Laboratory Activity 5 - Introduction to Event Handling in GUI Development	
Course Code: CPE009B	**Program:** Computer Engineering
Course Title: Object-Oriented Programming	**Date Performed: November 6, 2024**
Section: CPE21S1	**Date Submitted: November 6, 2024**
Name(s): GASPAR, AARON ROWEN O.	**Instructor: Ms. Maria Rizette Sayo**

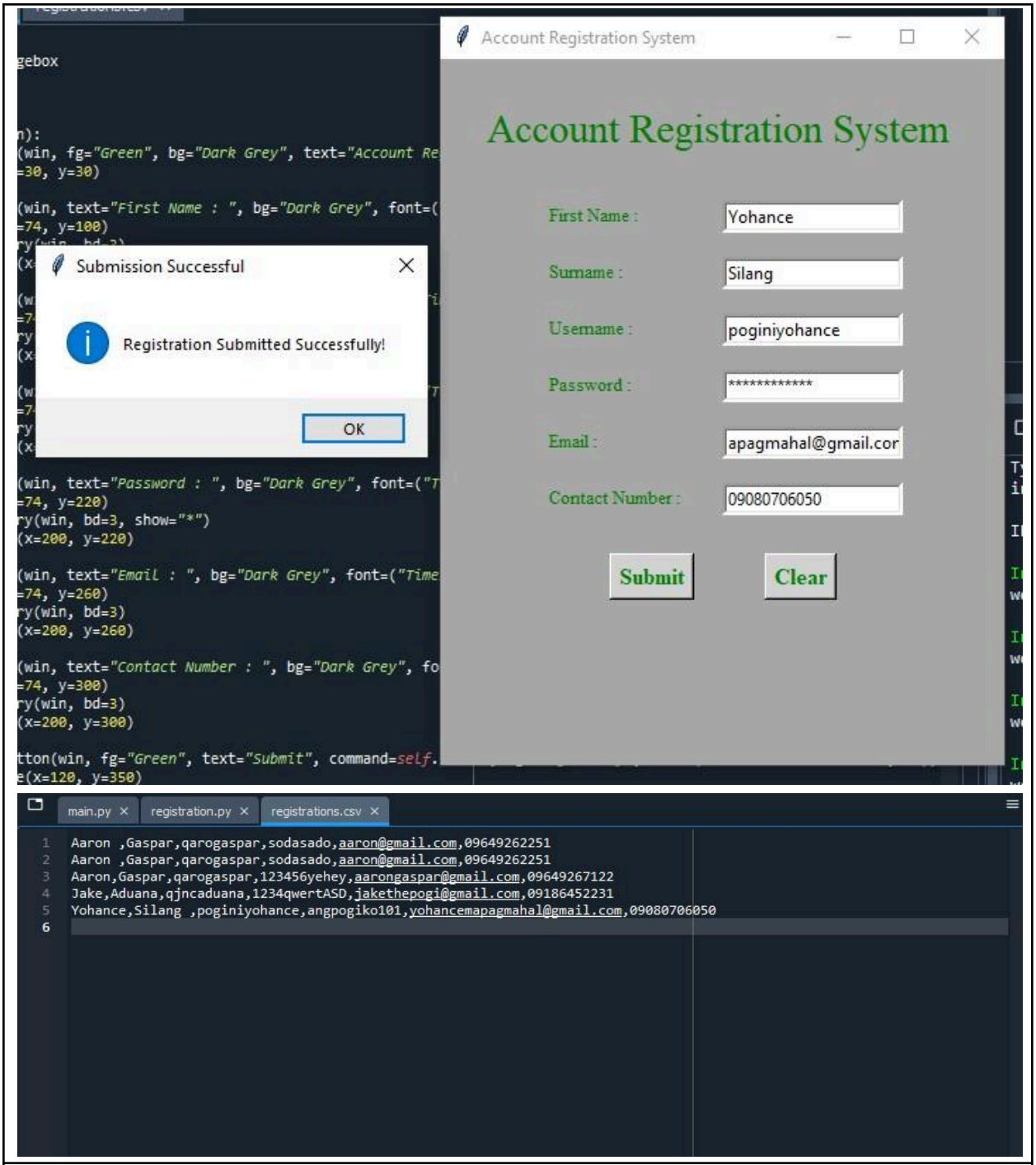
Task:

```
from tkinter import *
from tkinter import messagebox
import csv

class Registration:
    def __init__(self, win):
        self.Lbl1 = Label(win, fg="Green", bg="Dark Grey", text="Account Registration System", font=("Times New Roman", 22))
        self.Lbl1.place(x=30, y=30)

        self.Lbl2 = Label(win, text="First Name : ", bg="Dark Grey", font=("Times New Roman", 10), fg="Green")
        self.Lbl2.place(x=74, y=100)
        self.Entry1 = Entry(win, bd=3)
        self.Entry1.place(x=200, y=100)

        self.Lbl3 = Label(win, text="Surname : ", bg="Dark Grey", font=("Times New Roman", 10), fg="Green")
        self.Lbl3.place(x=74, y=140)
        self.Entry2 = Entry(win, bd=3)
        self.Entry2.place(x=200, y=140)

        self.Lbl4 = Label(win, text="Username : ", bg="Dark Grey", font=("Times New Roman", 10), fg="Green")
        self.Lbl4.place(x=74, y=180)
        self.Entry3 = Entry(win, bd=3)
        self.Entry3.place(x=200, y=180)

        self.Lbl5 = Label(win, text="Password : ", bg="Dark Grey", font=("Times New Roman", 10), fg="Green")
        self.Lbl5.place(x=74, y=220)
        self.Entry4 = Entry(win, bd=3, show="*")
        self.Entry4.place(x=200, y=220)

        self.Lbl6 = Label(win, text="Email : ", bg="Dark Grey", font=("Times New Roman", 10), fg="Green")
        self.Lbl6.place(x=74, y=260)
        self.Entry5 = Entry(win, bd=3)
        self.Entry5.place(x=200, y=260)

        self.Lbl7 = Label(win, text="Contact Number : ", bg="Dark Grey", font=("Times New Roman", 10), fg="Green")
        self.Lbl7.place(x=74, y=300)
        self.Entry6 = Entry(win, bd=3)
        self.Entry6.place(x=200, y=300)

        self.Button1 = Button(win, fg="Green", text="Submit", command=self.submit, bg="Light Grey", font=("Times New Roman bold", 12))
        self.Button1.place(x=120, y=350)
        self.Button2 = Button(win, fg="Green", text="Clear", command=self.clear, bg="Light Grey", font=("Times New Roman bold", 12))
        self.Button2.place(x=230, y=350)
```

```python
        win.config(bg="Dark Grey")

    def submit(self):
        first_name = self.Entry1.get()
        surname = self.Entry2.get()
        username = self.Entry3.get()
        password = self.Entry4.get()
        email = self.Entry5.get()
        contact_number = self.Entry6.get()

        if all([first_name, surname, username, password, email, contact_number]):
            # Save data to a CSV file
            with open('registrations.csv', 'a', newline='') as file:
                writer = csv.writer(file)
                writer.writerow([first_name, surname, username, password, email, contact_number])
            messagebox.showinfo("Submission Successful", "Registration Submitted Successfully!")
        else:
            messagebox.showwarning("Submission Failed", "Please fill in all fields")

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

if __name__ == "__main__":
    window = Tk()
    app = Registration(window)
    window.geometry("400x500+720+250")
    window.title("Account Registration System")
    window.mainloop()
```

main.py × registration.py × registrations.csv ×

```
1  Aaron ,Gaspar,qarogaspar,sodasado,aaron@gmail.com,09649262251
2  Aaron ,Gaspar,qarogaspar,sodasado,aaron@gmail.com,09649262251
3  Aaron,Gaspar,qarogaspar,123456yehey,aarongaspar@gmail.com,09649267122
4  Jake,Aduana,qjncaduana,1234qwertASD,jakethepogi@gmail.com,09186452231
5
```



Questions:

1. **What are the other signals available in PyQt5? (give at least 3 and describe each)**
    - clicked(): This signal is emitted when a button is clicked. It's commonly used to trigger actions in response to user interactions with buttons.
    - textChanged(): This signal is emitted whenever the text in a text input widget (like QLineEdit) changes. It's useful for validating input or updating other parts of the UI in real-time.

- stateChanged(): This signal is emitted when the state of a checkbox changes. It can be used to enable or disable other UI elements based on the checkbox state.

2. **Why do you think that event handling in Python is divided into signals and slots?**
   - Event handling in Python, particularly in frameworks like PyQt, is divided into signals and slots to provide a clear and efficient way to manage communication between objects. Signals are emitted when an event occurs, and slots are functions that respond to these signals. This separation allows for a modular and decoupled design, making it easier to manage and maintain the code.
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?**
   - Message boxes can enhance user experience by providing immediate feedback, guiding users through tasks, and alerting them to errors or important information. They can confirm actions (e.g., “Are you sure you want to delete this file?”), display success messages (e.g., “Registration Successful!”), or warn users about issues (e.g., “Please fill in all fields”). This helps users understand the application’s state and what actions they need to take, making the application more intuitive and user-friendly.
4. **What is Error-handling and how was it applied in the task performed?**
   - Error-handling is the process of anticipating, detecting, and responding to errors in a program. It ensures that the program can handle unexpected situations gracefully without crashing. In the task performed, error-handling was applied by checking if all fields were filled before proceeding with the registration. If any field was empty, a warning message was displayed to the user, preventing incomplete data from being saved.
5. **What are the reasons behind the need to implement error handling? Implementing error handling is crucial for several reasons:**
   - Improved User Experience: It ensures that users are informed about issues and can take corrective actions, leading to a smoother experience.
   - Data Integrity: It prevents data corruption or loss by handling errors gracefully.
   - Security: Proper error handling can prevent security vulnerabilities that might be exploited through unhandled exceptions.
   - Debugging: It makes it easier to identify and fix issues during development by providing clear error messages and logs.

Conclusion:

Learning PyQt is a rewarding journey that opens up a world of possibilities for creating sophisticated and user-friendly desktop applications. PyQt provides a comprehensive set of tools for building complex and feature-rich applications, and its integration with Python allows for rapid development and prototyping. One of the key aspects of PyQt is its event-driven programming model, which uses signals and slots to handle events. This model simplifies event handling and makes the code more modular and maintainable. Additionally, PyQt offers a rich set of widgets that can be customized to create intuitive and visually appealing user interfaces, ranging from basic buttons and text fields to advanced tables and graphics views.