         root.mainloop()
631
632
633 class Out:
634     def __init__(self, master,testOut,bank):
635         self.master = master
636         master.title("ECI TELECOM Test Interface")
637         master.geometry("800x600")
638
639         widgetF = Frame(master)
640         self.label = Label(widgetF,text="Ref No., Question, Options(A,B,C,D), Response, Correct Ans")
641         self.label.pack()
642         self.listbox = Listbox(widgetF)
643         self.listbox.pack()
644         count = 1
645         for f in testOut:
646             self.listbox.insert(count,f)
647             count+=1
648
649         self.listans = Listbox(widgetF)
650         self.listans.pack()
651         count= 1
652         data = open('%sFileSystem/SuperV/Test_sets/tBank/%s'%(univFilePath,bank))
653         for f in data:
654             self.listans.insert(count,f)
655             count+=1
656         widgetF.pack()
657
658
659
660
661 root = Tk()
662 GUI1 = TestGUI_Interface_loginScr(root)
663 root.mainloop()
664
665 ------------------------------------------------------------------------
```

```
666
667
668
669 ------------------------------------------------------------------------
670
671 generalFileFunctions.py
672 ------------------------------------------------------------------------
673
674 import csv;
675 import os;
676
677 univFilePath = "../"# A universal Filepath to the location of program's FileSystem
678
679 def questionbank_add(Q,A,B,C,D,CAns,CurrBank):#Function to add questions to a QuestionBank DataBase
680     with open("%sFileSystem/SuperV/Test_sets/tBank/%s"%(univFilePath,CurrBank), "a+") as file_handle:#FileHandling
681         reader = csv.reader(file_handle, delimiter=',');#Initiation of reader handle
682         writer = csv.writer(file_handle, delimiter=',');#Initiation of writer handle
683
684         file_handle.seek(0) #Resests the cursor to the top of the file
685         sl_no = 1           #Slot Number Counter to number the levels of data
686
687         for x in reader:#Iteration loop through reader handle
688            sl_no+=1     #Counts the number of rows present in the database
689         writer.writerow([sl_no,Q,A,B,C,D,CAns]);
690
691 def questionbank_preview(bank):#returns the entire contents of a QuestionBank databse
692     with open("%sFileSystem/SuperV/Test_sets/tBank/%s"%(univFilePath,bank), "a+") as file_handle:
693         reader = csv.reader(file_handle, delimiter=',');#Initiation of reader handle
694         writer = csv.writer(file_handle, delimiter=',');#Initiation of writer handle
695         preview = [] #Array that stores the entire database for live use in the program (eg: previewing database contents)
696
697         file_handle.seek(0);#Resests the cursor to the top of the file
698
699         for row in reader:      #Iteration loop through reader handle
700             preview.append(row) #Appends 1 row of database at a time (iteratively) into elements (2D array of elements of rows)
701
702         return preview          #Returns the entire database to the class that calls it
703             #print(row) DEBUGGER
704
705
706 def questionbank_remove(qNums,CurrBank):#Function to remove questions from a QuestionBank DataBase
707     row_mem = []#Holds entire questionbank database for live use by program
708     try:
709         with open("%sFileSystem/SuperV/Test_sets/tBank/%s"%(univFilePath,CurrBank), "r+") as file_handle:
710             reader = csv.reader(file_handle, delimiter=',');#Initiation of reader handle
711             writer = csv.writer(file_handle, delimiter=',');#Initiation of writer handle
712
713             file_handle.seek(0) #Resests the cursor to the top of the file
714             x_count = 0;        #Counter that checks for serial number of question to be removed
715
716             for row in reader:          #Iteration loop through reader handle
717                 if(row[0]!=qNums):      #Checks for serial number match
718                     row_mem.append(row) #If doesn't match, then append to live array (temp database)
719                 x_count += 1
720
721             if x_count == 0:
722                 return 'QuestionBank Empty' #Empty database handling statement
723
724             #print(row_mem) debugging purposes
725             file_handle.close()
726         with open("%sFileSystem/SuperV/Test_sets/tBank/%s"%(univFilePath,CurrBank), "w+") as file_handle:
727             reader = csv.reader(file_handle, delimiter=',');#Initiation of reader handle
728             writer = csv.writer(file_handle, delimiter=',');#Initiation of writer handle
729
730             x_count = 1
731
732             for row_append in row_mem:      #Iteration loop through temporary database array
733                 row_append[0] = x_count     #Serialisation
734                 writer.writerow(row_append) #Filewriter
735                 #print(row_append[0])       #debugging purposes
736                 x_count += 1
737
738     except Exception:
739         print("Unable to handle file")#Error handling erratic behaviour
740
741 def setBank(setName, confirmation=None):#Polymorphised function to create/remove test sets
742     #TestSet add new testBank polymorphised
743     if confirmation is None:#Functional Polymorphism in python
744         setName = setName + ".csv"
745         flag_duplicate = False  #Duplicate error checking
746         fileList = []           #holds directory content list
747         fileList = os.listdir("%sFileSystem/SuperV/Test_sets/tBank"%(univFilePath)) #Searches directory contents
748         for x in fileList:
749             if x == setName:
750                 flag_duplicate = True #Flags duplicate
751         if flag_duplicate == False: #Flag condition checking
752             try:
753                 open("%sFileSystem/SuperV/Test_sets/tBank/%s" %(univFilePath,setName), "w+")
754             except:
755                 print("unexpected error")   #Unexpected error handling
756         elif confirmation is not None:
757             print("duplicate exists")       #Duplicate error handling
758     else:
759         #TestSet remove testBank polymorphised
760         setName = setName + ".csv"
761         try:
762             os.remove("%sFileSystem/SuperV/Test_sets/tBank/%s" %(univFilePath,setName)) #Uses os library to remove test Database
763         except:
764             print("file does not exit")#Unexpected error handling
765             return 1;
766
767
768 #Legacy Code UPDATED
769 '''def setBank(setName,confirmation):#certified working + add try catch,polymorphism
770     #TestSet remove polymorphised
771     setName = setName + ".csv"
772     try:
773         os.remove("%sFileSystem/SuperV/Test_sets/tBank/%s" %(univFilePath,setName))
774     except:
775         print("file does not exit")#
776     #needs return statement'''
```

```
#DEBUGGER
#questionbank_add('Question','A','B','C','D','ans')
'''
questionbank_remove('1')
questionbank_preview()
make_set("lol")
rm_set("lol")
'''

--------------------------------------------------------------------------



--------------------------------------------------------------------------
adminFunctions.py
--------------------------------------------------------------------------
import csv
import os
import traceback
import hashlib
import os

univFilePath = "../"

def add_User(UID, passkey, passkey2,mailID):#Admin Function to add new user
    try:
        with open("%sFileSystem/SuperV/userBase.csv"%(univFilePath), "a+") as file_handle:#File handling
            reader = csv.reader(file_handle, delimiter=',');#File reader handling
            writer = csv.writer(file_handle, delimiter=',');#file writer handling
            file_handle.seek(0)                             #Reset cursor to database top

            for x in reader:    #Iteration of reader handle
                if x[0] == UID: #Duplicate error handling
                    return "User already exists"
            if passkey != passkey2:
                return "passwords do not match" #Password match check

            file_handle.seek(2)                 #Cursor seek to database bottom

            m = hashlib.sha256()
            m.update(passkey.encode('utf8'))
            hash = m.hexdigest()

            writer.writerow([UID,hash,mailID])                      #Write new user
            os.mkdir("%sFileSystem/SuperV/User/%s"%(univFilePath,UID))  #Make user file directory
            file_handle.close()     #Memory leak prevention
    except Exception:
        return traceback.print_exc()#Better error messages

def check_User(UID, passkey=None):#Functional Polymorphism
    if(passkey is not None):
        try:
            with open("%sFileSystem/SuperV/userBase.csv"%(univFilePath), "r") as file_handle:
                reader = csv.reader(file_handle, delimiter=',');#File writer handling
                file_handle.seek(0)#Reset cursor to database top
                #print(UID,passkey) DEBUGGER Code

                m = hashlib.sha256()
                m.update(passkey.encode('utf8'))
                hash = m.hexdigest()

                for x in reader:
                    #print(x[0],x[1]) DEBUGGER Code
                    if x[0] == UID and x[1] == hash: #Verifies user credentials from database
                        return True
                return False
        except:
            return "Error"
    else:
        try:
            with open("%sFileSystem/SuperV/userBase.csv"%(univFilePath), "r") as file_handle:
                reader = csv.reader(file_handle, delimiter=',');
                file_handle.seek(0)
                #print(UID,passkey)

                for x in reader:
                    #print(x[0],x[1])
                    if x[0] == UID: #Searches if user exists
                        return True
                return False
        except:
            return "Error"



def rm_User(UID, passkey):#Remove User credentials from database
    try:
        with open("%sFileSystem/SuperV/userBase.csv"%(univFilePath), "r+") as file_handle:#File handling
            reader = csv.reader(file_handle, delimiter=',');#File reader handling
            writer = csv.writer(file_handle, delimiter=',');#File writer handling

            row_mem = []#Array to temp store entire userBase database
            x_count = 0
            y_count = 0


            m = hashlib.sha256()
            m.update(passkey.encode('utf8'))
            hash = m.hexdigest()

            for row in reader:                          #Iteration loop
                if row[0] != UID or row[1] != hash:  #User delete confirmation credential validation
                    row_mem.append(row)
                x_count += 1
                y_count += 1

            if x_count == 0:
                return "User database empty"#Error handling for empty database
            file_handle.close()             #Memory leak prevention
        with open("%sFileSystem/SuperV/userBase.csv"%(univFilePath), "w+") as file_handle:#File handling
```

```
            writer = csv.writer(file_handle, delimiter=',');#File writer handling
            write_count = 0                                 #Counter variable


            for row in row_mem:     #Iteration loop to overwrite existing file without removed user
                writer.writerow(row)
                write_count += 1

            file_handle.close()     #Memory leak prevention

        if write_count == y_count - 1:
            return 0#operation confirmation
        else:
            return 1  #Returns if UID password correct
    except:
        return "Unable to remove user"                      #Exception handling unexpected behaviour

#add_User("Denaliis", "lol", "lol", "lolsoop@gmail.com")
#rm_User('Denalii', 'lol')

-------------------------------------------------------------------------



-------------------------------------------------------------------------
publishTest.py
-------------------------------------------------------------------------
import csv
import smtplib, ssl
import os
from datetime import date

today = str(date.today())

univFilePath = '../'

def setBankPublish(setName):#TestBank Subset publish to ready to use
    flag_duplicate = False  #Boolean flag variable
    fileList = []           #List of directory contents
    fileList = os.listdir("%sFileSystem/SuperV/Test_sets/tPublish"%(univFilePath))#Searched directory for content
    for x in fileList:              #Iteration through directory content list
        if x == setName:            #Duplicate error handling
            flag_duplicate = True   #Flag handling
    if flag_duplicate == False:
        try:
            open("%sFileSystem/SuperV/Test_sets/tPublish/%s" %(univFilePath,setName), "w+")#Exception Handling file open
        except:
            print("unexpected error")   #Unexpected behaviour handling
    else:
        print("duplicate exists")       #Duplicate error handling
        return 1

def questionbank_add_Publish(Q,A,B,C,D,CurrBank,Reference_No):#Add questions to final publishable subset TestBank
    with open("%sFileSystem/SuperV/Test_sets/tPublish/%s"%(univFilePath,CurrBank), "a+") as file_handle:#File handling
        reader = csv.reader(file_handle, delimiter=',');#File reader handle
        writer = csv.writer(file_handle, delimiter=',');#File writer handle

        file_handle.seek(0) #Returns cursor to file top
        sl_no = 1           #Slot number counter

        for x in reader:    #Iteration through reader handle
           sl_no+=1
        writer.writerow([sl_no,Q,A,B,C,D,Reference_No]);#Csv writerow File I/O

def questionbank_preview_publish(bank):#Flexible bank preview that holds question for test taking
    with open("%sFileSystem/SuperV/Test_sets/tPublish/%s"%(univFilePath,bank), "a+") as file_handle:
        reader = csv.reader(file_handle, delimiter=',');#File reader handle
        writer = csv.writer(file_handle, delimiter=',');#File writer handle
        preview = []        #Array that temporarily holds final publishing test live
        file_handle.seek(0);#Returns cursor to file top

        for row in reader:      #Iteration through reader handle
            preview.append(row) #Adding data to temp live array database

        return preview#Returns value to calling function

def questionbank_add_Answer(RealRef,Q,A,B,C,D,Response,CurrUser,CurrBank):#Records student response to publish qbase subset
    with open("%sFileSystem/SuperV/User/%s/%s"%(univFilePath,CurrUser,CurrBank), "a+") as file_handle:#File handling
        reader = csv.reader(file_handle, delimiter=',');#File reader handle
        writer = csv.writer(file_handle, delimiter=',');#File writer handle

        file_handle.seek(0) #Returns cursor to file top
        sl_no = 1           #Slot number counter

        for x in reader:#Iteration through reader handle
           sl_no+=1     #Counts slot number
        writer.writerow([sl_no,RealRef,Q,A,B,C,D,Response]);#Writes row that records student response + actual question
        #print([sl_no,RealRef,Q,A,B,C,D,Response]) DEBUGGER prompt

'''def publishTest(tBankName, randomised):
    try:
        with open("FileSystem/SuperV/%sProcess/%s" %(univFilePath, recipientStudentFile), "r+") as file_handle:
            reader = csv.reader(file_handle, delimiter=',');

            recipients = []
            person_count = 0

            for name in reader:
                recipients.append(name);
                person_count += 1
            #print(person_count) release 1

            file_handle.close()

        with open("%sFileSystem/SuperV/userBase.csv"%(univFilePath), "r+") as file_handle:
            reader = csv.reader(file_handle, delimiter=',');

            info_gather = []
            #person_count_checksum = 0 ??
            mismatch_list = []
            x = 0
```

```
999
1000            for x in range(person_count):#confirms exixtance of specified persons
1001                for name in reader:
1002                    #print(recipients[x][0],name[0]) release 2
1003                    if(recipients[x][0] == name[0]):#case sensitive
1004                        #print('lol')
1005                        info_gather.append([name[0],name[2]])
1006                        #print(recipients[x][0])
1007                        file_handle.seek(0)
1008                        break
1009                    #else:
1010                        #print('miss',recipients[x])
1011
1012                #print(x)
1013                file_handle.seek(0)
1014            print(info_gather)
1015            if(person_count_checksum != person_count):
1016                return "%s could not be located" %(mismatch_list) #future proj
1017            #print (recipients) debug now legacy
1018    except Exception:
1019        return "Error no such file"
1020 '''
1021 ------------------------------------------------------------------------
```