

Eric Gathinji

Programming in C# CST-150-0500

Grand Canyon University

10th July 2025

Activity 3

Github link: <https://github.com/Ericgathinji444/GCU>

Video link: https://youtu.be/ak2J3leGTWs?si=WjaA9PROq0_dWo-N

Updated flowchart

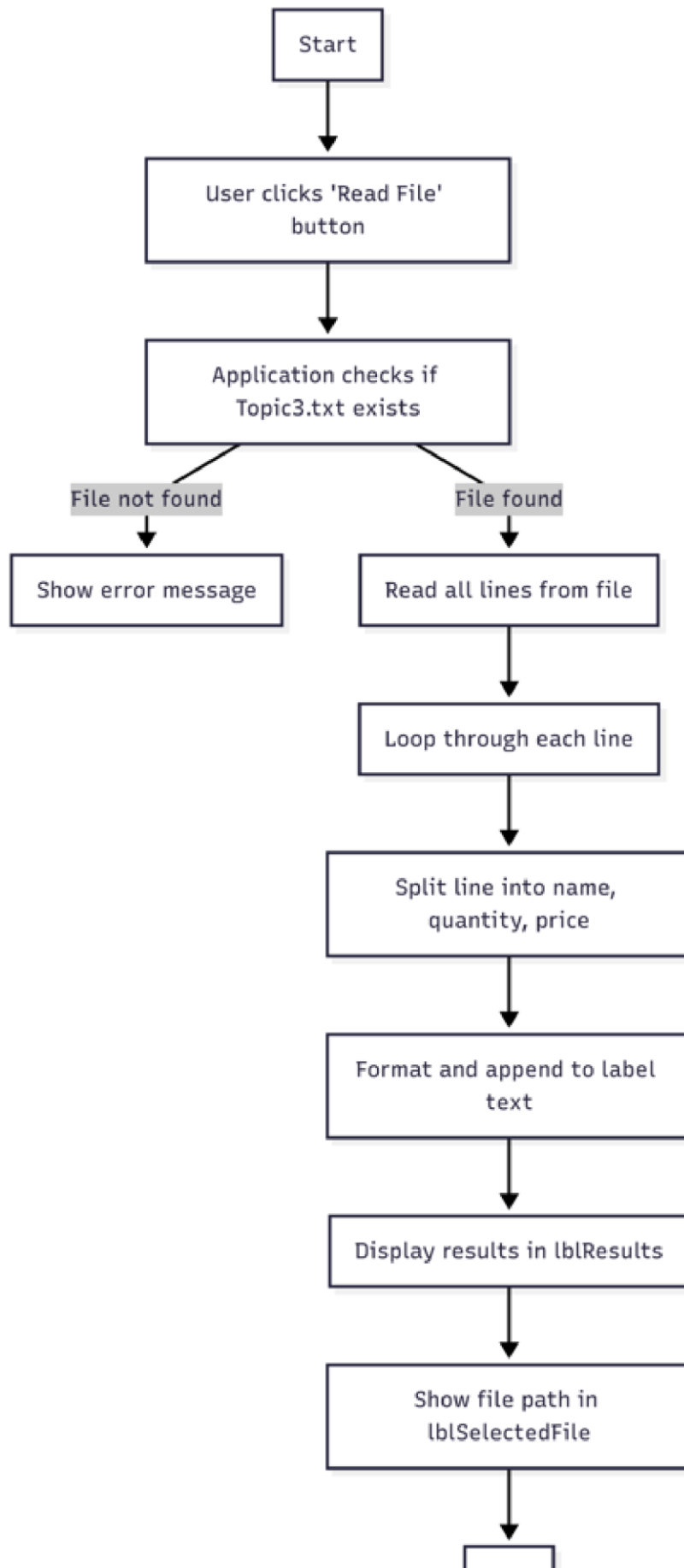


Figure 1: The updated flowchart flow of the application. The process starts when the user clicks the button and ends when the results are displayed.

Figure 1: Explanation

This flowchart visually represents the logical sequence followed by the Windows Forms app. It begins with a button click event, proceeds through a file existence check, reads data, and updates the UI components.

It models the interaction between user input, file handling, looping, and label updates.

Screenshot before Form is populated:

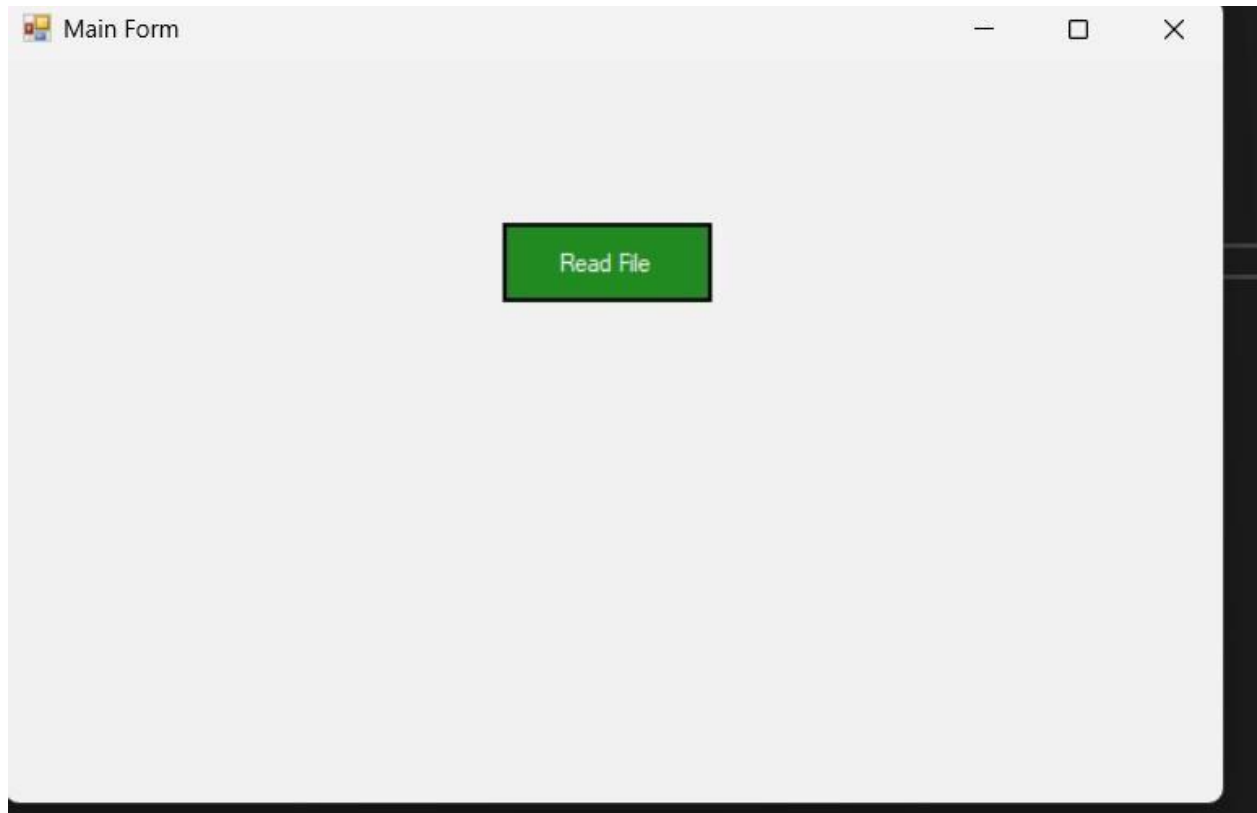


Figure 2:Main form before any interaction.

Explanation:

The form includes a "Read File" button styled in green, a label at the bottom for the selected file path, and a label at the center to show the formatted file data. At this stage, both labels are hidden.

Screenshot after form is populated:

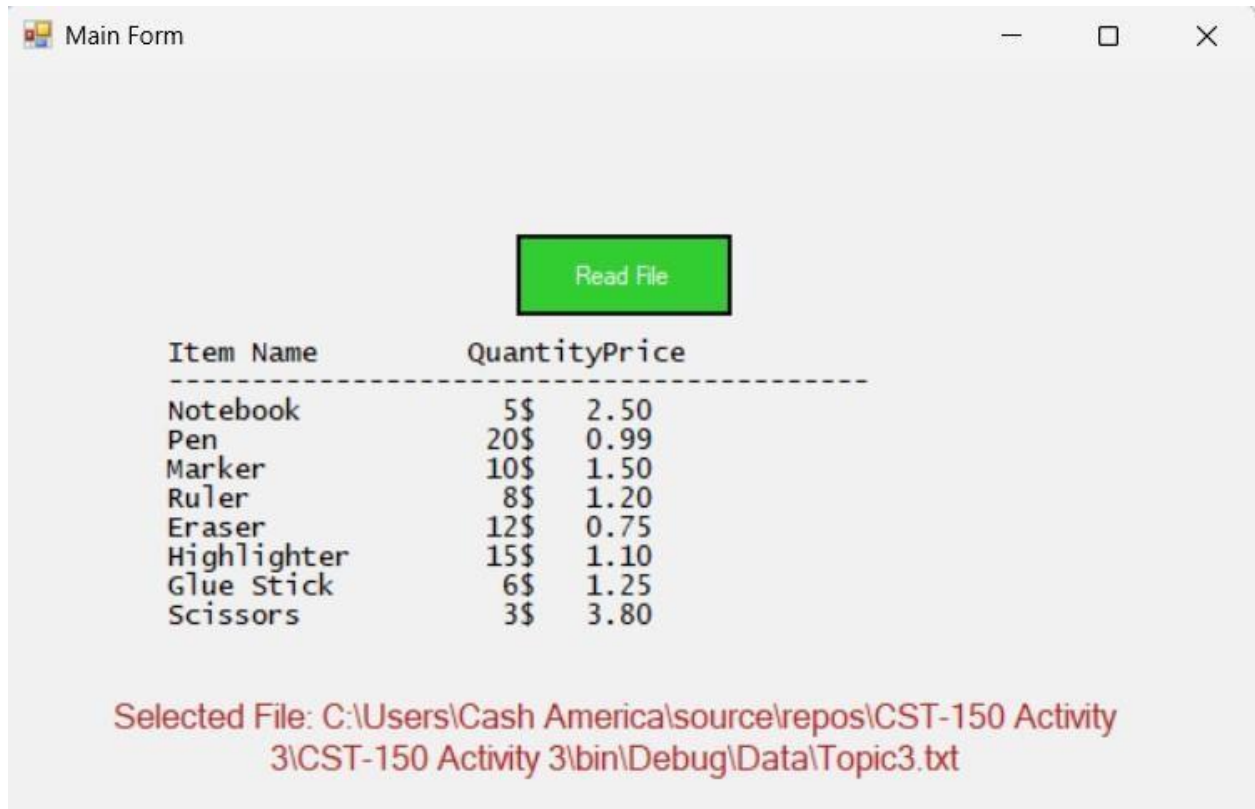


Figure 3: Main form after reading Topic3.txt.

Explanation:

This screenshot shows the label displaying the formatted results from the text file using a monospaced font. The lower label shows the path of the selected file. The app has read each line and split it into text and numeric components.

Screenshot(s) of the code behind

```
using System;
using System.IO;
using System.Windows.Forms;

namespace CST_150_Activity_3
{
    3 references
    public partial class FrmMain : Form
    {
        1 reference
        public FrmMain()
        {
            InitializeComponent();
            lblResults.Visible = false;
            lblSelectedFile.Visible = false;
        }

        /// <summary>
        /// Event handler for the "Read File" button click.
        /// Reads Topic3.txt from Data folder and displays results.
        /// </summary>
        1 reference
        private void BtnReadFileClickEvent(object sender, EventArgs e)
        {
            try
            {
                // Get path to Topic3.txt in the /Data folder
                string filePath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "Data", "Topic3.txt");

                if (!File.Exists(filePath))
                {
                    MessageBox.Show("The file Topic3.txt was not found.");
                    return;
                }

                // Get path to Topic3.txt in the /Data folder
                string filePath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "Data", "Topic3.txt");

                if (!File.Exists(filePath))
                {
                    MessageBox.Show("The file Topic3.txt was not found.");
                    return;
                }

                lblSelectedFile.Text = $"Selected File: {filePath}";
                lblSelectedFile.Visible = true;

                string[] lines = File.ReadAllLines(filePath);
                string results = "";

                // Add headers
                results += "Item Name           Quantity\tPrice\n";
                results += "-----\n";

                foreach (string line in lines)
                {
                    string[] parts = line.Split('\t');
                    if (parts.Length > 3)
                    {
                        string name = parts[0];
                        int quantity = int.Parse(parts[1]);
                        decimal price = decimal.Parse(parts[2]);

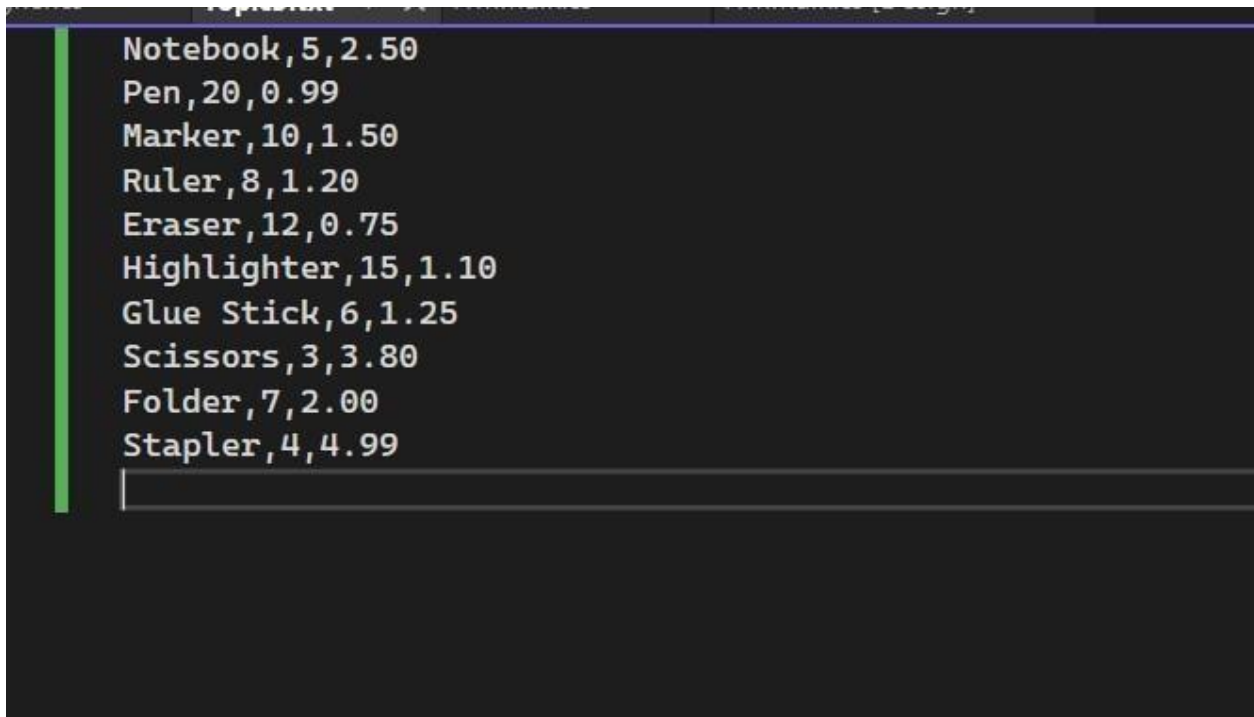
                        results += string.Format("{0,-16}\t{1,5}\t{2,7:F2}\n", name, quantity, price);
                    }
                }
            }
            catch { }
        }
    }
}
```

Figure 4:code-behind that handles file processing.

Figure 4:Explanation:

This code defines the button click event and processes the Topic3.txt file. It demonstrates error handling, file reading, and string formatting using .NET C#. Comments help identify each logical section of the method.

Screenshot of original text file



```
Notebook,5,2.50
Pen,20,0.99
Marker,10,1.50
Ruler,8,1.20
Eraser,12,0.75
Highlighter,15,1.10
Glue Stick,6,1.25
Scissors,3,3.80
Folder,7,2.00
Stapler,4,4.99
```

Figure 5:Contents of Topic3.txt file used for testing

Explanation:

This file contains ten lines of comma-separated values, each line representing an item in inventory. The app reads and displays these using a loop.

PART 2. Flowchart Section

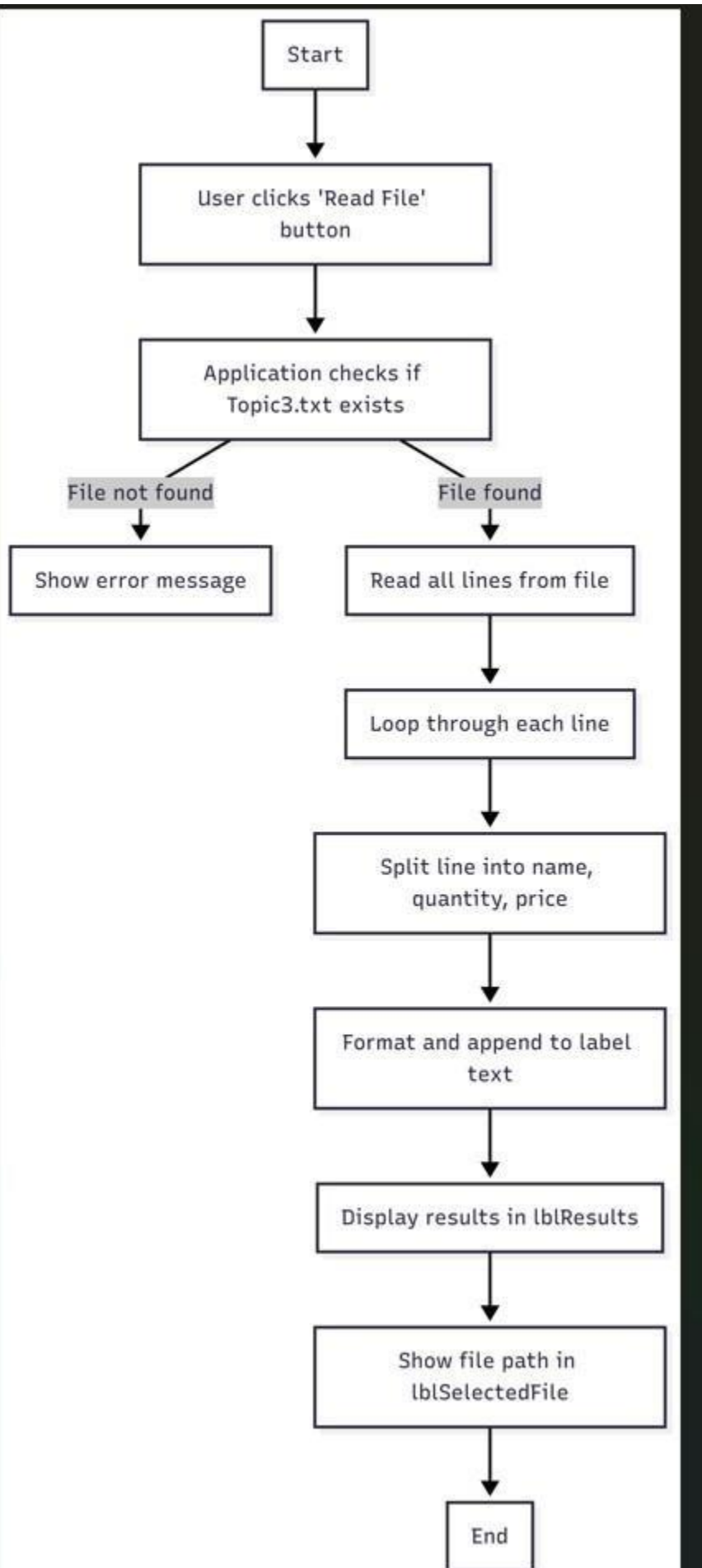
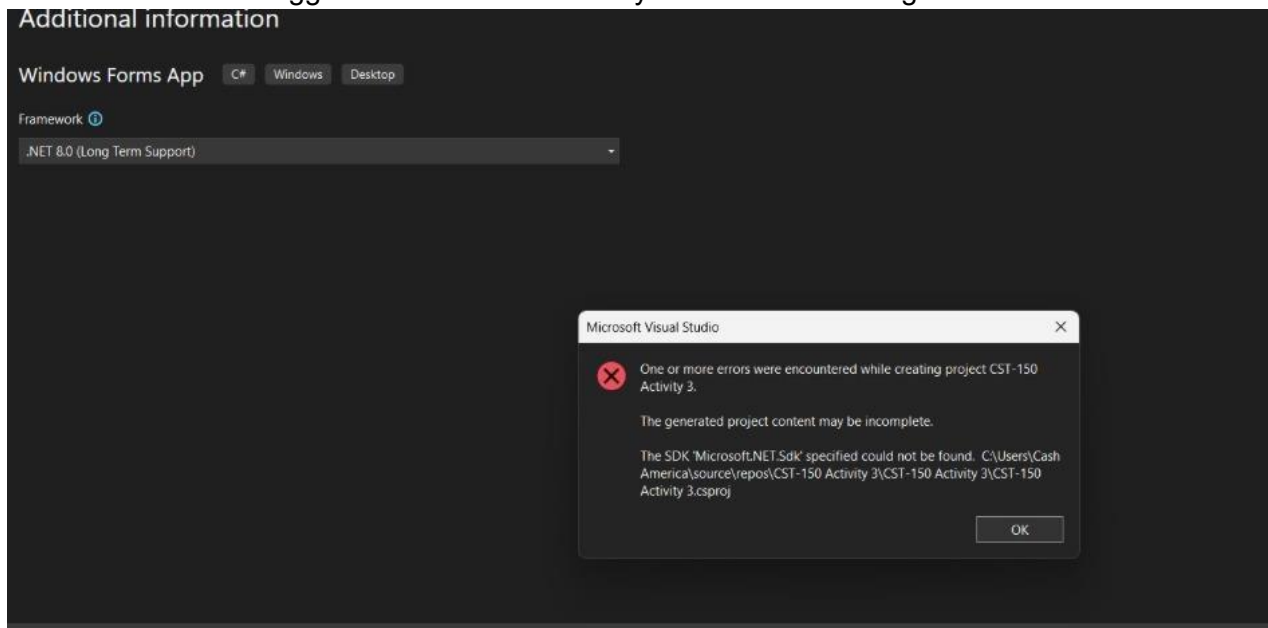


Figure 5: Final flowchart representation used to implement logic in code.

Explanation: This diagram helped structure the logic before implementation. Each decision (such as checking a file existence) and action (such as reading and formatting data) was planned using basic flowchart symbol.

ADD ONS

During the implementation of Activity 3, I encountered persistent and technical issues while attempting to use the Windows Forms App template (.NET 7.0 or later) in Visual Studio, as instructed. Despite following the correct setup process—including installing the recommended SDKs, verifying environment variables, and modifying Visual Studio workloads—I was repeatedly met with the following error: "The SDK 'Microsoft.NET.Sdk' specified could not be found." Additionally, I attempted the following troubleshooting steps: Installed and repaired .NET 7.0, 8.0, and 9.0 SDKs Verified the presence of the ".NET Desktop Development" workload Cleared NuGet and dotnet SDK caches Tried creating the project in a new directory with no spaces Used the Visual Studio Installer to update all components Unfortunately, none of these efforts resolved the issue. As a result, and to ensure timely submission of the assignment, I implemented the functionality using an alternative template that allowed for proper compilation. The user logic and interface closely mirror the expected outcome and the code fulfills the requirements of the assignment. I am willing to retry the assignment using Windows Form once my system supports it and I welcome new suggestions in future. Thank you for understanding.



Naming Conventions

The code is readable since I used meaningful names using PascalCase for methods and properties and CamelCase for variables.

Properties:

Properties follow PascalCase, for example, ItemName, price, and Stocklevel, which aligns with standard C# practices.

Code Structure

I structured the code into clearly defined sections.

Consistency

The formatting is consistent throughout my project.

OS-Windows 11

RAM: 8GB

Processor Intel Core i5

Tutor Discovery

I reached out to my instructor Mark Smithers after being informed my work was unsatisfactory. This comment helped me keep the flowcharts and everything else as required in the instructions.

I watched the video instructions too to ensure my work follows what is required.

Weekly activity:

Start Monday: June 30, 2025: 8:00 am End: 12:00 pm Milestone 3

Start Sunday: July 1st, 2025: 8:00 am End: 12:00 pm Milestone 3

Start Wednesday: July 9, 2025: 8:00 am End: 2:00 pm Activity 3

Start Thursday: July 10th, 2025: 4:00 am End 8:00 am Milestone 3

Follow-up questions

What was challenging?

Troubleshooting SDK issues in VS and trying to ensure compatibility with the Windows Forms App template.

What did you learn?

How to handle exceptions during file operations and efficiently parse and read data from external files.

How would you improve the project?

I would incorporate more thorough error handling to help users in the event of problems and improve the user interface for better engagement.

How can you use what you learned on the job?

I can use my knowledge of exception handling and file I/O to create reliable programs that improve user experience.

