**Lab Exercise 12 – Concepts of QML and Its Integration with Python Applications**

**Objective:**

Learn the **fundamentals of QML** and how to **integrate it with Python (PySide6)** to build interactive and responsive Qt applications.

**1. Learning Outcomes**

By completing this lab, you will:

✅ Understand **QML basics** (structure, UI components).  
✅ Integrate **QML with Python** using QQmlApplicationEngine.  
✅ Implement **Python-QML communication** using **context properties**.  
✅ Use **QML signals and slots** to handle user interactions.

**2. Project Setup & Requirements**

**Project Structure**

qml\_lab/

│── main.py # Python backend

│── main.qml # QML UI file

│── requirements.txt # Dependencies (PySide6)

**Install Dependencies**

Ensure you have **PySide6** installed:

pip install PySide6

**3. Creating the QML UI (main.qml)**

This **QML file** defines a **simple UI** with:

* A **Text field** (displays messages from Python)
* A **Button** (calls Python function when clicked)
* A **Text input field** (takes user input)

import QtQuick 6.0

import QtQuick.Controls 6.0

ApplicationWindow {

visible: true

width: 400

height: 250

title: "QML-Python Integration"

Column {

anchors.centerIn: parent

spacing: 10

Text {

id: label

text: "Enter your name:"

font.pixelSize: 18

}

TextField {

id: nameInput

width: 200

placeholderText: "Type here..."

}

Button {

text: "Submit"

onClicked: {

backend.processName(nameInput.text) // Call Python function

}

}

Text {

id: responseText

text: ""

font.pixelSize: 16

}

}

}

**4. Loading and Handling QML in Python (main.py)**

This Python script:

* Loads the QML file using QQmlApplicationEngine
* Connects Python **backend logic** to QML UI
* Handles **user input and updates UI** dynamically

import sys

from PySide6.QtWidgets import QApplication

from PySide6.QtQml import QQmlApplicationEngine

from PySide6.QtCore import QObject, Slot

class Backend(QObject):

"""Handles QML interactions and updates UI from Python."""

@Slot(str)

def processName(self, name):

"""Receives user input from QML and updates UI."""

print(f"Received name: {name}") # Print in Python console

# Update QML UI dynamically (responseText field)

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

if responseText:

responseText.setProperty("text", f"Hello, {name}!")

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

app = QApplication(sys.argv)

engine = QQmlApplicationEngine()

backend = Backend()

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

engine.load("main.qml")

if not engine.rootObjects():

sys.exit(-1)

sys.exit(app.exec())

**5. Running the Application**

Run the Python script:

python main.py

✅ **User enters their name** → Python processes it → UI updates dynamically.

**6. Key Features in This Exercise**

✅ **Loading QML UI in Python** (QQmlApplicationEngine)  
✅ **Connecting QML with Python logic** (setContextProperty)  
✅ **Using QML signals to call Python functions** (onClicked → backend.processName(name))  
✅ **Updating QML UI dynamically from Python** (setProperty("text", value))

**7. Next Steps & Enhancements**

* ✅ **Bidirectional communication** (Python updating QML variables directly)
* ✅ **Use signals & slots for real-time updates**
* ✅ **Integrate a QML ListView with Python-provided data**