COMP1811 - Python Project Report

Name	**Luis Nepomuceno**		
Login id	**lg6598x**		
Group member names	Jingjie Weng	Gurdeep Singh	Mohammed Islam Uddin
Group member login IDs	jw9085d	GS5299k	im7494x

1. BRIEF STATEMENT OF FEATURES YOU HAVE COMPLETED

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

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
Briefly explain your answer if you circled any parts in 1.2 On A4 we didn't implement the feature where the user can only vote once for each position. On A6 we didn't implement the feature where the results are saved on a file so it can be printed nor the percentage	

2. Concise List of Bugs and Weaknesses

*A concise list of bugs and/or weaknesses in your work (if you don't think there are any, then say so). Bugs that are declared in this list will lose you fewer marks than ones that you don't declare! (**100-200 word**, but word count depends heavily on the number of bugs and weaknesses identified.)*
(THIS SECTION SHOULD BE THE SAME FOR ALL GROUP MEMBERS)

2.1 Bugs

List each bug plus a brief description

On voting,if a lastname that doesn’t exist is entered,it breaks the code since it cannot grab the data from the database.

2.2 Weaknesses

List each weakness plus a brief description

On both the Voting Window and Results Window, after choosing either Factory Officer or Gsu Officer and pressing the select button next to it,it is also required to press enter on the button that is after the droplist that allows the user to select 1,2,3.

3. CLASSES AND OOP FEATURES

*List all the classes used in your program and include the attributes and behaviours for each. You may use a class diagram to illustrate these classes. Your narrative for section 3.2 should describe the design decisions you made and the OOP techniques used (**200-400 words**).*
(THIS SECTION SHOULD BE THE SAME FOR ALL GROUP MEMBERS)

3.1 CLASSES USED

MAINFRAME , IT DEFINES THE BASE FOR ALL THE FRAMES THAT ARE USED WITHIN THE SYSTEM.

LOGINWINDOW, IT DEFINES THE FRAME FOR THE LOGIN WINDOW AND THEN ASKS THE USER TO INPUT A USERNAME AND A PASSWORD,IT CHECKS THE DATABASE TO SEE IF THERE IS A ROW WHERE THE USERNAME AND PASSWORD ALINE AND IF SO ALOWS THE USER TO LOGIN,IF NOT THEN POPS UP A ALERT.

STARTWINDOW,DEFINES STARTED FRAME WHERE THE USER CAN CHOOSE EITHER TO VOTE OR TO CHECK THE RESULTS.

VOTEWINDOW DEFINES THE FRAME FOR THE VOTE WINDOW,DISPLAYS A DROP LIST THAT ALLOWS THE USER TO SELECT A CANDIDATE IT DESIRES TO VOTE FOR THEN IF IT'S NOT "PRESIDENT" IT SHOWS ANOTHER DROP LIST TO CHOOSED THE POSITION_ID AND THEN PASSES POSITION,POSITION_ID AND FACULTY(DEPENDING ON WHICH THE USER HAS CHOOSEN) INTO THE CANDIDATES

CANDIDATES, IT ACCESSES DE DATABASE SHOWS THE AVAIABLE CANDIDATES TO VOTE FOR AND ALLOWS THE USER TO INPUT THE LAST NAME OF THE CANDIDATE IT WANTS TO VOTE FOR AND UPDATES THE DATABASE,ADDING +1 VOTE TO THE CHOOSEN CANDIDATE.

SHOWRESULTS,DEFINES THE FRAME FOR THE RESULTS WINDOW THEN IT ALLOWS THE USER TO CHOOSE THE POSITION AND POSITION ID IT WANTS TO SEE THE RESULTS FOR,IN CASE THE USER CHOOSES FOR FACULTY_OFFICER THEN IT GETS THE FACULTY USING THE DATABASE.IT THEN SHOWS THE VOTES AND ALLOWS THE USER TO CHECK THE WINNER.

3.2 BRIEF EXPLANATION OF CLASS DESIGN AND OOP FEATURES USED

We utilized Naming conventions which is a good example of a OOP,the feature allows the whole code to become easier for the user to read and navigate within it.
We also utilized Camel case that states that we iniciate classes with a upper-case letter

4. Code for the Classes Created

*Add the **code for each of the classes you have implemented yourself** here. If you have contributed to parts of classes, please highlight those parts in a different colour. Copy and paste relevant code - actual code please, no screenshots! Make it easy for the tutor to read. Add explanation if necessary – though your in-code comments should be clear enough. (WRITE THIS SECTION INDIVIDUALLY)*

4.1 Class …

```
class MainFrame(tk.Tk):
    def __init__(self,*args,**kwargs):
        tk.Tk.__init__(self,*args,**kwargs)

        self.title("Main menu")
        self.geometry("500x500")

        window = tk.Frame(self)
        window.pack(side = "top", fill = "both", expand = True)
        window.grid_rowconfigure(0,weight =1)
        window.grid_columnconfigure(0,weight =1)

        self.frames = {}
        avaiable_frames = (StartWindow,LoginWindow,VoteWindow,ShowResults)
        for i in avaiable_frames:
            page_name = i.__name__
            frame = i(parent = window,controller = self)
            self.frames[page_name] = frame
            frame.grid(row=0,column = 0,sticky = "nsew")
        self.show_window("LoginWindow")
#defines which frame is going to appear on the screen.
    def show_window(self, window_name):
        frame = self.frames[window_name]
        frame.tkraise()
```

4.2 Class …

```
class Candidates(tk.Frame):
    #gets and shows the candidates for President
    def president(self,position):
        record = conn.execute(
            "SELECT First_name,Last_name from Gsu_Candidates WHERE Position =:position",
            {'position': position})
        Candidates.show_candidates(self,record)

    #gets and shows the candidates for FacultyOfficer
    def FacultyOfficer(self,position_ID,Faculty):
        record = conn.execute(
            "SELECT First_name,Last_name from Gsu_Candidates WHERE Position =:position AND
Position_ID=:position_ID AND Faculty=:Faculty",
            {'position': "Faculty_Officer", "position_ID": position_ID, "Faculty": Faculty})
        Candidates.show_candidates(self,record)

    # gets and shows the candidates to GsuOfficer
```

```
    def GsuOfficer(self, position_ID):
        record = conn.execute(
            "SELECT First_name,Last_name from Gsu_Candidates WHERE Position =:position AND
Position_ID=:position_ID",
            {'position': "Gsu_Officer", "position_ID": position_ID})
        Candidates.show_candidates(self,record)

    # shows the candidates for choosen position.
    def show_candidates(self,record):
        self.text = tk.Text(self, width=25, heigh=4)
        self.text.place(x=140, y=120)
        records = record.fetchall()
        for record in records:
            self.text.insert("0.1", "\n")
            self.text.insert("0.1", record)
        Candidates.GetVotes(self)
def GetVotes(self):
    self.warning = Label(self, text="Enter Last names to Vote:", width=20, font=("bold", 10))
    self.warning.place(x=150, y=240)
    self.Vote1 = Label(self, text="1st preference:", width=20, font=("bold", 10))
    self.Vote1.place(x=60, y=280)
    self.Vote2 = Label(self, text="2nd Preference:", width=20, font=("bold", 10))
    self.Vote2.place(x=60, y=320)
    self.Vote3 = Label(self, text="3rd Preference:", width=20, font=("bold", 10))
    self.Vote3.place(x=60, y=360)
    self.Vote4 = Label(self, text="4th Preference:", width=20, font=("bold", 10))
    self.Vote4.place(x=60, y=400)

    self.entry_Vote1 = Entry(self)
    self.entry_Vote1.place(x=200, y=280)
    self.entry_Vote2 = Entry(self)
    self.entry_Vote2.place(x=200, y=320)
    self.entry_Vote3 = Entry(self)
    self.entry_Vote3.place(x=200, y=360)
    self.entry_Vote4 = Entry(self)
    self.entry_Vote4.place(x=200, y=400)

    self.submit = tk.Button(self, bg="black", fg="white", text="Vote",
                        command=lambda: Candidates.Voting(self))
    self.submit.place(x=250, y=450)




def Voting(self):
    last_1_txt = getattr(self,"entry_Vote1")
    last_1 = last_1_txt.get()
    last_2_txt = getattr(self,"entry_Vote2")
    last_2 = last_2_txt.get()
    last_3_txt = getattr(self,"entry_Vote3")
    last_3 = last_3_txt.get()
    last_4_txt = getattr(self,"entry_Vote4")
    last_4 = last_4_txt.get()
```

```
    if last_1 == last_2 or last_1 == last_3 or last_1 == last_4 or last_2 == last_3 or last_2
== last_4 or last_3 == last_4:
        tm.showerror("Voting Error", "No duplicate voting allowed")
        self.entry_Vote1.delete(0, END)
        self.entry_Vote2.delete(0, END)
        self.entry_Vote3.delete(0, END)
        self.entry_Vote4.delete(0, END)
    else :
        tm.showinfo("Vote", "Voting Sucessfull")
        self.entry_Vote1.delete(0, END)
        self.entry_Vote2.delete(0, END)
        self.entry_Vote3.delete(0, END)
        self.entry_Vote4.delete(0, END)

        votes_1 = conn.execute("SELECT Vote_1  from Gsu_Candidates WHERE Last_name
=:Last_name",
                               {'Last_name': last_1})
        for row in votes_1.fetchall():
            actual_votes = row[0]
            New_vote_1 = actual_votes + 1

        votes_2 = conn.execute("SELECT Vote_2 from Gsu_Candidates WHERE Last_name
=:Last_name",
                               {'Last_name': last_2})
        for row in votes_2.fetchall():
            actual_votes = row[0]
            New_vote_2 = actual_votes + 1

        votes_3 = conn.execute("SELECT Vote_3 from Gsu_Candidates WHERE Last_name
=:Last_name",
                               {'Last_name': last_3})
        for row in votes_3.fetchall():
            actual_votes = row[0]
            New_vote_3 = actual_votes + 1

        votes_4 = conn.execute("SELECT Vote_4 from Gsu_Candidates WHERE Last_name
=:Last_name",
                               {'Last_name': last_4})
        for row in votes_4.fetchall():
            actual_votes = row[0]
            New_vote_4 = actual_votes + 1

        conn.execute("UPDATE Gsu_Candidates SET Vote_1 =:New_vote_1  WHERE Last_name
=:Last_name",
                     {"New_vote_1": New_vote_1, "Last_name": last_1})

        conn.execute("UPDATE Gsu_Candidates SET Vote_2 =:New_vote_2  WHERE Last_name
=:Last_name",
                     {"New_vote_2": New_vote_2, "Last_name": last_2})

        conn.execute("UPDATE Gsu_Candidates SET Vote_3 =:New_vote_3  WHERE Last_name
=:Last_name",
                     {"New_vote_3": New_vote_3, "Last_name": last_3})
```

```
        conn.execute("UPDATE Gsu_Candidates SET Vote_4 =:New_vote_4  WHERE Last_name
=:Last_name",
                     {"New_vote_4": New_vote_4, "Last_name": last_4})

        conn.commit()
```

4.3 CLASS …

```
    class StartWindow(tk.Frame):
        def __init__(self,parent,controller):
            tk.Frame.__init__(self,parent)
            self.controller = controller


            self.Header = Label(self,text = "Voting menu",width = 25,font= ("bold",20))
            self.Header.place (x=35,y=40)
            self.vote = Button(self,text ="Vote",width = 20,bg ="black",fg = "white",
                               command=lambda: vote_clicked(controller)).place(x = 160,y = 150)
            self.vote = Button(self, text="Show Results", width=20, bg="black", fg="white",
                               command=lambda: results_clicked(controller)).place(x=160, y=200)


    #Shows the voting Window
    def vote_clicked(controller):
        controller.show_window("VoteWindow")
    #Shows the Results window
    def results_clicked(controller):
        controller.show_window("ShowResults")
```

4.4 CLASS …

```
class VoteWindow(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller

        self.Header = Label(self, text="Voting system", width=20, font=("bold", 18))
        self.Header.place(x=120, y=20)

        self.signout = tk.Button(self, bg = "black", fg = "white", text="signout",
command=lambda:return_clicked(controller))
        self.signout.place(x=440, y=460)


        self.label_gsu = Label(self, text="GSU Position:", width=20, font=("bold", 11))
        self.label_gsu.place(x=-40, y=80)

        list1 = ["President", "GSU_Officer", "Faculty_Officer"]
```

```
        self.gsuSelected = tk.StringVar()
        self.droplist1 = tk.OptionMenu(self, self.gsuSelected, *list1)
        self.droplist1.config(width=15)
        self.gsuSelected.set("Select")
        self.droplist1.place(x=100, y=77)
        self.positionSelected = tk.StringVar()
        self.select = tk.Button(self, bg="black", fg="white", text="select",
                                command=lambda: Choice(self))
        self.select.place(x=240, y=79)
#checks what the user choose in gsuSelected and then if the user entered President passes
self,position to the Candidates.President
# if the user chooses Gsu_Officer or Faculty_Officer then it goes to Button2 where the user
can choose the Position_ID
def Choice(self):
        position = self.gsuSelected.get()
        if position == "President":
            Candidates.president(self,position)
        elif position == "GSU_Officer":
            button2(self)
        elif position == "Faculty_Officer":
            button2(self)
#makes a drop list and another button to appear on the window,where it lets the user choose
which Position_ID it wants to vote for.
def button2(self):
    position = self.gsuSelected.get()
    droplist2 = tk.OptionMenu(self, self.positionSelected, "1", "2", "3")
    droplist2.config(width=15)
    droplist2.place(x=300, y=77)
    select2 = tk.Button(self, bg="black", fg="white", text="select",
                        command=lambda: Choice2(self, position))
    select2.place(x=440, y=79)
# gets position_ID from the user selection on droplist 2 and then checks if the user chooses
Gsu_Officer,if so passes position_ID
# to GsuOfficer in candidates,else it gets the student Faculty and passes position_ID and
Faculty to FacultyOfficer
# in cadidates.
def Choice2(self,position):
        conn = sqlite3.connect("GsuCandidates.db")
        position_ID = self.positionSelected.get()
        if position == "GSU_Officer":
            Candidates.GsuOfficer(self,position_ID)
        else :
            with open("Students.txt") as f:
                data = f.readline()
                student_id = data
                tv= conn.execute("SELECT faculty from Students WHERE Student_ID=:Student_id",{
"Student_id": student_id})
            for row in tv.fetchall():
                Faculty = row[0]
            Candidates.FacultyOfficer(self,position_ID,Faculty)

#returns to the main page
```

```
def return_clicked(controller):
    controller.show_window("StartWindow")
```

4.5

```
class ShowResults(tk.Frame):
    def __init__(self,parent,controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller

        self.Header = Label(self, text="Voting Results", width=20, font=("bold", 18))
        self.Header.place(x=100, y=20)

        self.signout = tk.Button(self, bg="black", fg="white", text="signout",
                                 command=lambda: return_clicked(controller))
        self.signout.place(x=440, y=460)

        self.label_position = Label(self, text="")
        self.label_gsu = Label(self, text="GSU Position:", width=20, font=("bold", 11))
        self.label_gsu.place(x=-40, y=80)

        list1 = ["President", "Gsu_Officer", "Faculty_Officer"]

        self.gsuSelected = tk.StringVar()
        self.droplist1 = tk.OptionMenu(self, self.gsuSelected, *list1)
        self.droplist1.config(width=15)
        self.gsuSelected.set("Select")
        self.droplist1.place(x=100, y=77)
        self.positionSelected = tk.StringVar()
        self.select = tk.Button(self, bg="black", fg="white", text="select",
                                command=lambda: Choice(self))
        self.select.place(x=240, y=79)

#   Act as select funcionality,gets the position from gsuSelected and passes it on into report
(if it's President
#   only has 1 Position_ID or passes onto button 2 so the user can select the Position_ID.
def Choice(self):
    position = self.gsuSelected.get()
    if position == "President":
        report(self,position)
    elif position == "Gsu_Officer":
        button2(self,position)
    elif position == "Faculty_Officer":
        button2(self,position)

### acts as button 2 funcionality,it shows a list for 1,2,3 and allows the user to select the
Position_ID.
def button2(self,position):
    droplist2 = tk.OptionMenu(self, self.positionSelected, "1", "2", "3")
    droplist2.config(width=15)
    droplist2.place(x=300, y=77)
    select2 = tk.Button(self, bg="black", fg="white", text="select",
```

```
                                command=lambda: Choice2(self, position))
    select2.place(x=440, y=79)
### creates the button where it gets the position_ID and passes it on to report.
def Choice2(self, position):
    global Faculty
    conn = sqlite3.connect("GsuCandidates.db")
    Position_ID = self.positionSelected.get()
    if position == "Gsu_Officer":
        report(self,position,Position_ID,Faculty = "null")
    else:
        with open("Students.txt") as f:
            data = f.readline()
            student_id = data
            tv = conn.execute("SELECT faculty from Students WHERE Student_ID=:Student_id",
                                        {"Student_id": student_id})
        for row in tv.fetchall():
            Faculty = row[0]
        report(self,position,Position_ID,Faculty)
```

…

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)

Each feature was broke apart and studied in its total,allowing us to make it simples and fucntionable.The features were then worked 1 on 1 and tested to see if they were working and then the whole code was brought together,we used database as a means to store and get the data from the candidates as well as uploading the new data inside the database.
The user is allowed to login,and if he is outside the deadline he is blocked out of the system.
If he is withing the deadline the user can then Vote and when Voting he can choose which position to vote for and which position_ID to vote for,the same happens when seeing the results of the voting.

6. TESTING

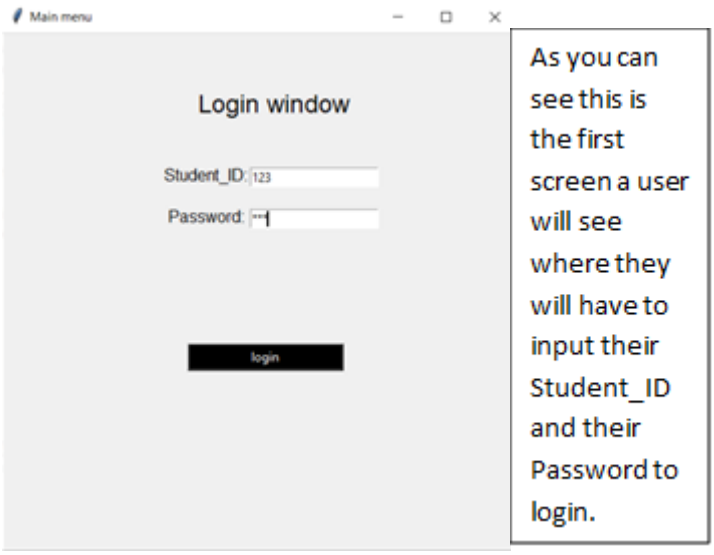
Describe the process you took to test your code and to make sure the program functions as required. Also, indicate the testing you've done after integrating code from other group members.
(WRITE THIS SECTION INDIVIDUALLY)

After programming the code,I did various tests on it.
I started by checking If the database was connected and if the values were being updated,then I discovered the bug where if the last name doesn't belong to the database then the code breaks since it can't find the "Last_name" withing the database and so it can't get the actual votes that candidate has nor can it update them.
I've also tested the login by entering incorrect username and passwork and it worked as I predicted it would,if the username or password are wrong it displays a popup alert to let the user know that the credentials are incorrect.
After impletending the code from the group, I performed the same tests and it worked without any errors.

7. Annotated Screenshots Demonstrating Implementation

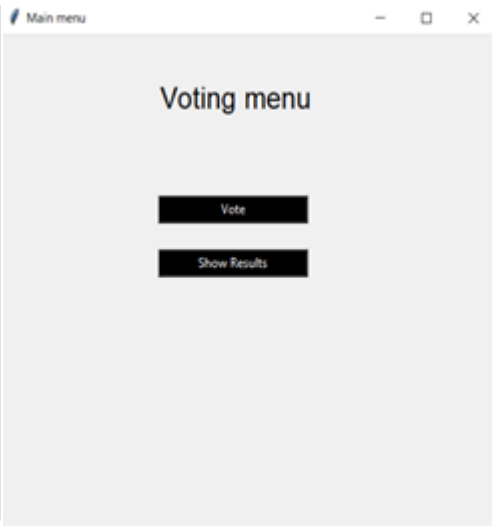
*Provide screenshots that demonstrate the features implemented. Give a brief description for **each** (**up to 100 words**) to explain the code in action. Make sure the screenshots make clear what you have implemented and achieved. (THIS SECTION SHOULD BE THE SAME FOR ALL GROUP MEMBERS)*

7.1 Feature A.1 - screenshots …

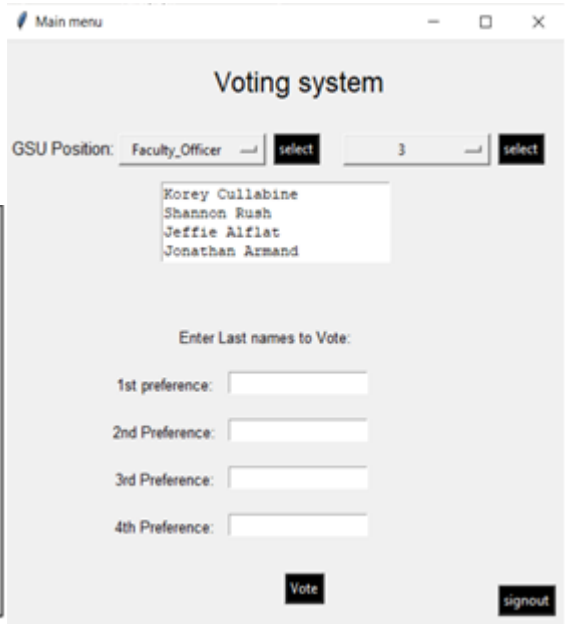


As you can see this is the first screen a user will see where they will have to input their Student_ID and their Password to login.

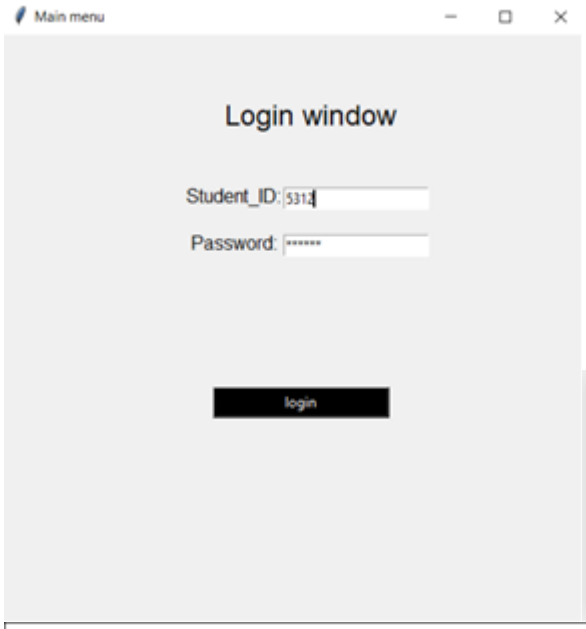
As you can see in this window shows the voting menu where after logged in, they will have two buttons where they can either vote or show the results for the GSU positions.

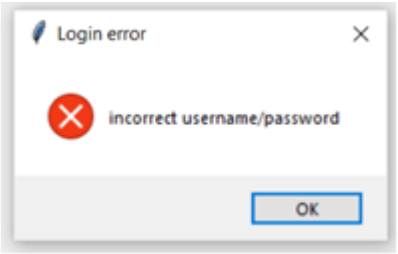


As you can see in this window shows the Voting for GSU position after selecting the preferred vote they can click on "Signout" button to exit, after clicking that it will take back to the Voting menu.



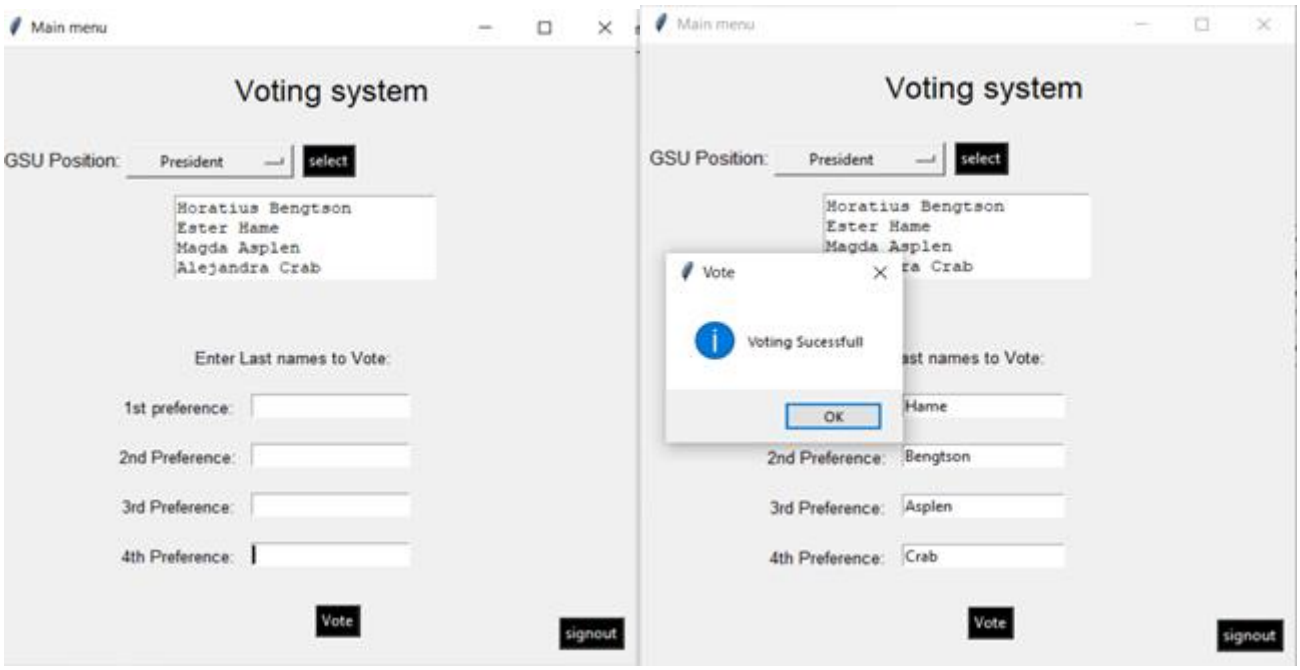
7.2 FEATURE A.2 - SCREENSHOTS …





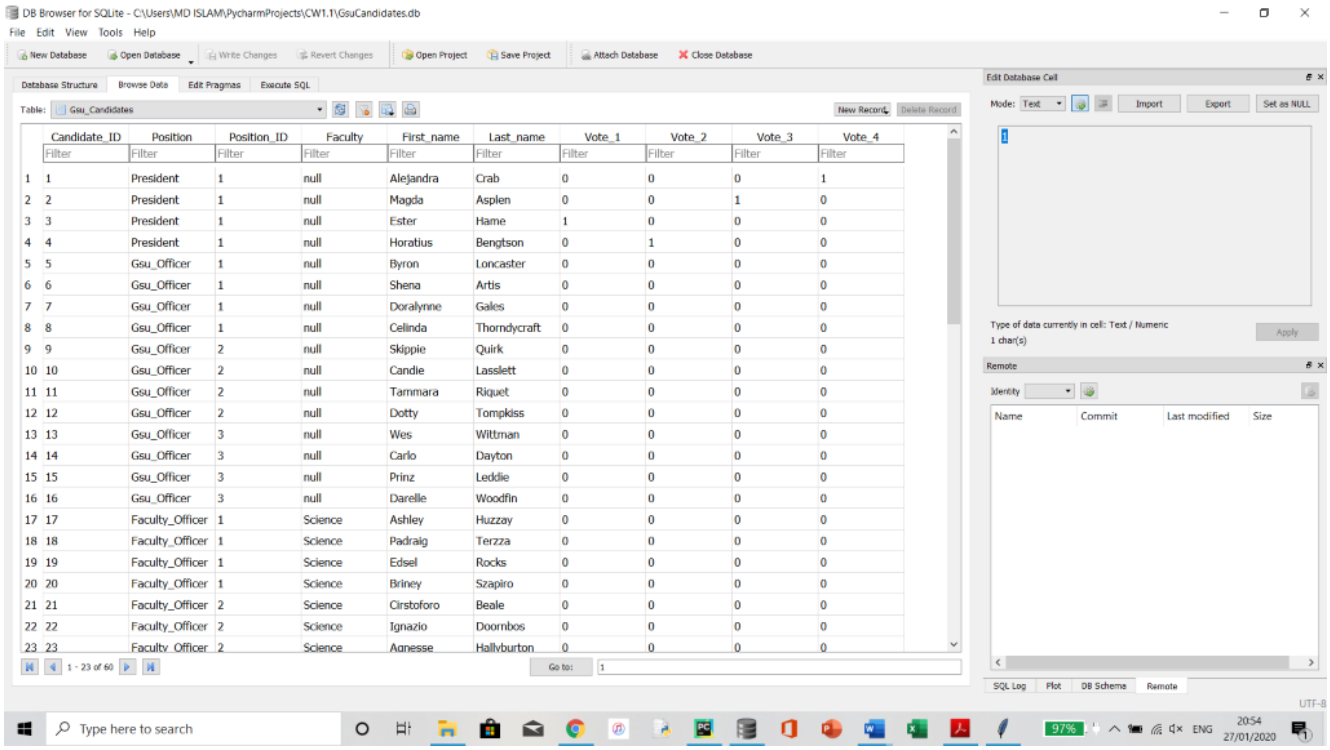
As you can see in this window shows the Login window which is created via a tkinter class called LoginFrame. In this window if any user input a wrong detail it will out put an error message that will say “incorrect username/password” as it shows in the second screenshots and this is done by checking the *Student_ID/Password* to the Students.txt file.

7.3 FEATURE A.3 - SCREENSHOTS …



As you can see in this window shows the candidates name which after inputting data's it will update in main sql database.

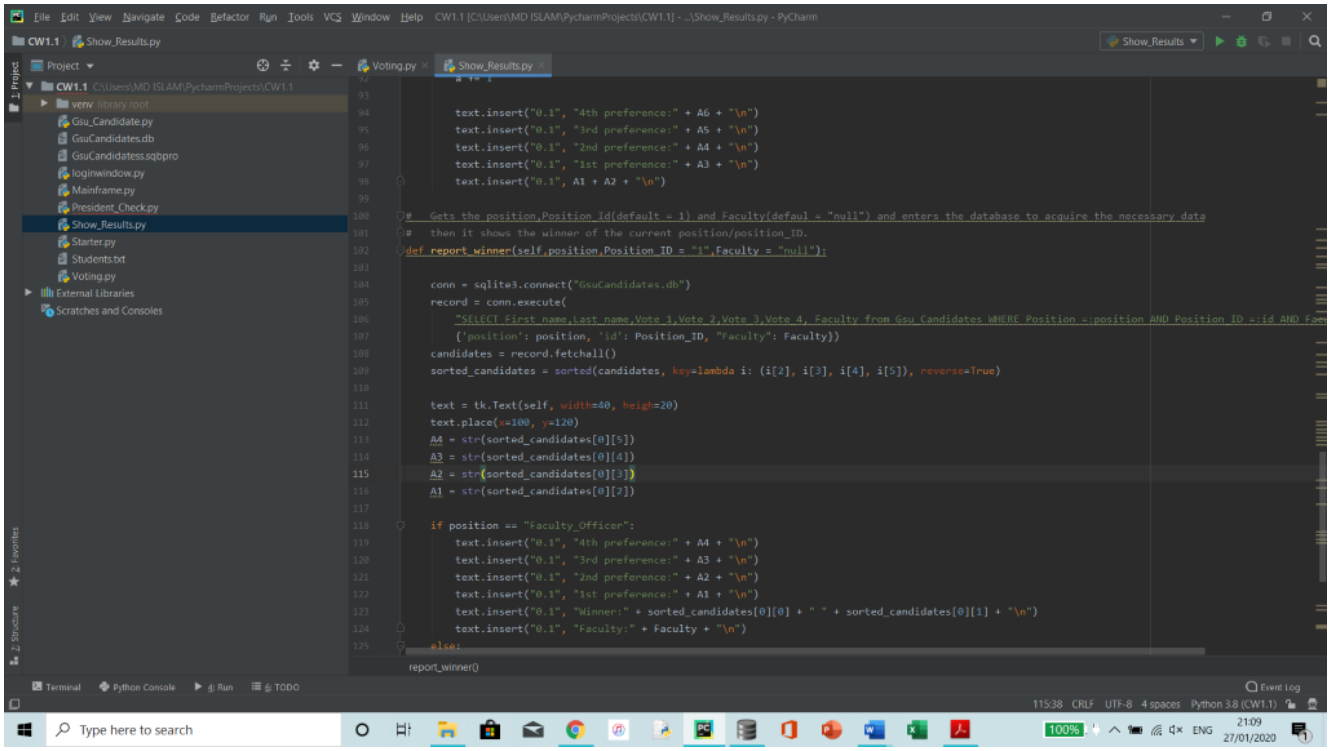
7.4 FEATURE A.4 - SCREENSHOTS …



As you can see in this window shows the candidates database where it can be edited by the program and it updates after the user voted in the program.

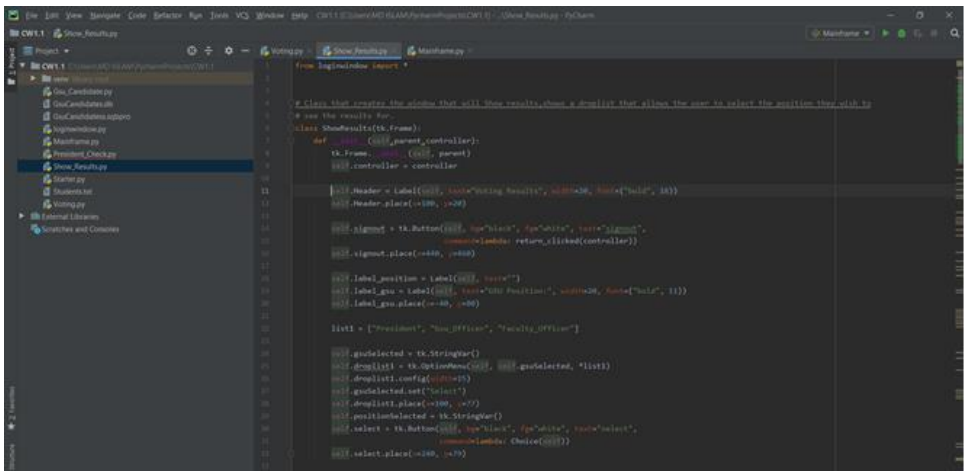
7.5 FEATURE A.5 - SCREENSHOTS …

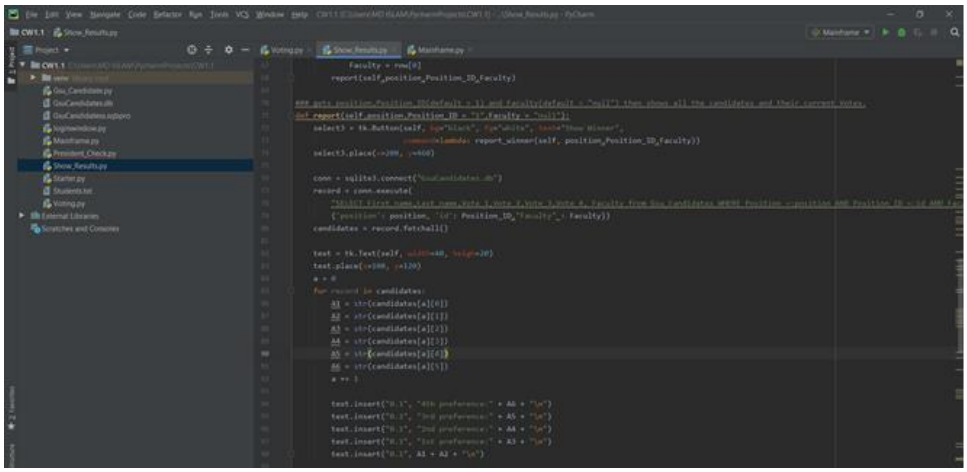
On A5,we acess the database to get the votes and then add +1 vote to the preference choosen by the user and then update the database with that new information.



7.6 FEATURE A.6 - SCREENSHOTS …

On A6 we get the data from the database and we display it to the user utilizing a .text function within tkinter.The user can then choose to see the winner only.





7.7 Feature B.1 - screenshots …

7.8 Feature B.2 - screenshots …

7.9 Feature B.3 - screenshots …

7.10 Feature B.4 - screenshots …

…

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 SECTION INDIVIDUALLY)*

8.1 Evaluate how well your design and implementation meet the requirements

The design of the code was fairly plain,we cloud have used ttk instead of tk within tkinter since ttk is overall better looking.As far as the code is concerned the whole design could have been cleaner but since the classes and funcionts where designed individually before they were all implemented together it isn't as good as I would hope it to be.

8.2 Evaluate you own and your group's performance

As far as my performance goes, I performed a lot of research concerning the GUI and how to utilize the database(sqlite3) that was build into python to get and update the data inside it,I also encountered some issues but was able to fix them while debugging the code.The group could have performed better,but the part where they contributed was clean and didn't had any major errors overral.

8.2.1 What went well?

Overral

The GUI and the login system went fairly well,and the impletentation of the whole code was smooth and without any major error or bug.

8.2.2 What went less well?

Utilizing the database to get data and update data within it took a little while to get used to,but at the end of the code it all went smoothly enough.

8.2.3 What was learnt?

I learnt a lot about GUI as well as using a database within python,also creating classes and importing those classes into other files.I also improved a little on basic SQL since SQLite 3(which is built into python) required basic SQL commands.

8.2.4 How would a similar task be completed differently?

Instead of using SQLITE3 we could have used text files to store the votes,login and candidates.

And the overall code could have been cut down if some def functions would have been used.

8.2.5 How could the module be improved?

On my opinion it was a interesting challenge,but there could have been more examples of the system we had to build withing the lectures.

9. 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
lg6598x	A1,partialA4,A6,partial A5	60%	Luis Nepomuceno	
jw9085d	Partial A4,partial A5	25%	Jingjie Weng	
im7494x	A2	10%	Mohammed Islam	
GS5299k	Database	5%	Gurdeep Singh	
	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 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)

Mainframe.py

```
import tkinter as tk
from Starter import StartWindow
from loginwindow import LoginWindow
from Voting import VoteWindow
from Show_Results import ShowResults

# Main class,defines the blueprint for all the windows within the system.
class MainFrame(tk.Tk):
    def __init__(self,*args,**kwargs):
        tk.Tk.__init__(self,*args,**kwargs)

        self.title("Main menu")
        self.geometry("500x500")

        window = tk.Frame(self)
        window.pack(side = "top", fill = "both", expand = True)
        window.grid_rowconfigure(0,weight =1)
        window.grid_columnconfigure(0,weight =1)

        self.frames = {}
        avaiable_frames = (StartWindow,LoginWindow,VoteWindow,ShowResults)
        for i in avaiable_frames:
            page_name = i.__name__
            frame = i(parent = window,controller = self)
            self.frames[page_name] = frame
            frame.grid(row=0,column = 0,sticky = "nsew")
        self.show_window("LoginWindow")
#defines which frame is going to appear on the screen.
    def show_window(self, window_name):
        frame = self.frames[window_name]
        frame.tkraise()



exe = MainFrame()
exe.mainloop()
```

Loginwindow.py

```
import tkinter as tk
from tkinter import *
```

```
import sqlite3
import tkinter.messagebox as tm
import datetime

today_date = datetime.datetime.now()
deadline = datetime.datetime(2020, 1, 28)

#Defines the first window to appear,allows the user to login
class LoginWindow(tk.Frame):
    def __init__(self,parent,controller):

        tk.Frame.__init__(self,parent)
        self.controller = controller

        self.Header = Label(self,text = "Login window",width = 20, font =("bold",18))
        self.Header.place(x = 120,y = 53)

        self.StudentID_label = Label(self,text = "Student_ID:",width = 20,font =("bold",12))
        self.StudentID_label.place(x = 105,y= 125)

        self.Password_label = Label(self,text = "Password:",width = 20,font =("bold",12))
        self.Password_label.place(x = 105,y= 165)

        self.entry_student_ID = Entry(self)
        self.entry_student_ID.place(x = 240,y= 130)

        self.entry_password = Entry(self,show = "*")
        self. entry_password.place(x = 240, y =170)

        self.login_button = Button(self,text = "login",width = 20,  bg = "black", fg =
"white",
                                   command=lambda: self.LoginClicked(controller))
        self.login_button.place(x=180 ,y=300)


#gets the data from the student ID and password entrys and then checks to see if the user is
within the database.
    def LoginClicked(self,controller):
        db = sqlite3.connect("GsuCandidates.db")
        cursor = db.cursor()



        StudentID_txt = getattr(self, "entry_student_ID")
        StudentID = StudentID_txt.get()

        password_txt = getattr(self, "entry_password")
        password = password_txt.get()

        query = "SELECT * FROM Students WHERE Student_ID = '{}' AND Password =
'{}'".format(StudentID, password)
```

```
        cursor.execute(query)
        rows = cursor.fetchall()
    # checks if there is a row where the Student ID and Password match,if so allow the user to
login but first performs
    # a check using the today_date and deadline,if the user is trying to vote outside the
deadline a pop up error will open up.
        if len(rows) == 1:
            if today_date <= deadline:
                f = open("Students.txt","w+")
                f.write(StudentID)
                tm.showinfo("login","Welcome")
                controller.show_window("StartWindow")
            else:
                tm.showerror("Login error", "Elections are over")
        else:
            tm.showerror("Login error","incorrect username/password")
```

Starter.py

```
from tkinter import *
import tkinter as tk
# defines the StartWindow where the user can select if he wants to vote or to check the
results of the election.
class StartWindow(tk.Frame):
    def __init__(self,parent,controller):
        tk.Frame.__init__(self,parent)
        self.controller = controller

        self.Header = Label(self,text = "Voting menu",width = 25,font= ("bold",20))
        self.Header.place (x=35,y=40)
        self.vote = Button(self,text ="Vote",width = 20,bg ="black",fg = "white",
                           command=lambda: vote_clicked(controller)).place(x = 160,y = 150)
        self.vote = Button(self, text="Show Results", width=20, bg="black", fg="white",
                           command=lambda: results_clicked(controller)).place(x=160, y=200)

#Shows the voting Window
def vote_clicked(controller):
    controller.show_window("VoteWindow")
#Shows the Results window
def results_clicked(controller):
    controller.show_window("ShowResults")
```

Voting.py

```
from loginwindow import *
from Gsu_Candidate import Candidates
#defines the Voting Window,where the usar can select President,Gsu_Officer or Faculty_Officer
class VoteWindow(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller
```

```
        self.Header = Label(self, text="Voting system", width=20, font=("bold", 18))
        self.Header.place(x=120, y=20)

        self.signout = tk.Button(self, bg = "black", fg = "white", text="signout",
command=lambda:self.return_clicked(controller))
        self.signout.place(x=440, y=460)



        self.label_gsu = Label(self, text="GSU Position:", width=20, font=("bold", 11))
        self.label_gsu.place(x=-40, y=80)

        list1 = ["President", "GSU_Officer", "Faculty_Officer"]

        self.gsuSelected = tk.StringVar()
        self.droplist1 = tk.OptionMenu(self, self.gsuSelected, *list1)
        self.droplist1.config(width=15)
        self.gsuSelected.set("Select")
        self.droplist1.place(x=100, y=77)
        self.positionSelected = tk.StringVar()
        self.select = tk.Button(self, bg="black", fg="white", text="select",
                                command=lambda: self.Choice())
        self.select.place(x=240, y=79)
#checks what the user choose in gsuSelected and then if the user entered President passes
self,position to the Candidates.President
# if the user chooses Gsu_Officer or Faculty_Officer then it goes to Button2 where the user
can choose the Position_ID
    def Choice(self):
        position = self.gsuSelected.get()
        if position == "President":
            Candidates.president(self,position)
        elif position == "GSU_Officer":
            self.button2()
        elif position == "Faculty_Officer":
            self.button2()
#makes a drop list and another button to appear on the window,where it lets the user choose
which Position_ID it wants to vote for.
    def button2(self):
        position = self.gsuSelected.get()
        droplist2 = tk.OptionMenu(self, self.positionSelected, "1", "2", "3")
        droplist2.config(width=15)
        droplist2.place(x=300, y=77)
        select2 = tk.Button(self, bg="black", fg="white", text="select",
                            command=lambda: self.Choice2(position))
        select2.place(x=440, y=79)
# gets position_ID from the user selection on droplist 2 and then checks if the user chooses
Gsu_Officer,if so passes position_ID
# to GsuOfficer in candidates,else it gets the student Faculty and passes position_ID and
Faculty to FacultyOfficer
# in cadidates.
    def Choice2(self,position):
        conn = sqlite3.connect("GsuCandidates.db")
```

```
        position_ID = self.positionSelected.get()
        if position == "GSU_Officer":
            Candidates.GsuOfficer(self,position_ID)
        else :
            with open("Students.txt") as f:
                data = f.readline()
                student_id = data
                tv= conn.execute("SELECT faculty from Students WHERE Student_ID=:Student_id",{
"Student_id": student_id})
            for row in tv.fetchall():
                Faculty = row[0]
            Candidates.FacultyOfficer(self,position_ID,Faculty)

#returns to the main page
    def return_clicked(self,controller):
        controller.show_window("StartWindow")
```

Show_Results.py

```
from loginwindow import *
# Class that creates the window that will Show results,shows a droplist that allows the user
to select the position they wish to
# see the results for.
class ShowResults(tk.Frame):
    def __init__(self,parent,controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller

        self.Header = Label(self, text="Voting Results", width=20, font=("bold", 18))
        self.Header.place(x=100, y=20)

        self.signout = tk.Button(self, bg="black", fg="white", text="signout",
                                 command=lambda: return_clicked(controller))
        self.signout.place(x=440, y=460)

        self.label_position = Label(self, text="")
        self.label_gsu = Label(self, text="GSU Position:", width=20, font=("bold", 11))
        self.label_gsu.place(x=-40, y=80)

        list1 = ["President", "Gsu_Officer", "Faculty_Officer"]

        self.gsuSelected = tk.StringVar()
        self.droplist1 = tk.OptionMenu(self, self.gsuSelected, *list1)
        self.droplist1.config(width=15)
        self.gsuSelected.set("Select")
        self.droplist1.place(x=100, y=77)
        self.positionSelected = tk.StringVar()
        self.select = tk.Button(self, bg="black", fg="white", text="select",
                                command=lambda: self.Choice())
        self.select.place(x=240, y=79)
```

```
#   Act as select funcionality,gets the position from gsuSelected and passes it on into report
(if it's President
#   only has 1 Position_ID or passes onto button 2 so the user can select the Position_ID.
    def Choice(self):
        position = self.gsuSelected.get()
        if position == "President":
            self.report(position)
        elif position == "Gsu_Officer":
            self.button2(position)
        elif position == "Faculty_Officer":
            self.button2(position)

### acts as button 2 funcionality,it shows a list for 1,2,3 and allows the user to select the
Position_ID.
    def button2(self,position):
        droplist2 = tk.OptionMenu(self, self.positionSelected, "1", "2", "3")
        droplist2.config(width=15)
        droplist2.place(x=300, y=77)
        select2 = tk.Button(self, bg="black", fg="white", text="select",
                            command=lambda: self.Choice2(position))
        select2.place(x=440, y=79)
### creates the button where it gets the position_ID and passes it on to report.
    def Choice2(self, position):
        global Faculty
        conn = sqlite3.connect("GsuCandidates.db")
        Position_ID = self.positionSelected.get()
        if position == "Gsu_Officer":
            self.report(position,Position_ID)
        else:
            with open("Students.txt") as f:
                data = f.readline()
            student_id = data
            tv = conn.execute("SELECT faculty from Students WHERE Student_ID=:Student_id",
                                    {"Student_id": student_id})
            for row in tv.fetchall():
                Faculty = row[0]
            self.report(position,Position_ID,Faculty)

### gets position,Position_ID(default = 1) and Faculty(default = "null") then shows all the
candidates and their current Votes.
    def report(self,position,Position_ID = "1",Faculty = "null"):
        select3 = tk.Button(self, bg="black", fg="white", text="Show Winner",
                        command=lambda: self.report_winner(position,Position_ID,Faculty))
        select3.place(x=200, y=460)

        conn = sqlite3.connect("GsuCandidates.db")
        record = conn.execute(
            "SELECT First_name,Last_name,Vote_1,Vote_2,Vote_3,Vote_4, Faculty from
Gsu_Candidates WHERE Position =:position AND Position_ID =:id AND Faculty=:Faculty",
            {'position': position, 'id': Position_ID,"Faculty" : Faculty})
        candidates = record.fetchall()
```

```
        text = tk.Text(self, width=40, heigh=20)
        text.place(x=100, y=120)
        a = 0
        for record in candidates:
            A1 = str(candidates[a][0])
            A2 = str(candidates[a][1])
            A3 = str(candidates[a][2])
            A4 = str(candidates[a][3])
            A5 = str(candidates[a][4])
            A6 = str(candidates[a][5])
            a += 1

            text.insert("0.1", "4th preference:" + A6 + "\n")
            text.insert("0.1", "3rd preference:" + A5 + "\n")
            text.insert("0.1", "2nd preference:" + A4 + "\n")
            text.insert("0.1", "1st preference:" + A3 + "\n")
            text.insert("0.1", A1 + A2 + "\n")

#   Gets the position,Position_Id(default = 1) and Faculty(defaul = "null") and enters the
database to acquire the necessary data
#   then it shows the winner of the current position/position_ID.
    def report_winner(self,position,Position_ID = "1",Faculty = "null"):

        conn = sqlite3.connect("GsuCandidates.db")
        record = conn.execute(
            "SELECT First_name,Last_name,Vote_1,Vote_2,Vote_3,Vote_4, Faculty from
Gsu_Candidates WHERE Position =:position AND Position_ID =:id AND Faculty=:Faculty",
            {'position': position, 'id': Position_ID, "Faculty": Faculty})
        candidates = record.fetchall()
        sorted_candidates = sorted(candidates, key=lambda i: (i[2], i[3], i[4], i[5]),
reverse=True)

        text = tk.Text(self, width=40, heigh=20)
        text.place(x=100, y=120)
        A4 = str(sorted_candidates[0][5])
        A3 = str(sorted_candidates[0][4])
        A2 = str(sorted_candidates[0][3])
        A1 = str(sorted_candidates[0][2])

        if position == "Faculty_Officer":
            text.insert("0.1", "4th preference:" + A4 + "\n")
            text.insert("0.1", "3rd preference:" + A3 + "\n")
            text.insert("0.1", "2nd preference:" + A2 + "\n")
            text.insert("0.1", "1st preference:" + A1 + "\n")
            text.insert("0.1", "Winner:" + sorted_candidates[0][0] + " " +
sorted_candidates[0][1] + "\n")
            text.insert("0.1", "Faculty:" + Faculty + "\n")
        else:
            text.insert("0.1", "4th preference:" + A4 + "\n")
            text.insert("0.1", "3rd preference:" + A3 + "\n")
            text.insert("0.1", "2nd preference:" + A2 + "\n")
```

```
            text.insert("0.1", "1st preference:" + A1 + "\n")
            text.insert("0.1", "Winner:" + sorted_candidates[0][0] + " " +
sorted_candidates[0][1] + "\n")

#returns to the main page
    def return_clicked(self,controller):
        controller.show_window("StartWindow")
```