**Lab Exercise 3 – Factory Design Pattern in PyQT**

In PyQt, you can implement the Factory design pattern to create objects without specifying the exact class of object that will be created. Here's an example of how to implement a simple factory in PyQt:

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

class ProductA:

def action(self):

return "Action from Product A"

class ProductB:

def action(self):

return "Action from Product B"

class ProductFactory:

@staticmethod

def create\_product(product\_type):

if product\_type == "A":

return ProductA()

elif product\_type == "B":

return ProductB()

else:

raise ValueError("Invalid product type")

class MainWindow(QMainWindow):

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

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

self.setWindowTitle("Factory Design Pattern Example")

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

central\_widget = QWidget()

self.setCentralWidget(central\_widget)

layout = QVBoxLayout()

button\_a = QPushButton("Product A")

button\_b = QPushButton("Product B")

button\_a.clicked.connect(lambda: self.display\_result("A"))

button\_b.clicked.connect(lambda: self.display\_result("B"))

layout.addWidget(button\_a)

layout.addWidget(button\_b)

self.result\_label = QLabel()

layout.addWidget(self.result\_label)

central\_widget.setLayout(layout)

def display\_result(self, product\_type):

product = ProductFactory.create\_product(product\_type)

result = product.action()

self.result\_label.setText(result)

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

app = QApplication([])

window = MainWindow()

window.show()

app.exec\_()

In this example, we have two product classes ProductA and ProductB. The ProductFactory class has a create\_product method that returns the respective product based on the product type provided. The MainWindow class sets up a simple GUI with two buttons that, when clicked, call the display\_result method to create the respective product and display its action on a label.