**Lab Exercise 15 – Managing UI Interactions Between QML and Backend Python Logic**

**Objective:**

This lab will teach you how to handle UI interactions between QML and Python using PySide6. You will:

* **Call Python functions from QML** (using setContextProperty()).
* **Trigger Python logic from QML events** (using signals and slots).
* **Update QML UI elements dynamically from Python**.

**1. Learning Outcomes**

By the end of this lab, you will:  
✅ Connect QML UI components to Python logic.  
✅ Implement QML button clicks that trigger Python functions.  
✅ Dynamically update QML labels and text fields from Python.  
✅ Use Qt’s **signals and slots** mechanism for communication.

**2. Project Setup & Requirements**

**Project Structure**

qml\_python\_interaction/

│── main.py # Python backend

│── ui.qml # QML UI file

│── requirements.txt # Dependencies (PySide6)

**Install Dependencies**

Ensure **PySide6** is installed:

pip install PySide6

**3. Step 1: Create the QML UI (ui.qml)**

This UI contains:

* A **Label** (displayText) to show messages.
* A **Button** that calls a Python function to update the label.
* A **TextField** where the user enters text, which is sent to Python.

import QtQuick 6.0

import QtQuick.Controls 6.0

ApplicationWindow {

visible: true

width: 400

height: 300

title: "QML-Python Interaction"

Column {

anchors.centerIn: parent

spacing: 15

Label {

id: displayText

text: "Enter text and click the button"

font.pixelSize: 18

}

TextField {

id: inputField

placeholderText: "Type something..."

width: 200

}

Button {

text: "Update Label"

onClicked: backend.updateLabel(inputField.text) // Calls Python function

}

}

}

**4. Step 2: Implement Python Backend (main.py)**

The Python script:

* Loads **QML UI** using QQmlApplicationEngine.
* Exposes a **Python function (updateLabel())** to QML.
* Dynamically updates the **Label text** in QML from Python.

import sys

from PySide6.QtWidgets import QApplication

from PySide6.QtQml import QQmlApplicationEngine

from PySide6.QtCore import QObject, Slot

class Backend(QObject):

"""Handles UI interactions from QML."""

@Slot(str)

def updateLabel(self, text):

"""Updates the QML Label with user input."""

label = engine.rootObjects()[0].findChild(QObject, "displayText")

if label:

label.setProperty("text", f"You entered: {text}")

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

app = QApplication(sys.argv)

engine = QQmlApplicationEngine()

backend = Backend()

engine.rootContext().setContextProperty("backend", backend) # Expose backend to QML

engine.load("ui.qml")

if not engine.rootObjects():

sys.exit(-1)

sys.exit(app.exec())

**5. Running the Application**

Run the Python script:

python main.py

✅ **Type text in the input field → Click "Update Label" → Label updates dynamically!**

**6. Key Concepts in This Exercise**

✅ **setContextProperty("backend", backend)** → Connects Python to QML.  
✅ **QML button click (onClicked) calls Python function (updateLabel()).**  
✅ **Python modifies QML UI dynamically (setProperty()).**  
✅ **Signals & slots mechanism for smooth UI updates.**

**7. Next Steps & Enhancements**

* ✅ Add a **Reset Button** to clear the label text from Python.
* ✅ Implement **real-time updates** using QML signals.
* ✅ Add a **counter** that increments each time the button is clicked.