**Lab Exercise 5 – Event Handling and Signal-Slot Mechanism in PySide6**

**Objective**

In this lab, you will learn how to handle user interactions using the **signal-slot mechanism** in PySide6. You will create a GUI application that responds to **button clicks**, **slider changes**, and other events.

**Prerequisites**

* Python installed (Python 3.7+ recommended)
* PySide6 installed (pip install PySide6)

**Concepts Covered**

1. **Signal-Slot Mechanism** – Connecting signals (events) to slots (functions).
2. **Handling Button Clicks** – Performing actions when a button is clicked.
3. **Handling Slider Changes** – Updating UI components dynamically.
4. **Custom Signals and Slots** – Creating and using custom events.

**Task 1: Creating a Basic Window with Buttons and a Slider**

You will build a GUI that:

* Has **two buttons** ("Click Me" and "Reset").
* Displays a **label** that changes when a button is clicked.
* Uses a **slider** to change the text size dynamically.

**Complete Code**

import sys

from PySide6.QtWidgets import QApplication, QWidget, QLabel, QPushButton, QVBoxLayout, QSlider

from PySide6.QtCore import Qt, Signal

class SignalSlotDemo(QWidget):

# Custom signal (emitted when the button is clicked)

custom\_signal = Signal(str)

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

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

self.setWindowTitle("Event Handling & Signal-Slot Mechanism - PySide6")

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

# Layout

self.layout = QVBoxLayout()

# QLabel to display messages

self.label = QLabel("Click a button or move the slider")

self.label.setAlignment(Qt.AlignmentFlag.AlignCenter)

self.label.setStyleSheet("font-size: 18px;")

# QPushButton for event handling

self.button = QPushButton("Click Me")

self.button.clicked.connect(self.on\_button\_click) # Connecting signal to slot

# Reset button

self.reset\_button = QPushButton("Reset")

self.reset\_button.clicked.connect(self.reset\_text)

# QSlider to change text size

self.slider = QSlider(Qt.Orientation.Horizontal)

self.slider.setMinimum(10)

self.slider.setMaximum(50)

self.slider.setValue(18)

self.slider.setTickPosition(QSlider.TickPosition.TicksBelow)

self.slider.valueChanged.connect(self.change\_text\_size) # Connecting slider change event

# Connect custom signal to a slot

self.custom\_signal.connect(self.update\_label)

# Adding widgets to layout

self.layout.addWidget(self.label)

self.layout.addWidget(self.button)

self.layout.addWidget(self.reset\_button)

self.layout.addWidget(self.slider)

self.setLayout(self.layout)

def on\_button\_click(self):

""" Handles button click event """

self.custom\_signal.emit("Button Clicked!") # Emitting custom signal

def update\_label(self, text):

""" Updates the label text when the custom signal is emitted """

self.label.setText(text)

def reset\_text(self):

""" Resets label text """

self.label.setText("Click a button or move the slider")

self.slider.setValue(18) # Reset slider

def change\_text\_size(self):

""" Updates label font size based on slider value """

size = self.slider.value()

self.label.setStyleSheet(f"font-size: {size}px;")

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

app = QApplication(sys.argv)

window = SignalSlotDemo()

window.show()

sys.exit(app.exec())

**How It Works**

1. **Signal-Slot Mechanism**
   * button.clicked.connect(self.on\_button\_click): When clicked, the button emits a signal that triggers the on\_button\_click method.
   * slider.valueChanged.connect(self.change\_text\_size): When the slider value changes, it updates the label font size.
2. **Custom Signals**
   * custom\_signal = Signal(str): A custom signal that takes a string argument.
   * self.custom\_signal.emit("Button Clicked!"): Emits the signal with a message.
   * self.custom\_signal.connect(self.update\_label): Connects the signal to the method that updates the label.