**Lab Exercise 25- Connection in QML**

In this exercise, we'll create a simple QML application that includes a slider and a rectangle. Moving the slider will change the color of the rectangle.

Create a QML file named main.qml and add the following code:

import QtQuick 2.0

import QtQuick.Controls 2.15

Rectangle {

width: 400

height: 400

Slider {

id: colorSlider

width: 200

anchors.horizontalCenter: parent.horizontalCenter

from: 0

to: 1

}

Rectangle {

id: colorRect

width: 200

height: 200

color: "lightblue"

anchors.horizontalCenter: parent.horizontalCenter

anchors.top: colorSlider.bottom

}

Connections {

target: colorSlider

onValueChanged: {

colorRect.color = Qt.rgba(colorSlider.value, 0.5, 0.5, 1)

}

}

}

In this exercise, we create a QML application with a slider and a rectangle. The Connections element listens to the onValueChanged signal of the slider and updates the color of the rectangle based on the slider's value.

To run the QML file, you can use the following Python script:

import sys

from PyQt5.QtWidgets import QApplication

from PyQt5.QtQuick import QQuickView

from PyQt5.QtCore import QUrl

if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication(sys.argv)

view = QQuickView()

url = QUrl.fromLocalFile('main.qml') # Provide the path to your QML file here

view.setSource(url)

view.show()

sys.exit(app.exec\_())

Make sure to replace 'main.qml' with the actual path to your QML file. When you run the Python script, it will display the QML application. Moving the slider will change the color of the rectangle dynamically. This exercise demonstrates the use of the Connections type in QML to handle signals and dynamically update the appearance of QML components based on user interaction.