PROGRAMMING III PROJECT DOCUMENT

Team members:

Liam Scalzulli (1947334)
Maxence Roy (1957042)

Description of the project:

We will develop the proposed topic #1: Inventory Tracker (renamed to 'Inventory Manager').

This app will let users keep track of a vast collection of items in their inventories. They will be able to add new items, sell existing items, save the current inventory state and load an existing inventory state.

Users will be able to keep track of the total value of their inventories alongside their total revenue accrued from previously sold items.

Users can also sort (ascending or descending) their inventory items by various properties such as ID, Name, Quantity, Cost and Optimal Quantity, in order to see items with the users preferred details in a specified order.

All of these features paired with a sleek dark themed UI with custom error messages!

Development approach:

This WPF application will keep track of items in a store. *Information **in bold** are additional features.*

In the main window, the user will see every item in the store and their quantities. (It will initially be empty)

- If an item is clicked, a "Details" window appears displaying the item's details (quantity, supplier, optimal quantity, location, category, **cost**). The user will be able to modify them.

- **A sort button will be present, alongside two combo boxes, one for choosing which field to sort by and the other to choose whether to sort in ascending or descending order (default ascending). When the sort button will be clicked, all items in the list will be sorted by the provided options.**

There will be a toolbar on top of the Main Window, providing most of the functionalities.

- There will be a button to create items. When clicked, a “Create” Window will appear, where the user will be able to enter all details about the item.

- **A sell button will allow the user to sell items from their inventory. When clicked, a “Sell” Window will appear, allowing the user to select available items and enter the amount they want to sell. Selling will increase the inventory’s revenue. The revenue will never decrease, even if the item is later deleted.**

- **A reset button will reset the inventory.**

- There will be a save and load button that stores the inventory properties **in a JSON file**.

Additionally, a “General Report” button will display a message box showing every item and their quantity, **cost per unit, and value**. Next to this, a “Quantity Analysis” will display items below and above optimal quantity.

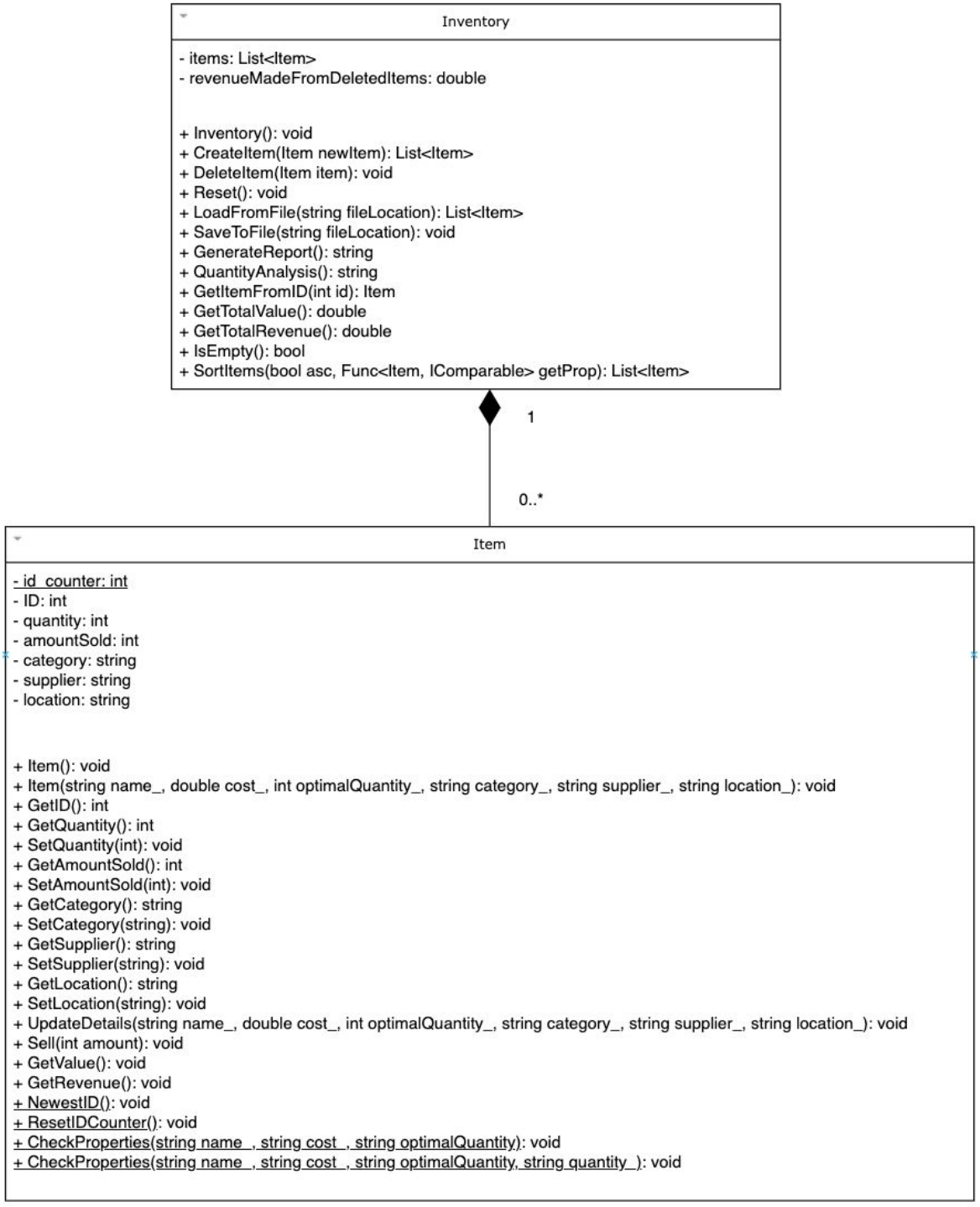
Under the two report buttons, the total value of the inventory (total cost of all items) will be displayed. The total revenue made from sold items will also be displayed.

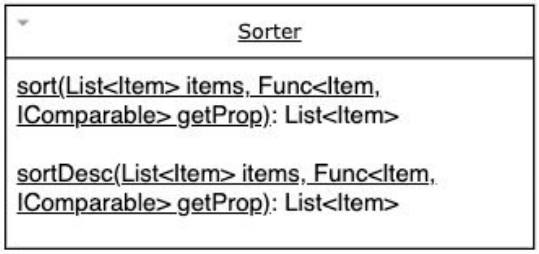
To keep the dark design and have more versatility, a custom message box will be implemented. It will be used for both error and warning messages. Warning messages will appear before any major irreversible action (e.g., resetting an unsaved inventory and deleting an item).

Finally, a special warning message box called the “Exit” message box will appear when the user tries to close the application with an unsaved inventory. The message box will allow the user to either save their work and quit, quit without saving, or go back to the application.

To achieve this project, we will split the classes and xaml files between the two of us. We will regularly talk to each other to ensure everything is going well. If we are both available at the same time, we will work together using the Live Share Session feature from Visual Studio, where we can work on the same code together. Otherwise, since we are using GitHub, we will always be able to access the other partner’s newest code.

Design for OOP classes (and their UML diagram):





Parts each member worked on:

Both of us will help each other if we need help.

- Create classes for Item and Inventory. (Mutually collaborated)
- Main Window XAML (Max)
- Create XAML (Max)
- Details XAML (Max)
- Sell XAML (Max)
- Create logic (Max)
- Details logic (Max)
- Sell logic (Liam)
- Delete button (Max)
- Load (Liam)
- Save (Liam)
- Add button (Max)
- Custom Message Box (Max)
- Exit Message Box (Max)
- Sorting + Sorter class (Liam)
- Report buttons (Liam)

Future works:

There are many features we could have added on top of what we did.

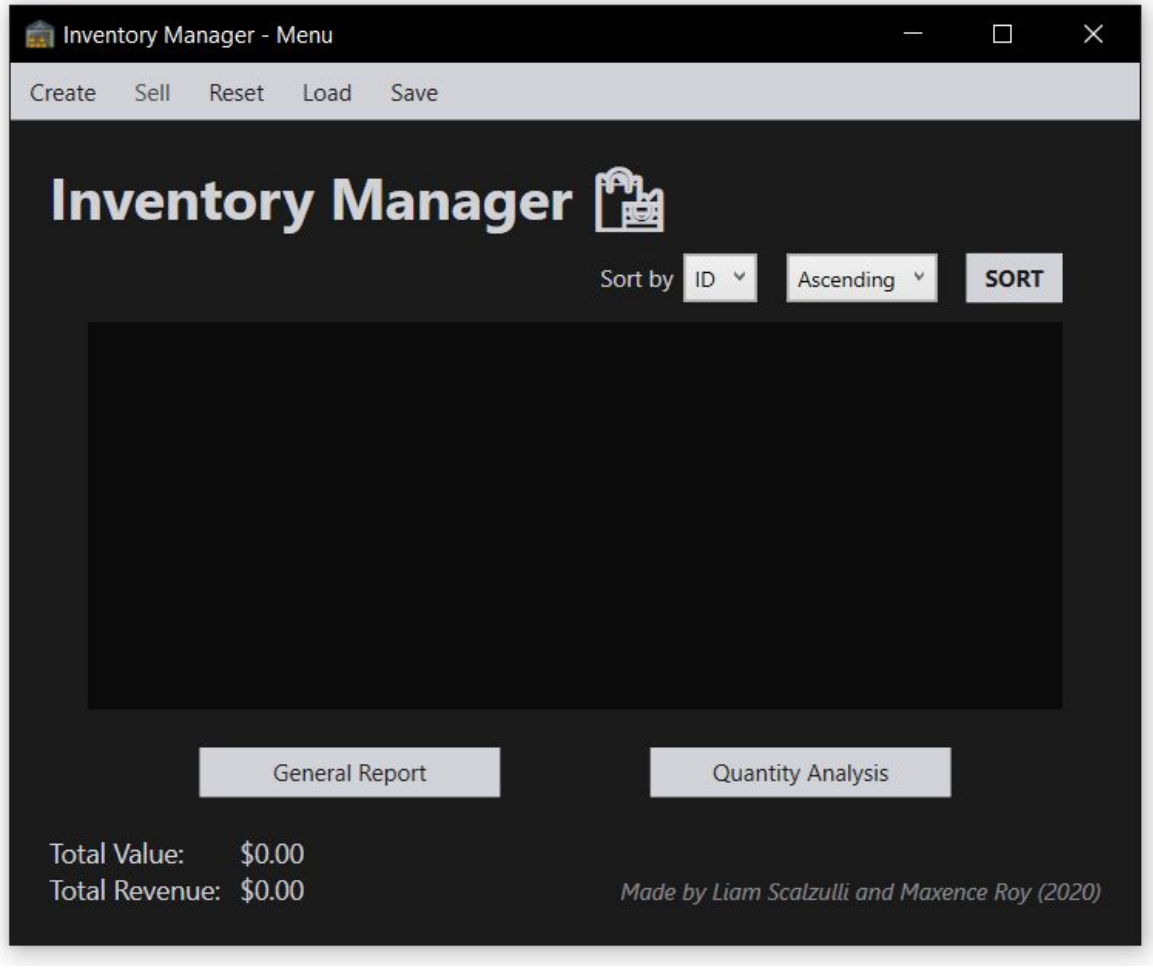
- **Images for Item:** Making the user be able to upload an image of their product, and then displaying that image every time that item is displayed (for instance, in the Main Window's item list) would make it easier to distinguish items between each other.

- **Save General Report and Quantity Analysis to a text file:** Allowing the user to save the report they generated into a .txt file would be useful for tracking items over time.

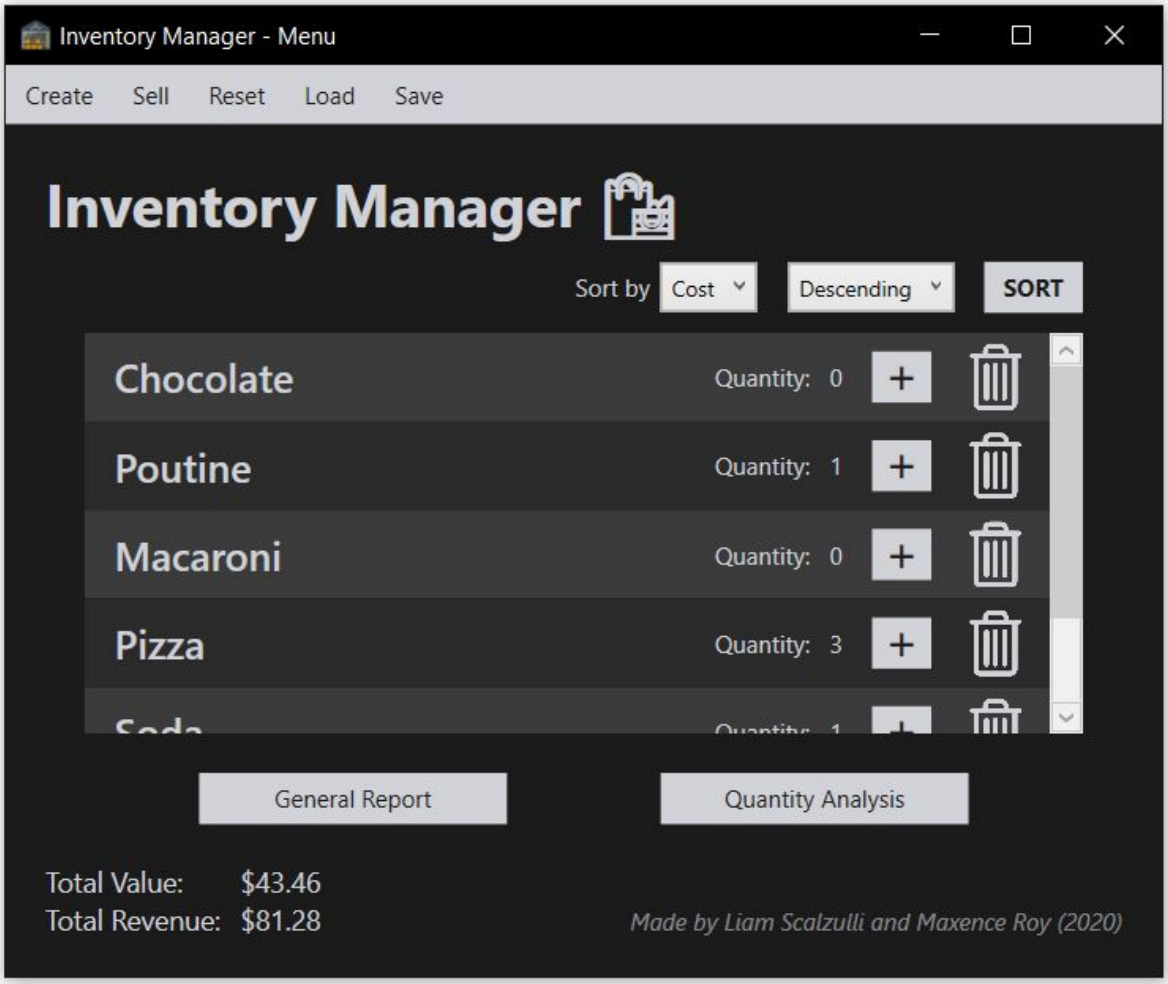
- **Host an actual database with accounts:** This would let different users use the application at the same time and be more convenient to load and save inventories.

- **Search Bar:** When an inventory has hundreds of items, it can be inconvenient to scroll to reach the desired item. A search bar would filter all items and only keep those that match the provided text.

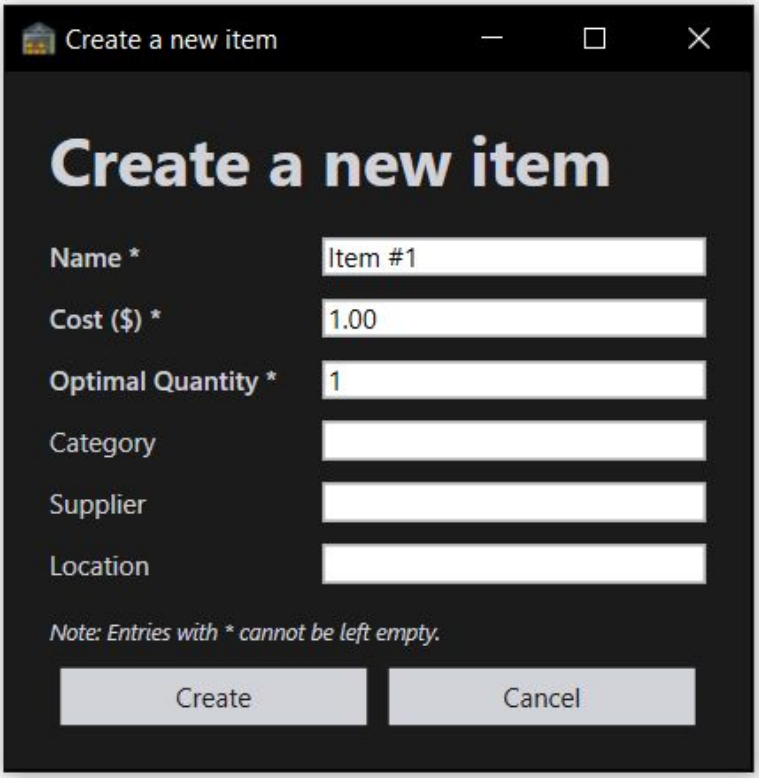
App Snapshots:



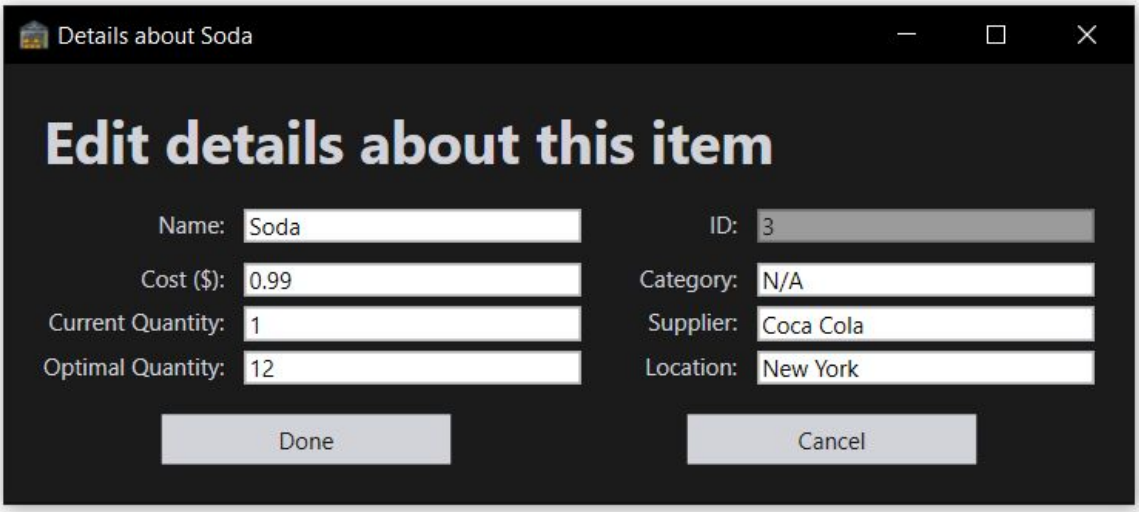
Appearance of the Main Window when the application is first launched.



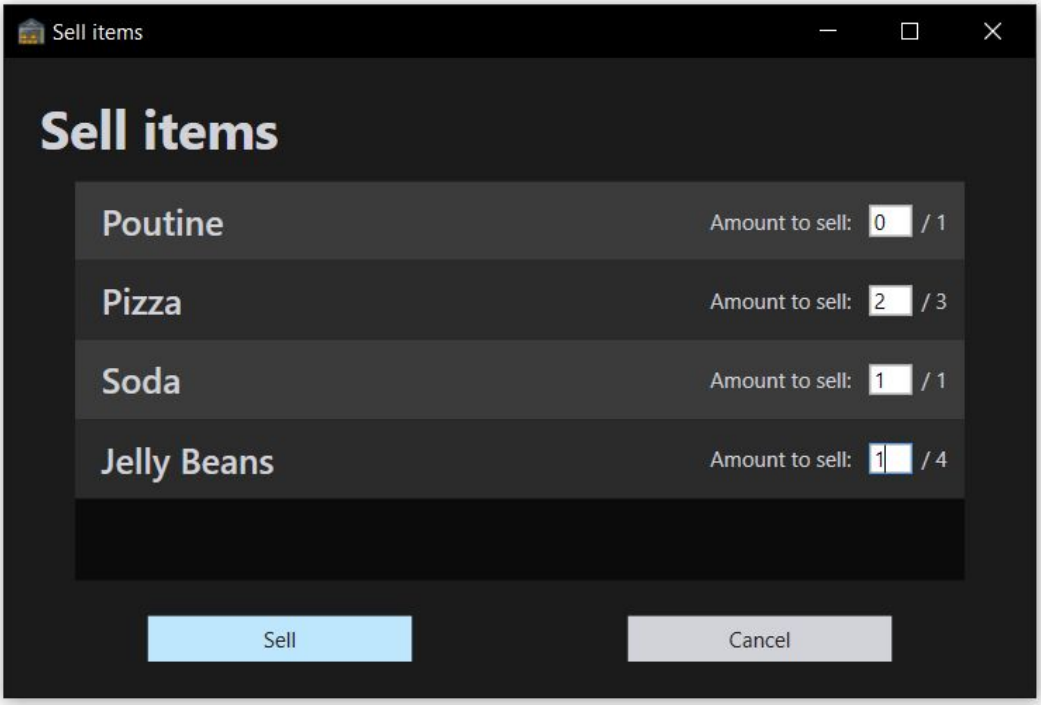
Appearance of the Main Window when inventory is filled with various items. Notice the combo boxes for sorting, add (+) buttons and delete (trash icon) buttons.



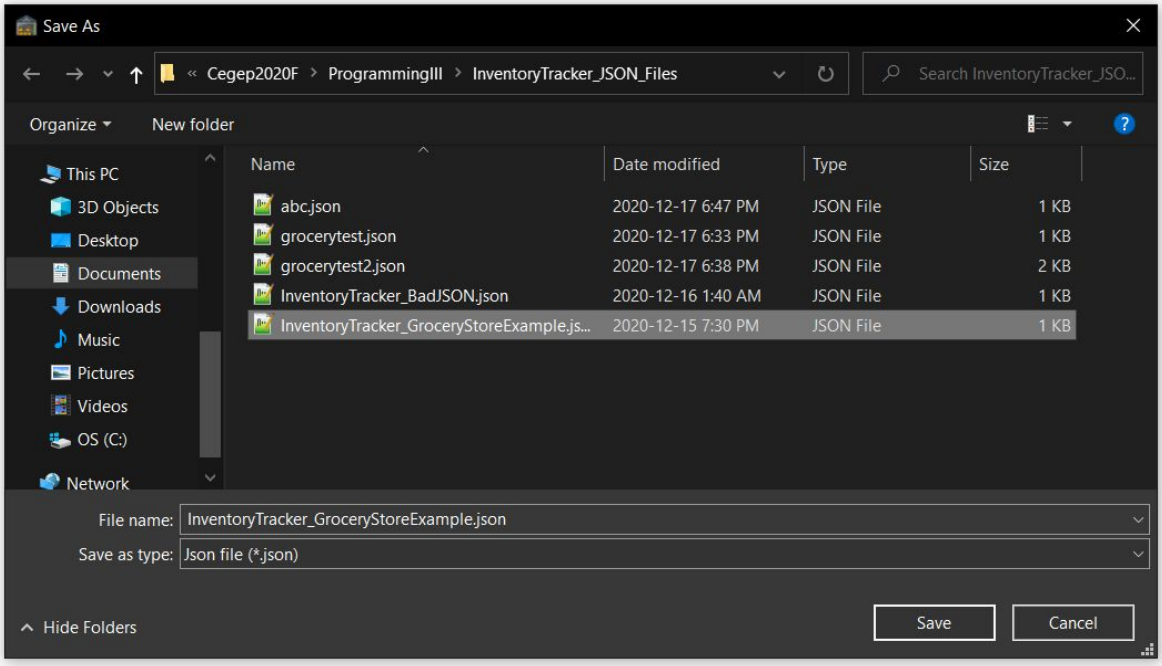
Appearance of the “Create” Window, which allows the user to create a new item.



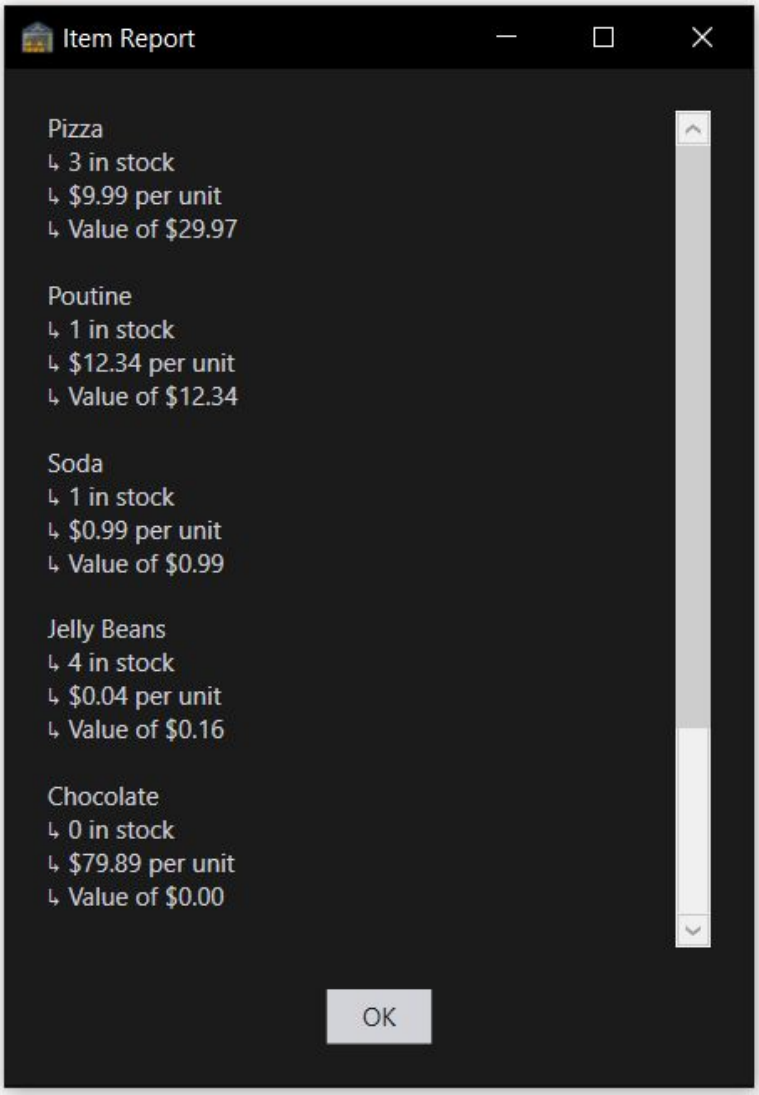
Appearance of the “Details” Window, which allows the user to edit the details of an existing item.



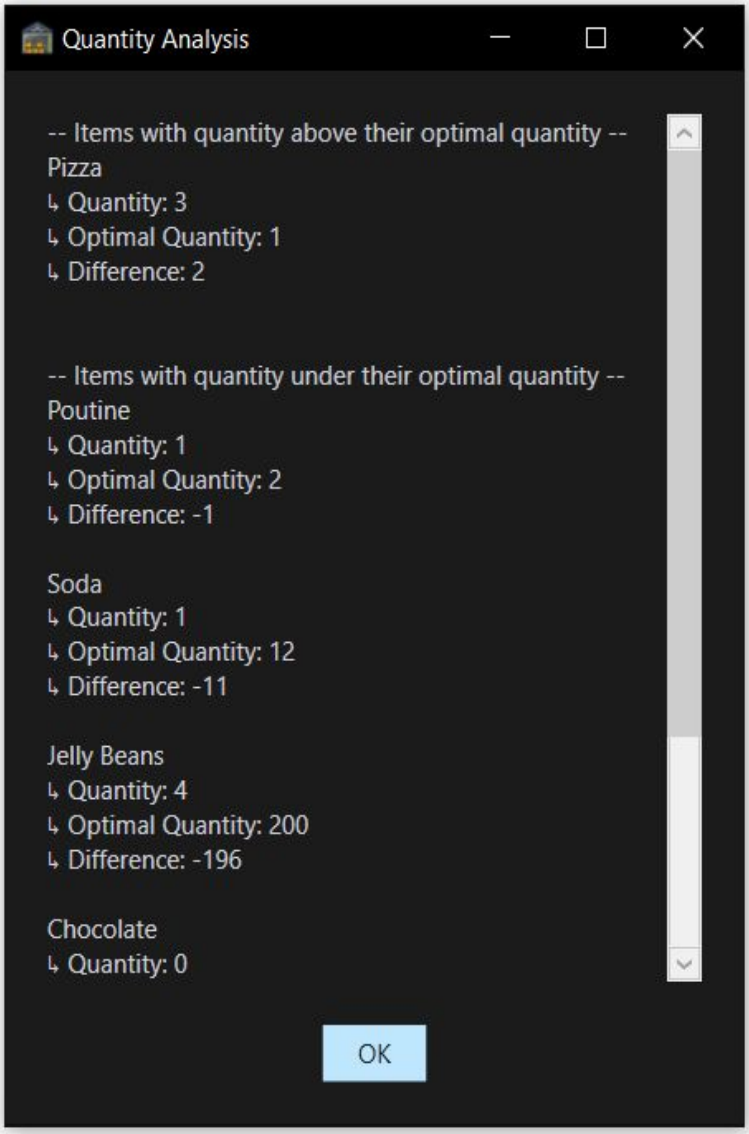
Appearance of the “Sell” Window, which allows the user to sell existing items and increase their total revenue.



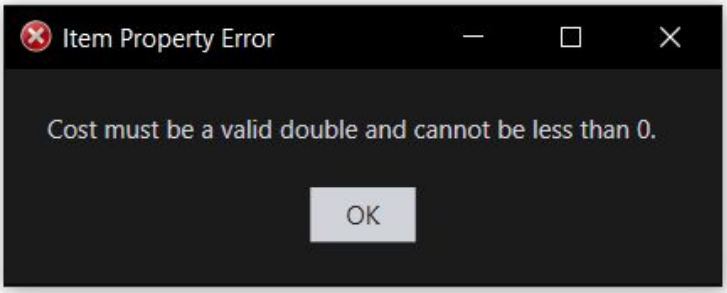
Appearance of the file explorer that appears when the user wants to save their inventory into a file.



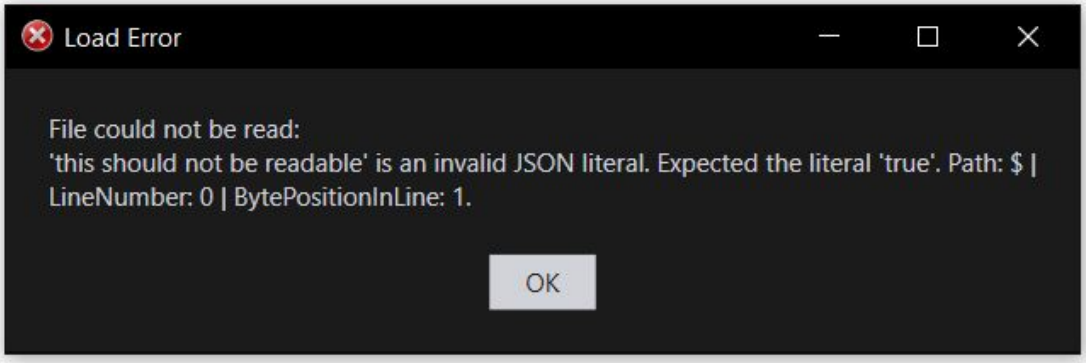
Appearance of a General Report, which lists every item with their quantity, cost per unit, and total value.



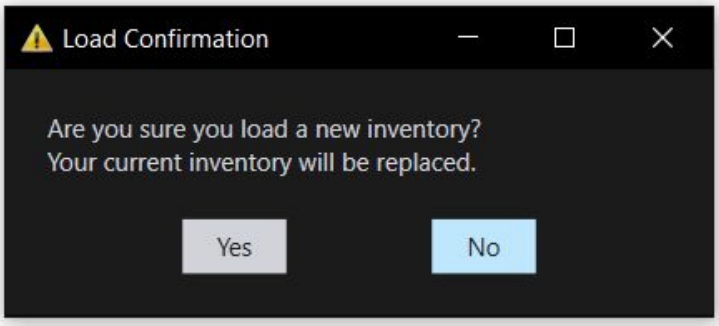
Appearance of a Quantity Analysis, which evaluates whether or not items are in stock above or below the optimal quantity.



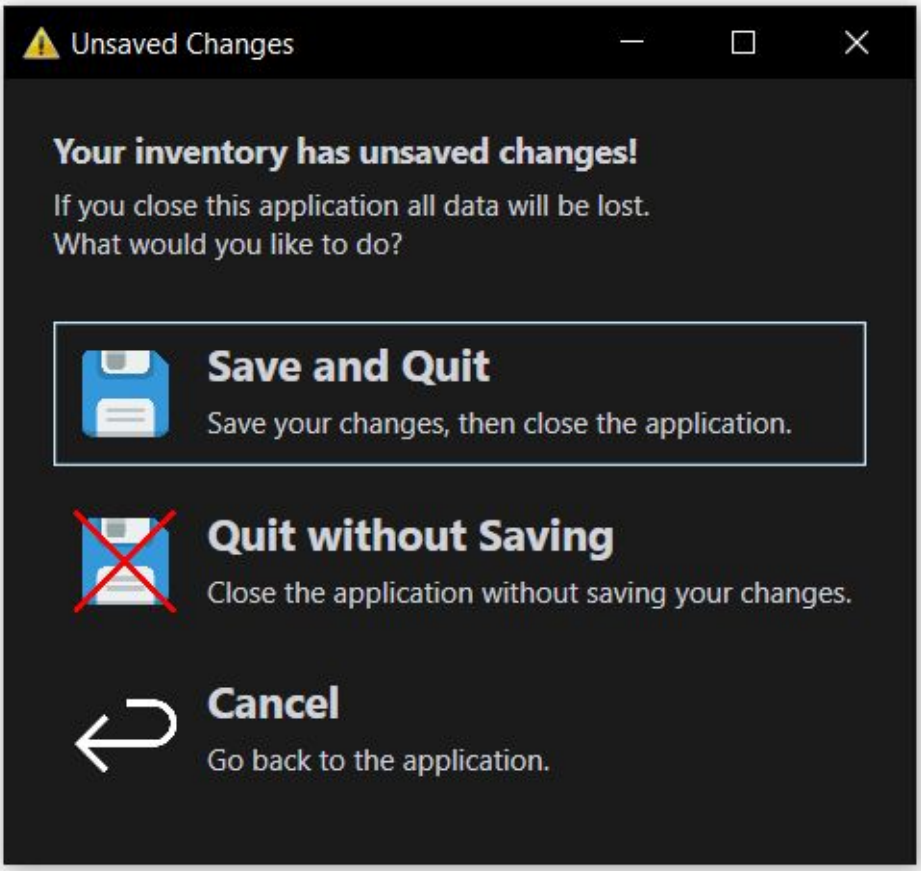
Appearance of an error message box. This example demonstrates an item property error where the user enters an invalid cost for an item they are creating.



Appearance of another error message box. This example demonstrates a load error when the application can't translate a provided JSON file into an inventory/



Appearance of a warning message box. This example occurs when the user tries to load a new inventory over their unsaved one.



Appearance of the exit message box. It appears when the user tries to close the application while having unsaved changes to their inventory.