**Lab Exercise 30– Unit Testing in PyQT**

Here's a lab exercise that demonstrates a simple PyQt unit test for a two-number input calculator:

import sys

import unittest

from PyQt5.QtWidgets import QApplication, QMainWindow, QVBoxLayout, QLineEdit, QPushButton, QWidget

class TwoNumberCalculator(QMainWindow):

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

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

self.setWindowTitle("Two Number Calculator")

self.central\_widget = QWidget()

self.setCentralWidget(self.central\_widget)

self.layout = QVBoxLayout()

self.central\_widget.setLayout(self.layout)

self.input\_line\_1 = QLineEdit()

self.input\_line\_2 = QLineEdit()

self.result\_line = QLineEdit()

self.layout.addWidget(self.input\_line\_1)

self.layout.addWidget(self.input\_line\_2)

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

self.add\_button = QPushButton("Add")

self.add\_button.clicked.connect(self.add\_numbers)

self.layout.addWidget(self.add\_button)

def add\_numbers(self):

try:

num1 = float(self.input\_line\_1.text())

num2 = float(self.input\_line\_2.text())

result = num1 + num2

self.result\_line.setText(str(result))

except ValueError:

self.result\_line.setText("Error: Invalid input")

class TestTwoNumberCalculator(unittest.TestCase):

def setUp(self):

self.app = QApplication(sys.argv)

self.window = TwoNumberCalculator()

def test\_addition(self):

self.window.input\_line\_1.setText("2")

self.window.input\_line\_2.setText("3")

self.window.add\_button.click()

self.assertEqual(self.window.result\_line.text(), "5")

def test\_invalid\_input(self):

self.window.input\_line\_1.setText("abc")

self.window.input\_line\_2.setText("def")

self.window.add\_button.click()

self.assertEqual(self.window.result\_line.text(), "Error: Invalid input")

def tearDown(self):

self.window.close()

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

app = QApplication(sys.argv)

window = TwoNumberCalculator()

window.show()

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

You can run this script to see the two-number input calculator interface and the unit tests in action. Adjust the tests and calculator logic according to your specific requirements.