**Extra Lab 2 – Loading Multiple Images in Multiple Different Labels from Multiple Different Folders Using Multithreading in PySide6**

**Objective:**

Learn how to load images from different folders asynchronously using QThread in PySide6.

**Prerequisites:**

Ensure you have the required Python libraries installed by running:

pip install PySide6

**Steps to Complete the Lab:**

**1. Implement the PySide6 Application**

Create a Python script (multithreaded\_folder\_image\_loader.py) that loads images from different folders using QThread.

import sys

import time

import os

from PySide6.QtWidgets import QApplication, QWidget, QLabel, QVBoxLayout, QPushButton, QFileDialog

from PySide6.QtGui import QPixmap

from PySide6.QtCore import Qt, QThread, Signal

class ImageLoaderThread(QThread):



image\_loaded = Signal(int, QPixmap) # Signal to send loaded image to UI



def \_\_init\_\_(self, index, image\_path):

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

self.index = index

self.image\_path = image\_path



def run(self):

""" Loads the image with a simulated delay """

time.sleep(2) # Simulating a delay in loading

pixmap = QPixmap(self.image\_path)



self.image\_loaded.emit(self.index, pixmap)

class MultiFolderImageLoaderApp(QWidget):

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

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

self.setWindowTitle("Multithreaded Folder Image Loader - PySide6")

self.setGeometry(200, 200, 800, 600)

# Layout

self.layout = QVBoxLayout()



# QLabel widgets to display images



self.labels = [QLabel(f"Image {i+1} not loaded") for i in range(4)]



for label in self.labels:



label.setAlignment(Qt.AlignmentFlag.AlignCenter)

label.setStyleSheet("border: 1px solid black; padding: 5px;")

self.layout.addWidget(label)



# Select Folder Button



self.select\_button = QPushButton("Select 4 Folders")



self.select\_button.clicked.connect(self.select\_folders)

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

# Load Button

self.load\_button = QPushButton("Load Images")



self.load\_button.clicked.connect(self.load\_images)



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

self.load\_button.setEnabled(False) # Disabled until folders are selected

self.setLayout(self.layout)



self.image\_folders = [] # Store selected folders

self.image\_paths = [] # Store image file paths

self.threads = [] # Store threads to prevent garbage collection

def select\_folders(self):

""" Opens dialog to select 4 folders containing images """

self.image\_folders = []



for i in range(4):

folder = QFileDialog.getExistingDirectory(self, f"Select Folder {i+1}")



if folder:

self.image\_folders.append(folder)



if len(self.image\_folders) == 4:



self.find\_images()



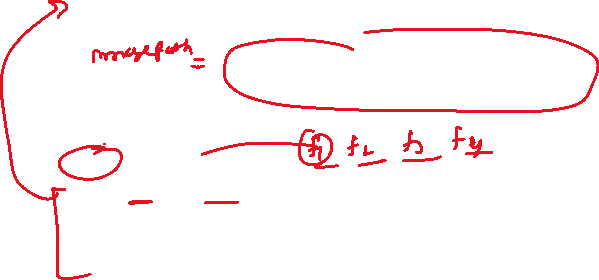
self.load\_button.setEnabled(True) # Enable load button after folder selection



def find\_images(self):

""" Finds the first image file in each selected folder """

self.image\_paths = []



for folder in self.image\_folders:



for file in os.listdir(folder):

if file.lower().endswith((".png", ".jpg", ".jpeg", ".bmp")):



self.image\_paths.append(os.path.join(folder, file))



break

def load\_images(self):



""" Starts separate threads for loading each image """

if len(self.image\_paths) < 4:



return # Ensure we have valid image paths



for i, image\_path in enumerate(self.image\_paths):



thread = ImageLoaderThread(i, image\_path)



thread.image\_loaded.connect(self.display\_image)



self.threads.append(thread)



thread.start()

def display\_image(self, index, pixmap):

""" Updates the corresponding QLabel with the loaded image """



self.labels[index].setPixmap(pixmap)



self.labels[index].setScaledContents(True)



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

app = QApplication(sys.argv)

window = MultiFolderImageLoaderApp()

window.show()

sys.exit(app.exec())

**Explanation:**

1. **Folder Selection:**
   * Users select **4 different folders** that contain images.
   * The application finds the **first image** in each folder.
2. **Multithreaded Image Loading:**
   * QThread is used to load images asynchronously.
   * A signal (image\_loaded) updates the UI when each image is loaded.
3. **Dynamic Display:**
   * 4 QLabel widgets display the loaded images.
   * The **"Load Images"** button starts image loading in parallel.

**Modified Code to Measure Time Taken by Each Thread**

Update the ImageLoaderThread class to calculate execution time.

import sys

import time

import os

from PySide6.QtWidgets import QApplication, QWidget, QLabel, QVBoxLayout, QPushButton, QFileDialog

from PySide6.QtGui import QPixmap

from PySide6.QtCore import Qt, QThread, Signal

class ImageLoaderThread(QThread):

image\_loaded = Signal(int, QPixmap, float) # Add time\_taken to signal

def \_\_init\_\_(self, index, image\_path):

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

self.index = index

self.image\_path = image\_path

def run(self):

""" Loads the image with a simulated delay and measures time taken """

start\_time = time.time() # Record start time

time.sleep(2) # Simulating a delay in loading

pixmap = QPixmap(self.image\_path)

end\_time = time.time() # Record end time

time\_taken = end\_time - start\_time # Compute time taken

print(f"Thread {self.index + 1} took {time\_taken:.2f} seconds to load image")

self.image\_loaded.emit(self.index, pixmap, time\_taken) # Send time\_taken to UI

class MultiFolderImageLoaderApp(QWidget):

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

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

self.setWindowTitle("Multithreaded Folder Image Loader - PySide6")

self.setGeometry(200, 200, 800, 600)

# Layout

self.layout = QVBoxLayout()

# QLabel widgets to display images and time taken

self.labels = [QLabel(f"Image {i+1} not loaded") for i in range(4)]

self.time\_labels = [QLabel("") for \_ in range(4)] # Labels for time taken

for label, time\_label in zip(self.labels, self.time\_labels):

label.setAlignment(Qt.AlignmentFlag.AlignCenter)

label.setStyleSheet("border: 1px solid black; padding: 5px;")

self.layout.addWidget(label)

self.layout.addWidget(time\_label) # Add time label below image

# Select Folder Button

self.select\_button = QPushButton("Select 4 Folders")

self.select\_button.clicked.connect(self.select\_folders)

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

# Load Button

self.load\_button = QPushButton("Load Images")

self.load\_button.clicked.connect(self.load\_images)

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

self.load\_button.setEnabled(False) # Disabled until folders are selected

self.setLayout(self.layout)

self.image\_folders = [] # Store selected folders

self.image\_paths = [] # Store image file paths

self.threads = [] # Store threads to prevent garbage collection

def select\_folders(self):

""" Opens dialog to select 4 folders containing images """

self.image\_folders = []

for i in range(4):

folder = QFileDialog.getExistingDirectory(self, f"Select Folder {i+1}")

if folder:

self.image\_folders.append(folder)

if len(self.image\_folders) == 4:

self.find\_images()

self.load\_button.setEnabled(True) # Enable load button after folder selection

def find\_images(self):

""" Finds the first image file in each selected folder """

self.image\_paths = []

for folder in self.image\_folders:

for file in os.listdir(folder):

if file.lower().endswith((".png", ".jpg", ".jpeg", ".bmp")):

self.image\_paths.append(os.path.join(folder, file))

break

def load\_images(self):

""" Starts separate threads for loading each image """

if len(self.image\_paths) < 4:

return # Ensure we have valid image paths

for i, image\_path in enumerate(self.image\_paths):

thread = ImageLoaderThread(i, image\_path)

thread.image\_loaded.connect(self.display\_image)

self.threads.append(thread)

thread.start()

def display\_image(self, index, pixmap, time\_taken):

""" Updates the corresponding QLabel with the loaded image and time taken """

self.labels[index].setPixmap(pixmap)

self.labels[index].setScaledContents(True)

self.time\_labels[index].setText(f"Loaded in {time\_taken:.2f} seconds")

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

app = QApplication(sys.argv)

window = MultiFolderImageLoaderApp()

window.show()

sys.exit(app.exec())

**How This Works**

1. **Measure time inside the run() method of ImageLoaderThread**:
   * start\_time = time.time() → Capture the start time.
   * end\_time = time.time() → Capture the end time.
   * Compute time taken: time\_taken = end\_time - start\_time.
2. **Print time taken in the console**:
   * print(f"Thread {self.index + 1} took {time\_taken:.2f} seconds to load image").
3. **Display time taken on the GUI**:
   * Each image has an additional QLabel (self.time\_labels[index]) displaying the time taken.

**Expected Output**

* **Console Output (Example)**

Thread 1 took 2.03 seconds to load image

Thread 2 took 2.05 seconds to load image

Thread 3 took 2.02 seconds to load image

Thread 4 took 2.06 seconds to load image

* **GUI Output** Each image will display with a label below it showing the loading time, e.g.:

Loaded in 2.03 seconds