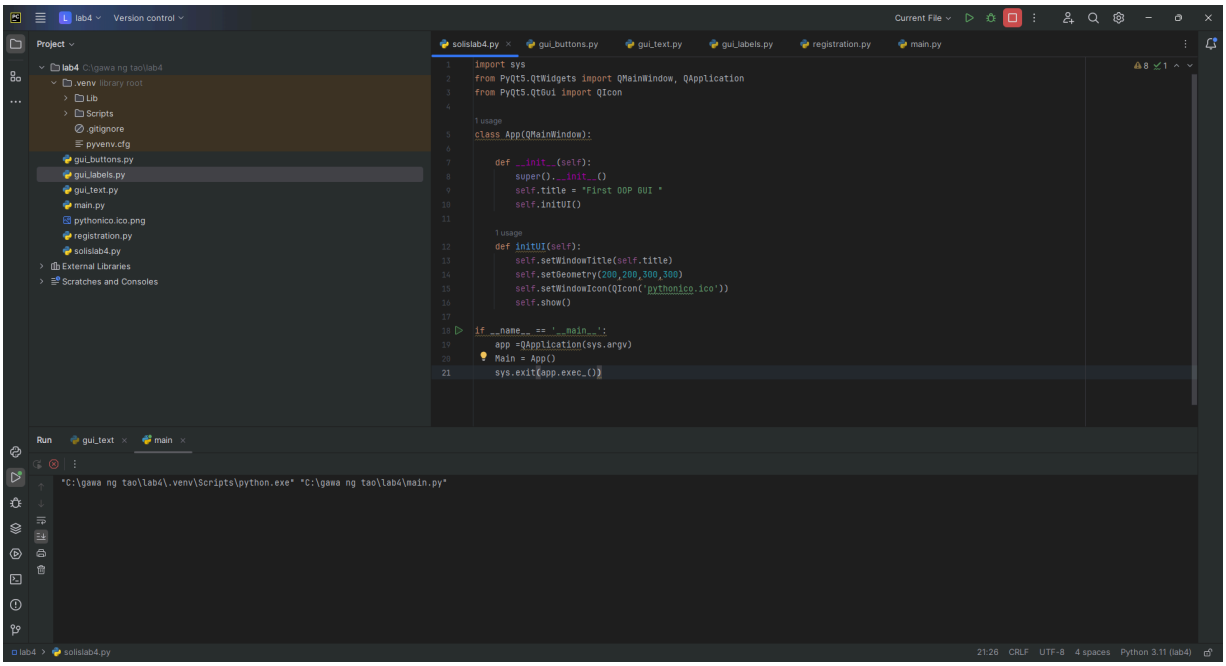
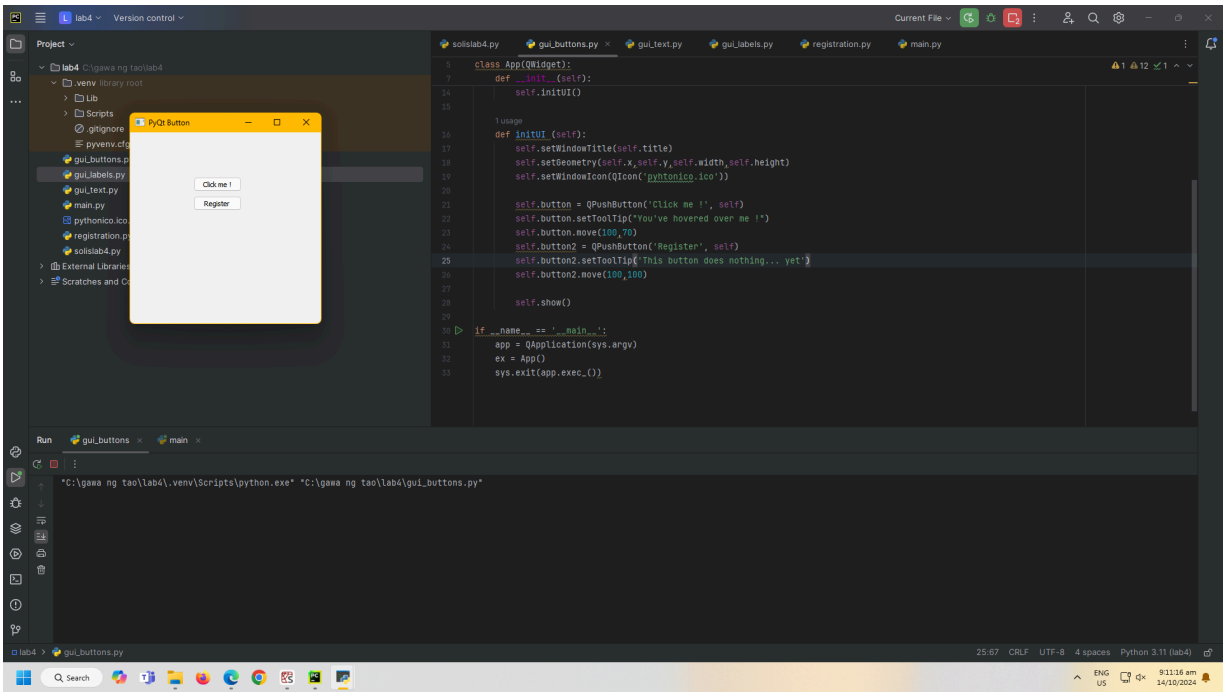
| ntroduction to GUI Development using Pycharm<br>**#<4> - I**ntroduction to GUI Development using Pycharm | |
|---|---|
| Solis, Paul VIncent M. | 10/14/24 |
| CPE21S4 / CPE - 009B | Prof. Sayo |

PROCEDURE:

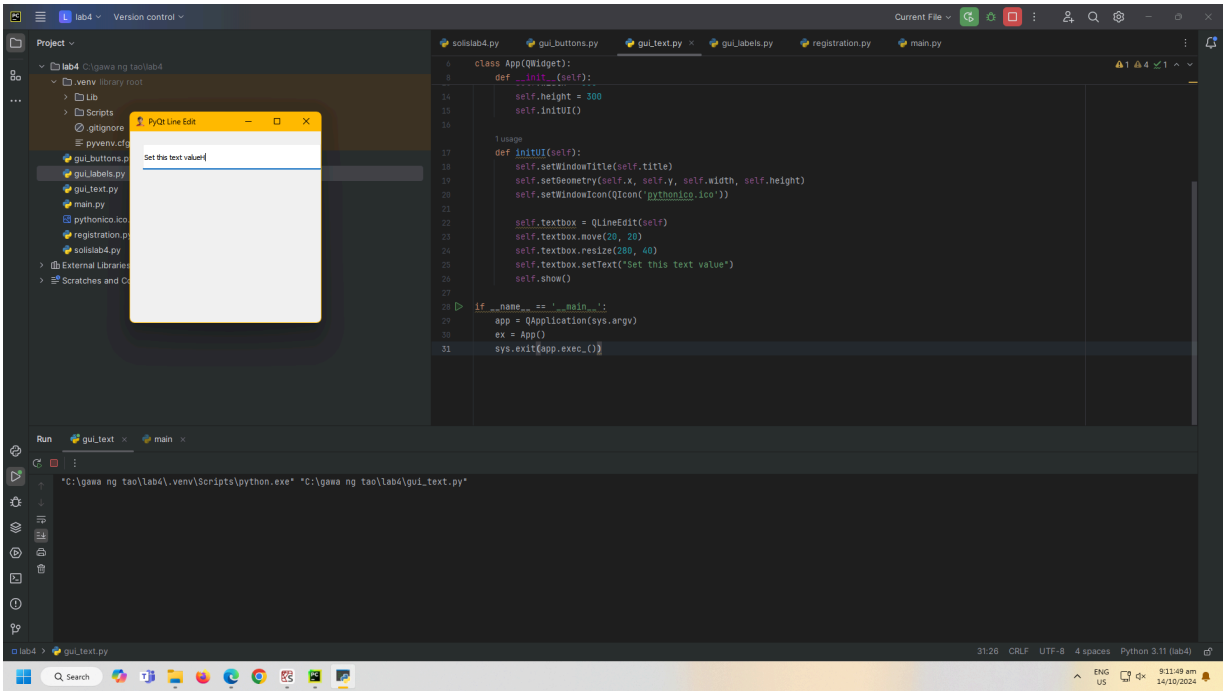
1.



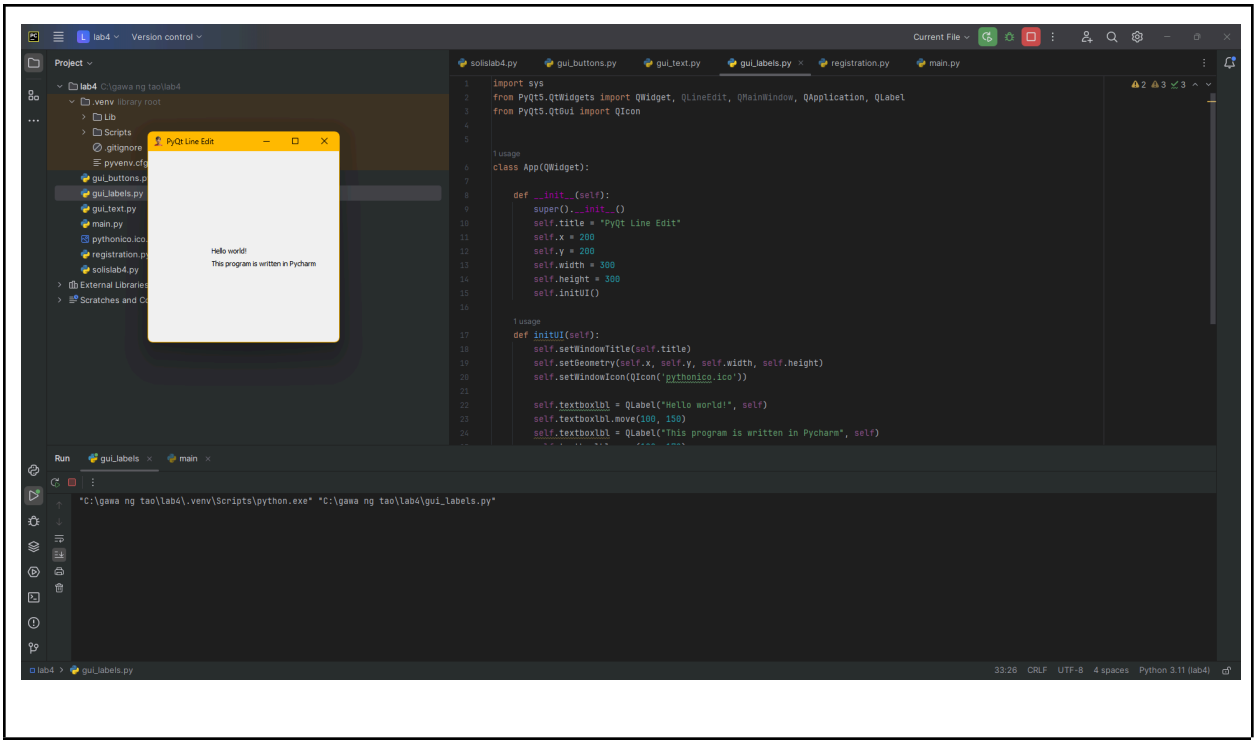

2.


PyQt Button
Click me !
Register
class App(QWidget):
def __init__(self):
self.initUI()
def initUI (self):
self.setWindowTitle(self.title)
self.setGeometry(self.x,self.y,self.width,self.height)
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.button2 = QPushButton('Register', self)
self.button2.setToolTip('This button does nothing... yet')
self.button2.move(100,100)
self.show()
if __name__ == '__main__':
app = QApplication(sys.argv)
ex = App()
sys.exit(app.exec_())
"C:\gawa ng tao\lab4\.venv\Scripts\python.exe" "C:\gawa ng tao\lab4\gui_buttons.py"

3.


PyQt Line Edit
Set this text value
class App(QWidget):
def __init__(self):
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'))
self.textbox = QLineEdit(self)
self.textbox.move(20, 20)
self.textbox.resize(280, 40)
self.textbox.setText("Set this text value")
self.show()
if __name__ == '__main__':
app = QApplication(sys.argv)
ex = App()
sys.exit(app.exec_())
"C:\gawa ng tao\lab4\.venv\Scripts\python.exe" "C:\gawa ng tao\lab4\gui_text.py"

4.




SUPPLEMENTARY ACTIVITY:

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)
    - Common GUI applications used by home users, students, and office employees include web browsers like Google Chrome, Firefox, and Safari, as well as Microsoft Office Suite applications like Word, Excel, PowerPoint, and Outlook. Additionally, many users interact with GUI-based operating systems like Windows, MacOS, and Linux distributions.
2. Based from your answer in question 1, why do you think home users, students, and office employees use those GUI programs?
    - Because these GUI applications offer an intuitive and user-friendly interface that makes it simple to access the internet, create and edit documents, and manage computer systems, they are used by home users, students, and office workers. This increases productivity, efficiency, and the overall computing experience.
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?
        - Firstly, PyCharm offers a GUI Designer, which allows developers to visually design and build GUI applications using a drag-and-drop interface. This feature enables developers to focus on the layout and appearance of their application without having to write extensive code. Secondly, PyCharm provides Code Completion, which helps developers to write code more efficiently by suggesting possible completions for classes, methods, and variables. This feature is particularly useful when working with GUI frameworks like Tkinter, PyQt, or wxPython. Thirdly, PyCharm offers Code Inspection, which analyzes the code for potential errors,

warnings, and improvements. This feature helps developers to identify and fix issues early on, ensuring that their GUI application is more stable and reliable. Lastly, PyCharm provides Debugging Tools, which enable developers to debug their GUI application more effectively. These tools allow developers to set breakpoints, inspect variables, and step through their code, making it easier to identify and fix issues

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)
    - A GUI program can be created and deployed on various platforms, including Windows, macOS, Linux, and mobile devices (iOS and Android), as well as on the web as web applications, and even in the cloud. Some GUI programs may be created on specific platforms due to Windows' widespread use and compatibility, macOS' sleek and user-friendly interface, Linux' open-source nature and flexibility, mobile devices' increasing demand for on-the-go accessibility, the web's need for accessibility from anywhere, and the cloud's scalability, security, and accessibility features.
5. What is the purpose of app = QApplication(sys.argv), ex = App(), and sys.exit(app.exec_())?
- The purpose of app = QApplication(sys.argv), ex = App(), and sys.exit(app.exec_()) is to create a Qt application instance, instantiate the main application window, and start the application's event loop, respectively, which together enable the GUI application to run and respond to user interactions.

CONCLUSION

- We talked about using Python and the Tkinter package to create a GUI application for a basic account registration system. First, we examined the functions of ex = App(), sys.exit(app.exec_()), and app = QApplication(sys.argv) in a PyQt5 program. Next, we updated the code to incorporate saving and input validation. The Account Registration System required fields for first and last names, usernames, passwords, email addresses, and phone numbers, therefore we next developed an object-oriented GUI application for it. The components of the GUI were arranged using absolute positioning, and the GUI was centered on the screen. In addition, we developed methods to submit and clear the input data, added a program title, and provided submit and clear buttons. We discussed a number of topics related to developing GUI applications during our talk, such as code organization, event processing, as well as input verification. The completed implementation produced an operational Account Registration System and complied with the requirements.

  registration.py

Python

```
from tkinter import *
```

```
class RegistrationWindow:
    def __init__(self, win):
        self.win = win
        self.win.title("Account Registration")
        self.win.geometry("400x400+100+100")
        self.win.configure(bg="LightGray")

        # Program Title
        self.title_label = Label(win, fg="Blue", font=("Helvetica", 16),
text="Account Registration")
        self.title_label.place(x=100, y=20)

        # Labels and Entry Fields
        self.labels = ["First Name", "Last Name", "Username", "Password",
"Email Address", "Contact Number"]
        self.entries = []

        for i, label in enumerate(self.labels):
            lbl = Label(win, fg="Green", text=label)
            lbl.place(x=50, y=70 + (i * 40))
            entry = Entry(win, bd=4)
            entry.place(x=150, y=70 + (i * 40))
            self.entries.append(entry)

        # Submit and Clear Buttons
        self.submit_button = Button(win, fg="Red", text="Submit",
command=self.submit)
        self.submit_button.place(x=100, y=320)

        self.clear_button = Button(win, fg="Red", text="Clear",
command=self.clear)
        self.clear_button.place(x=200, y=320)

    def submit(self):
        # Here you can add logic to save the input data
        user_data = {label: entry.get() for label, entry in
zip(self.labels, self.entries)}
        print("User  Data Submitted:", user_data)  # Replace with actual
saving logic

    def clear(self):
```

```
        for entry in self.entries:
            entry.delete(0, 'end')
```

main.py

Python

```
import sys
from tkinter import Tk
from registration import RegistrationWindow

if __name__ == "__main__":
    window = Tk()
    app = RegistrationWindow(window)
    window.configure(bg='gray')
    window.mainloop()
```

OUTPUT: