**Lab Exercise 16- Event Handling Techniques in PyQt**

**Lab Exercise: Event Handling Techniques in PyQt**

Creating a lab exercise for event handling techniques in PyQt can help students understand how to work with various types of events in graphical applications. This exercise focuses on handling mouse events, keyboard events, and custom events.

**Objective: Create a PyQt application that demonstrates event handling for mouse events, keyboard events, and custom events.**

**Requirements:**

* PyQt5: You should have PyQt5 installed.
* Python 3.x

**Instructions:**

* Import the required modules:

import sys

from PyQt5.QtWidgets import QApplication, QMainWindow, QLabel, QWidget

from PyQt5.QtCore import Qt, QEvent, pyqtSignal

* Create a custom widget class for event handling:

class EventHandlingWidget(QWidget):

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

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

self.setGeometry(100, 100, 400, 300)

self.setWindowTitle('Event Handling Exercise')

self.label = QLabel("Mouse and Keyboard Events Will Appear Here", self)

self.label.setGeometry(50, 50, 300, 50)

self.label.setAlignment(Qt.AlignCenter)

def mousePressEvent(self, event):

self.label.setText(f"Mouse Pressed at ({event.x()}, {event.y()})")

def keyPressEvent(self, event):

self.label.setText(f"Key Pressed: {event.key()}")

def customEvent(self, event):

if event.type() == MyCustomEvent.MyType:

self.label.setText("Custom Event Received")

* **Create a custom event class:**

class MyCustomEvent(QEvent):

MyType = QEvent.registerEventType()

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

super().\_\_init\_\_(MyCustomEvent.MyType)

* **Create a function to run the application:**

def run\_app():

app = QApplication(sys.argv)

ex = EventHandlingWidget()

ex.show()

# Create and send a custom event

custom\_event = MyCustomEvent()

app.sendEvent(ex, custom\_event)

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

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

run\_app()

* Instruct the students to run the program. The application will display a window with a label. Students can interact with the application in the following ways:
* Click the mouse to see the mouse press event.
* Type a key on the keyboard to see the key press event.
* A custom event is also sent when the application starts.

This lab exercise demonstrates event handling for mouse events, keyboard events, and custom events in a PyQt application. Students can use this as a starting point to explore more advanced event handling and implement specific functionalities based on various events.