Laboratory Activity No. 4 - Introduction to GUI Development using Pycharm

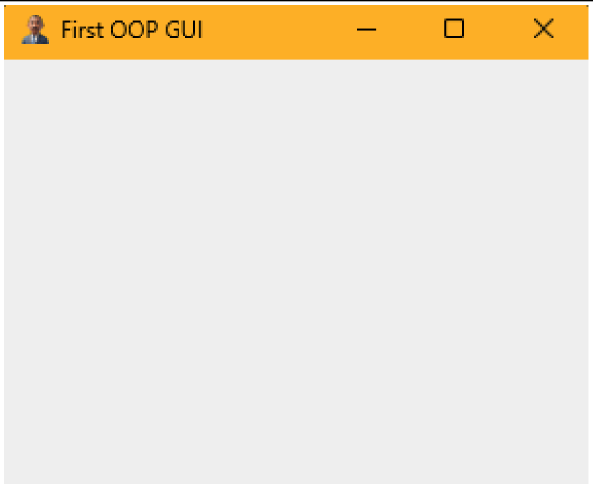
Sanchez, Christan Ray , R.	10/14/2024
CPE009 - CPE21S4	Prof. Ma. Rizette Sayo

5. Procedure:

Source Code

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

class App(QMainWindow):

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

        self.title = "First OOP GUI"
        self.initUI()

    def initUI(self):

self.setWindowTitle(self.title)

self.setGeometry(200,200,300,300)

self.setWindowIcon(QIcon('pythonico.i
co'))
        self.show()

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

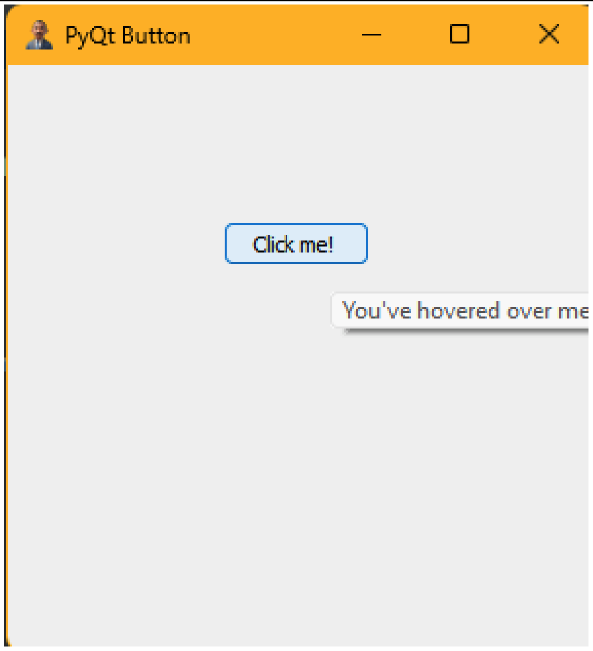
Output



Source Code

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

```
class App(QWidget):

   def __init__(self):
       super().__init__()  # Fixed
the super() call
       self.title = "PyQt Button"  #
Fixed assignment operator
       self.x = 200  # Fixed
assignment operator
       self.y = 200  # Fixed
assignment operator
       self.width = 300  # Fixed
assignment operator
       self.height = 300  # Fixed
spelling and assignment operator
       self.initUI()

   def initUI(self):

self.setWindowTitle(self.title)
       self.setGeometry(self.x,
self.y, self.width, self.height)

self.setWindowIcon(QIcon('pythonico.i
co'))  # Fixed filename

       self.button =
QPushButton('Click me!', self)
       self.button.setToolTip("You've
hovered over me!")
       self.button.move(110, 80)

       self.show()

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

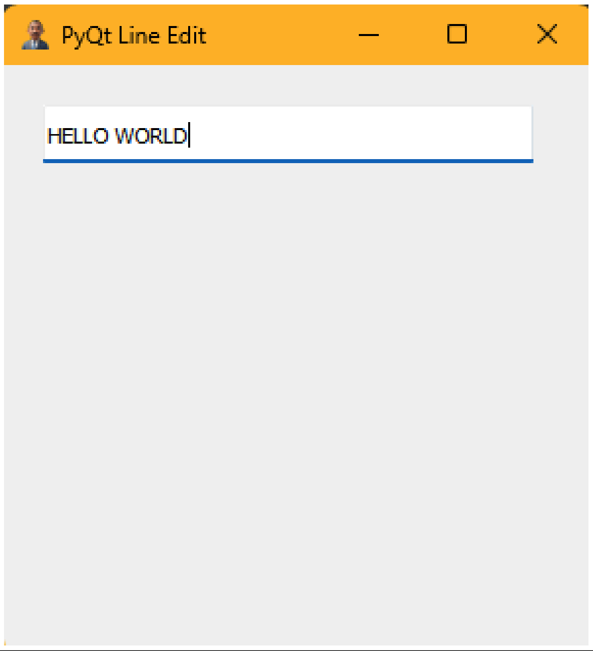
Output	

Source Code

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

class App(QWidget):
   def __init__(self):
       super().__init__()#initializes
the main window like in the previous
one
#window = QMainWindow()
       self.title="PyQt Line Edit"
       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.i
co'))

       #Create text
       self.textbox=QLineEdit(self)
       self.textbox.move(20,20)
       self.textbox.resize(280,30)
       self.textbox.setText("Set this
text value")
       self.show()
```

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

Output


PyQt Line Edit

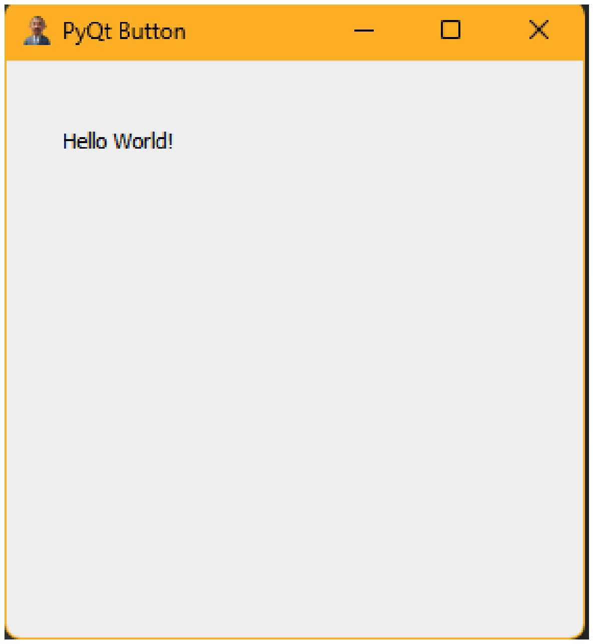
HELLO WORLD

Source Code

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


class App(QWidget):

   def __init__(self):
       super().__init__()
       self.title = "PyQt Button"
       self.x = 200
       self.y = 200
       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.i
co'))
```

```
        self.textboxlbl =
QLabel("Hello World! ", self)
        self.textboxlbl.move(30,35)

        self.show()

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

Output



```
Hello World!
```

SUPPLEMENTARY ACTIVITY

SOURCE CODE

```
# registration.py
from PyQt5.QtWidgets import QWidget,
QLabel, QLineEdit, QPushButton,
QVBoxLayout, QHBoxLayout, QMessageBox
from PyQt5.QtGui import QIcon
from PyQt5.QtCore import Qt
import re


class RegistrationForm(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Account
Registration")
        self.setGeometry(100, 100,
400, 300)

        # Set the window icon
```

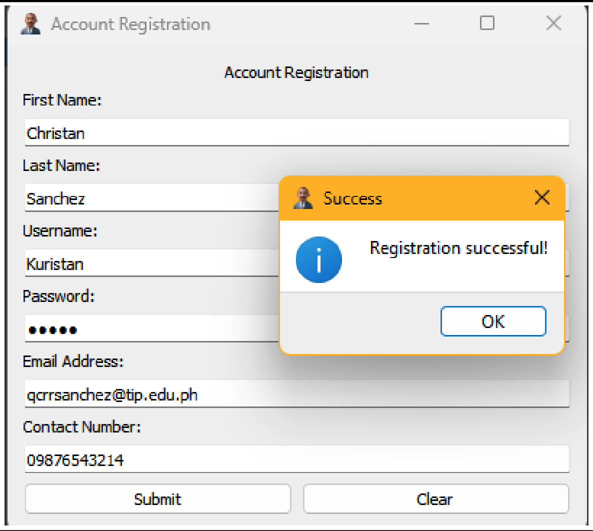
```python
self.setWindowIcon(QIcon('pythonico.i
co'))

       self.initUI()

   def initUI(self):
       title = QLabel("Account
Registration")

title.setAlignment(Qt.AlignCenter)

       self.first_name_label,
self.first_name_input =
self.create_input_field("First
Name:")
       self.last_name_label,
self.last_name_input =
self.create_input_field("Last Name:")
       self.username_label,
self.username_input =
self.create_input_field("Username:")

       self.password_label,
self.password_input =
self.create_input_field("Password:")

self.password_input.setEchoMode(QLine
Edit.Password)

       self.email_label,
self.email_input =
self.create_input_field("Email
Address:")
       self.contact_number_label,
self.contact_number_input =
self.create_input_field("Contact
Number:")

       self.submit_button =
QPushButton("Submit")
       self.clear_button =
QPushButton("Clear")


self.submit_button.clicked.connect(se
lf.submit_form)

self.clear_button.clicked.connect(sel
f.clear_form)

       form_layout = QVBoxLayout()
       form_layout.addWidget(title)

form_layout.addWidget(self.first_name
_label)
```

```
form_layout.addWidget(self.first_name
_input)

form_layout.addWidget(self.last_name_
label)

form_layout.addWidget(self.last_name_
input)

form_layout.addWidget(self.username_l
abel)

form_layout.addWidget(self.username_i
nput)

form_layout.addWidget(self.password_l
abel)

form_layout.addWidget(self.password_i
nput)

form_layout.addWidget(self.email_labe
l)

form_layout.addWidget(self.email_inpu
t)

form_layout.addWidget(self.contact_nu
mber_label)

form_layout.addWidget(self.contact_nu
mber_input)

       button_layout = QHBoxLayout()

button_layout.addWidget(self.submit_b
utton)

button_layout.addWidget(self.clear_bu
tton)


form_layout.addLayout(button_layout)
       self.setLayout(form_layout)

   def create_input_field(self,
label_text):
       label = QLabel(label_text)
       input_field = QLineEdit()
       input_field.setToolTip(f"Enter
your {label_text.lower()}")
       return label, input_field

   def validate_email(self, email):
       pattern =
r"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.
[a-zA-Z]{2,}$"
       return re.match(pattern,
email) is not None

   def submit_form(self):
       if not
self.validate_email(self.email_input.
text()):
           QMessageBox.warning(self,
"Error", "Please enter a valid email
address.")
           return
       QMessageBox.information(self,
"Success", "Registration
successful!")

   def clear_form(self):
       self.first_name_input.clear()
       self.last_name_input.clear()
       self.username_input.clear()
       self.password_input.clear()
       self.email_input.clear()

self.contact_number_input.clear()
```

FOR “register.py”

SOURCE CODE

```
# main.py
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	

Account Registration

Account Registration

First Name: Christan

Last Name: Sanchez

Username: Kuristan

Password: ●●●●●

Email Address: qcrrsanchez@tip.edu.ph

Contact Number: 09876543214

Success

Registration successful!

1. Common GUI Applications

- Web Browsers (e.g., Chrome, Firefox): These applications allow users to navigate the internet, access information, and interact with web services. They offer a user-friendly interface for searching, bookmarking, and browsing.
- Office Suites (e.g., Microsoft Office, Google Workspace): These applications include word processors, spreadsheets, and presentation software. They provide tools for document creation, data analysis, and presentations, catering to both professional and personal needs.
- Media Players (e.g., VLC, Windows Media Player): These applications enable users to play audio and video files. They often support various formats and provide features like playlists, equalizers, and streaming capabilities.

2. Reasons for Usage

Home users, students, and office employees gravitate towards these GUI applications due to their intuitive design and ease of use. They simplify complex tasks, allowing users to focus on productivity without needing extensive technical knowledge. The accessibility and familiarity of these tools also make them essential for daily tasks, learning, and collaboration.

3. Benefits of Using PyCharm for GUI Development

PyCharm enhances GUI development by providing integrated development environment (IDE) features like syntax highlighting, code completion, and debugging tools. It streamlines the workflow and reduces development time. Without frameworks like PyCharm or Tkinter, developers would face more challenges, such as manual GUI layout coding, managing event handling, and ensuring cross-platform compatibility, making the process significantly more complex and error-prone.

4. Platforms for GUI Development

- Windows: Many businesses and home users operate on Windows. Applications developed here often target a wide audience, leveraging the OS's extensive user base.
- Linux: Open-source enthusiasts and developers prefer Linux for its flexibility and customization options. Apps created here can benefit from the community-driven support and are often used in server environments.
- macOS: Designers and creative professionals favor macOS for its design aesthetics and user experience. Developing applications for this platform can tap into a market that values high-quality software with a focus on design.

5. Purpose of Key Lines in PyQt Applications

- `app = QApplication(sys.argv)`: Initializes the application and manages the control flow and main settings. It allows the program to interact with the system's event loop.

- `ex = App()`: Creates an instance of the main application window (in this case, the `App` class). This is where the GUI components are defined and organized.
- `sys.exit(app.exec_())`: Enters the application's main event loop. The program will run until the user closes the window. The `sys.exit()` ensures that the program exits cleanly, returning a status code to the operating system.

CONCLUSION

In conclusion, exploring GUI application development has been a rewarding experience. I now have a solid understanding of how to create user-friendly interfaces for basic systems, allowing me to bring ideas to life visually. This newfound knowledge not only enhances my programming skills but also opens up exciting possibilities for future projects. I truly enjoyed the process and look forward to applying what I've learned!