**Lab Exercise 11 – Handling User Input and Form Validation in PySide6/PyQt6**

**Objective:**

In this lab, you will learn how to handle user input and perform form validation using PySide6/PyQt6. You will create a basic form with text fields, radio buttons, and a submit button, implementing validation rules to ensure correct input.

**Step 1: Setup**

**Install Dependencies (If Not Installed)**

Ensure you have PySide6 installed. Run:

pip install PySide6

**Step 2: Create the Main Application Window**

**Key Components:**

* **QLineEdit** for name and email input.
* **QRadioButton** for selecting gender.
* **QPushButton** for form submission.
* **QLabel** for displaying validation messages.

**Code:**

import sys

from PySide6.QtWidgets import (

QApplication, QWidget, QLabel, QLineEdit, QPushButton,

QVBoxLayout, QHBoxLayout, QRadioButton, QMessageBox

)

from PySide6.QtGui import QIntValidator, QRegularExpressionValidator

from PySide6.QtCore import QRegularExpression

class UserForm(QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setWindowTitle("User Input Form - PySide6")

self.setGeometry(300, 200, 400, 250)

layout = QVBoxLayout()

# Name Input

self.name\_label = QLabel("Name:")

self.name\_input = QLineEdit()

self.name\_input.setPlaceholderText("Enter your name")

layout.addWidget(self.name\_label)

layout.addWidget(self.name\_input)

# Email Input with Validation

self.email\_label = QLabel("Email:")

self.email\_input = QLineEdit()

self.email\_input.setPlaceholderText("Enter your email")

email\_regex = QRegularExpression(r"^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$")

self.email\_input.setValidator(QRegularExpressionValidator(email\_regex))

layout.addWidget(self.email\_label)

layout.addWidget(self.email\_input)

# Age Input with Integer Validation

self.age\_label = QLabel("Age:")

self.age\_input = QLineEdit()

self.age\_input.setPlaceholderText("Enter your age (18-99)")

self.age\_input.setValidator(QIntValidator(18, 99)) # Only allow ages 18-99

layout.addWidget(self.age\_label)

layout.addWidget(self.age\_input)

# Gender Selection

self.gender\_label = QLabel("Gender:")

self.male\_radio = QRadioButton("Male")

self.female\_radio = QRadioButton("Female")

gender\_layout = QHBoxLayout()

gender\_layout.addWidget(self.male\_radio)

gender\_layout.addWidget(self.female\_radio)

layout.addWidget(self.gender\_label)

layout.addLayout(gender\_layout)

# Submit Button

self.submit\_button = QPushButton("Submit")

self.submit\_button.clicked.connect(self.validate\_form)

layout.addWidget(self.submit\_button)

self.setLayout(layout)

def validate\_form(self):

""" Validate user input and display a message box if valid """

name = self.name\_input.text().strip()

email = self.email\_input.text().strip()

age = self.age\_input.text().strip()

gender = "Male" if self.male\_radio.isChecked() else "Female" if self.female\_radio.isChecked() else None

if not name:

QMessageBox.warning(self, "Validation Error", "Name cannot be empty!")

return

if not email:

QMessageBox.warning(self, "Validation Error", "Please enter a valid email!")

return

if not age:

QMessageBox.warning(self, "Validation Error", "Age is required!")

return

if not gender:

QMessageBox.warning(self, "Validation Error", "Please select a gender!")

return

QMessageBox.information(self, "Success", f"Form Submitted!\n\nName: {name}\nEmail: {email}\nAge: {age}\nGender: {gender}")

if \_\_name\_\_ == "\_\_main\_\_":

app = QApplication(sys.argv)

window = UserForm()

window.show()

sys.exit(app.exec())

**Step 3: Explanation of Features**

1. **User Input Handling:**
   * QLineEdit fields for entering Name, Email, and Age.
   * QRadioButton for gender selection.
2. **Form Validation:**
   * **Email validation** using QRegularExpressionValidator (ensures a valid email format).
   * **Age validation** using QIntValidator(18, 99) (only allows ages between 18 and 99).
   * **Empty field check** using strip() to prevent blank spaces from being valid input.
   * **Gender selection validation** (ensures the user selects Male or Female).
3. **User Feedback:**
   * Displays validation warnings using QMessageBox.warning().
   * Shows a success message if all fields are valid using QMessageBox.information().

**Step 4: Test the Application**

Run the script and:

* Try submitting the form with missing or incorrect values.
* Ensure validation messages appear correctly.
* If all fields are valid, you should see a success message.