Laboratory Activity 4	
Reyes, Alexzander J.	10/14/24
CPE 009B - CPE21S4	Ma'am Rizette Sayo

Procedure:

Adding an icon:

```python
import sys
from PyQt5.QtWidgets import QMainWindow, QApplication
from PyQt5.QtGui import QIcon

class App(QMainWindow):

    def __init__(self):
        super().__init__()  # Initialize main window
        #window = QMainWIndow()
        self.title = "First OOP GUI"
        self.initUI()

    def initUI(self):
        self.setWindowTitle(self.title)
        self.setGeometry(200, 200, 300, 300)
        self.setWindowIcon(QIcon('pythonico.ico'))  # Sets an icon
        self.show()

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

Creating Buttons:

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

1 usage
class App(QWidget):

    def __init__(self):
        super().__init__()  # Initializes the main window
        self.title = "PyQt Button"
        self.x = 200  # Left
        self.y = 200  # 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'))

        # Create first button
        self.button = QPushButton('Click me!', self)
        self.button.setToolTip("You've hovered over me!")
        self.button.move(100, 70)  # button.move(x, y)

        # Create second button
        self.button2 = QPushButton('Register', self)
        self.button2.setToolTip("this button does nothing.. yet..")
        self.button2.move(100, 120)  # Positioning below the first button

        self.show()

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

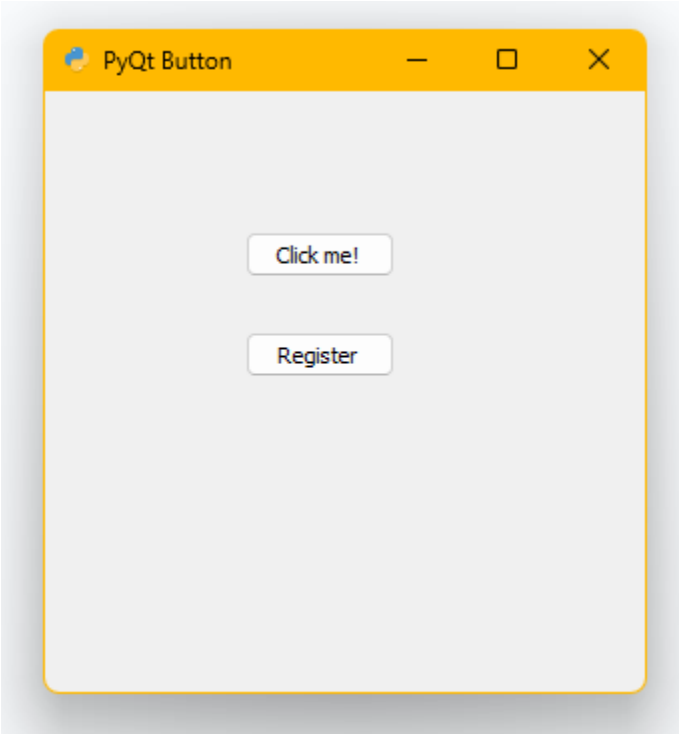
Creating Text Fields:

```python
import sys
from PyQt5.QtWidgets import QWidget, QApplication, QMainWindow, QPushButton, QLineEdit
from PyQt5.QtGui import QIcon

1 usage
class App(QWidget):

    def __init__(self):
        super().__init__()

        self.title = "PyQt Line Edit"
        self.x = 200  # Left
        self.y = 200  # 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'))

        # Create textbox
        self.textbox = QLineEdit(self)
        self.textbox.move(20, 20)
        self.textbox.resize(280, 40)

        self.show()

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

Creating Labels:

```python
import sys
from PyQt5.QtWidgets import QWidget, QApplication, QLabel, QLineEdit
from PyQt5.QtGui import QIcon

1 usage
class App(QWidget):

    def __init__(self):
        super().__init__()

        self.title = "PyQt Line Edit"
        self.x = 200  # Left
        self.y = 200  # 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'))

        # Create the first label
        self.textboxlbl = QLabel("Hello World!", self)
        self.textboxlbl.move(90, 25)  # Centered horizontally

        # Create the second label
        self.secondLabel = QLabel("This program is written in PyCharm", self)
        self.secondLabel.move(40, 75)  # Adjusted to be below the first label

        self.show()

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

Output:



SUPPLEMENTARY ACTIVITY

Registration.py

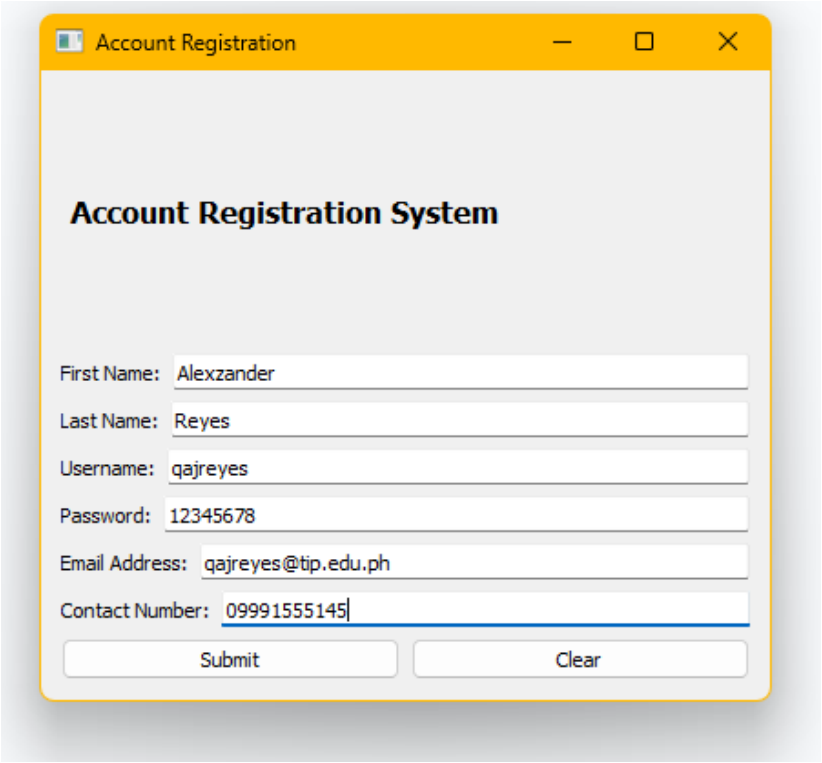
```python
from PyQt5.QtWidgets import QWidget, QLabel, QLineEdit, QPushButton, QVBoxLayout, QHBoxLayout, QApplication
from PyQt5.QtGui import QIcon
import sys


2 usages
class RegistrationForm(QWidget):
    def __init__(self):
        super().__init__()

        self.setWindowTitle("Account Registration")
        self.setGeometry(100, 100, 400, 400)
        self.setWindowIcon(QIcon('pythonico.ico'))

        self.initUI()

    1 usage
    def initUI(self):
        layout = QVBoxLayout()

        # Title Label
        title = QLabel("Account Registration System")
        title.setStyleSheet("font-size: 16px; font-weight: bold; margin-top: 5px; margin-bottom: 10px;")
        layout.addWidget(title)

        # Create Labels and Text Fields
        self.fields = [
            ("First Name:", QLineEdit()),
            ("Last Name:", QLineEdit()),
            ("Username:", QLineEdit()),
            ("Password:", QLineEdit()),
            ("Email Address:", QLineEdit()),
            ("Contact Number:", QLineEdit())
        ]

        for label_text, field in self.fields:
            label = QLabel(label_text)
            field.setPlaceholderText("Enter " + label_text.lower())
            h_layout = QHBoxLayout()
            h_layout.addWidget(label)
```

```python
            h_layout.addWidget(field)
            layout.addLayout(h_layout)

        # Create Submit and Clear Buttons
        self.submit_button = QPushButton("Submit")
        self.clear_button = QPushButton("Clear")

        button_layout = QHBoxLayout()
        button_layout.addWidget(self.submit_button)
        button_layout.addWidget(self.clear_button)

        layout.addLayout(button_layout)

        self.setLayout(layout)

        # Center the window
        self.center()

    1 usage
    def center(self):
        qr = self.frameGeometry()
        cp = QApplication.desktop().availableGeometry().center()
        qr.moveCenter(cp)
        self.move(qr.topLeft())
```

Main.py

```python
import sys
from PyQt5.QtWidgets import QApplication
from registration import RegistrationForm

if __name__ == '__main__':
    app = QApplication(sys.argv)
    form = RegistrationForm()
    form.show()
    sys.exit(app.exec_())
```

OUTPUT:



QUESTIONS

1. What are the common GUI Applications that general end-users such as home users, students, and office employees use? (give at least 3 and describe each)

- Web browsers: Among GUI apps, web browsers (like Google Chrome and Mozilla Firefox) are probably the most commonly used. They enable people to visit and use websites, enabling them to read news, watch videos, access social media, and use online tools.
- Word processors: These programs are necessary for creating, editing, and formatting documents (e.g., Microsoft Word, Google Docs). They provide an extensive graphical user interface (GUI) with toolbars, menus, and drag-and-drop text formatting and image or table insertion capabilities.
- Spreadsheet software (such as Google Sheets and Microsoft Excel): Spreadsheets let users calculate, make charts, and arrange data in rows and columns. Businesses utilize them extensively for reports, record-keeping, and data analysis.

2. Based from your answer in question 1, why do you think home users, students, and office employees use those GUI programs?

- Web browsers serve as the entry point to the internet. Students conduct online study, office workers frequently use browsers to access email and cloud-based applications, while home users use social media and leisure.

- Word processors are crucial tools for writers of essays, reports, and other works, both for students and professionals. They are essential for professional documentation since they are simple to format and add graphs or photos to.
- Spreadsheet software finds application in a wide range of fields, including personal budget management and professional data analysis and management. For organizing, calculating, and presenting data in business contexts, spreadsheets provide strong capability.

3. How does Pycharm help developers in making GUI applications, what would be the difference if developers made GUI programs without GUI Frameworks such as Pycharm or Tkinter?

- With built-in debugging tools, code guidance, and support for well-known GUI frameworks like Tkinter and PyQt, PyCharm makes developing GUI applications easier. The procedure is streamlined, resulting in increased speed and efficiency. Developers would have to spend more time and effort on laborious, difficult activities without GUI frameworks, such as handling low-level issues like cross-platform compatibility, event management, and window creation. Building and maintaining GUI apps is significantly simpler when frameworks and an IDE like PyCharm are used. Productivity is also increased and errors are decreased.

4. What are the different platforms a GUI program may be created and deployed on? (Three is required then state why might a program be created on that specific platform)

- Windows is one of the most popular platforms for deploying GUI applications, especially in business and home environments. It is widely used by office employees, students, and home users, making it a prime target for developers who want their software to reach a large audience. Many enterprise applications, games, and productivity tools are designed for Windows due to its dominant market share and compatibility with Microsoft Office tools and services.
- macOS is a preferred platform for creative professionals in fields such as design, video editing, and music production. Applications developed for macOS are often built to take advantage of the system's polished graphical interface, seamless integration with Apple's ecosystem, and optimized hardware-software performance. Developers target macOS for its reliability and appeal to users seeking high-quality design tools and software experiences.
- Linux is favored for developing and deploying open-source and technical applications, particularly for developers, system administrators, and scientific communities. It provides a high level of control, customization, and scripting capabilities, making it ideal for highly technical software, network tools, and scientific programs. Additionally, many developers create cross-platform GUI applications that can run on Linux to cater to technical audiences that prefer open-source environments.

5. What is the purpose of app = QApplication(sys.argv), ex = App(), and sys.exit(app.exec_())?

- app = QApplication(sys.argv) initializes the application and its event handling, ex = App() creates and displays the main window and its GUI elements, and sys.exit(app.exec_()) starts the event loop and ensures the program exits cleanly when it's closed. These lines are essential for running a Qt-based GUI application in Python.

CONCLUSION

In this task, we used PyCharm to create an object-oriented graphical user interface (GUI) for a basic account registration system. We used fundamental GUI development techniques, such Absolute Positioning and dynamic component positioning, by arranging the necessary user data (first and last name, username, password, email address, and contact number) in an understandable and structured manner. Labels were positioned next to text fields and action buttons (submit and clear) were strategically arranged in the layout to guarantee user-friendliness. Best coding techniques were followed to ensure modularity and maintainability of the GUI code by separating it into registration.py and launching it via main.py. All things considered, this work gave me real-world experience using PyCharm to create a neat, functional, and aesthetically organized GUI application.