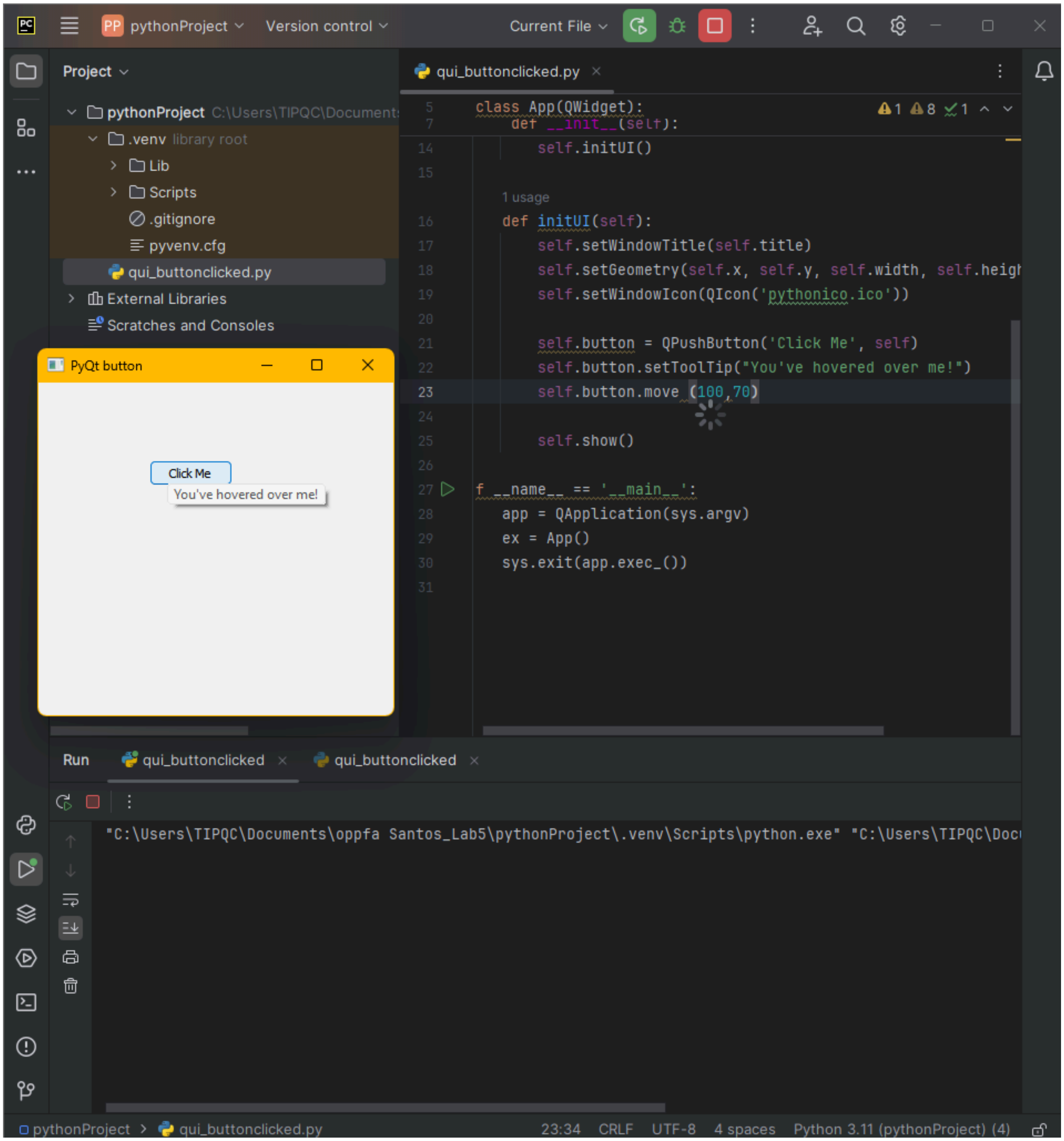
Activity Name #<number> - Activity Title	
Santos, Ma. Kassandra Nicole D	10/21/2024
Course/Section: CPE21S4	Ma'am Rizette Sayo

EVENT HANDLING

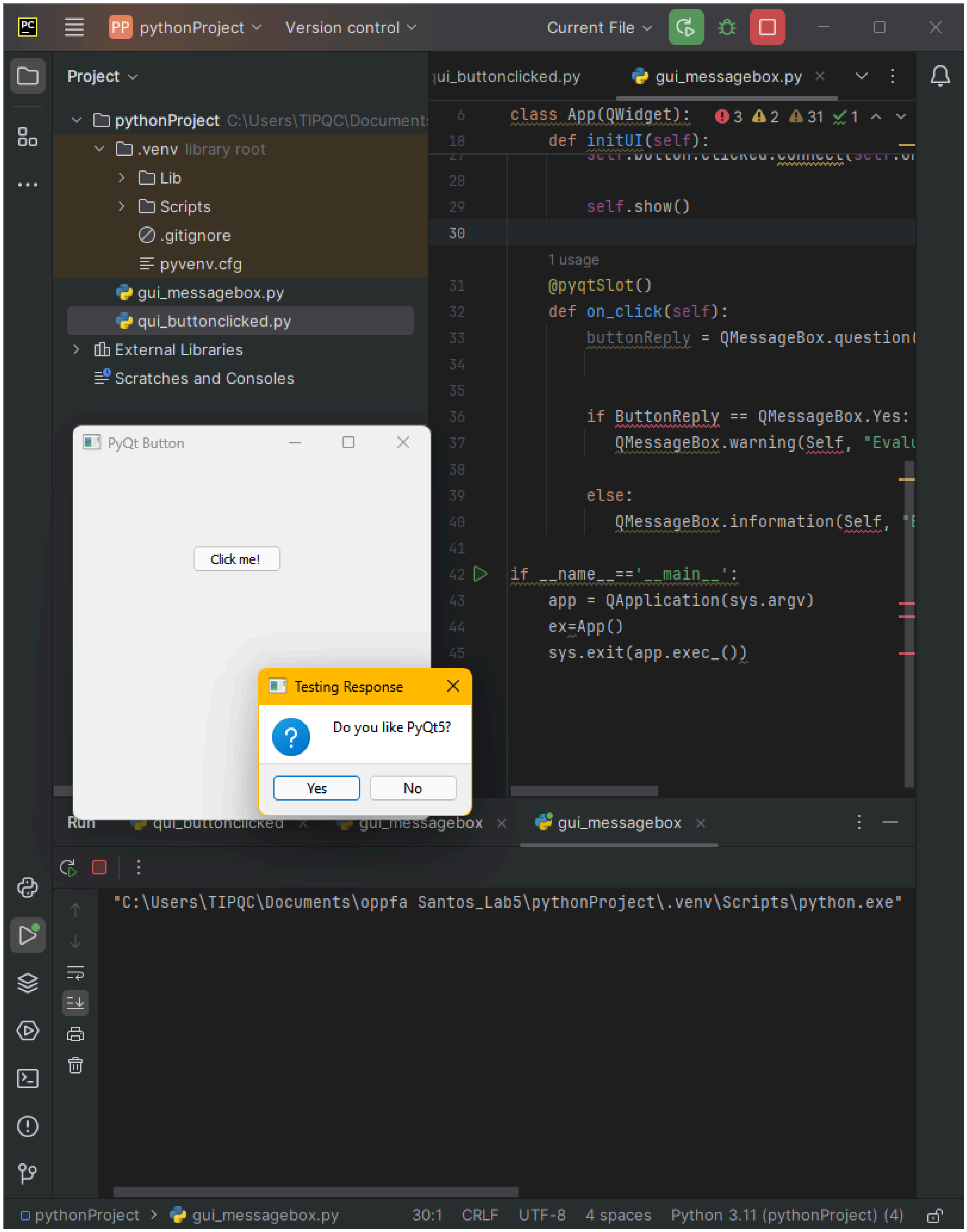


```
class App(QWidget):
    def __init__(self):
        self.initUI()

    1 usage
    def initUI(self):
        self.setWindowTitle(self.title)
        self.setGeometry(self.x, self.y, self.width, self.heigh
        self.setWindowIcon(QIcon('pythonico.ico'))

        self.button = QPushButton('Click Me', self)
        self.button.setToolTip("You've hovered over me!")
        self.button.move (100,70)

        self.show()

f __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = App()
    sys.exit(app.exec_())
```

```
"C:\Users\TIPQC\Documents\oppfa Santos_Lab5\pythonProject\.venv\Scripts\python.exe" "C:\Users\TIPQC\Doc
```

Observation: When you click on the button, it does nothing. However, when you just simply hover on the button its shows the message "You've hovered over me"

ADDING A MESSAGE BOX



```
class App(QWidget):
    def initUI(self):
        self.button.clicked.connect(self.on

        self.show()

    @pyqtSlot()
    def on_click(self):
        buttonReply = QMessageBox.question(


        if ButtonReply == QMessageBox.Yes:
            QMessageBox.warning(Self, "Evalu

        else:
            QMessageBox.information(Self, "E

if __name__=='__main__':
    app = QApplication(sys.argv)
    ex=App()
    sys.exit(app.exec_())
```

Testing Response

Do you like PyQt5?

Yes No

```
"C:\Users\TIPQC\Documents\oppfa Santos_Lab5\pythonProject\.venv\Scripts\python.exe"
```

Observation: When you click on the button this time, it shows a message box with the question Do you like pyqt5 in which the user can interact with it.

SUPPLEMENTARY

Python

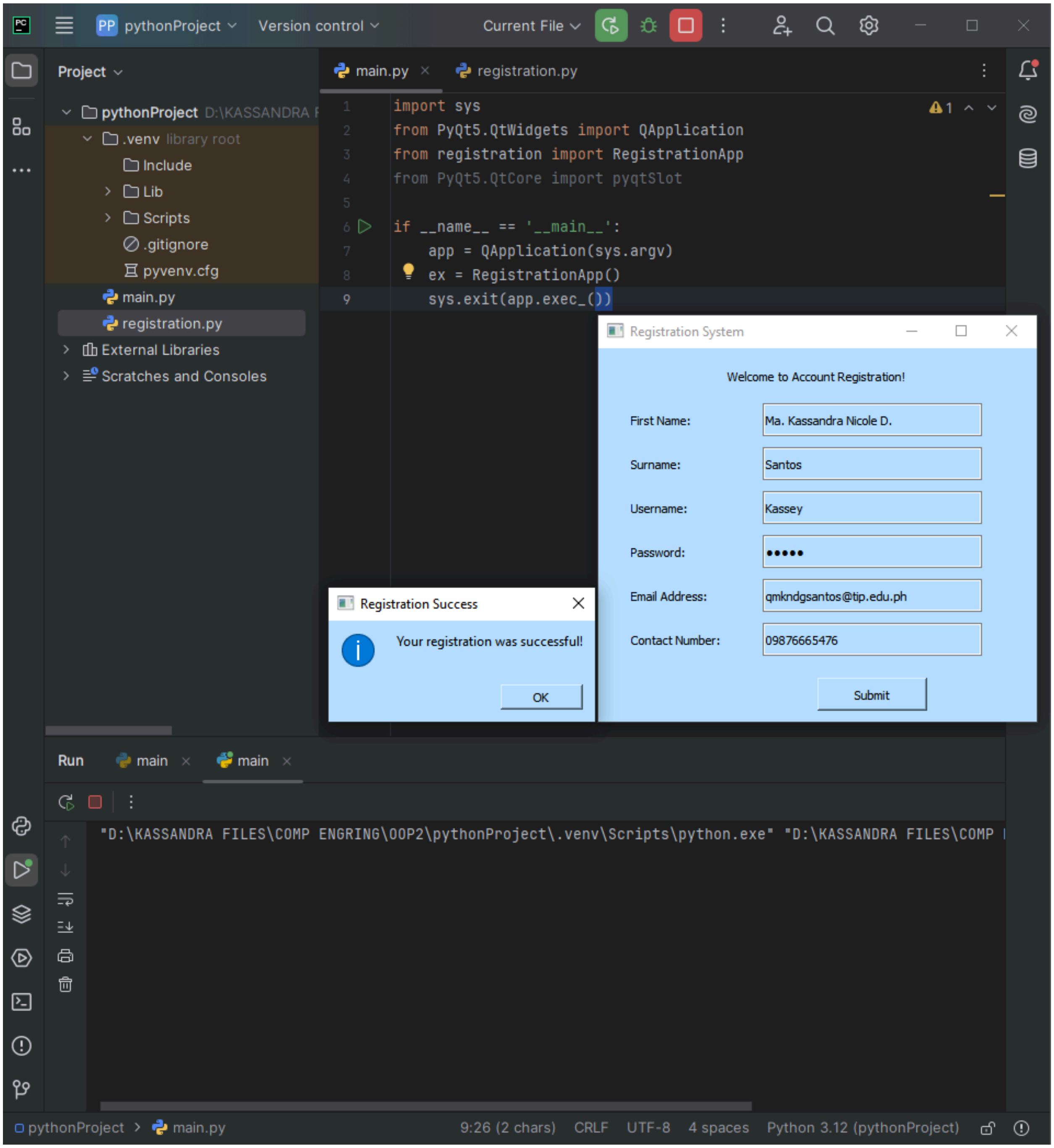
```python
#this is the main.py
import sys
from PyQt5.QtWidgets import QApplication
from registration import RegistrationApp
from PyQt5.QtCore import pyqtSlot

if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = RegistrationApp()
    sys.exit(app.exec_())
```

Python

```python
#this is the registration.py
from PyQt5.QtWidgets import QMainWindow, QApplication, QLabel, QLineEdit, QPushButton, QMessageBox
from PyQt5.QtGui import QIcon
from PyQt5.QtCore import Qt
from PyQt5.QtCore import pyqtSlot
import csv

class RegistrationApp(QMainWindow):

    def __init__(self):
        super().__init__()
        self.title = "Registration System"
        self.width = 400
        self.height = 400
        self.initUI()

    def initUI(self):
        self.setWindowTitle(self.title)
        self.setGeometry(100, 100, self.width, self.height)
        self.setWindowIcon(QIcon('pythonico.ico'))
        self.setStyleSheet("background-color: #B7E0FF;")
        self.center()

        self.program_title = QLabel("Welcome to Account Registration!", self)
        self.program_title.setAlignment(Qt.AlignCenter)
        self.program_title.setGeometry(0, 10, self.width, 30)

        self.label_first_name = QLabel("First Name:", self)
        self.label_first_name.move(30, 50)
        self.textbox_first_name = QLineEdit(self)
        self.textbox_first_name.move(150, 50)
        self.textbox_first_name.resize(200, 30)

        self.label_Surname = QLabel("Surname:", self)
        self.label_Surname.move(30, 90)
        self.textbox_Surname = QLineEdit(self)
        self.textbox_Surname.move(150, 90)
        self.textbox_Surname.resize(200, 30)

        self.label_username = QLabel("Username:", self)
        self.label_username.move(30, 130)
        self.textbox_username = QLineEdit(self)
        self.textbox_username.move(150, 130)
        self.textbox_username.resize(200, 30)

        self.label_password = QLabel("Password:", self)
        self.label_password.move(30, 170)
        self.textbox_password = QLineEdit(self)
        self.textbox_password.move(150, 170)
        self.textbox_password.resize(200, 30)
        self.textbox_password.setEchoMode(QLineEdit.Password)

        self.label_email = QLabel("Email Address:", self)
        self.label_email.move(30, 210)
```

```python
        self.textbox_email = QLineEdit(self)
        self.textbox_email.move(150, 210)
        self.textbox_email.resize(200, 30)

        self.label_contact = QLabel("Contact Number:", self)
        self.label_contact.move(30, 250)
        self.textbox_contact = QLineEdit(self)
        self.textbox_contact.move(150, 250)
        self.textbox_contact.resize(200, 30)

        self.submit_button = QPushButton('Submit', self)
        self.submit_button.setToolTip("You've hovered over me!")
        self.submit_button.move(200, 300)  # button.move(x,y)
        self.submit_button.clicked.connect(self.on_submit)

        self.show()

    def center(self):
        qr = self.frameGeometry()
        cp = QApplication.desktop().screen().rect().center()
        qr.moveCenter(cp)
        self.move(qr.topLeft())

    @pyqtSlot()
    def on_submit(self):
        if not (self.textbox_first_name.text() and self.textbox_Surname.text() and self.textbox_username.text() and
                self.textbox_password.text() and self.textbox_email.text() and self.textbox_contact.text()):
            QMessageBox.warning(self, "Invalid Submission", "Please fill out all fields.")
        else:
            self.submit()

    def submit(self):
        first_name = self.textbox_first_name.text()
        last_name = self.textbox_Surname.text()
        username = self.textbox_username.text()
        password = self.textbox_password.text()
        email = self.textbox_email.text()
        contact = self.textbox_contact.text()

        # Write the data to a CSV file
        with open('registration_data.csv', 'a', newline='') as file:
            writer = csv.writer(file)
            writer.writerow([first_name, last_name, username, password, email,
contact])

        # Show success message
        QMessageBox.information(self, "Registration Success", "Your
registration was successful!")
        print("The Form Has Been Submitted!")

        self.clear()

    def clear(self):
        self.textbox_first_name.clear()
        self.textbox_Surname.clear()
        self.textbox_username.clear()
        self.textbox_password.clear()
        self.textbox_email.clear()
        self.textbox_contact.clear()


if __name__ == '__main__':
    app = QApplication([])
    window = RegistrationApp()
    app.exec_()
```

OUTPUT



```
import sys
from PyQt5.QtWidgets import QApplication
from registration import RegistrationApp
from PyQt5.QtCore import pyqtSlot

if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = RegistrationApp()
    sys.exit(app.exec_())
```

Registration System

Welcome to Account Registration!

First Name: Ma. Kassandra Nicole D.

Surname: Santos

Username: Kassey

Password: •••••

Email Address: qmkndgsantos@tip.edu.ph

Contact Number: 09876665476

Submit

Registration Success

Your registration was successful!

OK

```
"D:\KASSANDRA FILES\COMP ENGRING\OOP2\pythonProject\.venv\Scripts\python.exe" "D:\KASSANDRA FILES\COMP
```

1. What are the other signals available in PyQt5?
    a. Based from the activity the other signals that are available are the following:
        i. QMessageBox
            1. Based on the activity, QMessageBox is a type of signal that pops up after you have done an action, if the action was made, it will trigger the QMessageBox to pop up a window giving the message
        ii. Qlabel

1. Qlabel is a type of signal that helps display a text, it is not clickable and it is commonly used in presenting information

iii. QPushbutton
   1. QPushbutton is a clickable command that performs an action. This signal can be used to submit a form, answer yes or no questions, and more.

2. Why do you think that event handling in Python is divided into signals and slots?
   a. I think that the signals and slots are divided into two because those two have their own respective functions or purposes, but one can't function without the other, when an event occurs in a particular widget such as the push button, the widget emits a signal for that event and the slots is that the response to the signal. In this activity, when you click the submission button when all of the needed information is complete it will sent out messagebox slot whenever it is clicked
3. How can message boxes be used to provide a better User Experience or how can message boxes be used to make a GUI Application more user-friendly?
   a. Message boxes are important because sending out message boxes is straightforward; if the user triggers an event, a message box appears; therefore the message would literally be "in their faces" as they do not need to navigate all over their monitor to find the information or message that they need
4. What is Error-handling and how was it applied in the task performed?
   a. error-handling happens when something unexpected happens when the computer runs the program, in this activity, error-handling occurs when the user does not complete the entire registration. Since the computer is programmed to ensure that all of the fields should have some sort of information placed per box which did not happen in this situation, the computer sent out a pop-up message to tell the user that he/she needs to complete the form
5. What maybe the reasons behind the need to implement error handling?
   a. error handling is important because it helps to make sure that the computer is prepared for unexpected events that are not originally planned or programmed in the system and also to find ways to fix the problem at hand, in this activity, when the computer sent out a warning message to the user that he or she needs to fill out all of the field, that is a way for the system to try solve the problem that is beyond what they are programmed to do

CONCLUSION

Event Handling is composed of two parts: signals and slots. These two work together to make sure that every event that occurs in the program runs smoothly, many kinds of signals would help make the program user-friendly such as message boxes, push buttons, and more. In this lesson, we also learned how to take precautionary actions if there is an event wherein the computer is not programmed to respond through error handling, this feature is important to make sure that the program will run smoothly and that the program will take any measure to resolve the problem such as sending out a warning message to the user to make sure to fill out all of the necessary fields in the activity. Overall, this lesson for me is important as it is a stepping stone to have a program that is efficient and user-friendly.