COMP1811 - Python Project Report

Name	**Zarin Tasnim**		
Login id	**001086197**		
Group member names	**Karina Busmane**	**Mahdi Rahman**	**Lakshman Thillainathan**
Group member login IDs	**001084791-x**	**001061067-7**	**001064670-1**

1. Brief statement of features you have completed

1.1 Circle the parts of the coursework you have **fully completed and are fully working**. Please be accurate.	**Part A** ☒ A.1 ☒ A.2 ☐ A.3 ☒ A.4 ☐ A.5 ☐ A.6 **Part B** ☒ B.1 ☐ B.2 ☐ B.3 ☐ B.4
1.2 Circle the parts of the coursework you have **partly completed or are partly working.**	**Part A** ☒ A.1 ☒ A.2 ☐ A.3 ☐ A.4 ☐ A.5 ☐ A.6 **Part B** ☐ B.1 ☐ B.2 ☐ B.3 ☐ B.4

A.1. When we run the program it shows our frontend system - login window, it asks for a username and password. It lets you enter voting application only if the username is saved in the file. Provides a message if it is wrong and doesn't let you login to the voting application.
A.2. Reads usernames and passwords from the text file to check if the details are correct. It prompts a message if it the username and/or password is wrong, which means that user is not eligible to vote or input is not right.
A.3. When admin logs in it allows to upload candidates for each position, this is a backend feature.
A.4. It allows to vote only on the current date which is entered manually. Shows all the candidates running for each of the positions. Checks duplications. When votes has been entered it saves all data in to *Votes.csv* file.
A.5. In the result window it shows the total number of votes for each candidate, shows the winner of each position based on vote count and if there is a tie uses preferences to choose the winner.
A.6. Allows to choose GSU position for which user wants to see results for, displays all the candidates and how many votes they got. Displays the winner for each position and how many votes they have, and percentage from total votes to how many were

2. Concise List of Bugs and Weaknesses

2.1 Bugs

1. One bug for this program is that there can be more than one preference for each person, it should not be like that. Instead, we should make it so you can only choose one candidate for each preference.
2. For window where we show the results of votes, we should have a back button to go back to the menu page, instead of exiting it and then we have to login again. It is easy, but we didn't realize that before and when we did it was due time limit - could add it with a button widget as we used it for all other buttons.
3. When you try to login with empty username and password boxes it should show a label that says "Insert your username and password" instead of "Incorrect username and password", we tried to fix this but because of errors we could not manage to work, so we left it like that as it isn't completely wrong.
4. All the windows could be in one, so it is more pages not windows. Now it opens a new window each time you open the next page. It would be better if we would use frames for that part, then it would open a new page instead of the whole window.
5. We didn't add diagrams for result page because of time limitation, but it could have been easily done by importing *matplotlib.figure* and *matplotlib.patches* and making a diagram who reads data from our CSV file.

6. Also, we didn't add total for all the votes together in total, because of the time limitation. We could make it using math and just to calculate each vote from CSV file, where we managed all votes together. So we could just sum them and that would be the total.
7. We used CSV files instead of text file for listing candidates and votes, because it is much cleaner and in our opinion much more understandable for anyone, but it could be made with a text file as well with the exact same principle as from CSV file.
8. We could have used dictionaries for the candidates and linked them easily rather than a list.

2.2 Weaknesses

1. Design. Working with Tkinter made everything much easier if we are talking about user interface, but as we didn't have any previous experience that was not as easy as we thought it will be. We wanted to add much more widgets and features, but while exploring big Tkinter opportunities we already run out of time, and that's why we made it simple and understandable for anyone, but not the best that it could be.
2. When we changed login part from SQL database to txt file, my program stopped working properly. Even if I enter password from txt file it still shows that it is not correct, even though it is. Then we decided to use a csv file as it would have been easier to sort the data.
3. We made the project in 4 separate parts, when we started to put it all together some of the parts stopped working, as example the part which tells that username and password is not right or the login input labels are empty.
4. In the voting screen GUI the user is able to select a candidate as not only one of their preferences but all of their preferences. This is a slight bug and it affects the final results. We have come to the conclusion that this could be sorted with a few if statements to make sure that 2 preferences cannot be clicked for one person.
5. The back button was not appearing on the results screen.
6. The *addCandididates* function did not append to the csv file.

3. Classes and OOP Features

3.1 Classes Used

-VoteWindow

-ApplyRole

-ResultsWindow

3.2 Brief Explanation of Class Design and OOP Features Used

We used OOP to create most of our interface as it allowed us to easily structure the widgets. We also used OOP to create our Candidates, this proved to be essential for our code as we were easily able to access all data on candidates through reading a list full of Candidate objects.

4. Code for the Classes Created

4.1 Class ApplyRole

```
class ApplyRole:
	def __init__(self,firstName,surName,role):
	self.firstName = firstName
	self.surName = surName
	self.role = role
	with open('vote_data.csv', mode='r') as vData:
	csv_reader = csv.reader(vData, delimiter=',')
	for row in csv_reader:
	if self.firstName == row[0]:
				self.firstChoice = int(row[2])
				self.secondChoice = int(row[3])
				self.thirdChoice = int(row[4])
				self.fourthChoice = int(row[5])
```

The ApplyRole class is used when creating Candidate objects, providing the candidate object with characteristics that all candidates will have. This also made it easy for us to split the roles apart. We also found this much more beneficial than constantly reading the csv or text file.

4.2 Class VoteWindow

```
class VoteWindow:
	def __init__(self,root):
	self.roleCount = 0
	self.roleList = ["GSUOfficer","FacultyOfficer","President"]
	self.Title = Label(root, text="Voting for "+self.roleList[self.roleCount])
	self.Title.grid(row=0,column=2)
	self.frame = Frame(root)
	FirstPref = Label(root,text="1st preference").grid(row=1, column=1)
	SecondPref = Label(root,text="2nd preference").grid(row=1, column=2)
	ThirdPref = Label(root,text="3rd preference").grid(row=1, column=3)
	FourthPref = Label(root,text="4th preference").grid(row=1, column=4)
	self.CandidateObjectList = self.CreateObjList()
	self.selectionScreen(root,self.roleCount)
```

```
        def selectionScreen(self,root,roleCount):
        rowNum = 2
        num = 0
        self.posOfCand = []
        pref1 = StringVar()
        pref2 = StringVar()
        pref3 = StringVar()
        pref4 = StringVar()
        self.TempRows = copy.deepcopy(self.CandidateObjectList)
        roleList = ["GSUOfficer","FacultyOfficer","President"]
        for i in self.TempRows:
        if i.role == roleList[self.roleCount]:
                TempText = str(i.firstName + " " + i.surName)
                self.TempRows[num] = Label(root,text=TempText)
                 self.TempRows[num].grid(row=rowNum)
    Radiobutton(root,variable=pref1,value=TempText).grid(row=rowNum,column=1)

Radiobutton(root,variable=pref2,value=TempText).grid(row=rowNum,column=2)
           Radiobutton(root,variable=pref3,value=TempText).grid(row=rowNum,column=3)
           Radiobutton(root,variable=pref4,value=TempText).grid(row=rowNum,column=4)
                rowNum +=1
                self.posOfCand.append(num)
        num+=1

        if self.roleCount == 2:
        self.confirmButton = Button(root,text="Confirm",command=lambda:
self.clicked(pref1.get(),pref2.get(),pref3.get(),pref4.get(),True))
        self.confirmButton.grid(row=rowNum,column=5)
        else:
        self.confirmButton = Button(root,text="Next",command=lambda:
self.clicked(pref1.get(),pref2.get(),pref3.get(),pref4.get(),False))
        self.confirmButton.grid(row=rowNum,column=5)

        def clicked(self,pref1,pref2,pref3,pref4,confirmed):
        self.confirmButton.destroy()
        for i in self.posOfCand:
        self.TempRows[i].destroy()
        for i in self.CandidateObjectList:
        if i.firstName and i.surName in pref1:
           print("#",i.firstName,i.surName, pref1)
                i.firstChoice+=1
                f = open('vote_data.csv','r')
                reader = csv.reader(f)
                mylist = list(reader)
                f.close()
                row_count = 0
                for row in mylist:
                if row[0] == i.firstName:
                print(i.firstName)
                mylist[row_count][2] = i.firstChoice
                else:
                row_count+=1
                my_new_list = open("vote_data.csv","w",newline ='')
                csv_writer = csv.writer(my_new_list)
                csv_writer.writerows(mylist)
                my_new_list.close()
        if i.firstName and i.surName in pref2:
                i.secondChoice+=1
                f = open('vote_data.csv','r')
                reader = csv.reader(f)
                mylist = list(reader)
                f.close()
                row_count = 0
                for row in mylist:
                if row[0] == i.firstName:
                  mylist[row_count][3] = i.secondChoice
                else:
                row_count+=1
                my_new_list = open("vote_data.csv","w",newline ='')
                csv_writer = csv.writer(my_new_list)
                csv_writer.writerows(mylist)
                my_new_list.close()
        if i.firstName and i.surName in pref3:
                i.thirdChoice+=1
                f = open('vote_data.csv','r')
                reader = csv.reader(f)
                mylist = list(reader)
                f.close()
                row_count = 0
                for row in mylist:
                if row[0] == i.firstName:
                mylist[row_count][4] = i.thirdChoice
                else:
                row_count+=1
                my_new_list = open("vote_data.csv","w",newline ='')
                csv_writer = csv.writer(my_new_list)
                csv_writer.writerows(mylist)
                my_new_list.close()
        if i.firstName and i.surName in pref4:
                i.fourthChoice+=1
                f = open('vote_data.csv','r')
                reader = csv.reader(f)
                mylist = list(reader)
                f.close()
                row_count = 0
                for row in mylist:
                if row[0] == i.firstName:
                print(i.firstName)
                mylist[row_count][5] = i.fourthChoice
                else:
                row_count+=1
                my_new_list = open("vote_data.csv","w",newline ='')
                csv_writer = csv.writer(my_new_list)
                csv_writer.writerows(mylist)
                my_new_list.close()

        if confirmed == True:
                root.destroy()
                Home.deiconify()
        else:
                self.roleCount+=1
                self.Title.config(text="Voting for "+self.roleList[self.roleCount])
                self.selectionScreen(root,self.roleCount)

        def CreateObjList(self):
                if not CandsObjectList:
                for i in CandsList:
                 CandsObjectList.append(str(i[1])+str(i[2]))
                CandsObjectList[-1] = ApplyRole(i[1],i[2],i[0])
        return CandsObjectList
```

The VoteWindow class contains all the widgets used within tkinter for our functions that happen when a button is pressed. When the class is initialised, we read from the vote_data csv file to the objects, this made the initialising easier.

The selectionScreen function creates the radio buttons assigning the inputs into variables that can be read from when inserting the selected data into the vote_data file. The amount of radio buttons and labels produced will depend on how many candidates are enrolling for that role (which is read off the text file).

The clicked function is used for when the button is pressed. We had to ensure that all the labels and buttons were reset in the vote window otherwise the buttons and labels would stack. We added the vote to the object of the candidate, then we writ into the excel file to store the totals. One con about using csv files is that you can’t replace specific cells unless you rewrite the entire line, we dealt with this by holding a temporary line for the data to then be inserted back into the file. This process is repeated for all the preferences selected.

The CreateObjList function is used to convert the list of candidates into objects. This is then returned as a list with several objects.

4.3 Class ResultsWindow

```
class ResultsWindow:
  def __init__(self,results):
      self.Title = Label(results, text="Role:")
      self.Title.grid(row=1,column=40)
      self.width = 1200
      self.height = 600
      screen_width = results.winfo_screenwidth()
      screen_height = results.winfo_screenheight()
      self.x = (screen_width/2) - (self.width/2)
      self.y = (screen_height/2) - (self.height/2)
      results.geometry("%dx%d+%d+%d" % (self.width, self.height, self.x, self.y))
      TableMargin = Frame(results, width=200,bg="yellow")
      TableMargin.grid(row=3,column=4)
      self.tree = ttk.Treeview(TableMargin, columns=("First Name","Sur Name", "1st preference", "2nd
preference", "3rd preference", "4th preference"), height=400)
      self.tree.heading('First Name', text="First Name", anchor=W)
      self.tree.heading('Sur Name', text="Sur Name", anchor=W)
      self.tree.heading('1st preference', text="1st preference", anchor=W)
      self.tree.heading('2nd preference', text="2nd preference", anchor=W)
      self.tree.heading('3rd preference', text="3rd preference", anchor=W)
      self.tree.heading('4th preference', text="4th preference", anchor=W)
      self.tree.column('#0', stretch=NO, minwidth=0, width=0)
      self.tree.column('#1', stretch=NO, minwidth=0, width=100)
      self.tree.column('#2', stretch=NO, minwidth=0, width=100)
      self.tree.column('#3', stretch=NO, minwidth=0, width=100)
      self.tree.column('#4', stretch=NO, minwidth=0, width=100)
      self.tree.column('#5', stretch=NO, minwidth=0, width=100)
      self.tree.grid(column=0,row=0)
      self.role = StringVar(results)
      roleCombo = ttk.Combobox(results, textvariable=self.role,
                        values=[
                          "President",
                          "GSUOfficer",
                          "FacultyOfficer"])
      roleCombo.grid(column=40,row=2)
      roleCombo.current(1)
      roleCombo.bind("<<ComboboxSelected>>",self.limp())
      okbutton = Button(results, text="OK",command=self.limp)
      okbutton.grid(column=41,row=2)



  def limp(self):
      self.tree.delete(*self.tree.get_children())
```

```
        GetList = VoteWindow(None)
        CandidateObjList = GetList.CreateObjList()
        for i in CandidateObjList:
            if i.role == self.role.get():
                firstName = i.firstName
                surName = i.surName
                firstpreference =i.firstChoice
                secondpreference =i.secondChoice
                thirdpreference =i.thirdChoice
                fourthpreference =i.fourthChoice

self.tree.insert("",0,values=(firstName,surName,firstpreference,secondpreference,thirdpreference,fourthpr
eference))
        temp1st = 0
        temp1stnum = 0
        totalVotes = 0
        for i in CandidateObjList:
            if i.role == self.role.get():
                totalVotes = totalVotes + i.firstChoice + i.secondChoice + i.thirdChoice + i.fourthChoice
                if int(i.firstChoice) > int(temp1stnum):
                    temp1stnum = i.firstChoice
                    temp1st = i
                elif int(i.firstChoice) == int(temp1stnum):
                    if int(i.secondChoice) > int(temp1st.secondChoice):
                        temp1stnum = i.secondChoice
                        temp1st = i
                    elif int(i.secondChoice) == int(temp1stnum):
                        if int(i.thirdChoice)>int(temp1st.thirdChoice):
                            temp1stnum = i.thirdChoice
                            temp1st = i
                        elif int(i.thirdChoice) == int(temp1stnum):
                            if int(i.fourthChoice) > int(temp1st.thirdChoice):
                                temp1st = i
                            elif int(i.fourthChoice) == int(temp1stnum):
                                print("Draw")

        receivedVotes = temp1st.firstChoice + temp1st.secondChoice + temp1st.thirdChoice +
temp1st.fourthChoice
        WinLabel = Label(results,text=" " + temp1st.firstName+" "+temp1st.surName+" is winner for
position of"+self.role.get()+"\nreceiving a total of "+str(receivedVotes) + " votes.\n" +
str(round((receivedVotes/totalVotes)*100))+"%")
        WinLabel.grid(row=2,column=42)
```

The ResultsWindow class is used to display the Results screen, we used the tree widget to display the results of the candidates. This is simply read from the vote_data csv file and output depending on what role the user has selected in the combobox.

The limp function is used to place the data accordingly to the headings. The information is read straight from the vote_data file and then read along with the list of candidate objects in order for it to be compared to see which data we need.

5. Description of the features implemented

Describe your implementation of the required features and how well do they work. Provide some exposition of the design decisions made and indicate how the features developed by group members were integrated.

(THIS SECTION SHOULD BE THE SAME FOR ALL GROUP MEMBERS)

This is how we made the login part with user input, it simply shows where to insert username and password with little insert windows. For password we used password type, so when user inserts the password no one can see it. Also, we made a little label : “Enter your username and password”, “Please complete the required field” for empty boxes and “Invalid username and/or password” if user wrote wrong information or is not in the system. We added an exit button which closes the login window.

When you enter the right username and password and you are allowed to vote, you enter menu where you can choose to vote, see the results and go back to the login page. If you enter the details for an admin (username =admin ,password = 123), you will be provided with an extra feature of being able to add a new candidate. We implemented this in the idle but not into the gui due to time constraints. We also made it so that if you are admin you are able to view results, as a normal user would not be able to view results until the specified day is met.

First option – vote button – Opens a new window with name of the candidates and radio buttons which are assigned to the 1st, 2nd, 3rd, 4th preferences and button next to record votes and to go to the next one where you can vote for President, GSU Officer and Faculty Officer. We struggled to visualise the way in which to display each candidate. First we thought about asking the user for which candidates they would like to vote for and took their text input. However, we found this to be very impractical and does not make full use of the widgets provided by tkinter. We also were using check boxes instead of radio buttons in the voting screen, we found this to be a struggle to work with due to the lack of being able to prevent the user from clicking 2 of the same preference. We also struggled to connect the window with the menu window as this was a learning experience for us with classes and tkinter. However, we overcame them through several hours of research and different drafts.

Second option – results button – Opens a new window where you can see how many votes each candidate received and who is the winner or if it is a tie. This was a huge struggle for us as we tried to use the tree widget within tkinter. We struggled to create the window through grid as we were unable to use the pack function due to several mixing errors with eachothers windows.

Third option – back button – goes back to the login page and logs out from the profile.

6. Testing

Describe the process you took to test your code and to make sure the program functions as required. Also, indicate the testing you've done after integrating code from other group members.

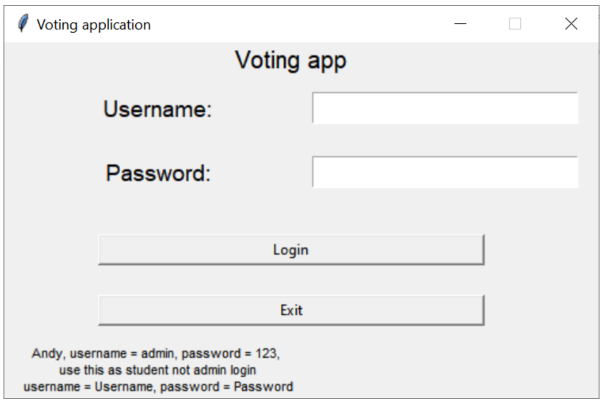
(WRITE THIS SECTION INDIVIDUALLY)

Creating GUI to add comments which then would be saved to a text file. The main struggle to save added comments to txt file. Learnt utilization of output data to save . Defining save option to save the following information written on the GUUI window to the txt file. Making class and adding radio buttons to complement the GUI window to make it more comparable with a daily used voting system. This set of code would allow the voter or admin to add comments regarding (don't fucking know who) which would be saved in the database in which our case is the text file.

7. Annotated Screenshots Demonstrating Implementation

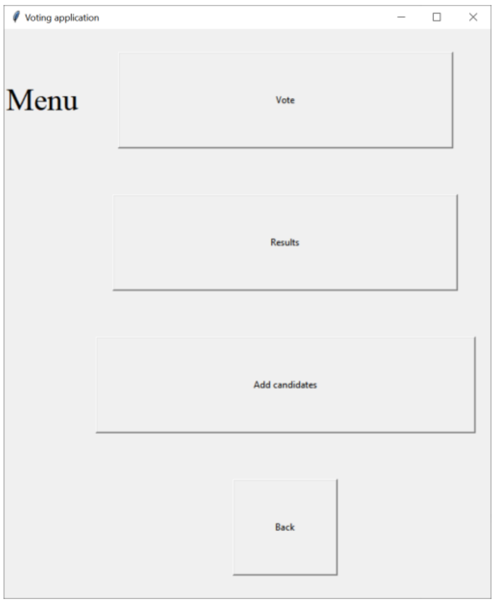
*Provide screenshots that demonstrate the features implemented. Give a brief description for **each** (**up to 100 words**) to explain the code in action. Make sure the screenshots make clear what you have implemented and achieved. (THIS SECTION SHOULD BE THE SAME FOR ALL GROUP MEMBERS)*

7.1 Feature A.1 - screenshots …

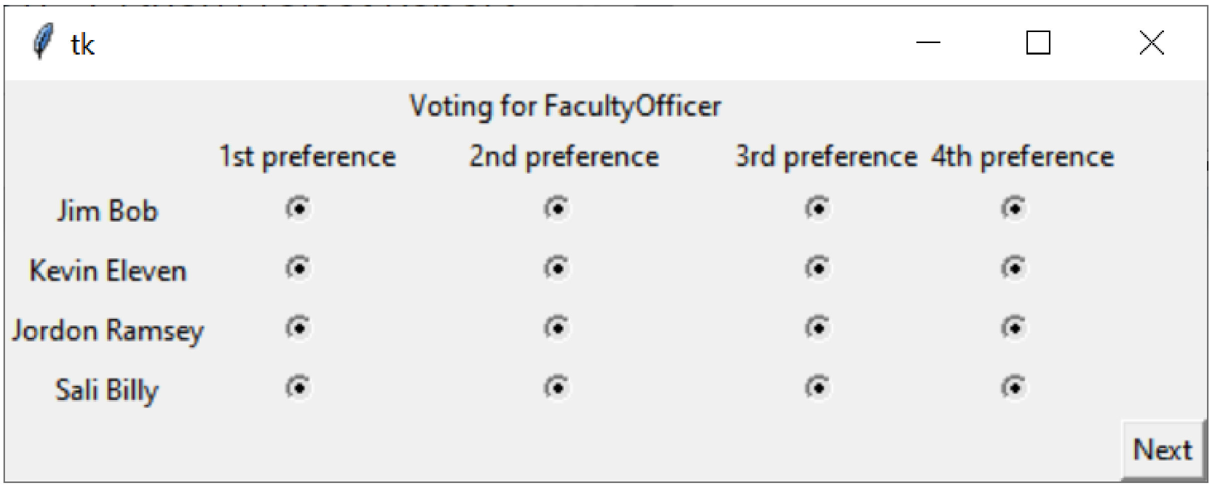


Andy, username = admin, password = 123,
use this as student not admin login
username = Username, password = Password

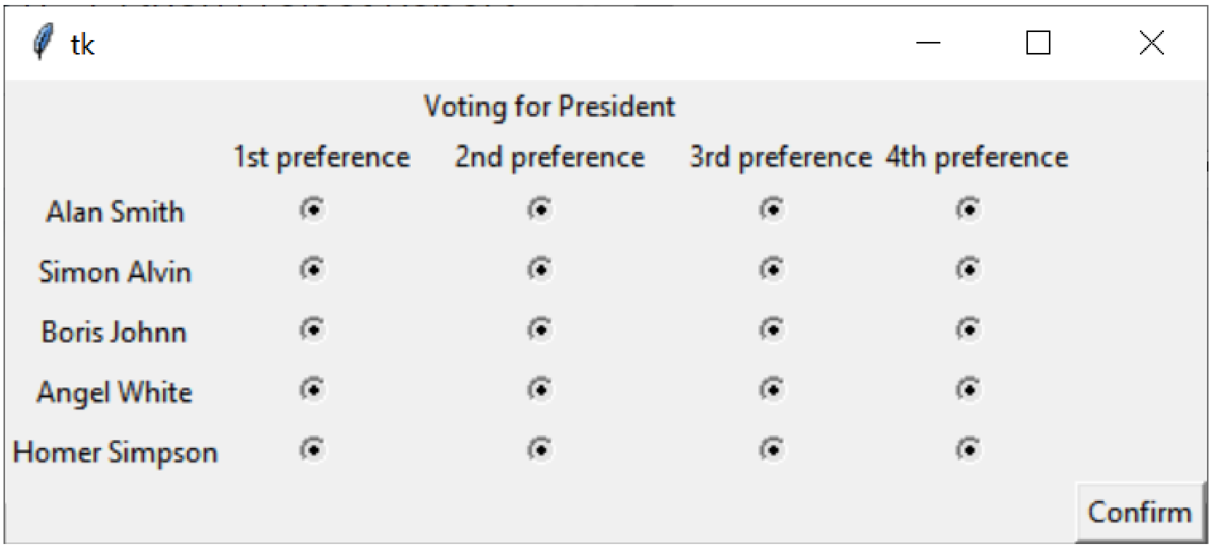
As you can see this is the interactive GUI of the Login Screen. It has a Login button when you want to login to main menu where you vote, or you can press the exit button to close the window.



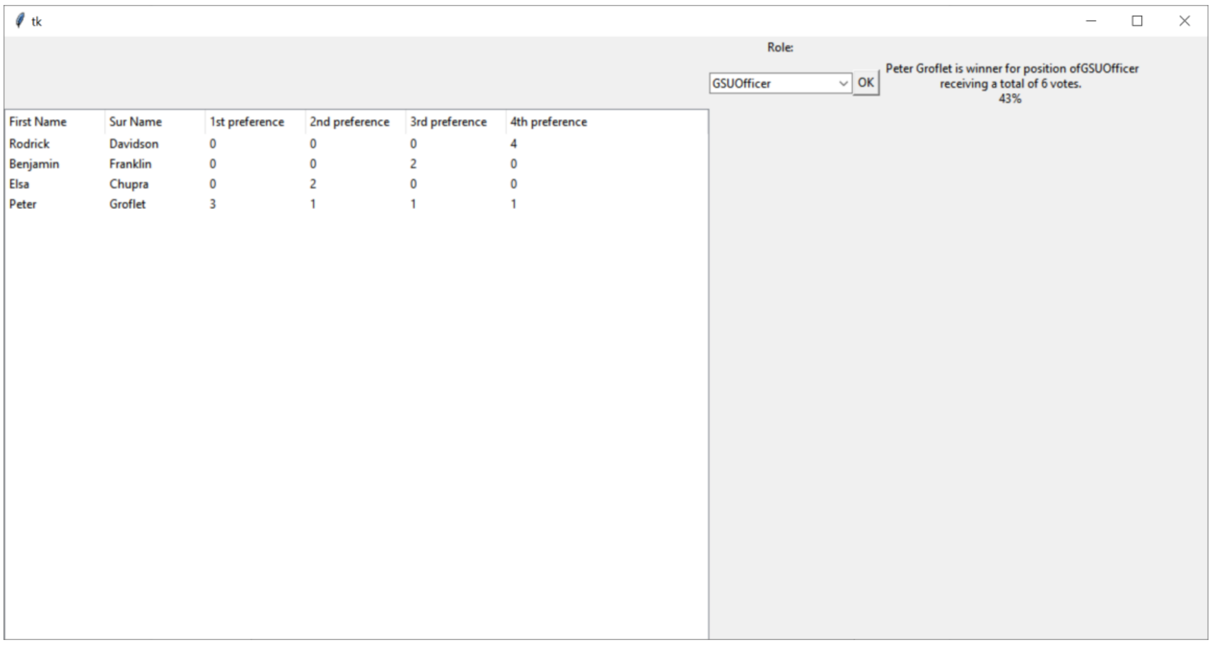
As you can see this is the main menu screen which connects all the other windows together, which allows access to the voting procedure as well as viewing the results.



This is the voting selection screen, the main part of the GUI, which takes in the users votes. Has a next button for the user to vote for the next position.



This is the last voting screen as it has a Confirm button to end the voting session and return to the menu.



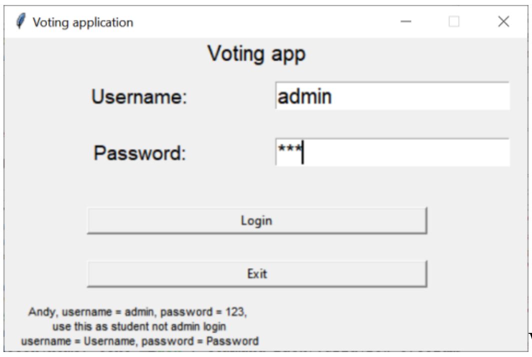
Peter Groflet is winner for position ofGSUOfficer receiving a total of 6 votes.
43%

First Name	Sur Name	1st preference	2nd preference	3rd preference	4th preference
Rodrick	Davidson	0	0	0	4
Benjamin	Franklin	0	0	2	0
Elsa	Chupra	0	2	0	0
Peter	Groflet	3	1	1	1

This is the Display window that visualises the results.

7.2 Feature A.2 - screenshots …

The images below show the outcome of the inputs you put into the login screen. As you can see the program checks whether you have entered a correct login infornmation.
The program also outputs “Incorrect password”, if the user enters a username correct but the password is wrong.


Voting application

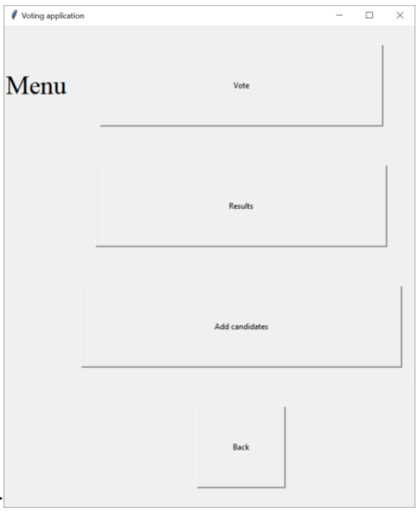
Voting app

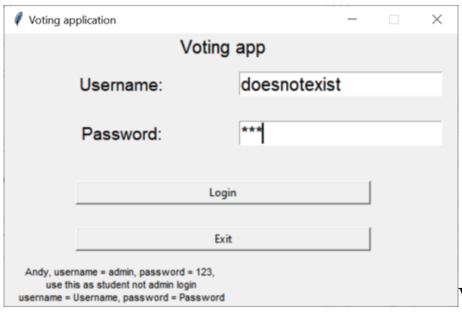
Username: admin

Password: ***

Andy, username = admin, password = 123,
use this as student not admin login
username = Username, password = Password

When entered --->


Voting application


Voting application

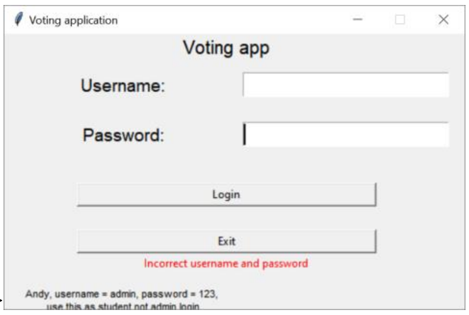
Voting app

Username: doesnotexist

Password: ***

Andy, username = admin, password = 123,
use this as student not admin login
username = Username, password = Password

When entered -->


Voting application

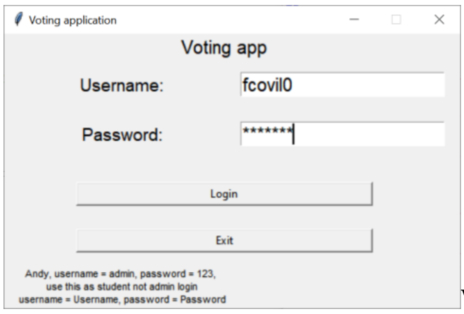
Voting app

Username:

Password:

Incorrect username and password

Andy, username = admin, password = 123,
use this as student not admin login


Voting application

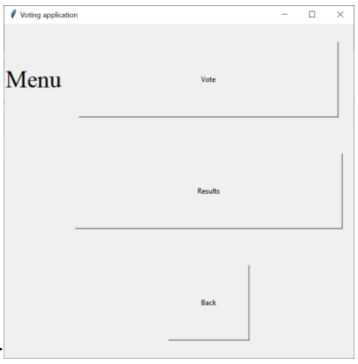
Voting app

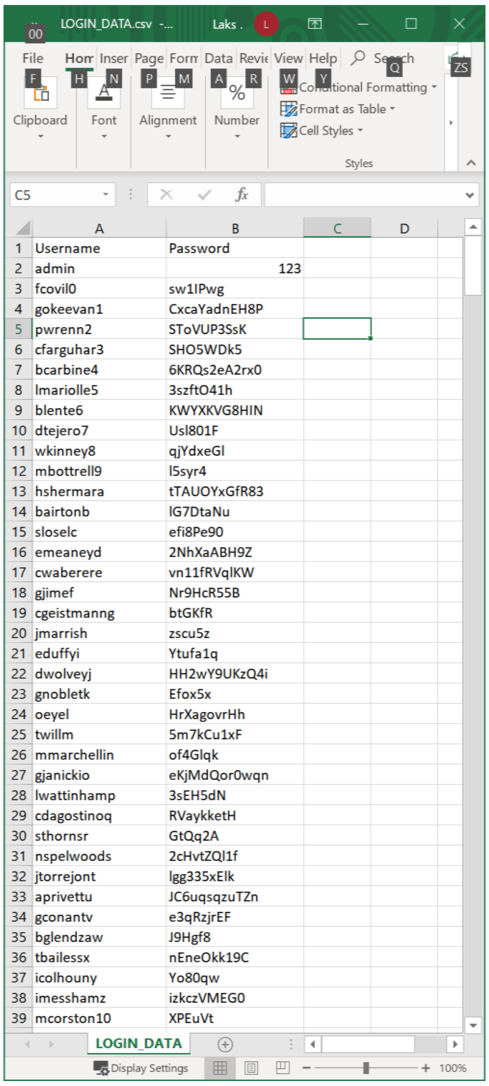
Username: fcovil0

Password: *******

Andy, username = admin, password = 123,
use this as student not admin login
username = Username, password = Password

When entered -->


Voting application

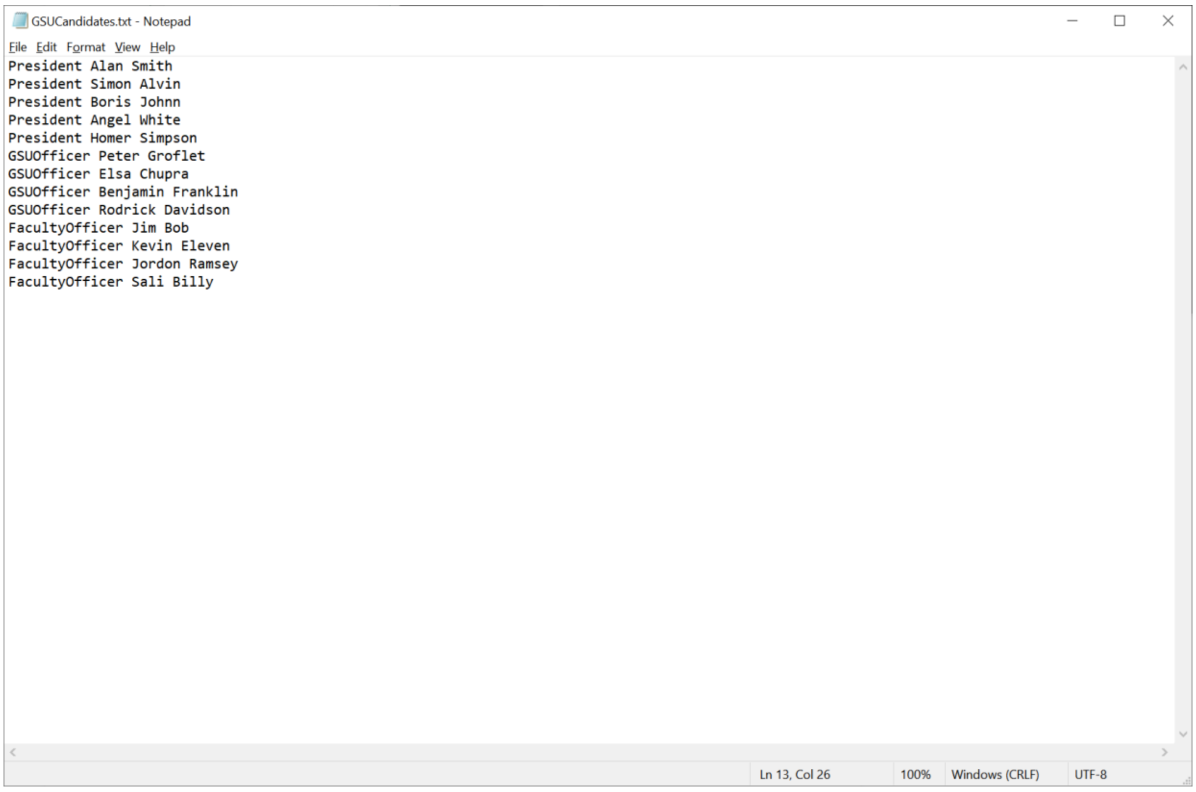


	A	B
1	Username	Password
2	admin	123
3	fcovil0	sw1IPwg
4	gokeevan1	CxcaYadnEH8P
5	pwrenn2	SToVUP3SsK
6	cfarguhar3	SHO5WDk5
7	bcarbine4	6KRQs2eA2rx0
8	lmariolle5	3szftO41h
9	blente6	KWYXKVG8HIN
10	dtejero7	Usl801F
11	wkinney8	qjYdxeGl
12	mbottrell9	l5syr4
13	hshermara	tTAUOYxGfR83
14	bairtonb	lG7DtaNu
15	sloselc	efi8Pe90
16	emeaneyd	2NhXaABH9Z
17	cwaberere	vn11fRVqlKW
18	gjimef	Nr9HcR55B
19	cgeistmanng	btGKfR
20	jmarrish	zscu5z
21	eduffyi	Ytufa1q
22	dwolveyj	HH2wY9UKzQ4i
23	gnobletk	Efox5x
24	oeyel	HrXagovrHh
25	twillm	5m7kCu1xF
26	mmarchellin	of4Glqk
27	gjanickio	eKjMdQor0wqn
28	lwattinhamp	3sEH5dN
29	cdagostinoq	RVaykketH
30	sthornsr	GtQq2A
31	nspelwoods	2cHvtZQl1f
32	jtorrejont	lgg335xElk
33	aprivettu	JC6uqsqzuTZn
34	gconantv	e3qRzjrEF
35	bglendzaw	J9Hgf8
36	tbailessx	nEneOkk19C
37	icolhouny	Yo80qw
38	imesshamz	izkczVMEG0
39	mcorston10	XPEuVt

This is all the Login data.

7.3 Feature A.3 - screenshots …

This is the GSUCandidates.txt file that is read from to create the vote window as well as the



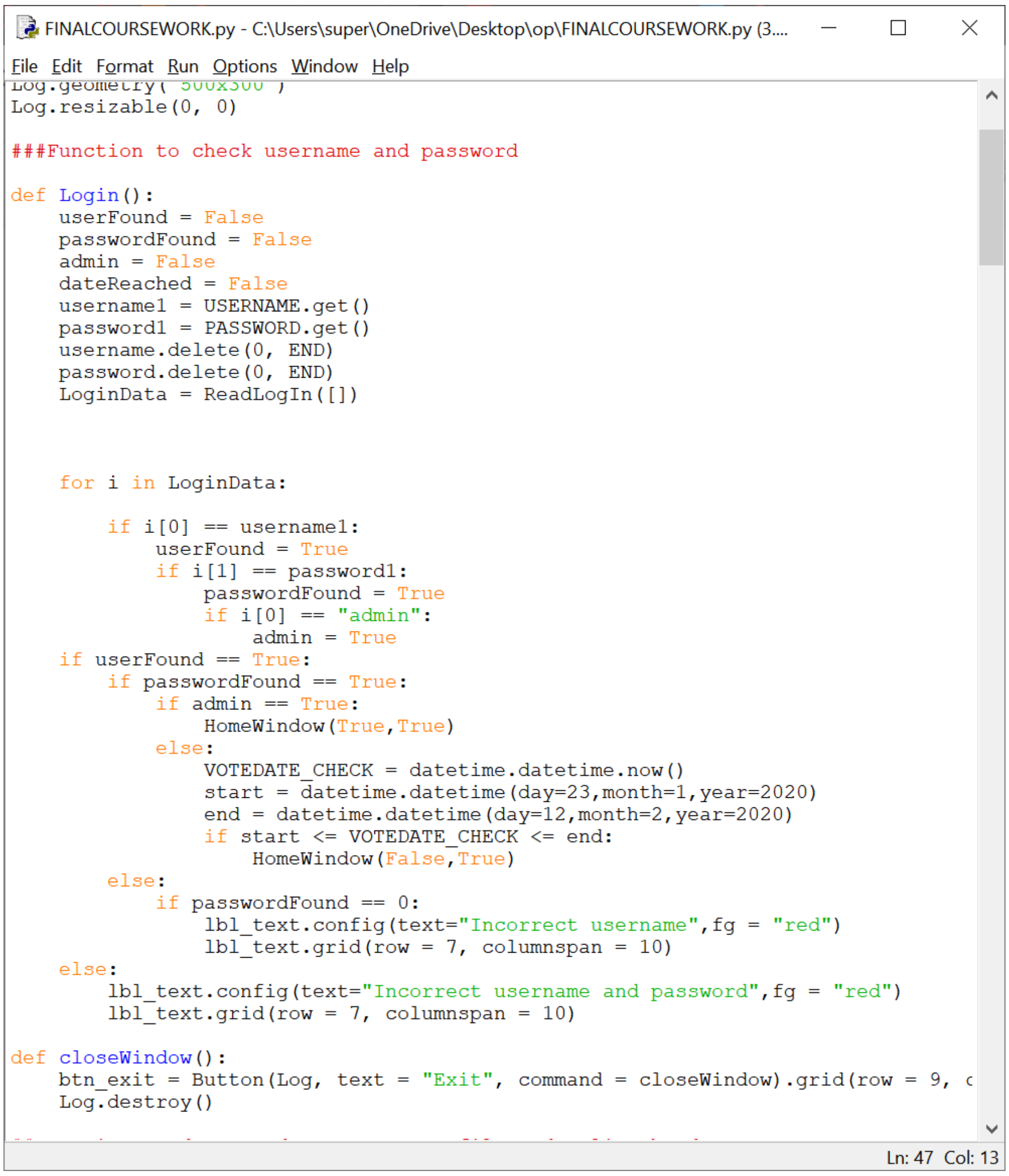
GSUCandidates.txt - Notepad

File Edit Format View Help

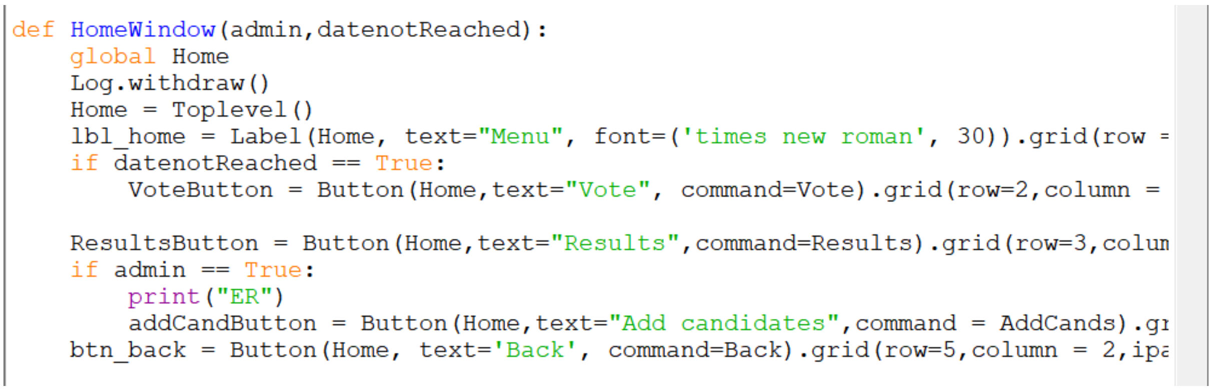
```
President Alan Smith
President Simon Alvin
President Boris Johnn
President Angel White
President Homer Simpson
GSUOfficer Peter Groflet
GSUOfficer Elsa Chupra
GSUOfficer Benjamin Franklin
GSUOfficer Rodrick Davidson
FacultyOfficer Jim Bob
FacultyOfficer Kevin Eleven
FacultyOfficer Jordon Ramsey
FacultyOfficer Sali Billy
```

Ln 13, Col 26 100% Windows (CRLF) UTF-8

7.4 Feature A.4 - screenshots …


FINALCOURSEWORK.py - C:\Users\super\OneDrive\Desktop\op\FINALCOURSEWORK.py (3....
File Edit Format Run Options Window Help
Ln: 47 Col: 13

```
Log.geometry( 500x500 )
Log.resizable(0, 0)

###Function to check username and password

def Login():
    userFound = False
    passwordFound = False
    admin = False
    dateReached = False
    username1 = USERNAME.get()
    password1 = PASSWORD.get()
    username.delete(0, END)
    password.delete(0, END)
    LoginData = ReadLogIn([])



    for i in LoginData:

        if i[0] == username1:
            userFound = True
            if i[1] == password1:
                passwordFound = True
                if i[0] == "admin":
                    admin = True
    if userFound == True:
        if passwordFound == True:
            if admin == True:
                HomeWindow(True,True)
            else:
                VOTEDATE_CHECK = datetime.datetime.now()
                start = datetime.datetime(day=23,month=1,year=2020)
                end = datetime.datetime(day=12,month=2,year=2020)
                if start <= VOTEDATE_CHECK <= end:
                    HomeWindow(False,True)
        else:
            if passwordFound == 0:
                lbl_text.config(text="Incorrect username",fg = "red")
                lbl_text.grid(row = 7, columnspan = 10)
    else:
        lbl_text.config(text="Incorrect username and password",fg = "red")
        lbl_text.grid(row = 7, columnspan = 10)

def closeWindow():
    btn_exit = Button(Log, text = "Exit", command = closeWindow).grid(row = 9, c
    Log.destroy()
```

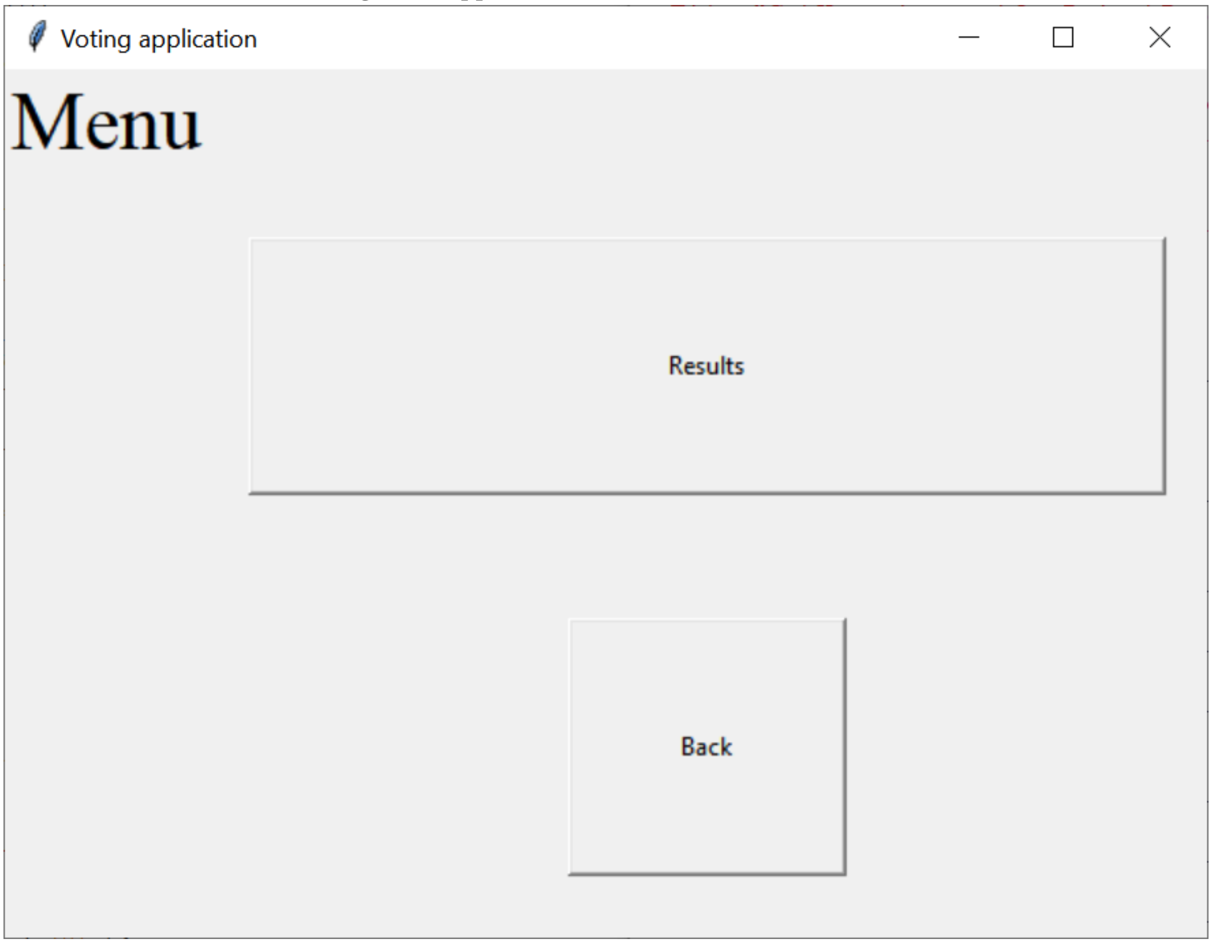


```
def HomeWindow(admin,datenotReached):
    global Home
    Log.withdraw()
    Home = Toplevel()
    lbl_home = Label(Home, text="Menu", font=('times new roman', 30)).grid(row =
    if datenotReached == True:
        VoteButton = Button(Home,text="Vote", command=Vote).grid(row=2,column =

    ResultsButton = Button(Home,text="Results",command=Results).grid(row=3,colum
    if admin == True:
        print("ER")
        addCandButton = Button(Home,text="Add candidates",command = AddCands).gr
    btn_back = Button(Home, text='Back', command=Back).grid(row=5,column = 2,ipa
```

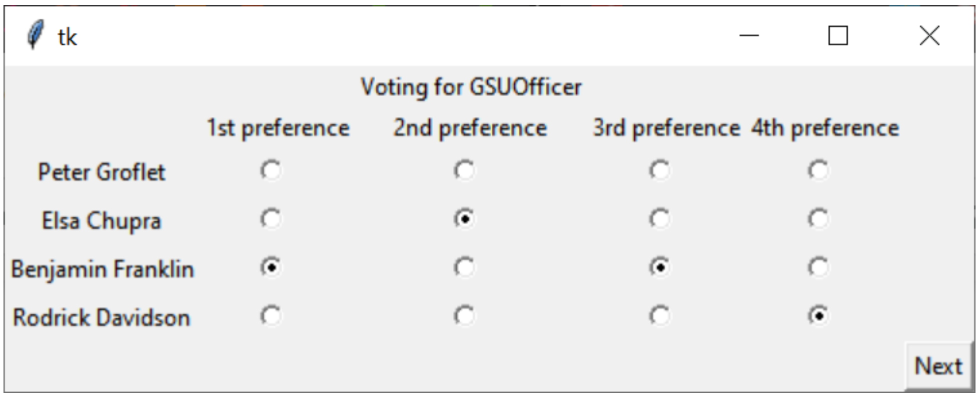
Show time

If wasnt in time:

If time was before then nothing will happen, if time was after then:



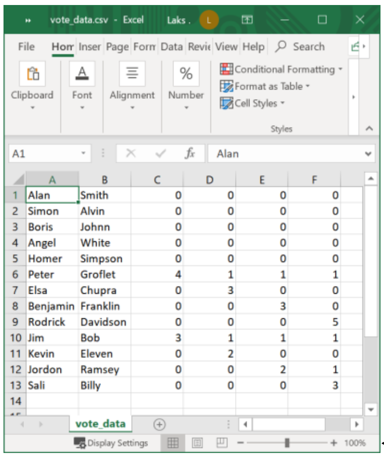
Unless you are admin then you would have access to everything.


tk

Voting for GSUOfficer

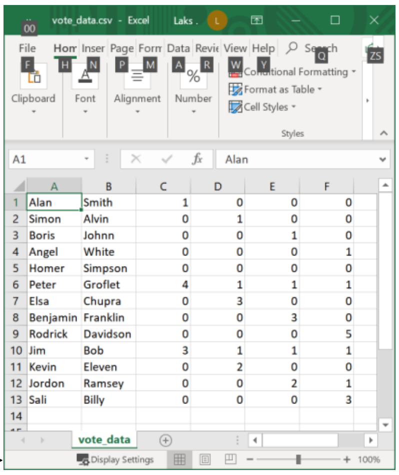
	1st preference	2nd preference	3rd preference	4th preference
Peter Groflet	○	○	○	○
Elsa Chupra	○	●	○	○
Benjamin Franklin	●	○	●	○
Rodrick Davidson	○	○	○	●

If the user enters these plots, then it is saved into csv file.



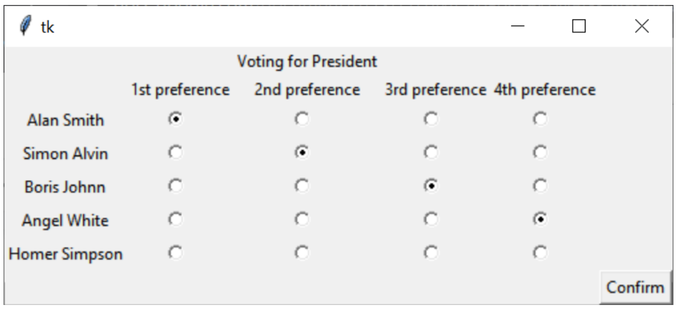
	A	B	C	D	E	F
1	Alan	Smith	0	0	0	0
2	Simon	Alvin	0	0	0	0
3	Boris	Johnn	0	0	0	0
4	Angel	White	0	0	0	0
5	Homer	Simpson	0	0	0	0
6	Peter	Groflet	4	1	1	1
7	Elsa	Chupra	0	3	0	0
8	Benjamin	Franklin	0	0	3	0
9	Rodrick	Davidson	0	0	0	5
10	Jim	Bob	3	1	1	1
11	Kevin	Eleven	0	2	0	0
12	Jordon	Ramsey	0	0	2	1
13	Sali	Billy	0	0	0	3
14						

<--Before After -->



	A	B	C	D	E	F
1	Alan	Smith	1	0	0	0
2	Simon	Alvin	0	1	0	0
3	Boris	Johnn	0	0	1	0
4	Angel	White	0	0	0	1
5	Homer	Simpson	0	0	0	0
6	Peter	Groflet	4	1	1	1
7	Elsa	Chupra	0	3	0	0
8	Benjamin	Franklin	0	0	3	0
9	Rodrick	Davidson	0	0	0	5
10	Jim	Bob	3	1	1	1
11	Kevin	Eleven	0	2	0	0
12	Jordon	Ramsey	0	0	2	1
13	Sali	Billy	0	0	0	3
14						

As you can see the candidates I selected have incremented by 1.

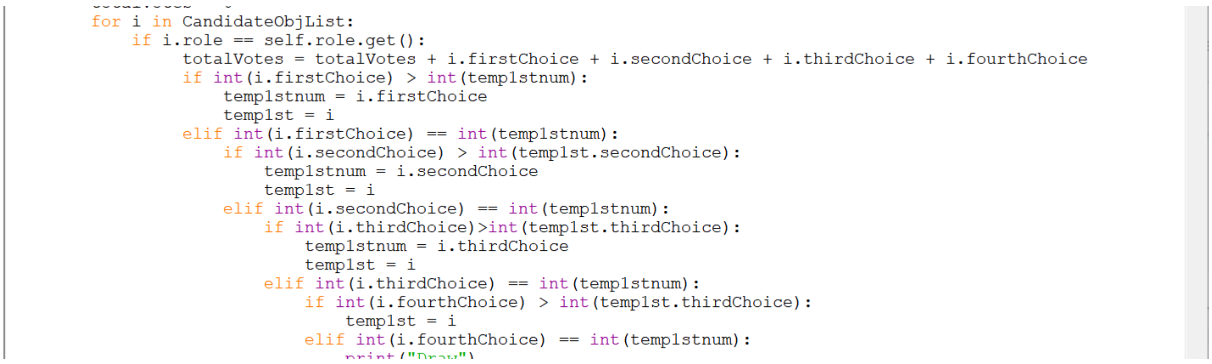


Voting for President

	1st preference	2nd preference	3rd preference	4th preference
Alan Smith	●	○	○	○
Simon Alvin	○	●	○	○
Boris Johnn	○	○	●	○
Angel White	○	○	○	●
Homer Simpson	○	○	○	○

The Gui does not allow you to select more than one option for each of the preferences. You can also see when pressing the next button lets you vote for the next GSU position.

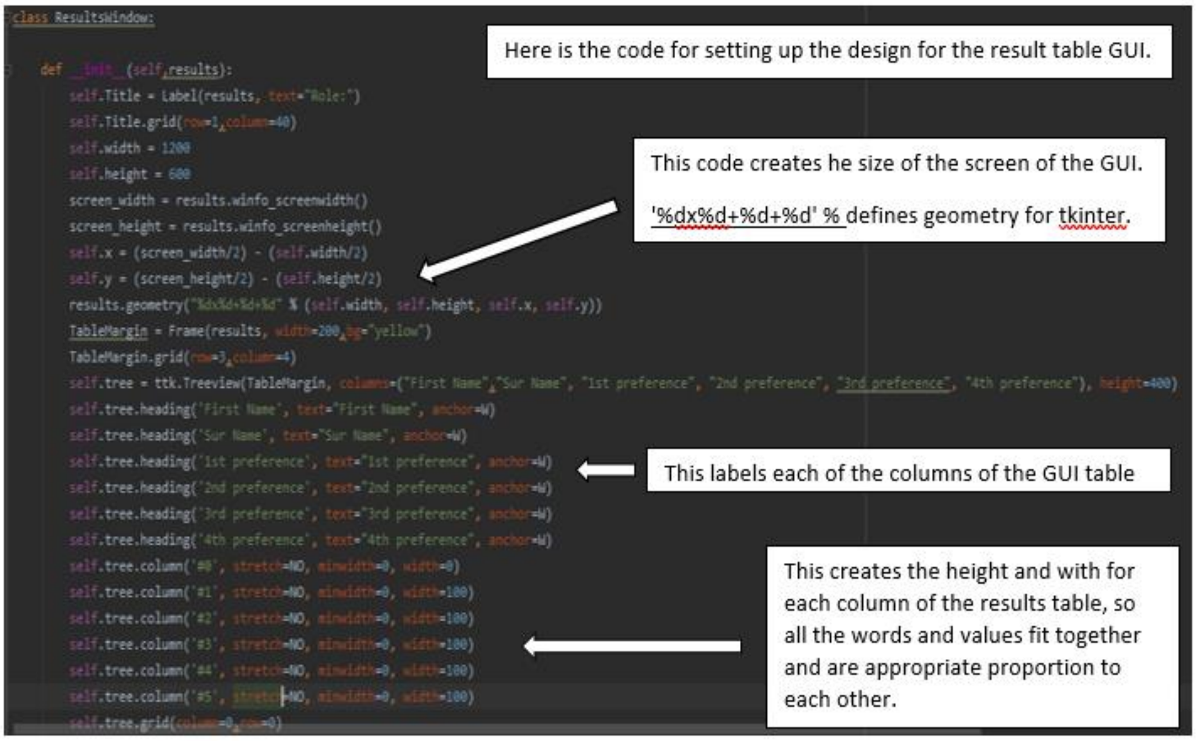
7.5 Feature A.5 - screenshots …



```
for i in CandidateObjList:
    if i.role == self.role.get():
        totalVotes = totalVotes + i.firstChoice + i.secondChoice + i.thirdChoice + i.fourthChoice
        if int(i.firstChoice) > int(templstnum):
            templstnum = i.firstChoice
            templst = i
        elif int(i.firstChoice) == int(templstnum):
            if int(i.secondChoice) > int(templst.secondChoice):
                templstnum = i.secondChoice
                templst = i
            elif int(i.secondChoice) == int(templstnum):
                if int(i.thirdChoice)>int(templst.thirdChoice):
                    templstnum = i.thirdChoice
                    templst = i
                elif int(i.thirdChoice) == int(templstnum):
                    if int(i.fourthChoice) > int(templst.thirdChoice):
                        templst = i
                    elif int(i.fourthChoice) == int(templstnum):
```

As you can see this compares all the 1st preferences until finding the highest one, if there are 2 of the highest one then it would check the second preference. This happens until the fourth preference, and if highly unlikely 2 candidates have an exact draw, then it prints draw.

7.6 Feature A.6 - screenshots …



```
class ResultsWindow:

    def __init__(self,results):
        self.Title = Label(results, text="Role:")
        self.Title.grid(row=1,column=40)
        self.width = 1200
        self.height = 600
        screen_width = results.winfo_screenwidth()
        screen_height = results.winfo_screenheight()
        self.x = (screen_width/2) - (self.width/2)
        self.y = (screen_height/2) - (self.height/2)
        results.geometry("%dx%d+%d+%d" % (self.width, self.height, self.x, self.y))
        TableMargin = Frame(results, width=200,bg="yellow")
        TableMargin.grid(row=3,column=4)
        self.tree = ttk.Treeview(TableMargin, columns=("First Name","Sur Name", "1st preference", "2nd preference", "3rd preference", "4th preference"), height=400)
        self.tree.heading('First Name', text="First Name", anchor=W)
        self.tree.heading('Sur Name', text="Sur Name", anchor=W)
        self.tree.heading('1st preference', text="1st preference", anchor=W)
        self.tree.heading('2nd preference', text="2nd preference", anchor=W)
        self.tree.heading('3rd preference', text="3rd preference", anchor=W)
        self.tree.heading('4th preference', text="4th preference", anchor=W)
        self.tree.column('#0', stretch=NO, minwidth=0, width=0)
        self.tree.column('#1', stretch=NO, minwidth=0, width=100)
        self.tree.column('#2', stretch=NO, minwidth=0, width=100)
        self.tree.column('#3', stretch=NO, minwidth=0, width=100)
        self.tree.column('#4', stretch=NO, minwidth=0, width=100)
        self.tree.column('#5', stretch=NO, minwidth=0, width=100)
        self.tree.grid(column=0,row=0)
```

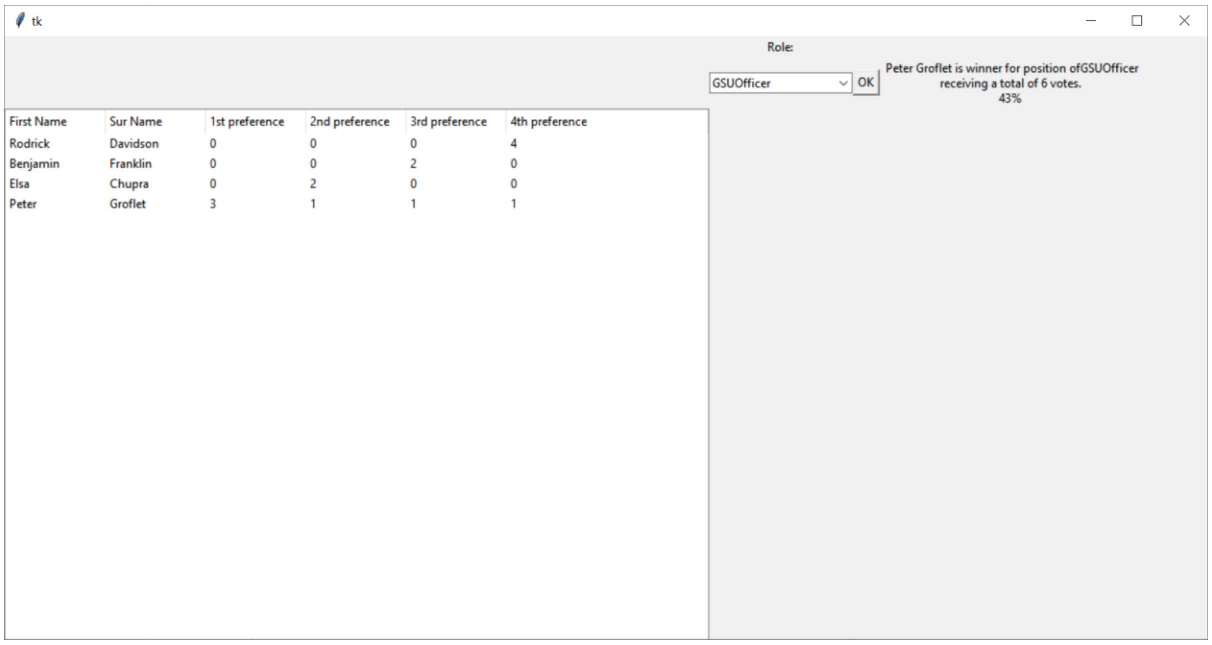
Here is the code for setting up the design for the result table GUI.

This code creates he size of the screen of the GUI.

'%dx%d+%d+%d' % defines geometry for tkinter.

This labels each of the columns of the GUI table

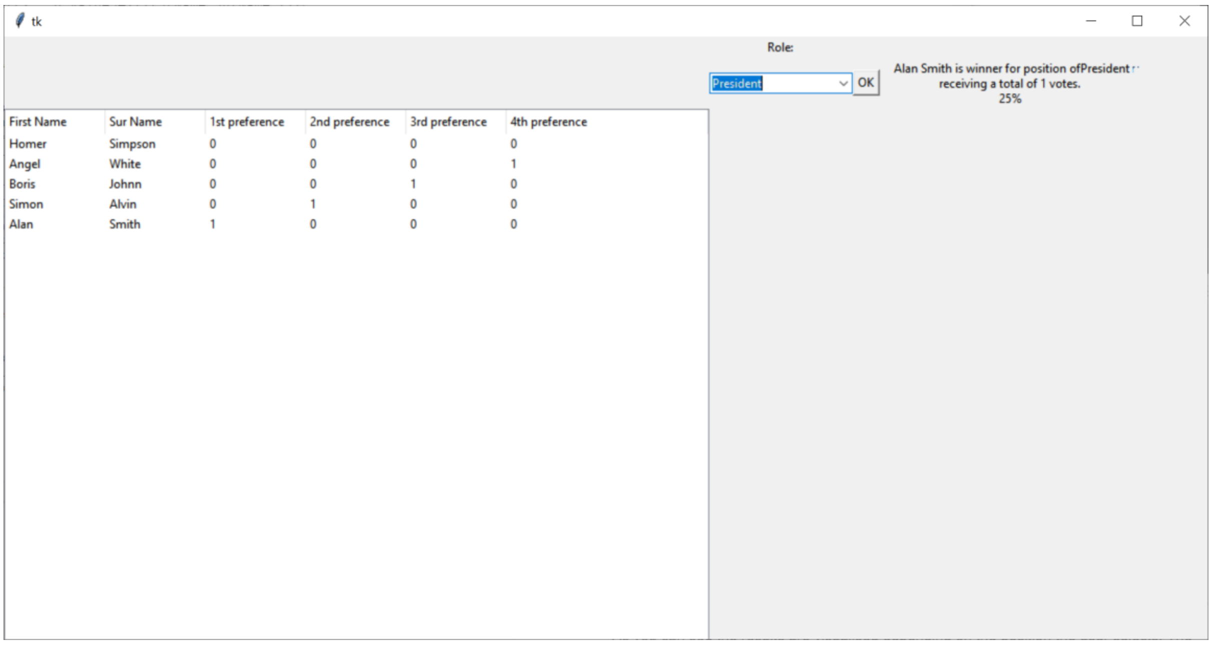
This creates the height and with for each column of the results table, so all the words and values fit together and are appropriate proportion to each other.



Peter Groflet is winner for position ofGSUOfficer receiving a total of 6 votes.
43%

First Name	Sur Name	1st preference	2nd preference	3rd preference	4th preference
Rodrick	Davidson	0	0	0	4
Benjamin	Franklin	0	0	2	0
Elsa	Chupra	0	2	0	0
Peter	Groflet	3	1	1	1

As you can see the results are visualised depending on the position the user selects. The winner label is also changed as well as the percentage of votes they received within the position selected.

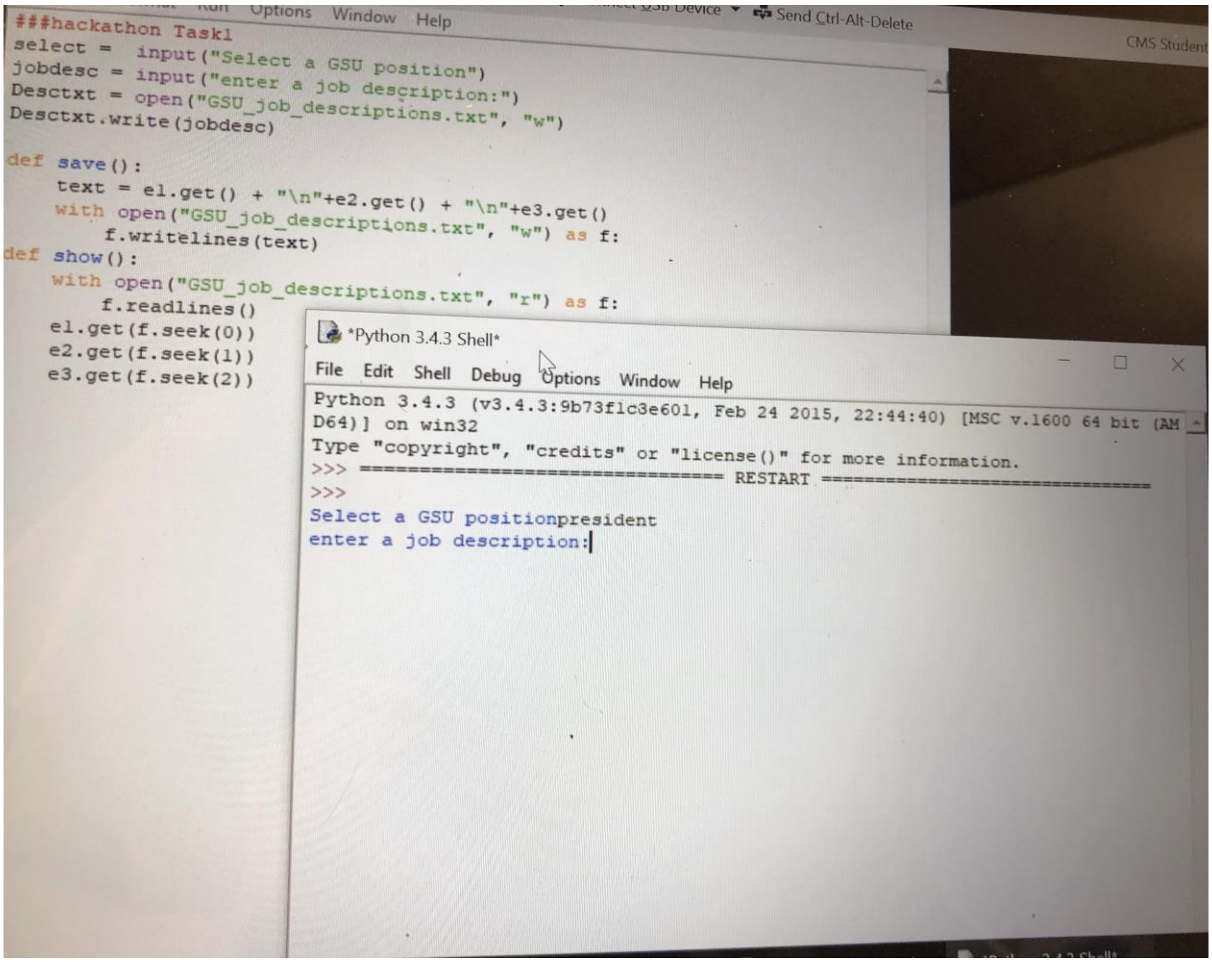


Alan Smith is winner for position ofPresident
receiving a total of 1 votes.
25%

First Name	Sur Name	1st preference	2nd preference	3rd preference	4th preference
Homer	Simpson	0	0	0	0
Angel	White	0	0	0	1
Boris	Johnn	0	0	1	0
Simon	Alvin	0	1	0	0
Alan	Smith	1	0	0	0

7.7 Feature B.1 - screenshots …

Task 1- 001086197



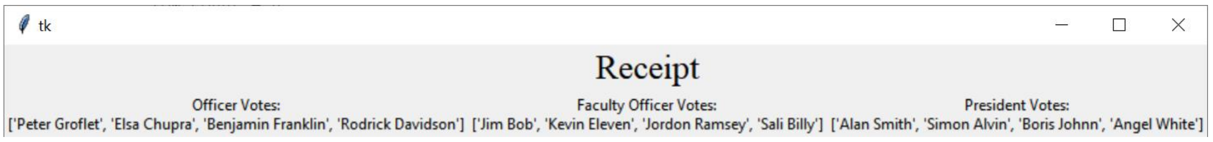
```
###hackathon Task1
select =  input("Select a GSU position")
jobdesc = input("enter a job description:")
Desctxt = open("GSU_job_descriptions.txt", "w")
Desctxt.write(jobdesc)

def save():
    text = e1.get() + "\n"+e2.get() + "\n"+e3.get()
    with open("GSU_job_descriptions.txt", "w") as f:
        f.writelines(text)
def show():
    with open("GSU_job_descriptions.txt", "r") as f:
        f.readlines()
    e1.get(f.seek(0))
    e2.get(f.seek(1))
    e3.get(f.seek(2))
```

```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:44:40) [MSC v.1600 64 bit (AM
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
Select a GSU positionpresident
enter a job description:
```

select = input("Select a GSU position")

```
jobdesc = input("enter a job description:")

Desctxt = open("GSU_job_descriptions.txt", "w")

Desctxt.write(jobdesc)



def save():

   text = e1.get() + "\n"+e2.get() + "\n"+e3.get()

   with open("GSU_job_descriptions.txt", "w") as f:

      f.writelines(text)

def show():

   with open("GSU_job_descriptions.txt", "r") as f:

      f.readlines()

   e1.get(f.seek(0))

   e2.get(f.seek(1))

   e3.get(f.seek(2))
```

7.8 Feature B.2 - screenshots …



```
Receipt
Officer Votes:
['Peter Groflet', 'Elsa Chupra', 'Benjamin Franklin', 'Rodrick Davidson']
Faculty Officer Votes:
['Jim Bob', 'Kevin Eleven', 'Jordon Ramsey', 'Sali Billy']
President Votes:
['Alan Smith', 'Simon Alvin', 'Boris Johnn', 'Angel White']
```

```
self.Choices = []
```

<u>(from line 196)</u>

```
self.Choices.append([pref1,pref2,pref3,pref4])
```

<u>(from line 243)</u>

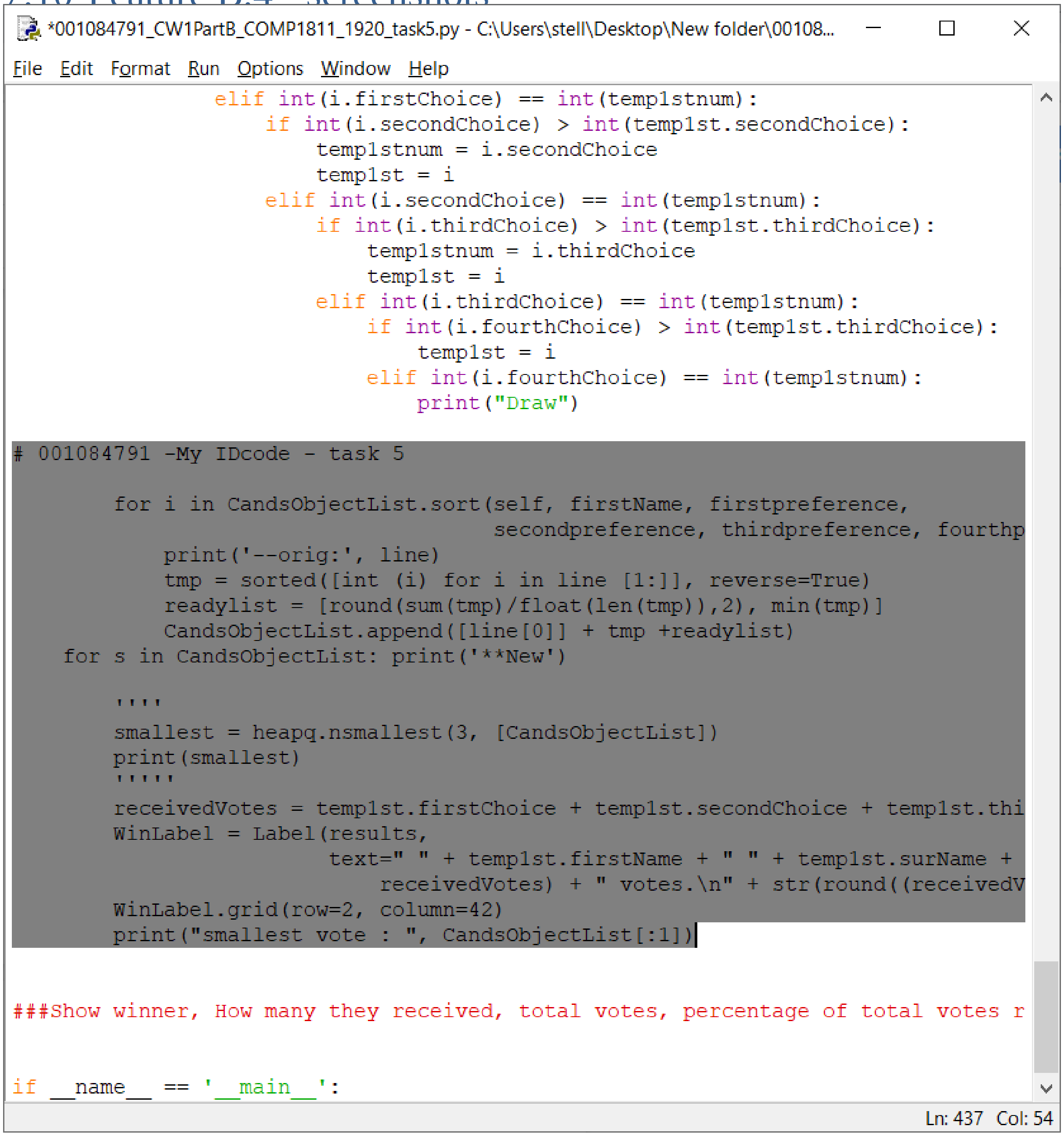
```
lbl_Title = Label(Receipt,text="Receipt", font=('times new roman',20)).grid(row=1,column=2)

        lbl_Officer = Label(Receipt,text="Officer Votes:\n"+str(self.Choices[0])).grid(row=2,column=1)

        lbl_Faculty = Label(Receipt,text="Faculty Officer
Votes:\n"+str(self.Choices[1])).grid(row=2,column=2)
```

```
        lbl_President = Label(Receipt,text="President
Votes:\n"+str(self.Choices[2])).grid(row=2,column=3)

        Receipt.mainloop()
```

(from line 320 to 324)

7.9 Feature B.3 - screenshots …

7.10 Feature B.4 - screenshots


*001084791_CW1PartB_COMP1811_1920_task5.py - C:\Users\stell\Desktop\New folder\00108...
File Edit Format Run Options Window Help
Ln: 437 Col: 54

```
                elif int(i.firstChoice) == int(templstnum):
                    if int(i.secondChoice) > int(templst.secondChoice):
                        templstnum = i.secondChoice
                        templst = i
                    elif int(i.secondChoice) == int(templstnum):
                        if int(i.thirdChoice) > int(templst.thirdChoice):
                            templstnum = i.thirdChoice
                            templst = i
                        elif int(i.thirdChoice) == int(templstnum):
                            if int(i.fourthChoice) > int(templst.thirdChoice):
                                templst = i
                            elif int(i.fourthChoice) == int(templstnum):
                                print("Draw")

# 001084791 -My IDcode - task 5

        for i in CandsObjectList.sort(self, firstName, firstpreference,
                                      secondpreference, thirdpreference, fourthp
            print('--orig:', line)
            tmp = sorted([int (i) for i in line [1:]], reverse=True)
            readylist = [round(sum(tmp)/float(len(tmp)),2), min(tmp)]
            CandsObjectList.append([line[0]] + tmp +readylist)
    for s in CandsObjectList: print('**New')

        ''''
        smallest = heapq.nsmallest(3, [CandsObjectList])
        print(smallest)
        '''''
        receivedVotes = templst.firstChoice + templst.secondChoice + templst.thi
        WinLabel = Label(results,
                         text=" " + templst.firstName + " " + templst.surName +
                             receivedVotes) + " votes.\n" + str(round((receivedV
        WinLabel.grid(row=2, column=42)
        print("smallest vote : ", CandsObjectList[:1])


###Show winner, How many they received, total votes, percentage of total votes r


if __name__ == '__main__':
```

8. Evaluation

*Give a summary of your implementation and discuss what you would do if you had more time to work on the project. Answer the following questions for the reflection and write **350-400 words overall**. Please include an actual word count for this section. (WRITE THIS S ECTION INDIVIDUALLY)*

8.1 Evaluate how well your design and implementation meet the requirements

I was successful in creating the interactive GUI with a functional display of Radio buttons and given information used precisely according to the objective. In order to make the vote screen classes were used called as VoterWindow, ApplyRole, ResultWindow and all the data was imported from a csv file. Radio button includes all the candidates and voters can vote according to preferences of four. I was also able to successfully get the voters result output to a text file.

8.2 Evaluate you own and your group's performance

Each member of the group was assigned a task. This allowed all the group members to contribute easily. If any member faced difficulties, we would discuss as a group and eventually come up with a solution for it. I was really lucky with the group I had. Every member was responsible and did their best to complete the coursework

8.2.1 What went well?

According to me, display created with working radio buttons was done properly. I faced difficulty to output results saved in text file as per the objective but eventually could do it. I was also successful in using new python functions which broadened my knowledge of writing code and understanding in each step. Group cooperation and time management for the coursework was very efficient.

8.2.2 What went less well?

It was difficult to read the csv file and integrating with GUI. Saving the results of votes to text file was also a challenging task as it was very new to all of us. Although we managed to do it and now have a better understanding of the way it works.

8.2.3 What was learnt?

Overall I learnt new functions, improved logic while writing code. I now have a better knowledge of GUI and Tkinter. I learnt how to read and write csv file and make use of it in the code. I believe this task will be really beneficial for me in future.

8.2.4 How would a similar task be completed differently?

As I have better understanding about the task so I can organize it in a more efficient way as well as I can finish it well ahead of submission. I would like to add more widgets and more graphics to how the GUI would look after the coding and implementation. I can also use more classes and functions in the code.

8.2.5 How could the module be improved?

The module can teach in more detail about tkinter and GUI in particular. Also, we can have separate slides about text and csv file to understand the concept and use of it in ore depth.

Group Pro forma

Describe the division of work and agree percentage contributions. The pro forma must be signed by all group members and an identical copy provided in each report. If you cannot agree percentage contributions, please indicate so in the notes column and provide your reasoning.

(THIS SECTION SHOULD BE THE SAME FOR ALL GROUP MEMBERS)

Group Member ID	Tasks/Features Completed	%Contribution	Signature	Notes
Member 1	Main Menu	25%	Zarin Tasnim	
Member 2	Login screen	25%	Karina Busmane	
Member 3	Results Screen	25%	Mahdi Rahman	
Member 4	Vote Screen	25%	Lakshman Thillainathan	
	Total	100%		

Appendix A: Code Listing

*Provide a complete listing of all the *.py files in your PyCharm project. Make sure your code is* ***well commented*** *and applies professional Python convention (refer to [PEP 8](PEP 8) for details). The code listed here must match that uploaded to Moodle. Please copy and paste the actual code – no screenshots please! (THIS SECTION SHOULD BE THE SAME FOR ALL GROUP MEMBERS)*

```
# import libraries so we could use their functions


from tkinter import *

import copy, csv, ctypes, datetime

import tkinter.ttk as ttk


global username, password


# Log in Window


Log = Tk()

Log.title("Voting application")

Log.geometry("500x300")

Log.resizable(0, 0)


# Function to get username and password from user input

def Login():

    userFound = False

    passwordFound = False

    admin = False

    dateReached = False

    username1 = USERNAME.get()
```

```
password1 = PASSWORD.get()

username.delete(0, END)

password.delete(0, END)

LoginData = ReadLogIn([])


# here it compares user input to saved usernames and passwords from

# csv files


for i in LoginData:

    if i[0] == username1:

        userFound = True

        if i[1] == password1:

            passwordFound = True

            if i[0] == "admin":

                admin = True

# reads if it is admin or student so could open the right window

# admin has more opportunities to do with files

# manually confirms if it is the right date to vote

if userFound == True:

    if passwordFound == True:

        if admin == True:

            HomeWindow(True, True)

        else:

            VOTEDATE_CHECK = datetime.datetime.now()

            start = datetime.datetime(day=23, month=1, year=2020)

            end = datetime.datetime(day=12, month=2, year=2020)
            if start <= VOTEDATE_CHECK <= end:
                HomeWindow(False, True)
        else:
            if passwordFound == 0:
                lbl_text.config(text="Incorrect username", fg="red")
                lbl_text.grid(row=7, columnspan=10)
    else:
        lbl_text.config(text="Incorrect username and password", fg="red")
        lbl_text.grid(row=7, columnspan=10)



# function for exiting this particular application
def closeWindow():
    btn_exit = Button(Log, text="Exit", command=closeWindow).grid(row=9, columnspan=12)
    Log.destroy()


# Function used to read LOG_DATA.csv file and split the data
# in to username and password for login details


def ReadLogIn(data):
    with open("LOGIN_DATA.csv") as f:
        LoginData = csv.reader(f)
        for row in LoginData:
            data.append(row)
    return data
```

```
# Function to open Main Menu window


def HomeWindow(admin,datenotReached):
    global Home
    Log.withdraw()
    Home = Toplevel()
    lbl_home = Label(Home, text="Menu", font=('times new roman', 30)).grid(row =2)
    if datenotReached == True:
        # adds buttons for menu page so we could move to the next window, using command and grid to
tell location
        VoteButton = Button(Home,text="Vote", command=Vote).grid(row=2,column =
2,ipadx=200,ipady= 50,padx=20,pady=30)
        ResultsButton = Button(Home,text="Results",command=Results).grid(row=3,column =
2,ipadx=200,ipady = 50, padx =20, pady =30)
    # lets add candidates only for admin
    if admin == True:
        print("ER")
        addCandButton = Button(Home,text="Add candidates",command =
AddCands).grid(row=4,column=2,ipadx=200,ipady=50,padx=20,pady=30)
    btn_back = Button(Home, text='Back', command=Back).grid(row=5,column = 2,ipadx=50,ipady
= 50,padx=25,pady = 30)


# Function to open Vote window
def Vote():
    global root
    Home.withdraw()
    print("VOTE")
    root = Tk()
    Vote = VoteWindow(root)
    root.mainloop()



# Function to go to results screen
def Results():
    global results
    Home.withdraw()
    results = Tk()
    Results = ResultsWindow(results)
    results.mainloop()

# Function to add candidates for each position with name and surname input from admin
# writes new candidates into csv file as well

def AddCands():
    position = input("What position would you like to enroll for?")
    firstName = input("What is your first name")
    surName = input("What is your sur name")
    with open('GSUCandidates.txt', "a") as txtfile:
        txtfile.write("\n" + position + " " + firstName + " " + " " + surName)
    row_contents = [position, firstName, surName, 0, 0, 0, 0]
    with open('vote_data.csv', 'a')as csvfile:
        csvfile.write(row_contents)

def Back():
    Home.destroy()
    Log.deiconify()


# Variables

USERNAME = StringVar()

PASSWORD = StringVar()


# Frames

Top = Frame(Log, width=200, height=150)

Top.grid(row=0, column=0)

Form = Frame(Log, width=200, height=200)

Form.grid(row=1, column=0)


# Labels

lbl_title = Label(Top, text="Voting app", font=('arial', 15))

lbl_title.grid(row=0, sticky="n")

lbl_username = Label(Form, text="Username:", font=('arial', 14), bd=15)

lbl_username.grid(row=1, columnspan=1, padx=50)

lbl_password = Label(Form, text="Password:", font=('arial', 14), bd=15)

lbl_password.grid(row=2, columnspan=1, padx=50)

lbl_andy = Label(Form, text="Andy, username = admin, password = 123, "

                          "\nuse this as student not admin login"

                          "\nusername = Username, password = Password"

                 , font=('arial', 8), bd=15)

lbl_andy.grid(row=10, column=0)

lbl_text = Label(Form)

lbl_text.grid(row=3, columnspan=2)
```

```
# Entry widgets

username = Entry(Form, textvariable=USERNAME, font=(14))

username.grid(row=1, column=1)

password = Entry(Form, textvariable=PASSWORD, show="*", font=(14))

password.grid(row=2, column=1)


# Button widgets

btn_login = Button(Form, text="Login", width=45, command=Login)

btn_login.grid(pady=25, row=3, columnspan=2)

btn_login.bind('<Return>', Login)

btn_exit = Button(Form, text="Exit", width=45, command=closeWindow)

btn_exit.grid(padx=50, row=5, columnspan=12)


# Create the list of the Candidates

CandsList = []

CandsObjectList = []

Cands = open("GSUCandidates.txt")


for line in Cands:

    temp = [x.strip("\n") for x in line.split(" ")]

    CandsList.append(temp)


# Class for Candidates

class ApplyRole:

    def __init__(self, firstName, surName, role):
        # print("Called for ",firstName, surName)
        self.firstName = firstName
        self.surName = surName
        self.role = role
        with open('vote_data.csv', mode='r') as vData:
            csv_reader = csv.reader(vData, delimiter=',')
            for row in csv_reader:
                if self.firstName == row[0]:
                    self.firstChoice = int(row[2])
                    self.secondChoice = int(row[3])
                    self.thirdChoice = int(row[4])
                    self.fourthChoice = int(row[5])


# Main Voting window
class VoteWindow:


    def __init__(self, root):
        if root == None:
            self.CandidateObjectList = self.CreateObjList()
        else:
            self.roleCount = 0
            self.roleList = ["GSUOfficer", "FacultyOfficer", "President"]
            self.Title = Label(root, text="Voting for " + self.roleList[self.roleCount])
            self.Title.grid(row=0, column=2)
            self.frame = Frame(root)
            FirstPref = Label(root, text="1st preference").grid(row=1, column=1)
        SecondPref = Label(root, text="2nd preference").grid(row=1, column=2)

        ThirdPref = Label(root, text="3rd preference").grid(row=1, column=3)

        FourthPref = Label(root, text="4th preference").grid(row=1, column=4)

        self.CandidateObjectList = self.CreateObjList()

        self.selectionScreen(root, self.roleCount)


    def selectionScreen(self, root, roleCount):

        rowNum = 2

        num = 0

        self.posOfCand = []

        pref1 = StringVar(root)

        pref2 = StringVar(root)

        pref3 = StringVar(root)

        pref4 = StringVar(root)

        self.TempRows = copy.deepcopy(self.CandidateObjectList)

        roleList = ["GSUOfficer", "FacultyOfficer", "President"]

        # Adds GUI for vote select screen, radio buttons for preferences

        for i in self.TempRows:

            if i.role == roleList[self.roleCount]:

                TempText = str(i.firstName + " " + i.surName)

                self.TempRows[num] = Label(root, text=TempText)

                self.TempRows[num].grid(row=rowNum)

                Radiobutton(root, variable=pref1, value=TempText).grid(row=rowNum, column=1)

                Radiobutton(root, variable=pref2, value=TempText).grid(row=rowNum, column=2)

                Radiobutton(root, variable=pref3, value=TempText).grid(row=rowNum, column=3)

                Radiobutton(root, variable=pref4, value=TempText).grid(row=rowNum, column=4)
                rowNum += 1

                self.posOfCand.append(num)

            num += 1

        # reads information from radiobuttons

        if self.roleCount == 2:

            self.confirmButton = Button(root, text="Confirm",

                                        command=lambda: self.clicked(pref1.get(), pref2.get(), pref3.get(),
pref4.get(),

                                                                     True))

            self.confirmButton.grid(row=rowNum, column=5)

        else:

            self.confirmButton = Button(root, text="Next",

                                        command=lambda: self.clicked(pref1.get(), pref2.get(), pref3.get(),
pref4.get(),

                                                                     False))

            self.confirmButton.grid(row=rowNum, column=5)



    def clicked(self, pref1, pref2, pref3, pref4, confirmed):

        self.confirmButton.destroy()

        for i in self.posOfCand:

            self.TempRows[i].destroy()

        for i in self.CandidateObjectList:

            if i.firstName and i.surName in pref1:

                i.firstChoice += 1

                f = open('vote_data.csv', 'r')

                reader = csv.reader(f)

                mylist = list(reader)
    f.close()
    row_count = 0
    for row in mylist:
        if row[0] == i.firstName:
            print(i.firstName)
            mylist[row_count][2] = i.firstChoice
        else:
            row_count += 1
    # writes data from radiobuttons to vote_data.csv file for each preference
    my_new_list = open("vote_data.csv", "w", newline='')
    csv_writer = csv.writer(my_new_list)
    csv_writer.writerows(mylist)
    my_new_list.close()
if i.firstName and i.surName in pref2:
    i.secondChoice += 1
    f = open('vote_data.csv', 'r')
    reader = csv.reader(f)
    mylist = list(reader)
    f.close()
    row_count = 0
    for row in mylist:
        if row[0] == i.firstName:
            mylist[row_count][3] = i.secondChoice
        else:
            row_count += 1
    my_new_list = open("vote_data.csv", "w", newline='')
    csv_writer = csv.writer(my_new_list)

    csv_writer.writerows(mylist)

    my_new_list.close()

if i.firstName and i.surName in pref3:

    i.thirdChoice += 1

    f = open('vote_data.csv', 'r')

    reader = csv.reader(f)

    mylist = list(reader)

    f.close()

    row_count = 0

    for row in mylist:

        if row[0] == i.firstName:

            mylist[row_count][4] = i.thirdChoice

        else:

            row_count += 1

    my_new_list = open("vote_data.csv", "w", newline='')

    csv_writer = csv.writer(my_new_list)

    csv_writer.writerows(mylist)

    my_new_list.close()

if i.firstName and i.surName in pref4:

    i.fourthChoice += 1

    f = open('vote_data.csv', 'r')

    reader = csv.reader(f)

    mylist = list(reader)

    f.close()

    row_count = 0
```

```
                for row in mylist:
                    if row[0] == i.firstName:
                        print(i.firstName)
                        mylist[row_count][5] = i.fourthChoice
                    else:
                        row_count += 1
                my_new_list = open("vote_data.csv", "w", newline='')
                csv_writer = csv.writer(my_new_list)
                csv_writer.writerows(mylist)
                my_new_list.close()
            if confirmed == True:
                root.destroy()
                Home.deiconify()
            else:
                self.roleCount += 1
                self.Title.config(text="Voting for " + self.roleList[self.roleCount])
                self.selectionScreen(root, self.roleCount)


        def CreateObjList(self):
            if not CandsObjectList:
                for i in CandsList:
                    CandsObjectList.append(str(i[1]) + str(i[2]))
                    CandsObjectList[-1] = ApplyRole(i[1], i[2], i[0])
            return CandsObjectList


# this class to show results of votes
```

```
class ResultsWindow:

    def __init__(self, results):

        self.Title = Label(results, text="Role:")

        self.Title.grid(row=1, column=40)

        self.width = 1200

        self.height = 600

        screen_width = results.winfo_screenwidth()

        screen_height = results.winfo_screenheight()

        self.x = (screen_width / 2) - (self.width / 2)

        self.y = (screen_height / 2) - (self.height / 2)

        results.geometry("%dx%d+%d+%d" % (self.width, self.height, self.x, self.y))

        TableMargin = Frame(results, width=200, bg="yellow")

        TableMargin.grid(row=3, column=4)

        self.tree = ttk.Treeview(TableMargin, columns=(

            "First Name", "Sur Name", "1st preference", "2nd preference", "3rd preference", "4th
preference"),

                                height=400)

        self.tree.heading('First Name', text="First Name", anchor=W)

        self.tree.heading('Sur Name', text="Sur Name", anchor=W)

        self.tree.heading('1st preference', text="1st preference", anchor=W)

        self.tree.heading('2nd preference', text="2nd preference", anchor=W)

        self.tree.heading('3rd preference', text="3rd preference", anchor=W)

        self.tree.heading('4th preference', text="4th preference", anchor=W)

        self.tree.column('#0', stretch=NO, minwidth=0, width=0)

        self.tree.column('#1', stretch=NO, minwidth=0, width=100)

        self.tree.column('#2', stretch=NO, minwidth=0, width=100)

        self.tree.column('#3', stretch=NO, minwidth=0, width=100)
        self.tree.column('#4', stretch=NO, minwidth=0, width=100)

        self.tree.column('#5', stretch=NO, minwidth=0, width=100)

        self.tree.grid(column=0, row=0)


        self.role = StringVar(results)

        roleCombo = ttk.Combobox(results, textvariable=self.role,

                                values=[

                                    "President",

                                    "GSUOfficer",

                                    "FacultyOfficer"])

        roleCombo.grid(column=40, row=2)

        roleCombo.current(1)

        roleCombo.bind("<<ComboboxSelected>>", self.limp())

        okbutton = Button(results, text="OK", command=self.limp)

        okbutton.grid(column=41, row=2)


    def limp(self):

        self.tree.delete(*self.tree.get_children())

        GetList = VoteWindow(None)

        CandidateObjList = GetList.CreateObjList()

        for i in CandidateObjList:

            if i.role == self.role.get():

                firstName = i.firstName

                surName = i.surName

                firstpreference = i.firstChoice

                secondpreference = i.secondChoice
            thirdpreference = i.thirdChoice

            fourthpreference = i.fourthChoice

            self.tree.insert("", 0, values=(

                firstName, surName, firstpreference, secondpreference, thirdpreference,
    fourthpreference))

        temp1st = 0

        temp1stnum = 0

        totalVotes = 0

        for i in CandidateObjList:

          if i.role == self.role.get():

            totalVotes = totalVotes + i.firstChoice + i.secondChoice + i.thirdChoice + i.fourthChoice

            if int(i.firstChoice) > int(temp1stnum):

              temp1stnum = i.firstChoice

              temp1st = i

            elif int(i.firstChoice) == int(temp1stnum):

              if int(i.secondChoice) > int(temp1st.secondChoice):

                temp1stnum = i.secondChoice

                temp1st = i

              elif int(i.secondChoice) == int(temp1stnum):

                if int(i.thirdChoice) > int(temp1st.thirdChoice):

                  temp1stnum = i.thirdChoice

                  temp1st = i

                elif int(i.thirdChoice) == int(temp1stnum):

                  if int(i.fourthChoice) > int(temp1st.thirdChoice):

                    temp1st = i

                  elif int(i.fourthChoice) == int(temp1stnum):

                    print("Draw")
        # Show winner, How many they received, total votes, percentage of total votes received(total of
winner/total votes)

        receivedVotes = temp1st.firstChoice + temp1st.secondChoice + temp1st.thirdChoice +
temp1st.fourthChoice

        WinLabel = Label(results,

                         text=" " + temp1st.firstName + " " + temp1st.surName + " is winner for position of"
+ self.role.get() + "\nreceiving a total of " + str(

                             receivedVotes) + " votes.\n" + str(round((receivedVotes / totalVotes) * 100)) +
"%")

        WinLabel.grid(row=2, column=42)



# INITIALIATION

if __name__ == '__main__':

    Log.mainloop()
```