**Extra Lab 1 – Creating a Live Updating Line Chart from an Excel Table in PySide6**

**Objective:**

Learn how to continuously monitor an Excel file for updates and dynamically refresh a line chart in a PySide6 GUI.

**Prerequisites:**

Ensure you have the required Python libraries installed by running:

pip install PySide6 pandas matplotlib openpyxl

**Steps to Complete the Lab:**

**1. Prepare an Excel File (data.xlsx)**

Create an Excel file (data.xlsx) with the following structure:

|  |  |
| --- | --- |
| **Date** | **Sales ($)** |
| 01-03-2025 | 100 |
| 02-03-2025 | 150 |
| 03-03-2025 | 200 |
| 04-03-2025 | 250 |
| 05-03-2025 | 300 |
| 06-03-2025 | 350 |
| 07-03-2025 | 400 |
| 08-03-2025 | 450 |
| 09-03-2025 | 500 |
| 10-03-2025 | 550 |
| 11-03-2025 | 600 |
| 12-03-2025 | 650 |
| 13-03-2025 | 700 |
| 14-03-2025 | 750 |
| 15-03-2025 | 800 |
| 16-03-2025 | 850 |
| 17-03-2025 | 900 |
| 18-03-2025 | 950 |
| 19-03-2025 | 1000 |

You can manually update this file while the GUI is running to see the live updates.

**2. Implement the PySide6 Application**

Create a Python script (live\_chart\_gui.py) that monitors the Excel file and updates the chart in real time.

import sys

import pandas as pd

import matplotlib.pyplot as plt

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

from PySide6.QtGui import QPixmap

from PySide6.QtCore import Qt, QTimer

from matplotlib.backends.backend\_agg import FigureCanvasAgg as FigureCanvas

import os

class LiveChartApp(QWidget):

def \_\_init\_\_(self, excel\_file):

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

self.excel\_file = excel\_file

self.setWindowTitle("Live Excel Line Chart Viewer")

self.setGeometry(200, 200, 600, 500)

# Layout setup

self.layout = QVBoxLayout()

# QLabel to display chart

self.label = QLabel("Chart will appear here")

self.label.setAlignment(Qt.AlignmentFlag.AlignCenter)

self.layout.addWidget(self.label)

self.setLayout(self.layout)

# Timer to refresh chart every 5 seconds

self.timer = QTimer()

self.timer.timeout.connect(self.update\_chart)

self.timer.start(5000) # 5 seconds interval

self.update\_chart() # Initial chart display

def update\_chart(self):

""" Reads Excel data and updates the chart """

if not os.path.exists(self.excel\_file):

self.label.setText("Excel file not found!")

return

try:

# Read Excel data

df = pd.read\_excel(self.excel\_file)

# Check for required columns

if "Date" not in df.columns or "Sales ($)" not in df.columns:

self.label.setText("Invalid Excel format! Ensure columns: Date, Sales ($)")

return

# Convert date column to datetime format

df["Date"] = pd.to\_datetime(df["Date"])

# Generate line chart

fig, ax = plt.subplots()

ax.plot(df["Date"], df["Sales ($)"], marker="o", linestyle="-", color="b", label="Sales Trend")

ax.set\_xlabel("Date")

ax.set\_ylabel("Sales ($)")

ax.set\_title("Live Sales Over Time")

ax.legend()

ax.grid()

# Save figure as image

chart\_path = "live\_chart.png"

fig.savefig(chart\_path)

plt.close(fig)

# Display the chart in QLabel

self.display\_chart(chart\_path)

except Exception as e:

self.label.setText(f"Error reading Excel: {e}")

def display\_chart(self, chart\_path):

pixmap = QPixmap(chart\_path)

self.label.setPixmap(pixmap)

self.label.setScaledContents(True)

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

excel\_file = "data.xlsx" # Change this if using a different file

app = QApplication(sys.argv)

window = LiveChartApp(excel\_file)

window.show()

sys.exit(app.exec())

**Explanation:**

1. **Read Excel File:**
   * Uses pandas to load data from the Excel file.
   * Checks for valid column names (Date and Sales ($)).
2. **Generate Live Line Chart:**
   * Uses matplotlib to plot a **live-updating** line chart.
   * Saves the chart as an image (live\_chart.png).
3. **Live Updates using QTimer:**
   * A QTimer runs every **5 seconds** to check for updates.
   * The chart updates automatically when new data is added to data.xlsx.

**Expected Outcome:**

* The application starts by displaying the **initial** line chart.
* If you **add new data** to data.xlsx and save it, the **chart updates every 5 seconds**.
* If the file is missing or has invalid data, an error message is displayed.