**Lab Exercise 3 – Loading and Displaying Images in a Window using PySide6**

**Objective**

In this lab, you will learn how to load and display images in a PySide6 window using QLabel and QPixmap.

**Prerequisites**

* Python 3.7+
* PySide6 installed (pip install PySide6)
* An image file (e.g., image.jpg) in the project directory

**Part 1: Displaying an Image using QLabel and QPixmap**

**Step 1: Create a Python File**

1. Create a file display\_image.py.
2. Write the following code:

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

from PySide6.QtGui import QPixmap

import sys

class ImageWindow(QWidget):

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

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

self.setWindowTitle("Image Viewer")

self.resize(500, 400)

layout = QVBoxLayout()

# Create QLabel to hold the image

self.image\_label = QLabel(self)

pixmap = QPixmap("image.jpg") # Load the image file

self.image\_label.setPixmap(pixmap)

self.image\_label.setScaledContents(True) # Scale the image to fit the label

layout.addWidget(self.image\_label)

self.setLayout(layout)

app = QApplication(sys.argv)

window = ImageWindow()

window.show()

sys.exit(app.exec())

**Explanation**

* QLabel is used to display the image.
* QPixmap("image.jpg") loads the image.
* setPixmap(pixmap) sets the image to the label.
* setScaledContents(True) scales the image to fit the label size.
* QVBoxLayout() is used to organize widgets vertically.

**Part 2: Select and Load an Image using QFileDialog**

**Step 1: Modify the Code to Allow Image Selection**

1. Modify display\_image.py to allow the user to select an image file.

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

from PySide6.QtGui import QPixmap

import sys

class ImageWindow(QWidget):

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

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

self.setWindowTitle("Image Viewer")

self.resize(600, 500)

layout = QVBoxLayout()

self.image\_label = QLabel(self)

self.image\_label.setPixmap(QPixmap()) # Empty placeholder

self.image\_label.setScaledContents(True)

# Button to open file dialog

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

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

layout.addWidget(self.image\_label)

layout.addWidget(self.load\_button)

self.setLayout(layout)

def load\_image(self):

file\_path, \_ = QFileDialog.getOpenFileName(self, "Open Image", "", "Images (\*.png \*.jpg \*.jpeg \*.bmp)")

if file\_path:

pixmap = QPixmap(file\_path)

self.image\_label.setPixmap(pixmap)

app = QApplication(sys.argv)

window = ImageWindow()

window.show()

sys.exit(app.exec())

**Explanation**

* QFileDialog.getOpenFileName() allows the user to select an image.
* load\_image() updates QLabel with the selected image.
* The QPushButton triggers the image selection.

**Lab Tasks**

1. Modify the UI to use a **grid layout** instead of a vertical layout.
2. Add a **zoom-in and zoom-out button** to adjust the image size.
3. Display the image **filename** in the window title.

**Updated Code: Image Viewer with Grid Layout and Zoom Controls**

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

from PySide6.QtGui import QPixmap

from PySide6.QtCore import Qt

import sys

class ImageWindow(QWidget):

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

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

self.setWindowTitle("Image Viewer")

self.resize(600, 500)

self.layout = QGridLayout()

self.setLayout(self.layout)

# QLabel to display the image

self.image\_label = QLabel(self)

self.image\_label.setPixmap(QPixmap()) # Placeholder

self.image\_label.setScaledContents(True)

# Buttons

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

self.zoom\_in\_button = QPushButton("Zoom In")

self.zoom\_out\_button = QPushButton("Zoom Out")

self.load\_button.setFixedSize(100, 50)  
 self.zoom\_in\_button.setFixedSize(100, 50)  
 self.zoom\_out\_button.setFixedSize(100, 50)

# Connect buttons to actions

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

self.zoom\_in\_button.clicked.connect(self.zoom\_in)

self.zoom\_out\_button.clicked.connect(self.zoom\_out)

# Add widgets to grid layout

self.layout.addWidget(self.image\_label, 0, 0, 1, 3) # Image spans 3 columns

self.layout.addWidget(self.load\_button, 1, 0)

self.layout.addWidget(self.zoom\_in\_button, 1, 1)

self.layout.addWidget(self.zoom\_out\_button, 1, 2)

self.current\_pixmap = None # Store the loaded image

self.zoom\_factor = 1.0 # Track zoom level

def load\_image(self):

file\_path, \_ = QFileDialog.getOpenFileName(self, "Open Image", "", "Images (\*.png \*.jpg \*.jpeg \*.bmp)")

if file\_path:

self.current\_pixmap = QPixmap(file\_path)

self.zoom\_factor = 1.0 # Reset zoom

self.update\_image\_display()

def zoom\_in(self):

if self.current\_pixmap:

self.zoom\_factor \*= 1.2 # Increase size by 20%

self.update\_image\_display()

def zoom\_out(self):

if self.current\_pixmap:

self.zoom\_factor \*= 0.8 # Decrease size by 20%

self.update\_image\_display()

def update\_image\_display(self):

if self.current\_pixmap:

scaled\_pixmap = self.current\_pixmap.scaled(

self.current\_pixmap.size() \* self.zoom\_factor,

Qt.AspectRatioMode.KeepAspectRatio

)

self.image\_label.setPixmap(scaled\_pixmap)

app = QApplication(sys.argv)

window = ImageWindow()

window.show()

sys.exit(app.exec())