Owen Lindsey

Professor Smithers, Mark

08/20/2023

CST-150

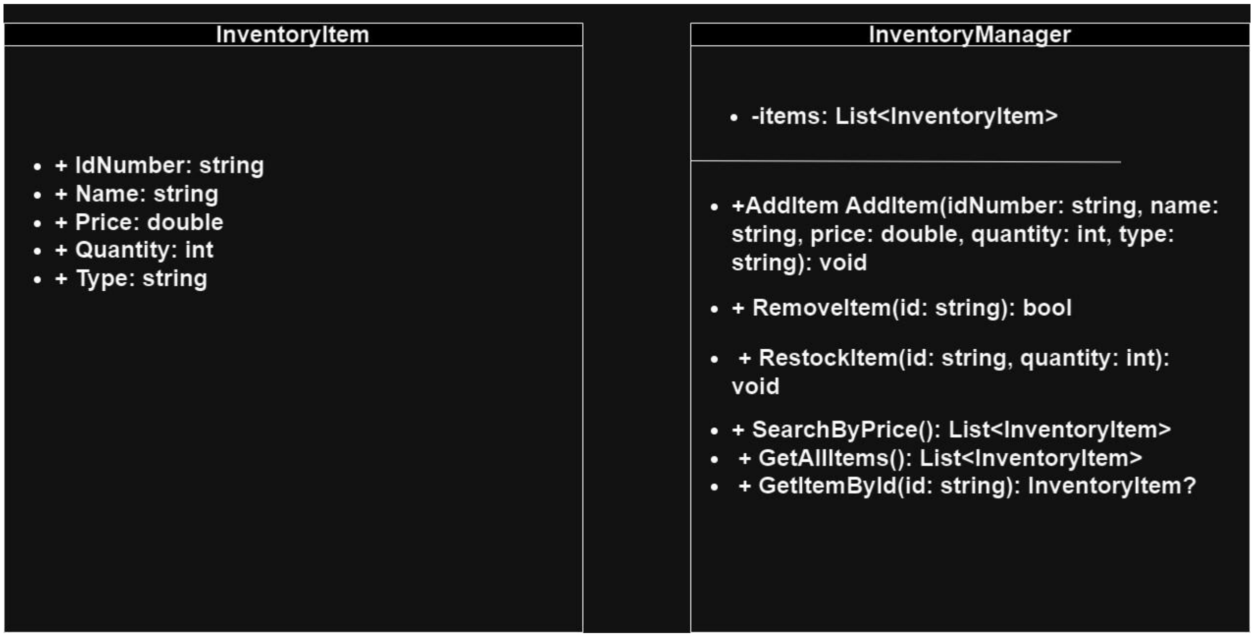
Grand Canyon University

Milestone

Video: https://youtu.be/atI3LlvYSc0

GitHub:

UML


InventoryItem
+ IdNumber: string
+ Name: string
+ Price: double
+ Quantity: int
+ Type: string
InventoryManager
-items: List<InventoryItem>
+AddItem AddItem(idNumber: string, name: string, price: double, quantity: int, type: string): void
+ RemoveItem(id: string): bool
+ RestockItem(id: string, quantity: int): void
+ SearchByPrice(): List<InventoryItem>
+ GetAllItems(): List<InventoryItem>
+ GetItemById(id: string): InventoryItem?

Written discussion

What I Learned:

- Importance of Iterative Development: Each functionality, from setting up the system to debugging, gave me a deeper appreciation for iterative development. By building the system piece by piece and continuously refining, I could adapt to challenges and ensure the application remained on the right track.

- Collaborative Problem Solving: Working collaboratively allowed me to understand the power of two minds working together. Having another perspective, like from ChatGPT, was invaluable in tackling complex problems and understanding different approaches to solutions.

Challenges I Faced:

- Errors and Debugging: Throughout development, I encountered several issues that needed debugging. Addressing these in real-time, especially when some errors were not immediately clear, tested my problem-solving skills.

- Naming Confusions: A significant challenge I faced was the overlap in naming conventions, which highlighted the importance of clear and distinct naming in coding to avoid confusion.

- Unit Testing: Ensuring the application worked as intended was crucial, and setting up a testing environment with XUnit provided its own set of challenges. Making the tests work with the actual code was not always straightforward but was a learning experience in itself.

Future Improvements:

- Code Clarity: There's always room for refactoring. With more time, I'd love to revisit parts of the code to enhance clarity and maintainability.

- Database Integration: Moving from a list-based system to a database-backed structure is a definite future step. This would elevate the application's scalability and robustness.

- User Experience: If this were a GUI application, I'd focus on enhancing the user interface to make it more intuitive and feature-rich.

- Expand Testing: To ensure thorough coverage, I'd want to expand the testing suite, addressing more edge cases and possibly venturing into integration testing.

Application to Future Projects:

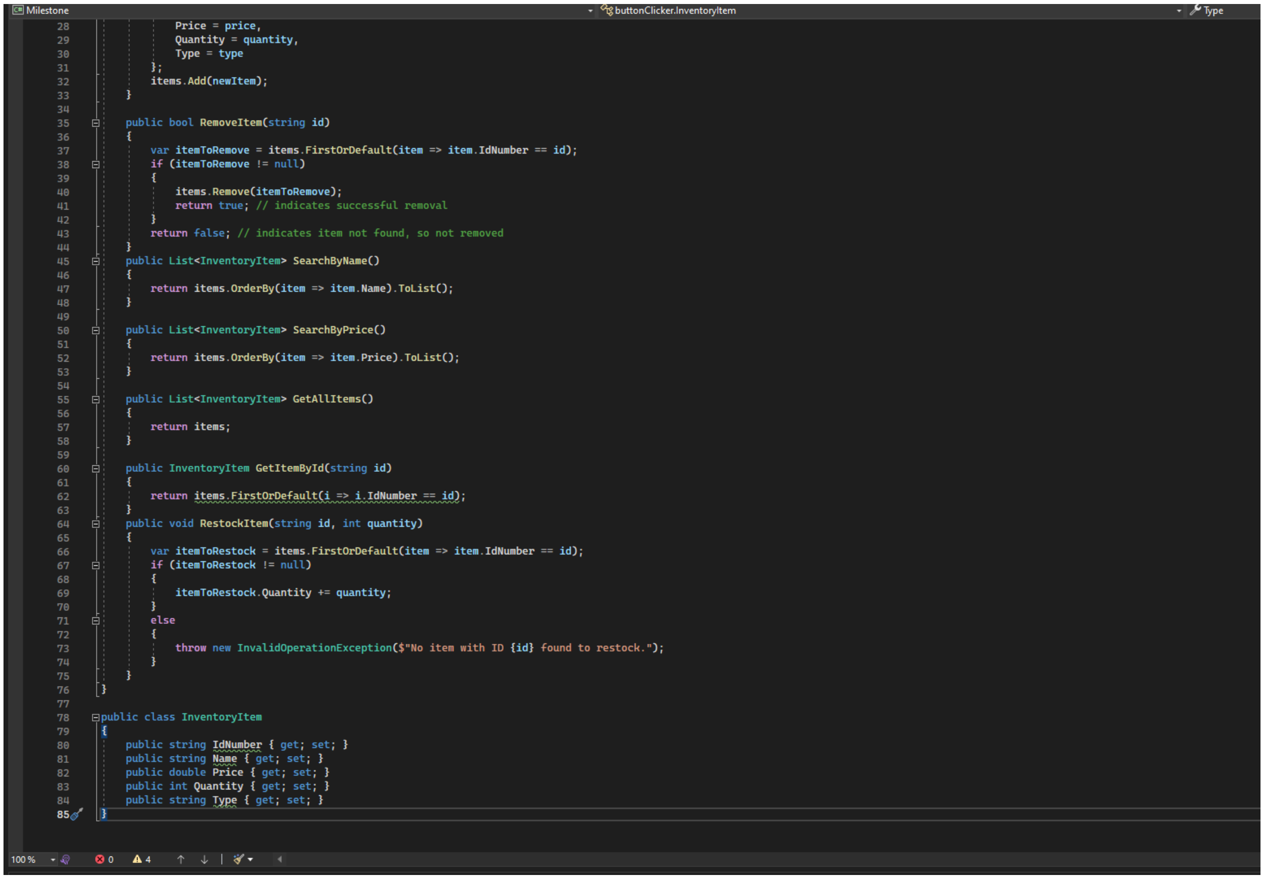
- Code Modularity: The experience reaffirmed the value of writing adaptable and modular code. This principle isn't just for inventory systems but is universally applicable, whether I'm developing a game, a mobile app, or a web service.

- Rigorous Error Handling: No matter the project, robust error handling is key. Ensuring users don't face unhandled errors but instead receive useful feedback will always be a priority.

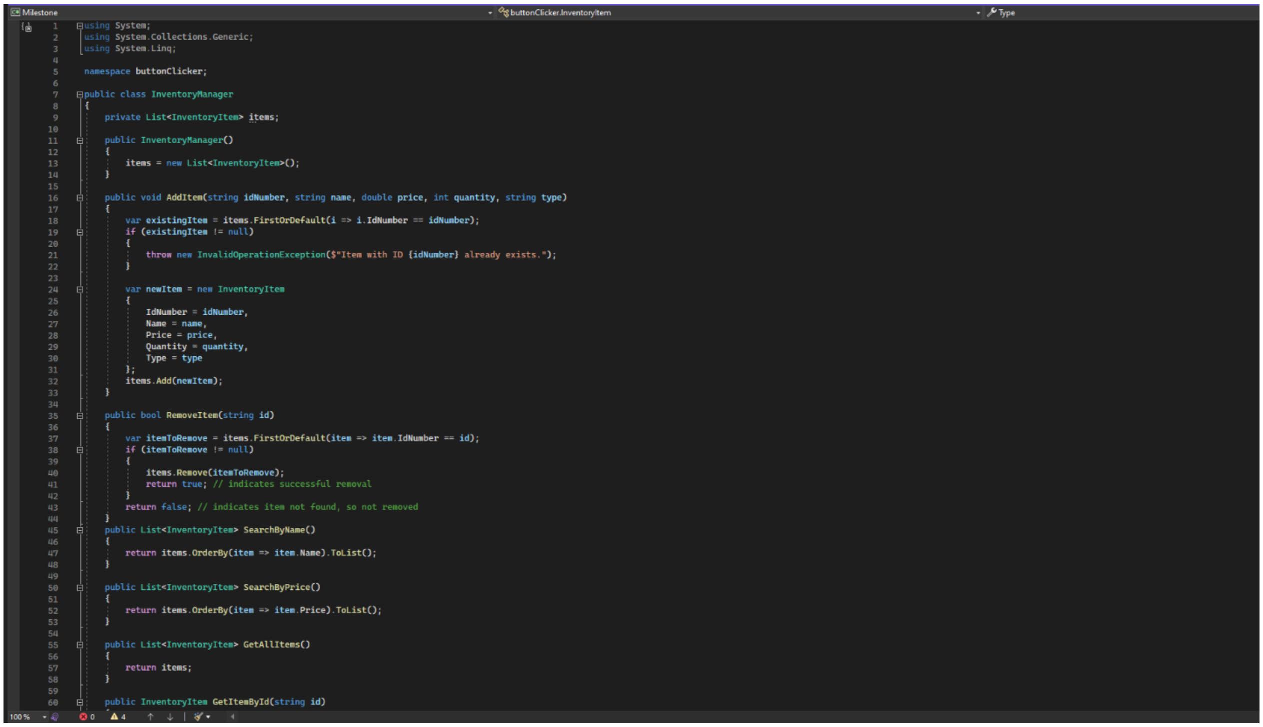
- Emphasis on Testing: This project underlined the importance of testing. I've seen firsthand how crucial it is to ensure every functionality is working as intended, and I'll carry this lesson forward in all my future development endeavors.

Screen-shots

Main Application:

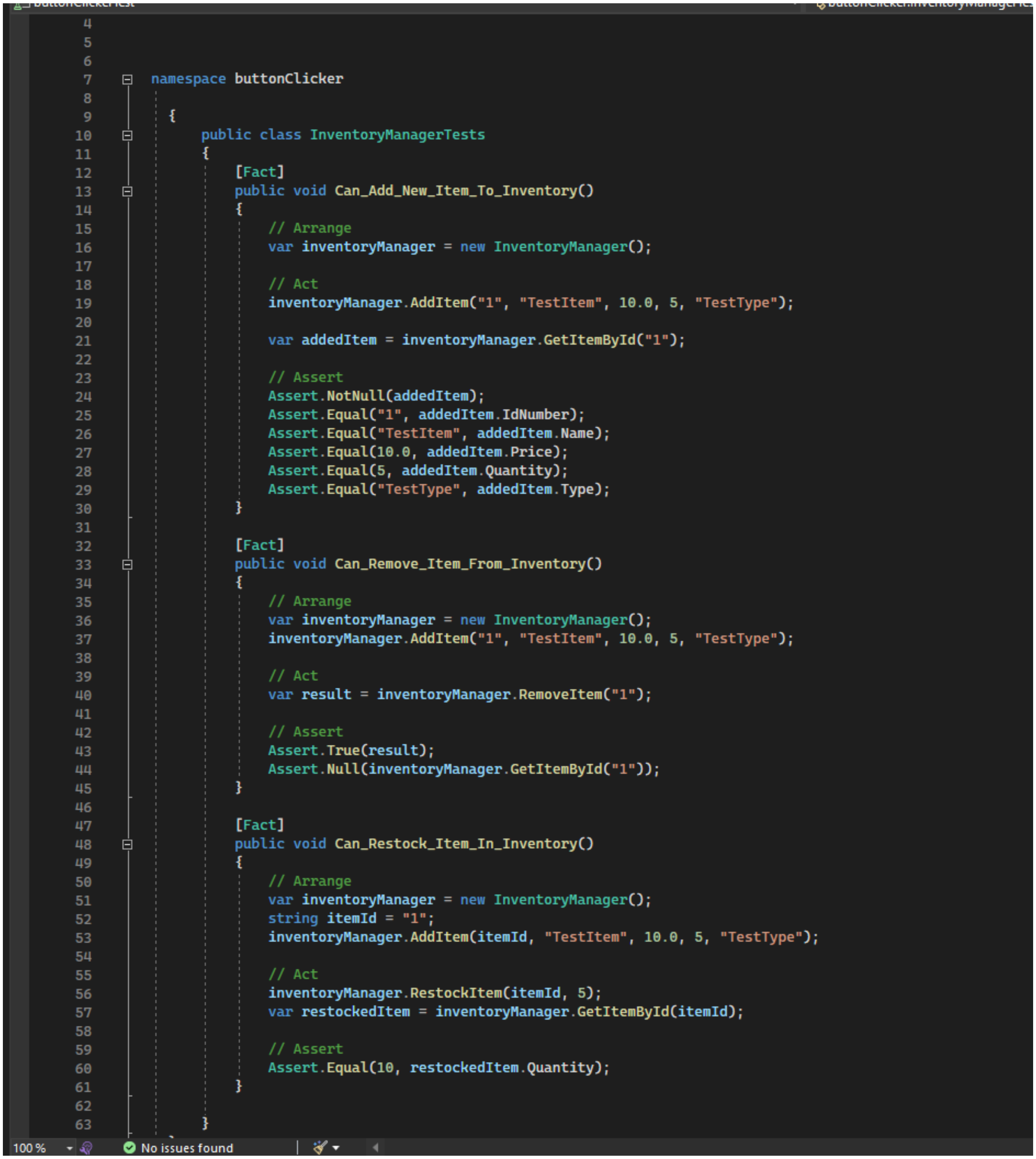


```
            Price = price,
            Quantity = quantity,
            Type = type
        };
        items.Add(newItem);
    }

    public bool RemoveItem(string id)
    {
        var itemToRemove = items.FirstOrDefault(item => item.IdNumber == id);
        if (itemToRemove != null)
        {
            items.Remove(itemToRemove);
            return true; // indicates successful removal
        }
        return false; // indicates item not found, so not removed
    }
    public List<InventoryItem> SearchByName()
    {
        return items.OrderBy(item => item.Name).ToList();
    }

    public List<InventoryItem> SearchByPrice()
    {
        return items.OrderBy(item => item.Price).ToList();
    }

    public List<InventoryItem> GetAllItems()
    {
        return items;
    }

    public InventoryItem GetItemById(string id)
    {
        return items.FirstOrDefault(i => i.IdNumber == id);
    }
    public void RestockItem(string id, int quantity)
    {
        var itemToRestock = items.FirstOrDefault(item => item.IdNumber == id);
        if (itemToRestock != null)
        {
            itemToRestock.Quantity += quantity;
        }
        else
        {
            throw new InvalidOperationException($"No item with ID {id} found to restock.");
        }
    }
}

public class InventoryItem
{
    public string IdNumber { get; set; }
    public string Name { get; set; }
    public double Price { get; set; }
    public int Quantity { get; set; }
    public string Type { get; set; }
}
```



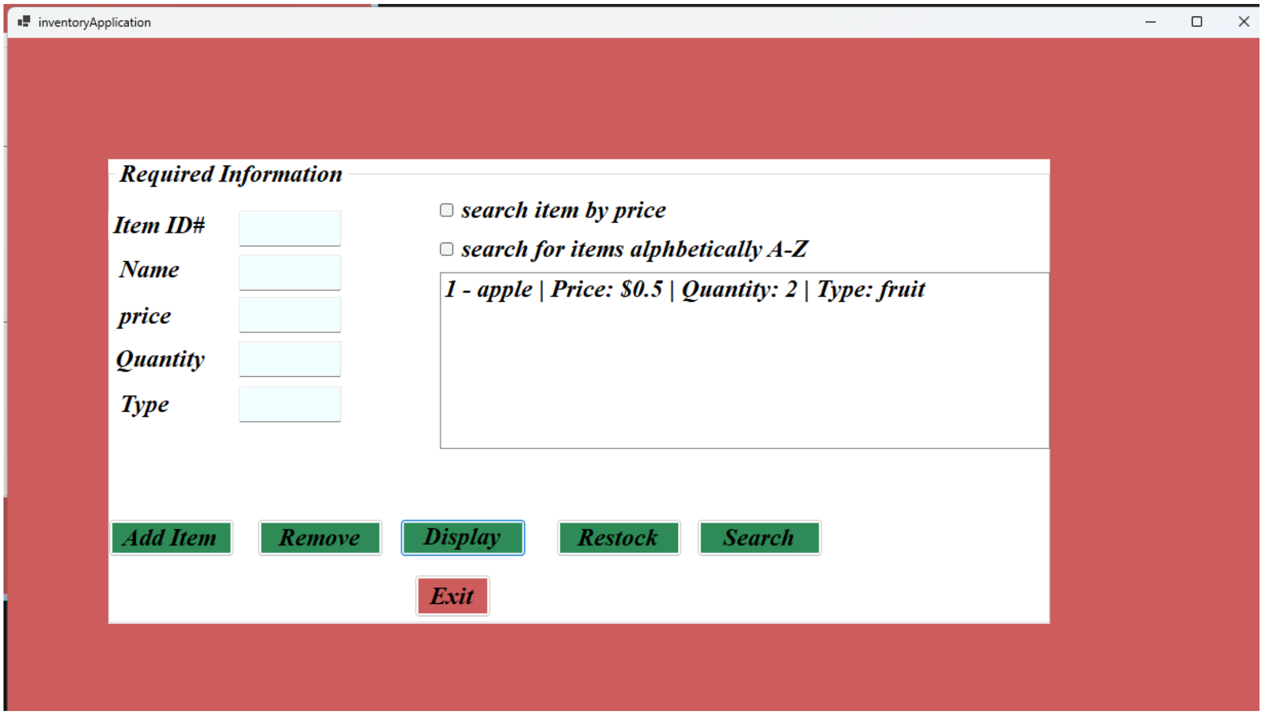
```csharp
using System;
using System.Collections.Generic;
using System.Linq;

namespace buttonClicker;

public class InventoryManager
{
    private List<InventoryItem> items;

    public InventoryManager()
    {
        items = new List<InventoryItem>();
    }

    public void AddItem(string idNumber, string name, double price, int quantity, string type)
    {
        var existingItem = items.FirstOrDefault(i => i.IdNumber == idNumber);
        if (existingItem != null)
        {
            throw new InvalidOperationException($"Item with ID {idNumber} already exists.");
        }

        var newItem = new InventoryItem
        {
            IdNumber = idNumber,
            Name = name,
            Price = price,
            Quantity = quantity,
            Type = type
        };
        items.Add(newItem);
    }

    public bool RemoveItem(string id)
    {
        var itemToRemove = items.FirstOrDefault(item => item.IdNumber == id);
        if (itemToRemove != null)
        {
            items.Remove(itemToRemove);
            return true; // indicates successful removal
        }
        return false; // indicates item not found, so not removed
    }
    public List<InventoryItem> SearchByName()
    {
        return items.OrderBy(item => item.Name).ToList();
    }

    public List<InventoryItem> SearchByPrice()
    {
        return items.OrderBy(item => item.Price).ToList();
    }

    public List<InventoryItem> GetAllItems()
    {
        return items;
    }

    public InventoryItem GetItemById(string id)
```

Xunit test code

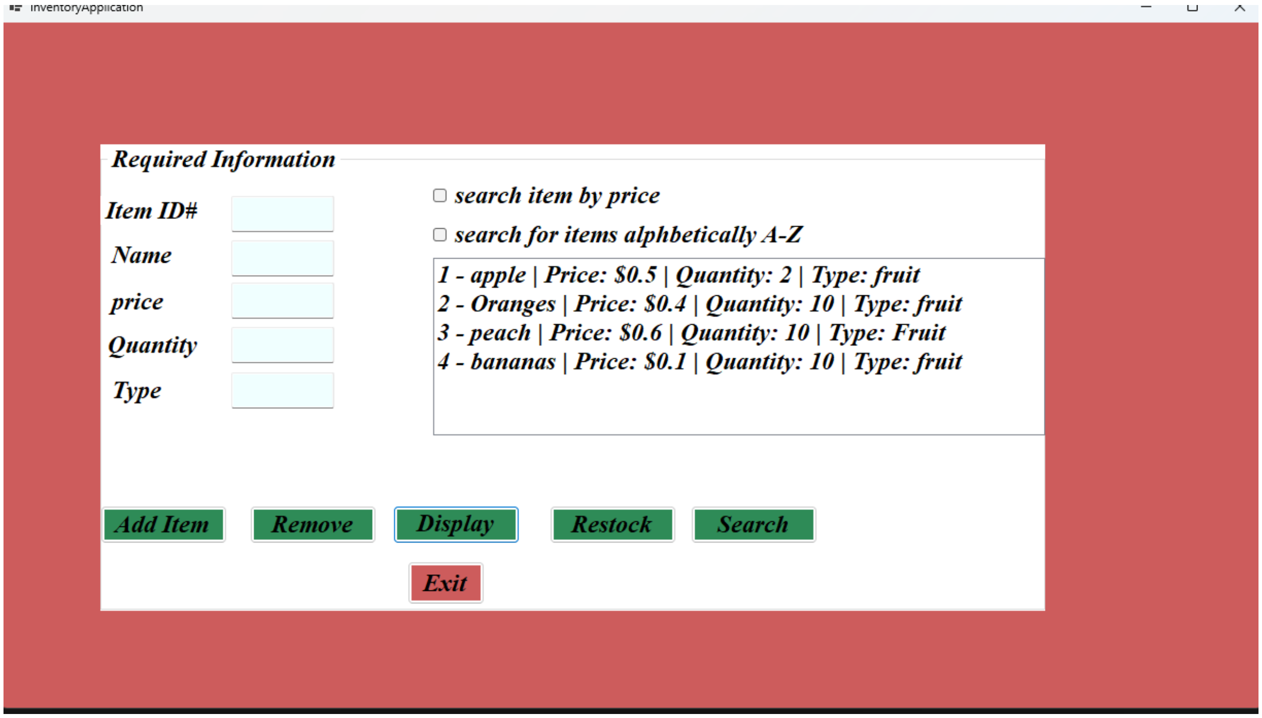


```
namespace buttonClicker
  {
      public class InventoryManagerTests
      {
          [Fact]
          public void Can_Add_New_Item_To_Inventory()
          {
              // Arrange
              var inventoryManager = new InventoryManager();

              // Act
              inventoryManager.AddItem("1", "TestItem", 10.0, 5, "TestType");

              var addedItem = inventoryManager.GetItemById("1");

              // Assert
              Assert.NotNull(addedItem);
              Assert.Equal("1", addedItem.IdNumber);
              Assert.Equal("TestItem", addedItem.Name);
              Assert.Equal(10.0, addedItem.Price);
              Assert.Equal(5, addedItem.Quantity);
              Assert.Equal("TestType", addedItem.Type);
          }

          [Fact]
          public void Can_Remove_Item_From_Inventory()
          {
              // Arrange
              var inventoryManager = new InventoryManager();
              inventoryManager.AddItem("1", "TestItem", 10.0, 5, "TestType");

              // Act
              var result = inventoryManager.RemoveItem("1");

              // Assert
              Assert.True(result);
              Assert.Null(inventoryManager.GetItemById("1"));
          }

          [Fact]
          public void Can_Restock_Item_In_Inventory()
          {
              // Arrange
              var inventoryManager = new InventoryManager();
              string itemId = "1";
              inventoryManager.AddItem(itemId, "TestItem", 10.0, 5, "TestType");

              // Act
              inventoryManager.RestockItem(itemId, 5);
              var restockedItem = inventoryManager.GetItemById(itemId);

              // Assert
              Assert.Equal(10, restockedItem.Quantity);
          }

      }
```

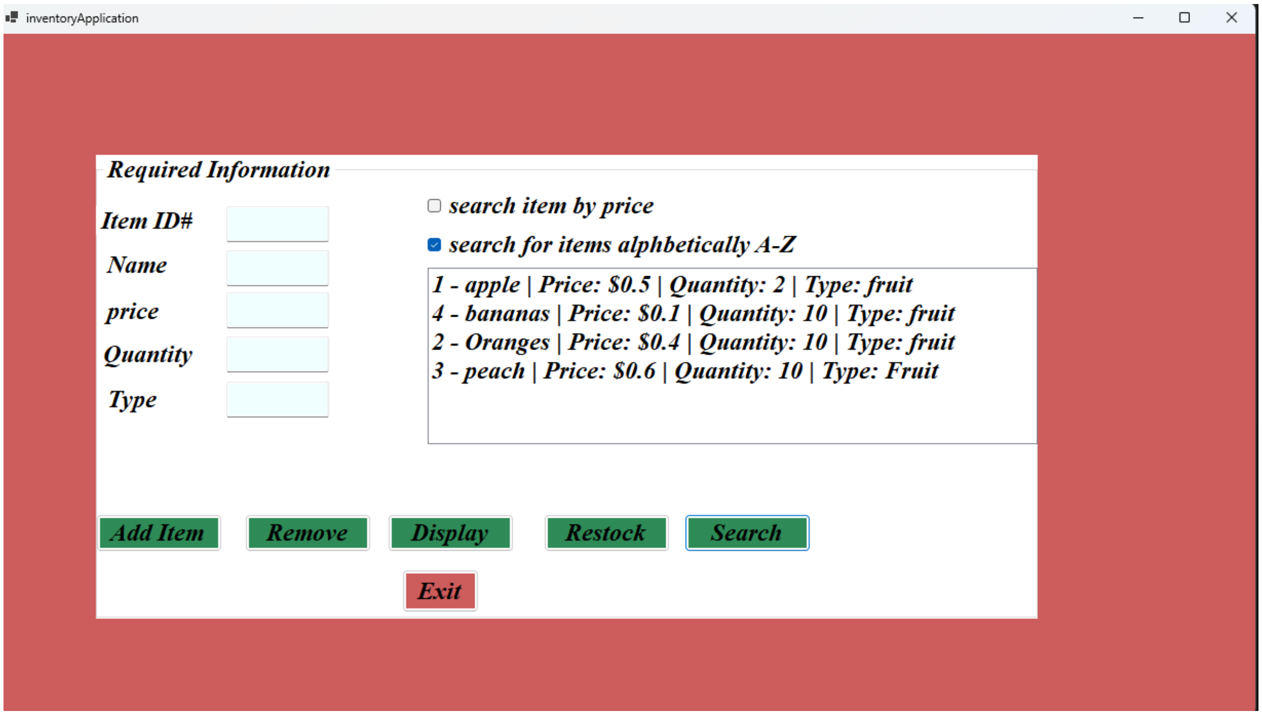
Application running:



```
1 - apple | Price: $0.5 | Quantity: 2 | Type: fruit
```



```
1 - apple | Price: $0.5 | Quantity: 2 | Type: fruit
2 - Oranges | Price: $0.4 | Quantity: 10 | Type: fruit
3 - peach | Price: $0.6 | Quantity: 10 | Type: Fruit
4 - bananas | Price: $0.1 | Quantity: 10 | Type: fruit
```



Required Information

Item ID#

Name

price

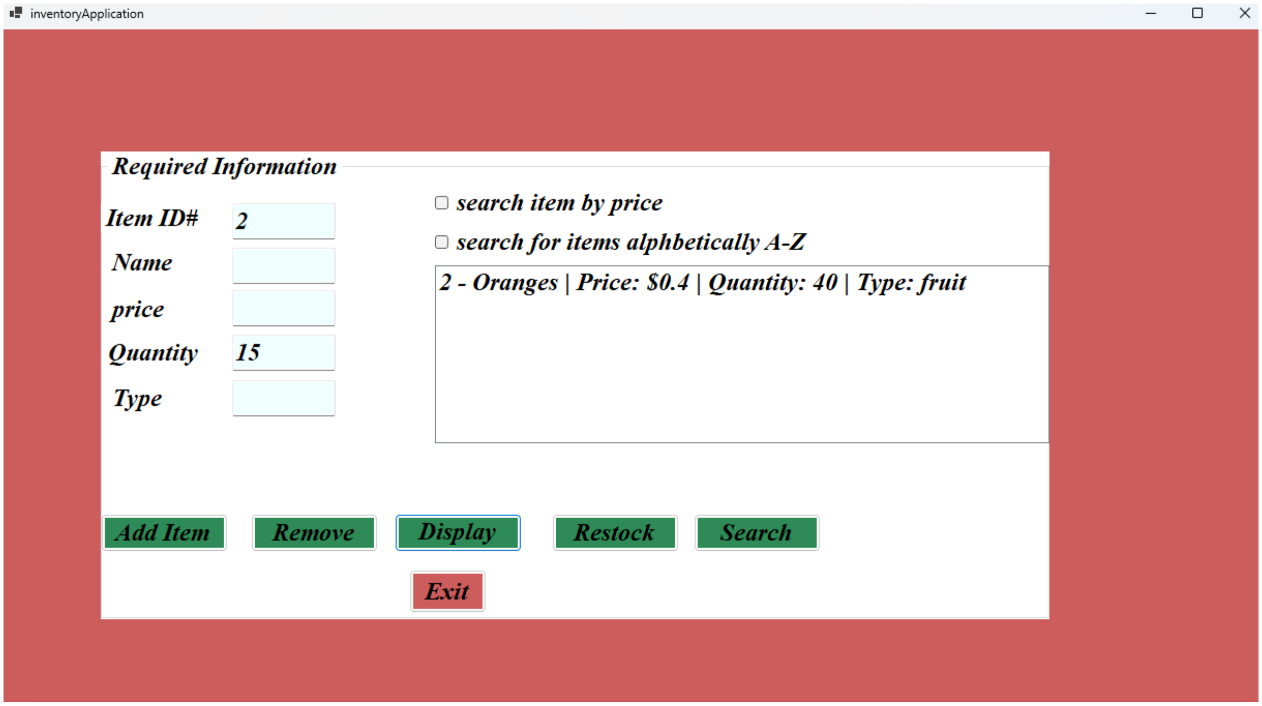
Quantity

Type

☐ search item by price

☑ search for items alphbetically A-Z

```
1 - apple | Price: $0.5 | Quantity: 2 | Type: fruit
4 - bananas | Price: $0.1 | Quantity: 10 | Type: fruit
2 - Oranges | Price: $0.4 | Quantity: 10 | Type: fruit
3 - peach | Price: $0.6 | Quantity: 10 | Type: Fruit
```



Required Information

Item ID# 2

Name

price

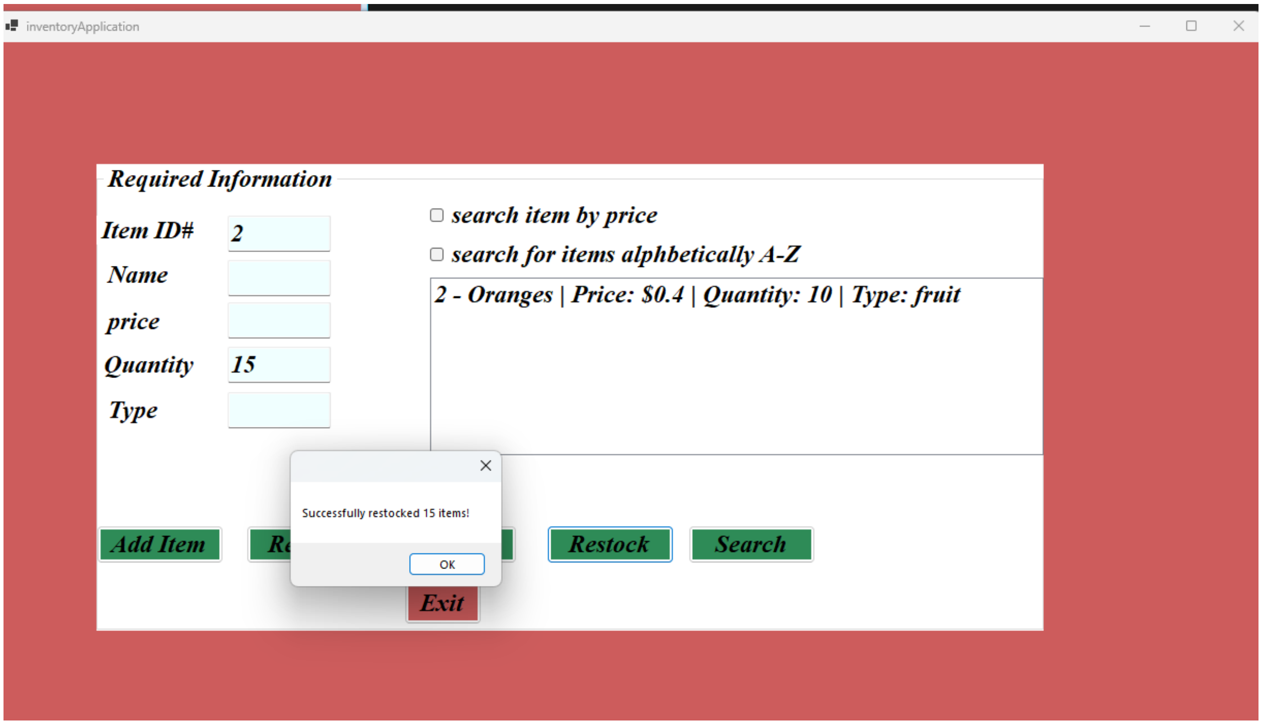
Quantity 15

Type

☐ search item by price

☐ search for items alphbetically A-Z

```
2 - Oranges | Price: $0.4 | Quantity: 40 | Type: fruit
```



Required Information

Item ID# 2

Name

price

Quantity 15

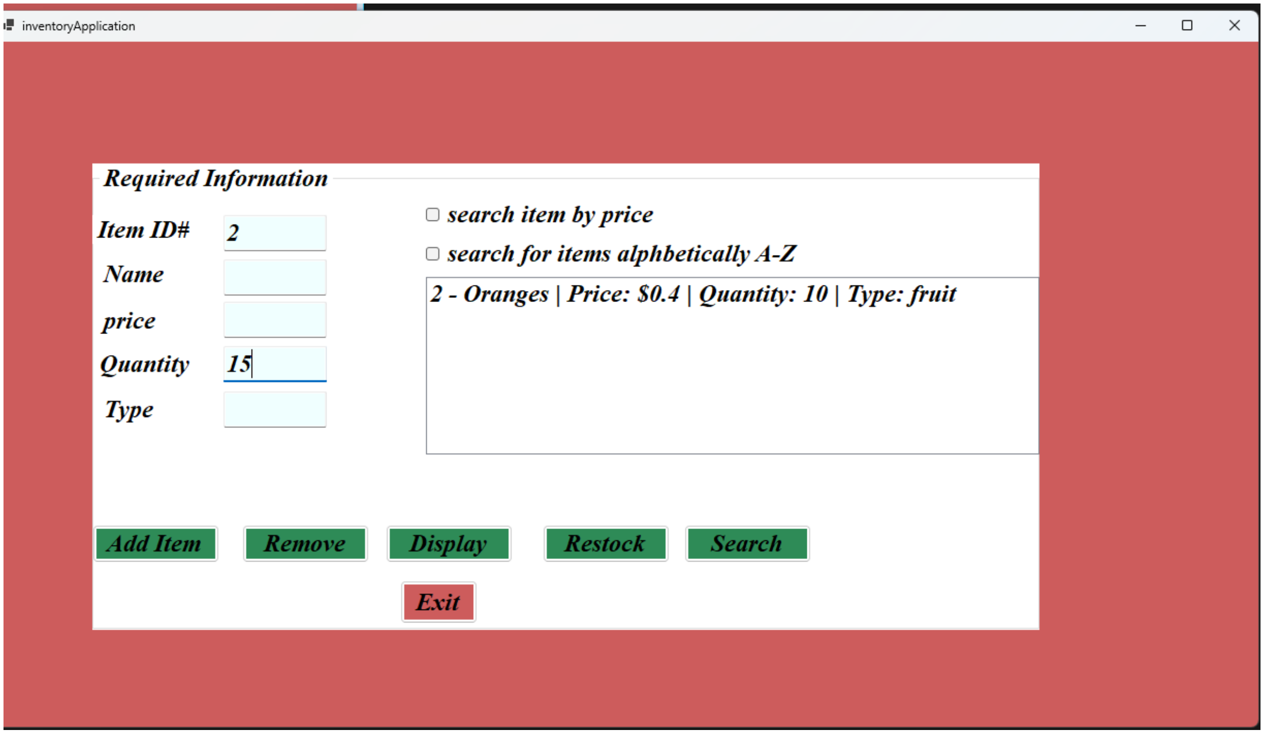
Type

□ search item by price

□ search for items alphbetically A-Z

2 - Oranges | Price: $0.4 | Quantity: 10 | Type: fruit

Successfully restocked 15 items!



Required Information

Item ID# 2

Name

price

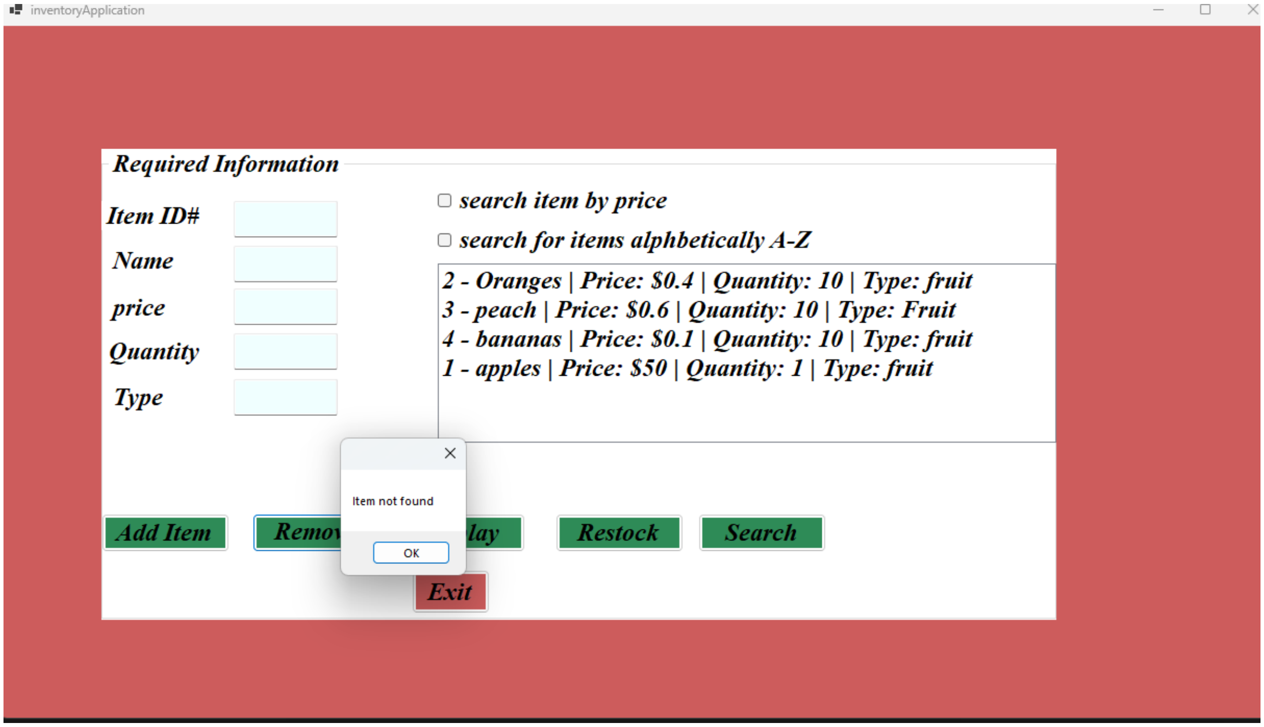
Quantity 15

Type

□ search item by price

□ search for items alphbetically A-Z

2 - Oranges | Price: $0.4 | Quantity: 10 | Type: fruit



Item ID#

Name

price

Quantity

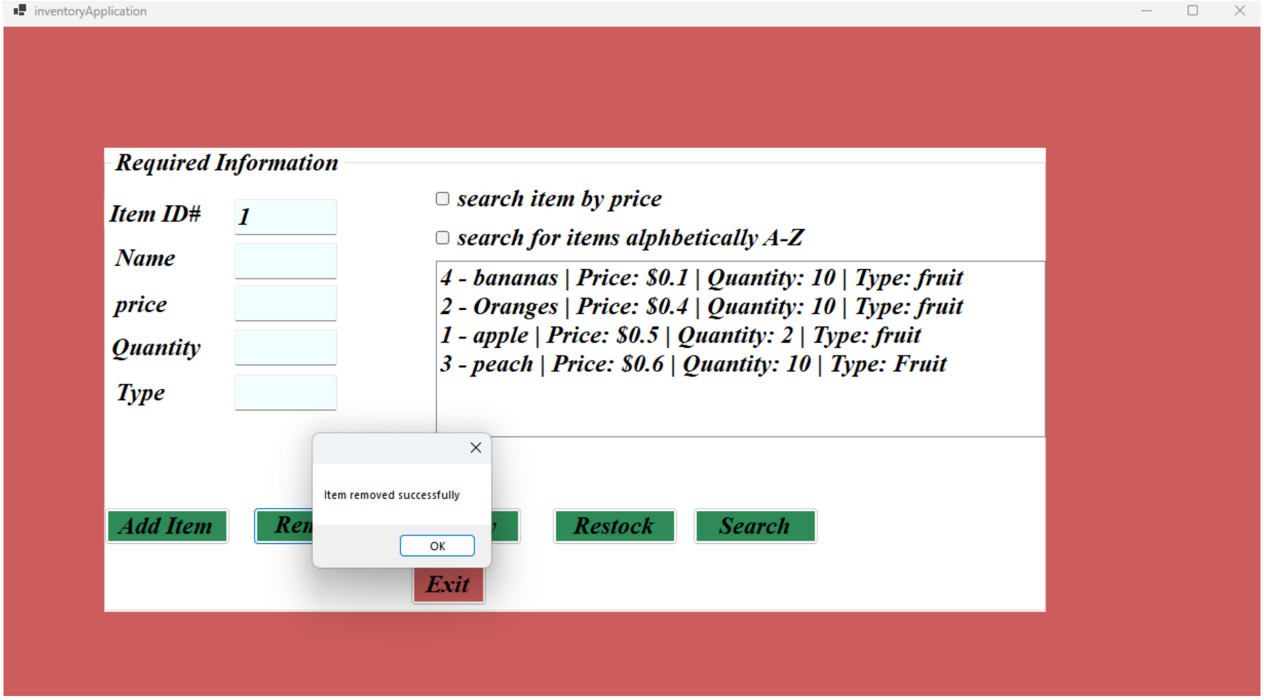
Type

□ search item by price

□ search for items alphbetically A-Z

```
2 - Oranges | Price: $0.4 | Quantity: 10 | Type: fruit
3 - peach | Price: $0.6 | Quantity: 10 | Type: Fruit
4 - bananas | Price: $0.1 | Quantity: 10 | Type: fruit
1 - apples | Price: $50 | Quantity: 1 | Type: fruit
```

Item not found



Item ID# 1

Name

price

Quantity

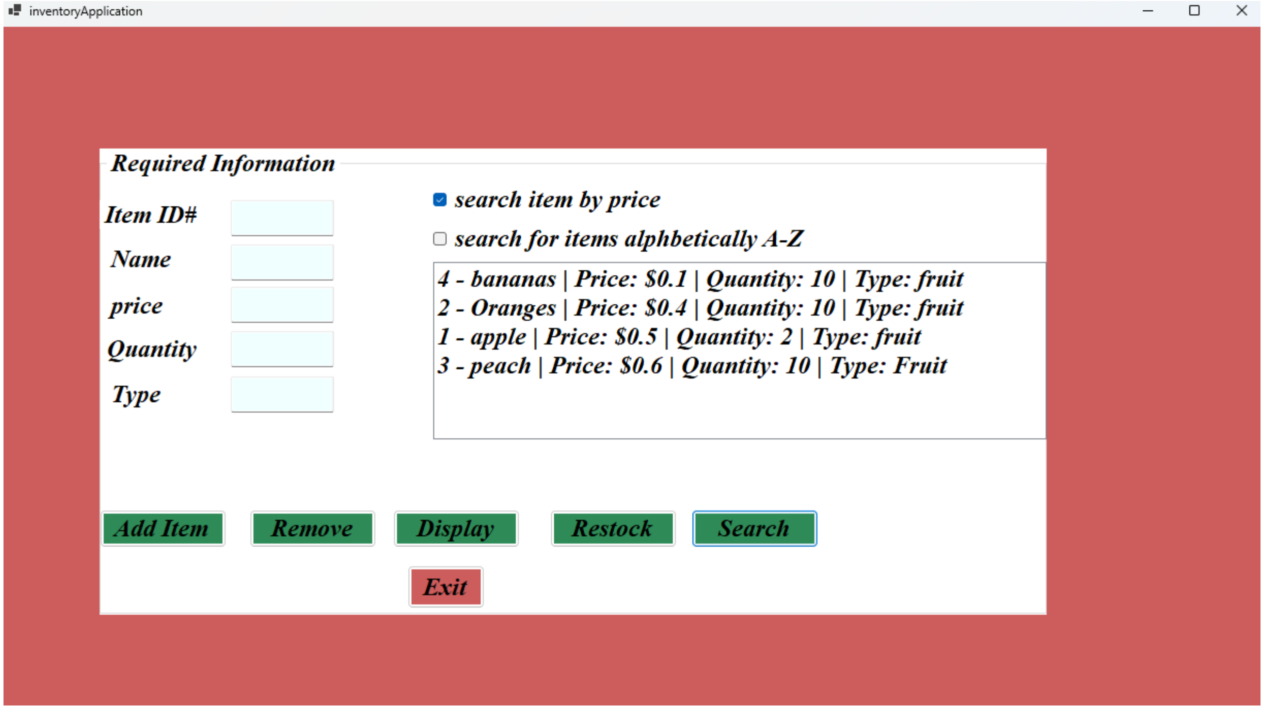
Type

□ search item by price

□ search for items alphbetically A-Z

```
4 - bananas | Price: $0.1 | Quantity: 10 | Type: fruit
2 - Oranges | Price: $0.4 | Quantity: 10 | Type: fruit
1 - apple | Price: $0.5 | Quantity: 2 | Type: fruit
3 - peach | Price: $0.6 | Quantity: 10 | Type: Fruit
```

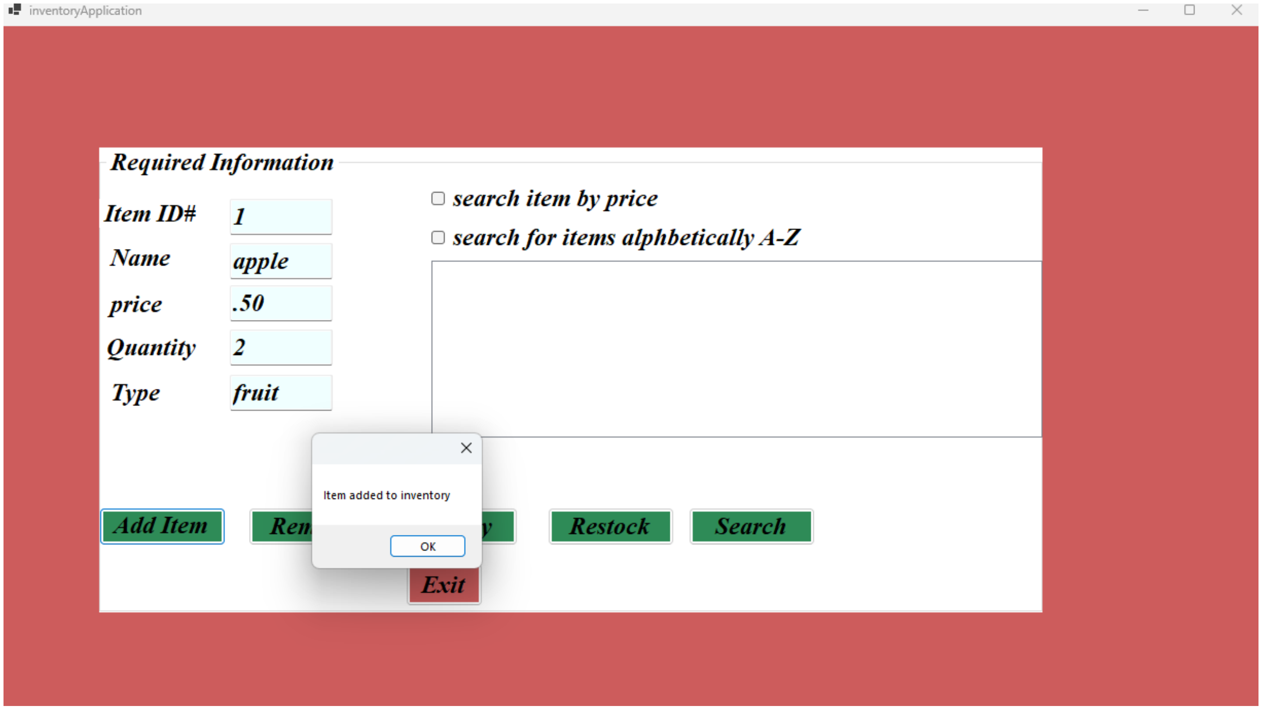
Item removed successfully



☑ search item by price

□ search for items alphbetically A-Z

```
4 - bananas | Price: $0.1 | Quantity: 10 | Type: fruit
2 - Oranges | Price: $0.4 | Quantity: 10 | Type: fruit
1 - apple | Price: $0.5 | Quantity: 2 | Type: fruit
3 - peach | Price: $0.6 | Quantity: 10 | Type: Fruit
```



Item ID# 1

Name apple

price .50

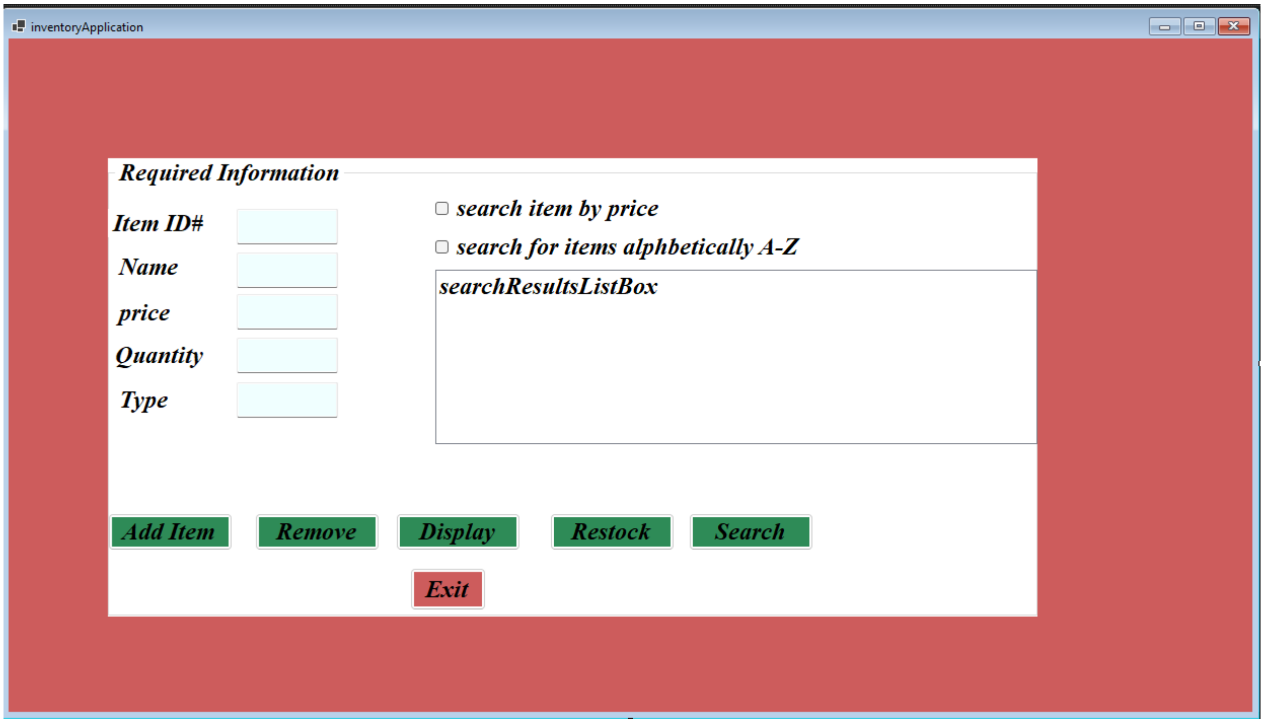
Quantity 2

Type fruit

□ search item by price

□ search for items alphbetically A-Z

Item added to inventory


inventoryApplication
Required Information
Item ID#
Name
price
Quantity
Type
□ search item by price
□ search for items alphbetically A-Z
searchResultsListBox
Add Item
Remove
Display
Restock
Search
Exit