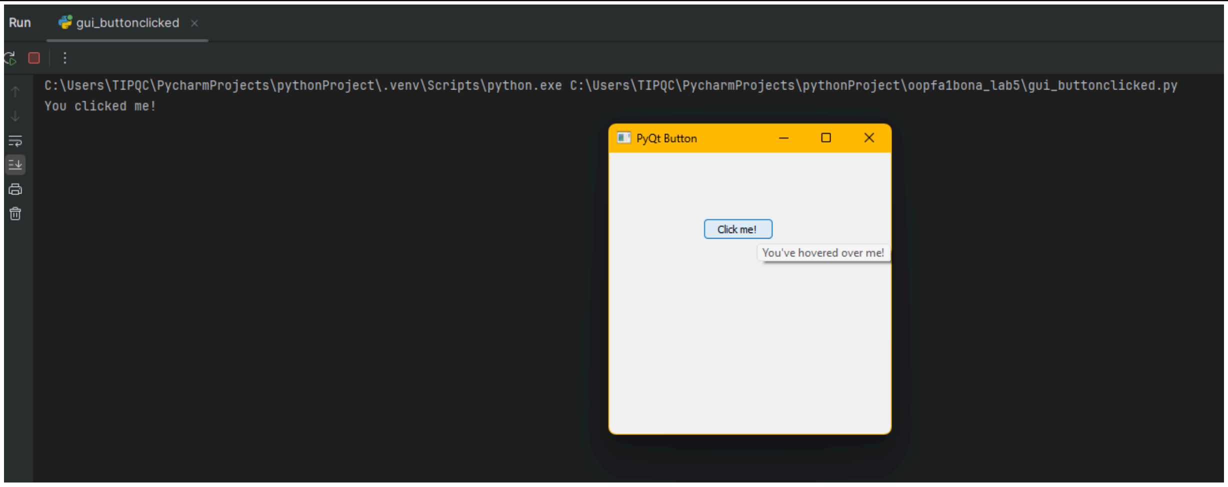
Activity Name # 5 - Introduction to Event Handling in GUI Development	
Bona, Andrei Nycole So	10/21/24
CPE009B - CPE21S4	Prof. Maria Rizette Sayo

gui_buttonclicked.py

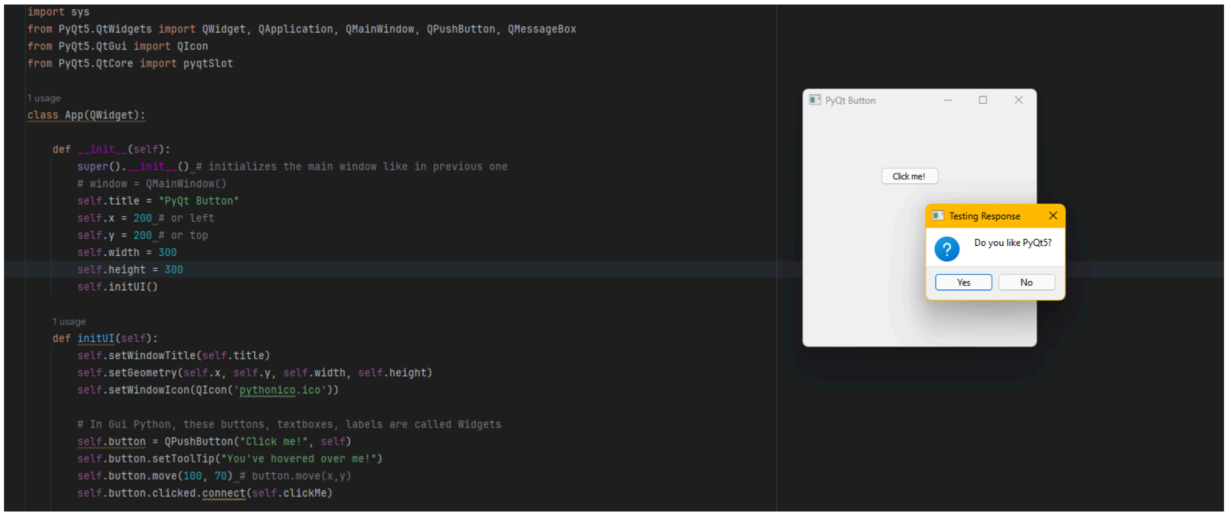
```
import sys
from PyQt5.QtWidgets import QWidget, QApplication, QMainWindow, QPushButton
from PyQt5.QtGui import QIcon
from PyQt5.QtCore import pyqtSlot

class App(QWidget):

   def __init__(self):
       super().__init__() # initializes the main window like in previous one
       # window = QMainWindow()
       self.title = "PyQt Button"
       self.x = 200 # or left
       self.y = 200 # or top
       self.width = 300
       self.height = 300
       self.initUI()

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

       # In Gui Python, these buttons, textboxes, labels are called Widgets
       self.button = QPushButton("Click me!", self)
       self.button.setToolTip("You've hovered over me!")
       self.button.move(100, 70) # button.move(x,y)
       self.button.clicked.connect(self.on_click)

       self.show()
   @pyqtSlot()
   def on_click(self):
       print('You clicked me!')



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



```
C:\Users\TIPQC\PycharmProjects\pythonProject\.venv\Scripts\python.exe C:\Users\TIPQC\PycharmProjects\pythonProject\oopfa1bona_lab5\gui_buttonclicked.py
You clicked me!
```

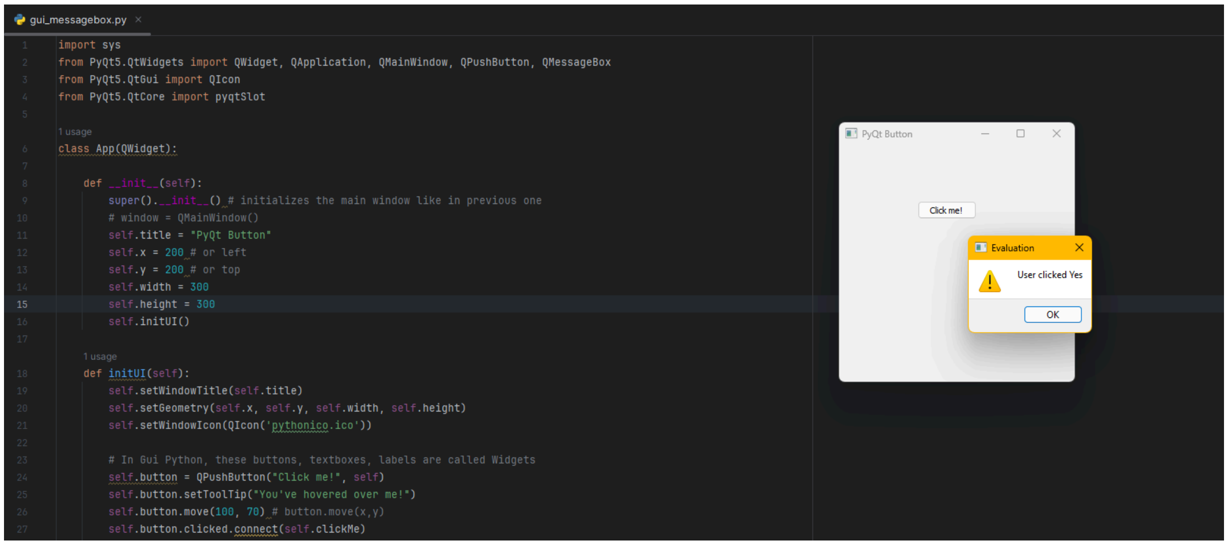
gui_messagebox.py

```
import sys
from PyQt5.QtWidgets import QWidget, QApplication, QMainWindow, QPushButton,
QMessageBox
from PyQt5.QtGui import QIcon
from PyQt5.QtCore import pyqtSlot

class App(QWidget):

   def __init__(self):
       super().__init__() # initializes the main window like in previous one
       # window = QMainWindow()
       self.title = "PyQt Button"
       self.x = 200 # or left
       self.y = 200 # or top
       self.width = 300
       self.height = 300
       self.initUI()

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

       # In Gui Python, these buttons, textboxes, labels are called Widgets
       self.button = QPushButton("Click me!", self)
       self.button.setToolTip("You've hovered over me!")
       self.button.move(100, 70) # button.move(x,y)
       self.button.clicked.connect(self.clickMe)

       self.show()

   @pyqtSlot()
   def clickMe(self):
       buttonReply = QMessageBox.question(self, "Testing Response", "Do you
like PyQt5?",
                                          QMessageBox.Yes | QMessageBox.No,
QMessageBox.Yes)
       if buttonReply == QMessageBox.Yes:
            QMessageBox.warning(self, "Evaluation", "User clicked Yes", 
QMessageBox.Ok, QMessageBox.Ok)
        else:
            QMessageBox.information(self, "Evaluation", "User clicked No", 
QMessageBox.Ok, QMessageBox.Ok)

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



```
import sys
from PyQt5.QtWidgets import QWidget, QApplication, QMainWindow, QPushButton, QMessageBox
from PyQt5.QtGui import QIcon
from PyQt5.QtCore import pyqtSlot

1 usage
class App(QWidget):

    def __init__(self):
        super().__init__() # initializes the main window like in previous one
        # window = QMainWindow()
        self.title = "PyQt Button"
        self.x = 200 # or left
        self.y = 200 # or top
        self.width = 300
        self.height = 300
        self.initUI()

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

        # In Gui Python, these buttons, textboxes, labels are called Widgets
        self.button = QPushButton("Click me!", self)
        self.button.setToolTip("You've hovered over me!")
        self.button.move(100, 70) # button.move(x,y)
        self.button.clicked.connect(self.clickMe)
```

Do you like PyQt5?

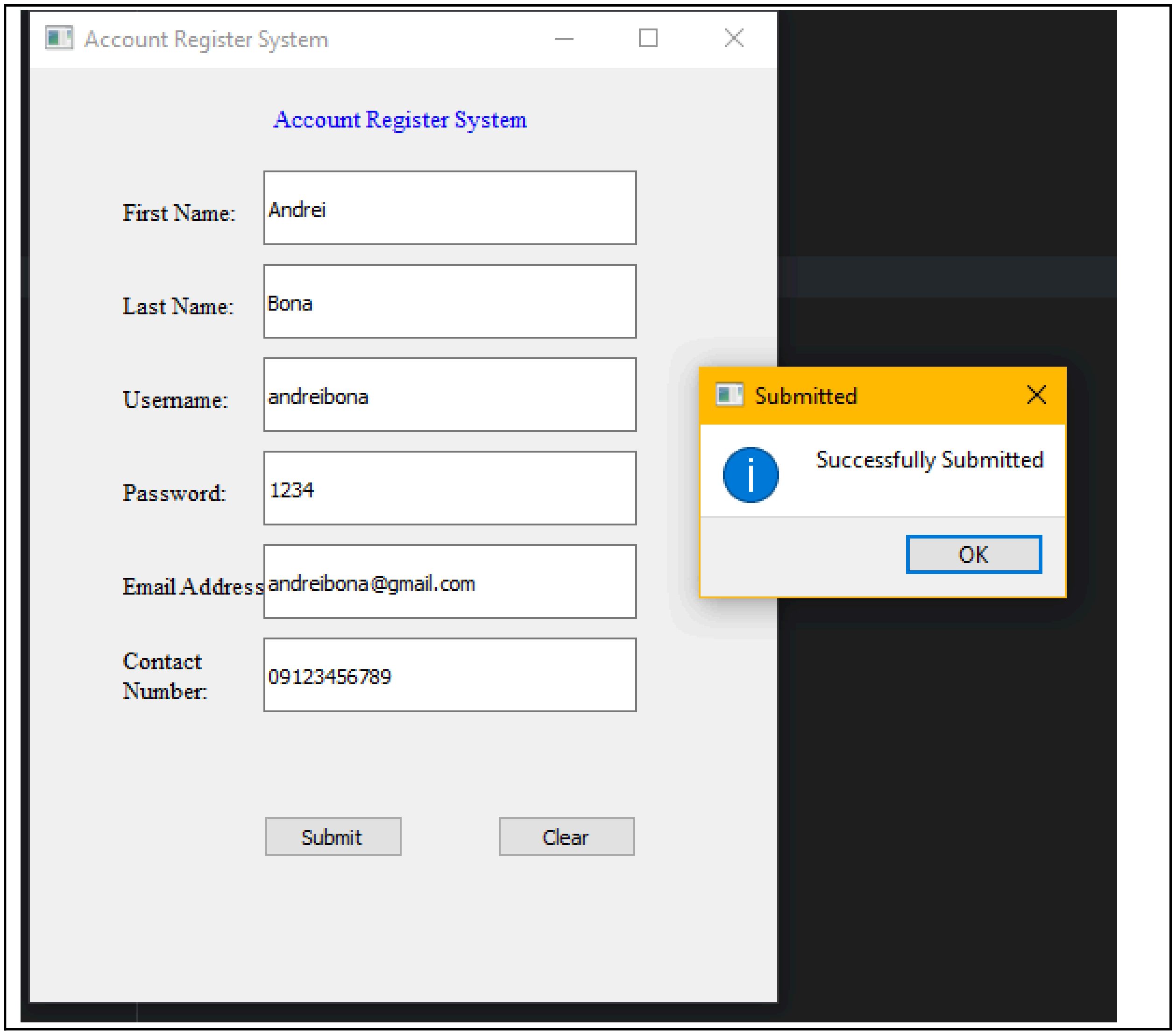


gui_messagebox.py

```
import sys
from PyQt5.QtWidgets import QWidget, QApplication, QMainWindow, QPushButton, QMessageBox
from PyQt5.QtGui import QIcon
from PyQt5.QtCore import pyqtSlot

1 usage
class App(QWidget):

    def __init__(self):
        super().__init__() # initializes the main window like in previous one
        # window = QMainWindow()
        self.title = "PyQt Button"
        self.x = 200 # or left
        self.y = 200 # or top
        self.width = 300
        self.height = 300
        self.initUI()

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

        # In Gui Python, these buttons, textboxes, labels are called Widgets
        self.button = QPushButton("Click me!", self)
        self.button.setToolTip("You've hovered over me!")
        self.button.move(100, 70) # button.move(x,y)
        self.button.clicked.connect(self.clickMe)
```

User clicked Yes

6. Supplementary Activity:



Account Register System

First Name: Andrei

Last Name: Bona

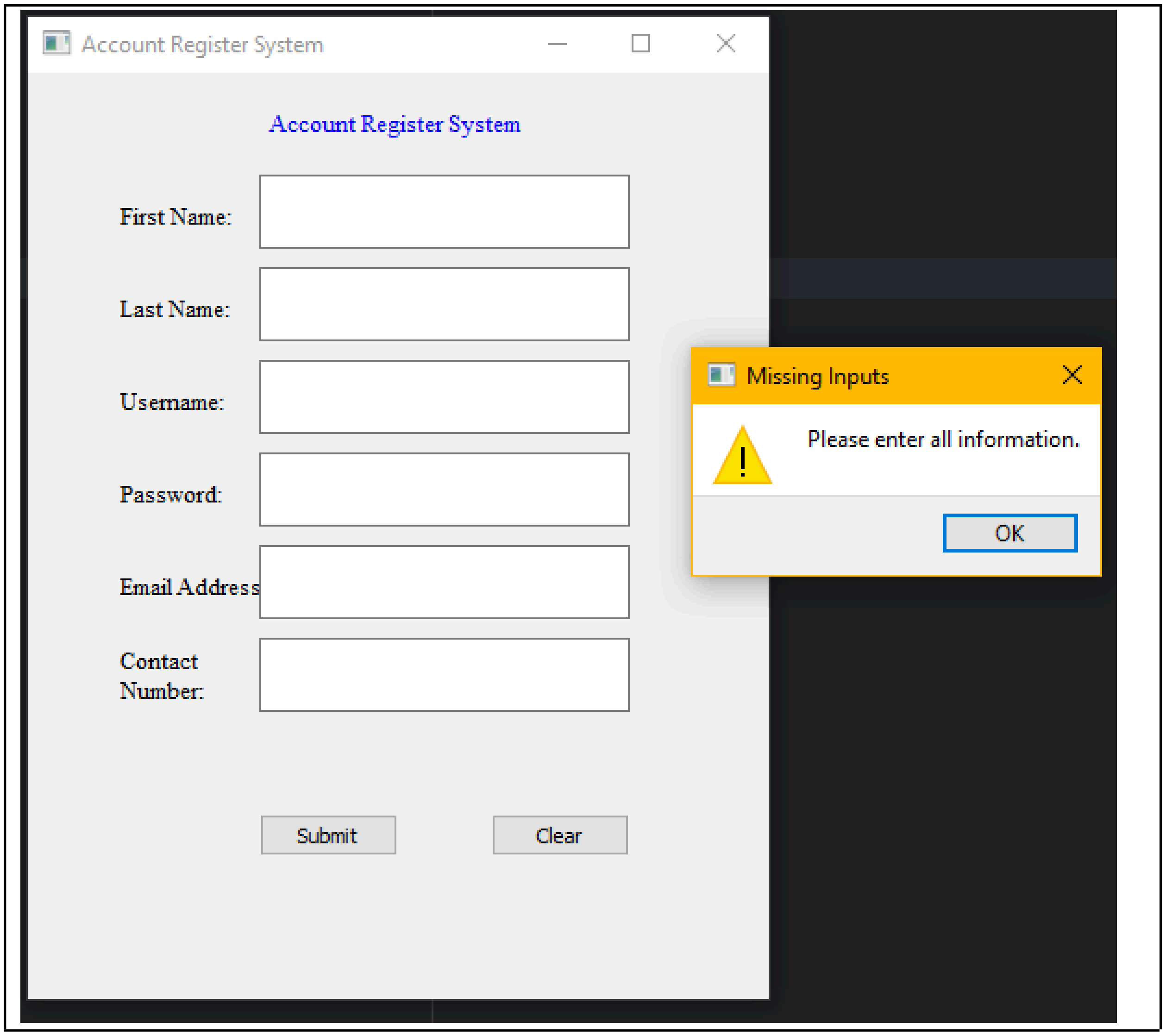
Username: andreibona

Password: 1234

Email Address andreibona@gmail.com

Contact Number: 09123456789

Successfully Submitted



Account Register System

Account Register System

First Name:

Last Name:

Username:

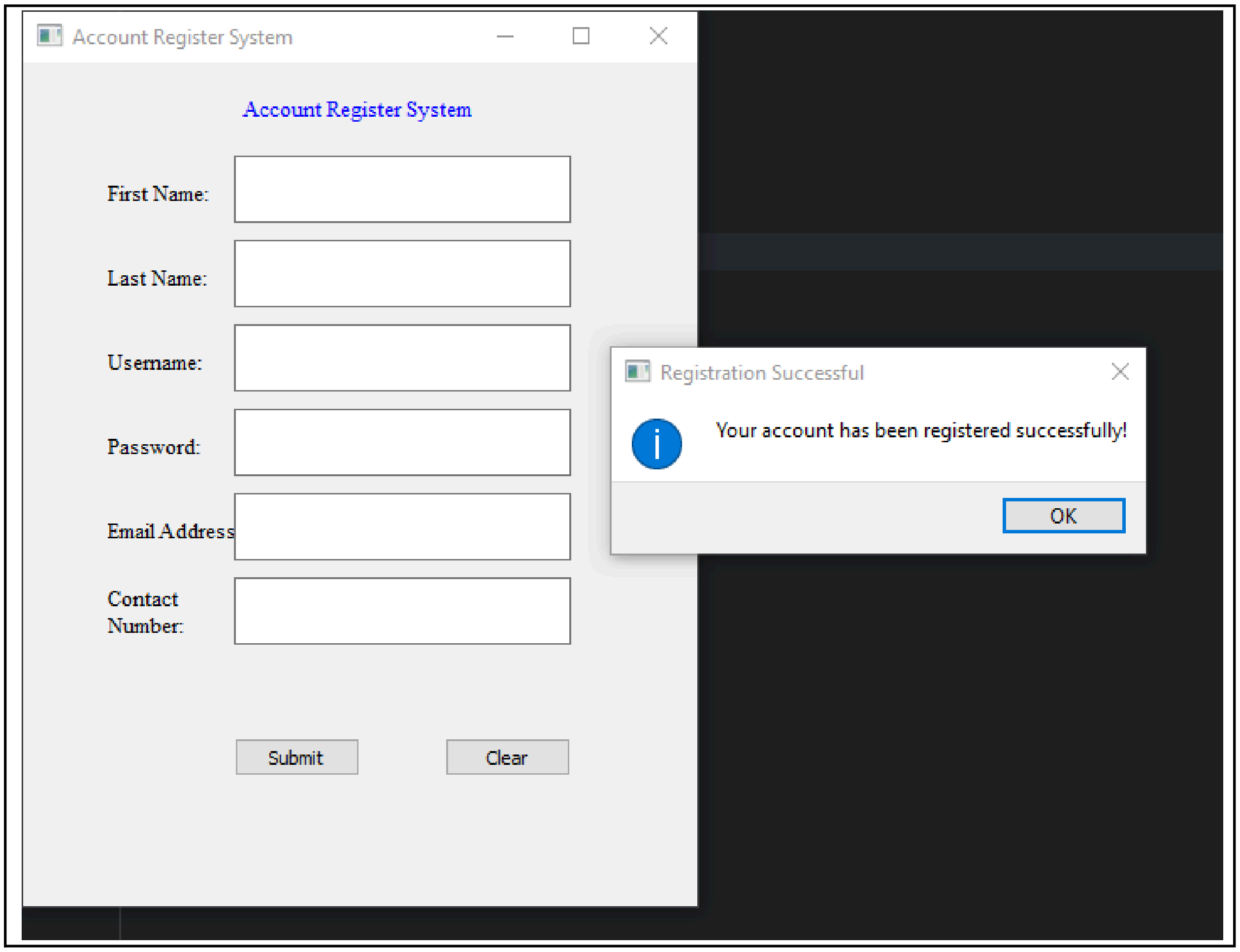
Password:

Email Address

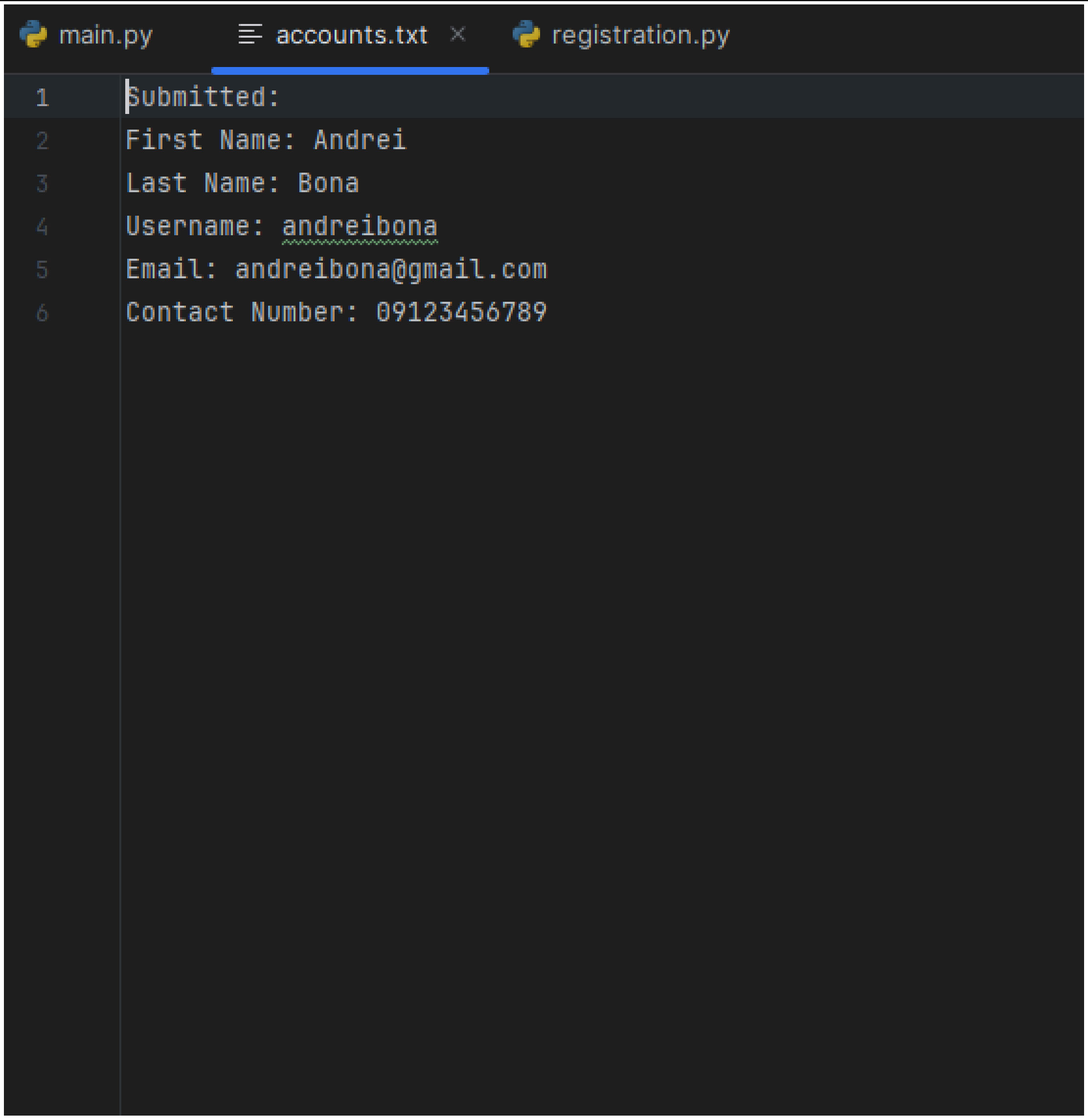
Contact Number:

Missing Inputs

Please enter all information.



Your account has been registered successfully!



```
Submitted:
First Name: Andrei
Last Name: Bona
Username: andreibona
Email: andreibona@gmail.com
Contact Number: 09123456789
```

Questions

1. What are the other signals available in PyQt5? (give at least 3 and describe each)
clicked(): This one gets triggered when you click a button.
textChanged(): It fires up whenever the text in a text box changes.
valueChanged(): This signal is emitted when you adjust the value of a slider or spin box.

2. Why do you think that event handling in Python is divided into signals and slots?
They keep things organized by separating what happens (signals) from how the app reacts (slots). This makes the code cleaner and easier to manage.

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?
They give users quick, clear feedback and help guide their decisions, making the app feel more user-friendly.

4. What is Error-handling and how was it applied in the task performed?
It's all about preparing for and managing mistakes in the app so it doesn't crash. It helps the app respond calmly to unexpected issues.

5. What maybe the reasons behind the need to implement error handling?
It boosts reliability, makes the user experience smoother, prevents crashes, and makes it easier to fix bugs later on.

7. Conclusion:

In conclusion, this activity successfully introduces us to the fundamentals of event handling in GUI applications. By identifying the key components of a GUI and creating a simple application using PyQt5 widgets, we gain practical skills that enhance our understanding of interactive software development. This foundational knowledge paves the way for more advanced programming concepts and applications in the future.

8. Assessment Rubric: