

Topic:	View Inventory Form
Assigned Members:	Presno & Tojino

Opening the View Inventory form this would be initialized:

```

4 references
public viewInventoryForm()
{
    InitializeComponent(); // Initialize form components
    InitializeSortComboBox(); // Set up sorting options in the combo box
    LoadData(); // Load inventory data into the DataGridView
}

```

Sort Options:

The screenshot shows the 'Glorious Sole' application window. On the left is a sidebar with buttons: Dashboard, View Inventory (highlighted), Update Inventory, Invoice Logs, Manage Account, and Logout. The main area displays a 'Sort by' dropdown menu with the following options: 'Price (High to Low)' (selected), 'Recently Updated', 'A-Z', 'Z-A', 'Price (Low to High)', and 'Price (High to Low)'. Below the dropdown is a table of inventory items.

Brand	Name	Quantity	PricePerPiece	Total
Jordan	13000.00	6500
Jordan	...	3	12500.00	1625
Nike	...	15	4500.00	1125
Nike	GT Hustle 2	12	4000.00	4800
Nike	Nike Air Dunk J...	10	4000.00	4000
Jordan	Luka 2 "Nebula"	20	4000.00	8000
Adidas	Harden 7 "Silve...	18	3500.00	6300
Adidas	Trae Young 2 "...	15	3500.00	5250
Adidas	BYW Select "M...	25	3000.00	7500
Nike	PG 4 "PCG Teal"	30	2000.00	6000

This sets the sorting options on the Combo Box:

```

// Method to populate the sorting options in the combo box
1 reference
private void InitializeSortComboBox()
{
    cbSortBy.Items.Add("Recently Updated");
    cbSortBy.Items.Add("A-Z");
    cbSortBy.Items.Add("Z-A");
    cbSortBy.Items.Add("Price (Low to High)");
    cbSortBy.Items.Add("Price (High to Low)");
    cbSortBy.SelectedIndex = 1; // Default sort order is A-Z
}

```

When choosing an option for the Combo Box the LoadData() would load various data from the database and apply its sorting:

Default (A-Z) sorted view:

The screenshot shows a web application window titled 'Glorious Sole'. On the left is a red sidebar with navigation buttons: 'Dashboard', 'View Inventory', 'Update Inventory', 'Invoice Logs', 'Manage Account', and 'Logout'. The main content area has a 'Sort by' dropdown menu set to 'A-Z'. Below it is a table with 7 columns: Brand, Model, Size, Quantity, PricePerPiece, and Total. The table is sorted alphabetically by Brand. The first row is Adidas Trae Young 2 '...', and the last row is Nike PG 4 'PCG Teal'. A scrollbar is visible at the bottom of the table.


	Brand	Model	Size	Quantity	PricePerPiece	Total
▶	Adidas	Trae Young 2 "...	9.50	15	3500.00	5250
	Adidas	BYW Select "M...	10.00	25	3000.00	7500
	Adidas	Harden 7 "Silve...	11.00	18	3500.00	6300
	Jordan	Clot x Jordan 5 ...	10.00	13	12500.00	1625
	Jordan	Jordan 5 "Ragin...	11.00	5	13000.00	6500
	Jordan	Luka 2 "Nebula"	9.50	20	4000.00	8000
	Nike	GT Hustle 2	10.50	12	4000.00	4800
	Nike	Nike Air Dunk J...	9.00	10	4000.00	4000
	Nike	Air Force Pulse	11.00	25	4500.00	1125
	Nike	PG 4 "PCG Teal"	8.50	30	2000.00	6000
*						

Recently Updated sort:

The screenshot shows the same 'Glorious Sole' web application, but the 'Sort by' dropdown menu is now set to 'Recently Updated'. The table data is sorted accordingly, with Nike Air Force Pulse at the top and Adidas Harden 7 'Silve...' at the bottom. The rest of the interface remains the same.

	Brand	Model	Size	Quantity	PricePerPiece	Total
▶	Nike	Air Force Pulse	11.00	25	4500.00	1125
	Adidas	Trae Young 2 "...	9.50	15	3500.00	5250
	Jordan	Clot x Jordan 5 ...	10.00	13	12500.00	1625
	Jordan	Jordan 5 "Ragin...	11.00	5	13000.00	6500
	Nike	GT Hustle 2	10.50	12	4000.00	4800
	Nike	Nike Air Dunk J...	9.00	10	4000.00	4000
	Jordan	Luka 2 "Nebula"	9.50	20	4000.00	8000
	Adidas	BYW Select "M...	10.00	25	3000.00	7500
	Nike	PG 4 "PCG Teal"	8.50	30	2000.00	6000
	Adidas	Harden 7 "Silve...	11.00	18	3500.00	6300
*						

Z-A sort Option:

**Glorious Sole**

Dashboard

View Inventory

Update Inventory

Invoice Logs


Manage Account

Logout

Sort by Z-A

	Brand	Model	Size	Quantity	PricePerPiece	Tota
▶	Nike	GT Hustle 2	10.50	12	4000.00	4800
	Nike	Nike Air Dunk J...	9.00	10	4000.00	4000
	Nike	PG 4 "PCG Teal"	8.50	30	2000.00	6000
	Nike	Air Force Pulse	11.00	25	4500.00	1125
	Jordan	Clot x Jordan 5 ...	10.00	13	12500.00	1625
	Jordan	Jordan 5 "Ragin...	11.00	5	13000.00	6500
	Jordan	Luka 2 "Nebula"	9.50	20	4000.00	8000
	Adidas	BYW Select "M...	10.00	25	3000.00	7500
	Adidas	Harden 7 "Silve...	11.00	18	3500.00	6300
	Adidas	Trae Young 2 "...	9.50	15	3500.00	5250
*						

Price (Low to High) sort Option:

**Glorious Sole**

Dashboard

View Inventory

Update Inventory

Invoice Logs

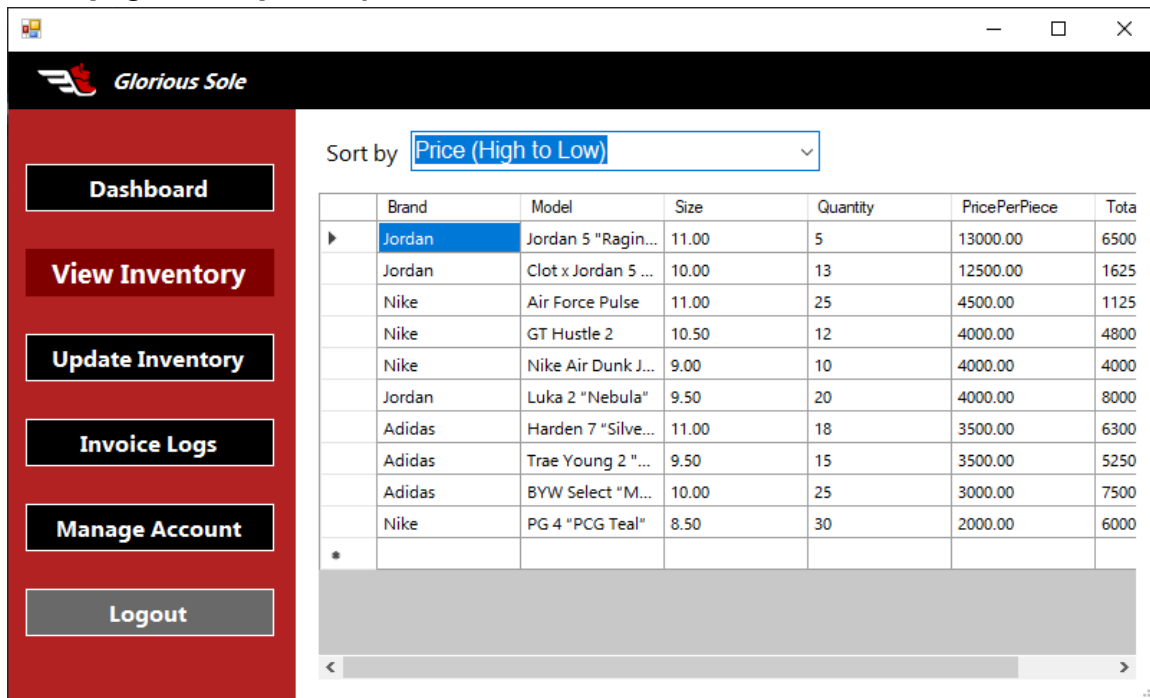
Manage Account

Logout

Sort by Price (Low to High)

	Brand	Model	Size	Quantity	PricePerPiece	Tota
▶	Nike	PG 4 "PCG Teal"	8.50	30	2000.00	6000
	Adidas	BYW Select "M...	10.00	25	3000.00	7500
	Adidas	Harden 7 "Silve...	11.00	18	3500.00	6300
	Adidas	Trae Young 2 "...	9.50	15	3500.00	5250
	Nike	GT Hustle 2	10.50	12	4000.00	4800
	Nike	Nike Air Dunk J...	9.00	10	4000.00	4000
	Jordan	Luka 2 "Nebula"	9.50	20	4000.00	8000
	Nike	Air Force Pulse	11.00	25	4500.00	1125
	Jordan	Clot x Jordan 5 ...	10.00	13	12500.00	1625
	Jordan	Jordan 5 "Ragin...	11.00	5	13000.00	6500
*						

Price (High to Low) sort Option:



	Brand	Model	Size	Quantity	PricePerPiece	Tota
▶	Jordan	Jordan 5 "Ragin...	11.00	5	13000.00	6500
	Jordan	Clot x Jordan 5 ...	10.00	13	12500.00	1625
	Nike	Air Force Pulse	11.00	25	4500.00	1125
	Nike	GT Hustle 2	10.50	12	4000.00	4800
	Nike	Nike Air Dunk J...	9.00	10	4000.00	4000
	Jordan	Luka 2 "Nebula"	9.50	20	4000.00	8000
	Adidas	Harden 7 "Silve...	11.00	18	3500.00	6300
	Adidas	Trae Young 2 "...	9.50	15	3500.00	5250
	Adidas	BYW Select "M...	10.00	25	3000.00	7500
	Nike	PG 4 "PCG Teal"	8.50	30	2000.00	6000
*						

We access the database by using SQL Connections:

```
// Connection string to connect to the local database  
string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename='C:\Users\reyes\Downloads\lms + Desing\lmsDesign-SISON\InventoryManagementSystem-GloriousSole\SQL_InventoryManagementSystem-GloriousSole\GS_IMS.mdf';Integrated Security
```

With the use of SQL Connection, we can access the overview of the Inventory Table from the Database itself thru this code:

```
private void LoadData(string sortOrder = "A-Z")  
{  
    using (SqlConnection con = new SqlConnection(connectionString))  
    {  
        using (SqlCommand cm = new SqlCommand())  
        {  
            cm.Connection = con;  
  
            try  
            {  
                con.Open(); // Open the database connection  
                string query = @"SELECT Brand, Model, Size, Quantity, PricePerPiece, (Quantity * PricePerPiece) AS TotalPrice FROM Inventory";  
            }  
        }  
    }  
}
```

For them to show we use Switch Case to determine the option chosen by the user:

```
// Determine the sorting order based on user selection
switch (sortOrder)
{
    case "A-Z":
        query += " ORDER BY Brand ASC"; // Sort by Brand ascending
        break;
    case "Z-A":
        query += " ORDER BY Brand DESC"; // Sort by Brand descending
        break;
    case "Price (Low to High)":
        query += " ORDER BY PricePerPiece ASC"; // Sort by Price ascending
        break;
    case "Price (High to Low)":
        query += " ORDER BY PricePerPiece DESC"; // Sort by Price descending
        break;
    case "Recently Updated":
        query += " ORDER BY UpdatedAt DESC"; // Sort by most recently updated
        break;
    default:
        break;
}
```

Upon which sorting option does the user picks it would set the SQL command for the query then automatically updates the DataGrid View:

```
cm.CommandText = query; // Set the SQL command text

SqlDataAdapter da = new SqlDataAdapter(cm);
DataTable dt = new DataTable();
da.Fill(dt); // Fill the DataTable with data from the database
dgvInventoryView.DataSource = dt; // Bind the DataTable to the DataGridView
```

And lastly, a catch block would catch any errors that has occurred

```
catch (Exception ex)
{
    MessageBox.Show("Error: " + ex.Message); // Show any errors that occur
}
```