**Lab Exercise 7 – Customizing Widgets for User Interaction and Data Handling in PySide6**

**Objective**

In this lab, you will learn how to customize PySide6 widgets to enhance user interaction and manage data. You will:

* Modify QPushButtons, QLineEdits, and QComboBoxes.
* Use stylesheets to customize widget appearance.
* Handle user input dynamically.

**Prerequisites**

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

**Concepts Covered**

1. **Customizing QPushButton styles and behavior.**
2. **Handling text input using QLineEdit.**
3. **Using QComboBox for selection.**
4. **Displaying data dynamically in QLabel.**
5. **Applying stylesheets (CSS-like styling) to widgets.**

**Task: Build an Interactive Form with Custom Widgets**

**Features**



* A **QLineEdit** for entering a name.
* A **QComboBox** to select a category.



* A **QPushButton** that displays a message when clicked.



* A **QLabel** that dynamically updates based on user input.



**Complete Code**

import sys

from PySide6.QtWidgets import QApplication, QWidget, QLabel, QLineEdit, QPushButton, QVBoxLayout, QComboBox

from PySide6.QtCore import Qt

class CustomWidgetsApp(QWidget):

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

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

self.setWindowTitle("Customizing Widgets - PySide6")

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

# Layout



self.layout = QVBoxLayout()

# QLabel - Display Message



self.label = QLabel("Enter your name:")



self.label.setStyleSheet("font-size: 16px; font-weight: bold;")



# QLineEdit - Text Input



self.text\_input = QLineEdit()

self.text\_input.setPlaceholderText("Type your name here...")



self.text\_input.setStyleSheet("padding: 5px; font-size: 14px;")



self.text\_input.textChanged.connect(self.update\_label)



# QComboBox - Selection Menu

self.combo = QComboBox()



self.combo.addItems(["Select Category", "Student", "Developer", "Designer"])



self.combo.setStyleSheet("padding: 5px; font-size: 14px;")

self.combo.currentIndexChanged.connect(self.update\_label)



# QPushButton - Clickable Button

self.button = QPushButton("Submit")



self.button.setStyleSheet(

"background-color: #4CAF50; color: white; padding: 8px; font-size: 14px; border-radius: 5px;"

)

self.button.clicked.connect(self.display\_message)



# QLabel - Output Message



self.result\_label = QLabel("")

self.result\_label.setStyleSheet("font-size: 16px; color: blue; font-weight: bold;")

# Add widgets to layout

self.layout.addWidget(self.label)



self.layout.addWidget(self.text\_input)



self.layout.addWidget(self.combo)



self.layout.addWidget(self.button)



self.layout.addWidget(self.result\_label)



self.setLayout(self.layout)

def update\_label(self):

""" Updates the label dynamically based on user input """

name = self.text\_input.text()



category = self.combo.currentText()



if name and category != "Select Category":

self.result\_label.setText(f"Hello {name}, you are a {category}!")



else:



self.result\_label.setText("")

def display\_message(self):

""" Displays a message when the button is clicked """

if self.result\_label.text():

self.result\_label.setStyleSheet("color: green; font-weight: bold;")

else:

self.result\_label.setText("Please enter your name and select a category!")

self.result\_label.setStyleSheet("color: red; font-weight: bold;")

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

app = QApplication(sys.argv)

window = CustomWidgetsApp()

window.show()

sys.exit(app.exec())

**How It Works**

1. **QLineEdit** accepts user input and updates dynamically.
2. **QComboBox** lets users select a category.
3. **QPushButton** displays a message when clicked.
4. **QLabel** updates text and styling dynamically.
5. **Stylesheets** (setStyleSheet) modify the appearance of widgets.