CST-150 Activity 7 Guide

Contents

Part 1

Working with Multiple Forms

Overview

This activity focuses on using multiple forms. The activity also includes a couple of items that will assist with the final milestone application with deleting inventory and searching inventory. The search results will be shown on a secondary form.

Execution

Execute this assignment according to the following guidelines:

1. Activity 7 is a continuation of Activity 6 Part 1.
    a. Open Activity 6 in Visual Studio.
    b. Open Visual Studio and select "Open a project or solution" as shown in Figure 1.
    c. Select "CST-150 ListTogv" and open the project by selecting the solution file "CST-150 ListTogv.sln." The .sln is the solution file.

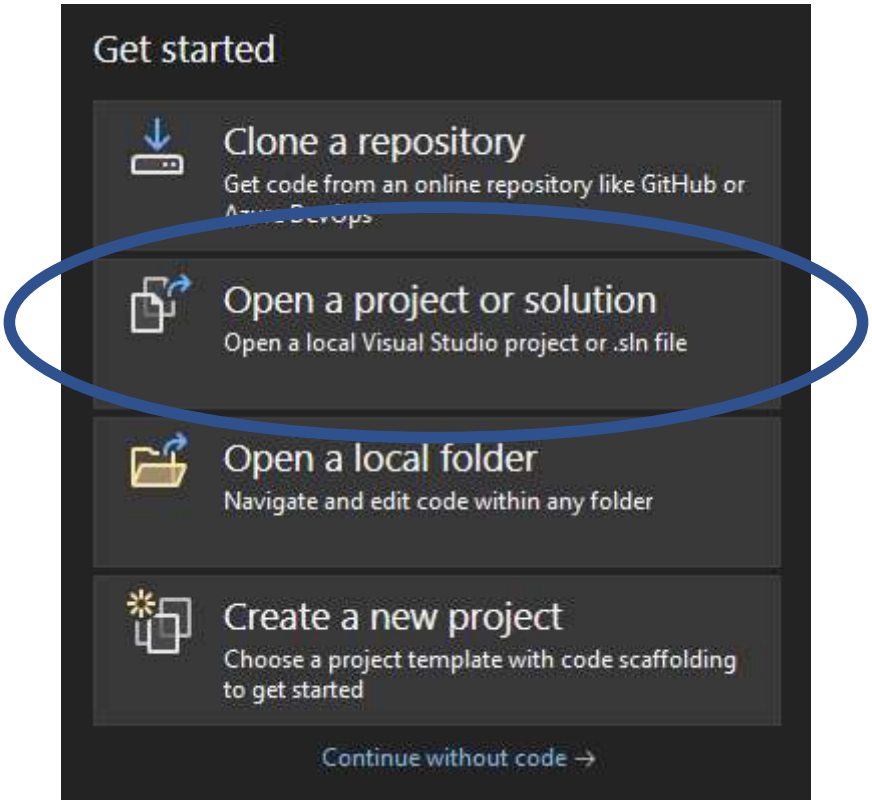

Figure 1: Open an existing solution.

2. Delete row in Data Grid View.

a. In FrmInventory, design screen.

b. Add a new button to the form as show in Figure 2.

c. Name it "btnDelete."

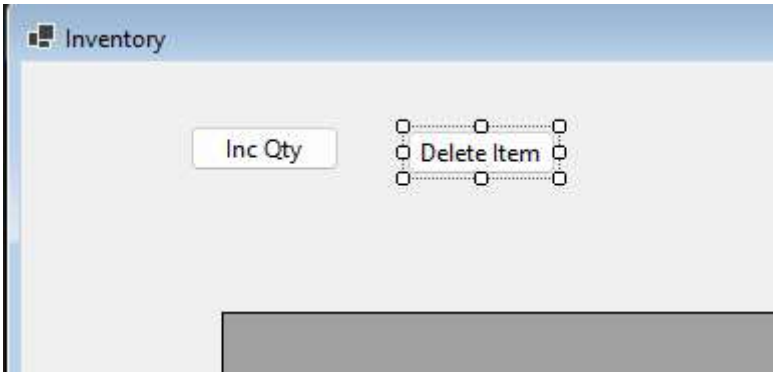

Figure 2: Add Delete Button to Form

d. With the button selected create a click event handler named "BtnDeleteItem_EventHandler" as shown in Figure 3.

```
/// <summary>
/// Delete an item from inventory
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 reference
private void BtnDeleteItem_EventHandler(object sender, EventArgs e)
{

}
```

Figure 3: Button Click Event

e. Remove an item from the List and DataGridView as shown in Figure 4.

f. Follow the comments in the code to fully understand the process.

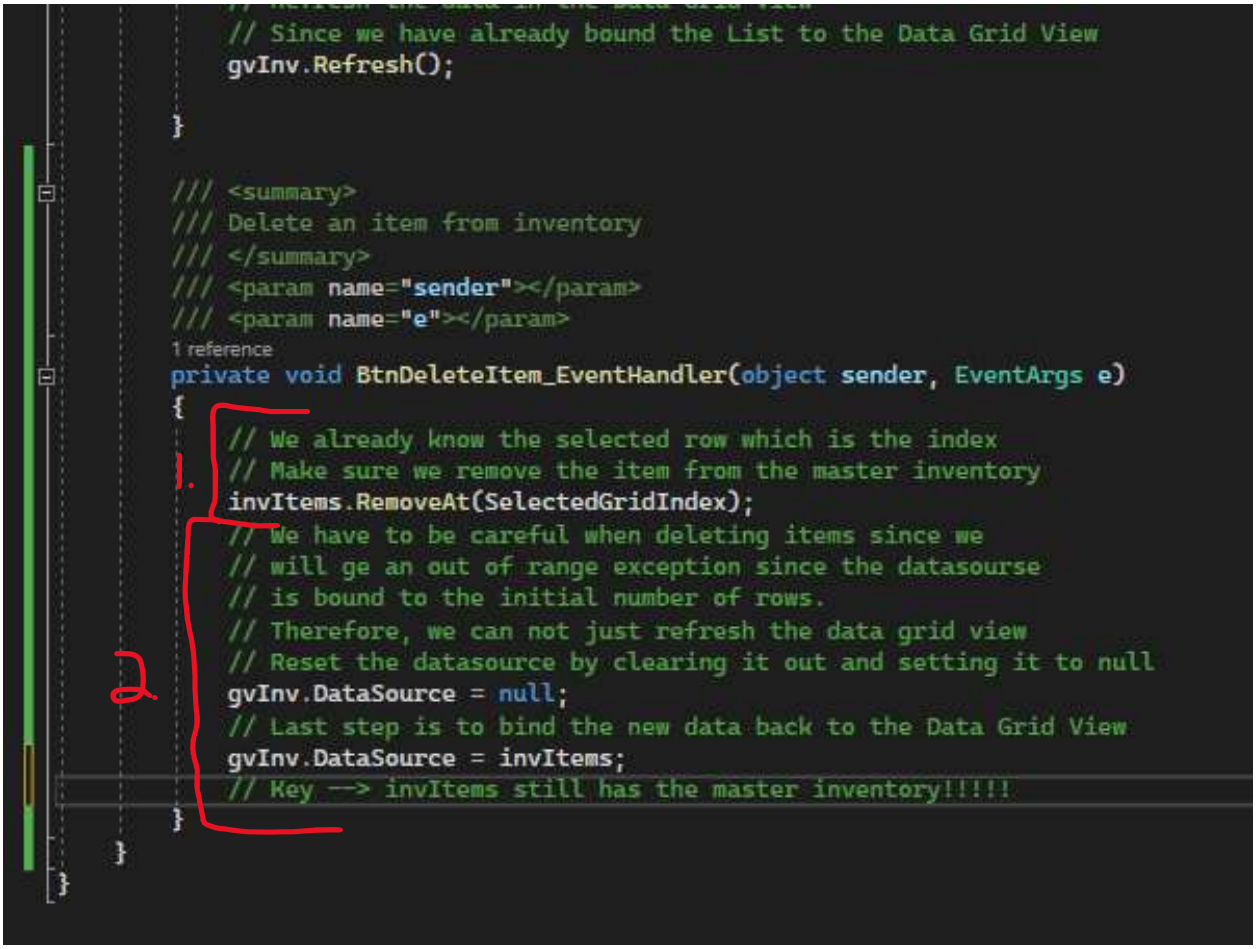

Figure 4: Remove item.

3. Searching for Inventory Items and Display on Secondary Form
    a. Keep in mind that the examples shown in these activities are basic examples of how to perform operations/functions in the milestones. Each student is expected to learn from these activities and expand on the activities for each milestone. At no time is it acceptable to use the activity code exactly as provided in the activities.
    b. In FrmInventory, design screen.
    c. Add a new label to the form as show in Figure 5 and change the font size property to 12pts.

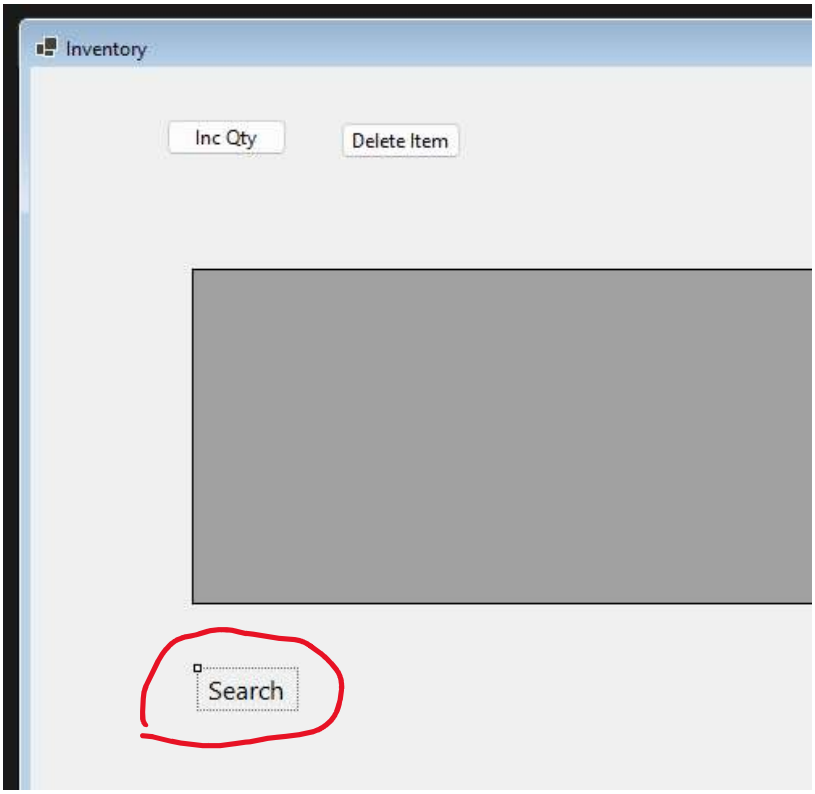

Figure 5: Search Label

d. Add a Text Box just to the right of the label as shown in Figure 6.

e. Name this text box "txtSearch" with font size of 12 pt.

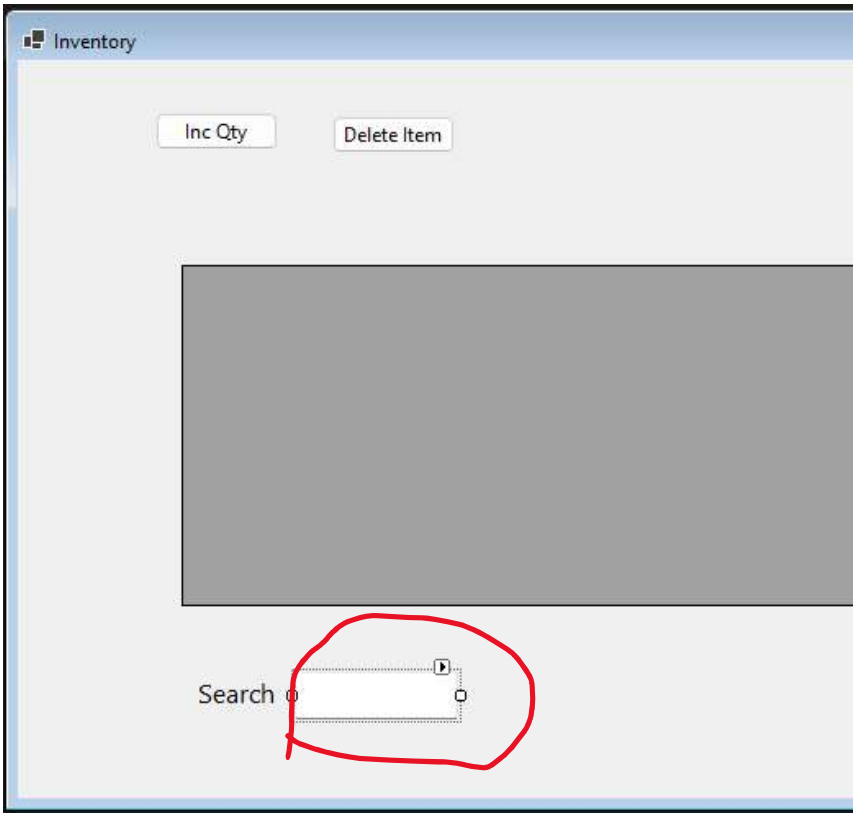

Figure 6: Search Text Box

f. Add a button just to the right of the text box that will be used to invoke the search feature as shown in Figure 7.

g. Name this button "btnSearch" with font size of 12 pt. and text property changed to "Go." Resize the button so it fits the text appropriately.

h. Create the event handler "BtnSearch_ClickEvent" as shown in Figure 8.

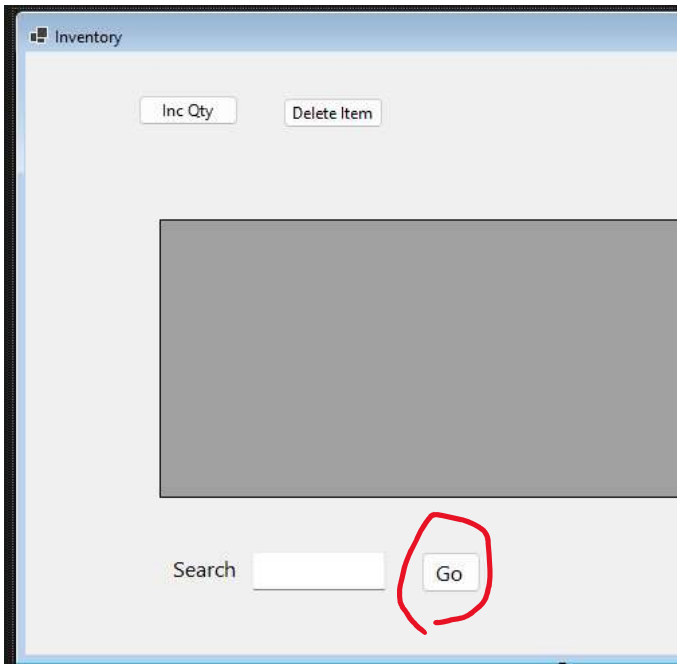

Figure 7: Go Button

```
    /// <summary>
    /// Search Event Handler
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    1 reference
    private void BtnSearch_ClickEvent(object sender, EventArgs e)
    {

    }
}
```

Figure 8: Button Click Event Handler

i. The first step is to create a list we can put the search results into as shown in Figure 9.

j. This new list can then be sent to the secondary form leaving the master inventory list untouched.

```
3 references
public partial class FrmInventory : Form
{
    // Create the class level object
    // This is called an inventory reference variable
    // This is our master inventory object that MUST
    // always contain the most update-to-date inventory
    List<InvItem> invItems = new List<InvItem>();
    // List that will hold all items found for search
    List<InvItem> invSearch = new List<InvItem>();

    // Properties
    3 references
    private int SelectedGridIndex { get; set; }
```

Figure 9: Search List

k. Since searching is logic, we need to place this code in the Business Layer. Go to the Inventory.cs file.

l. Create a new method that returns a List of type InvItem.

m. This new method needs to receive the master inventory List, searchItem List, and the actual criteria to search as shown in Figure 10.

```
    // The just return the List
    return invItems;
}

/// <summary>
/// Search the item in the main Inventory List and return the New Search List
/// </summary>
/// <param name="invItems"></param>
/// <param name="searchItem"></param>
/// <param name="searchCriteria"></param>
/// <returns></returns>
0 references
public List<InvItem> SearchItem(List<InvItem> invItems, List<InvItem> searchItem, string searchCriteria)
{
}
```

Figure 10: Business Layer

n. To search the master inventory and place the results in invSearch, follow the code as shown in Figure 11.

o. Follow the comments in the code to fully understand the process. These comments should also be in the activity when submitted.

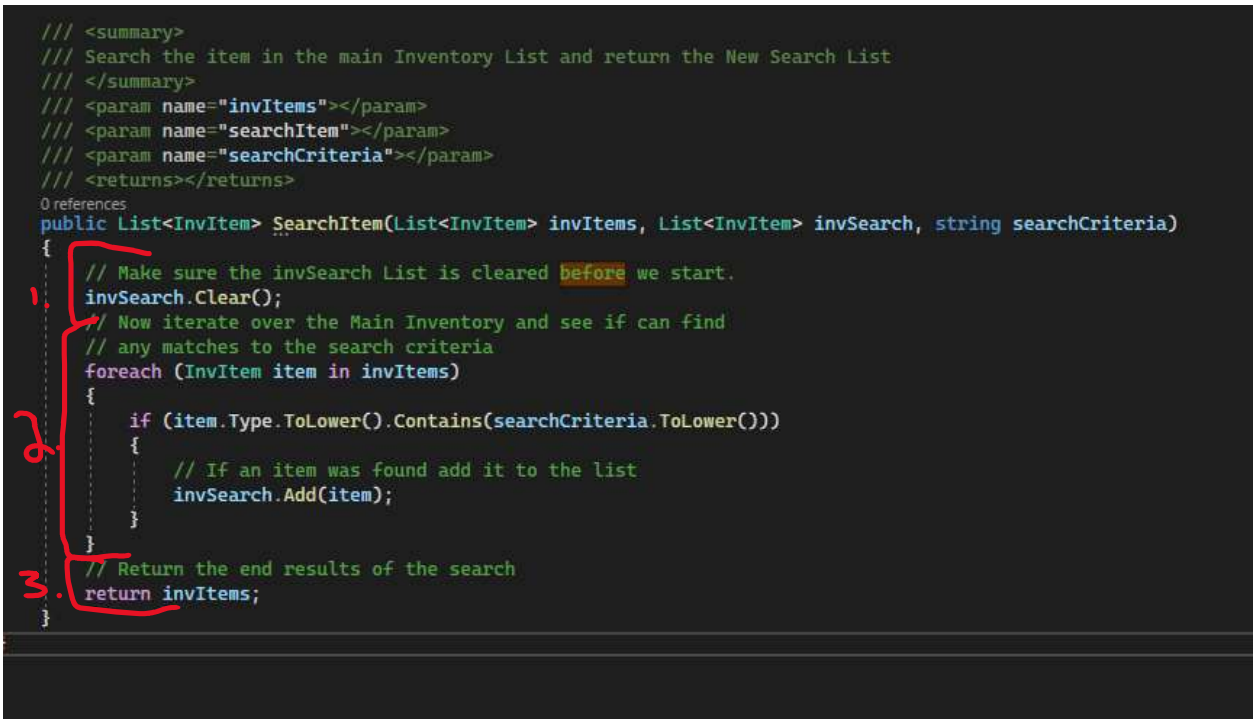

Figure 11: Search Code.

p. Now since the search code is written in the Business Layer, we can return back to the Presentation Layer and call this search method.

q. Follow the steps provided in Figure 12, then we will create the secondary form.

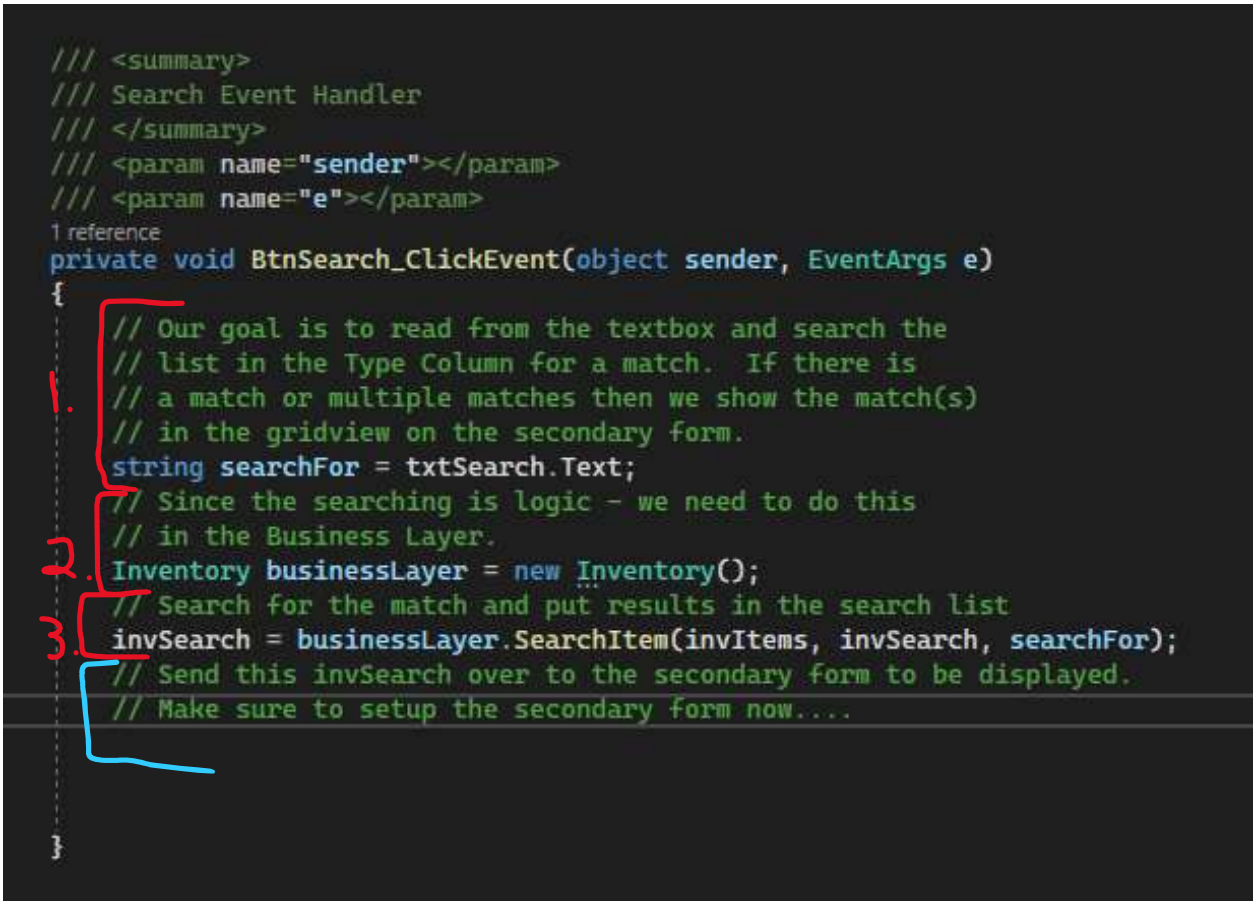

Figure 12: Presentation Layer

r. Before we can pass information to a secondary form at the end of Figure 12, we need to create the secondary form first.

s. Add a new form as shown in Figure 13 and name it "FrmSecondary."

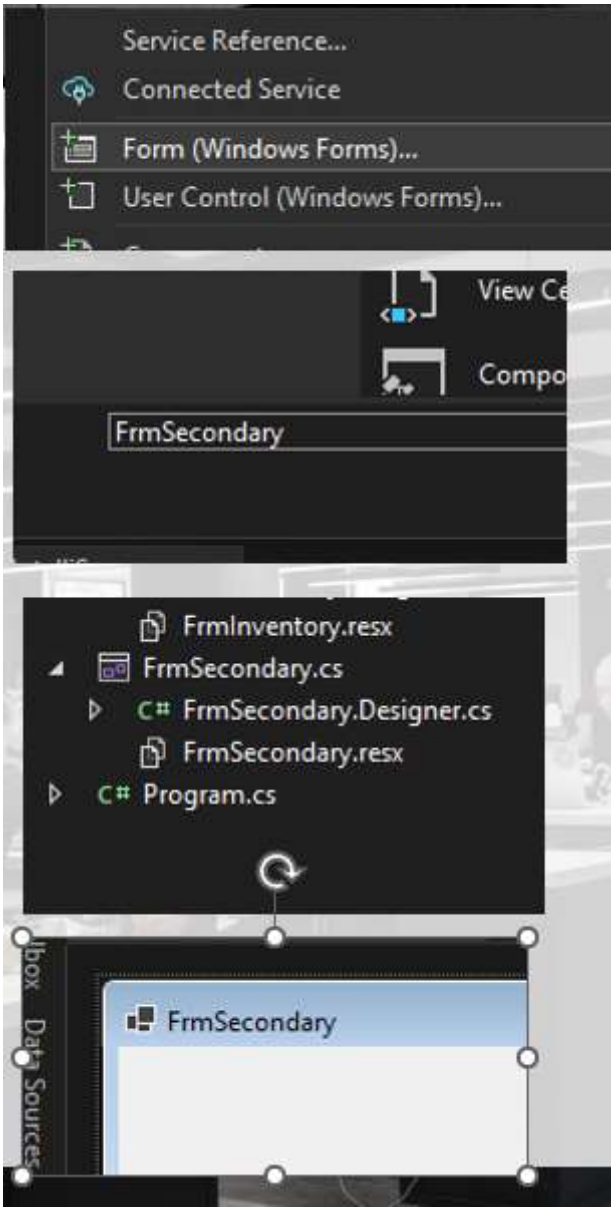

Figure 13: Add Secondary Form.

t. In FrmSecondary, add a "DataGridView" and name it "gvSearchResults" as shown in Figure 14.

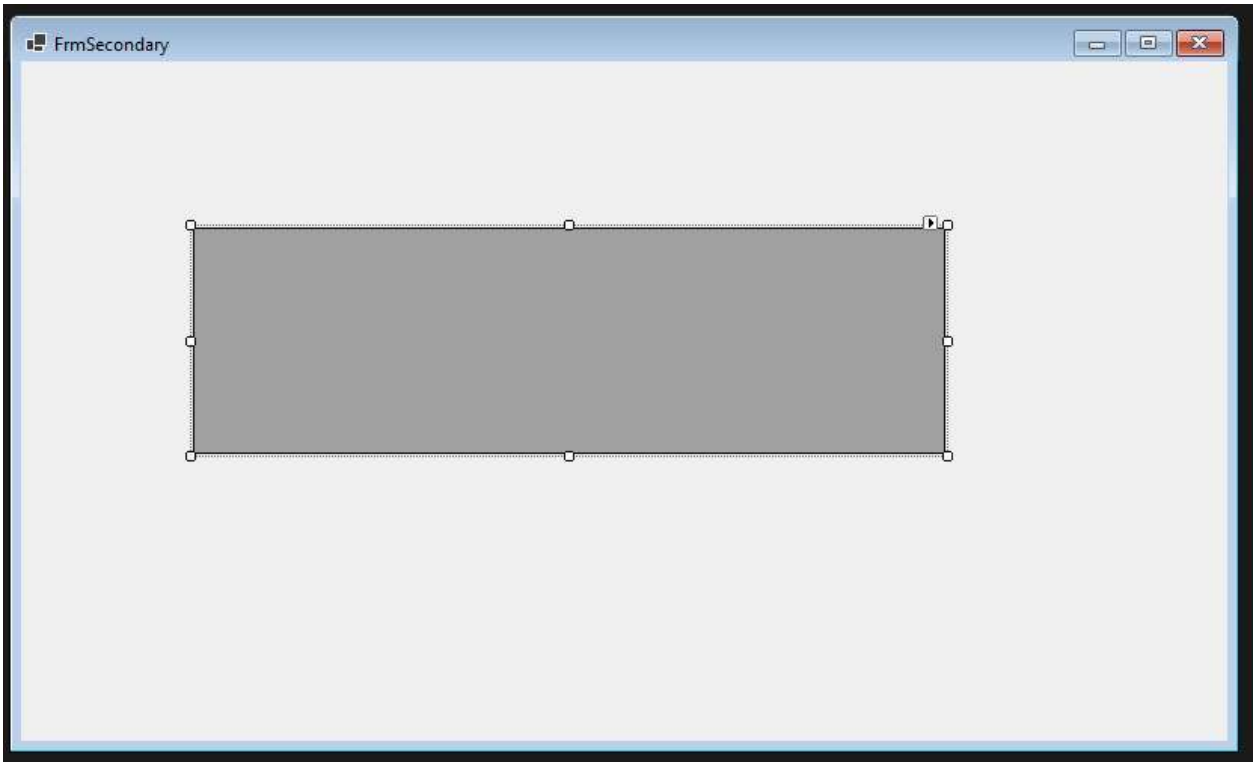

Figure 14: FrmSecondary with Data Grid View

u. In "FrmSecondary," go into the code behind.

v. Go into the constructor and create a parameterized constructor as shown in Figure 15, labeled "2." that takes the search list passed into the secondary form and puts it into mySearch object.

w. At the class level, instantiate the List and create an object named "mySearch" labeled by "1."

x. Create a Form Load Event handler as shown in "3." that binds the data to the data grid view using mySearch.

y. There might be a conflict with the accessibility of the InvItem class, so change the accessibility to "public" as shown in Figure 16.

```
namespace CST_150_ListTogv
{
    4 references
    public partial class FrmSecondary : Form
    {
        // Class level List
        List<InvItem> mySearch = new List<InvItem>();

        /// <summary>
        /// Parameterized Constructor
        /// Since this class is public we have to
        /// make InvItems.cs a public class so
        /// </summary>
        /// <param name="invSearch"></param>
        1 reference
        public FrmSecondary(List<InvItem> invSearch)
        {
            InitializeComponent();

            this.mySearch = invSearch;
        }

        /// <summary>
        /// When the form is loaded populate the grid
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        1 reference
        private void FrmLoad_EventHandler(object sender, EventArgs e)
        {
            gvSearchResults.DataSource = this.mySearch;
        }
    }
}
```

Figure 15: Code Behind Secondary Form

```
{
    /// <summary>
    /// Model Class that will structur
    /// all my inventory items.
    /// </summary>
    19 references
    public class InvItem
    {
        // Define the Properties
        3 references
        public string Type { get; set;
        1 reference
```

Figure 16: Change Accessibility.

a. In "FrmInventory," go into the code behind.

b. Complete the BtnSearch_ClickEvent handler.

c. Send the search results via the constructor and then show the form as shown in Figure 17.

d. Run the program and verify everything is working and the secondary form shows the search results.

```
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 reference
private void BtnSearch_ClickEvent(object sender, EventArgs e)
{
    // Our goal is to read from the textbox and search the
    // list in the Type Column for a match.  If there is
    // a match or multiple matches then we show the match(s)
    // in the gridview on the secondary form.
    string searchFor = txtSearch.Text;
    // Since the searching is logic - we need to do this
    // in the Business Layer.
    Inventory businessLayer = new Inventory();
    // Search for the match and put results in the search list
    invSearch = businessLayer.SearchItem(invItems, invSearch, searchFor);
    // Send this invSearch over to the secondary form to be displayed.
    // Make sure to setup the secondary form now....
    FrmSecondary frmSecondary = new FrmSecondary(invSearch);
    // Now to show the form
    // Notive .Show() method shows the form in a non modal form.  This means
    // you can click on the parent form.
    // .ShowDialog() method shows the form modally, meaning you cannot
    // go to the parent form.
    frmSecondary.ShowDialog();
```

Figure 17: Secondary Form Populated.

4. On Your Own, close the Secondary Form.

    a. Using Chapter 10 Section 6 in the textbook, learn how to close the secondary form returning back to the main form.

5. Submit the Activity as described in the digital classroom.