COMP1811 - Python Project Report

Name	**Han Ha**				
Login id	**gh081w**				
Group member names	**Francesco**	**Han**	**Kamelia**	**Alex**	**Hadil**
Group member login IDs	**001086733**	**001052068**	**001085165**	**001068910**	**001090152**

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 ☐B.5
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 ☒B.5

Briefly explain your answer if you circled any parts in 1.2

Regarding part B of this Voting System project, I chose to implement a programme as the task 1.5. required, which is to find top 3 candidates running for the GSU president with the highest vote numbers apart from the winner. I managed to order the candidates according to their voted preference but could not produce the complete code to extract top 3.

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

As a group, we had encountered several bugs and problems which were ineffective in the system we were producing. One of which was that when we created code for submitting a vote, it was not working. After trial and error, we understood that it was due to the push button of the submit vote from the interface was not connected to the function for SQL and in Python. This allowed us to work with the code and fix the problem as the function had a syntax error.
Another bug we faced was within the voting section of the system. We used a SQL database to have all the candidates for voting and allowed the implementation of a voter to submit up to four of the votes- however we had a bug which was that people can ultimately vote unlimited times.
Additionally, a bug we faced in the second GUI (for the GSU Election Results) , connecting the code form the interface to running it in python was stopping it from running due to an invalid syntax. Thankfully we overcame this by immediately removing the line of code that was causing the invalid syntax and causing the code not to run.

2.2 Weaknesses

List each weakness plus a brief description

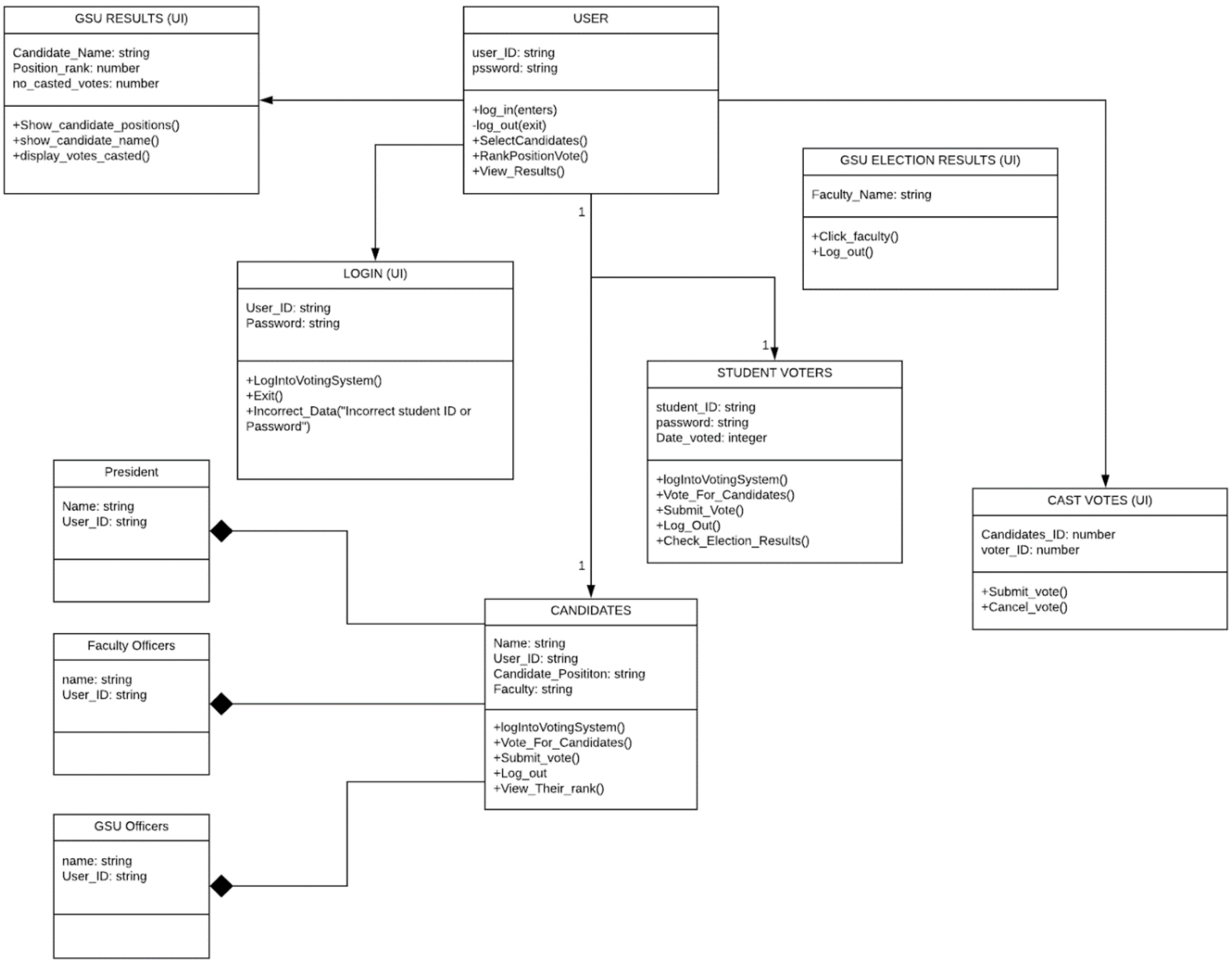
In some aspect, as a group we discussed the use of a SQL database and thought that its use may be a weakness and disadvantage with the system we are implementing. This was due to the struggle we had with finding the right SQL command syntax in Python to connect the database we had to the interface and the voting system we implemented. Furthermore, having to include the 80+ generated candidates and students we thought we may be able to find a different way of processing the database without having to type out every single one manually. However, with further use of the database- we found a syntax command that would allow us to use the database to show the names of the candidates and how many votes each one has received.

Furthermore, a weakness that came out of using database was the SQL knowledge we needed to know as a group to understand how the work through the SQL commands and using SQ3Lite for us to create a database. However, the outcome of using SQ3Lite, strengthened our database knowledge as a group and allowed us to understand how to work through it and use it in future code.

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



3.2 Brief Explanation of Class Design and OOP Features Used

In the class diagram, there are four User Interface classes which connect to the user. These are the; Login UI, Cast Votes UI, GSU Results UI and the GSU Election Results UI. The login UI has several different attributes, including the user ID and the user Password. The operations section includes the ability to Log into the voting system, to exit the system, and if entered incorrect data the warning appears.

The cast votes UI has candidate ID and the voter ID as the attributes and its operations are made up of submitting the votes and cancelling your votes. The Election results UI is a front-end interface which includes the attribute, faculty name as a string, whilst the operations are to click the push button which allows you to access the results from each faculty.

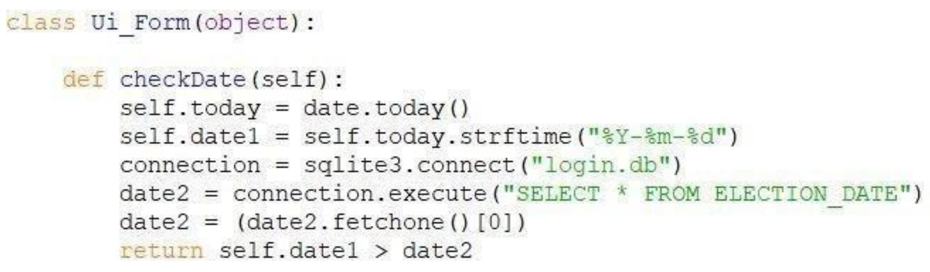
One student voter and candidate can be one user. The candidate class includes attributes of its name, user ID, their candidate position and the Faculty they are in whilst the attributes it has includes the ability to log into the system, to vote for the candidates, to submit the vote, log out and view their ranks, whilst the student voters can submit a vote, they can cancel their vote as well as view election results.

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)*

1. UI_FORM( ):

```
class Ui_Form(object):

    def checkDate(self):

        self.today = date.today()

        self.date1 = self.today.strftime("%Y-%m-%d")

        connection = sqlite3.connect("login.db")

        date2 = connection.execute("SELECT * FROM ELECTION_DATE")

        date2 = (date2.fetchone()[0])

        return self.date1 > date2
```



```
class Ui_Form(object):

    def checkDate(self):
        self.today = date.today()
        self.date1 = self.today.strftime("%Y-%m-%d")
        connection = sqlite3.connect("login.db")
        date2 = connection.execute("SELECT * FROM ELECTION_DATE")
        date2 = (date2.fetchone()[0])
        return self.date1 > date2
```

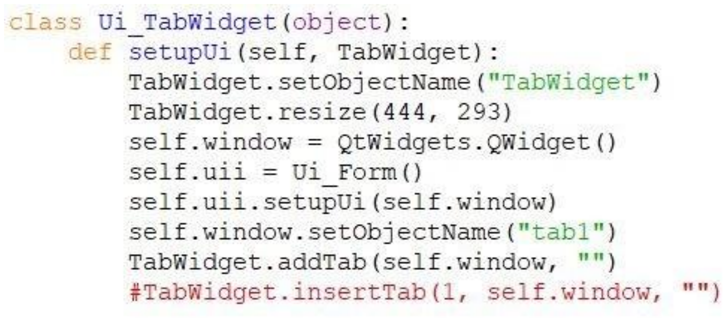
2. UI_TABWIDGET( ):

```
class Ui_TabWidget(object):

    def setupUi(self, TabWidget, user):

        self.username = user

        TabWidget.setObjectName("TabWidget")

        TabWidget.resize(444, 293)

        self.window =  QtWidgets.QWidget()

        self.uii = Ui_Form()

        self.uii.setupUi(self.window,self.username)

        self.window.setObjectName("tab1")

        TabWidget.addTab(self.window, "")
```

#TabWidget.insertTab(1, self.window, "")

self.retranslateUi(TabWidget)

QtCore.QMetaObject.connectSlotsByName(TabWidget)



```
class Ui_TabWidget(object):
    def setupUi(self, TabWidget):
        TabWidget.setObjectName("TabWidget")
        TabWidget.resize(444, 293)
        self.window = QtWidgets.QWidget()
        self.uii = Ui_Form()
        self.uii.setupUi(self.window)
        self.window.setObjectName("tab1")
        TabWidget.addTab(self.window, "")
        #TabWidget.insertTab(1, self.window, "")
```

3. UI_MAINWINDOW( ):

class Ui_MainWindow(object):

def setupUi(self, MainWindow):

MainWindow.setObjectName("MainWindow")

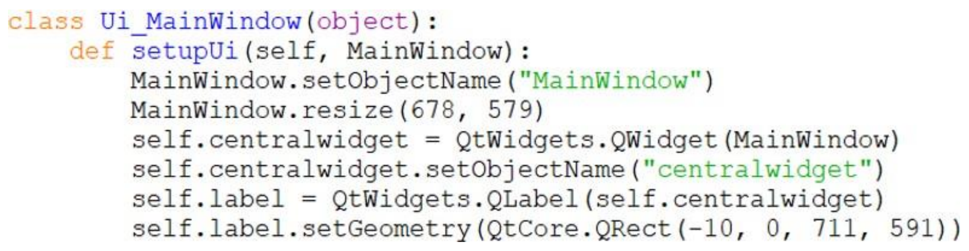
MainWindow.resize(678, 579)

self.centralwidget = QtWidgets.QWidget(MainWindow)

self.centralwidget.setObjectName("centralwidget")

self.label = QtWidgets.QLabel(self.centralwidget)

self.label.setGeometry(QtCore.QRect(-10, 0, 711, 591)) *[more lines in electronicresults.py]*



```
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(678, 579)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.label = QtWidgets.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(-10, 0, 711, 591))
```

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)

The first thing we implemented in the system was the login GUI (Graphical User Interface) which essentially is a pop-up screen that allows students and candidates to enter their users and password to access the voting system. With the backend code that includes the functions - when clicking enter, if the student or candidate is not on the database (as a registered user id) it would cause a warning pop up to show saying that this person is not eligible to vote.

In order to upload the candidates, we opted for a database rather than a txt file as we thought it would be easier to access and manipulate the data. So, instead of manipulating strings in a txt file, we were able to run SQL queries to upload the candidates, upload their scores, and eventually figure out the winner(s).

Moreover, to cast votes, users make use of some checkboxes to choose the candidate they want to give their vote(s) to, and they are also able to select their ‘preference’.

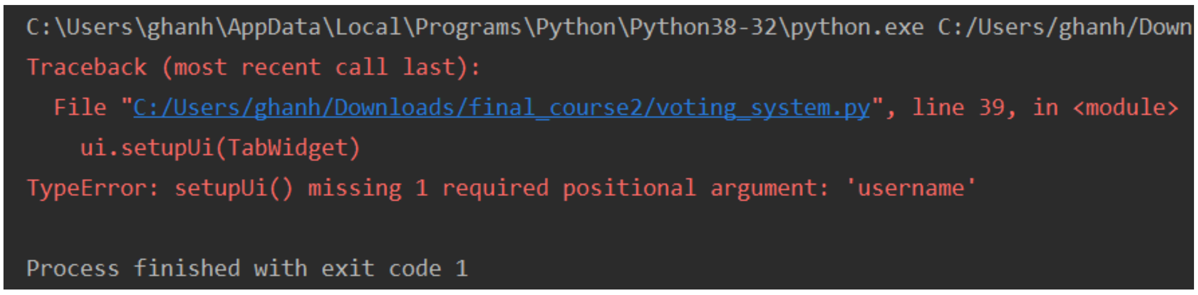
We further implemented a GSU Election Results GUI which had several pushbutton widgets that allow you to access the results of each candidate, separated by faculty. Once again, with the code implemented it allows you to access and view all candidates and how many votes by preference each received.

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)

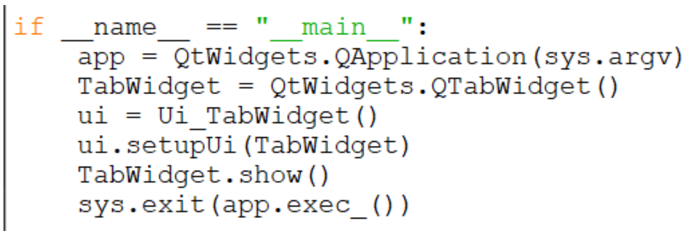
To ensure all the Python code and database files of our group run well, the postproduction process was taken to test bugs and make improvement. Fortunately, most of our code were running well except from ones that mentioned below:

Voting_system.py: When I tried running it for the first time, error occurred because **setupUi()** was missing a required positional argument: 'username' (Picture 6.1.). In order to debug, I decided to remove the if statement as the picture 6.2. showing. After removing this part, the program was running well (Picture 6.3.).



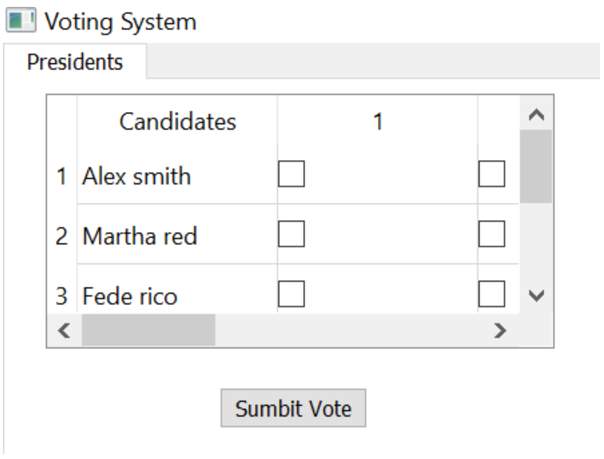
```
C:\Users\ghanh\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/ghanh/Down
Traceback (most recent call last):
  File "C:/Users/ghanh/Downloads/final_course2/voting_system.py", line 39, in <module>
    ui.setupUi(TabWidget)
TypeError: setupUi() missing 1 required positional argument: 'username'

Process finished with exit code 1
```

Picture 6.1.



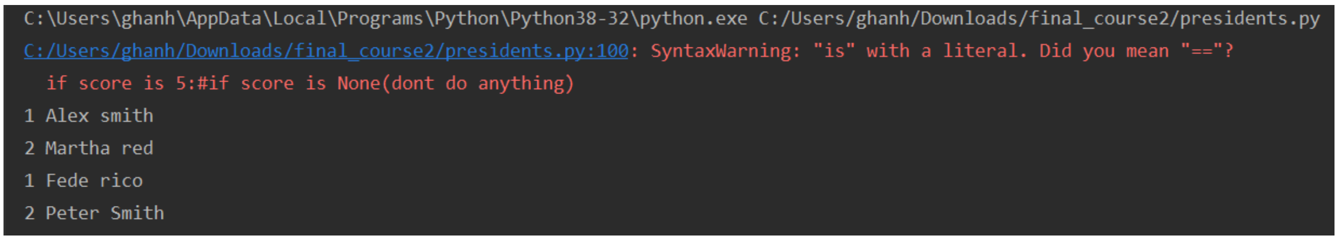
```
if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    TabWidget = QtWidgets.QTabWidget()
    ui = Ui_TabWidget()
    ui.setupUi(TabWidget)
    TabWidget.show()
    sys.exit(app.exec_())
```

Picture 6.2.



Picture 6.3.

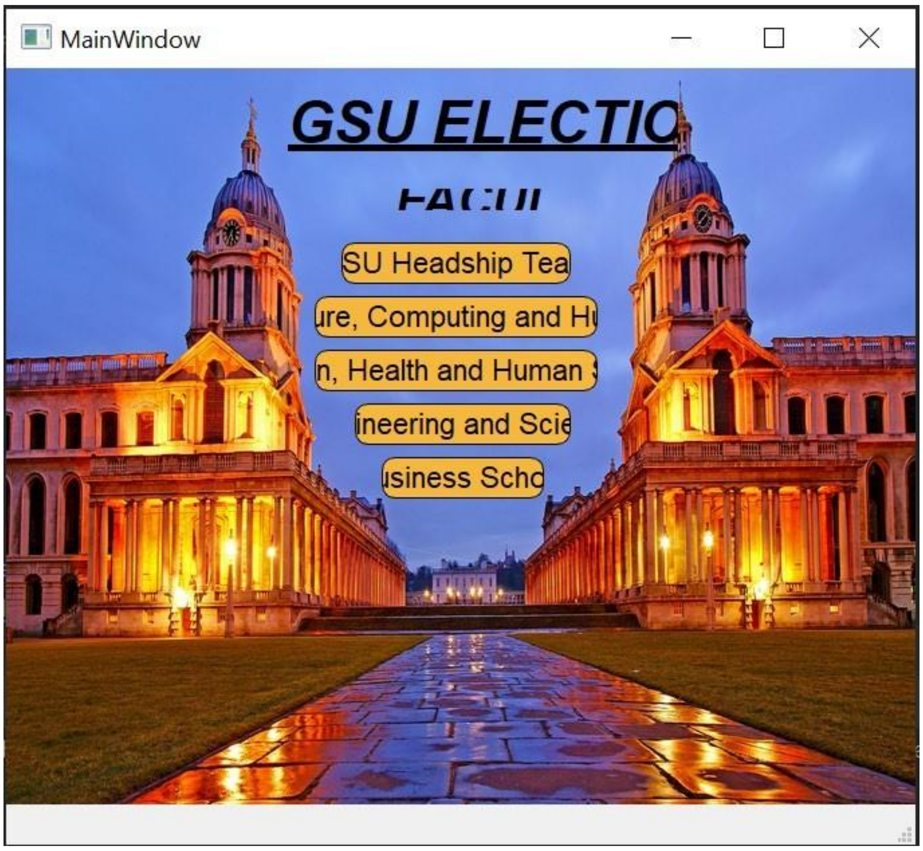
president.py: The second problem was due to syntax error, a literal "**is**" in the code bits was changed into programming language "**==**" to make the file runnable (Picture 6.4.).



```
C:\Users\ghanh\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/ghanh/Downloads/final_course2/presidents.py
C:/Users/ghanh/Downloads/final_course2/presidents.py:100: SyntaxWarning: "is" with a literal. Did you mean "=="?
  if score is 5:#if score is None(dont do anything)
1 Alex smith
2 Martha red
1 Fede rico
2 Peter Smith
```

Picture 6.4.

electionresults.py: The program seemed to work perfectly until I came to a problem that all the buttons representing for GSU Headship team and other 4 faculties, which were made of PyQt5 Designer for GUI, were not connected to the code. Although affords were made, unfortunately, I could not come over this error (Picture 6.5.).



Picture 6.5.

Not adjustable screen: The last error was for the GUI could not adopt to different screen resolution, which can be seen in picture 6.3. and 6.5 which I found out by testing but unable to adjust the code.

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)*

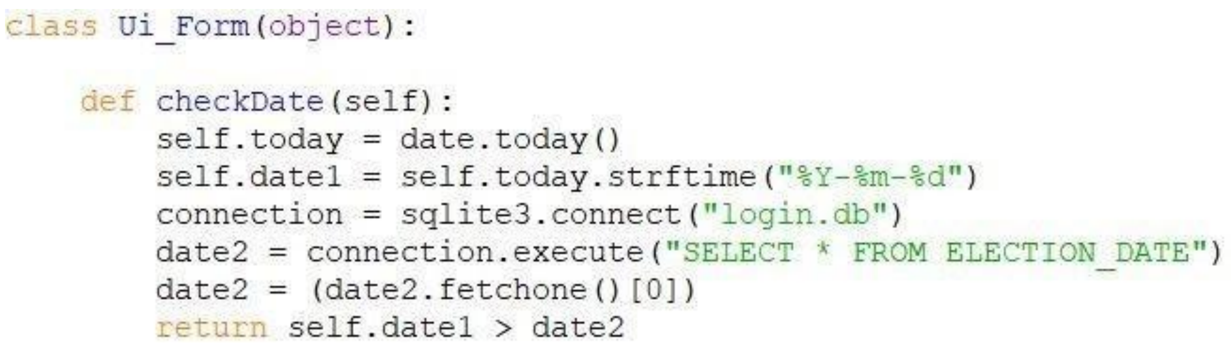
7.1 Feature A.1 – Login.dB

The file local variable values are specified in the first line with UTF-8 (8-bit Unicode Transformation Format). Then the main PyQt components, widgets, real-time/date, the Redis cache, SQLite3, which is a full-featured, fast and reliable SQL database engine, System-specific parameters and functions (sys), PyQt5 – set of Python buildings which enables Python to be used as an alternative application development language and QMessageBox Class are imported.

We create an object Ui from the User Interface Form Class.

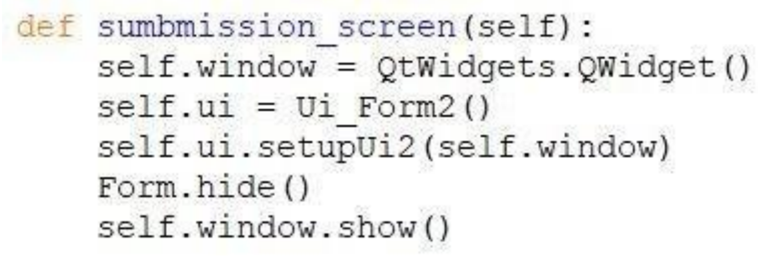
These are all subclasses of the parent class.

This is a function, determining the real date (year/month/day) and time today. We call the instances of the class. The **sqlite3** command makes the connection between the SQL data and the program. The bit **SELECT * FROM ELECTION_DATE** calls is the data records we call. The **fetchone** method returns a single record. Then we end the function and returns the result.



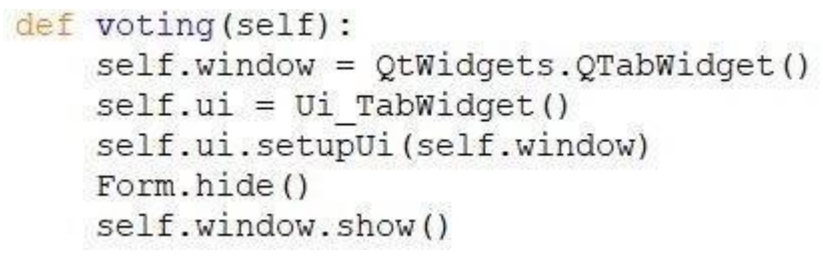
```
class Ui_Form(object):

    def checkDate(self):
        self.today = date.today()
        self.date1 = self.today.strftime("%Y-%m-%d")
        connection = sqlite3.connect("login.db")
        date2 = connection.execute("SELECT * FROM ELECTION_DATE")
        date2 = (date2.fetchone()[0])
        return self.date1 > date2
```

Defining submission screen function and access the attributes and methods of the class. We inherit the **QtWidgets.QWidget()** and make it the subclass type. Then the user interface form is specified. We include the **ui.setup** method.



```
def sunbmission_screen(self):
    self.window = QtWidgets.QWidget()
    self.ui = Ui_Form2()
    self.ui.setupUi2(self.window)
    Form.hide()
    self.window.show()
```

Here we define function voting and again call the attributes of the class. This subclass has the same methods as the **submission_screen** class. With the Qt widgets we create the window

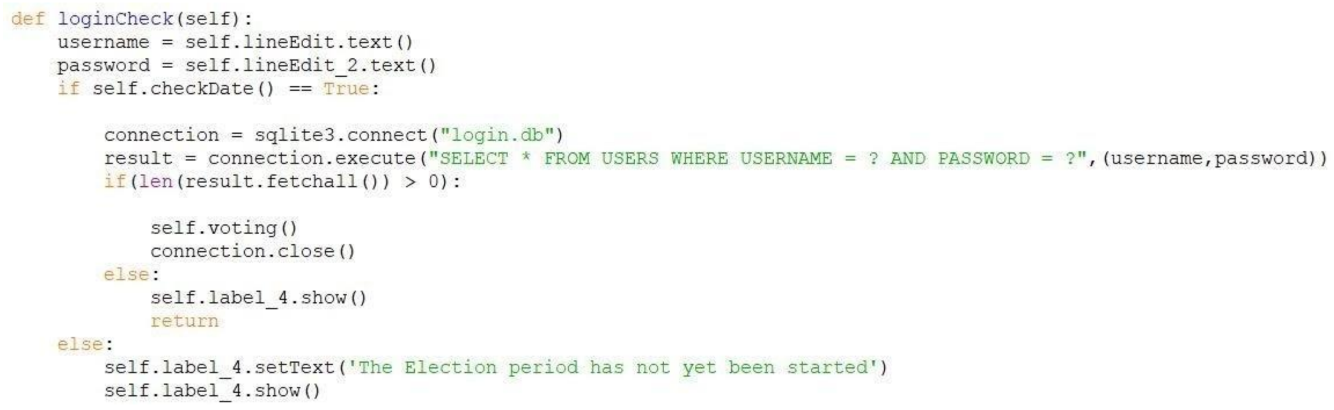


```
def voting(self):
    self.window = QtWidgets.QTabWidget()
    self.ui = Ui_TabWidget()
    self.ui.setupUi(self.window)
    Form.hide()
    self.window.show()
```

In the code bits below, we are defining a function that checks if the login username or password are correct with a comparison of the text file with all the data. If the information is correct, the connection is made(sqliet3) and then the program selects users from the SQL database. The **fetchall** command fetches all the rows of data. Then the connection between loginCheck subclass and Voting subclass is successful and the user is taken to the voting section.

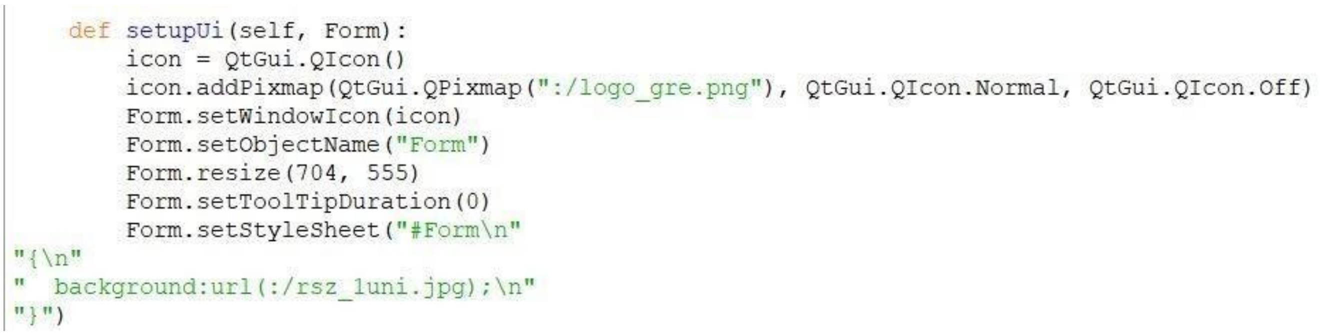
If not, it sends a message that the election hasn't started yet or that the StudentID or password is incorrect.

;



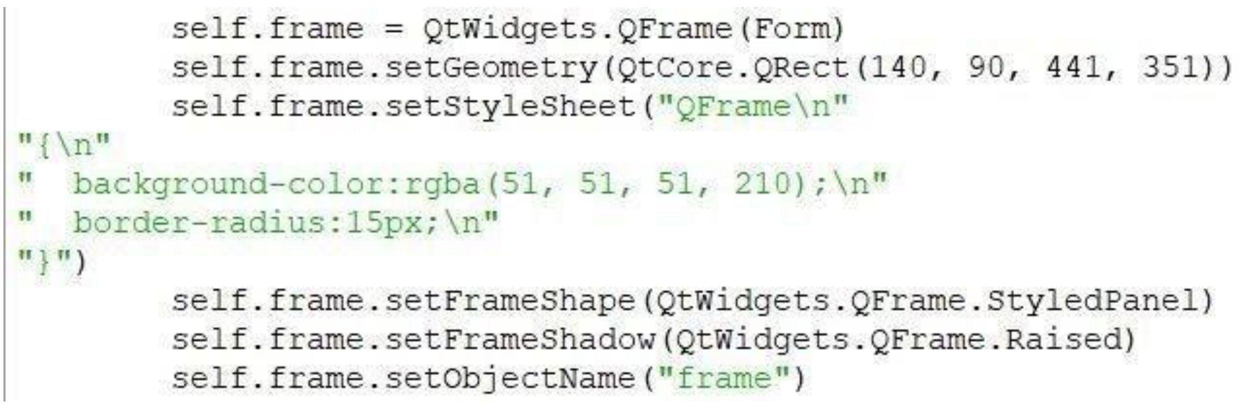
```
def loginCheck(self):
    username = self.lineEdit.text()
    password = self.lineEdit_2.text()
    if self.checkDate() == True:

        connection = sqlite3.connect("login.db")
        result = connection.execute("SELECT * FROM USERS WHERE USERNAME = ? AND PASSWORD = ?",(username,password))
        if(len(result.fetchall()) > 0):

            self.voting()
            connection.close()
        else:
            self.label_4.show()
            return
    else:
        self.label_4.setText('The Election period has not yet been started')
        self.label_4.show()
```

The subclass below we create a function that setup the User Interface, which creates a widget object. First, we make a **pixmap** using **QtGui.QIcon.** We set a window icon, name - **"Form"** and size of the pixmap. **ToolTipDuration** specifies the duration of displaying the tooltip, in milliseconds, which in our case will be shown immediately. The background photo of the login window is set at the end.



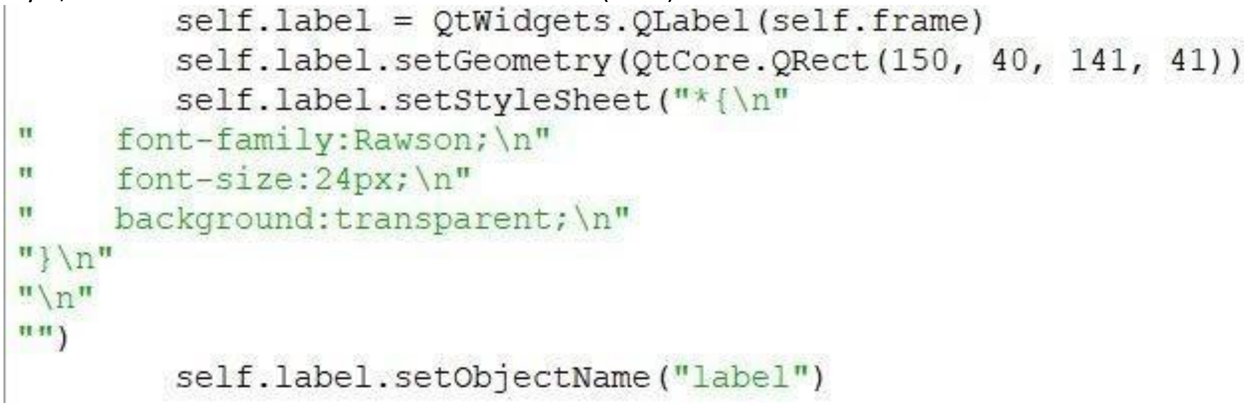
```
    def setupUi(self, Form):
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap(":/logo_gre.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
        Form.setWindowIcon(icon)
        Form.setObjectName("Form")
        Form.resize(704, 555)
        Form.setToolTipDuration(0)
        Form.setStyleSheet("#Form\n"
"{\n"
"  background:url(:/rsz_1uni.jpg);\n"
"}")
```

The code below specifies all the characteristics of the window frame and it's created with the Qt Designer. Using the Qt widgets, as the methods and attributes of the class Form are accessed, the geometry size, style - background image, its size and border-radius, shape, shadow and name(frame) of the window is set up.



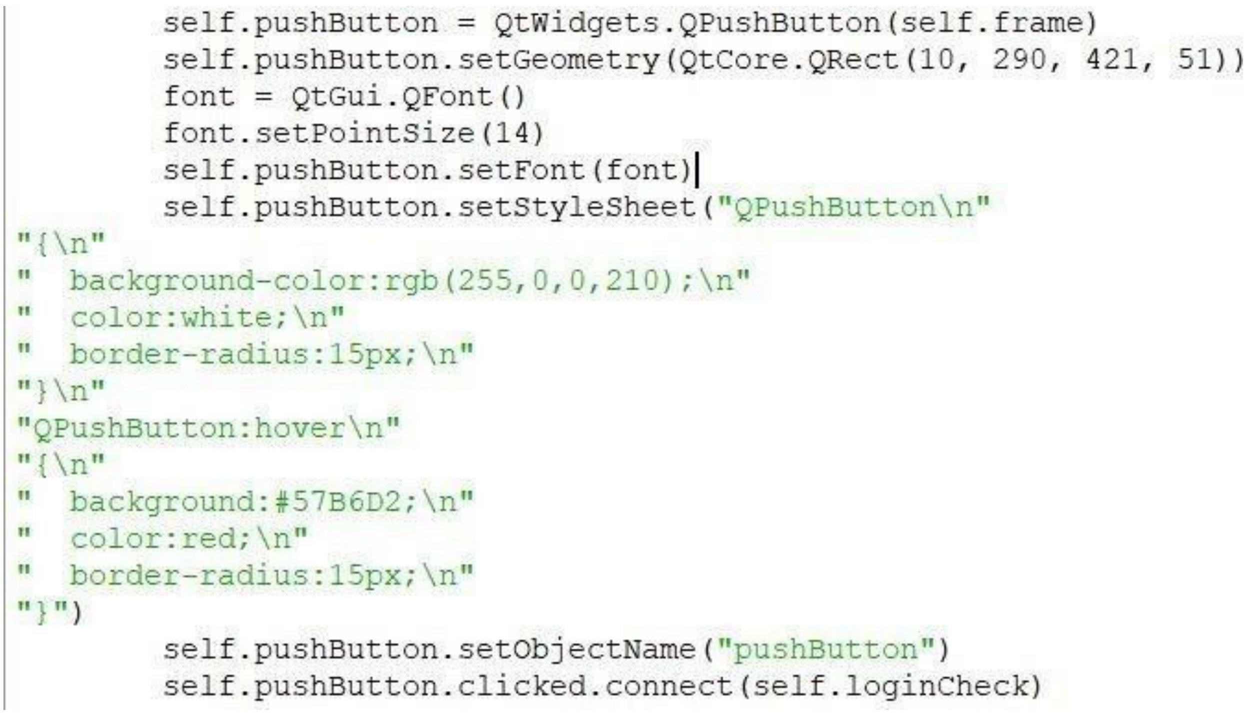
```
        self.frame = QtWidgets.QFrame(Form)
        self.frame.setGeometry(QtCore.QRect(140, 90, 441, 351))
        self.frame.setStyleSheet("QFrame\n"
"{\n"
"  background-color:rgba(51, 51, 51, 210);\n"
"  border-radius:15px;\n"
"}")
        self.frame.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.frame.setFrameShadow(QtWidgets.QFrame.Raised)
        self.frame.setObjectName("frame")
```

The code below specifies the label(variable) of the frame. Using the widgets, we set the size of it, the font's style, size and its colour and the name of the label(label).



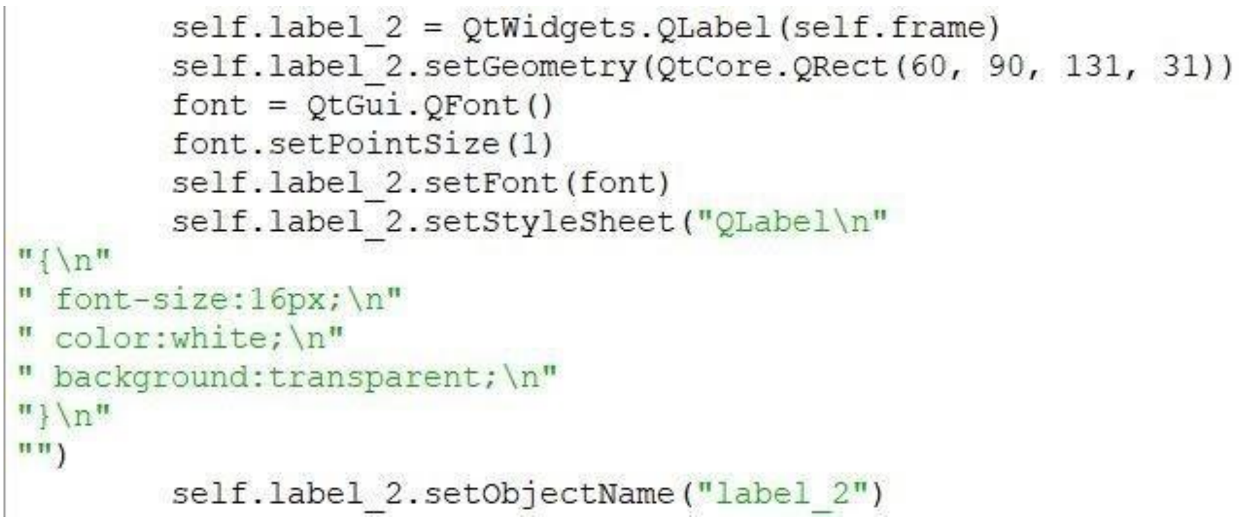
```
        self.label = QtWidgets.QLabel(self.frame)
        self.label.setGeometry(QtCore.QRect(150, 40, 141, 41))
        self.label.setStyleSheet("*{\n"
"    font-family:Rawson;\n"
"    font-size:24px;\n"
"    background:transparent;\n"
"}\n"
"\n"
"")
        self.label.setObjectName("label")
```

The code bits below define a variable button called Push Button, which is contained in the frame window called through the Qt widgets. The sizes of the button are set, the font style and size. As the background white colour of the button and 15 pixels border-radius. Then we set the same properties but for the button when the user is hovering on it. When the button is clicked it connects with the Login Check function.



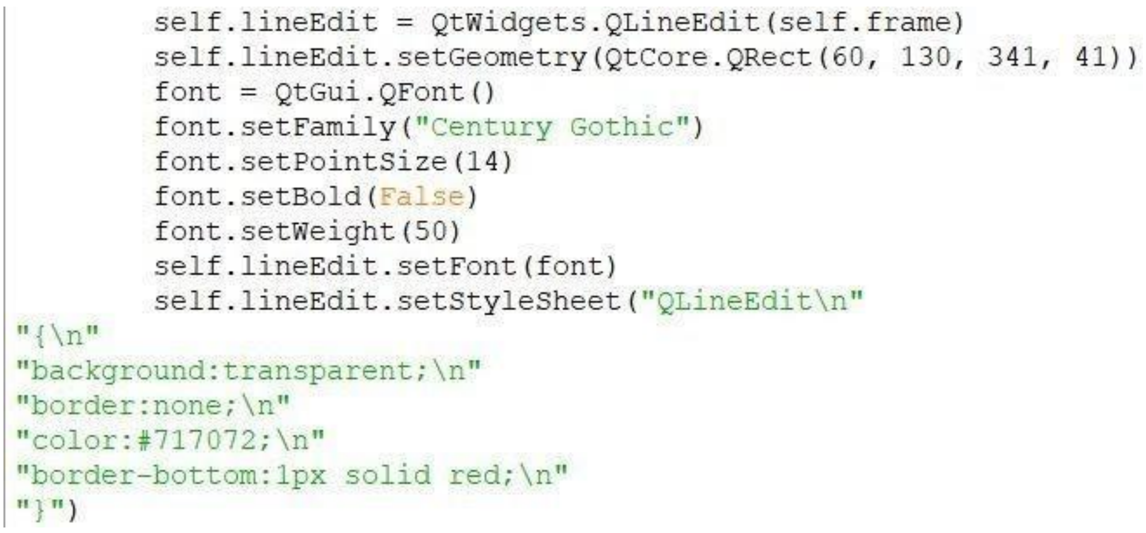
```
        self.pushButton = QtWidgets.QPushButton(self.frame)
        self.pushButton.setGeometry(QtCore.QRect(10, 290, 421, 51))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.pushButton.setFont(font)
        self.pushButton.setStyleSheet("QPushButton\n"
"{\n"
"  background-color:rgb(255,0,0,210);\n"
"  color:white;\n"
"  border-radius:15px;\n"
"}\n"
"QPushButton:hover\n"
"{\n"
"  background:#57B6D2;\n"
"  color:red;\n"
"  border-radius:15px;\n"
"}")
        self.pushButton.setObjectName("pushButton")
        self.pushButton.clicked.connect(self.loginCheck)
```

It follows a piece of code function that creates the second label of the frame. As always, we set the size, the font(16 pixels) and colour(white) of the letters, background image, in this case transparent, and the name of it.



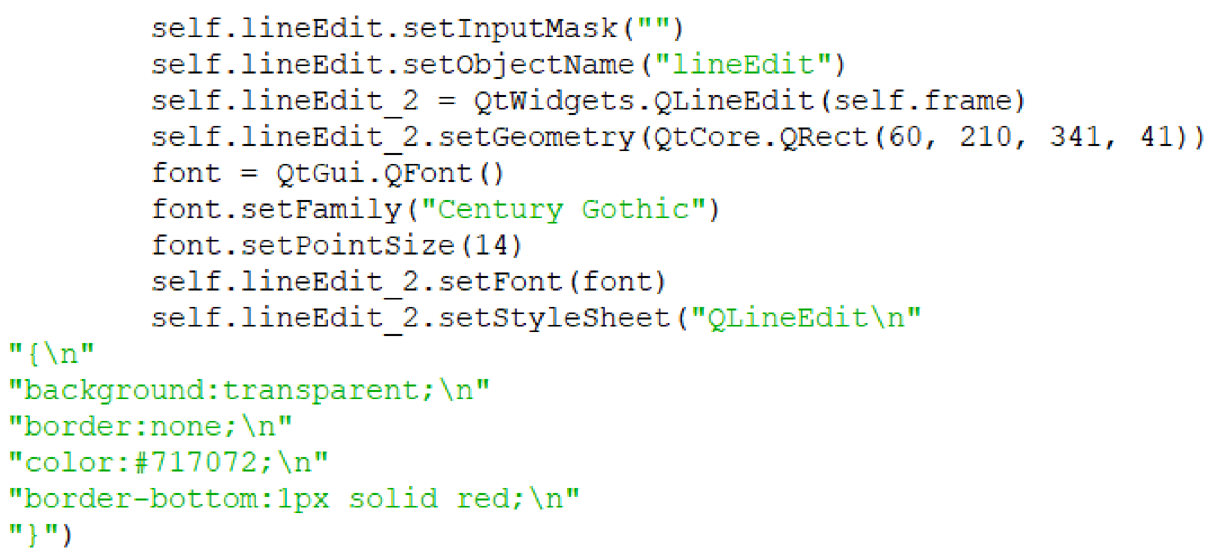
```
        self.label_2 = QtWidgets.QLabel(self.frame)
        self.label_2.setGeometry(QtCore.QRect(60, 90, 131, 31))
        font = QtGui.QFont()
        font.setPointSize(1)
        self.label_2.setFont(font)
        self.label_2.setStyleSheet("QLabel\n"
"{\n"
" font-size:16px;\n"
" color:white;\n"
" background:transparent;\n"
"}\n"
"")
        self.label_2.setObjectName("label_2")
```

A line variable is created, which allows the user to insert and edit a text line. We are working with Qt bindings. The size of the line, font, size and weight of the text are defined and set up. The background is plain, there isn't a border.



```
        self.lineEdit = QtWidgets.QLineEdit(self.frame)
        self.lineEdit.setGeometry(QtCore.QRect(60, 130, 341, 41))
        font = QtGui.QFont()
        font.setFamily("Century Gothic")
        font.setPointSize(14)
        font.setBold(False)
        font.setWeight(50)
        self.lineEdit.setFont(font)
        self.lineEdit.setStyleSheet("QLineEdit\n"
"{\n"
"background:transparent;\n"
"border:none;\n"
"color:#717072;\n"
"border-bottom:1px solid red;\n"
"}")
```

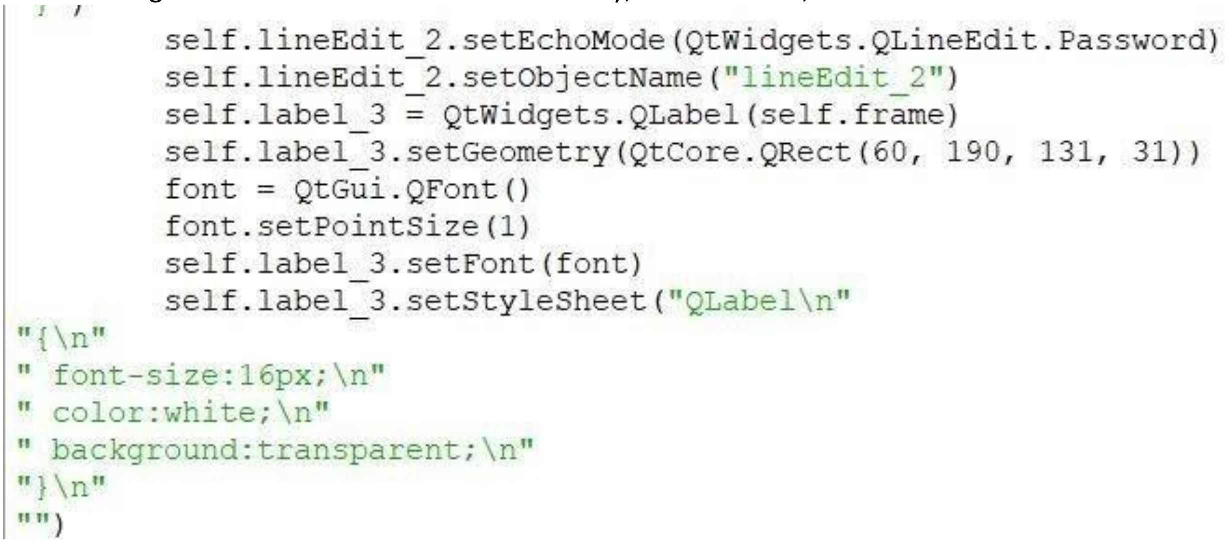
Then we pass an empty string and return to normal QlineEdit operation. The variable below creates a second text line(LineEdit_2), we are using the Qt widgets to connect the frame with the line. The size of the line, font(Century Gothic) and size(14) of the text are established. As well as the transparent background, no border and the extra colours we set



```
            self.lineEdit.setInputMask("")
            self.lineEdit.setObjectName("lineEdit")
            self.lineEdit_2 = QtWidgets.QLineEdit(self.frame)
            self.lineEdit_2.setGeometry(QtCore.QRect(60, 210, 341, 41))
            font = QtGui.QFont()
            font.setFamily("Century Gothic")
            font.setPointSize(14)
            self.lineEdit_2.setFont(font)
            self.lineEdit_2.setStyleSheet("QLineEdit\n"
    "{\n"
    "background:transparent;\n"
    "border:none;\n"
    "color:#717072;\n"
    "border-bottom:1px solid red;\n"
    "}")
```

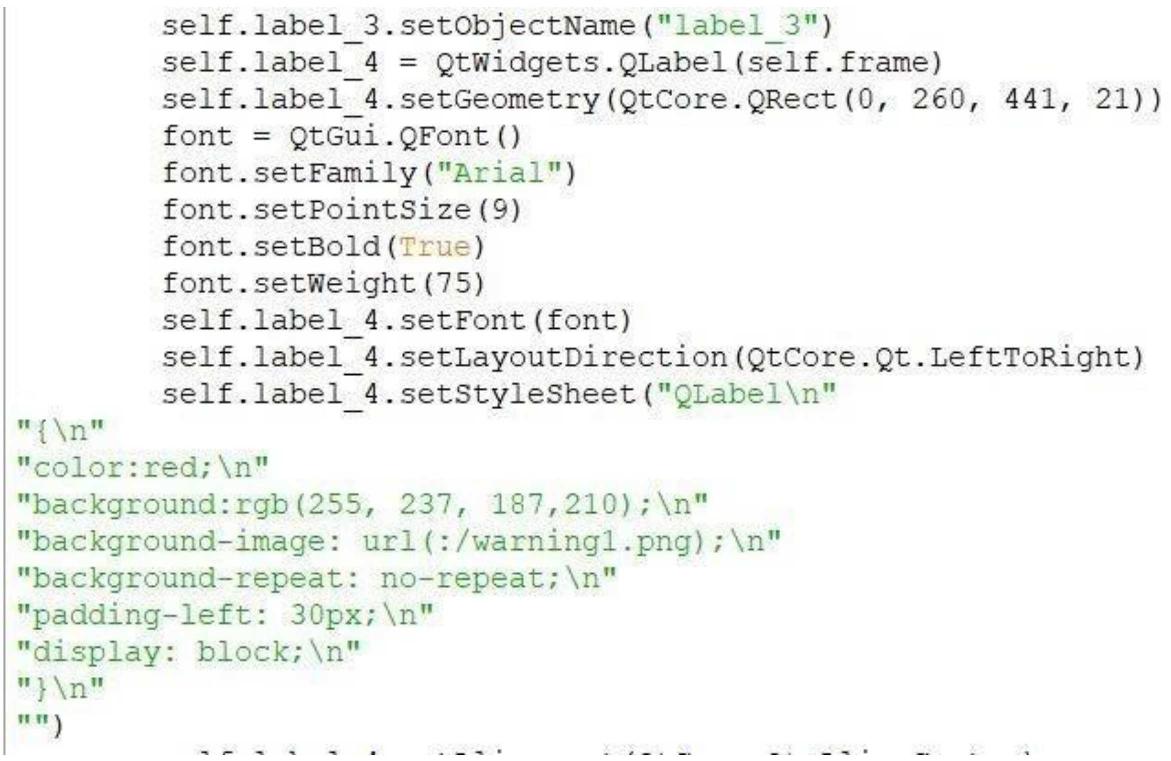
up.

This same text line is purposed for the user password, called from the SQL database through the Qt widgets. We set up Echo mode, which determines how the text in the line is displayed; we name the line and then we start creating the third label in the frame. Geometry, fonts and size, colours are established.



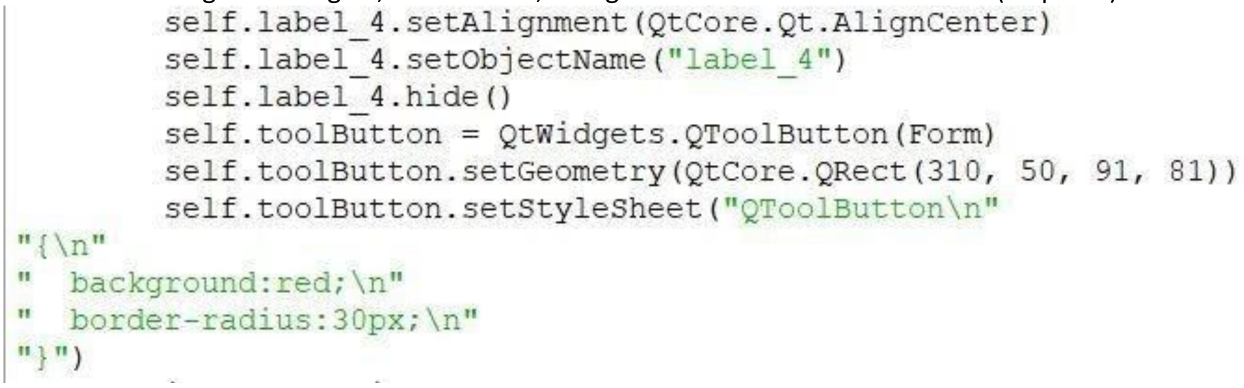
```
        self.lineEdit_2.setEchoMode(QtWidgets.QLineEdit.Password)
        self.lineEdit_2.setObjectName("lineEdit_2")
        self.label_3 = QtWidgets.QLabel(self.frame)
        self.label_3.setGeometry(QtCore.QRect(60, 190, 131, 31))
        font = QtGui.QFont()
        font.setPointSize(1)
        self.label_3.setFont(font)
        self.label_3.setStyleSheet("QLabel\n"
"{\n"
" font-size:16px;\n"
" color:white;\n"
" background:transparent;\n"
"}\n"
"")
```

We create the name of the third label and create the next label. The same characteristics follow with small differences in the size, font and weight of the text. The direction of the layout is set to move left to right and the background image, image sizes and other colours are set.



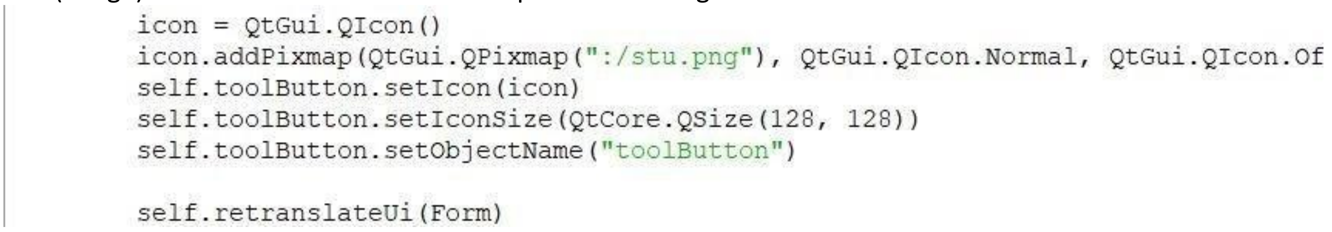
```
        self.label_3.setObjectName("label_3")
        self.label_4 = QtWidgets.QLabel(self.frame)
        self.label_4.setGeometry(QtCore.QRect(0, 260, 441, 21))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(9)
        font.setBold(True)
        font.setWeight(75)
        self.label_4.setFont(font)
        self.label_4.setLayoutDirection(QtCore.Qt.LeftToRight)
        self.label_4.setStyleSheet("QLabel\n"
"{\n"
"color:red;\n"
"background:rgb(255, 237, 187,210);\n"
"background-image: url(:/warning1.png);\n"
"background-repeat: no-repeat;\n"
"padding-left: 30px;\n"
"display: block;\n"
"}\n"
"")
```

We continue to configure the label, align the label through the Qt widgets and the Qt Align Center. We create the name of the label and then set to be hidden.
In the meantime, we create a new button called toolButton. Inherit the attributes and methods of the main class Form through the widgets, the size of it, background colour and border-radius(30 pixels)



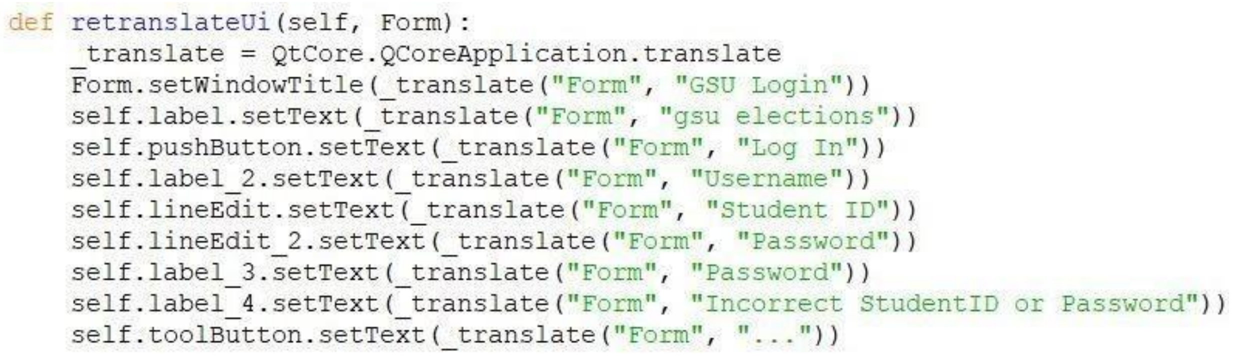
```
        self.label_4.setAlignment(QtCore.Qt.AlignCenter)
        self.label_4.setObjectName("label_4")
        self.label_4.hide()
        self.toolButton = QtWidgets.QToolButton(Form)
        self.toolButton.setGeometry(QtCore.QRect(310, 50, 91, 81))
        self.toolButton.setStyleSheet("QToolButton\n"
"{\n"
"  background:red;\n"
"  border-radius:30px;\n"
"}")
```

A new variable icon is created using the Qt and a pixmap is created using the **QtGui.QIcon.** Then we add an icon(image) to the toolButton and we set up the size. Using the retranslateUi we set the text of the UI Form.



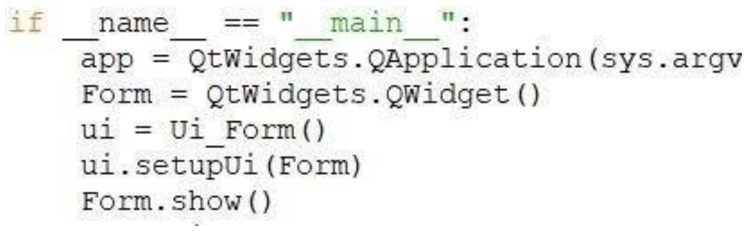
```
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap(":/stu.png"), QtGui.QIcon.Normal, QtGui.QIcon.Of
        self.toolButton.setIcon(icon)
        self.toolButton.setIconSize(QtCore.QSize(128, 128))
        self.toolButton.setObjectName("toolButton")

        self.retranslateUi(Form)
```

The defined function below translates the object’s text from all the variables in the Ui Form. Basically, the titles and text of the widgets are set to the right button, label, line etc.



```
def retranslateUi(self, Form):
    _translate = QtCore.QCoreApplication.translate
    Form.setWindowTitle(_translate("Form", "GSU Login"))
    self.label.setText(_translate("Form", "gsu elections"))
    self.pushButton.setText(_translate("Form", "Log In"))
    self.label_2.setText(_translate("Form", "Username"))
    self.lineEdit.setText(_translate("Form", "Student ID"))
    self.lineEdit_2.setText(_translate("Form", "Password"))
    self.label_3.setText(_translate("Form", "Password"))
    self.label_4.setText(_translate("Form", "Incorrect StudentID or Password"))
    self.toolButton.setText(_translate("Form", "..."))
```

After Python interpreter reads the source files and the variables are defined, we create a special_name variable with a value_main and then the program it executes all the codes contained in the file.



```
if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv
    Form = QtWidgets.QWidget()
    ui = Ui_Form()
    ui.setupUi(Form)
    Form.show()
```

7.2 Feature A.2 – Voting_system.py

In the file named **Voting_system.py**, a form implementation generated from PyQt5 UI code is provided to create a well-organised voting table, which is linked to **login.py**. This table gives each voter a list of candidates among Greenwich Student Union (GSU) and 4 faculties with different positions such as president, officer, etc. after they succeed on login to the system through their unique username and correct password.
We create a class, **Ui_TabWidget**, to bundle data and functionality together, in which voting form and table window were designed and connected to other files.

```
class Ui_TabWidget(object):
    def setupUi(self, TabWidget):
        TabWidget.setObjectName("TabWidget")
        TabWidget.resize(444, 293)
        self.window = QtWidgets.QWidget()
        self.uii = Ui_Form()
        self.uii.setupUi(self.window)
        self.window.setObjectName("tab1")
        TabWidget.addTab(self.window, "")
        #TabWidget.insertTab(1, self.window, "")

        self.retranslateUi(TabWidget)
        QtCore.QMetaObject.connectSlotsByName(TabWidget)
```

Function **retranslateUi** is used to define interface of the voting system. With the use of **QtCore** and **QtCoreApplication** classes, loops of main events are provided. These two classes not only contain event loops, but they are also responsible for handling the internationalisation and translations by using **translate()** application string. All of function and classes are based on PyQt, including the class **TabWidget** inheriting from **QtCore**.

```
    def retranslateUi(self, TabWidget):
        _translate = QtCore.QCoreApplication.translate
        TabWidget.setWindowTitle(_translate("TabWidget", "Voting System"))
        TabWidget.setTabText(TabWidget.indexOf(self.window), _translate("TabWidget", "Presidents"))
```

The code bits below are using a special variable**__name__** so as to check and only execute all the code found in "main" file, which was assigned as hard-code string "**_main_**". With the **QtWidgets.Application(sys.argv)**, we enable **QtWidgets.Application** wizard to contain command line arguments (**sys.argv**), which is already imported at the beginning of the code file. Also, **sys.exit()** in this case is provided to quit all the application and return the process that runs main loops and checking the code status **app.exec()**.

```
if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    TabWidget = QtWidgets.QTabWidget()
    ui = Ui_TabWidget()
    ui.setupUi(TabWidget)
    TabWidget.show()
    sys.exit(app.exec_())
```

7.3 Feature A.3 – Project_images.py

Variable **qt_version** is defined by applying a **for** loop function to iterate the integer object **v** in the range defined by **QtCore.qVersion().split('.')** method. In the term of the **if** statement implemented inside the loop, Regularised extension of the Canonical Correlation Analysis function is chosen to check the connexions within two data matrices based on the number of columns and rows.

```
qt_version = [int(v) for v in QtCore.qVersion().split('.')]
if qt_version < [5, 8, 0]:
    rcc_version = 1
    qt_resource_struct = qt_resource_struct_v1
else:
    rcc_version = 2
    qt_resource_struct = qt_resource_struct_v2

def qInitResources():
    QtCore.qRegisterResourceData(rcc_version, qt_resource_struct, qt_resource_name, qt_resource_data)

def qCleanupResources():
    QtCore.qUnregisterResourceData(rcc_version, qt_resource_struct, qt_resource_name, qt_resource_data)

qInitResources()
```

Functions **qInitResources()** and **qCleanupResources()** use their sub static functions **registerResource()** and **unregisterResource()** respectively. The **registerResource()** plays a role in leaving a resource out when it is called at run-time, which is a binary-formed resource created by **rcc**. On the other hand, unregisterResource() function is in charge of abolishing a reference to a given file.

```
qt_version = [int(v) for v in QtCore.qVersion().split('.')]
if qt_version < [5, 8, 0]:
    rcc_version = 1
    qt_resource_struct = qt_resource_struct_v1
else:
    rcc_version = 2
    qt_resource_struct = qt_resource_struct_v2

def qInitResources():
    QtCore.qRegisterResourceData(rcc_version, qt_resource_struct, qt_resource_name, qt_resource_data)

def qCleanupResources():
    QtCore.qUnregisterResourceData(rcc_version, qt_resource_struct, qt_resource_name, qt_resource_data)

qInitResources()
```

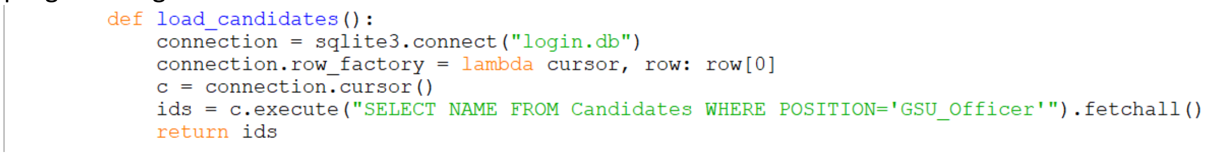
All of the code mentioned is used mainly for the GUI part of the voting system.

7.4 FEATURE A.4 – PRESIDENTS.PY, BUSINESS_SCHOOL.PY

This code part is mainly for the frontend content with the implementation of *PyQt*. The class below is to create a graphic user interface (GUI) by establishing a form which creates and allows to name tables, creates buttons, data fields (columns) and record of data (row) from the information taken from voting system file.

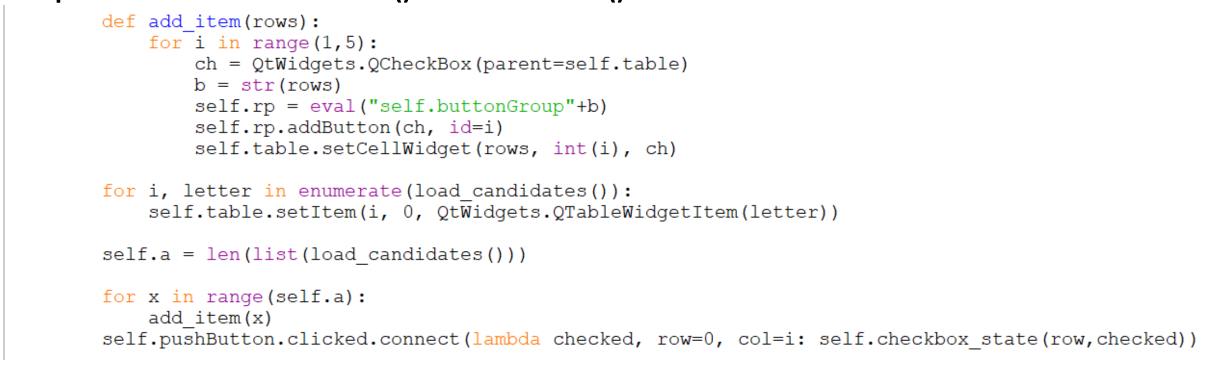
```
class Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(400, 300)
        # create table:
        self.buttonGroup0 = QtWidgets.QButtonGroup(Form)
        self.buttonGroup1 = QtWidgets.QButtonGroup(Form)
        self.buttonGroup2 = QtWidgets.QButtonGroup(Form)
        self.buttonGroup3 = QtWidgets.QButtonGroup(Form)
        self.buttonGroup4 = QtWidgets.QButtonGroup(Form)
        self.table = QtWidgets.QTableWidget(Form)
        self.table.setGeometry(QtCore.QRect(30, 10, 381, 191))
        self.table.setRowCount(4)
        self.table.setColumnCount(5)
        #Add title
        item = QtWidgets.QTableWidgetItem('Candidates')
        self.table.setHorizontalHeaderItem(0,item)
        item = QtWidgets.QTableWidgetItem('1')
        self.table.setHorizontalHeaderItem(1,item)
        item = QtWidgets.QTableWidgetItem('2')
        self.table.setHorizontalHeaderItem(2,item)
        item = QtWidgets.QTableWidgetItem('3')
        self.table.setHorizontalHeaderItem(3,item)
        item = QtWidgets.QTableWidgetItem('4')
        self.table.setHorizontalHeaderItem(4,item)
        QtCore.QMetaObject.connectSlotsByName(Form)
        #pushButton
        self.pushButton = QtWidgets.QPushButton("Sumbit Vote",Form)
        self.pushButton.setGeometry(QtCore.QRect(160, 230, 111, 31))
        self.pushButton.setObjectName("pushButton")
        #label
        self.label = QtWidgets.QLabel(Form)
        self.label.setGeometry(QtCore.QRect(0, 211, 441, 20))
        self.label.setText('Check all the checkbox')
        self.label.hide()
```

The code provided below shows role of the function **load_candidates()**, which has the manner of implementation by linking to a database file named **login.db**. In order to communicate with the database, our group decided to use the Structure Query Language (SQL), particularly connect each row specific position with **lambda** function. What is noticable of this part is that SQL implementations are directly taken into Python's programming.



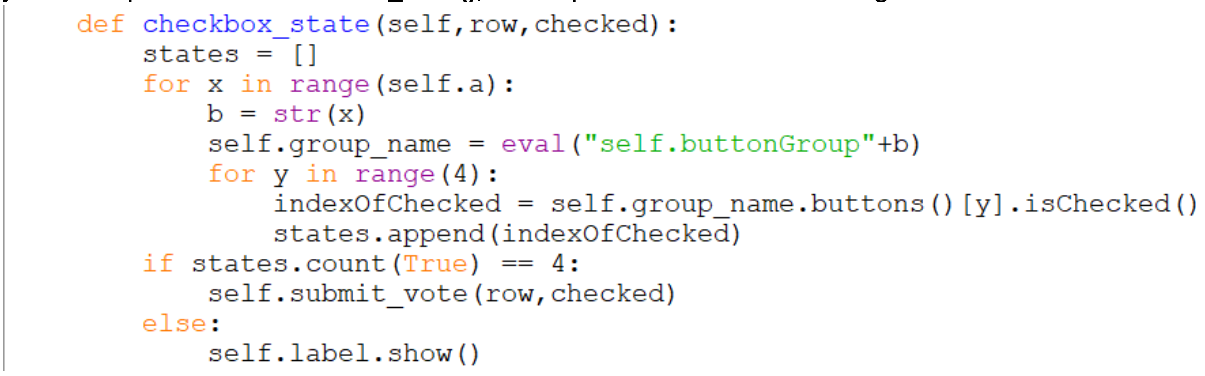
```
def load_candidates():
    connection = sqlite3.connect("login.db")
    connection.row_factory = lambda cursor, row: row[0]
    c = connection.cursor()
    ids = c.execute("SELECT NAME FROM Candidates WHERE POSITION='GSU_Officer'").fetchall()
    return ids
```

Voters can send their vote by choosing each button representing as each candidate and their scores, which appears in the code file with the function defined as **add_item()**. In that function, **for** loop sub-function is used in a range of 4 (**range(1, 5)**) and also selecting button and submit button are connected by using **self.pushButton.clicked.connect()** and **add.button()**.



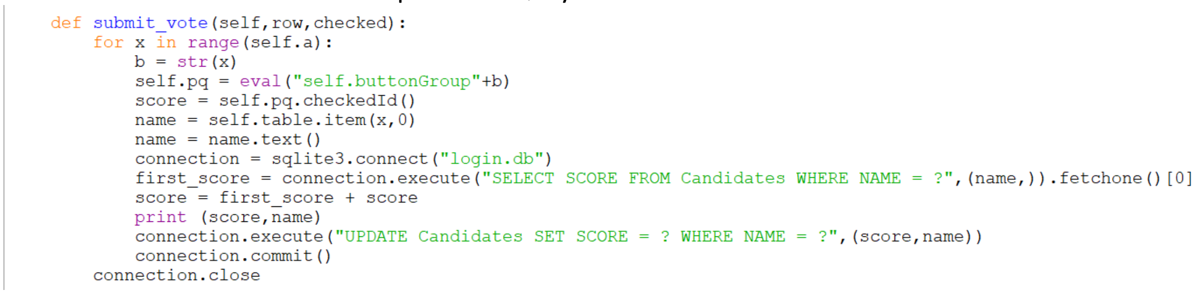
```
def add_item(rows):
    for i in range(1,5):
        ch = QtWidgets.QCheckBox(parent=self.table)
        b = str(rows)
        self.rp = eval("self.buttonGroup"+b)
        self.rp.addButton(ch, id=i)
        self.table.setCellWidget(rows, int(i), ch)

for i, letter in enumerate(load_candidates()):
    self.table.setItem(i, 0, QtWidgets.QTableWidgetItem(letter))

self.a = len(list(load_candidates()))

for x in range(self.a):
    add_item(x)
self.pushButton.clicked.connect(lambda checked, row=0, col=i: self.checkbox_state(row,checked))
```

After voting, we use this code bits below to make sure that voters have already voted for all 4 candidates in each department by a function called **checkbox_state()** before they submit their selection. Sharing the same job as the previous function **add_item()**, **for** loop and **if** statement is being used.



```
def checkbox_state(self,row,checked):
    states = []
    for x in range(self.a):
        b = str(x)
        self.group_name = eval("self.buttonGroup"+b)
        for y in range(4):
            indexOfChecked = self.group_name.buttons()[y].isChecked()
            states.append(indexOfChecked)
    if states.count(True) == 4:
        self.submit_vote(row,checked)
    else:
        self.label.show()
```

Finally, once voting is done, the results will appear via SQL database **login.db** where name and scores of GSU candidates can be viewed after updated. SQL syntax in this code are **SELECT** and **UPDATE**.



```
def submit_vote(self,row,checked):
    for x in range(self.a):
        b = str(x)
        self.pq = eval("self.buttonGroup"+b)
        score = self.pq.checkedId()
        name = self.table.item(x,0)
        name = name.text()
        connection = sqlite3.connect("login.db")
        first_score = connection.execute("SELECT SCORE FROM Candidates WHERE NAME = ?",(name,)).fetchone()[0]
        score = first_score + score
        print (score,name)
        connection.execute("UPDATE Candidates SET SCORE = ? WHERE NAME = ?",(score,name))
        connection.commit()
    connection.close
```

7.5 References

Joseph, L., 2015. *Learning Robotics using Python.* 2nd ed. Birmingham: Packt Publishing Ltd.

Joshi, G., 2020. *Getting Started With PyQt.* [Online]
Available at: https://girishjoshi.io/post/gettingstartedwithpyqt/
[Accessed 27 January 2020].

Qt Group, 2020. *Qt Documentation.* [Online]
Available at: https://doc.qt.io/qt-5/index.html
[Accessed 27 January 2020].

Siahaan, V. & Sianipar, R. H., 2020. *LEARNING PyQt5: A Step by Step Tutorial to Develop MySQL-Based Applications.* 1st ed. London: Sparta Publishing.

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

About the voting system design, it is qualified to meet all requirements. Thanks to the use of PyQt5, we enabled to build clear, vibrant GUIs (especially login UI) as well as tabular forms which is in charge of visualizing the vote election and vote results, allows all the users to use in the simplest way. Features implemented and code reliability, on the other hand, have performed perfectly as our expectation. We accomplished in creating all the features listed in the requirement session (A1-6) that our codes can run and return results correctly despite some bugs mentioned in part 2.1. that we either overcome or not.

8.2 Evaluate you own and your group's performance

For project to process smoothly and on schedule, we combined the group meeting on Monday tutorial along with working from home and communicate with group members via WhatsApp, Skype, etc. to ensure the code quality and seamlessness. Each of our group member held a different work as assigned, contributing to effectively produce the final product. About my own performance, I finished fully the parts that I was assigned.

8.2.1 What went well?

This Python project was going well as can be seen in the final product. All the members of the group completed all the separate and mutual work flowingly and helped others for the best performance. Although we had difficulties in both lack of programming knowledge and experience, we went through and equipped ourselves lessons which are helpful for our further study and future careers.

8.2.2 What went less well?

As we decided to use PyQt5 and SQLite3 which most of the group member had never used before, it was consuming a significant amount of time for us to install and get familiar with these platforms.

8.2.3 What was learnt?

The project gave us a chance to practice our programming knowledge and ability throughout the process of building the voting system, considered as the most important and practical outcome of this course. Second are numbers of skills apart from coding such as finding materials and sources, drawing and jointing each member's contribution together. Lastly and personally, I encountered to new GUI building platform PyQt5 and database platform SQLite3.

8.2.4 How would a similar task be completed differently?

We would like to try and apply different GUI toolkits, Kivy or Tkinter for example, onto building different interfaces for this project.

8.2.5 How could the module be improved?

It will be better to establish other learning activities instead of quizzes that allow students with different coding abilities to work together as not every student is at the same level as the other. By doing that, I believe that upper programming students can help lower ones and may they experience new coding method that they have never experienced before.

[Overall: 423 words]

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
Francesco **001086733**	A.2/A.3	20%	FR	
Han **001052068**	A.2/A.5	20%	HH	
Kamelia **001085165**	A.5/A.6	20%	KI	
Hadil **001050152**	A.3/A.6	20%	HB	
Alex **001068910**	A.1/A.4/A.6	20%	AU	
	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)

1. *Login.py*

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'uni-proj2.ui'
#
# Created by: PyQt5 UI code generator 5.9.2
#
# WARNING! All changes made in this file will be lost!

from PyQt5 import QtCore, QtGui, QtWidgets
from voting_system import Ui_TabWidget
from datetime import date
import project_images
import sqlite3
import sys
from PyQt5.QtWidgets import QMessageBox

class Ui_Form(object):

    def checkDate(self):
        self.today = date.today()
        self.date1 = self.today.strftime("%Y-%m-%d")
        connection = sqlite3.connect("login.db")
        date2 = connection.execute("SELECT * FROM ELECTION_DATE")
        date2 = (date2.fetchone()[0])
        return self.date1 > date2

    def voting(self,user):
        print (user)
        self.window = QtWidgets.QTabWidget()
        self.ui = Ui_TabWidget()
        self.ui.setupUi(self.window, user)
        Form.hide()
        self.window.show()

    def loginCheck(self):
        username = self.lineEdit.text()
        password = self.lineEdit_2.text()
        if self.checkDate() == True:

            connection = sqlite3.connect("login.db")
            result = connection.execute("SELECT * FROM USERS WHERE USERNAME = ? AND PASSWORD =
            ?",(username,password))
            if(len(result.fetchall()) > 0):

                self.voting(username)
                connection.close()
            else:
                self.label_4.show()
                return
            else:
                self.label_4.setText('The Election period has not yet been started')
                self.label_4.show()

    def setupUi(self, Form):
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap(":/logo_gre.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
        Form.setWindowIcon(icon)
        Form.setObjectName("Form")
        Form.resize(704, 555)
        Form.setToolTipDuration(0)
        Form.setStyleSheet("#Form\n"
"{\n"
"  background:url(:/rsz_1uni.jpg);\n"
"}")
        self.frame = QtWidgets.QFrame(Form)
        self.frame.setGeometry(QtCore.QRect(140, 90, 441, 351))
        self.frame.setStyleSheet("QFrame\n"
"{\n"
"  background-color:rgba(51, 51, 51, 210);\n"
"  border-radius:15px;\n"
"}")
        self.frame.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.frame.setFrameShadow(QtWidgets.QFrame.Raised)
        self.frame.setObjectName("frame")
        self.label = QtWidgets.QLabel(self.frame)
        self.label.setGeometry(QtCore.QRect(150, 40, 141, 41))
        self.label.setStyleSheet("*{\n"
"    font-family:Rawson;\n"
"    font-size:24px;\n"
"    background:transparent;\n"
"}\n"
"\n"
"")
        self.label.setObjectName("label")
        self.pushButton = QtWidgets.QPushButton(self.frame)
        self.pushButton.setGeometry(QtCore.QRect(10, 290, 421, 51))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.pushButton.setFont(font)
        self.pushButton.setStyleSheet("QPushButton\n"
"{\n"
"  background-color:rgb(255,0,0,210);\n"
"  color:white;\n"
"  border-radius:15px;\n"
"}\n"
"QPushButton:hover\n"
"{\n"
"  background:#57B6D2;\n"
"  color:red;\n"
"  border-radius:15px;\n"
"}")
```

```
        self.pushButton.setObjectName("pushButton")
        self.pushButton.clicked.connect(self.loginCheck)
        self.label_2 = QtWidgets.QLabel(self.frame)
        self.label_2.setGeometry(QtCore.QRect(60, 90, 131, 31))
        font = QtGui.QFont()
        font.setPointSize(1)
        self.label_2.setFont(font)
        self.label_2.setStyleSheet("QLabel\n"
"{\n"
" font-size:16px;\n"
" color:white;\n"
" background:transparent;\n"
"}\n"
"")
        self.label_2.setObjectName("label_2")
        self.lineEdit = QtWidgets.QLineEdit(self.frame)
        self.lineEdit.setGeometry(QtCore.QRect(60, 130, 341, 41))
        font = QtGui.QFont()
        font.setFamily("Century Gothic")
        font.setPointSize(14)
        font.setBold(False)
        font.setWeight(50)
        self.lineEdit.setFont(font)
        self.lineEdit.setStyleSheet("QLineEdit\n"
"{\n"
"background:transparent;\n"
"border:none;\n"
"color:#717072;\n"
"border-bottom:1px solid red;\n"
"}")
        self.lineEdit.setInputMask("")
        self.lineEdit.setObjectName("lineEdit")
        self.lineEdit_2 = QtWidgets.QLineEdit(self.frame)
        self.lineEdit_2.setGeometry(QtCore.QRect(60, 210, 341, 41))
        font = QtGui.QFont()
        font.setFamily("Century Gothic")
        font.setPointSize(14)
        self.lineEdit_2.setFont(font)
        self.lineEdit_2.setStyleSheet("QLineEdit\n"
"{\n"
"background:transparent;\n"
"border:none;\n"
"color:#717072;\n"
"border-bottom:1px solid red;\n"
"}")
        self.lineEdit_2.setEchoMode(QtWidgets.QLineEdit.Password)
        self.lineEdit_2.setObjectName("lineEdit_2")
        self.label_3 = QtWidgets.QLabel(self.frame)
        self.label_3.setGeometry(QtCore.QRect(60, 190, 131, 31))
        font = QtGui.QFont()
        font.setPointSize(1)
        self.label_3.setFont(font)
        self.label_3.setStyleSheet("QLabel\n"
"{\n"
" font-size:16px;\n"
" color:white;\n"
" background:transparent;\n"
"}\n"
"")
        self.label_3.setObjectName("label_3")
        self.label_4 = QtWidgets.QLabel(self.frame)
        self.label_4.setGeometry(QtCore.QRect(0, 260, 441, 21))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(9)
        font.setBold(True)
        font.setWeight(75)
        self.label_4.setFont(font)
        self.label_4.setLayoutDirection(QtCore.Qt.LeftToRight)
        self.label_4.setStyleSheet("QLabel\n"
"{\n"
"color:red;\n"
"background:rgb(255, 237, 187,210);\n"
"background-image: url(:/warning1.png);\n"
"background-repeat: no-repeat;\n"
"padding-left: 30px;\n"
"display: block;\n"
"}\n"
"")
        self.label_4.setAlignment(QtCore.Qt.AlignCenter)
        self.label_4.setObjectName("label_4")
        self.label_4.hide()
        self.toolButton = QtWidgets.QToolButton(Form)
        self.toolButton.setGeometry(QtCore.QRect(310, 50, 91, 81))
        self.toolButton.setStyleSheet("QToolButton\n"
"{\n"
"  background:red;\n"
"  border-radius:30px;\n"
"}")
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap(":/stu.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.toolButton.setIcon(icon)
        self.toolButton.setIconSize(QtCore.QSize(128, 128))
        self.toolButton.setObjectName("toolButton")

        self.retranslateUi(Form)
        QtCore.QMetaObject.connectSlotsByName(Form)


    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "GSU Login"))
        self.label.setText(_translate("Form", "gsu elections"))
        self.pushButton.setText(_translate("Form", "Log In"))
        self.label_2.setText(_translate("Form", "Username"))
        self.lineEdit.setText(_translate("Form", "Student ID"))
        self.lineEdit_2.setText(_translate("Form", "Password"))
        self.label_3.setText(_translate("Form", "Password"))
        self.label_4.setText(_translate("Form", "Incorrect StudentID or Password"))
        self.toolButton.setText(_translate("Form", "..."))

if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    Form = QtWidgets.QWidget()
    ui = Ui_Form()
    ui.setupUi(Form)
    Form.show()
    sys.exit(app.exec_())
```

2. *Voting_system.py*

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'tab_wid.ui'
#
# Created by: PyQt5 UI code generator 5.13.2
#
# WARNING! All changes made in this file will be lost!


from PyQt5 import QtCore, QtGui, QtWidgets
import sys
from presidents import Ui_Form


class Ui_TabWidget(object):
    def setupUi(self, TabWidget, user):
        self.username = user
        TabWidget.setObjectName("TabWidget")
        TabWidget.resize(444, 293)
        self.window = QtWidgets.QWidget()
        self.uii = Ui_Form()
        self.uii.setupUi(self.window,self.username)
        self.window.setObjectName("tab1")
        TabWidget.addTab(self.window, "")
        #TabWidget.insertTab(1, self.window, "")

        self.retranslateUi(TabWidget)
        QtCore.QMetaObject.connectSlotsByName(TabWidget)

    def retranslateUi(self, TabWidget):
        _translate = QtCore.QCoreApplication.translate
        TabWidget.setWindowTitle(_translate("TabWidget", "Voting System"))
        TabWidget.setTabText(TabWidget.indexOf(self.window), _translate("TabWidget", "Presidents"))
```

3. *Electionresults.py*

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'electionresults.ui'
#
# Created by: PyQt5 UI code generator 5.13.2
```

```
#
# WARNING! All changes made in this file will be lost!


from PyQt5 import QtCore, QtGui, QtWidgets
import project_images

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(678, 579)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.label = QtWidgets.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(-10, 0, 711, 591))
        self.label.setStyleSheet("background-image: url(:/rsz_1uni.jpg);\n"
"background-image: url(:/rsz_1uni.jpg);")
        self.label.setText("")
        self.label.setObjectName("label")
        self.pushButton_7 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_7.setGeometry(QtCore.QRect(230, 210, 211, 31))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(11)
        font.setBold(False)
        font.setWeight(50)
        self.pushButton_7.setFont(font)
        self.pushButton_7.setStyleSheet("background-color: rgb(246, 184, 61);\n"
"border-style:outset;\n"
"border-width:1px;\n"
"border-radius:10px;\n"
"border-color: rgb(0, 0, 0);")
        self.pushButton_7.setAutoExclusive(False)
        self.pushButton_7.setObjectName("pushButton_7")
        self.label_2 = QtWidgets.QLabel(self.centralwidget)
        self.label_2.setGeometry(QtCore.QRect(210, 10, 291, 61))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(22)
        font.setBold(True)
        font.setItalic(True)
        font.setUnderline(True)
        font.setWeight(75)
        self.label_2.setFont(font)
        self.label_2.setStyleSheet("")
        self.label_2.setObjectName("label_2")
        self.label_3 = QtWidgets.QLabel(self.centralwidget)
        self.label_3.setGeometry(QtCore.QRect(290, 90, 111, 16))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(18)
        font.setBold(True)
        font.setItalic(True)
        font.setWeight(75)
        self.label_3.setFont(font)
        self.label_3.setStyleSheet("")
        self.label_3.setObjectName("label_3")
        self.pushButton_10 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_10.setGeometry(QtCore.QRect(230, 170, 211, 31))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(11)
        font.setBold(False)
        font.setWeight(50)
        self.pushButton_10.setFont(font)
        self.pushButton_10.setStyleSheet("background-color: rgb(246, 184, 61);\n"
"border-style:outset;\n"
"border-width:1px;\n"
"border-radius:10px;\n"
"border-color: rgb(0, 0, 0);")
        self.pushButton_10.setAutoExclusive(False)
        self.pushButton_10.setObjectName("pushButton_10")
        self.pushButton_9 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_9.setGeometry(QtCore.QRect(260, 250, 161, 31))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(11)
        font.setBold(False)
        font.setWeight(50)
        self.pushButton_9.setFont(font)
        self.pushButton_9.setStyleSheet("background-color: rgb(246, 184, 61);\n"
"border-style:outset;\n"
"border-width:1px;\n"
"border-radius:10px;\n"
"border-color: rgb(0, 0, 0);")
        self.pushButton_9.setAutoExclusive(False)
        self.pushButton_9.setObjectName("pushButton_9")
        self.pushButton_11 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_11.setGeometry(QtCore.QRect(280, 290, 121, 31))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(11)
        font.setBold(False)
        font.setWeight(50)
        self.pushButton_11.setFont(font)
        self.pushButton_11.setStyleSheet("background-color: rgb(246, 184, 61);\n"
"border-style:outset;\n"
"border-width:1px;\n"
"border-radius:10px;\n"
"border-color: rgb(0, 0, 0);")
        self.pushButton_11.setAutoExclusive(False)
        self.pushButton_11.setObjectName("pushButton_11")
        self.pushButton_8 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_8.setGeometry(QtCore.QRect(250, 130, 171, 31))
        font = QtGui.QFont()
        font.setFamily("Arial")
        font.setPointSize(11)
        font.setBold(False)
        font.setWeight(50)
        self.pushButton_8.setFont(font)
        self.pushButton_8.setStyleSheet("background-color: rgb(246, 184, 61);\n"
"border-style:outset;\n"
"border-width:1px;\n"
"border-radius:10px;\n"
"border-color: rgb(0, 0, 0);")
        self.pushButton_8.setAutoExclusive(False)
        self.pushButton_8.setObjectName("pushButton_8")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 678, 18))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
        self.pushButton_7.setText(_translate("MainWindow", "Education, Health and Human Sciences"))
        self.label_2.setText(_translate("MainWindow", "GSU ELECTION RESULTS"))
        self.label_3.setText(_translate("MainWindow", "FACULTIES:"))
        self.pushButton_10.setText(_translate("MainWindow", "Architecture, Computing and Humanities"))
        self.pushButton_9.setText(_translate("MainWindow", "Engineering and Science"))
        self.pushButton_11.setText(_translate("MainWindow", "Business School"))
        self.pushButton_8.setText(_translate("MainWindow", "GSU Headship Team"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
        sys.exit(app.exec_())
```

4. *Business_School.py/President.py (as they are just slightly different by their function names)*

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'prova_widget.ui'
#
# Created by: PyQt5 UI code generator 5.13.2
#
# WARNING! All changes made in this file will be lost!


from PyQt5 import QtCore, QtGui, QtWidgets
```

```
import sqlite3
import sys

class Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(400, 300)
        # create table:
        self.buttonGroup0 = QtWidgets.QButtonGroup(Form)
        self.buttonGroup1 = QtWidgets.QButtonGroup(Form)
        self.buttonGroup2 = QtWidgets.QButtonGroup(Form)
        self.buttonGroup3 = QtWidgets.QButtonGroup(Form)
        self.buttonGroup4 = QtWidgets.QButtonGroup(Form)
        self.table = QtWidgets.QTableWidget(Form)
        self.table.setGeometry(QtCore.QRect(30, 10, 381, 191))
        self.table.setRowCount(16)
        self.table.setColumnCount(5)
        #Add title
        item = QtWidgets.QTableWidgetItem('Candidates')
        self.table.setHorizontalHeaderItem(0,item)
        item = QtWidgets.QTableWidgetItem('1')
        self.table.setHorizontalHeaderItem(1,item)
        item = QtWidgets.QTableWidgetItem('2')
        self.table.setHorizontalHeaderItem(2,item)
        item = QtWidgets.QTableWidgetItem('3')
        self.table.setHorizontalHeaderItem(3,item)
        item = QtWidgets.QTableWidgetItem('4')
        self.table.setHorizontalHeaderItem(4,item)
        QtCore.QMetaObject.connectSlotsByName(Form)
        #pushButton
        self.pushButton = QtWidgets.QPushButton("Sumbit Vote",Form)
        self.pushButton.setGeometry(QtCore.QRect(160, 230, 111, 31))
        self.pushButton.setObjectName("pushButton")
        #label
        self.label = QtWidgets.QLabel(Form)
        self.label.setGeometry(QtCore.QRect(0, 211, 441, 20))
        self.label.setText('Check all the checkbox')
        self.label.hide()

        def load_candidates():
            connection = sqlite3.connect("login.db")
            connection.row_factory = lambda cursor, row: row[0]
            c = connection.cursor()
            ids = c.execute("SELECT NAME FROM Candidates WHERE POSITION='GSU_Officer'").fetchall()
            return ids

        def add_item(rows):
            for i in range(1,5):
                ch = QtWidgets.QCheckBox(parent=self.table)
                b = str(rows)
                self.rp = eval("self.buttonGroup"+b)
                self.rp.addButton(ch, id=i)
                self.table.setCellWidget(rows, int(i), ch)
        for i, letter in enumerate(load_candidates()):
            self.table.setItem(i, 0, QtWidgets.QTableWidgetItem(letter))

        self.a = len(list(load_candidates()))

        for x in range(self.a):
            add_item(x)
        self.pushButton.clicked.connect(lambda checked, row=0, col=i: self.checkbox_state(row,checked))

    def checkbox_state(self,row,checked):
        states = []
        for x in range(self.a):
            b = str(x)
            self.group_name = eval("self.buttonGroup"+b)
            for y in range(4):
                indexOfChecked = self.group_name.buttons()[y].isChecked()
                states.append(indexOfChecked)
        if states.count(True) == 4:
            self.submit_vote(row,checked)
        else:
            self.label.show()


    def submit_vote(self,row,checked):
        for x in range(self.a):
            b = str(x)
            self.pq = eval("self.buttonGroup"+b)
            score = self.pq.checkedId()
            name = self.table.item(x,0)
            name = name.text()
            connection = sqlite3.connect("login.db")
            first_score = connection.execute("SELECT SCORE FROM Candidates WHERE NAME = ?",(name,)).fetchone()[0]
            score = first_score + score
            print (score,name)
            connection.execute("UPDATE Candidates SET SCORE = ? WHERE NAME = ?",(score,name))
            connection.commit()
        connection.close

if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    Form = QtWidgets.QWidget()
    ui = Ui_Form()
    ui.setupUi(Form)
    Form.show()
            sys.exit(app.exec_())
```

5. Graph.py

```
from __future__ import unicode_literals
import sys
import os,sqlite3
import random
import matplotlib
import numpy as np
# Make sure that we are using QT5
matplotlib.use('Qt5Agg')
from PyQt5 import QtCore, QtWidgets

from numpy import arange, sin, pi
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as FigureCanvas
from matplotlib.figure import Figure

progversion = "0.1"


class MyMplCanvas(FigureCanvas):
    """Ultimately, this is a QWidget (as well as a FigureCanvasAgg, etc.)."""

    def __init__(self, parent=None, width=5, height=4, dpi=100):
        fig = Figure(figsize=(width, height), dpi=dpi)
        self.axes = fig.add_subplot(111)

        self.compute_initial_figure()

        FigureCanvas.__init__(self, fig)
        self.setParent(parent)

        FigureCanvas.setSizePolicy(self,
                                   QtWidgets.QSizePolicy.Expanding,
                                   QtWidgets.QSizePolicy.Expanding)
        FigureCanvas.updateGeometry(self)

    def compute_initial_figure(self):
        pass


class MyStaticMplCanvas(MyMplCanvas):
    """Simple canvas with a sine plot."""
    def count_votes(self):
        counts = {}
        c = sqlite3.connect("login.db")
        ids = c.execute("SELECT NAME FROM Candidates WHERE POSITION='GSU_Officer'").fetchall()
        for x in ids:
            name = (x[0])
            votes = c.execute("SELECT COUNT(*) FROM COUNTING WHERE CANDIDATE = ?;",(name,)).fetchone()[0]
            counts[name] = votes
        #Single Transferable vote
        highest = max(counts, key=counts.get)
        value = counts[highest]
        return counts
    def quota(self):
        seats = 1
        c = sqlite3.connect("login.db")
        total_votes = c.execute("SELECT COUNT(*) FROM COUNTING WHERE POSITION = 'GSU_Officer';").fetchone()[0]
        quota = (total_votes/(seats+1))+1
        return quota
    def compute_initial_figure(self):
        y = self.quota()
        self.axes.axhline(y, color='r', linestyle='-')
        D = self.count_votes()
        self.axes.bar(*zip(*D.items()))

class ApplicationWindow(QtWidgets.QMainWindow):
    def __init__(self):
        QtWidgets.QMainWindow.__init__(self)
        self.setAttribute(QtCore.Qt.WA_DeleteOnClose)
        self.setWindowTitle("application main window")
        self.main_widget = QtWidgets.QWidget(self)

        l = QtWidgets.QVBoxLayout(self.main_widget)
        sc = MyStaticMplCanvas(self.main_widget, width=5, height=4, dpi=100)
        l.addWidget(sc)
        self.main_widget.setFocus()
        self.setCentralWidget(self.main_widget)


qApp = QtWidgets.QApplication(sys.argv)

aw = ApplicationWindow()
aw.setWindowTitle("Election Result")
aw.show()
        sys.exit(qApp.exec_())
```

6. Project_images

```
# -*- coding: utf-8 -*-

# Resource object code
#
# Created by: The Resource Compiler for PyQt5 (Qt v5.13.2)
#
# WARNING! All changes made in this file will be lost!

from PyQt5 import QtCore

qt_resource_data = b"\
\x00\x0c\x30\x80\
\xff\...\x00\x00\x01\x6e\xd5\xc6\xd4\xfe\
"

qt_version = [int(v) for v in QtCore.qVersion().split('.')]
if qt_version < [5, 8, 0]:
    rcc_version = 1
    qt_resource_struct = qt_resource_struct_v1
else:
    rcc_version = 2
    qt_resource_struct = qt_resource_struct_v2

def qInitResources():
    QtCore.qRegisterResourceData(rcc_version, qt_resource_struct, qt_resource_name, qt_resource_data)

def qCleanupResources():
    QtCore.qUnregisterResourceData(rcc_version, qt_resource_struct, qt_resource_name, qt_resource_data)

        qInitResources()
```