

**FACULTY OF ENGINEERING, BUILT ENVIRONMENT AND INFORMATION TECHNOLOGY (FOEBEIT)**

**NDPeripherals Inventory Management System**

**Nandita Dhanda**

**Course Name:** Windows Programming

**Course Code:** TCS3294

**Lecturer’s Name:** Mr. Seow Soon Loong

**Due Date:** 14 August 2020

Table of Contents

[CHAPTER 1: INTRODUCTION AND REQUIREMENT SPECIFICATION 3](#_Toc48295084)

[1.1 Functional and Non-Functional Requirements 4](#_Toc48295085)

[CHAPTER 2: SYSTEM DESIGN 5](#_Toc48295086)

[2.1 Use Case Diagrams and Description 5](#_Toc48295087)

[2.2 Class Diagram 11](#_Toc48295088)

[CHAPTER 3: SOFTWARE DEVELOPMENT 13](#_Toc48295089)

[3.1 Login Window 13](#_Toc48295090)

[3.2 Inventory Management System Interface – Manage Inventory Tab 17](#_Toc48295091)

[3.3 Inventory Management System – Sales Transaction 37](#_Toc48295092)

[3.4 Inventory Management System– Generate Report 61](#_Toc48295093)

[3.5 SQL Database 67](#_Toc48295094)

[APPENDIX A: SOFTWARE INSTALLATION AND DEPLOYMENT 72](#_Toc48295095)

[APPENDIX B: USER MANUAL 73](#_Toc48295096)

[Appendix C: Source Code 74](#_Toc48295097)

[Section 1: Login Window Source Code – login.cs 74](#_Toc48295098)

[Section 2 – mainSystem.cs 76](#_Toc48295099)

[Section 3 – Item.cs 91](#_Toc48295100)

[Section 4 – Sales.cs 95](#_Toc48295101)

[Section 5 – inventory.cs 98](#_Toc48295102)

[Section 6 – Reports 98](#_Toc48295103)

# CHAPTER 1: INTRODUCTION AND REQUIREMENT SPECIFICATION

The aim of this project is to design a windows application using C# programming language of an inventory management system for ND Peripherals. ND Peripherals sells peripheral devices for computer systems divided into three main categories – input devices, output devices and storage devices.

The primary user of the system is the system admin. The Admin account is pre-programmed into the system. No new account can be created. The admin will be able to log into the system with the login credentials in order to access the features of the system.

The system will store information on the store’s inventory. The admin user will be able to add new items to the system, update the details of existing items on the system and delete items from the system. The system also includes a search feature which will allow the admin to search the inventory for specific items using the keywords in the item’s name.

The system will also allow the admin to initiate sales transactions and manage the cart. The admin will be able to select items to be sold from the list provided, enter the quantity of items to be purchased and add the items to the cart. Any items that the admin wishes to remove from the cart can also be removed. Upon adding all required items to the cart, the system will display the grand total of the amount that needs to be paid. The admin will be able to enter the amount tendered and checkout the cart to complete the purchase. Data of each sales transaction is recorded for future reference.

Finally, the admin will be able to generate reports on the system related to the inventory and sales data. An inventory report will list out all the available items in the inventory and its associated data along with the value of the total inventory and total stock available. The sales report will display data on sales including date of sale, sales value and number of items purchased.

## 1.1 Functional and Non-Functional Requirements

**Functional Requirements:**

|  |  |
| --- | --- |
| **FR Number** | **Description** |
| FR1 | System should only be accessible by Admin user through log in feature |
| FR2 | Search item – Admin should be able to search for item in inventory list |
| FR3 | Manage Inventory - Admin should be able to manage inventory. Including – add new items to inventory, update details of existing item and delete items from inventory |
| FR4 | Admin should be able to select items to purchase and add them to cart |
| FR5 | Admin should be able to remove items from cart |
| FR6 | Checkout - Admin should be able to check out items from cart and make final purchase |
| FR7 | System should update records of stock available, number of items sold, and total stock and inventory value after each transaction. |
| FR8 | Generate report - System should be able to generate reports using inventory and sales data |
| FR9 | System should log out user if log out button is clicked or if application window is closed. |

**Non-Functional Requirements:**

|  |  |
| --- | --- |
| **NFR Number** | **Description** |
| NFR1 | System should be in the form of a windows application |
| NFR2 | System should be able to run on Windows operating system on Desktop or Laptop systems |
| NFR3 | System should be able to launch within 2 seconds |
|  | System user should not be able to modify log in details or user account details. |
| NFR4 | New or modified data in database should be updated within 2 seconds for all users accessing the system |
| NFR5 | System must be secure allowing only authorized users to access the data |
| NFR6 | System should autogenerate report weekly |
| NFR7 | All transactions and currency value used in application must be in Ringgit Malaysia |

# CHAPTER 2: SYSTEM DESIGN

## 2.1 Use Case Diagrams and Description

A close up of a map

Description automatically generated

Figure 1: Inventory Management System Use Case Diagram

**Use Case Descriptions\***

\*Please note: industrial standards for use case descriptions require them to follow single line spacing which is why single-line spacing of 1.5 is not used in this part of the report.

|  |  |
| --- | --- |
| **Use Case 1** | **Login** |
| **Description** | Admin can log in to access the system |
| **Actor** | Admin |
| **Pre-condition** | System is installed on user’s machine and has log in credentials to Admin dashboard |
| **Trigger** | Admin wants to sign into the system |
| **Main Success Scenario** | 1. Launch application 2. Screen displays login window form 3. Admin fills out username and password in the form 4. Admin clicks on login button 5. System validates login credentials 6. System opens Dashboard screen |
| **Alternative Flow** | 1a) IF login credentials are invalid then system displays error message in a message box. Log in attempt fails |
| **Post Condition** | Admin successfully logs in and is able to access system dashboard |

|  |  |
| --- | --- |
| **Use Case 2** | **Search Item** |
| **Description** | Admin can search for item in search field |
| **Actor** | Admin |
| **Pre-condition** | Admin is logged into the system |
| **Trigger** | Admin wants to search for an item in the inventory |
| **Main Success Scenario** | Search feature is available in Manage Inventory tab and Sales Transaction Tab.  **Manage Inventory:**   1. Click on manage inventory tab 2. Enter search criteria or keywords in search field 3. System will display search results in data table   **Sales Transaction:**   1. Click on sales transaction tab 2. Enter search criteria or keywords in search field 3. System will display search results in list view table |
| **Alternative Flow** | 2a) IF results are found matching keywords entered, system will display empty data table / list view |
| **Post Condition** | System displays search results |

|  |  |
| --- | --- |
| **Use Case 3** | **Add Item** |
| **Description** | Add new item to inventory |
| **Actor** | Admin |
| **Pre-condition** | Item does not exist in inventory |
| **Trigger** | Admin wants add new item |
| **Main Success Scenario** | 1. Admin clicks on manage inventory tab 2. Admin fills out all fields in “Add Item” panel 3. Admin clicks on “Add” button 4. System will validate input of fields 5. New item will be added to items database table in inventory |
| **Alternative Flow** | 3a) **Empty Fields**   1. If there are any empty textbox fields, system will display an error message 2. Item will not be added to inventory 3. Admin will be required to fill out all fields and try again   3b) **Invalid data type entered**   1. If invalid data is entered in the textbox fields, system will display an error message 2. Item will not be added to inventory 3. Admin will be required to enter valid data in the textbox fields and try again |
| **Post Condition** | Data table will refresh and display new item added to inventory |

|  |  |
| --- | --- |
| **Use Case 4** | **Update Item** |
| **Description** | Update item in inventory |
| **Actor** | Admin |
| **Pre-condition** | Item already exists in database |
| **Trigger** | Admin wants update an item details |
| **Main Success Scenario** | 1. Admin clicks on manage inventory tab 2. System displays list of inventory items in data table 3. Admin double clicks on item that needs to be updated 4. Item details will be populated in textbox fields in “Edit Item” panel 5. Update item details 6. Admin clicks on “Update” button 7. System validates details entered 8. Item details are updated on database |
| **Alternative Flow** | 4a) **Empty Fields**   1. If there are any empty textbox fields, system will display an error message 2. Item details will not be updated 3. Admin will be required to fill out all fields and try again   4b) **Invalid data type entered**   1. If invalid data is entered in the textbox fields, system will display an error message 2. Item will not be updated 3. Admin will be required to enter valid data in the textbox fields and try again |
| **Post Condition** | Inventory data table is refreshed and displays updated details of item |

|  |  |
| --- | --- |
| **Use Case 5** | **Delete Item** |
| **Description** | Delete item from inventory |
| **Actor** | Admin |
| **Pre-condition** | Item exists in database |
| **Trigger** | Admin wants delete an item from the inventory |
| **Main Success Scenario** | 1. Click on manage inventory tab 2. Screen displays list of inventory items in data table 3. Admin double-clicks on item that needs to be deleted 4. Item details will be populated in textbox fields in “Edit Item” panel 5. Admin clicks on “Delete” button 6. Item is deleted from database |
| **Alternative Flow** |  |
| **Post Condition** | Item is successfully deleted. Inventory data table is refreshed and displays updated list of items in database |

|  |  |
| --- | --- |
| **Use Case 6** | **Make Sales Transaction** |
| **Description** | Initiate new sales transaction |
| **Actor** | Admin |
| **Pre-condition** | Admin is logged into the system and on the “Sales Transaction” tab |
| **Trigger** | Admin wants to make a new sales transaction |
| **Main Success Scenario** | 1. Admin clicks on “Sales Transaction” tab 2. System displays sales transaction interface 3. Admin clicks on item in list view table 4. System displays details of number of “items in stock” and “selling price” in respective fields 5. Admin enters quantity in “quantity” field 6. System displays final price of item in “final price” field 7. Admin clicks “Add To Cart” button 8. Item is added to cart in purchase list 9. Admin repeats from step from step 3 – 8 until satisfied |
| **Included Points** | US 8 Checkout |
| **Alternate Flow** | 1. IF invalid data value is entered in “Quantity” field, system displays an error message 2. Admin is required to re-enter quantity using correct data type |
| **Post Condition** | New sales transaction is initiated. Items are added to cart (purchase list) |

|  |  |
| --- | --- |
| **Use Case 7** | **Remove From Cart** |
| **Description** | Remove item from cart (purchase list) |
| **Actor** | Admin |
| **Pre-condition** | Item has been added to cart (purchase list) |
| **Trigger** | Admin wants to delete an item from cart (purchase list) |
| **Main Success Scenario** | 1. Admin selects item from purchase list in cart 2. Admin clicks on “Remove From Cart” button 3. Item is deleted (removed) from purchase list in cart |
| **Alternate Flow** |  |
| **Post Condition** | Item is removed from cart (purchase list) |

|  |  |
| --- | --- |
| **Use Case 8** | **Checkout (Included Use Case)** |
| **Description** | Checkout items from cart and complete sales transaction |
| **Actor** | Admin |
| **Pre-condition** | Items have been added to purchase list in cart |
| **Trigger** | Admin wants to checkout items from cart and complete sales transaction |
| **Main Success Scenario** | 1. System displays items added to purchase list 2. System displays “Total” amount that needs to be paid 3. Admin enters tendered amount in “Tendered” textbox field 4. System displays balance to be returned 5. Admin clicks on “checkout” button 6. System validates details 7. System displays sale summary and success message |
| **Alternate Flow** | **8a) Tendered amount not entered**   1. IF Admin does not entered tendered amount, System does not process checkout function 2. System displays an error message 3. Admin is required to entered tendered amount   **8b) Tendered amount is less than Total amount**   1. IF tendered amount is less than total amount, transaction fails 2. System displays error message 3. Admin is required to tender correct amount and try again. |
| **Post Condition** | Sales transaction is complete. System refreshes and clears items from cart. Stock values are updated in system |

|  |  |
| --- | --- |
| **Use Case 9** | **Generate Report** |
| **Description** | Generate weekly reports |
| **Actor** | Admin |
| **Pre-condition** | Sales data and Inventory data exists on System |
| **Trigger** | Admin wants to generate reports |
| **Main Success Scenario** | 1. Admin clicks on “Generate Reports” Tab 2. Admin selects the report to view    1. Inventory report       1. System displays inventory report with details on items available in stock, inventory value and total stock    2. Sales report       1. System displays sales report with sales data |
| **Alternate Flow** |  |
| **Post Condition** | System displays report |

|  |  |
| --- | --- |
| **Use Case 10** | **Logout** |
| **Description** | Log out from system |
| **Actor** | Admin |
| **Pre-condition** | Admin is logged into system |
| **Trigger** | Admin wants to log out of system |
| **Main Success Scenario** | 1. Admin clicks on “Logout” button 2. Admin is logged out of system 3. System displays log in window |
| **Alternate Flow** |  |
| **Post Condition** | System logs admin out and displays log in window |

|  |  |
| --- | --- |
| **Use Case 11** | **Exit Application** |
| **Description** | Exit system application |
| **Actor** | Admin |
| **Pre-condition** | Application has been launched |
| **Trigger** | Admin wants to exit from the system application |
| **Main Success Scenario** | 1. Admin clicks on “Exit” button 2. System closes all open windows of application 3. System exits user from the application. |
| **Alternate Flow** |  |
| **Post Condition** | System is closed |

## 2.2 Class Diagram

Two class diagrams were drawn in order to design and build the system. **Class diagram A** was designed during the initial stages of the software development after analyzing the system’s requirements and after evaluating the Use Cases. **Class Diagram B** was designed after the system was built based upon the actual classes, methods and associations in the system. Both the diagrams have been presented below.

**Class Diagram A:**

A close up of a map

Description automatically generated

Class Diagram A

**Class Diagram B:**

A close up of text on a white background

Description automatically generated

# CHAPTER 3: SOFTWARE DEVELOPMENT

Please note: the detailed source code of each page and class has been provided in **Appendix C.** This section of the report will only show snippets of the code or method being described

## 3.1 Login Window

For the screenshots of the entire source code of the login.cs class, please refer to **Appendix C Section 1**

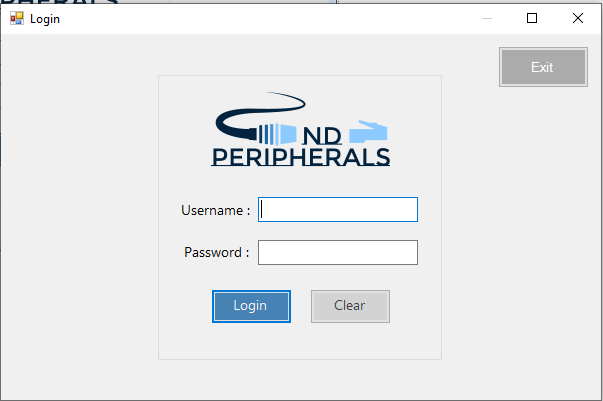


Figure : Log in window user interface

Figure 2 represents the graphical user interface of the log in window and Figure 3 refers to how the interface appears when the fields are filled. When system is launched, the log in window is the first to appear. The user is only able to access the system if they have the correct login credentials. The log in screen has the option to login, clear fields and exit



Figure : Log In Window - User Experience

**Login Button:**

Upon clicking on the login button, the system will validate the credentials entered. Figure 4. Demonstrates the source code of the event triggered when the log in button is clicked. As can be observed from the screenshot of the code, the system first establishes a connection with the database and matches the credentials entered in the username and password fields with the data in the database. If the credentials match, the system allows the user to log in by creating an object of the main inventory and opening it using the “.Show()” method. Subsequently, the system clears the data initially entered into the log in form and hides the log in window.

If the form is submitted without filling out the log in credentials or with invalid credentials, an error message will display on the screen, refer to Figure 5, and any data entered in the username and password fields will be cleared.

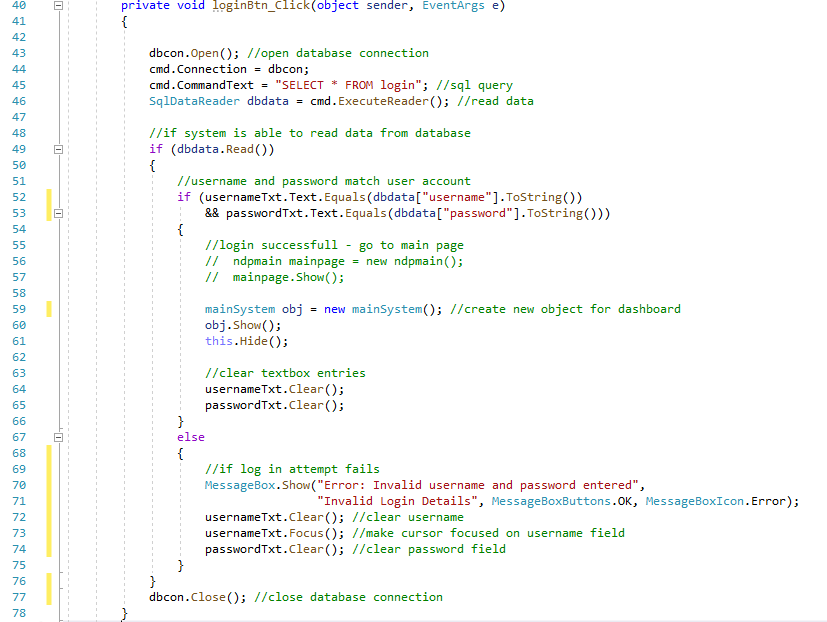


Figure : Login Button On Click - Source Code



Figure : Log In Window - Error Message

**Clear Button**



Figure : Clear Button

Figure 6 shows the clear button which can be seen on the login window interface. Upon clicking on this button, any values entered into the username and password textboxes will be removed. Please refer to Figure 7. For the source code of the event triggered when the clear button is clicked.

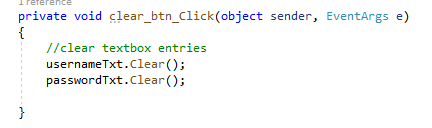


Figure : Clear Button Source Code

**Exit Button**



Figure : Exit Button

Figure 8 represents the exit button which is can be seen on the log in interface. This button is also placed on the windows of the main interface to allow the user to exit the application at any given time. The reason why a designated button has been designed and implemented to allow the user to exit the system is because the upon logging in, the windows application only hides the log in window. It cannot close it because it is the first window that the system opens and closing it would exit the user from the system. This means that even if the user exits the system after logging in using the “cross” button on their window, the system will keep running in the background. To address this issue, a designated exit button has been implemented into the system which will exit the user completely from the system and close the hidden login window as well.

When this button is clicked, the system will first show a confirmation dialog box to confirm whether the user really wants to exit the application or not which can be seen in Figure 9. For the source code please refer to Figure 10.



Figure : Exit Confirmation

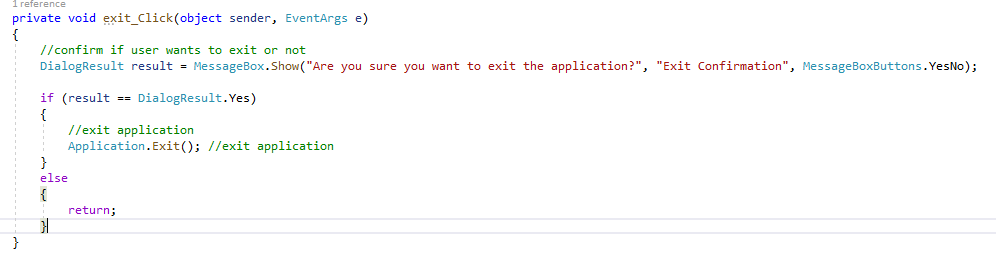


Figure : Exit Source Code

## 3.2 Inventory Management System Interface – Manage Inventory Tab

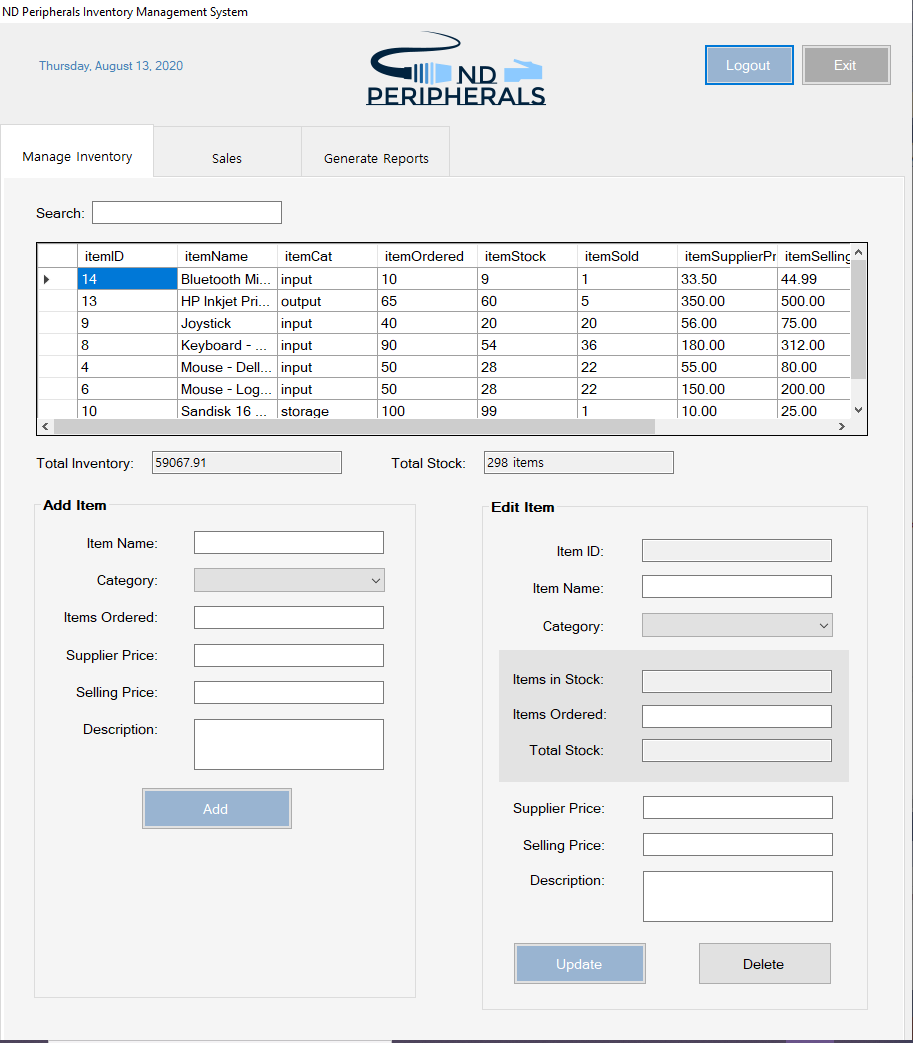


Figure : Main System Interface - Manage Inventory Tab Selected

The system has been designed using menu tabs. Figure 11 represents the interface of the system’s window that will appear after the user logs into the system. By default, the “Manage Inventory Tab” will be selected. For the entire source code of the system, please refer to **Appendix C.** The Manage Inventory tab allows the user to search the inventory for items, view total inventory value, view total stock available, add new items, update existing items or delete items from the inventory.

**3.2.1. View Items**

The user is able to view the items in the inventory and its associated details in a data-grid table in the Manage Inventory tab. The table displays the item ID, item name, category, number of items ordered, number of items in stock, number of items sold, supplier price, selling price of the store, item’s stock value and description of the item in this table. Figure 12 and Figure 13 represent the item table.

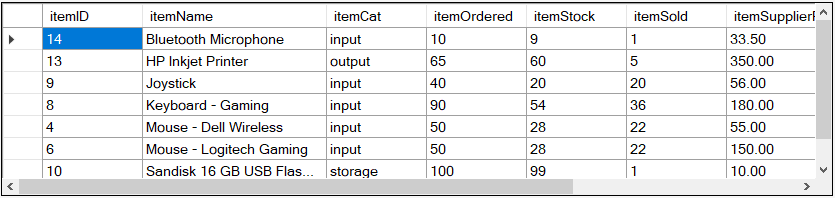


Figure : Item Table - Data Grid 1

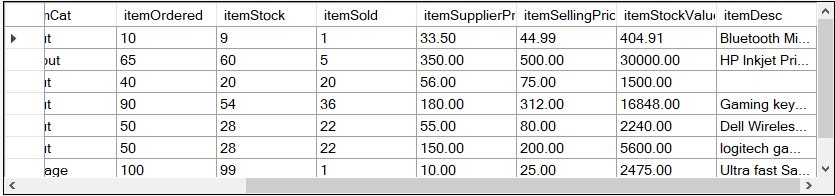


Figure : Item Table - Data Grid 2

In order to display the item datagrid table, a displayData( ) method was written and called on window load. Figure 14 and Figure 15 represent the source code of the method and method call.

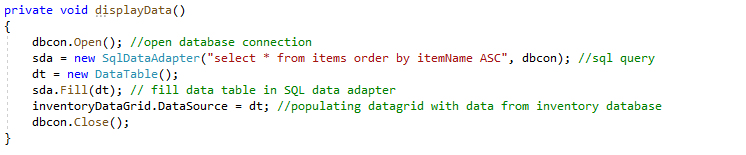


Figure : method to display item inventory data

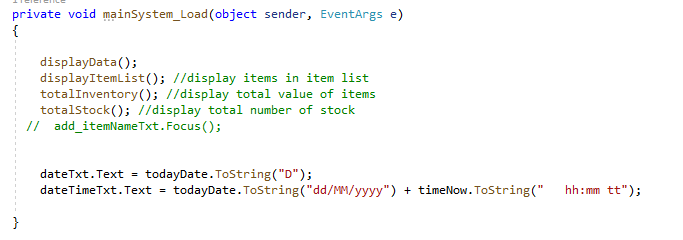


Figure : displayData method call

**3.2.2 Search Item**

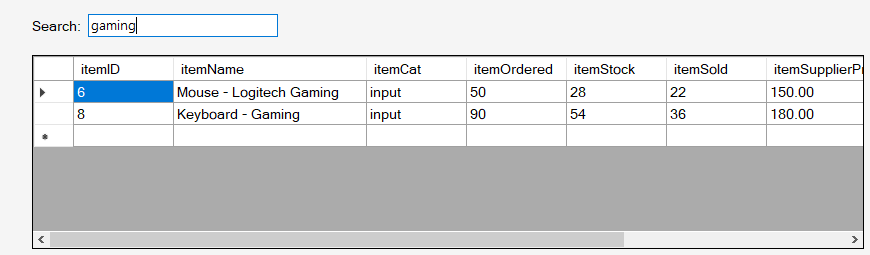


Figure : Search For Item in Inventory

The user is able to search for an item in the inventory by typing the search term in the search field. The system will return the results that match the text entered in the search field. Figure 16 represents how the search field works and Figure 17 shows the source code of the search method and method call

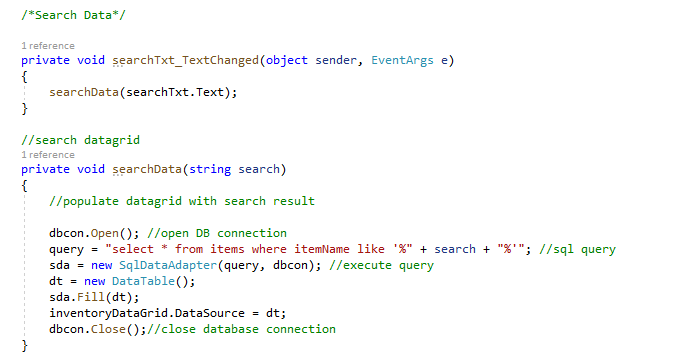


Figure : Search Method and Method Call

**3.2.3 Total Inventory and Total Stock**



Figure : Total Inventory Value and Total Stock Value

The system displays the Total Inventory value and the Total Stock value in the fields that can be seen in Figure 18. The total inventory is the total selling value of all of the items currently in the store. The stock value is the sum of the number of pieces of all of the items in the store.

In order to calculate the total inventory, first the total selling value of each item was calculated and stored in the database in the itemStockValue field using the formula :

Items in stock \* selling price

After that the sum of all the total selling value was calculated to get the total inventory.

The source code can be seen in Figure 19 and Figure 20.

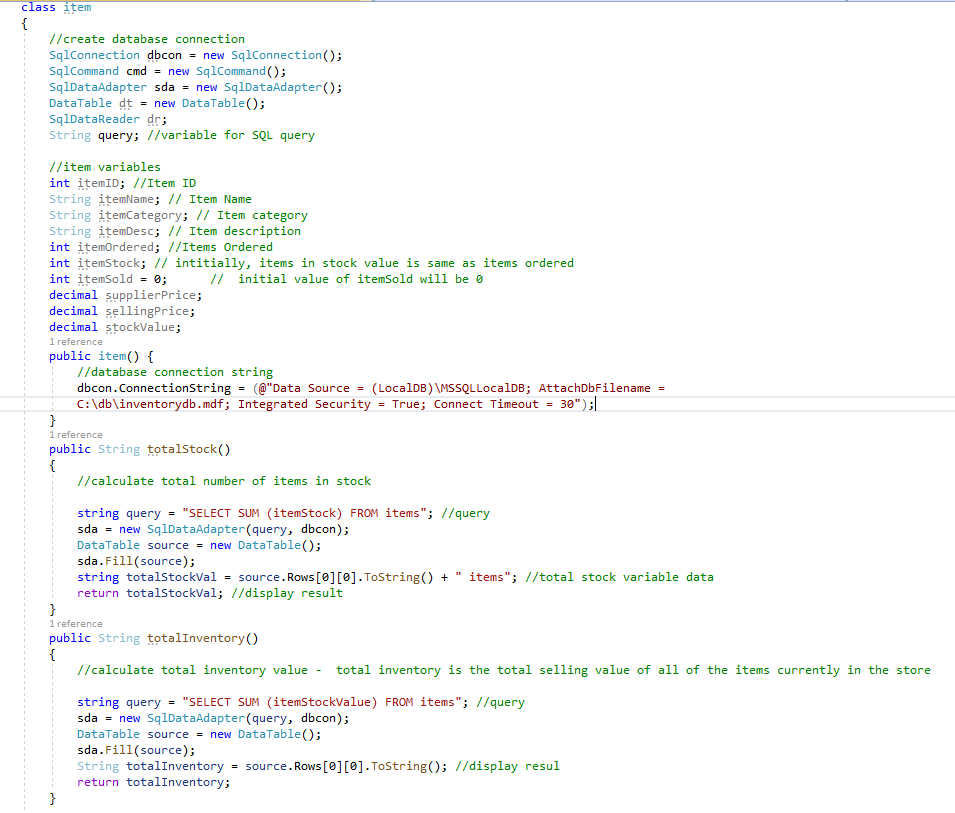


Figure : Total Inventory and Total Stock methods in item class

The method used to calculate the total inventory value and total stock value have been declared in the item class as can be seen in figure 19. An object of the item class is created in the mainSystem.cs and the respective methods are called and used to display the respective values.



Figure : an object of the item class

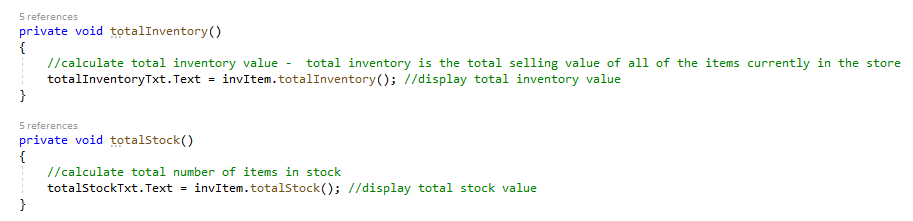


Figure : totalInventory and totalStock methods called in mainSystem.cs windows form class as a method of an object



Figure : calling the methods wherever needed

**3.2.4 Add Item**

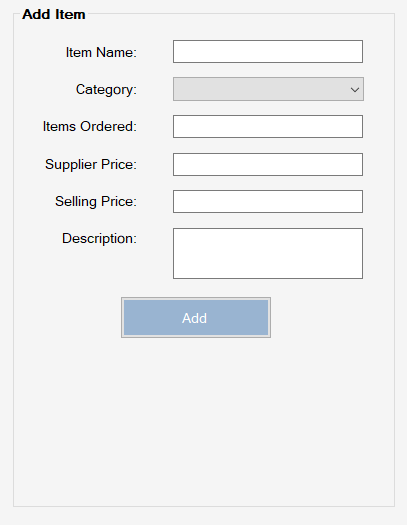


Figure : Add Item Panel

Figure 23 represents the add item panel which the admin user can use to add a new item to the inventory. It is required for all the fields to be filled out with correct data type to add a new item to the inventory.

The method to add a new item to the database (Figure 24) has been declared in the **item** class and when the Add button click event is triggered, it is called in the **mainSystem** class using an instance of the item class (invite object) that was declared previously in the mainSystem class (Figure 25).



Figure : addItem method declared in item class

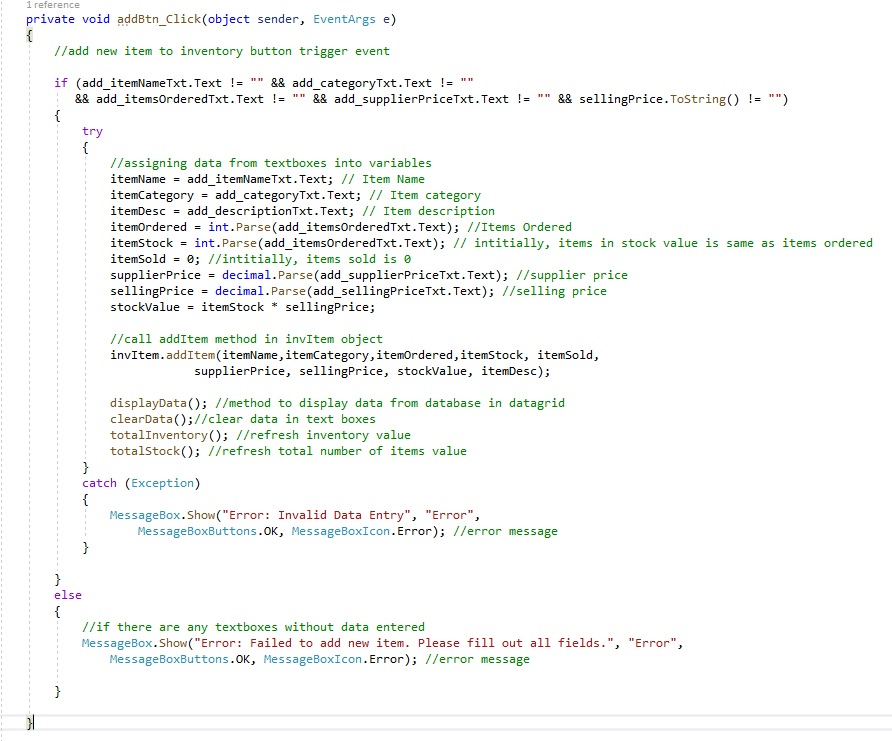


Figure : addBtn\_Click method - calling addItem( ) method in invItem object

*As it can be seen in Figure 25, the initial value of itemSold (number of items sold) is set to 0 and the number of items ordered is equal to the number of items in stock.*

**Validations**

Figure 24 and figure 25 also show the validations used to prevent invalid data or empty forms to be submitted. If the user tries to do so, the system will display the error messages shown in Figure 26 for an empty form submitted and Figure 27 for invalid data type.



Figure : Empty fields error message

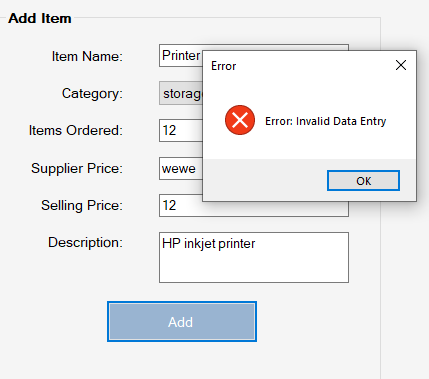


Figure : invalid data entry error

**Successful Attempt of Adding New Item**

If the user enters the correct data for the item and there are no other errors, the system displays a success message after successfully adding the item to the database as can be seen Figure 28.

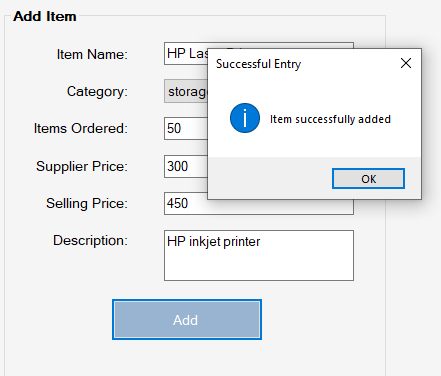


Figure : Item Added Success Message

After adding the item, the inventory datagrid displays the newly added item in the list and it is automatically sorted according to item name in ascending order as can be seen in Figure 29 with initial value of itemSold set to 0 and the value of itemOrdered = itemStock (item in stock is equal to the items ordered when an item is initially added to the inventory).

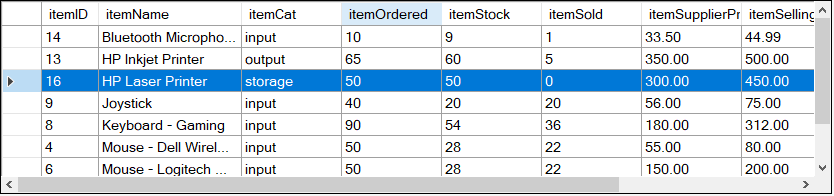


Figure : Item added to inventory item datagrid

Finally, the total inventory value and total stock value are recalculated and updated on the system UI by calling of the totalInventory( ) and totalStock( ) methods as can be seen in Figure 30.



Figure : Updated total inventory and total stock values

**3.2.5. Update Item**

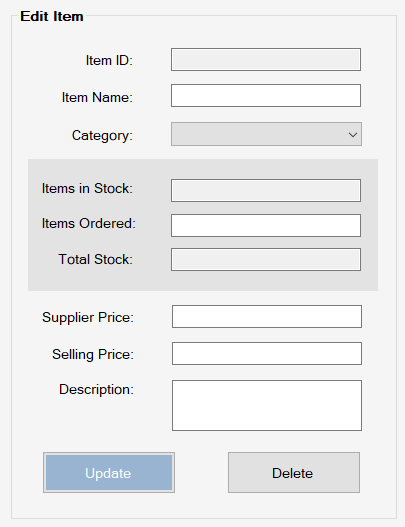


Figure :Edit Item Panel

Figure 31 shows the Edit panel which allows the user to update or delete a selected item. In order to update the details of and item or select an item to delete, the user is required to double click on the item from the datagrid item. After double clicking on the item, the editable details of the item will populate in the respective fields in the Edit Item panel. The source code that demonstrates the double click event method can be seen in Figure 32.

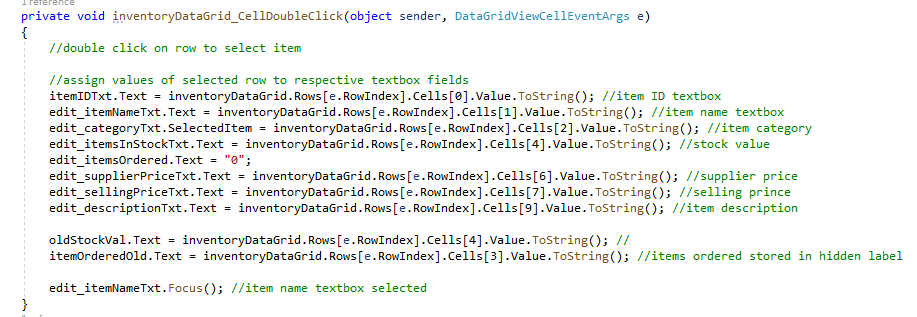


Figure : InventoryDataGrid\_CellDoubleClick event method

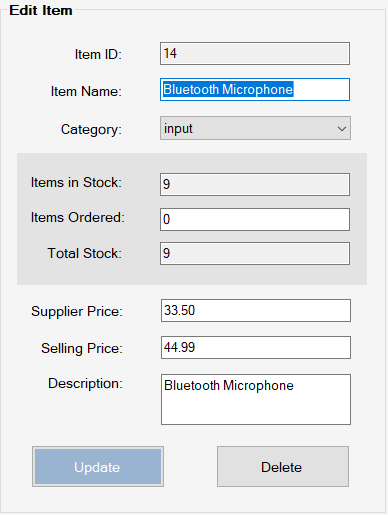


Figure : Item details populated in Edit Item panel

Figure 33 shows the editable details populated in the respective fields. Item ID, Items in Stock and Total Stock are read-only fields and the user cannot edit the details in these fields.

Items in Stock field displays the number of items currently in stock. After the user enters the number of items ordered, the figure will be added to the Items in Stock value and the result will display in the Total Stock field. This is to show what the total stock will be after new items are added to the stock (Figure 34)

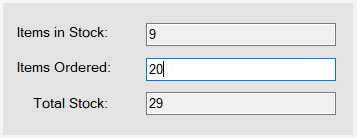


Figure : Stock Values Updated

Figure 35 shows the source code along with respective validations of the method called when the update button is clicked on and Figure 36 shows the method from the item class using the invite object to update the values in the database:

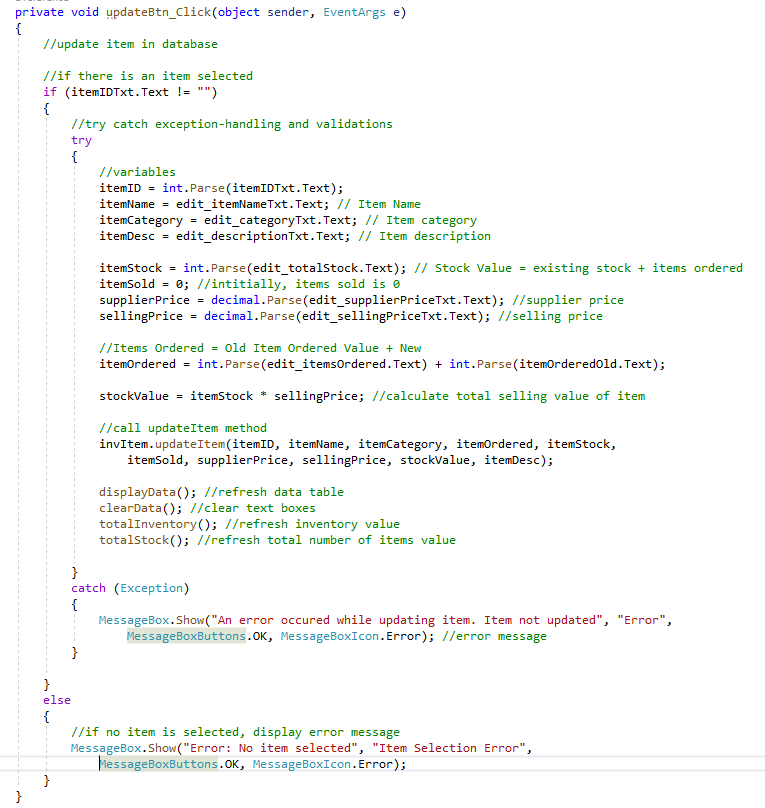


Figure : updateBtn\_Click( ) method source code



Figure : updateItem( ) method to update item details in database

**Validations**

If the user clicks on the “Update” button when no item is selected, the system displays an error message as can be seen in Figure 37.

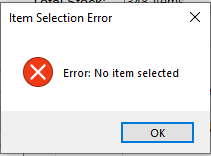


Figure : No items selected error

If the user enters invalid data type, the system will display an error message like what can be seen in Figure 38.



Figure : Unable to update error message

**Successful Update Attempt**

If there were no errors with the data entered by the user, the system successfully updates the item details in the database and displays a success message box as can be seen below in Figure 39

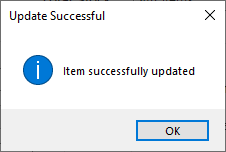
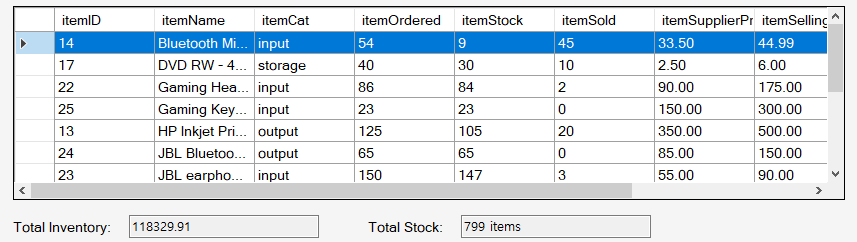


Figure :Successfully Updated Message Box

Subsequently, similar to when a user adds a new item, the total inventory and total stock values will be updated. Finally, the inventory’s item datagrid table will display the updated values.

**Before:**



**After:**

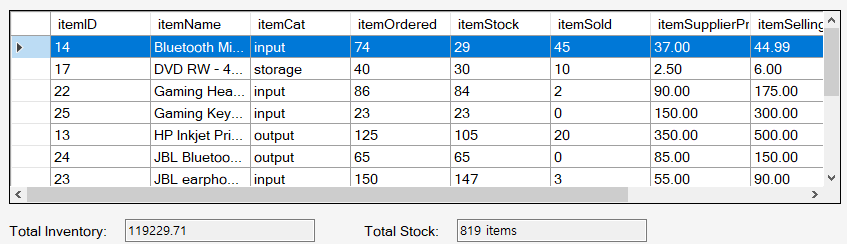


Figure : Updated values in datagrid, total inventory and total stock

**3.2.6 Delete Item**

In order to delete an item, similar to the update feature, the item details need to be selected and populated into Edit Item panel by double clicking on the item in the datagrid table. After an item is selected, in order to delete it, the user must click on the Delete button. The source code of the delete method along with respective validations can be seen in Figure 41.

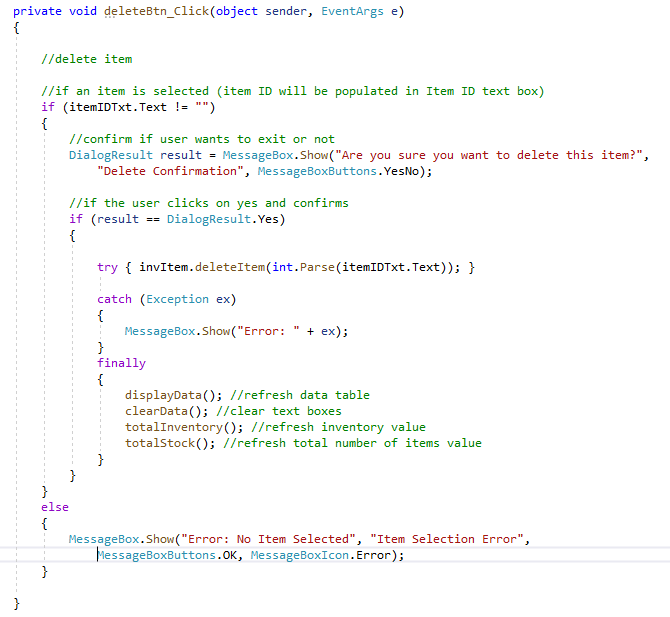


Figure : deleteBtn\_Click( ) method triggered when delete button is clicked

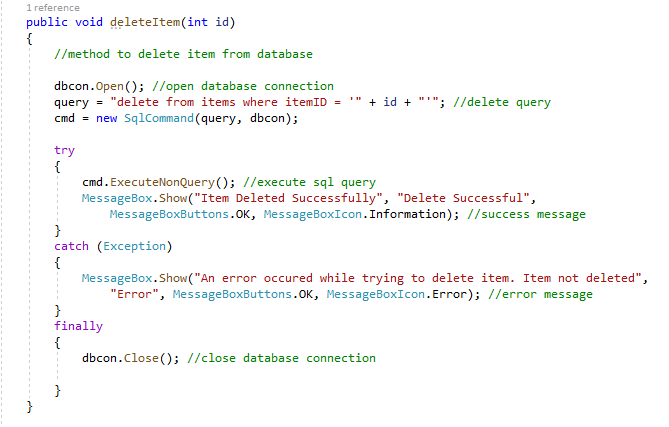


Figure : deleteItem( ) method called from item class

**Validations**

If no item is selected, the system will display the error message that can be seen in Figure 43.



Figure : No item selected error

**Confirmation**

Before deleting an item from the database, the system first shows a confirmation message to confirm if the user wants to delete the item or not (Figure 44)

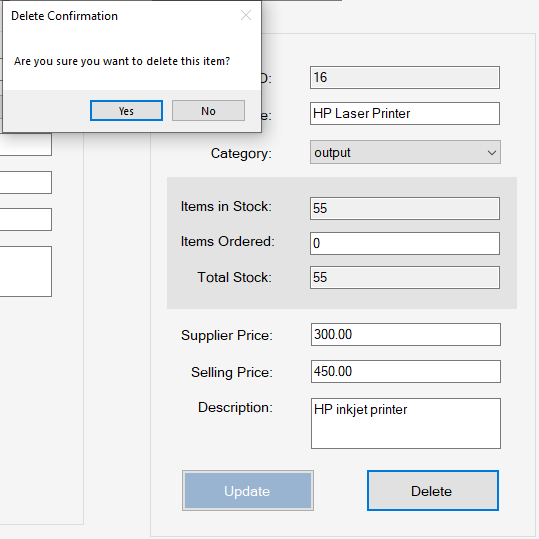


Figure : Delete confirmation

**Successfully Deleted Item**

After deleting the item successfully, the system displays a success message (Figure 45).

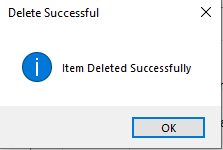


Figure : Delete success message

Subsequently, the item inventory datagrid table is updated and updated values of the Total Inventory and Total Stock are displayed.

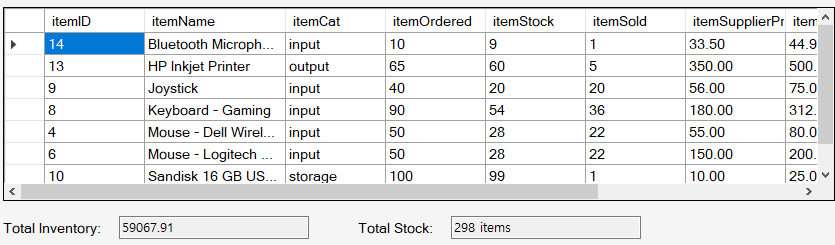


Figure : Item datagrid table , Total Inventory and Total Stock values updated after deleting item

## 3.3 Inventory Management System – Sales Transaction

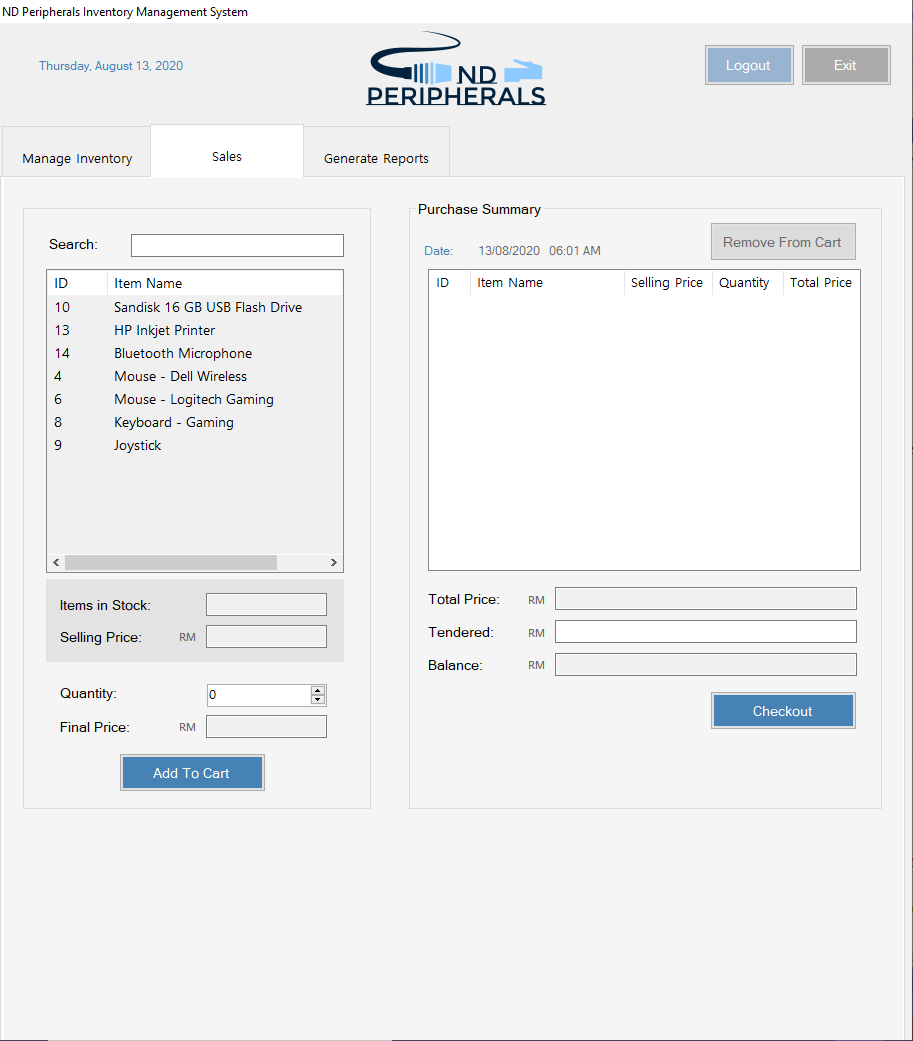


Figure : Sales Tab Selected UI

Figure 47 represents the interface when the sales tab is selected. The sales tab interface gives the user the ability to add an item to the cart, remove an item from the cart, view the purchase summary and checkout to complete their transaction.

**3.3.1 List of Items**

List of items available in the inventory is displayed in a listview in the window as can be seen in Figure 48 below. Please refer to Figure 49 and Figure 50 to see the source code of the method and method call used to display the items in the list view

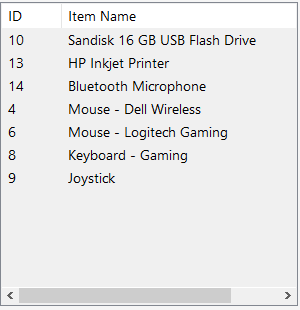


Figure : List View of Items

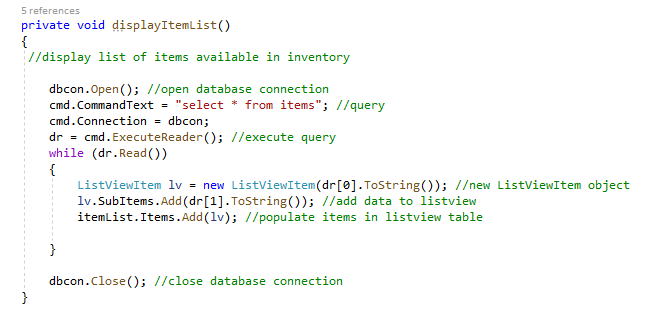


Figure : displayItemList( ) method definition



Figure : method call

**3.3.1 Search Item**

Similar to the search feature integrated in the Manage Inventory tab, there is a search feature integrated in the Sales tab as well that allows users to search for items in the item list given as can be seen in Figure 51. The source code can be seen in Figure 52.

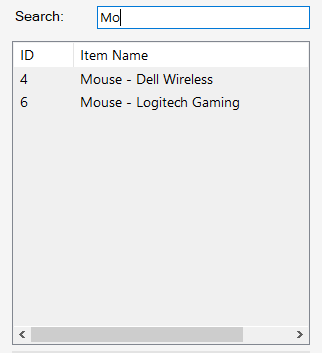


Figure : Item List Search

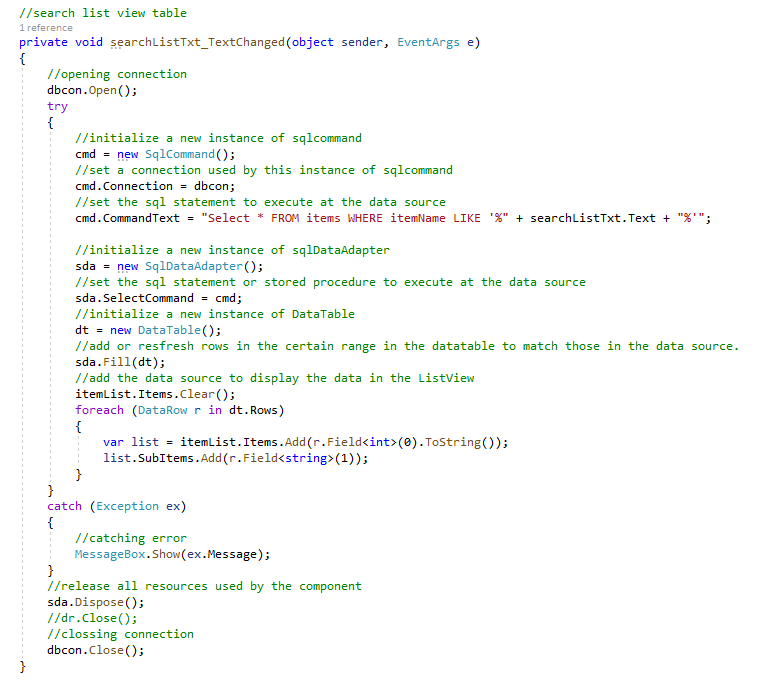


Figure : searchListTxt\_TextChanged( ) method - search list method souce code

**3.3.2 Add To Cart Panel**

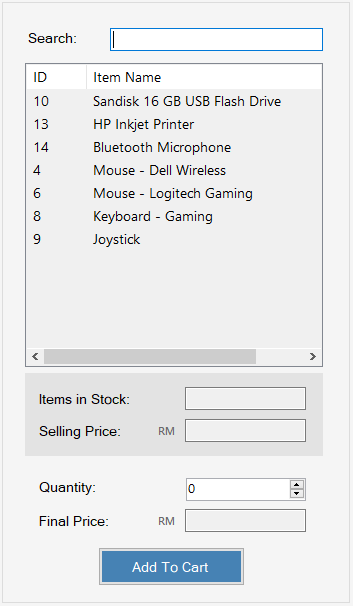


Figure : Add To Cart panel

The area designated to allow the user to select an object to add to their cart includes the search feature, items displayed in a list format with item ID and item name displayed. To view the price of each item and the quantity available in stock, the user will have to click on the item from the item list to select an item. The item stock values and selling price will be populated in the respective fields as can be seen in figure 54. The source code of the method can be seen in Figure 55.

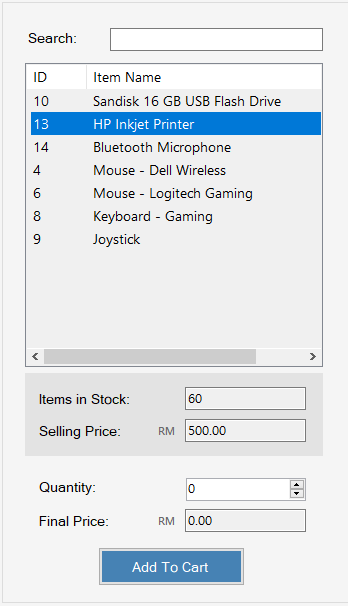


Figure : Item Selected from list

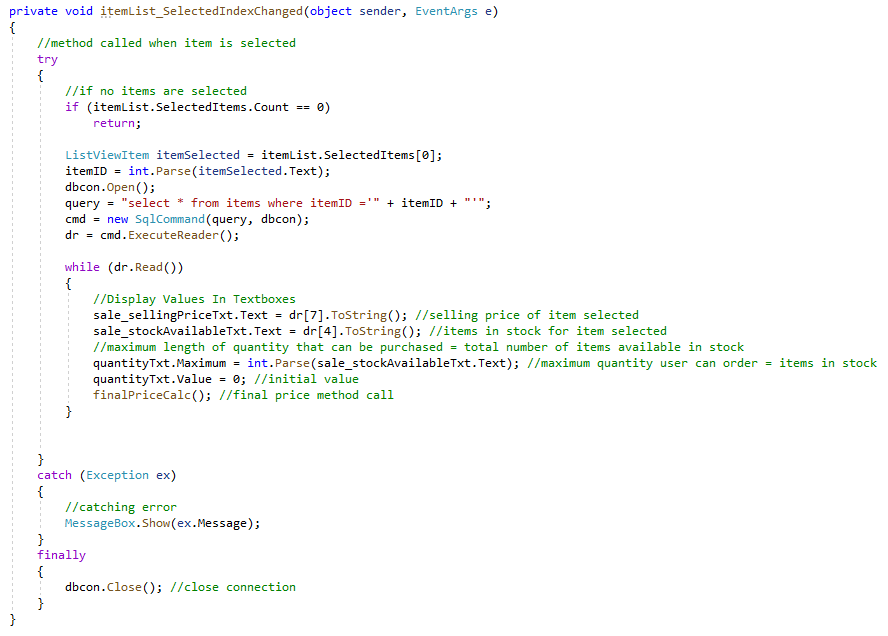


Figure : Item Selected method

**Quantity and Final Price**

The admin user can enter the quantity of the item that will be purchased in the “Quantity” number box. Upon entering the value , the system will calculate the final price for the item in the Final Price field as can be seen in Figure 56.

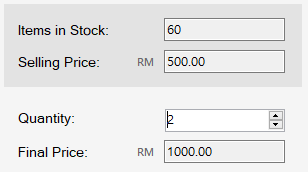


Figure : Quantity and Final Price



Figure : finalPriceCalc() method and method call

Figure 57 represents the source code of the method defined to calculate the final price and the respective method call.

**Please note:** the maximum value of the quantity that can be entered in the Quantity number text box is set to the total number of this item that is available in the stock which can be seen in the code in Figure 55.

**Add To Cart – Validation**

The system carries out validations when the add to cart button to check whether an item is selected and to ensure the correct values are entered. Figure 58 displays the error message that will be seen if the user clicks on the Add to Cart button without selecting an item.

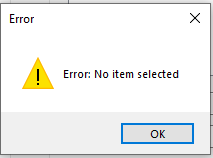


Figure : Error Message - no item selected

If the admin clicks the add to cart button without entering the quantity to be purchased, the error message seen in Figure 59 will be displayed.

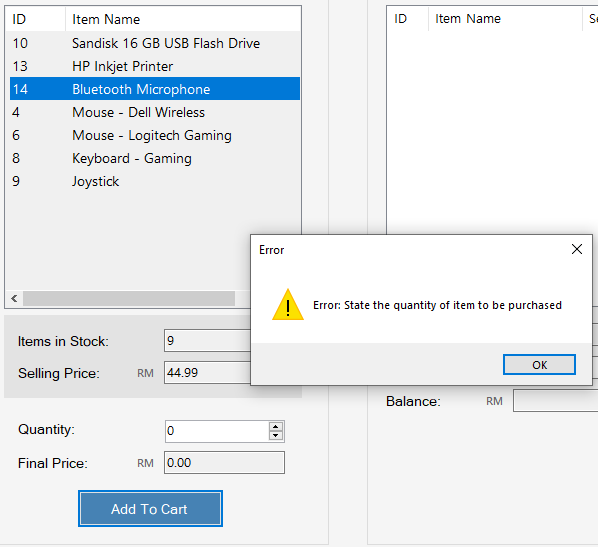


Figure : Error Message - quantity not entered

**Note:** since a number box has been taken instead of a textbox for the Quantity criteria, no invalid characters can be added and the field is set to allow values from 0 to items available in stock to be added, hence, no additional validation is required for this field.

**Successfully Add Item To Cart**

If there are no errors, the system will add the item selected at the stated quantity to the cart which can be seen in the Purchase Summary Panel in Figure 60. Subsequently, the quantity and final price fields will be cleared out to allow the user to select another item and make a purchase.

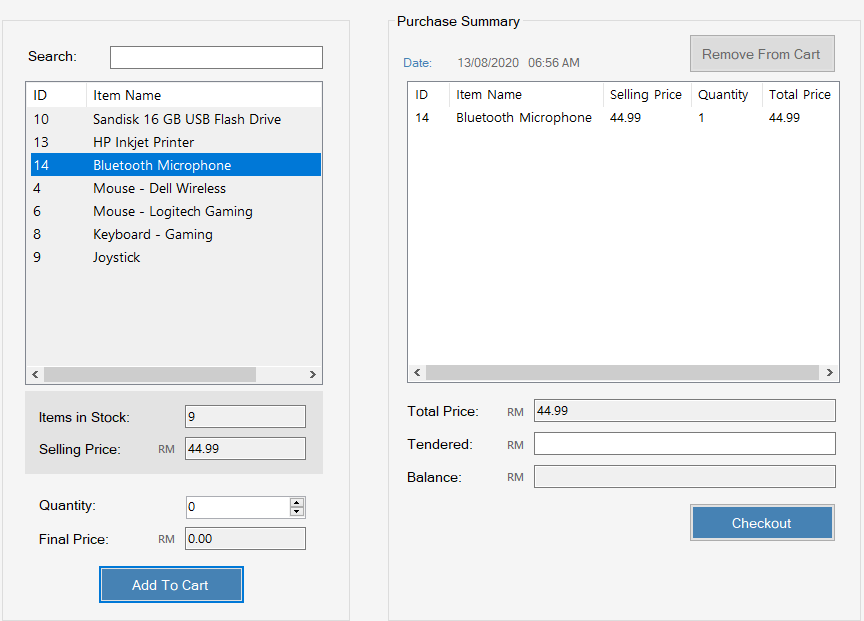


Figure : Added item to purchase summary (cart)

Please note, since the cart is displayed in the Purchase Summary panel which is adjacent to the items panel, no success message is shown upon adding the item to the cart for efficient user experience.

Additionally, after an item is added to the cart to be purchased, the system calculates the total price that the user has to pay and displays it in the Total Price field. Each time a new item is added, the price is automatically calculated and updated which can be seen in Figure 61.

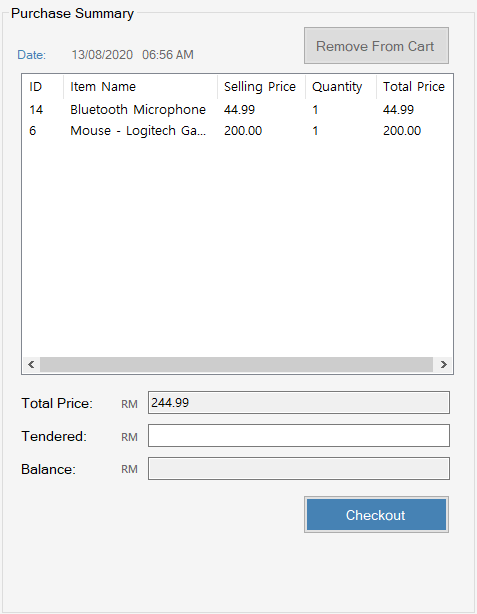
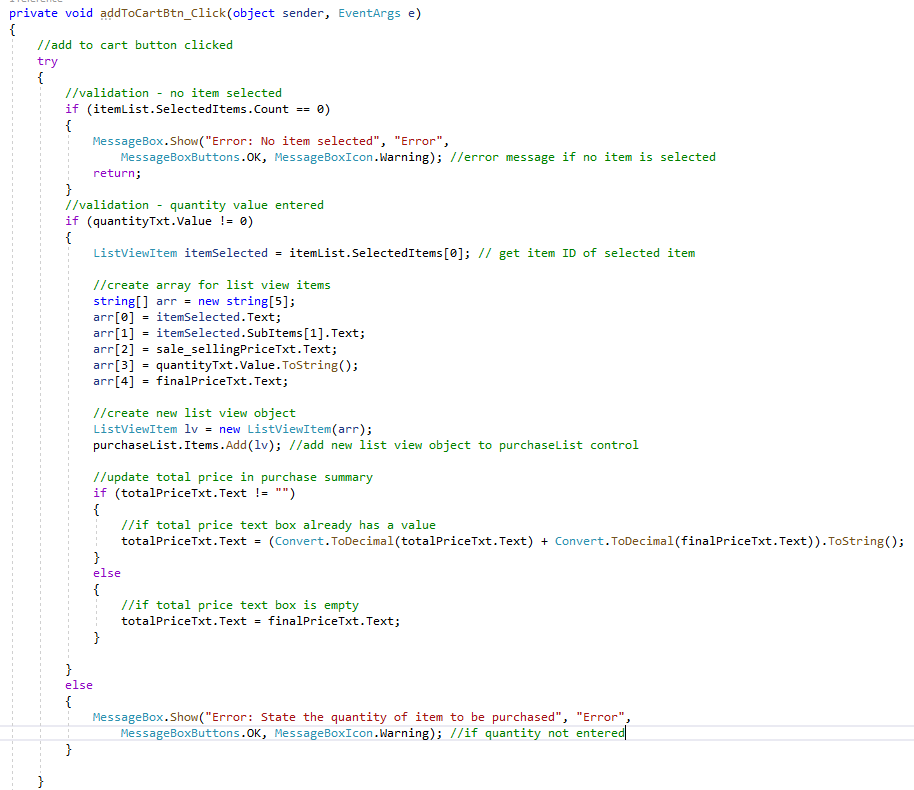


Figure : Total Price updated after more items added to cart

Please refer to Figure 62 for source code of the method definition declared to add an item to cart



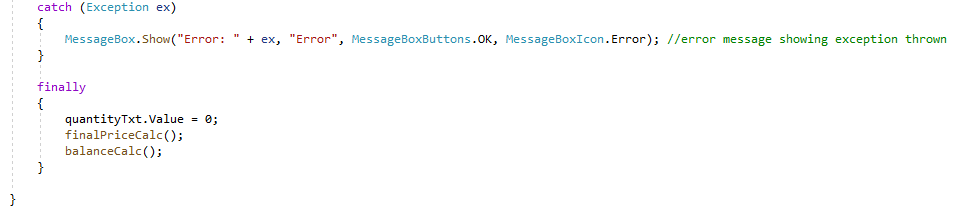


Figure : addToCartBtn\_Click method definition

**3.3.3 Remove From Cart**

Figure 63 represents the method definition of the removeFromCartBtn\_click( ) method which is called when the user wants to remove an item from the cart.



Figure : Remove From Cart method definition

**Validation**

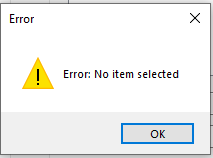


Figure : No item selected error message

In order to delete an item, the item must first be selected from the list of items added to the cart. If no item is selected, the system will display the error message that can be seen in Figure 64.

**Successfully Removed**

Once the item is removed from the cart, the list of items in the cart is updated and the total amount is recalculated.

**3.3.4 Checkout**

Once the admin has filled the cart with the items that will be purchased by the customer, the next step before checking out is to enter the tendered amount. The tendered amount is the money paid by the customer to purchase the items.

**Balance**

Balance refers to the amount of money the admin will return to the customer for paying an access amount from the Total Price that needs to be paid to make the purchase. Balance is calculated with the following formula:

Balance = Tendered Amount – Total Price

Figure 65 displays the balance being calculated with the tendered amount is entered in the system. The balance is only calculated if the tendered amount is greater than the total price, which can be seen in the method definition of the balanceCalc( ) method in figure 66.

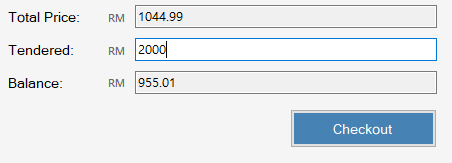


Figure : Balance calculated

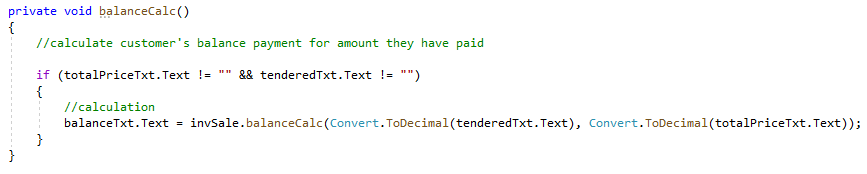


Figure : method definition to calculate balance

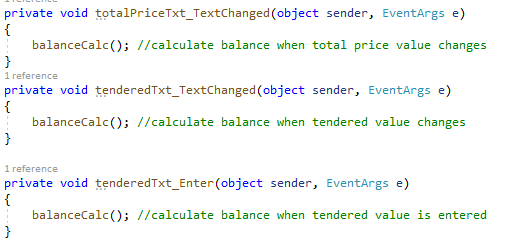


Figure : method call to calculate balance

Figure 67 shows the method call of the balanceCalc( ) method. The method is called when the total price value changes (in case a new item is added to the cart or removed from the cart), or when the tendered amount is entered /changed in the tendered textbox field.

**Checkout – Validation**

The transaction cannot be completed if the admin does not enter the tendered amount. The error message can be seen in Figure 68

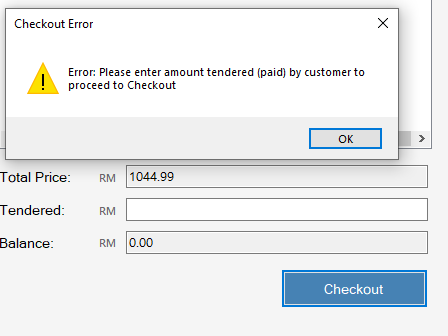


Figure : Error Message Box - Tendered amount not entered

The system also reports an error if the tendered amount is less than the total price as can be seen in Figure 69.

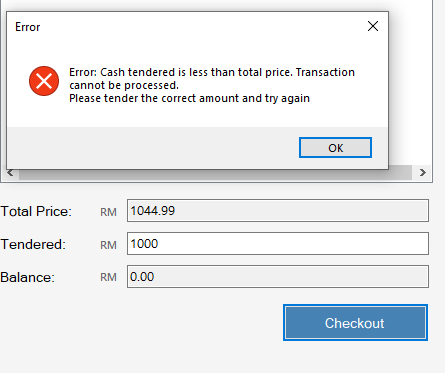


Figure : Error message box - tendered amount less than total amount

**Checkout Confirmation**

If the amount tendered is entered correctly and there are no other errors, upon clicking on the checkout button, the system will show a confirmation dialog box to confirm whether the admin user wishes to checkout and complete the sales transaction as can be see in Figure 70

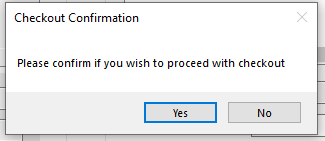


Figure : Confirm checkout

If the user clicks on the “Yes” button to confirm, the system proceeds with the transaction and displays a success message along with a purchase summary as can be seen in Figure 71.



Figure : Purchase Success Message

**Update Stock Values**

After the transaction is successfully completed, the stock values of the items purchased are updated on the database. Please refer to Figure 72 for the method call of the updateStockValue() method of the **sales** class that is defined to update the stock value of each item purchased.

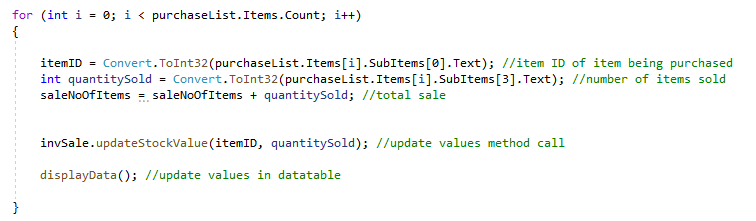


Figure : for loop that calls updateStockValue( ) method for each item purchased

Figure 73 represents the method definition of the updateStockValue( ) method. As it can be seen in the source code, first the ID of item is used to retrieve the existing values of that item. Then calculations are performed to calculate the new number of items in stock after subtracting the quantity sold from the previous number of items in stock. After the new values are obtained, the values are updated in the database. In addition to the number of items available in stock, the total stock value (total selling value of the stock) is also recalculated and updated in the system. Finally, the displayData( ), totalInventory( ) and totalStock( ) methods are called to recalculate and update the values displayed in the Item datagrid, Total Inventory field and Total Stock field respectively on the Manage Inventory tab area (Refer to Figure 72 for the displayData( ) method call and Figure 74 for the other two)

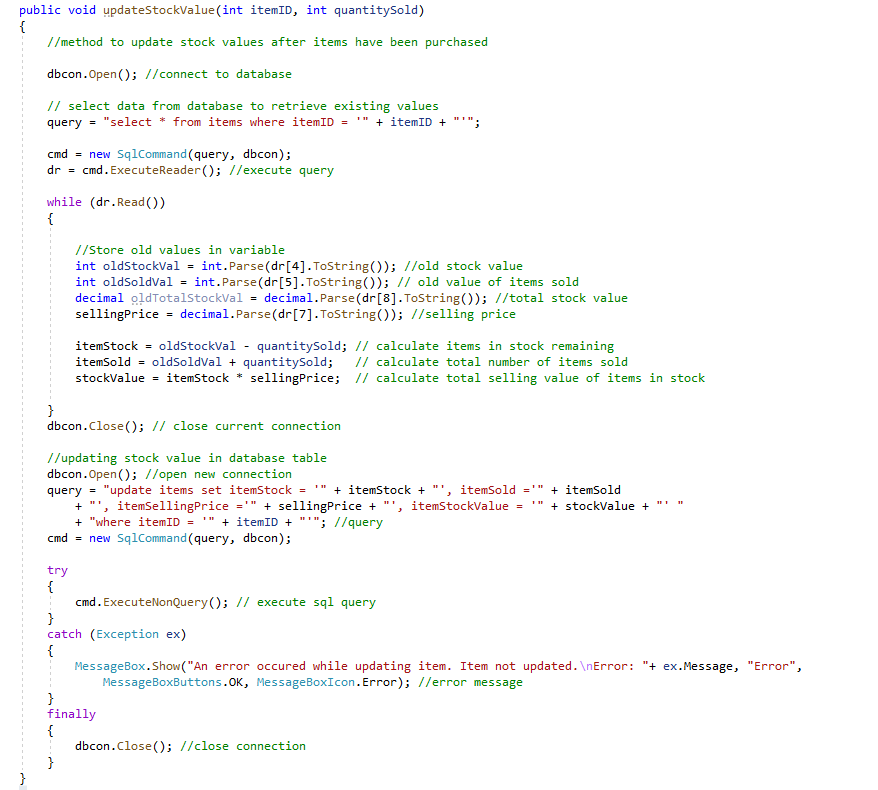


Figure : updateStockValue( ) method definition



Figure : method calls

**Example of Update Stock :**

1. Figure 75 shows that the item with Item ID 8, Keyboard – Gaming has 55 items in stock and 37 items sold.

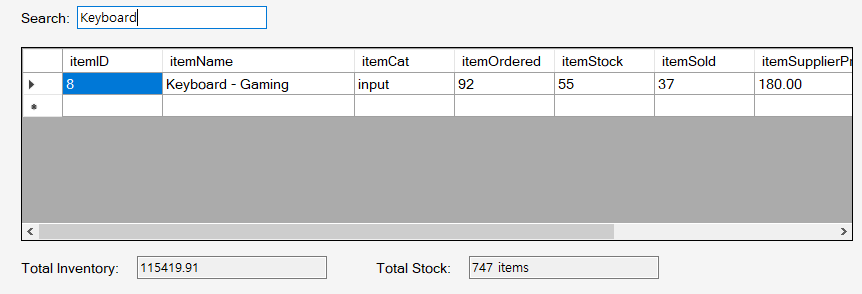


Figure : Items available before purchase and items sold

2. In this example, a quantity of 5 Keyboard-Gaming are added to the cart and purchased as can be seen in Figure 76 and Figure 77

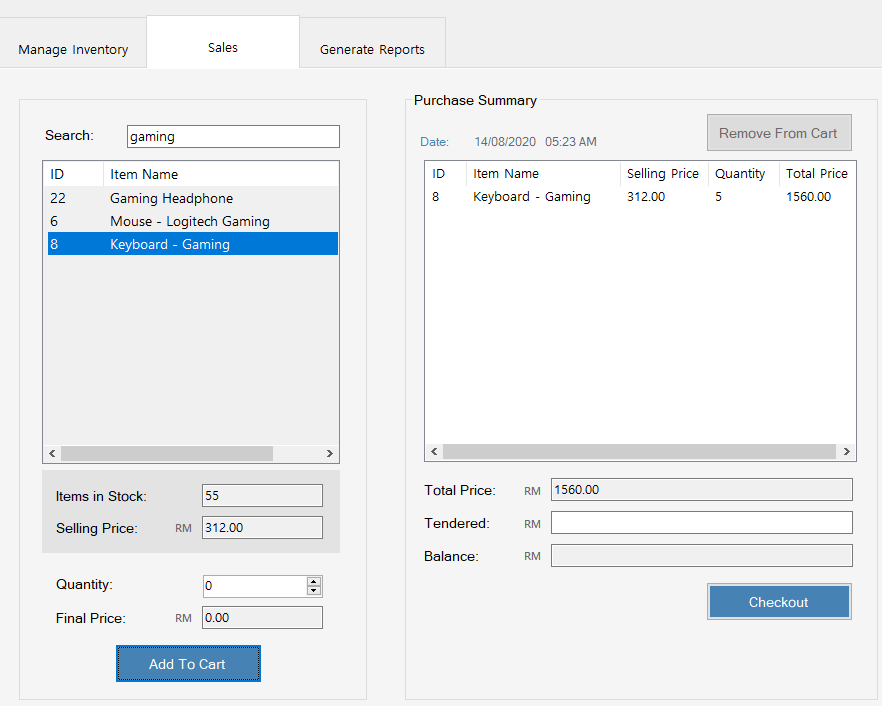


Figure : Keyboard Gaming - Added to Cart

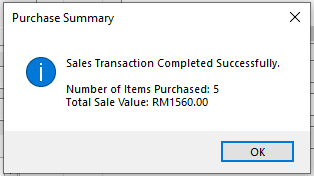


Figure : Purchase Summary of 5 items purchased

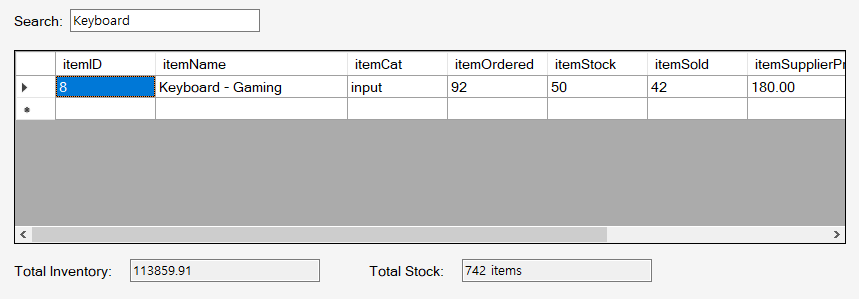
3. From Figure 78 it can be seen that after the sales transaction was successfully completed, the value of the items in stock and items sold were updated to 50 items in stock and 42 items sold respectively. Additionally, the value of the Total Inventory and Total Stock were also recalculated and updated as can be seen below the datatable in Figure 78.

Figure : Values updated

**Sales Database**

When a purchase is made, the sale value, total number of items purchased and date of purchase are recorded in the sales table in the database. Each sale is assigned its own unique ID. Figure 79 shows the sales record of the example given for the “Update Stock Value” above - The record for the sales transaction is added to the sales table and the sales ID is 41. Figure 80 represents the definition of the method newSale( ) which is called through the invSale object of the sale class instantiated in the mainSystem( ) class. This method shows how the new sale record was made after successful purchase.

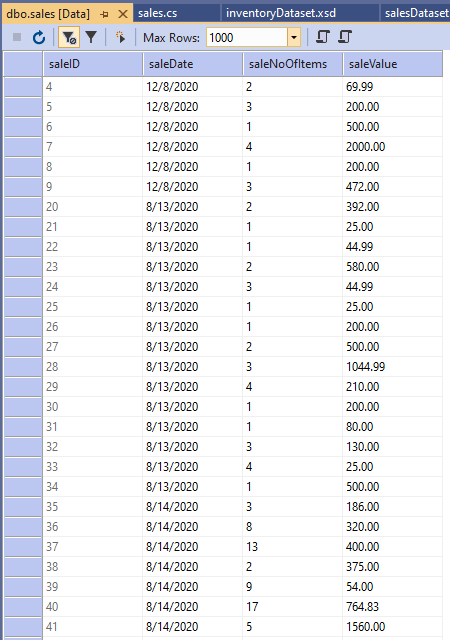


Figure : Sales Record Created

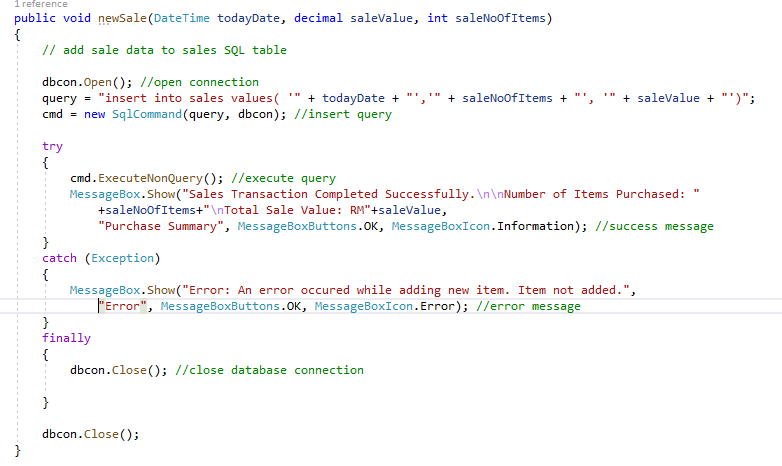


Figure : newSale( ) Method Definition

To review the entire source code of the checkoutBtn\_Click( ) method which is triggered through the event of clicking the checkout button, please refer to Figure 81 and Figure 82. The method also includes the coding for the validations that were demonstrated above. For the definition of the methods updateStockValue( ) and newSale( ) that were called in the checkoutBtn\_Click( ) method to update the stock values and create a new sales record, please refer back to Figure 72 and 80 respectively.

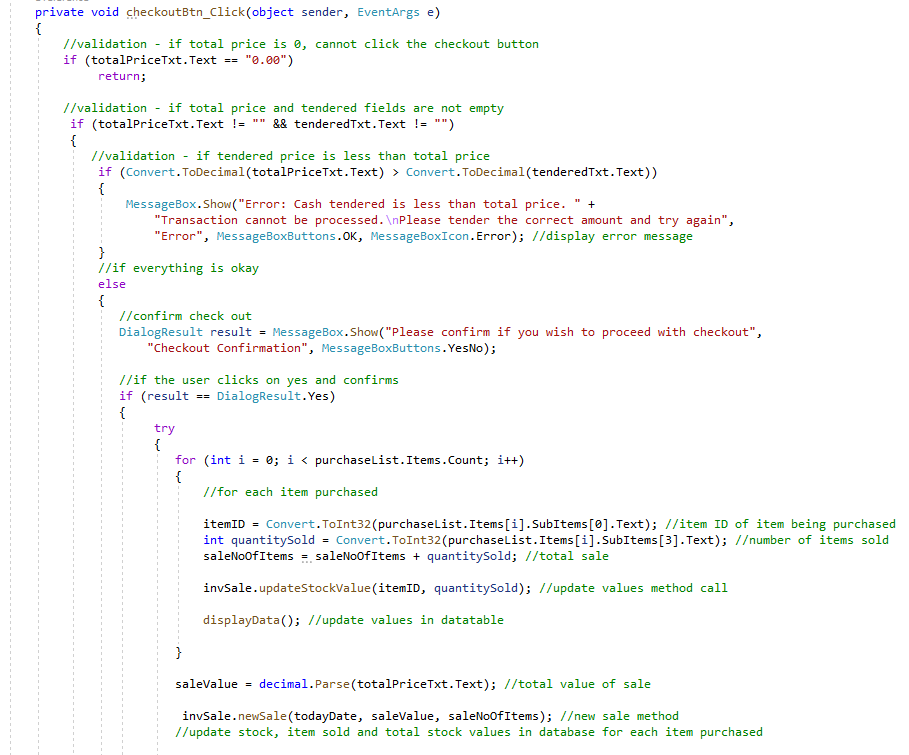


Figure : checkoutBtn\_Click( ) Method definition - triggered by clicking on checkout button



Figure : checkoutBtn\_Click( ) Method Defintion - continued code

## 3.4 Inventory Management System– Generate Report

The system allows the admin to generate an inventory report and a sales report. The Generate Report tab consists of two buttons which, upon being clicked, will generate the respective reports. For the entire source code please refer to **Appendix C**

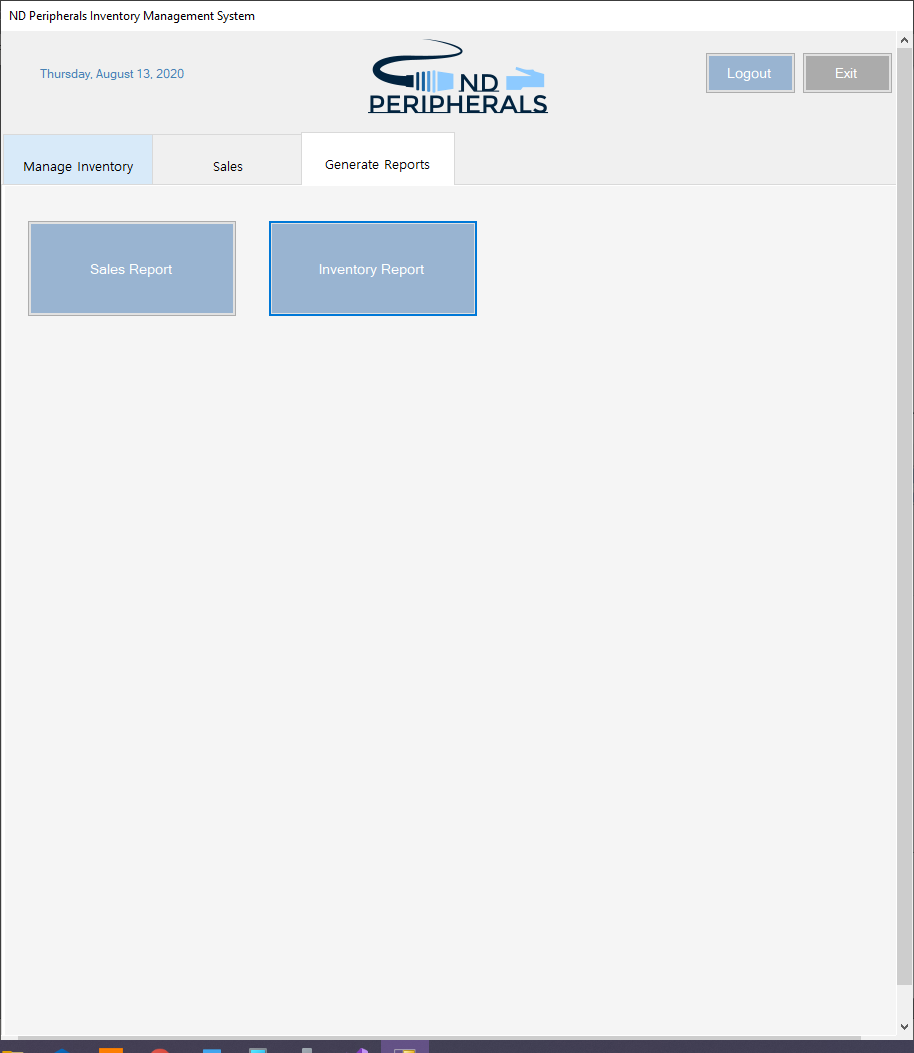


Figure : Generate Report Interface

**3.4.1 Inventory Report**

Upon clicking on the “Inventory Report” button in the generate report table, the system generates the latest report of the inventory. The inventory report provides information on all the items in the inventory, total items in stock and total inventory value. It can be exported, saved and printed as well. The report can be seen in Figure 84.

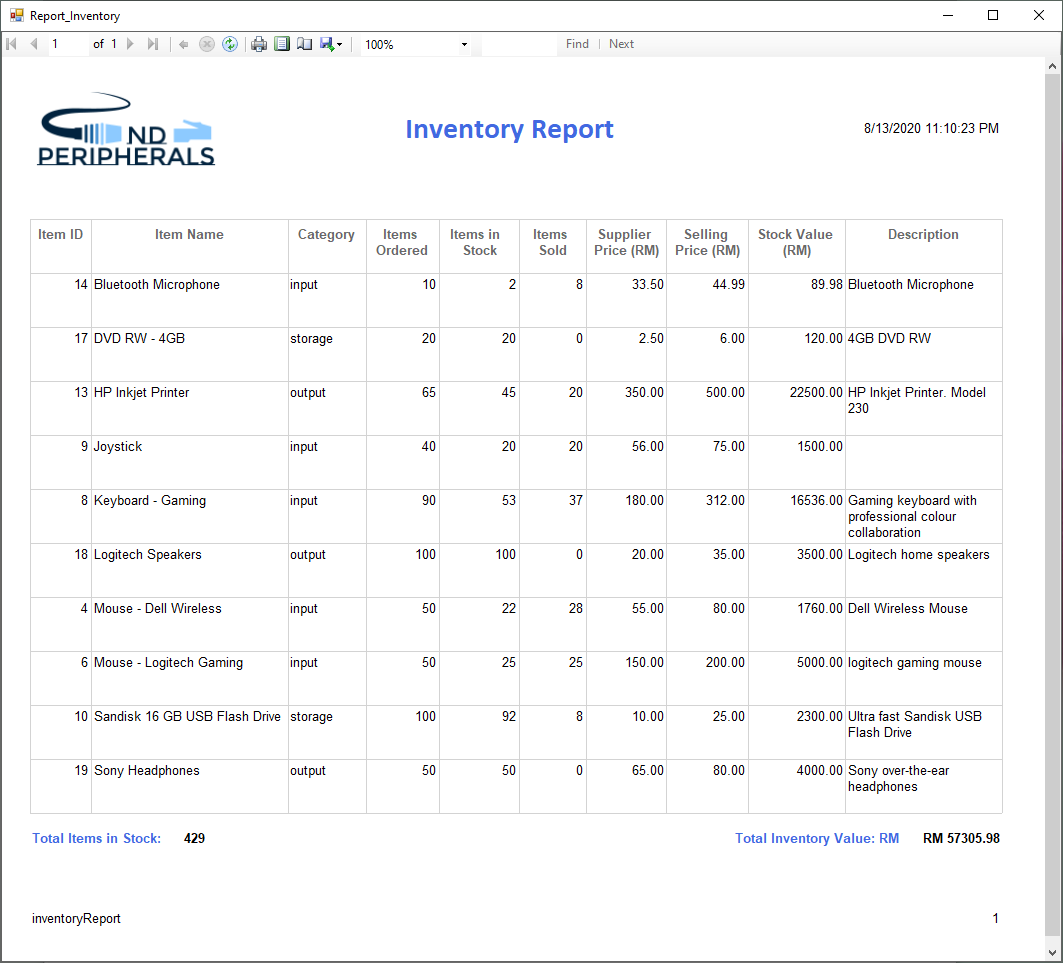


Figure : Inventory Report

Figure 85 shows the source code of how the repInv object is created from the Report\_Inventory class within the intventoryReportBtn\_Click( ) method to show the report when the inventory report button is clicked.

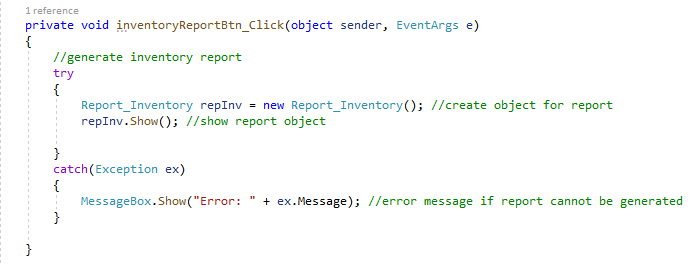


Figure : Inventory report object

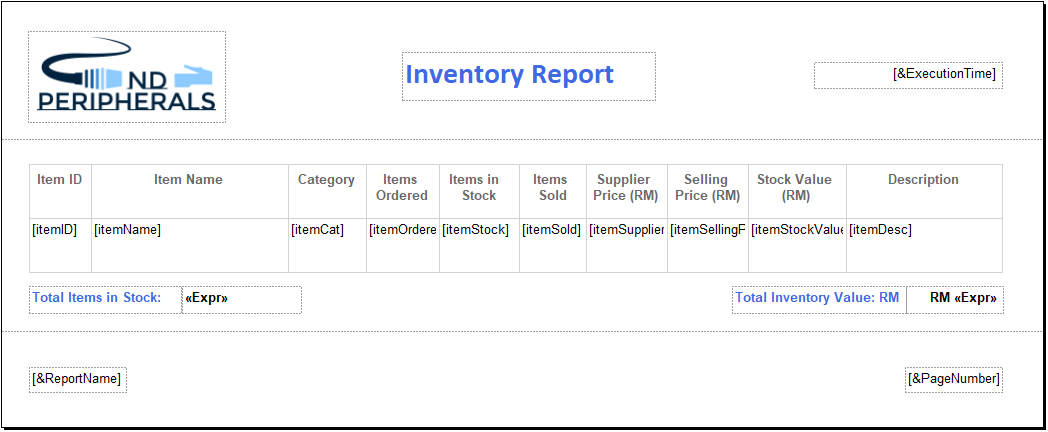


Figure : Inventory report design view

Figure 86 shows the Report Design that uses the inventory dataset that to retrieve the values from the database. Figure 87 Shows the inventory dataset dataset while Figure 88 shows the Report\_Inventory class which was defined to display the report when the object of the class was created and the .Show( ) method was called when the button was clicked which can be seen in figure 86.

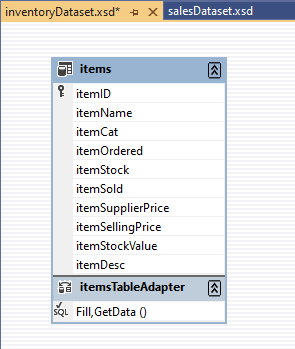


Figure : inventory dataset

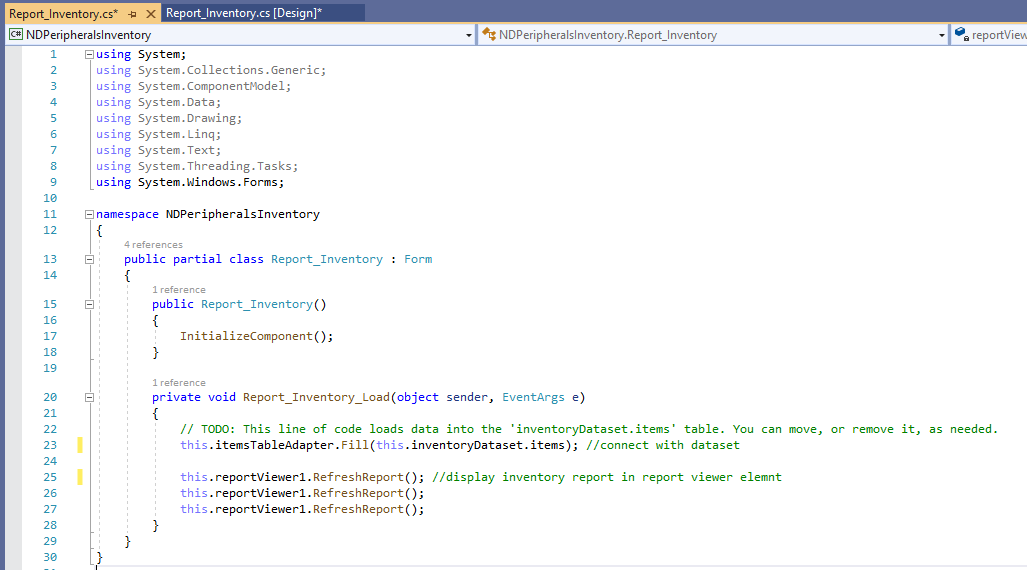


Figure : Report\_Inventory class

**3.4.2 Sales Reports**

Upon clicking on the “Sales Report” button in the generate report table, the system generates the latest report of the sales data. The sales report provides information on sales which includes date of purchase, total number of items bought and total sales value. It can be exported, saved and printed as well. The report can be seen in Figure 89.



Figure : Sales Report

Figure 90 shows the source code of how the repSales object is created from the Report\_Sales class within the salesReportBtn\_Click( ) method to show the report when the inventory report button is clicked.

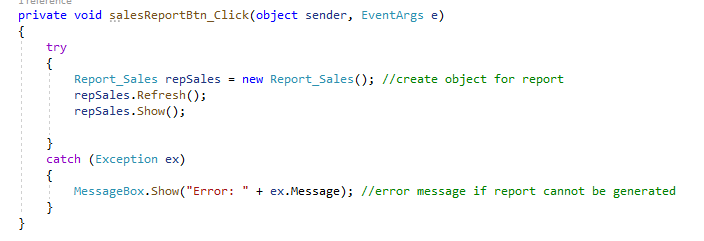


Figure : Sales Report object

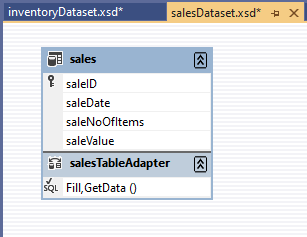


Figure : salesDataset

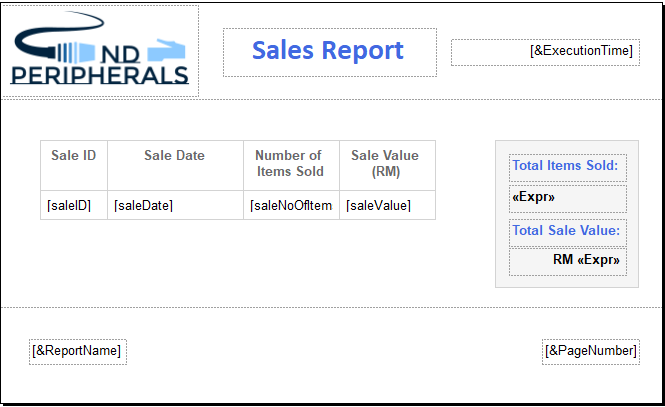


Figure : Sales report design view

Figure 92 shows the Report Design that uses the sales dataset that to retrieve the values from the database. Figure 91 Shows the sales dataset while Figure 93 shows the Report\_Sales class which was defined to display the report when the object of the class was created and the .Show( ) method was called when the button was clicked which can be seen in figure 90.

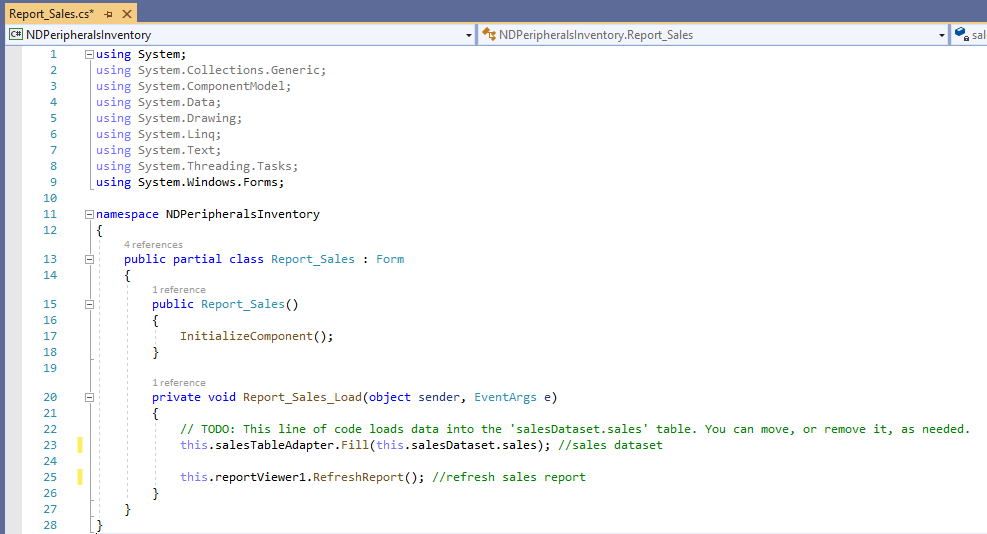


Figure : Report\_Sales class

## 3.5 SQL Database

MicroSoft SQL Database was used for data storage in the system.

The name of the database of the system is inventorydb.mdf as can be seen in Figure 94. The database has 3 main tables – login, items and sales. Each table will be discussed below.

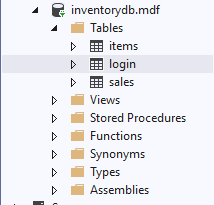


Figure : inventorydb.mdf database and tables

**3.5.1 login table**

Login table stores the user’s login details which will allow the user to log into the system. It consists of 2 fields – username and password. The primary key is the username. Figure 95 shows the definition (design) of the login table and Figure 96 shows the table data of the login table.



Figure : Login Table Definition

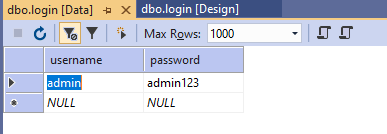


Figure : login table data

**3.5.2 item table**

The item table stores the data of items in the store’s inventory. The fields in the item table are **itemID** for the unique ID of each item, **itemName** for the name of each item, **itemCat** for the item’s category, **itemOrdered** for the number of items ordered, **itemStock**  for the number of items in stock, **itemSold** for the number of items sold, **itemSupplierPrice** for the supplier price (buying price) of each item, **itemSellingPrice** for the selling price (sale price) of each item, **itemStockValue** for the total selling value of each item (this will be used later to calculate total inventory value) and **itemDesc** for a short description of each item. itemID is the primary key. Please refer to Figure 97 for the design (definition) of the item table and to Figure 98 for the table data.

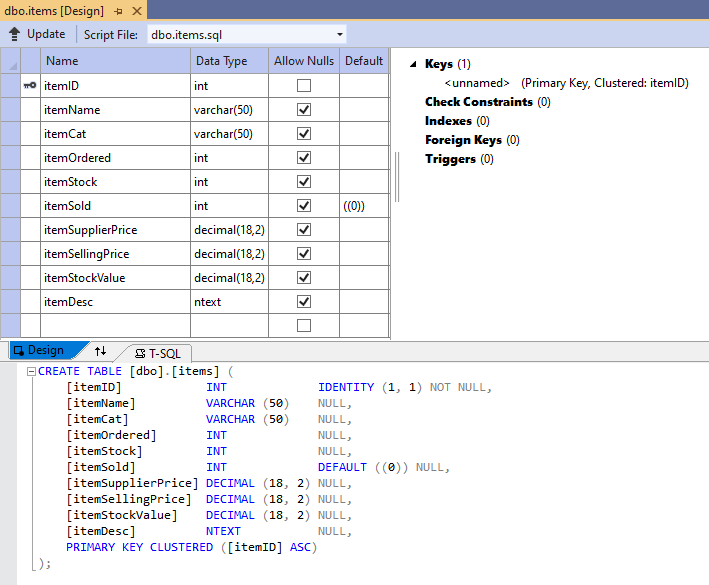


Figure : item table design (definition)

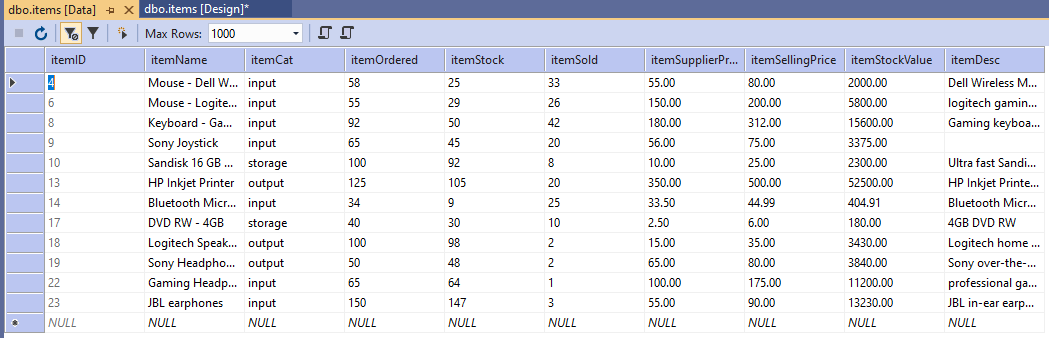


Figure : item table

**3.5.3 sales table**

sales table is used to keep a record of each sales transaction that takes place. It has 4 fields, **saleID** which is a unique ID assigned to each sales transaction, **salesDate** which is the date when the transaction took place, **saleNoOfItems** which is the total number of items purchased and **saleValue** which is the total value of the purchase made. Figure 99 represents the sales table design (definition) and Figure 100 shows the table data.

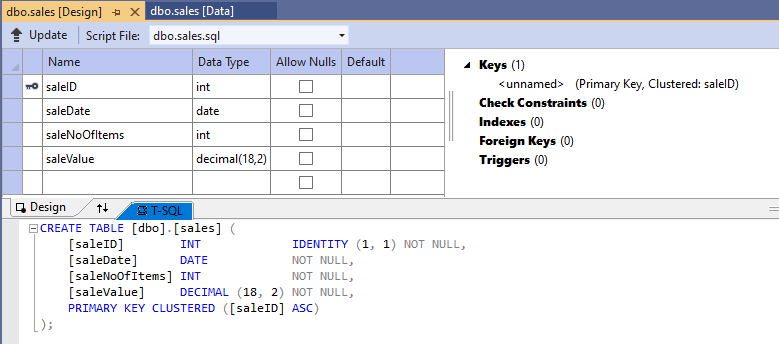


Figure : sales table definition (design)

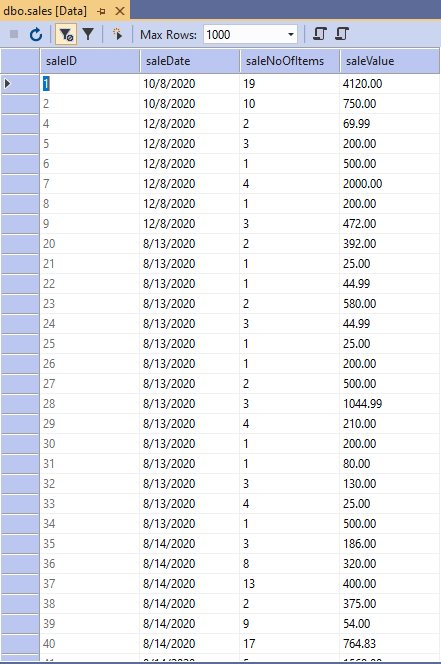


Figure : sales table data

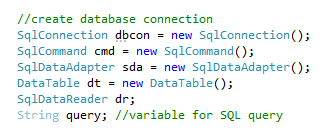
**3.5.4 Database connectivity**

In order to establish the database connectivity in the system, the following steps were taken:

1. The following library/package was included in the header of the class:



2. Objects for required SQL classes were created within the class definition



3. The database connection string was defined in the constructor if the class. The connection string is necessary to define the file path of the database file.



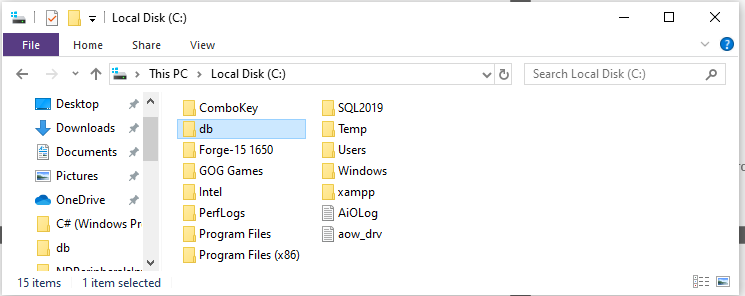
# APPENDIX A: SOFTWARE INSTALLATION AND DEPLOYMENT

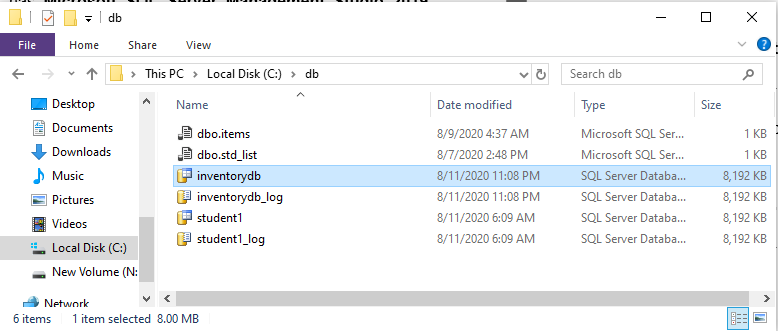
**System Pre-requisites:**

This windows application was developed using **Microsoft Visual Studio 2019**. In order to deploy the code, please ensure that you have Microsoft Visual Studio 2019 and above installed on your computer system. Also ensure that your system has **Microsoft SQL Server Management Studio 2019** installed on your system.

**Database:**

For the correct file path, copy the “db” folder from the project folder to your C:/ such that the file path of your database file will be C:\db\inventorydb.mdf





**Login Details:**

Username – admin

Password – admin123

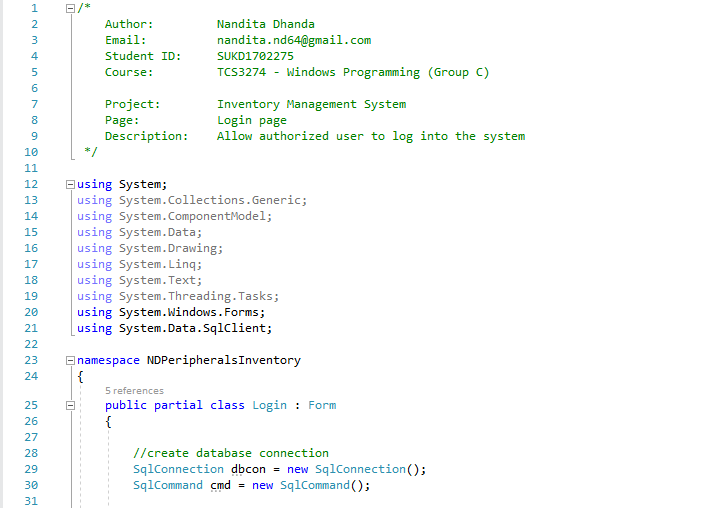
# APPENDIX B: USER MANUAL

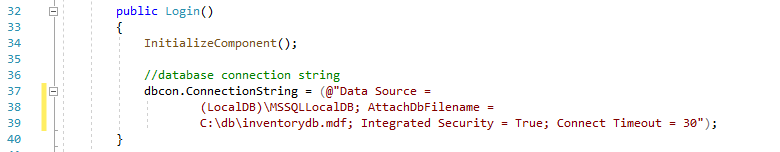
For the user manual, please refer to the video demonstration submitted alongside the system. It includes a detailed step-by-step procedure of how the system operates

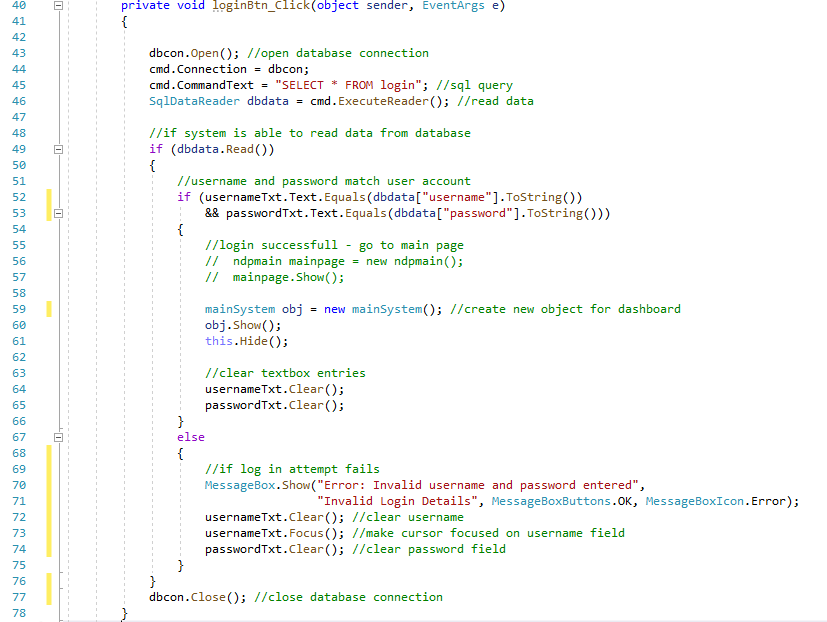
# Appendix C: Source Code

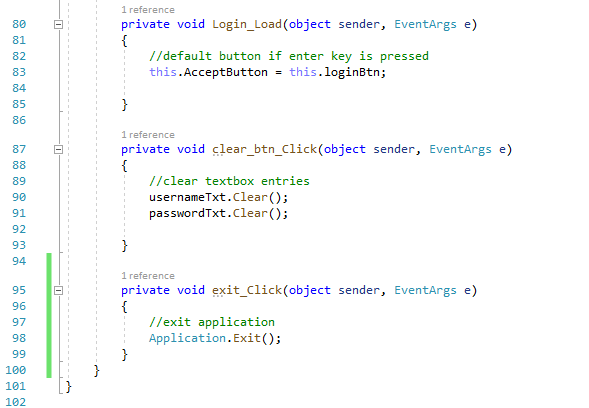
Please note: the detailed explanation of how the code is implemented in each section of the system has been provided in the report sections above.

## Section 1: Login Window Source Code – login.cs

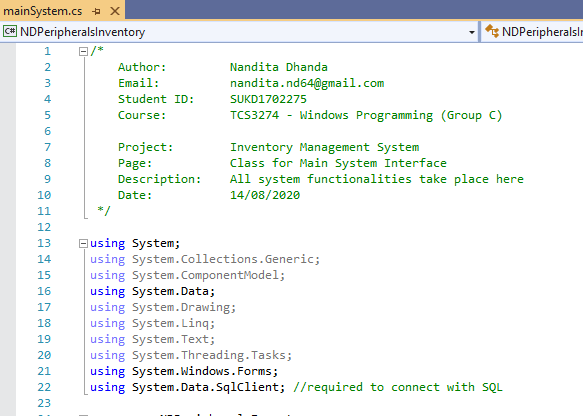


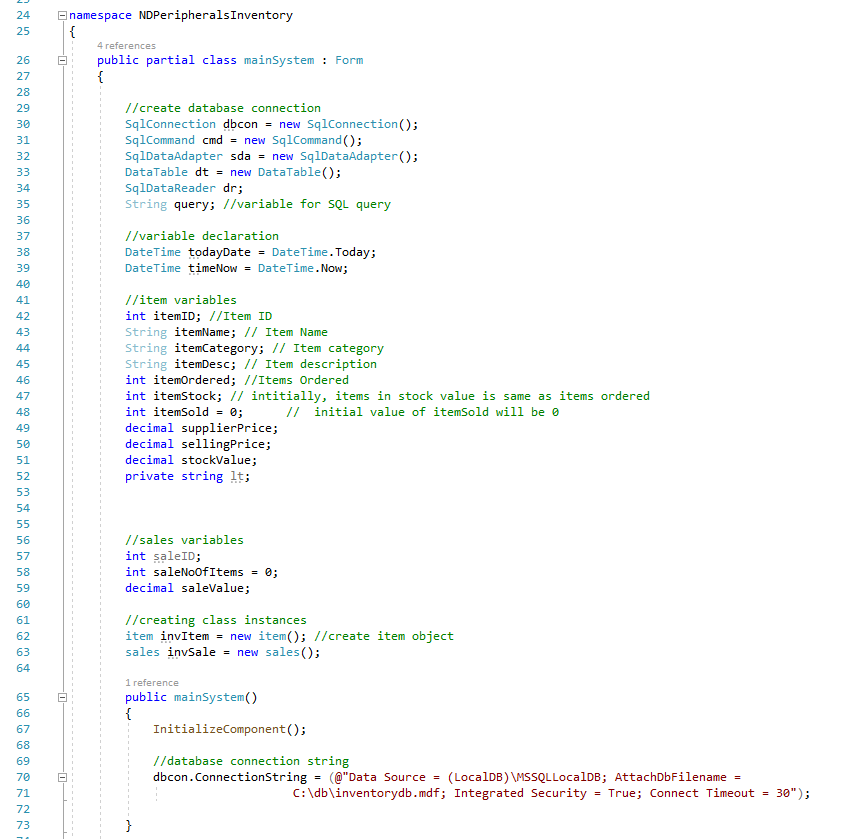


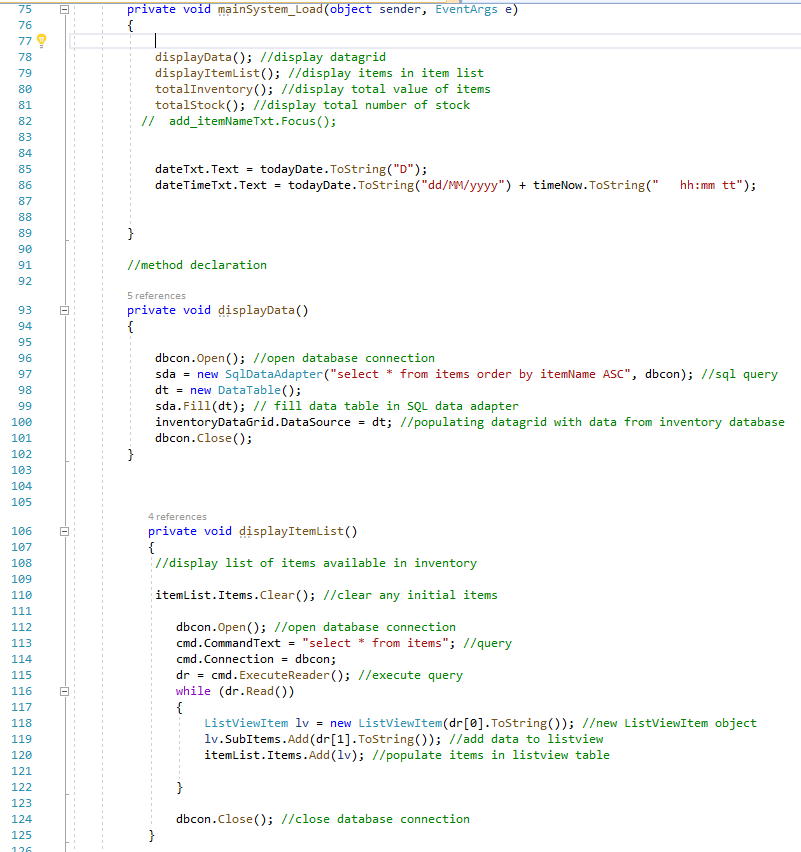


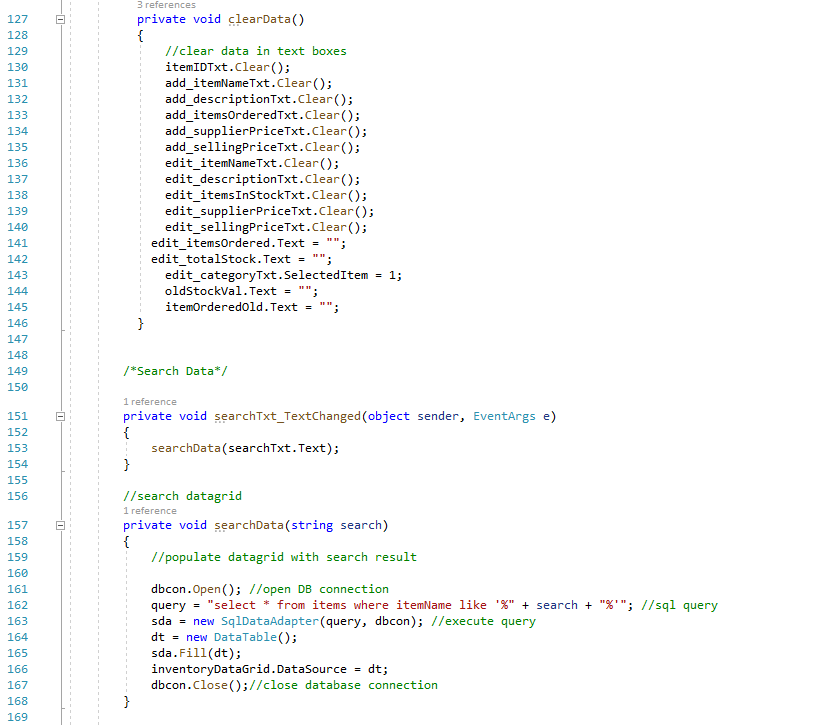


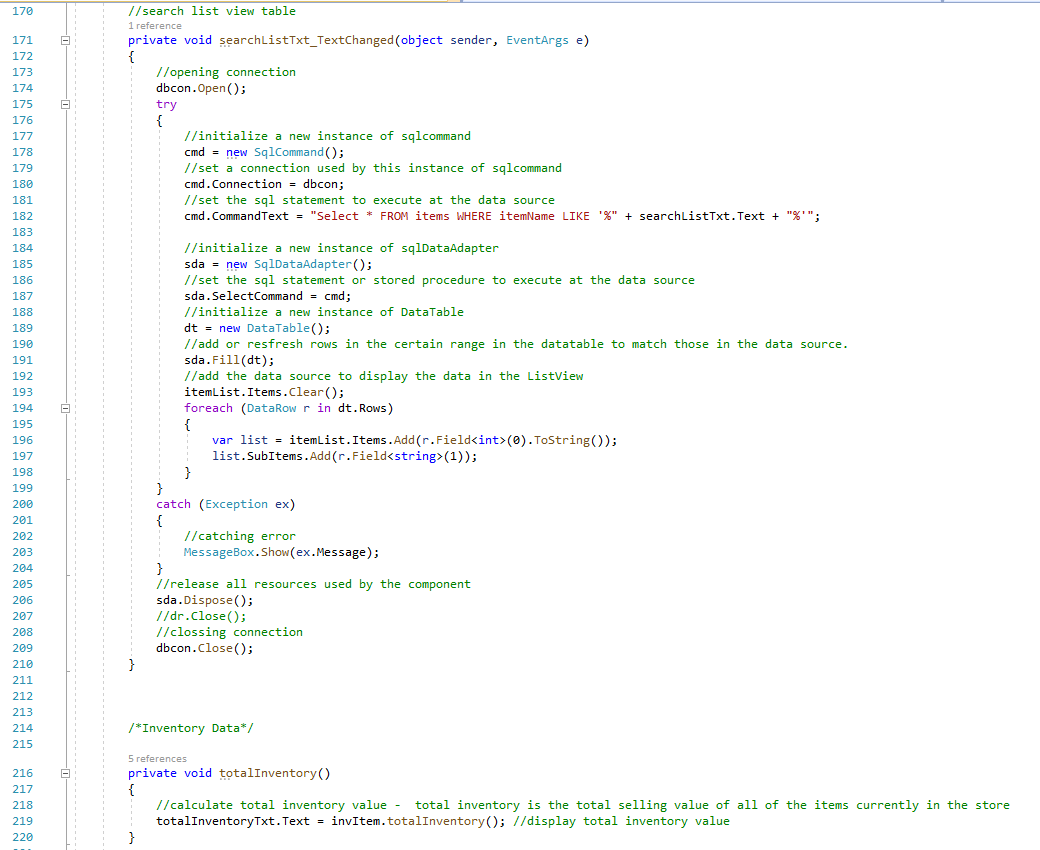
## Section 2 – mainSystem.cs

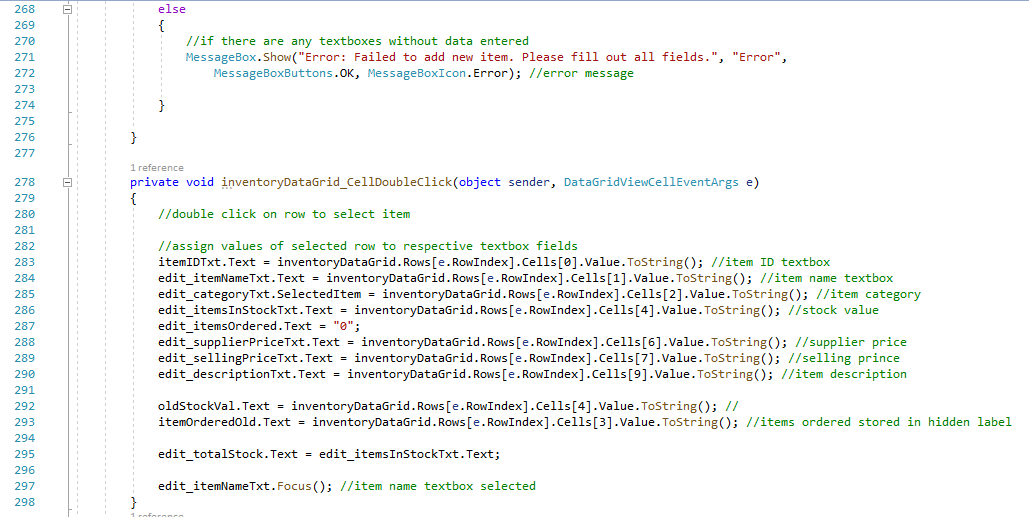
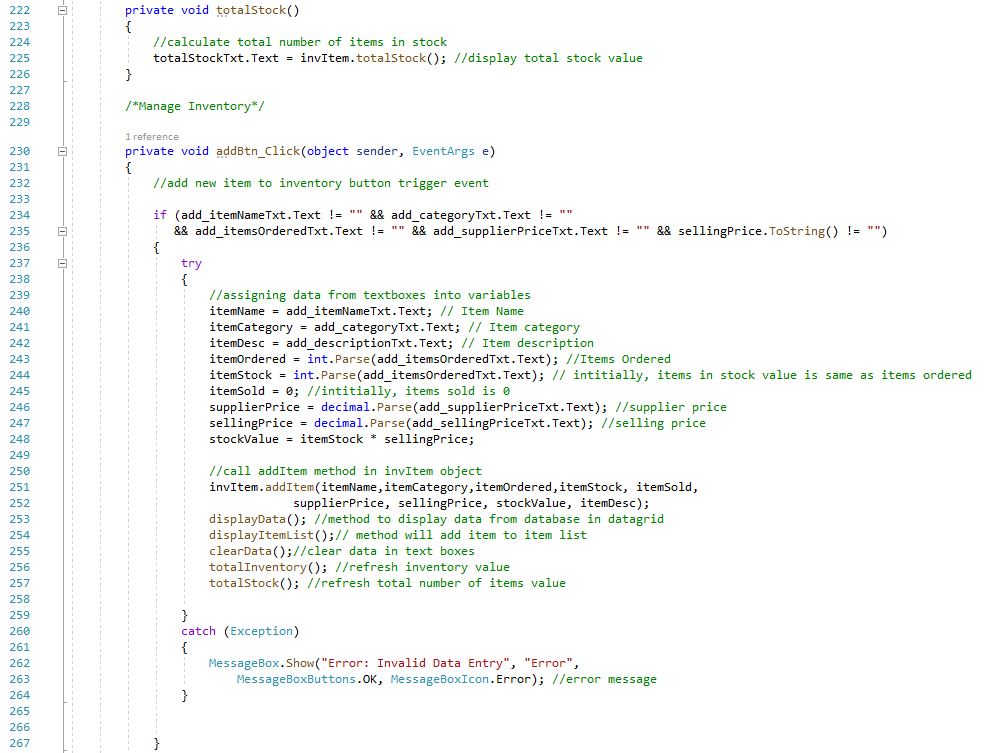


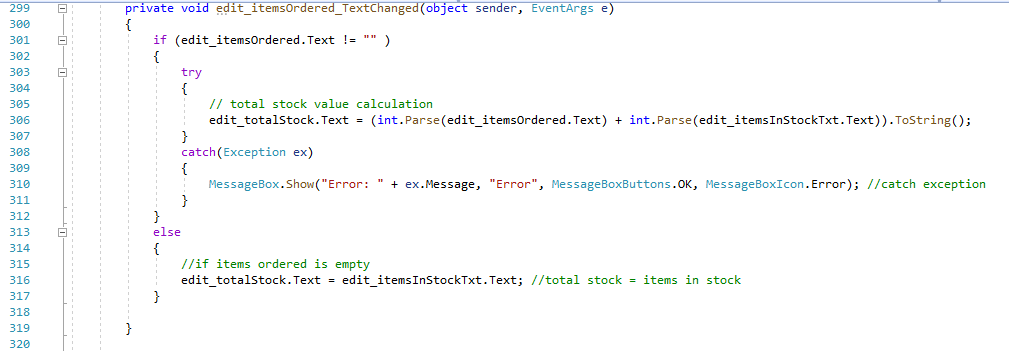


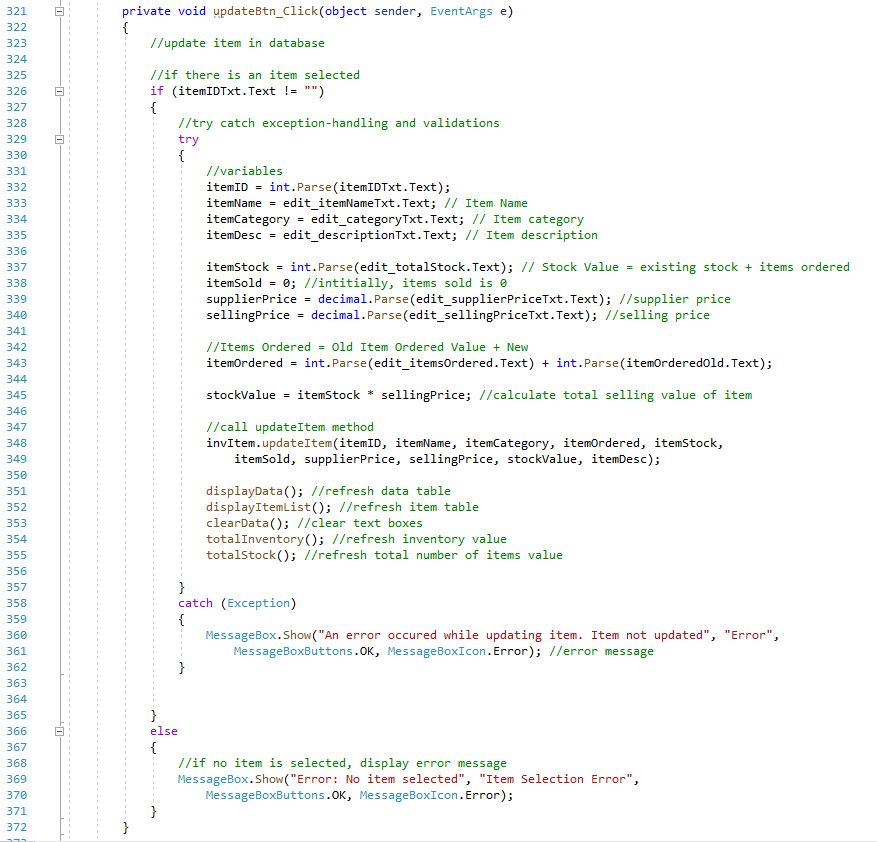


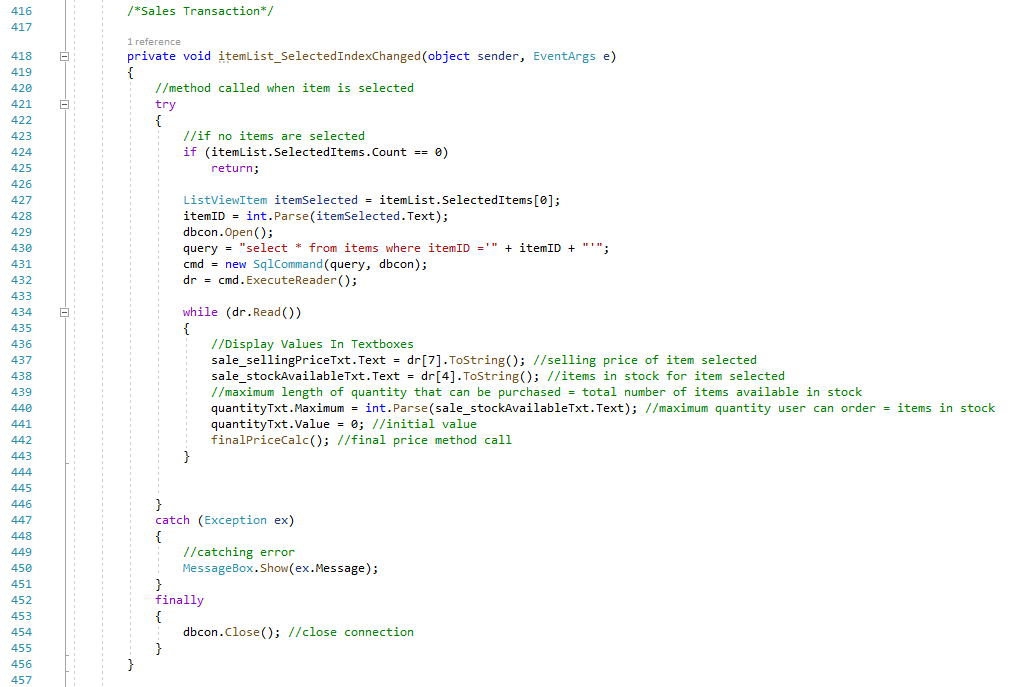
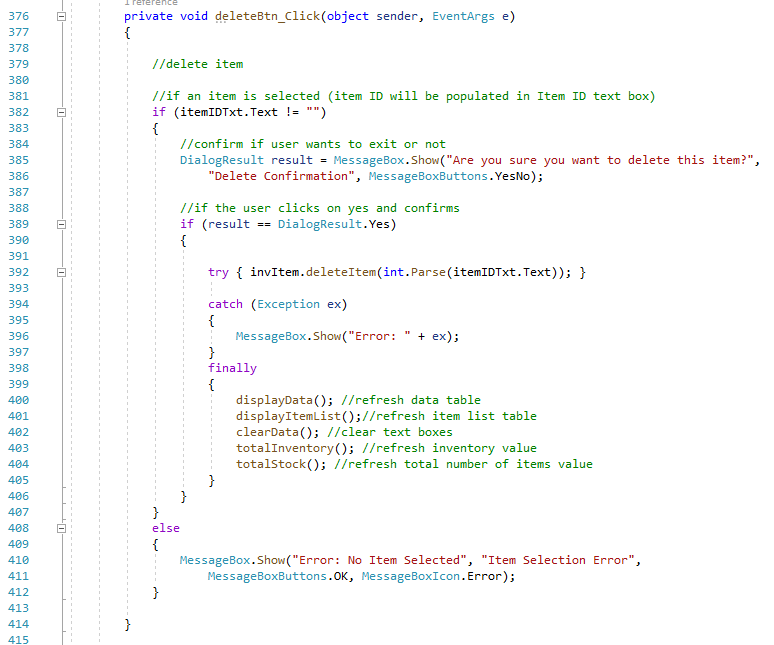


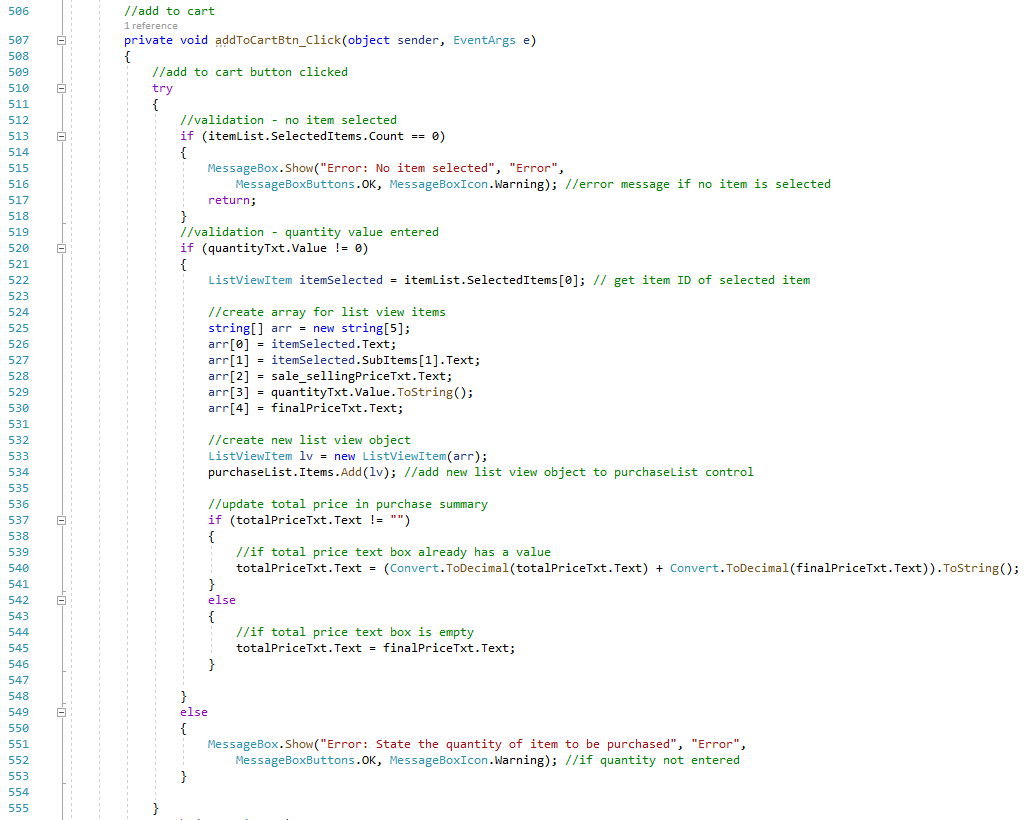




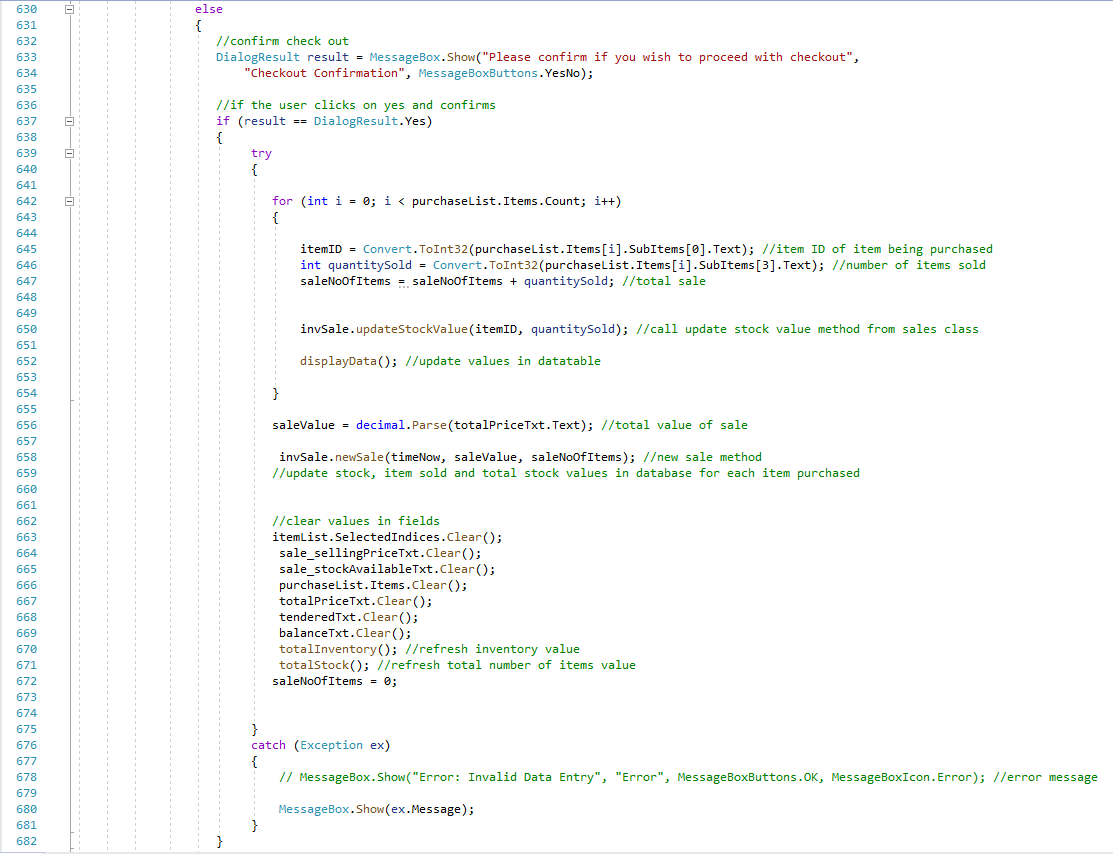
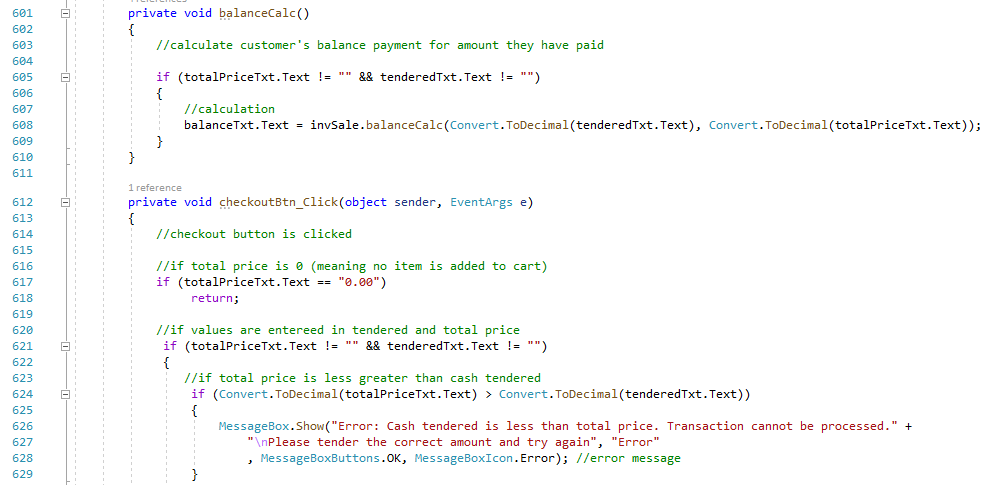








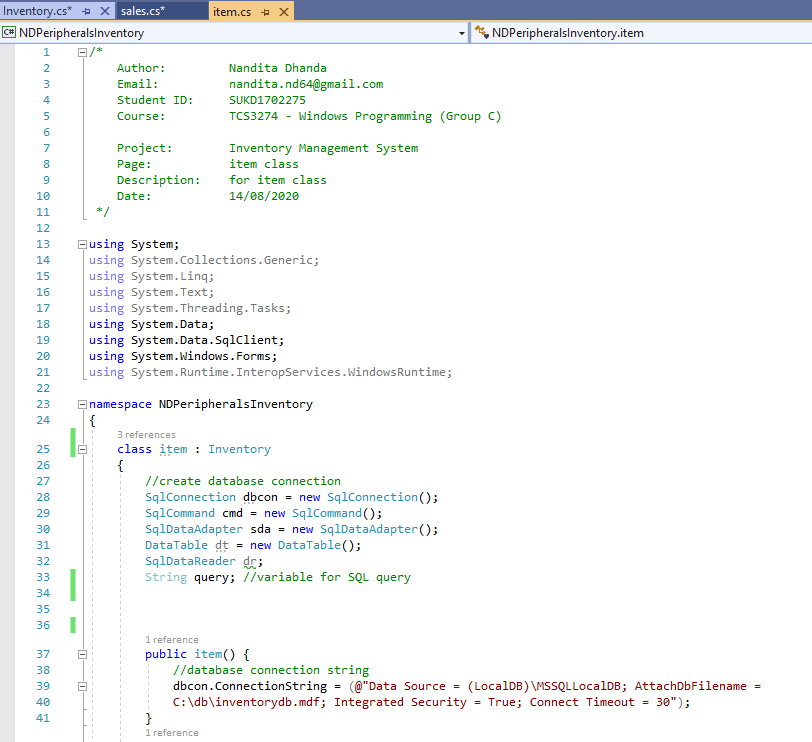


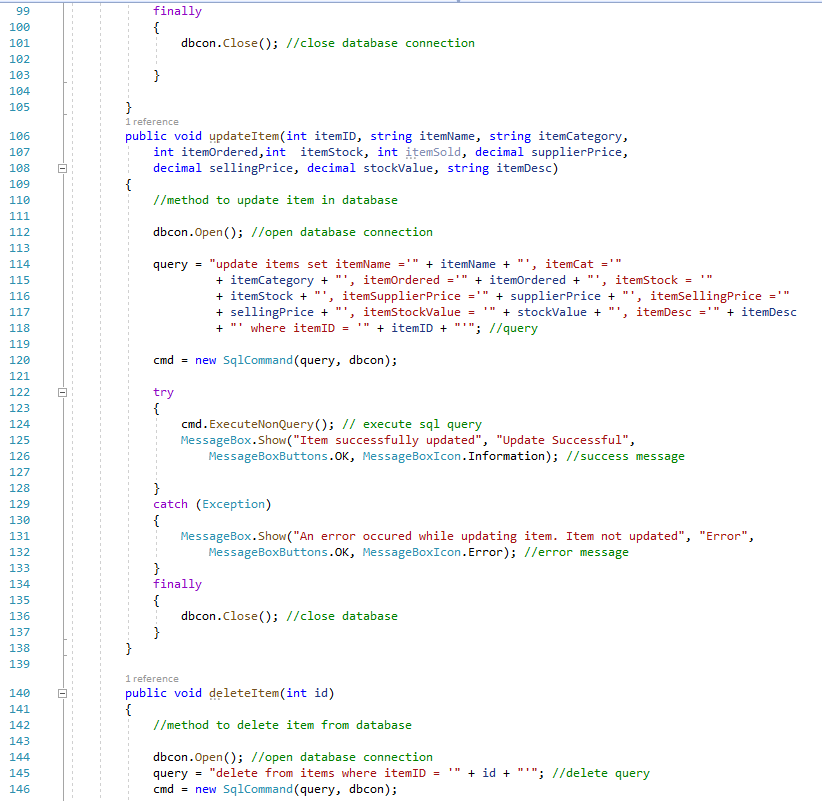


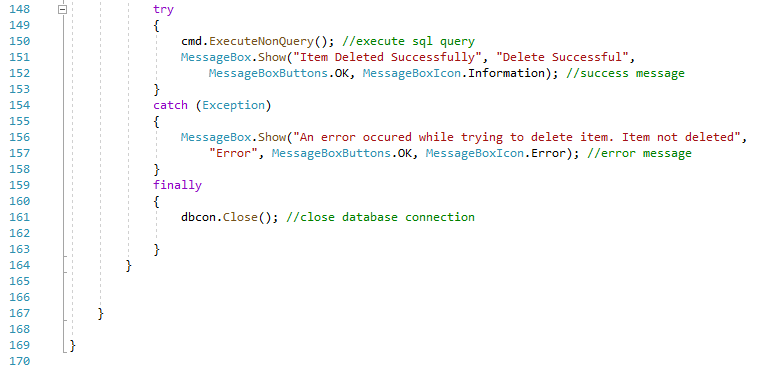




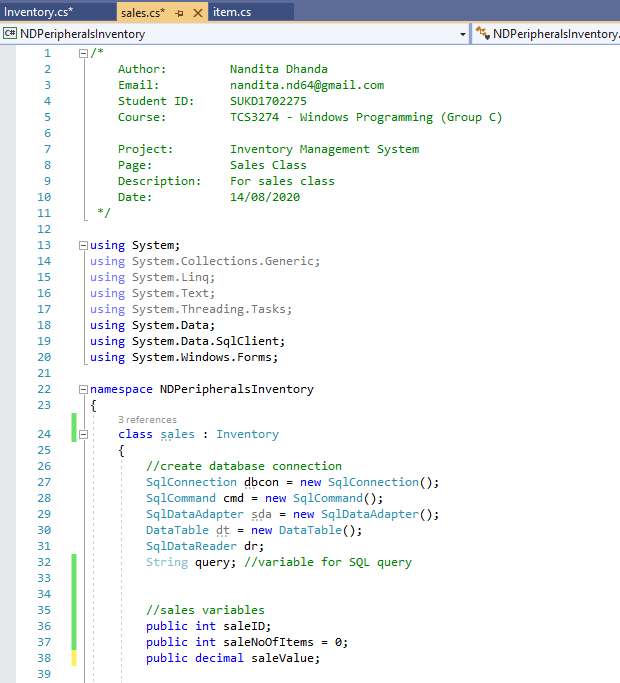
## Section 3 – Item.cs

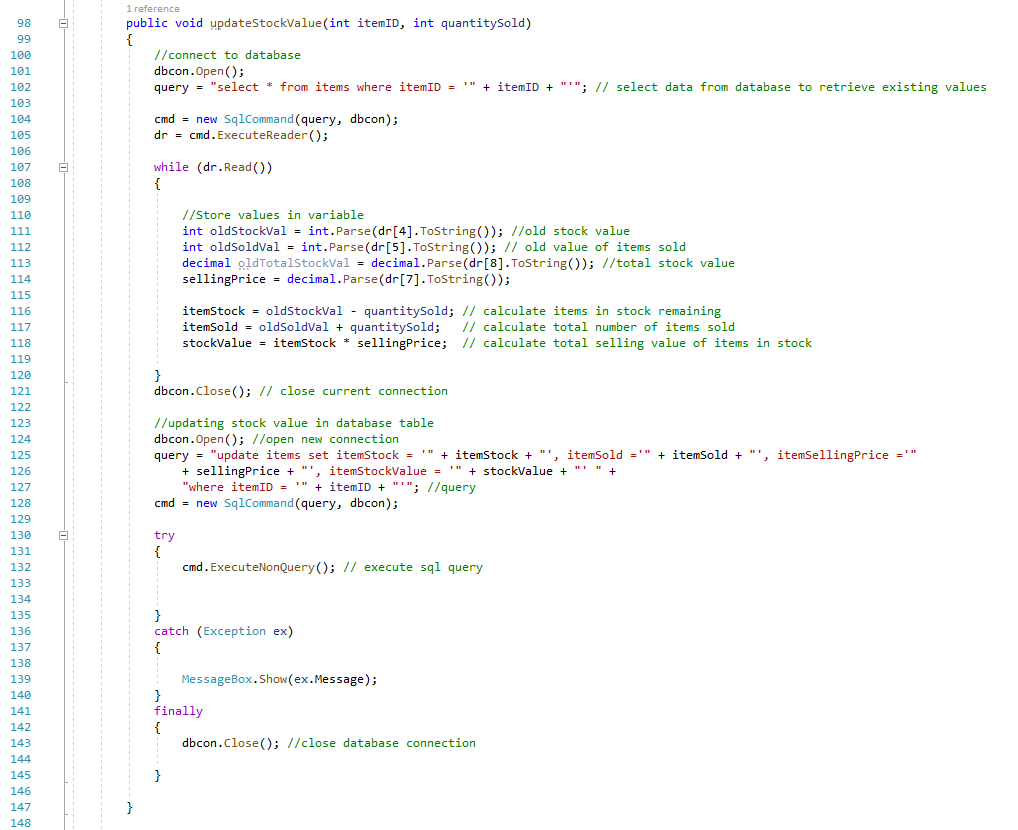
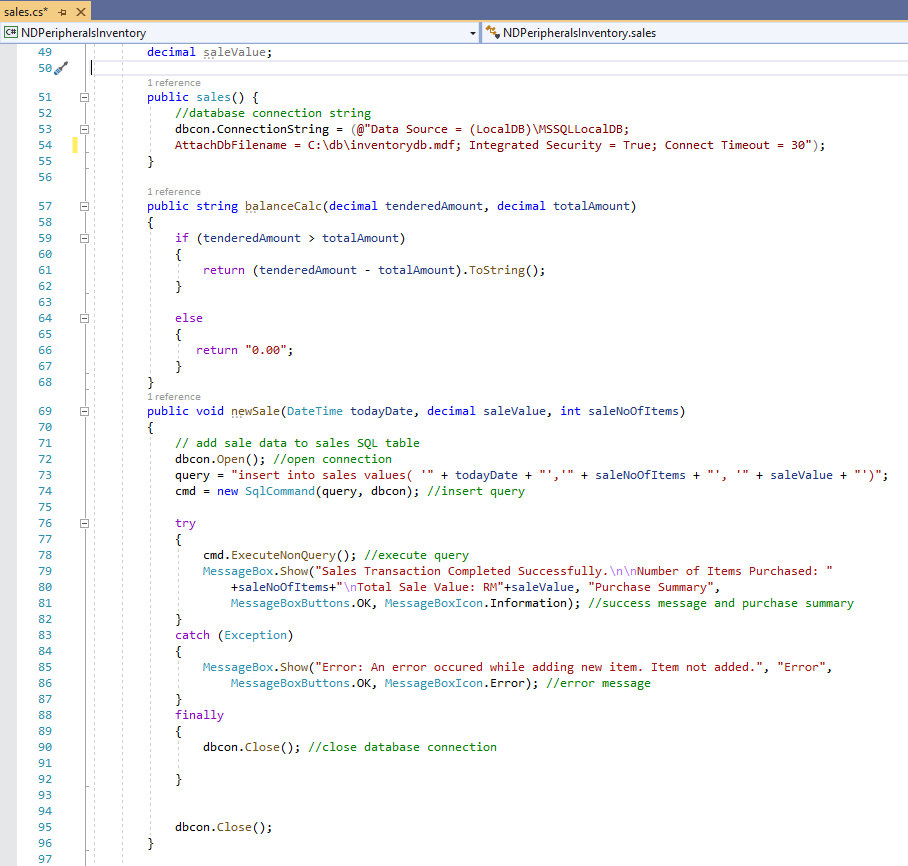




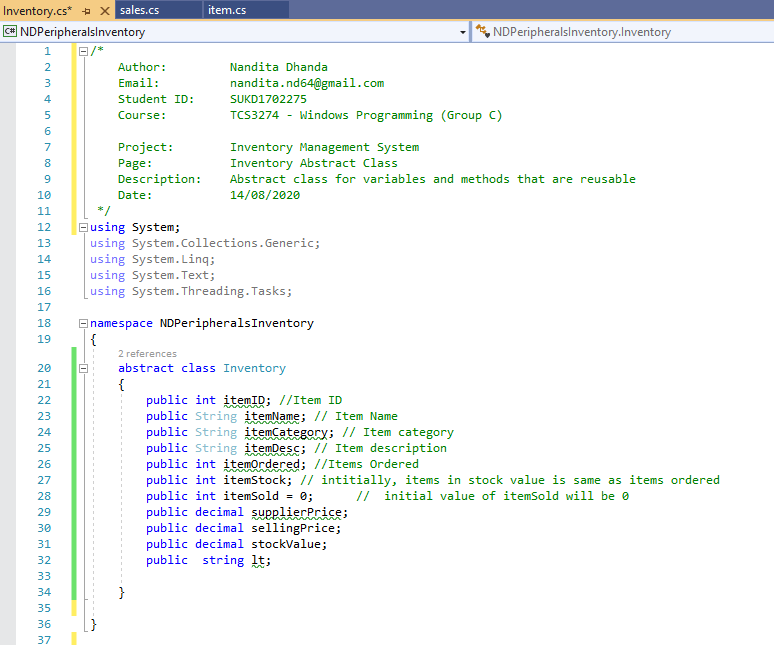


## Section 4 – Sales.cs



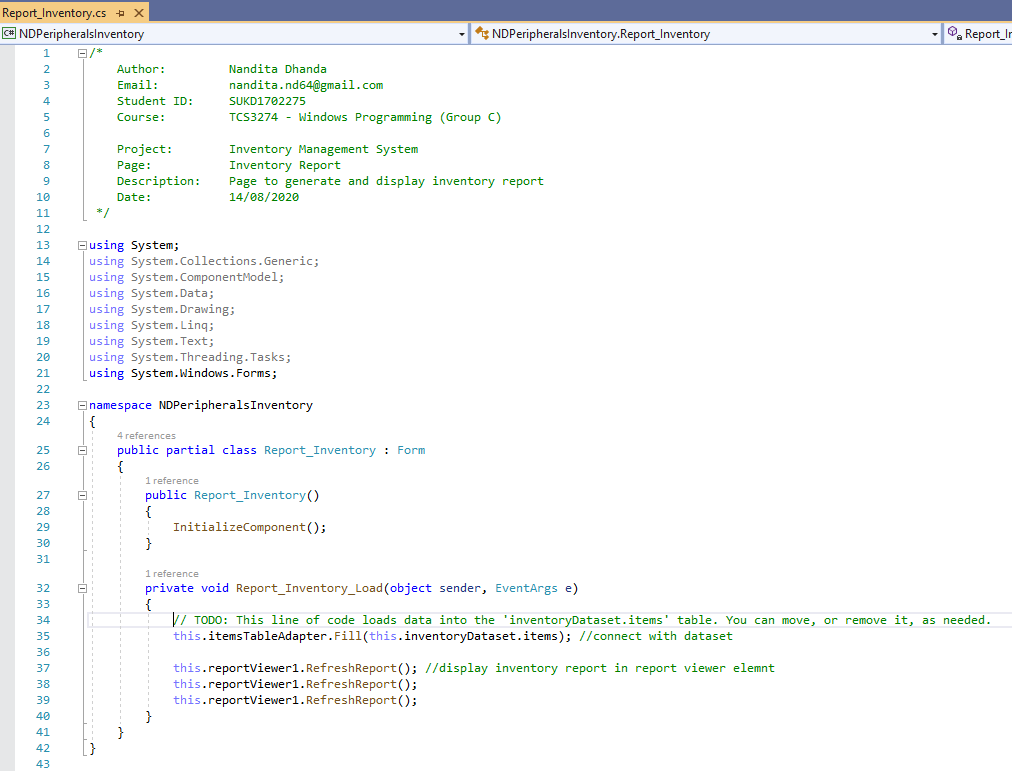


## Section 5 – inventory.cs



## Section 6 – Reports

**Report\_Inventory.cs**



**Reports\_Sales.cs**

