**Lab Exercise 8 – Basics of Python Threading and QThread in PyQt-PySide**

**Objective**

In this lab, you will learn how to use Python's threading module and Qt’s QThread to perform background tasks in a PyQt/PySide6 application. You will:

* Understand Python's built-in threading module.
* Use QThread to run a task without blocking the UI.
* Implement a **long-running task** with a progress update.

**Prerequisites**

* Python installed (Python 3.7+ recommended).
* PySide6 installed (pip install PySide6).

**Concepts Covered**

1. **Python's threading module** – Running tasks in the background.
2. **QThread in PySide6** – Executing tasks without freezing the GUI.
3. **Signal-slot mechanism** – Updating the UI from a thread.

**Task: Create a UI That Runs a Background Task**

**Features**

* A **QPushButton** to start a long-running task.
* A **QProgressBar** to show progress.
* Uses **QThread** to prevent UI freezing.

**Step 1: Implement Basic Threading with QThread**

import sys

import time

from PySide6.QtWidgets import QApplication, QWidget, QVBoxLayout, QPushButton, QProgressBar

from PySide6.QtCore import QThread, Signal

class WorkerThread(QThread):

""" Background thread that performs a long-running task. """

progress = Signal(int) # Custom signal to send progress updates

def run(self):

""" Runs the background task """

for i in range(1, 101): # Simulate task from 1% to 100%

time.sleep(0.05) # Simulate work (sleep for 50ms)

self.progress.emit(i) # Emit progress value

class ThreadExample(QWidget):

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

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

self.setWindowTitle("QThread Example - PySide6")

self.setGeometry(300, 300, 400, 200)

# Layout

self.layout = QVBoxLayout()

# Progress Bar

self.progress\_bar = QProgressBar()

self.progress\_bar.setValue(0)

# Start Button

self.start\_button = QPushButton("Start Task")

self.start\_button.clicked.connect(self.start\_task)

# Add widgets to layout

self.layout.addWidget(self.progress\_bar)

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

self.setLayout(self.layout)

def start\_task(self):

""" Starts the background thread """

self.thread = WorkerThread()

self.thread.progress.connect(self.progress\_bar.setValue) # Update progress bar

self.thread.start() # Start the thread

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

app = QApplication(sys.argv)

window = ThreadExample()

window.show()

sys.exit(app.exec())

**Step 2: Explanation**

1. **WorkerThread (QThread)**
   * Runs a simulated long task (loop with time.sleep(0.05)).
   * Emits a signal (progress.emit(i)) to update the progress bar.
2. **Main Window (ThreadExample)**
   * Starts the worker thread when the button is clicked.
   * Updates the progress bar dynamically using the progress signal.