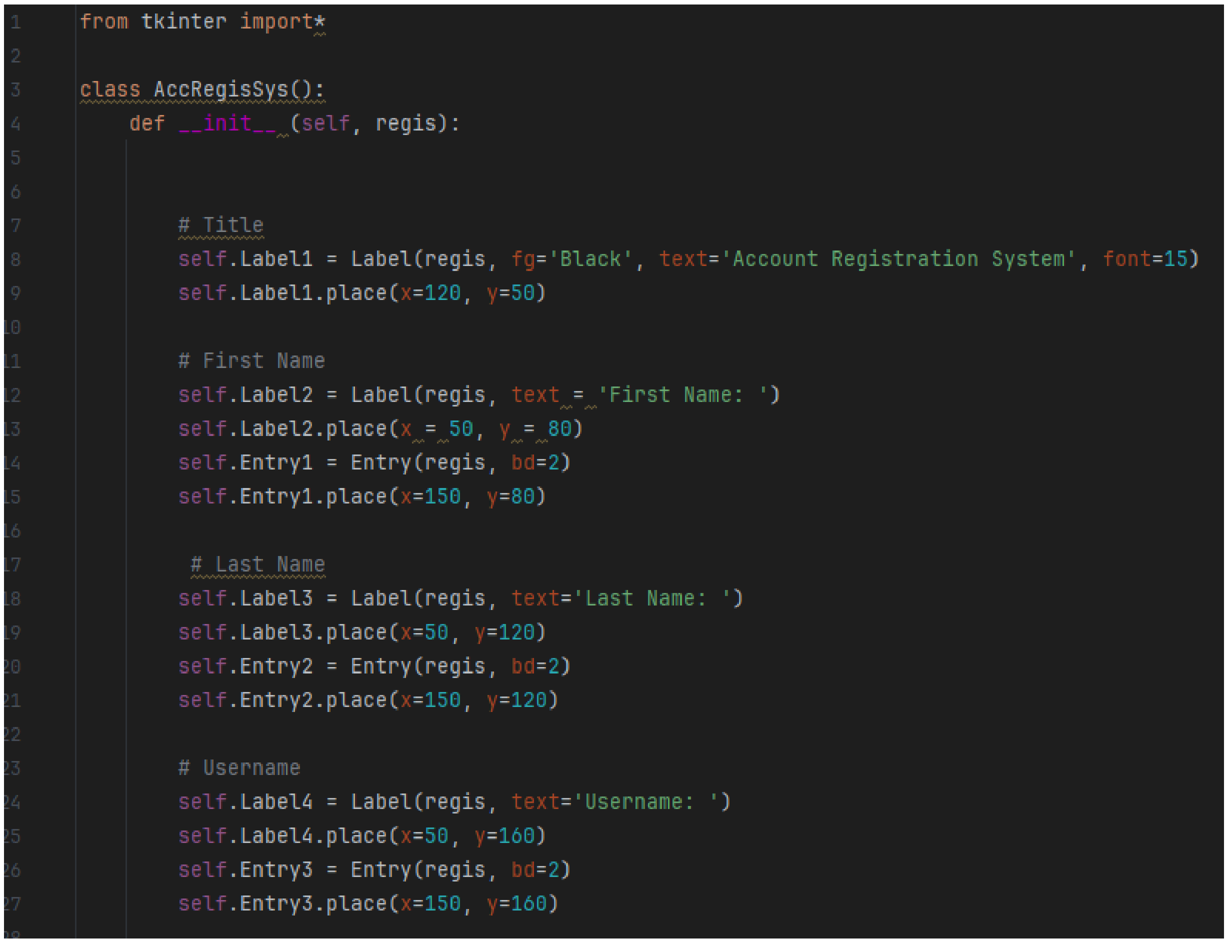
Laboratory Activity No. 4 - Introduction to GUI Development using Pycharm	
Name: San Juan, Edson Ray E	10/14/2024
Course/Section: CPE21S4	Prof. Maria Rizette Sayo

6. Supplementary Activity:

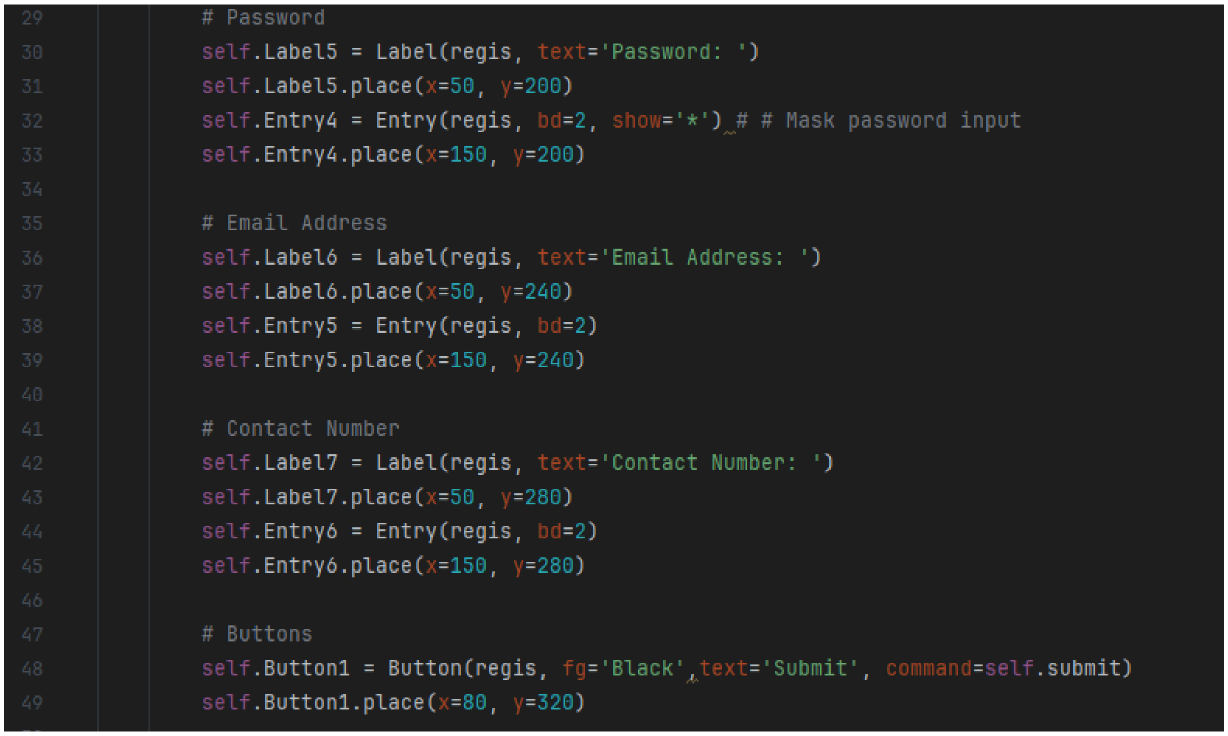
Task
Create an Object-Oriented GUI Application for a simple Account Registration System with the following required information: first name, last name, username, password, email address, contact number.

Requirements:
- The GUI must be centered on your screen.
- The GUI Components should be organized according to the order of information required using Absolute Positioning.
- The position of the components should be automatically computed based on the top component.
- All the text fields should be accompanied with their corresponding label on the left side while the text field is on the right side.
- There should a program title other than the Window Title.
- There should be a submit button and clear button at the bottom (submit button on the left, clear button on the right).
- The program should be launched on **main.py** while the GUI Codes should be on a separate file called
  registration.py

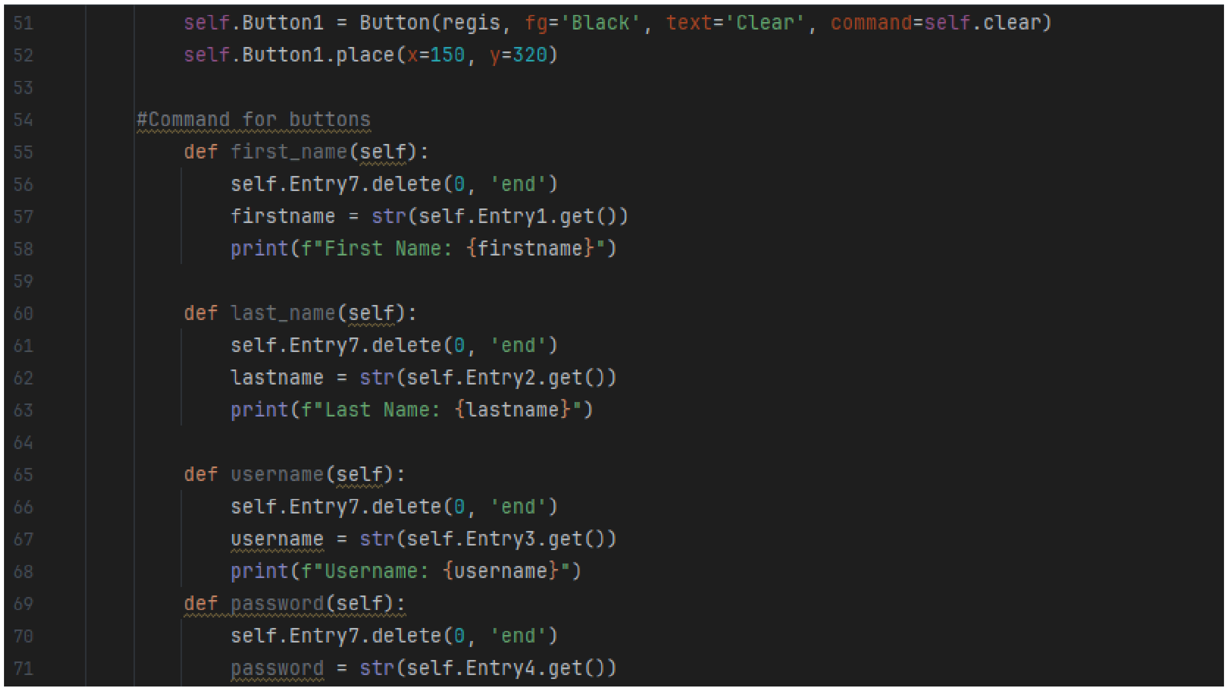
Code (registration.py):



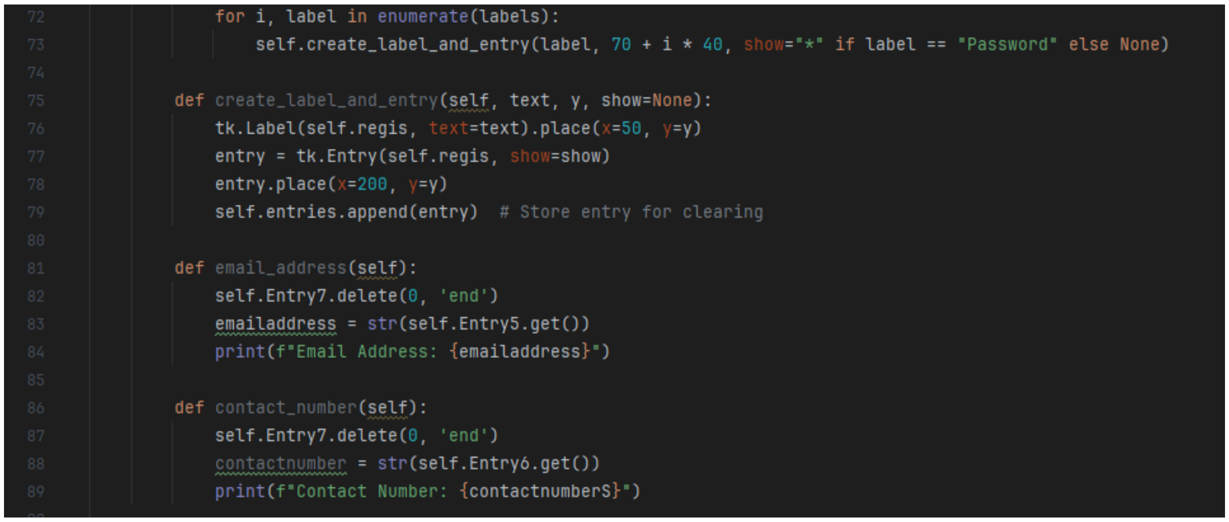
```python
from tkinter import*

class AccRegisSys():
    def __init__(self, regis):


        # Title
        self.Label1 = Label(regis, fg='Black', text='Account Registration System', font=15)
        self.Label1.place(x=120, y=50)

        # First Name
        self.Label2 = Label(regis, text = 'First Name: ')
        self.Label2.place(x = 50, y = 80)
        self.Entry1 = Entry(regis, bd=2)
        self.Entry1.place(x=150, y=80)

         # Last Name
        self.Label3 = Label(regis, text='Last Name: ')
        self.Label3.place(x=50, y=120)
        self.Entry2 = Entry(regis, bd=2)
        self.Entry2.place(x=150, y=120)

        # Username
        self.Label4 = Label(regis, text='Username: ')
        self.Label4.place(x=50, y=160)
        self.Entry3 = Entry(regis, bd=2)
        self.Entry3.place(x=150, y=160)
```



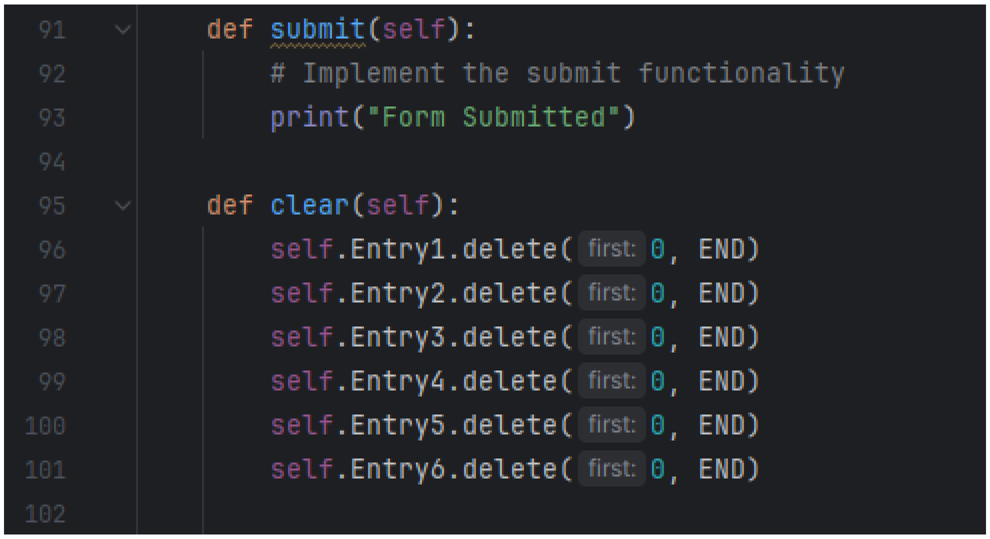
```python
        # Password
        self.Label5 = Label(regis, text='Password: ')
        self.Label5.place(x=50, y=200)
        self.Entry4 = Entry(regis, bd=2, show='*') # # Mask password input
        self.Entry4.place(x=150, y=200)

        # Email Address
        self.Label6 = Label(regis, text='Email Address: ')
        self.Label6.place(x=50, y=240)
        self.Entry5 = Entry(regis, bd=2)
        self.Entry5.place(x=150, y=240)

        # Contact Number
        self.Label7 = Label(regis, text='Contact Number: ')
        self.Label7.place(x=50, y=280)
        self.Entry6 = Entry(regis, bd=2)
        self.Entry6.place(x=150, y=280)

        # Buttons
        self.Button1 = Button(regis, fg='Black',text='Submit', command=self.submit)
        self.Button1.place(x=80, y=320)
```



```python
    self.Button1 = Button(regis, fg='Black', text='Clear', command=self.clear)
    self.Button1.place(x=150, y=320)

#Command for buttons
    def first_name(self):
        self.Entry7.delete(0, 'end')
        firstname = str(self.Entry1.get())
        print(f"First Name: {firstname}")

    def last_name(self):
        self.Entry7.delete(0, 'end')
        lastname = str(self.Entry2.get())
        print(f"Last Name: {lastname}")

    def username(self):
        self.Entry7.delete(0, 'end')
        username = str(self.Entry3.get())
        print(f"Username: {username}")
    def password(self):
        self.Entry7.delete(0, 'end')
        password = str(self.Entry4.get())
```

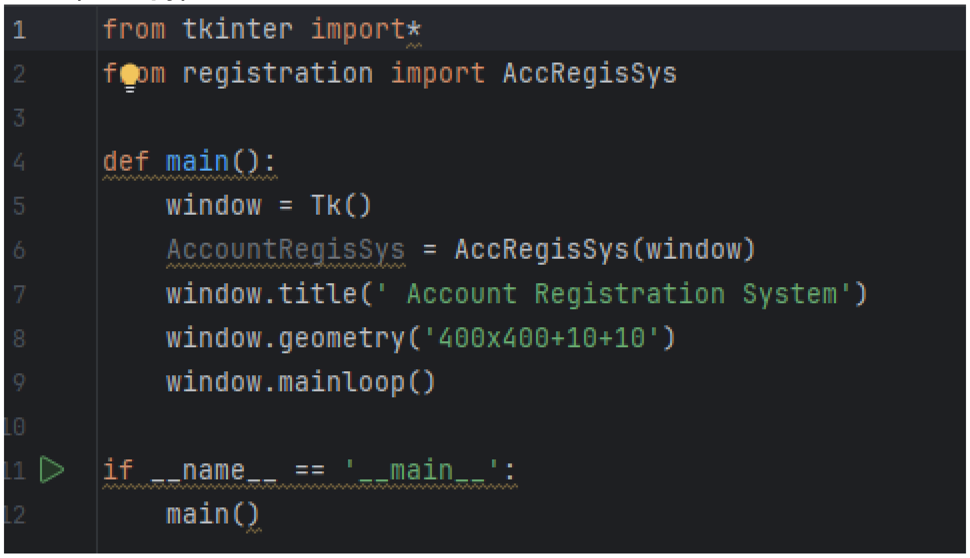


```python
        for i, label in enumerate(labels):
            self.create_label_and_entry(label, 70 + i * 40, show="*" if label == "Password" else None)

    def create_label_and_entry(self, text, y, show=None):
        tk.Label(self.regis, text=text).place(x=50, y=y)
        entry = tk.Entry(self.regis, show=show)
        entry.place(x=200, y=y)
        self.entries.append(entry)  # Store entry for clearing

    def email_address(self):
        self.Entry7.delete(0, 'end')
        emailaddress = str(self.Entry5.get())
        print(f"Email Address: {emailaddress}")

    def contact_number(self):
        self.Entry7.delete(0, 'end')
        contactnumber = str(self.Entry6.get())
        print(f"Contact Number: {contactnumberS}")
```



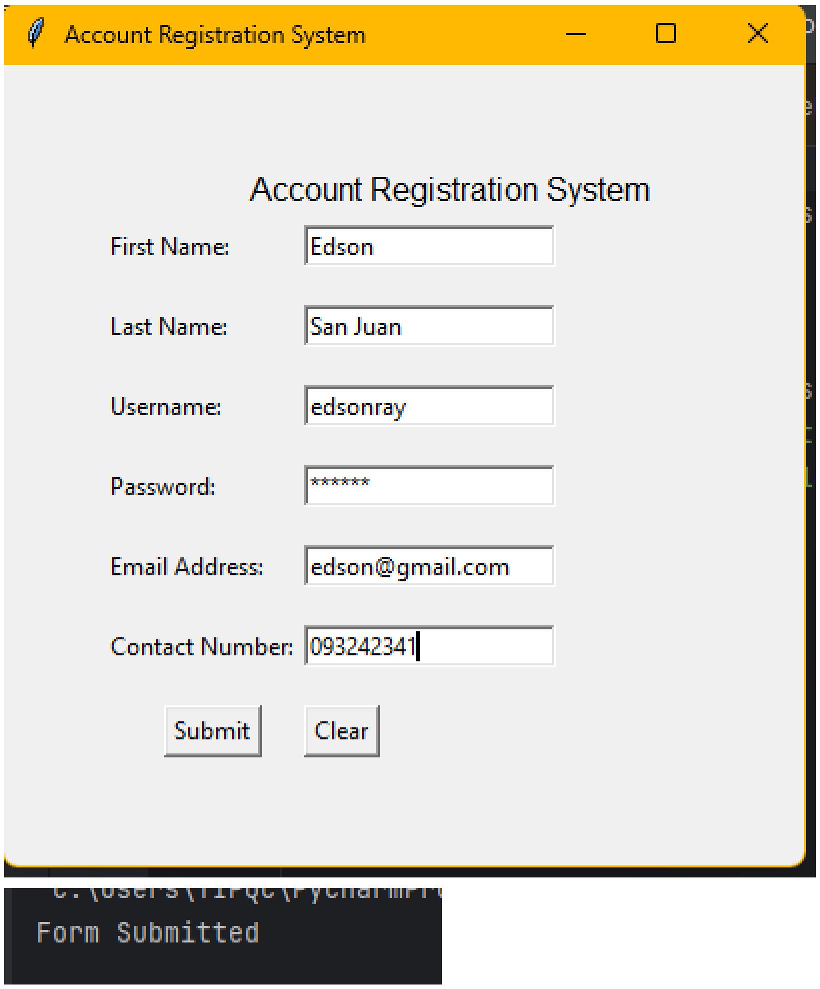
```python
    def submit(self):
        # Implement the submit functionality
        print("Form Submitted")

    def clear(self):
        self.Entry1.delete( first: 0, END)
        self.Entry2.delete( first: 0, END)
        self.Entry3.delete( first: 0, END)
        self.Entry4.delete( first: 0, END)
        self.Entry5.delete( first: 0, END)
        self.Entry6.delete( first: 0, END)

```

Code (main.py)



```python
from tkinter import*
from registration import AccRegisSys

def main():
    window = Tk()
    AccountRegisSys = AccRegisSys(window)
    window.title(' Account Registration System')
    window.geometry('400x400+10+10')
    window.mainloop()

if __name__ == '__main__':
    main()
```

Output:



Account Registration System

First Name:	Edson
Last Name:	San Juan
Username:	edsonray
Password:	******
Email Address:	edson@gmail.com
Contact Number:	093242341

```
Form Submitted
```

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 like Google Chrome and Mozilla Firefox enable users to access and navigate the internet, making it easy to visit websites, watch videos, read news, and perform online searches for research, shopping, social media, and streaming content.

Word processors such as Microsoft Word and Google Docs allow users to create, edit, and format text documents, providing tools for spell-checking, grammar checking, and various formatting options, which are widely used for writing essays, reports, letters, and resumes.

Spreadsheets like Microsoft Excel and Google Sheets help users organize, analyze, and visualize data in tabular form, featuring tools like formulas, charts, and pivot tables, and are commonly used for budgeting, data analysis, and tracking tasks or projects.

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

Home users, students, and office employees use these GUI programs because they are user-friendly and efficient. Web browsers make it easy to access information and connect with others online. Word processors simplify document creation and editing, which is essential for writing assignments and reports. Spreadsheets help organize and analyze data, making tasks like budgeting and project tracking easier. Overall, these tools enhance productivity and simplify everyday tasks.

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?

PyCharm helps developers create GUI applications by providing tools like code editing, debugging, and project management. It simplifies the development process with features like code completion and syntax highlighting.

Without GUI frameworks like Tkinter, developers would have to build interfaces from scratch, dealing with complex low-level details. Frameworks streamline this by offering pre-built components and event handling, making it easier to focus on functionality.

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: Many users and businesses use Windows, making it a popular platform for GUI applications. Developers create programs here to reach a large audience and leverage Windows-specific features.

macOS: This platform is favored by creative professionals and designers. Developers choose macOS to create applications that take advantage of its unique design elements and user experience.

Linux: Often used in servers and by tech enthusiasts, Linux is chosen for its open-source nature. Developers create applications on this platform to cater to users who prefer customizable and secure 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 allows it to handle command-line arguments.

ex = App(): Creates an instance of your main application window, setting up the GUI.

sys.exit(app.exec_()): Starts the event loop, waiting for user actions, and ensures the program exits cleanly when closed.

7. Conclusion:

The laboratory activity provided hands-on experience in developing a simple GUI application using Python's Tkinter framework. Through creating the registration.py and main.py files, participants learned the importance of organizing code and separating functionality for better maintainability.

The activity emphasized practical skills in user interface design, input handling, and event-driven programming. Participants also explored the rationale behind using GUI applications in everyday tasks and gained insight into the various platforms where these applications can be deployed.

Overall, this laboratory experience enhanced understanding of GUI programming concepts and prepared participants for future projects in software development.