**Lab Exercise 33– TDD in QML**

Here is an example of a simple test driver script using QML. In this exercise, we'll create a basic QML application and test it using the QML test driver approach.

**main.qml:**

import QtQuick 2.15

import QtQuick.Controls 2.15

ApplicationWindow {

visible: true

width: 400

height: 400

title: "Test Driver Development in QML"

Rectangle {

id: rect

width: 100

height: 50

color: "lightblue"

border.color: "blue"

border.width: 2

radius: 10

anchors.centerIn: parent

Text {

id: buttonText

text: "Click me"

anchors.centerIn: parent

}

MouseArea {

anchors.fill: parent

onClicked: {

buttonText.text = "Clicked"

}

}

}

}

**test\_driver.py:**

import sys

from PySide6.QtCore import QUrl

from PySide6.QtQuick import QQuickView

from PySide6.QtWidgets import QApplication

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

app = QApplication(sys.argv)

view = QQuickView()

view.setSource(QUrl("main.qml"))

root\_object = view.rootObject()

button\_text = root\_object.findChild(QObject, "buttonText")

assert button\_text.text() == "Click me"

rect = root\_object.findChild(QObject, "rect")

rect.mousePressEvent()

assert button\_text.text() == "Clicked"

sys.exit(app.exec())

In this example, we have a simple QML application that consists of a rectangle with a text element and a mouse area to handle click events. The test\_driver.py script loads the QML file, finds the necessary QML objects, and performs basic tests to check if the button text changes when the rectangle is clicked.

Ensure you have the necessary PySide6 package installed. You can run the Python script using any Python interpreter or by executing it from the command line.