Milestone 7

Ian M. McConihay

College of Science, Engineering and Technology, Grand Canyon University

CST-150: C# Programming I

Mark Smithers

October 6, 2024

Video Link:

https://www.loom.com/share/812deda53f964378a0ab29f817309c9f?sid=03141830-ac3a-4855-80dd-df7e6252f634

Github: https://github.com/Ian-McConihay/CST-150

What was challenging?

Implementing the second form.

What did you learn?

Multiple Forms in WinForms.

How would you improve on the project?

Create a splash page.

How can you use what you learned on the job?

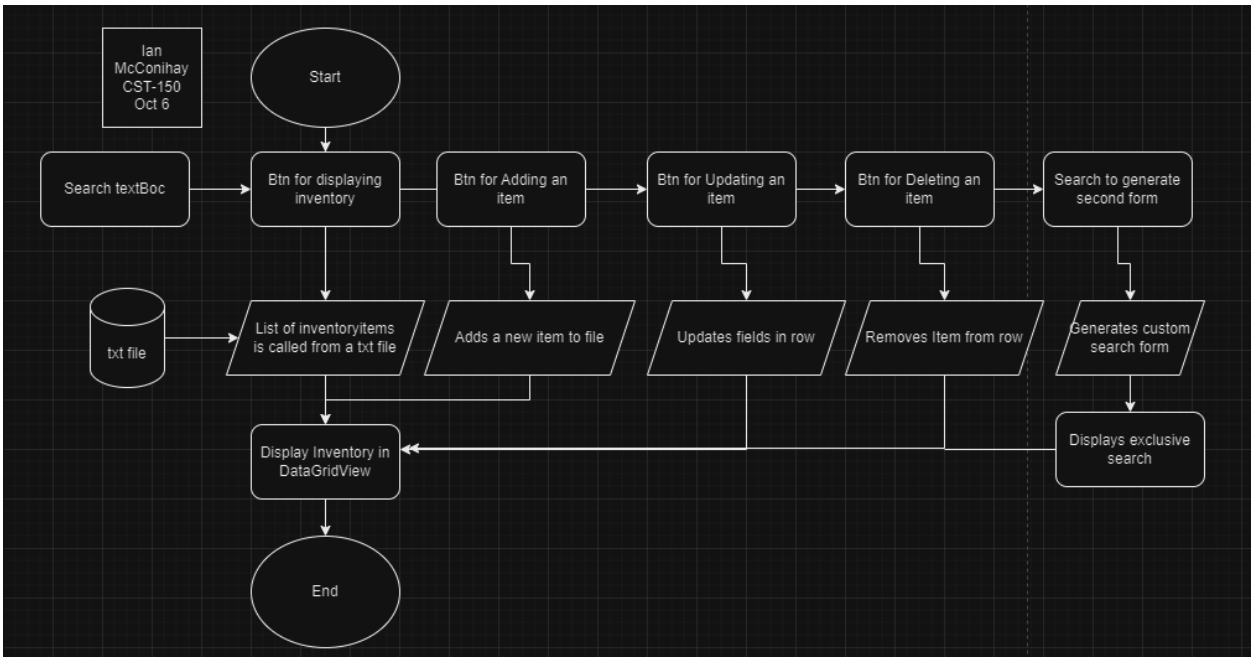
Crossing form data could be used for multiple applications crossing data.

Analysis on how global awareness, perspectives, and ethics could be added to learning additional software development techniques and applications.

Integrating global awareness, diverse perspectives, and ethical considerations into software development education enhances the learning experience and prepares students for real-world challenges. By exposing learners to global issues, such as data privacy, environmental sustainability, and accessibility, educators can emphasize the societal impacts of technology. Incorporating diverse perspectives fosters innovation; students can explore how cultural differences influence software usability and design. For example, understanding varying user needs across regions can lead to more inclusive applications, ensuring that products serve a wider audience. Ethical training is crucial in an era of rapid technological advancement. By discussing case studies on data breaches, algorithmic bias, and the implications of AI, students

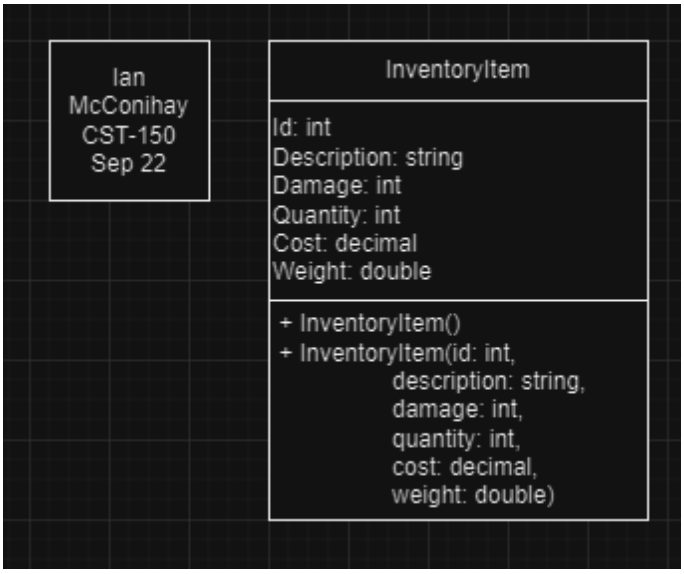
can develop a strong moral compass. This not only cultivates responsible developers but also encourages them to advocate for ethical practices in their workplaces. Ultimately, merging these elements into software development curriculum enriches the learning process, making it not just about coding, but about creating technology that is thoughtful, inclusive, and aligned with global values. This holistic approach prepares students to contribute positively to a complex, interconnected world.

Figure 1: FlowChart



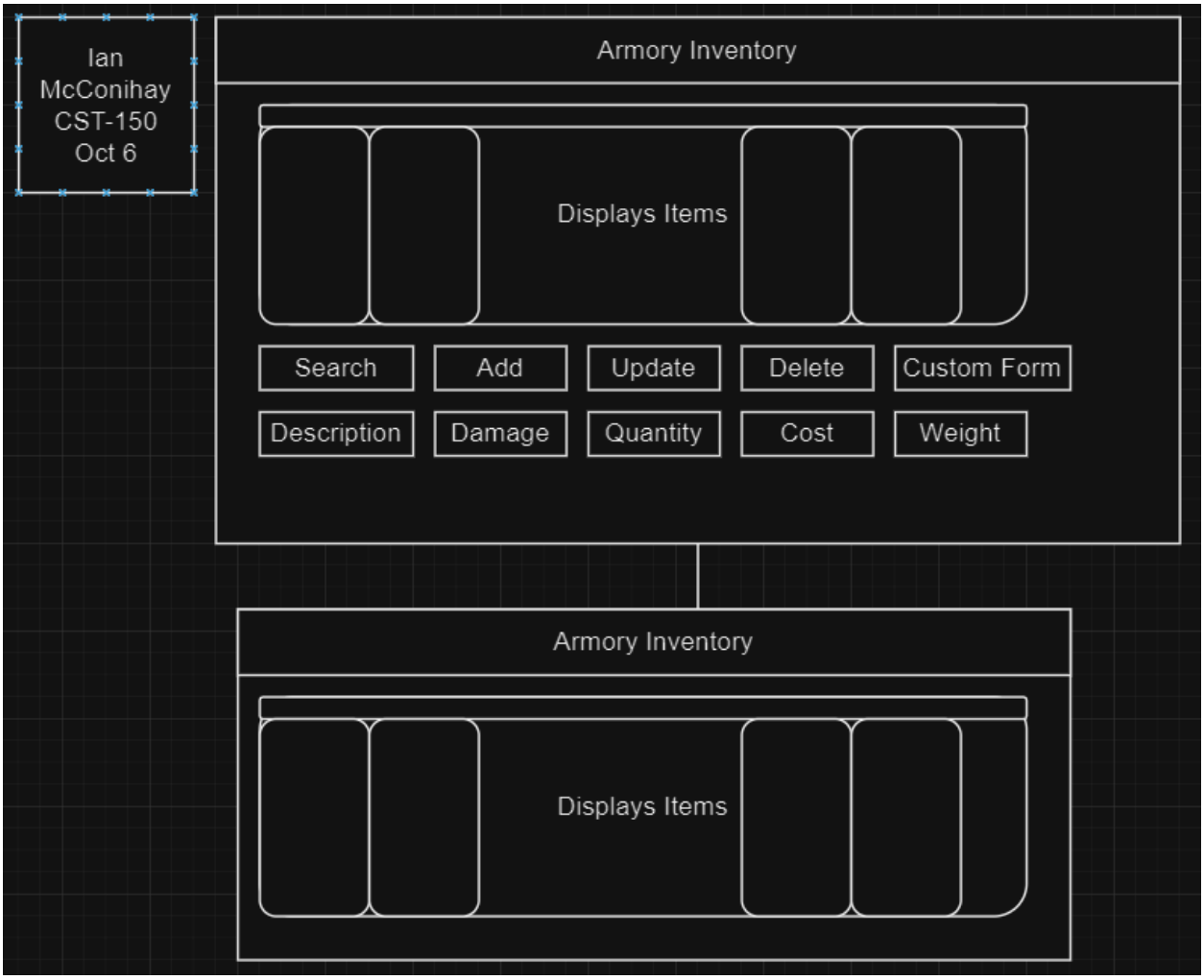
At the start of this application the text file data persists into a grid view of inventory items. There will be a series of buttons to perform Adding items, updating, deleting, and a search box. For the update button there will be a text field to generate a separate form for the user.

Figure 2: UML InventoryItem



No Changes.

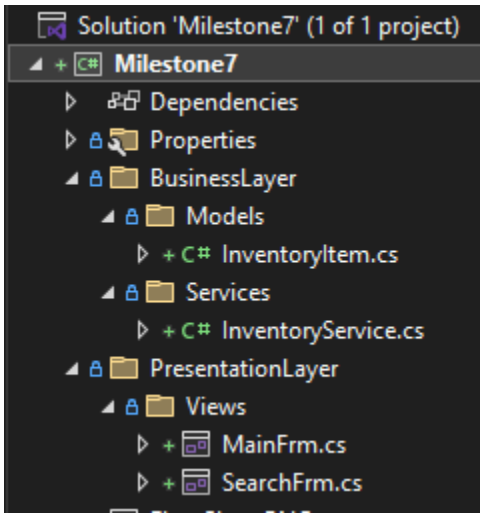
Figure 3: Wireframe



The updated wire frame has a handful of changes. The form has a name when displayed for the user. There are now a series of buttons to manage the inventory items. A few items and the table have been adjusted as well. A second form will be created using the custom search.

N-Layer

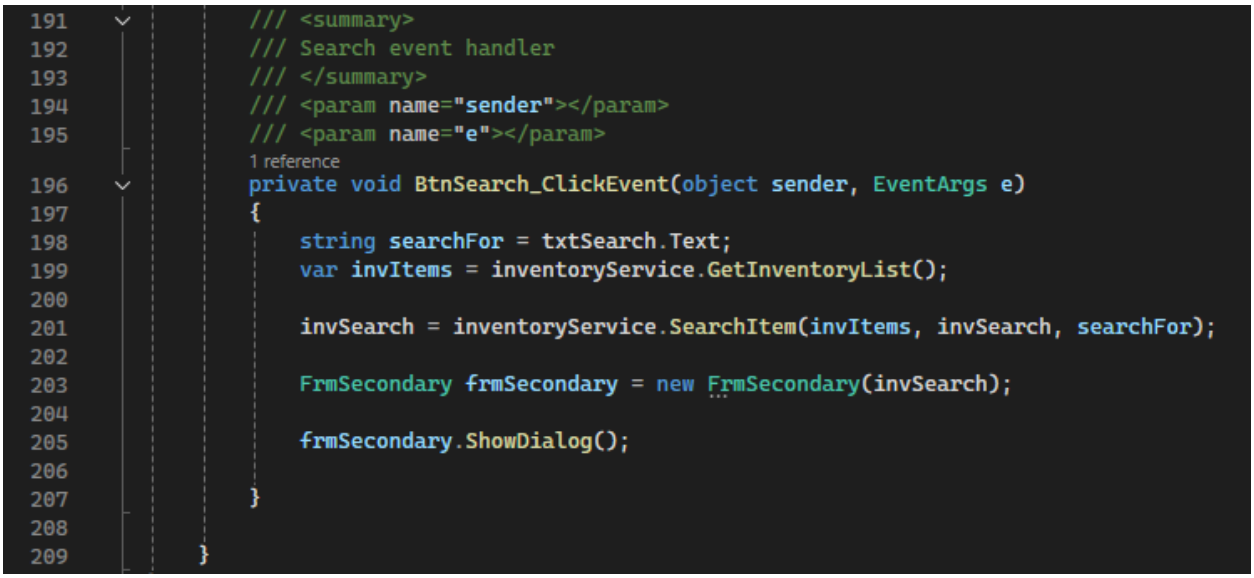
Figure 4: N-Layer



Here is a screenshot of the file structure for the application. N-layer was required for milestone. InventoryItem and InventoryService has been moved to the BusinessLayer and The PresentationLayer contains the Main form and the Search form for design.

Application Screenshots

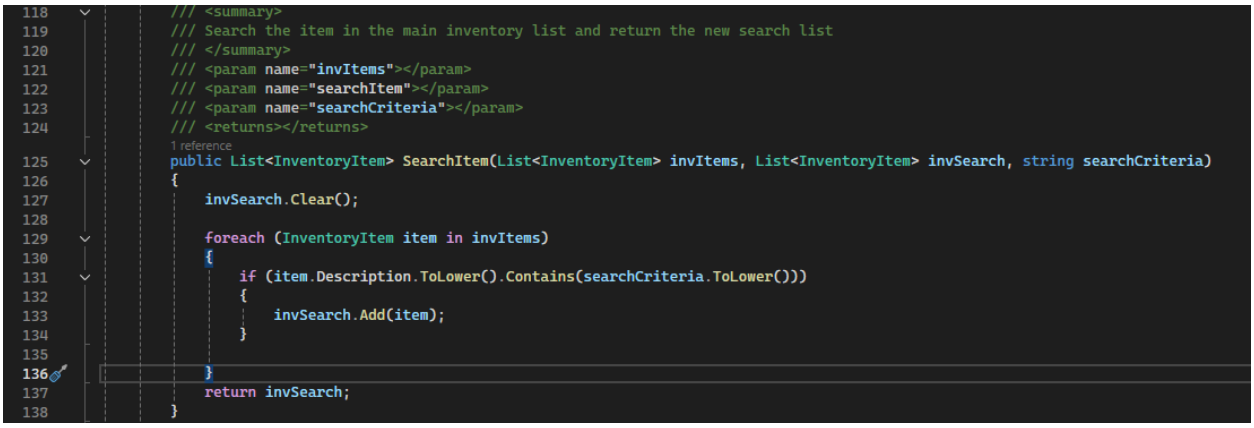
Figure 5: Code



```
191         /// <summary>
192         /// Search event handler
193         /// </summary>
194         /// <param name="sender"></param>
195         /// <param name="e"></param>
            1 reference
196         private void BtnSearch_ClickEvent(object sender, EventArgs e)
197         {
198             string searchFor = txtSearch.Text;
199             var invItems = inventoryService.GetInventoryList();
200
201             invSearch = inventoryService.SearchItem(invItems, invSearch, searchFor);
202
203             FrmSecondary frmSecondary = new FrmSecondary(invSearch);
204
205             frmSecondary.ShowDialog();
206
207         }
208
209     }
```

Figure 5 shows the click event for my search. The method targets the search text to query a new list to be generated into a new WinForm. I call my service method to construct the logic that is then populated for the second form.

Figure 6: Code



```
118     /// <summary>
119     /// Search the item in the main inventory list and return the new search list
120     /// </summary>
121     /// <param name="invItems"></param>
122     /// <param name="searchItem"></param>
123     /// <param name="searchCriteria"></param>
124     /// <returns></returns>
            1 reference
125     public List<InventoryItem> SearchItem(List<InventoryItem> invItems, List<InventoryItem> invSearch, string searchCriteria)
126     {
127         invSearch.Clear();
128
129         foreach (InventoryItem item in invItems)
130         {
131             if (item.Description.ToLower().Contains(searchCriteria.ToLower()))
132             {
133                 invSearch.Add(item);
134             }
135
136         }
137         return invSearch;
138     }
```

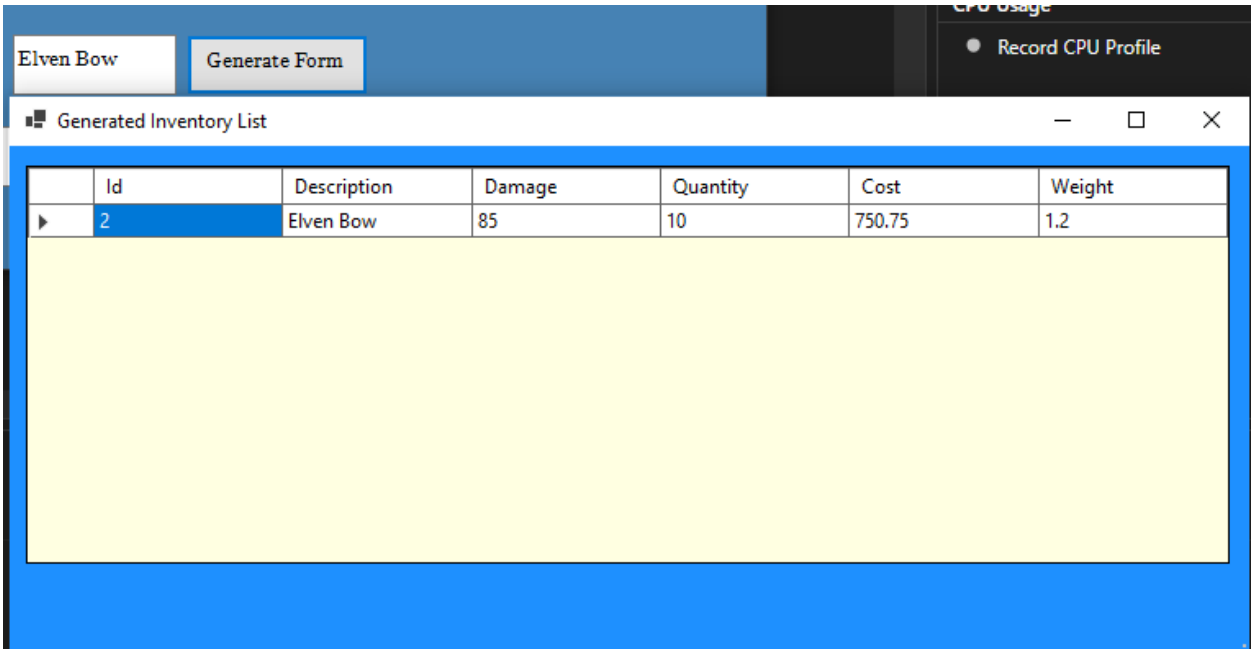
Figure 6 shows the Inventory service method SerachItem. This method clears the List each time for a new call. Then it iterates through the list to find descriptions that match the query to be added to the new List.

Figure 7: Code

```
/*
* Ian McConihay
* CST-150
* Milestone 7
* Oct 6 2024
*
*/

using Milestone7.BusinessLayer.Models;

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

        1 reference
        public FrmSecondary(List<InventoryItem> invSearch)
        {
            InitializeComponent();

            this.mySearch = invSearch;
        }

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

        1 reference
        private void closeButton_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

Figure 7 is the new SearchFrm. This has the event handler to provide the DataSource with the new search List. This form also has a method that allows the user to click anywhere in the form and it will close.

Figure 8: Application Start



Here is the second form being populated. The main form has “Elven Bow” as the search. The new generated form shows a separate grid of the custom search request.

Bug Reports

Bug Report: NONE

Class name

Method name :

Steps to reproduce the bug:

Expected results

Actual results

details: N/A

Solution

1. List your computer specs (type of computer, OS, memory, etc)

Device name	DESKTOP-IAQ5CCD
Processor	Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80 GHz
Installed RAM	8.00 GB (7.88 GB usable)
Device ID	A0AC8D02-4885-4491-B27B-B40F0A0D2E35
Product ID	00356-02139-31547-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	Touch support with 10 touch points

2. Create 3 test cases

Valid File with Proper Data: The method should correctly populate the armoryInventory array with InventoryItem objects based on properly formatted data in the file.

File is Empty: The method should display a warning message indicating the file is empty and leave the armoryInventory array uninitialized.

File with Incorrect Data Format: The method should show an error message indicating an issue with loading data and only initialize valid InventoryItem objects in the armoryInventory array.

3. List 3 Programming conventions that will be used all milestones

Naming, Format, and Documentation Conventions

4. Create Use case diagram

System Boundary: Representing the WinForms application.
Use Case: "View Inventory" indicating the functionality provided by the application.
Actor: "User" who interacts with the system to view the inventory.

Monday
Start: 900pm End: 9:30pm Activity: Read announcements
Start: 930pm End: 1030 Activity: DQ1 and DQ 2
Start: 1030pm End: 1100pm Activity: Read Book
Tuesday
Start: 900pm End: 9:30pm Activity: Participation post
Start: 930pm End: 1030 Activity: Activity 7
Start: 1030pm End: 1100pm Activity: Read Book
Wednesday
Start: End: Activity: N/A
Start: End: Activity: N/A
Start: End: Activity: N/A
Thursday
Start: 900pm End: 9:30pm Activity: Participation post
Start: 930pm End: 1030 Activity: Activity 7
Start: 1030pm End: 1100pm Activity: Read Book
Friday
Start: 900pm End: 9:30pm Activity: Participation post
Start: 930pm End: 1030 Activity: Milestone
Start: 1030pm End: 1100pm Activity: Read Book
Saturday
Start: 900pm End: 9:30pm Activity: Activity 7
Start: 930pm End: 1030 Activity: Milestone
Start: 1030pm End: 1100pm Activity: Milestone
Sunday
Start: 900pm End: 9:30pm Activity: Activity 7
Start: 930pm End: 1030 Activity: Milestone
Start: 1030pm End: 1100pm Activity: Milestone