CESG 505 Assignment 6 Ran Ju #1621899

Simple Graphic User Interface (GUI) for your midterm project.

Given:

- Your flight database from your midterm solution
- Your Graph class, including a findShortestPath(self) method

ToDo:

- create a GUI that creates a Graph containing your flight database as part of the constructor
- implement a method Graph.getAirportCodes(self) to Graph that returns a sorted list of available airports (by three-character code)

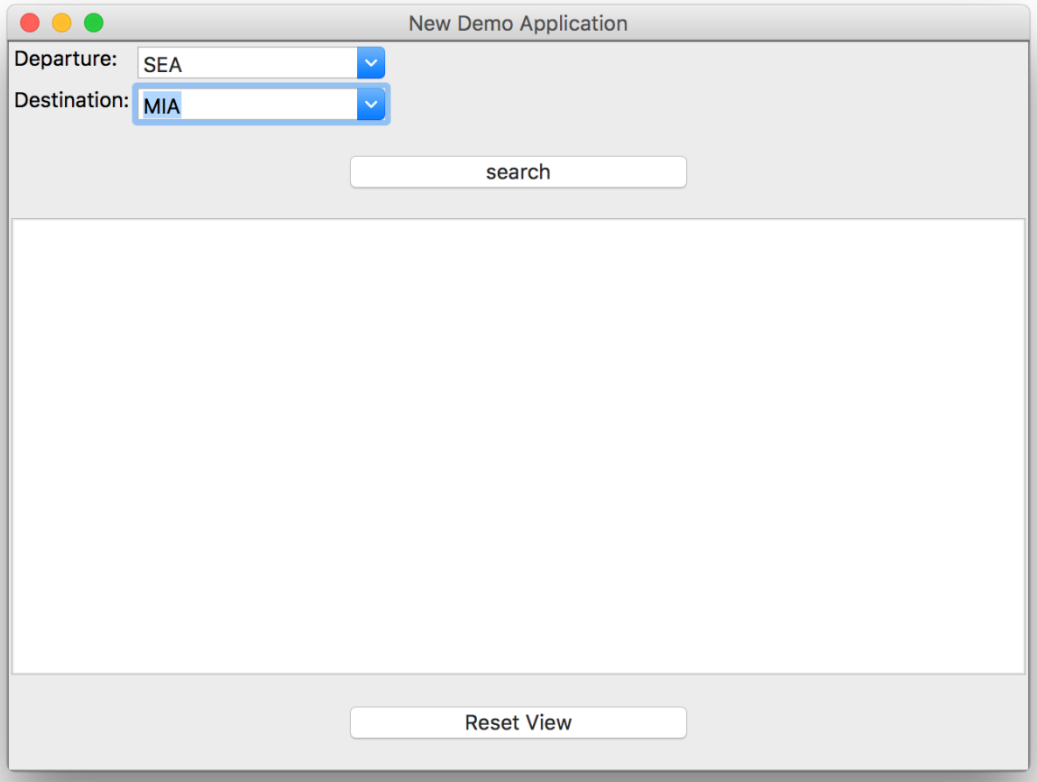
  Here is the code that I add in my graph class:

```
def getAirportCodes(self):
    self.cur.execute(getAllAirports)
    air = self.cur.fetchall()
    res = []
    for city in air:
        res.append(city[0])
    return res
getAllAirports = '''
SELECT AirportName
FROM Airports
'''
```

  Here is the result:

  ['SEA', 'SFO', 'DIA', 'OAK', 'JFK', 'MIA', 'DFW', 'LGA', 'SLC', 'ORD']

- create a GUI similar to the following image:



-
- use the list of airport codes provided by your new Graph.getAirportCodes() to populate both QComboBoxes
- implement a callback function to find the shortest flight between the selected airports. Display an error message in the QTextEdit if departure airport matches the destination.
- show the result of your search in the shown QTextEdit -- only if no input error occured.
- make sure that a new selection of either airport erases the text field.
- make the "reset view"-button move the app to the center of the screen and resize to half screen width and half screen height.
- allow selection and copy from the text field. Prevent the user from changing that text other than through a search.

- add a menu that presents a simple About dialog and a Quit feature.

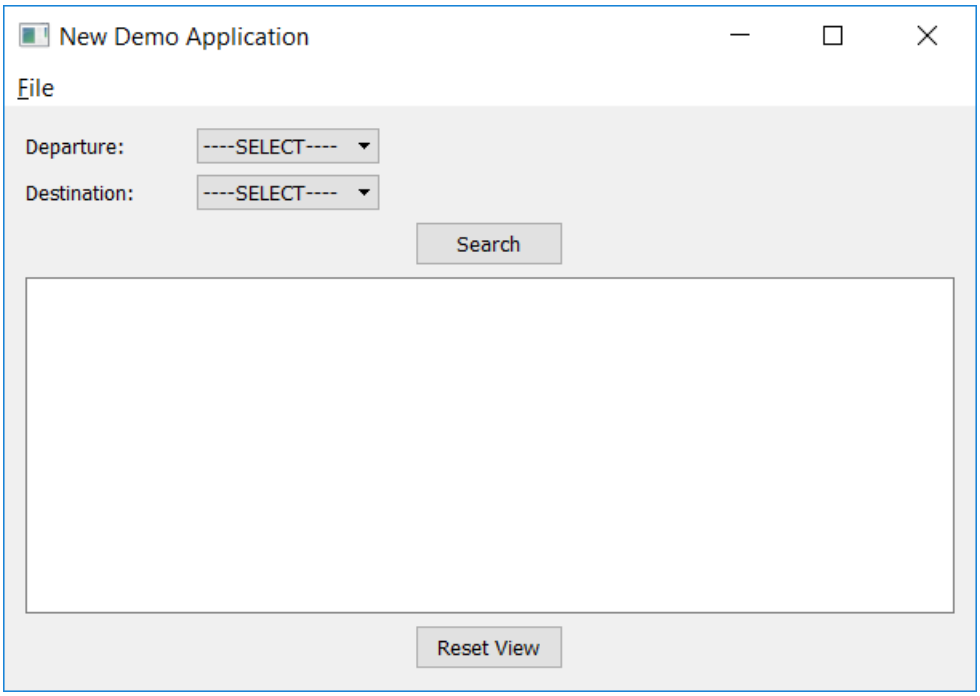


Figure 1

As we can see in the figure 1, I create a GUI just like teacher's demo. To be more specific, there are two combo-boxes that contain all the cities in the database. Also, there are two buttons, "Search" button can search the shortest path between two cities we choose. "Reset View" button can eliminate the text edit, resize and relocate the application to the default value. Moreover, this application has a menu bar which has two options that can choose. The first one is "About", which using a new message box to give the introduction of this application. The "Exit" button can quit this program.

Here are all my steps that design this application:

1. The mainWindow:

   The mainWindow's design are shown on Figure 1. Here is my code of the constructor:

```
g = Graph('Graph.db')
list = ['----SELECT----'] + g.getAirportCodes()
class flightSearch(QWidget):
    def __init__(self, parent=None):
        super().__init__(parent)
        screen = QApplication.desktop().screenGeometry()
        x = (screen.width() - self.size().width()) / 2
        y = (screen.height() - self.size().height()) / 2
```

```
self.x = x
self.y = y
self.move(x, y)
self.setWindowTitle("New Demo Application")


label_1 = QLabel('Departure:', self)
label_1.setMaximumWidth(100)
label_1.setMinimumWidth(100)

self.depart = QComboBox(self)
self.depart.addItems(list)

label_2 = QLabel('Destination:', self)
label_2.setMaximumWidth(100)
label_2.setMinimumWidth(100)
self.destin = QComboBox(self)

self.search = QPushButton('Search', self)
self.search.move(250,70)

self.search.clicked.connect(self.on_search)
self.destin.addItems(list)
self.result = QTextEdit(self)
self.result.setMinimumSize(580, 210)
self.result.setReadOnly(True)
self.result.setAlignment(Qt.AlignCenter)
self.result.move(10,110)
self.reset = QPushButton('Reset View', self)
self.reset.setMaximumWidth(100)
self.reset.move(250,330)
self.reset.clicked.connect(self.on_reset)

bar = QMenuBar()
file_menu = bar.addMenu('&File')
open_action = QAction('About', self)
close_action = QAction('Quit', self)
open_action.setMenuRole(QAction.NoRole)
close_action.setMenuRole(QAction.NoRole)
file_menu.addAction(open_action)
file_menu.addAction(close_action)
close_action.triggered.connect(self.close)
open_action.triggered.connect(self.msg)
```

```
lyt1 = QHBoxLayout()
lyt1.addWidget(label_1)
lyt1.addWidget(self.depart)
lyt1.addStretch()
lyt2 = QHBoxLayout()
lyt2.addWidget(label_2)
lyt2.addWidget(self.destin)
lyt2.addStretch()
lyt3 = QVBoxLayout()
lyt3.setMenuBar(bar)
lyt3.addLayout(lyt1)
lyt3.addLayout(lyt2)
lyt3.addWidget(self.search,alignment=Qt.AlignCenter)
lyt3.addWidget(self.result)
lyt3.addWidget(self.reset,alignment=Qt.AlignCenter)

self.setLayout(lyt3)
self.size = self.sizeHint()
```

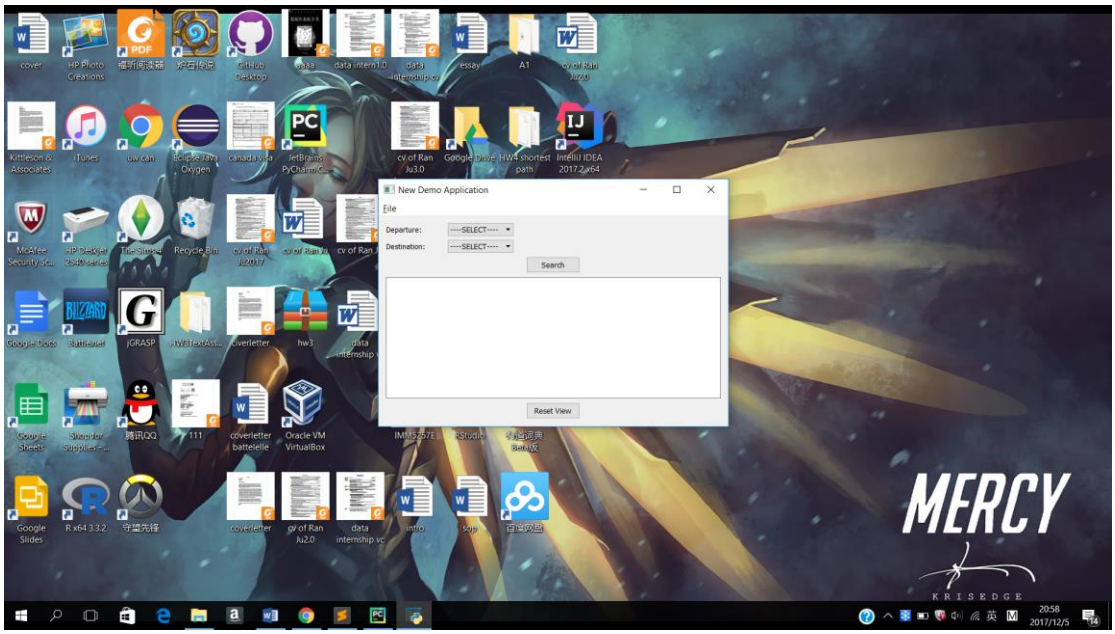


Figure2

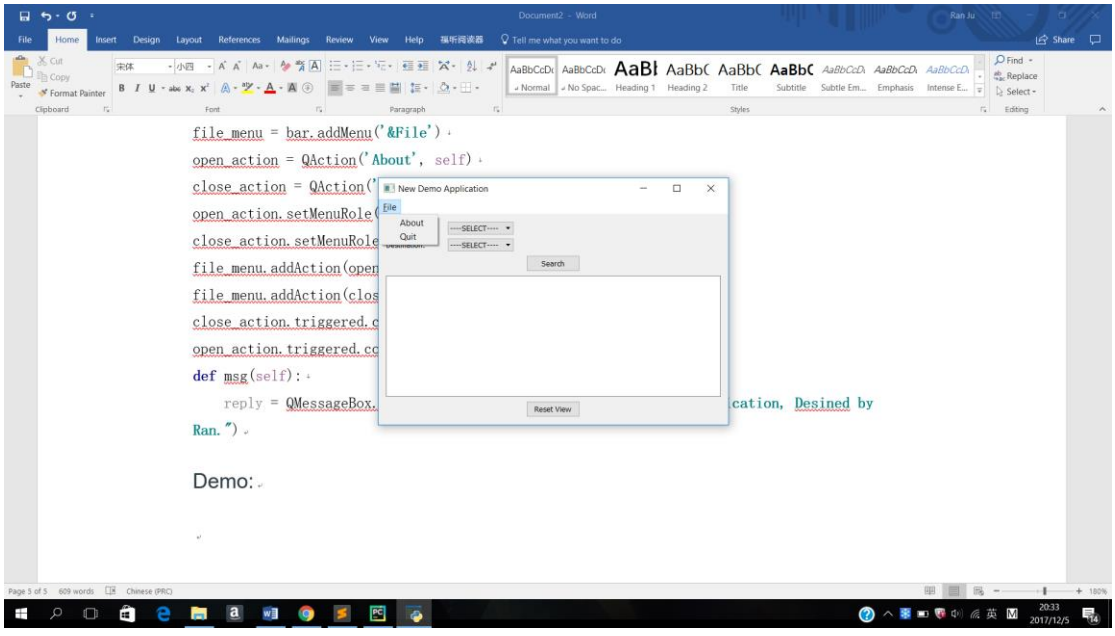
To be more specific, I use **Layout** to organize the interface. So, I create 2 QHBoxLayout that contain one label and its combo box. Then using a QVBoxLayout to get the menu bar, two QHBoxLayouts, the search button,a texit edit and the reset button together. Also, those button should using Qt.AlignCenter to make it center. The default position is the center of the screen(Figure 2).Also, the self.size is default size, self.x and self.y is default position in the screen, which I should use in the call back function of reset button. In the text edit, I use.readOnly() so that the text cannot revise.

2. Menu Bar call back function:

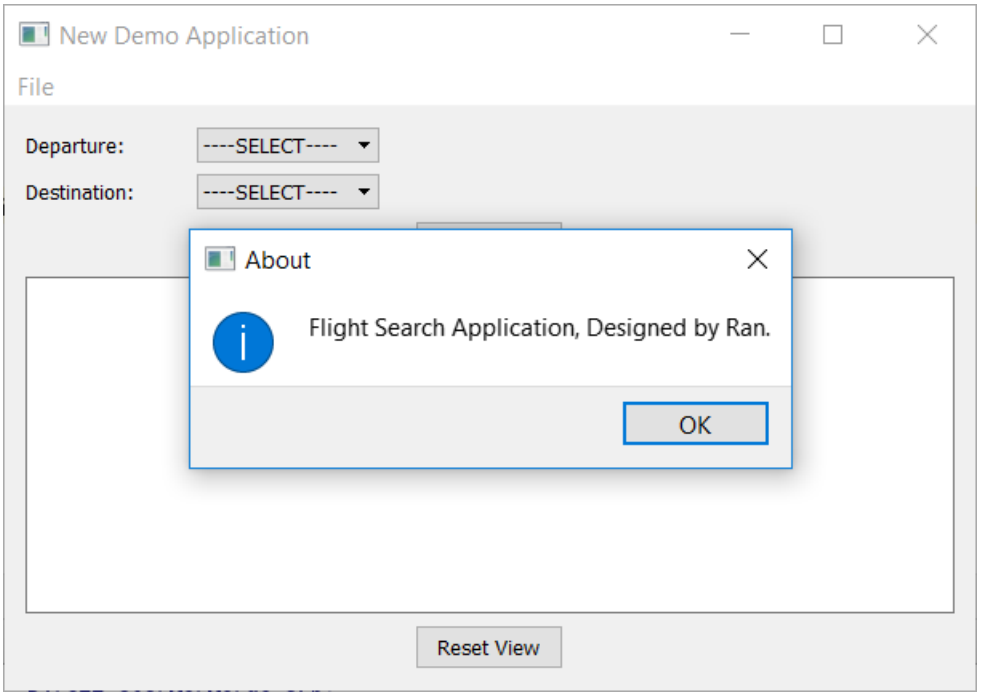
Here is the code:

```
bar = QMenuBar()
file_menu = bar.addMenu('&File')
open_action = QAction('About', self)
close_action = QAction('Quit', self)
open_action.setMenuRole(QAction.NoRole)
close_action.setMenuRole(QAction.NoRole)
file_menu.addAction(open_action)
file_menu.addAction(close_action)
close_action.triggered.connect(self.close)
open_action.triggered.connect(self.msg)
def msg(self):
    reply = QMessageBox.information(self,"About","Flight Search Application, Designed by Ran.")
```

Demo:

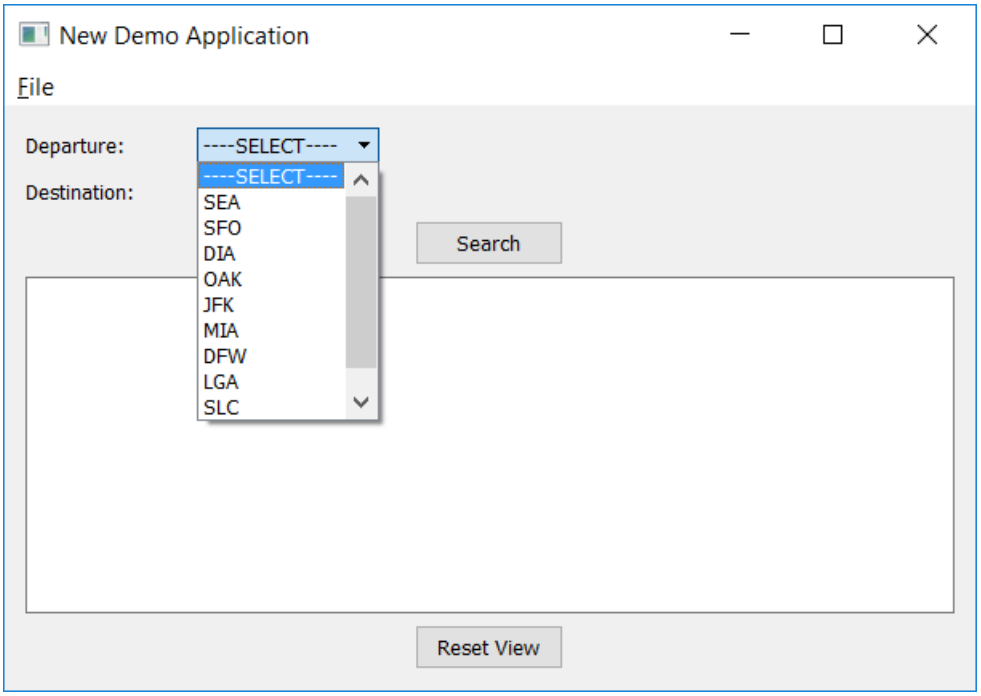


Here is the simple demo of the menu bar: it just has a “file” in the menu, in the file, it has two options: “About” and “Exit”. When I click the “About”, it shows the message like that:



3. Combo-box selection:

Two combo boxes shows all cities in the database. Here is the demo:



4. Search button call back function

When we choose the right city, it can show the shortest path in the text edit. And When we choose the same city or nothing, it can show error message.Here is the code:

```
def on_search(self,event):
    location1 = self.depart.currentText()
```

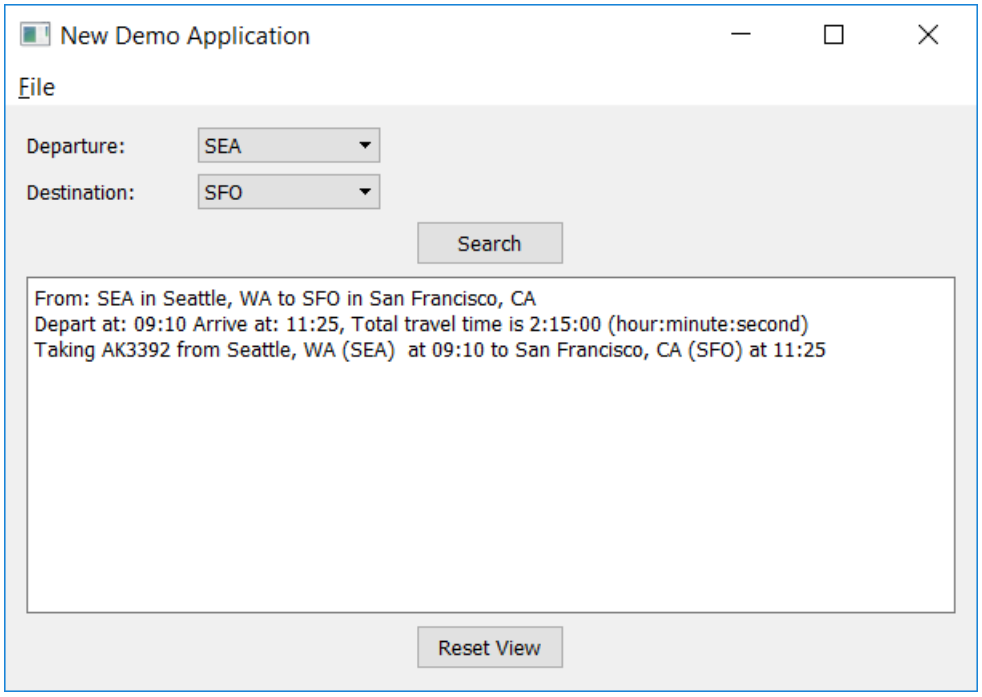
```
    location2 = self.destin.currentText()

    if location1 == list[0] or location2 == list[0] or location1 == location2:
        self.result.setText("")
        self.msgError()
    else:
        ans = g.getShortestPath(location1,location2)
        self.result.setText(ans)
```

Error Message function:

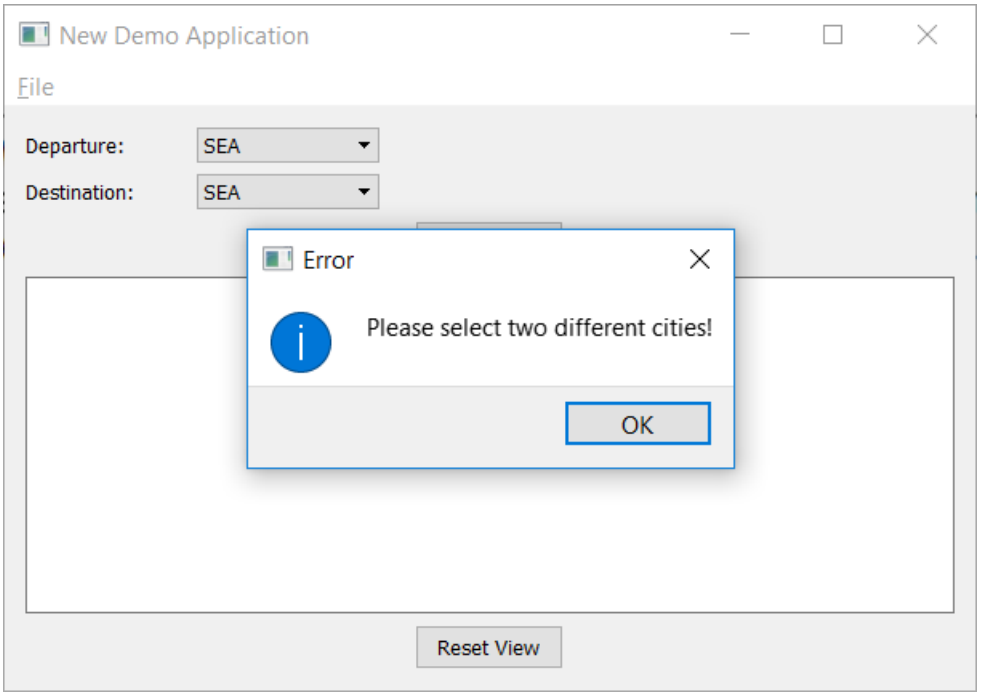
```
def msgError(self):
    reply = QMessageBox.information(self, "Error", "Please select two different cities!")
```

Here is the simple demo:

When we choose right cities:



When we choose the same city or nothing:



5. Reset button call back function:

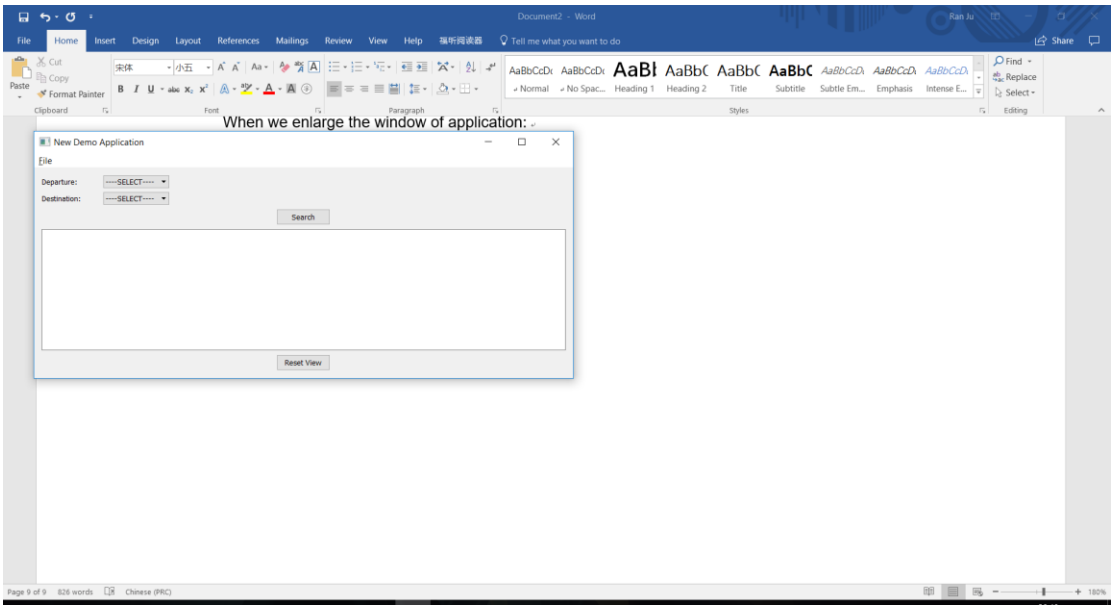
It can make the application return the default value.

```
def on_reset(self,event):
    self.depart.clear()
    self.depart.addItems(list)
    self.destin.clear()
    self.destin.addItems(list)
    self.result.clear()
    self.move(self.x,self.y)
    self.resize(self.size)
```

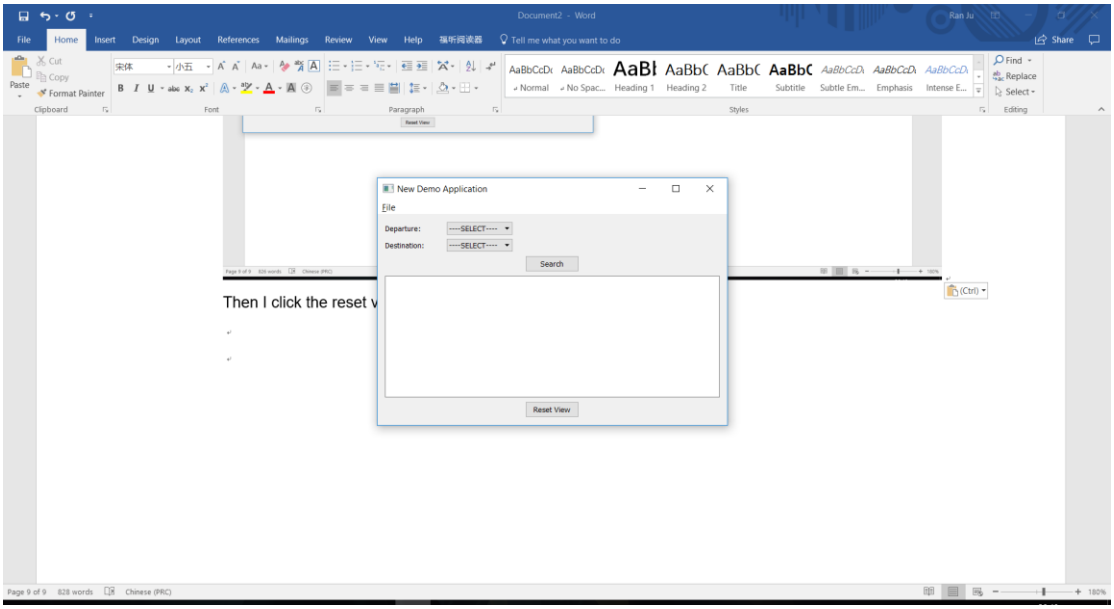
More specific, self.move(self.x,self.y) can return the default position, self.resize(self.size) can return the default size. Self.clear() can clear all content of the text edit.

Here is the demo:

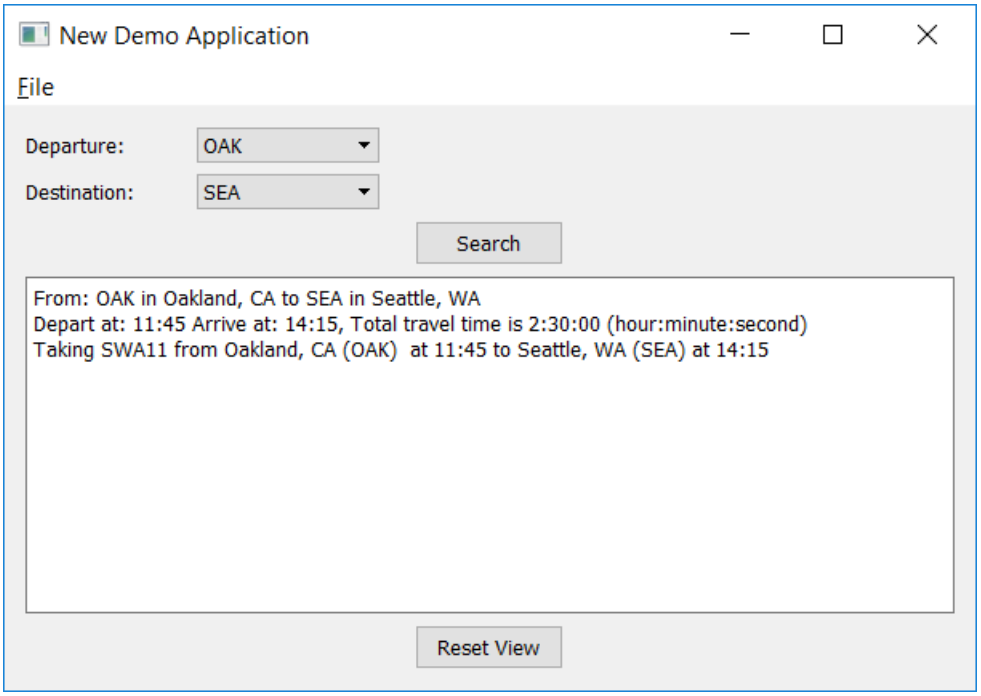
When we enlarge and move the window of application:



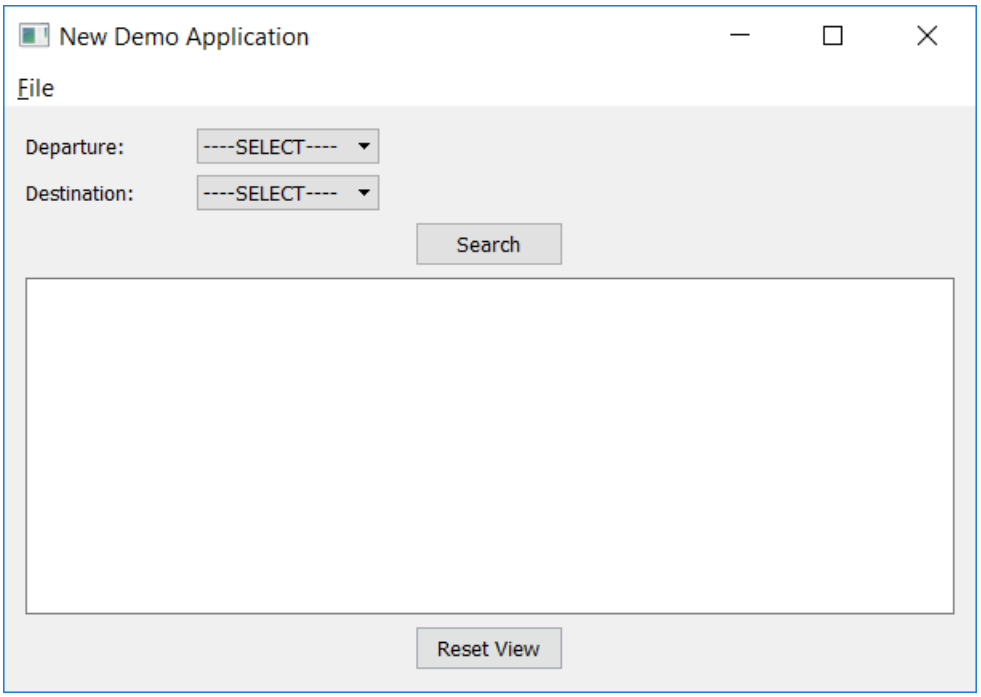
Then I click the reset view button:



When it has content of text edit:



Then click the reset button:



So the demo shows all function that I create can run successfully!

Appendix:

```
import sys
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
from q2 import *
g = Graph('Graph.db')
list = ['----SELECT----'] + g.getAirportCodes()
```

```
class flightSearch(QWidget):
    def __init__(self, parent=None):
        super().__init__(parent)
        screen = QApplication.desktop().screenGeometry()
        x = (screen.width() - self.size().width()) / 2
        y = (screen.height() - self.size().height()) / 2
        self.x = x
        self.y = y
        self.move(x, y)
        self.setWindowTitle("New Demo Application")


        label_1 = QLabel('Departure:', self)
        label_1.setMaximumWidth(100)
        label_1.setMinimumWidth(100)

        self.depart = QComboBox(self)
        self.depart.addItems(list)

        label_2 = QLabel('Destination:', self)
        label_2.setMaximumWidth(100)
        label_2.setMinimumWidth(100)
        self.destin = QComboBox(self)

        self.search = QPushButton('Search', self)
        self.search.move(250,70)

        self.search.clicked.connect(self.on_search)
        self.destin.addItems(list)
        self.result = QTextEdit(self)
        self.result.setMinimumSize(580, 210)
        self.result.setReadOnly(True)
        self.result.setAlignment(Qt.AlignCenter)
        self.result.move(10,110)
        self.reset = QPushButton('Reset View', self)
        self.reset.setMaximumWidth(100)
        self.reset.move(250,330)
        self.reset.clicked.connect(self.on_reset)

        bar = QMenuBar()
        file_menu = bar.addMenu('&File')
        open_action = QAction('About', self)
```

```
        close_action = QAction('Quit', self)
        open_action.setMenuRole(QAction.NoRole)
        close_action.setMenuRole(QAction.NoRole)
        file_menu.addAction(open_action)
        file_menu.addAction(close_action)
        close_action.triggered.connect(self.close)
        open_action.triggered.connect(self.msg)

        lyt1 = QHBoxLayout()
        lyt1.addWidget(label_1)
        lyt1.addWidget(self.depart)
        lyt1.addStretch()
        lyt2 = QHBoxLayout()
        lyt2.addWidget(label_2)
        lyt2.addWidget(self.destin)
        lyt2.addStretch()
        lyt3 = QVBoxLayout()
        lyt3.setMenuBar(bar)
        lyt3.addLayout(lyt1)
        lyt3.addLayout(lyt2)
        lyt3.addWidget(self.search,alignment=Qt.AlignCenter)
        lyt3.addWidget(self.result)
        lyt3.addWidget(self.reset,alignment=Qt.AlignCenter)

        self.setLayout(lyt3)
        self.size = self.sizeHint()

    def msg(self):
        reply = QMessageBox.information(self,"About","Flight Search Application, Desined by
Ran.")

    def on_reset(self,event):
        self.depart.clear()
        self.depart.addItems(list)
        self.destin.clear()
        self.destin.addItems(list)
        self.result.clear()
        self.move(self.x,self.y)
        self.resize(self.size)



    def on_search(self,event):
        location1 = self.depart.currentText()
```

```
        location2 = self.destin.currentText()

        if location1 == list[0] or location2 == list[0] or location1 == location2:
            self.result.setText("")
            self.msgError()

        else:
            ans = g.getShortestPath(location1,location2)
            self.result.setText(ans)

    def on_click(self):
        self.se.show()



    def msgError(self):
        reply = QMessageBox.information(self, "Error", "Please select two different
cities!")


if __name__ == '__main__':
    app = QApplication(sys.argv)
    # creating main window
    mw = flightSearch()
    mw.show()
    sys.exit(app.exec_())
```