Retail Inventory and Sales Management

http://cb-ot-devst05.ad.wsu.edu/MF03c.fostanes/Assignments/Final%20Project/Final%20Project.aspx

Christian Fostanes

MIS 325

December 13, 2020

Overview:

This project is a simple transaction recorder and inventory management program. It is meant to help small businesses become more automated with simple transactions and managing the inventory. It also includes sales report elements like summary tables and charts to help a business owner or a manager determine which items are the best-selling and how much revenue the business has generated.

Problem and Customer:

Some small businesses still rely on physical documents to keep track of their inventory. While this is still a viable way of keeping records and managing inventory, it is better to have an automated way manage inventory in order to keep up with today's technological demands. Also, keeping count of how many units of each item are left and which ones are the most popular can be tedious.

The targeted customer demographic of this project is small businesses. These small businesses are usually run by a single person or a small team of people. The use of the program I have created will help ease the workload of running a business.

Solution and Benefits of Project:

This program would eliminate the need to keep physical documents of inventory items. It will automate the inventory management process in order to keep up with modern technological demands. The program allows for the calculation and recording of a transaction. It also keeps tracks of simple metrics such as total revenue generated and how many units of each item and in total have been sold. It also automatically creates column and pie charts for an easier visual representation of aforementioned metrics. Lastly, the program allows the user to manage the inventory by adding and removing items from the inventory list and also updating the prices of existing items.

It is a simple program to aid a small business that still relies on physical documents to keep track of sales and inventory. It also eliminates the need to perform manual calculations for a transaction. Simple sales reports are automatically created to help user determine which items are the best sellers.

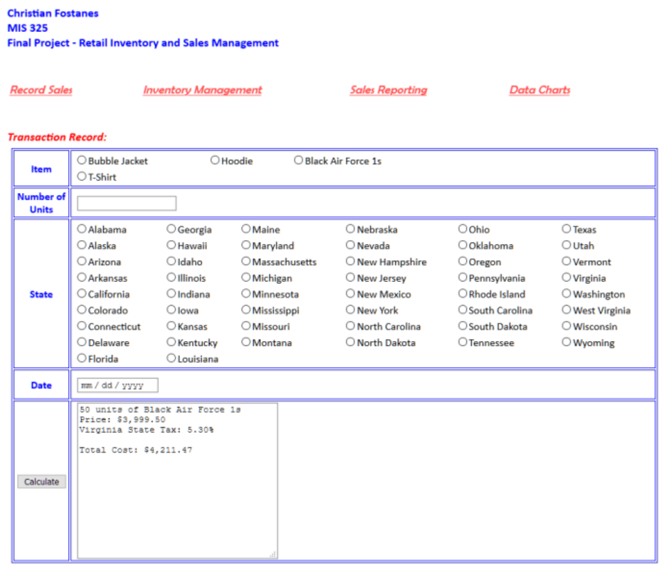
Program Functionality:

The program has a total of 4 tabs, each displaying a different functionality. The first tab is for performing and recording transactions. The user will pick and inventory item from a list and input how many units of that item is to be purchased. Next, the user will pick a state from the list so sales tax can be applied to the transaction. Lastly, a date must be inputted for the sake of record-keeping. After the transaction is done, a textbox will show the details and a log of the transaction will be recorded.

On the second tab, the user will be able to view and manage inventory. The default stock of all items in the inventory is set to 100 because that seems like a decent starting point. Whenever an item is purchased, the inventory is decreased. Currently, there are only three items in the inventory list as an example. In this tab, the user will be able to add new items to the inventory by entering the item name and the price for the item. The new item will be added to the list that is shown in the first tab, in the inventory list in the current tab and in the sales reporting tab. The user will also be able to update the price of an existing item. To update the price of an item, the user will first need the select the item from the list and enter the desired price for it. Lastly, the user will be able remove item from the inventory. Just like with adding new items, removing items will remove the item from the list in the first tab. To remove an item from the inventory, the user will first need to select which item from the list to remove.

The third tab of the program contains summary tables that keep track of metrics such as units sold, and generated revenue. The first table keeps track of grand totals. The second and third tables also keep track of the same things but categorized by each item and state, respectively.

Finally, the fourth tab contains charts that are based on the tables in the third tab. These visual charts contain information that might help the user determine which items are the most popular and which ones generate the most revenue.



Christian Fostanes
MIS 325
Final Project - Retail Inventory and Sales Management

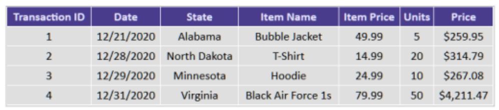
Record Sales *Inventory Management* *Sales Reporting* *Data Charts*

Transaction Record:

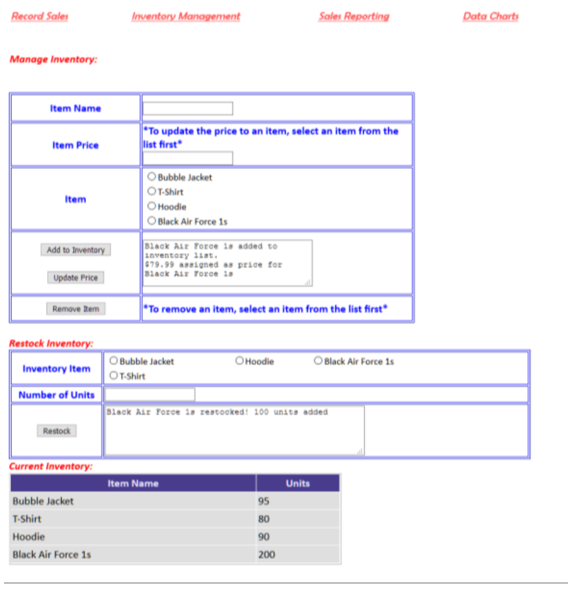
Item	○ Bubble Jacket	○ Hoodie	○ Black Air Force 1s			
	○ T-Shirt					
Number of Units						
State	○ Alabama	○ Georgia	○ Maine	○ Nebraska	○ Ohio	○ Texas
	○ Alaska	○ Hawaii	○ Maryland	○ Nevada	○ Oklahoma	○ Utah
	○ Arizona	○ Idaho	○ Massachusetts	○ New Hampshire	○ Oregon	○ Vermont
	○ Arkansas	○ Illinois	○ Michigan	○ New Jersey	○ Pennsylvania	○ Virginia
	○ California	○ Indiana	○ Minnesota	○ New Mexico	○ Rhode Island	○ Washington
	○ Colorado	○ Iowa	○ Mississippi	○ New York	○ South Carolina	○ West Virginia
	○ Connecticut	○ Kansas	○ Missouri	○ North Carolina	○ South Dakota	○ Wisconsin
	○ Delaware	○ Kentucky	○ Montana	○ North Dakota	○ Tennessee	○ Wyoming
	○ Florida	○ Louisiana				
Date	mm / dd / yyyy					
Calculate						

```
50 units of Black Air Force 1s
Price: $3,999.50
Virginia State Tax: 5.30%

Total Cost: $4,211.47
```



Transaction ID	Date	State	Item Name	Item Price	Units	Price
1	12/21/2020	Alabama	Bubble Jacket	49.99	5	$259.95
2	12/28/2020	North Dakota	T-Shirt	14.99	20	$314.79
3	12/29/2020	Minnesota	Hoodie	24.99	10	$267.08
4	12/31/2020	Virginia	Black Air Force 1s	79.99	50	$4,211.47



Manage Inventory:

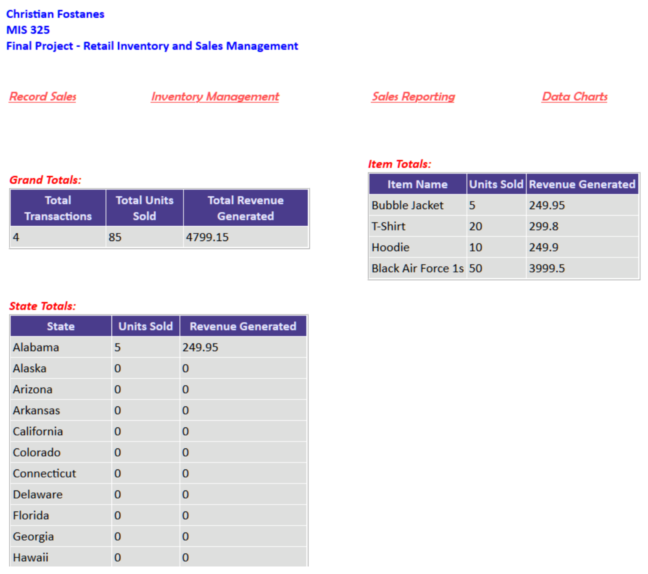
Item Name	
Item Price	***To update the price to an item, select an item from the list first***
Item	○ Bubble Jacket ○ T-Shirt ○ Hoodie ○ Black Air Force 1s
Add to Inventory Update Price	Black Air Force 1s added to inventory list. $79.99 assigned as price for Black Air Force 1s
Remove Item	***To remove an item, select an item from the list first***

Restock Inventory:

Inventory Item	○ Bubble Jacket ○ Hoodie ○ Black Air Force 1s ○ T-Shirt
Number of Units	
Restock	Black Air Force 1s restocked! 100 units added

Current Inventory:

Item Name	Units
Bubble Jacket	95
T-Shirt	80
Hoodie	90
Black Air Force 1s	200



Christian Fostanes
MIS 325
Final Project - Retail Inventory and Sales Management

Grand Totals:

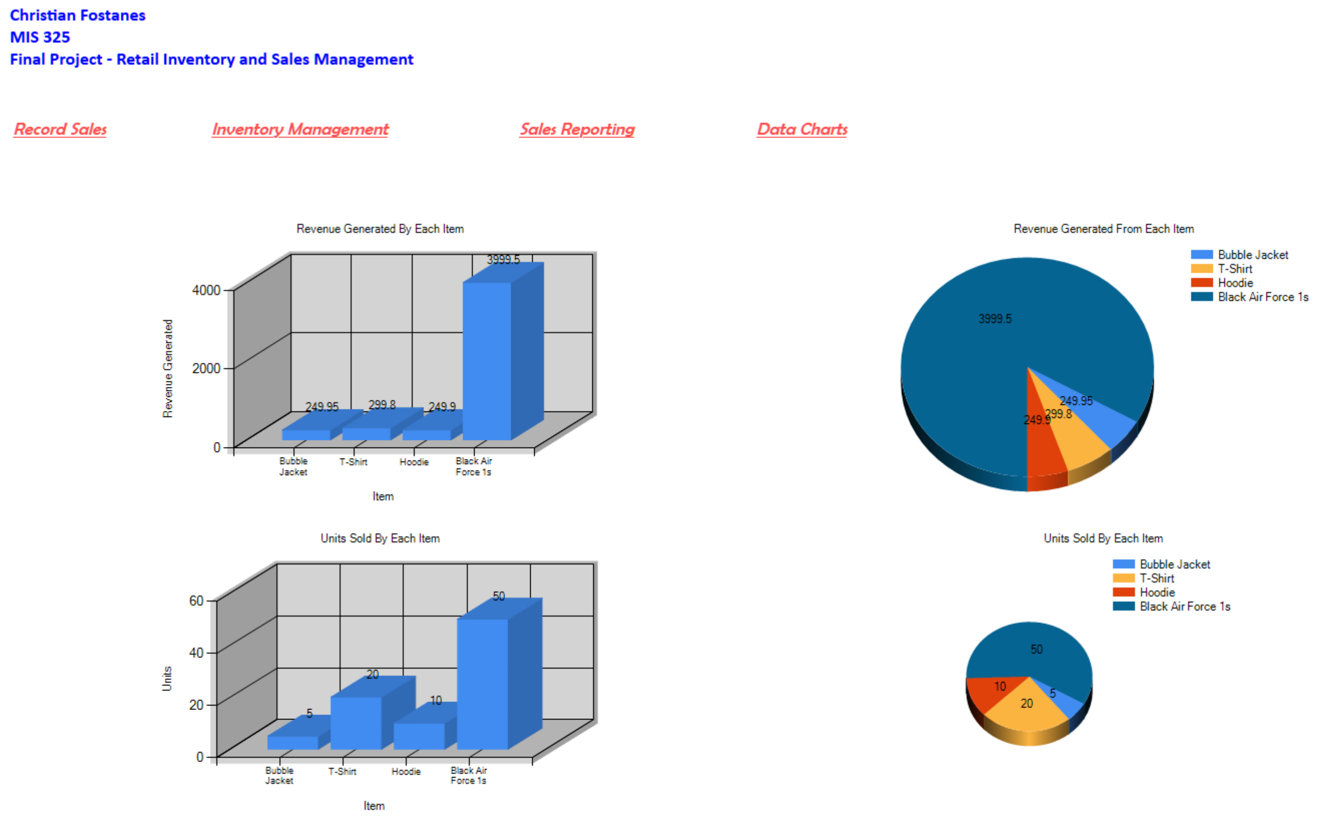
Total Transactions	Total Units Sold	Total Revenue Generated
4	85	4799.15

Item Totals:

Item Name	Units Sold	Revenue Generated
Bubble Jacket	5	249.95
T-Shirt	20	299.8
Hoodie	10	249.9
Black Air Force 1s	50	3999.5

State Totals:

State	Units Sold	Revenue Generated
Alabama	5	249.95
Alaska	0	0
Arizona	0	0
Arkansas	0	0
California	0	0
Colorado	0	0
Connecticut	0	0
Delaware	0	0
Florida	0	0
Georgia	0	0
Hawaii	0	0



Christian Fostanes
MIS 325
Final Project - Retail Inventory and Sales Management

```
Imports System.Data
Partial Class Assignments_Final_Project_Final_Project
    Inherits System.Web.UI.Page
    Public Shared dtSalesRecord, dtInventory, dtItemTotals, dtCategoryTotals,
dtStateTotals, dtGrandTotals As New DataTable

    'Handles the switching of different views'
    Protected Sub recordSalesLink_Click(sender As Object, e As EventArgs) Handles
recordSalesLink.Click
        MultiView1.ActiveViewIndex = 0
    End Sub
    Protected Sub inventoryLink_Click(sender As Object, e As EventArgs) Handles
inventoryLink.Click
        MultiView1.ActiveViewIndex = 1
    End Sub
    Protected Sub salesReportLink_Click(sender As Object, e As EventArgs) Handles
salesReportLink.Click
        MultiView1.ActiveViewIndex = 2
    End Sub
    Protected Sub dataChartsLink_Click(sender As Object, e As EventArgs) Handles
dataChartsLink.Click
        MultiView1.ActiveViewIndex = 3
        DrawChart()
    End Sub

    'Creates data tables'
    Private Sub Assignments_Final_Project_Final_Project_Init(sender As Object, e As
EventArgs) Handles Me.Init
        'Ensures that the data tables are only created once'
        If dtInventory.Columns.Count > 0 Then
            Exit Sub
        ElseIf dtItemTotals.Columns.Count > 0 Then
            Exit Sub
        ElseIf dtCategoryTotals.Columns.Count > 0 Then
            Exit Sub
        ElseIf dtStateTotals.Columns.Count > 0 Then
            Exit Sub
        ElseIf dtGrandTotals.Columns.Count > 0 Then
            Exit Sub
        End If

        'Creates the columns for sales record data table'
        With dtSalesRecord.Columns
            .Add("Transaction ID", GetType(Integer))
            .Add("Date", GetType(String))
            .Add("State", GetType(String))
            .Add("Item Name", GetType(String))
            .Add("Item Price", GetType(String))
            .Add("Units", GetType(Integer))
            .Add("Price", GetType(String))
        End With
        'Auto increments the first column by 1'
        With dtSalesRecord.Columns("Transaction ID")
            .AutoIncrement = True
            .AutoIncrementSeed = 1
            .AutoIncrementStep = 1
        End With
        'Creates columns for inventory data table'
        With dtInventory.Columns
            .Add("Item Name", GetType(String))
            .Add("Units", GetType(Integer))
        End With
        'Sets default values to a number. I chose to put 100 because that's a decent
amount of stock to begin with'
        dtInventory.Columns("Units").DefaultValue = 100
        'Takes items in list and turns it to a new row'
        For Each liInventory As ListItem In itemList.Items
            Dim drInventory As DataRow = dtInventory.NewRow
            drInventory.Item("Item Name") = liInventory.Text
            dtInventory.Rows.Add(drInventory)
        Next
        'Outputs the data table'
        inventoryGrid.DataSource = dtInventory
        inventoryGrid.DataBind()

        'Creates column for indiviudal items data table'
        With dtItemTotals.Columns
            .Add("Item Name", GetType(String))
            .Add("Units Sold", GetType(Integer))
            .Add("Revenue Generated", GetType(Decimal))
        End With
        'Sets the default values to 0'
        dtItemTotals.Columns("Units Sold").DefaultValue = 0
        dtItemTotals.Columns("Revenue Generated").DefaultValue = 0
        'Creates rows for each item in list'
        For Each liItem As ListItem In itemList.Items
            Dim drItem As DataRow = dtItemTotals.NewRow
            drItem.Item("Item Name") = liItem.Text
            dtItemTotals.Rows.Add(drItem)
        Next
        'Outputs the data table'
        itemGrid.DataSource = dtItemTotals
        itemGrid.DataBind()

        'Creates the columns for state totals data table'
        With dtStateTotals.Columns
            .Add("State", GetType(String))
            .Add("Units Sold", GetType(Integer))
            .Add("Revenue Generated", GetType(Decimal))
        End With
        'Sets default values to 0'
        dtStateTotals.Columns("Units Sold").DefaultValue = 0
        dtStateTotals.Columns("Revenue Generated").DefaultValue = 0
        'Creates row for each state'
        For Each liState As ListItem In stateList.Items
            Dim drState As DataRow = dtStateTotals.NewRow
            drState.Item("State") = liState.Text
            dtStateTotals.Rows.Add(drState)
        Next
        'Outputs data table'
        stateGrid.DataSource = dtStateTotals
        stateGrid.DataBind()

        'Creates columns for grand totals data table'
        With dtGrandTotals.Columns
```

```
            .Add("Total Transactions", GetType(Integer))
            .Add("Total Units Sold", GetType(Integer))
            .Add("Total Revenue Generated", GetType(Decimal))
        End With
        'Sets default values to 0'
        dtGrandTotals.Columns("Total Transactions").DefaultValue = 0
        dtGrandTotals.Columns("Total Units Sold").DefaultValue = 0
        dtGrandTotals.Columns("Total Revenue Generated").DefaultValue = 0
        'dtGrandTotals.Columns("Best Selling Item").DefaultValue = 0
        'Creates a row for the grand totals data table'
        Dim drGrandTotals As DataRow = dtGrandTotals.NewRow
        dtGrandTotals.Rows.Add(drGrandTotals)
        'Outputs data table'
        totalsGrid.DataSource = dtGrandTotals
        totalsGrid.DataBind()
    End Sub

    'Creates and configures charts'
    Private Sub DrawChart()
        'Configues item revenue column chart'
        With itemRevenueColumn
            .DataSource = dtItemTotals
            .DataBind()
            .Titles.Add("Revenue Generated By Each Item")
        End With
        'Creates the titles for x and y axis'
        With itemRevenueColumn.ChartAreas(0)
            .AxisX.Title = "Item"
            .AxisY.Title = "Revenue Generated"
            .Area3DStyle.Enable3D = True
        End With
        'Assigning which column in data table is x and y axis'
        With itemRevenueColumn.Series("Series1")
            .ChartType = DataVisualization.Charting.SeriesChartType.Column
            .XValueMember = "Item Name"
            .YValueMembers = "Revenue Generated"
            .IsValueShownAsLabel = True
        End With

        'Configures the item revenue pie chart'
        With itemRevenuePie
            .DataSource = dtItemTotals
            .DataBind()
            .Titles.Add("Revenue Generated From Each Item")
            .Legends.Add("Legend1")
            .Legends("Legend1").Enabled = True
            .ChartAreas(0).Area3DStyle.Enable3D = True
        End With
        With itemRevenuePie.Series("Series1")
            .ChartType = DataVisualization.Charting.SeriesChartType.Pie
            .XValueMember = "Item Name"
            .YValueMembers = "Revenue Generated"
            .IsValueShownAsLabel = True
            .IsVisibleInLegend = True

            'Configures item units sold column chart'
            With itemUnitsColumn
                .DataSource = dtItemTotals
                .DataBind()
                .Titles.Add("Units Sold By Each Item")
            End With
            'Creates the titles for x and y axis'
            With itemUnitsColumn.ChartAreas(0)
                .AxisX.Title = "Item"
                .AxisY.Title = "Units"
                .Area3DStyle.Enable3D = True
            End With
            'Assigning which column in data table is x and y axis'
            With itemUnitsColumn.Series("Series1")
                .ChartType = DataVisualization.Charting.SeriesChartType.Column
                .XValueMember = "Item Name"
                .YValueMembers = "Units Sold"
                .IsValueShownAsLabel = True
            End With

            'Configures the item units sold pie chart'
            With itemUnitsPie
                .DataSource = dtItemTotals
                .DataBind()
                .Titles.Add("Units Sold By Each Item")
                .Legends.Add("Legend1")
                .Legends("Legend1").Enabled = True
                .ChartAreas(0).Area3DStyle.Enable3D = True
            End With
            With itemUnitsPie.Series("Series1")
                .ChartType = DataVisualization.Charting.SeriesChartType.Pie
                .XValueMember = "Item Name"
                .YValueMembers = "Units Sold"
                .IsValueShownAsLabel = True
                .IsVisibleInLegend = True
            End With
        End With
    End Sub

    'Handles the calculation and recording of transaction'
    Protected Sub calculateBtn_Click(sender As Object, e As EventArgs) Handles
calculateBtn.Click
        Dim totalPrice, beforeTax As Decimal
        'User error check'
        If itemList.SelectedIndex = -1 Then
            outputTxt.Text = "Please select an item"
        ElseIf purchaseUnitsTxt.Text = Nothing Then
            outputTxt.Text = "Please enter a number for units to be purchased"
        ElseIf purchaseUnitsTxt.Text < 0 Or IsNumeric(purchaseUnitsTxt.Text) = False Then
            outputTxt.Text = "Please enter a positive numeric value for the units to be
purhased"
        ElseIf stateList.SelectedIndex = -1 Then
            outputTxt.Text = "Please select a state"
        ElseIf dateTxt.Text = Nothing Then
            outputTxt.Text = "Please enter a date for the purchase"
        Else
            With dtInventory.Rows(itemList.SelectedIndex)
                'Checks if selected item is sold out'
                If .Item("Units") = 0 Then
                    outputTxt.Text = "That item is out of stock. Please restock or select
a different item"
```

```
                    Exit Sub
                    'Checks if number of units to be purchased is more than current
stock'
                ElseIf purchaseUnitsTxt.Text > .Item("Units") Then
                    outputTxt.Text = "There's not enough stock for that purchase. Please
decrease units to purchase"
                    Exit Sub
                Else
                    'Calculates total price'
                    .Item("Units") -= purchaseUnitsTxt.Text
                    beforeTax = itemList.SelectedItem.Value * purchaseUnitsTxt.Text
                    totalPrice = beforeTax + (beforeTax * (stateList.SelectedItem.Value /
100))

                    'Outputs purchase details to textbox on webpage'
                    outputTxt.Text = purchaseUnitsTxt.Text & " units of " &
itemList.SelectedItem.Text & vbNewLine & "Price: " & FormatCurrency(beforeTax, 2) &
                        vbNewLine & stateList.SelectedItem.Text & " State Tax: " &
Convert.ToDecimal(stateList.SelectedItem.Value).ToString("N2") &
                        "%" & vbNewLine & vbNewLine & "Total Cost: " &
FormatCurrency(totalPrice, 2)

                    'Updates the sales record data table'
                    Dim drInvoice As DataRow = dtSalesRecord.NewRow
                    drInvoice.Item("Date") =
DateTime.Parse(dateTxt.Text).ToString("MM/dd/yyyy")
                    drInvoice.Item("State") = stateList.SelectedItem.Text
                    drInvoice.Item("Item Name") = itemList.SelectedItem.Text
                    drInvoice.Item("Item Price") = itemList.SelectedItem.Value
                    drInvoice.Item("Units") = purchaseUnitsTxt.Text
                    drInvoice.Item("Price") = FormatCurrency(totalPrice, 2)
                    dtSalesRecord.Rows.Add(drInvoice)
                    invoiceGrid.DataSource = dtSalesRecord
                    invoiceGrid.DataBind()
                    If .Item("Units") < 50 Then
                        outputTxt.Text &= vbNewLine & vbNewLine &
itemList.SelectedItem.Text & " stock is running low. Please restock soon"
                    End If
                End If
            End With
            inventoryGrid.DataSource = dtInventory
            inventoryGrid.DataBind()

            'Updates the item totals data table'
            With dtItemTotals.Rows(itemList.SelectedIndex)
                .Item("Units Sold") += purchaseUnitsTxt.Text
                .Item("Revenue Generated") += beforeTax
            End With
            itemGrid.DataSource = dtItemTotals
            itemGrid.DataBind()

            'Updates the state totals data table'
            With dtStateTotals.Rows(stateList.SelectedIndex)
                .Item("Units Sold") += purchaseUnitsTxt.Text
                .Item("Revenue Generated") += beforeTax
            End With
            stateGrid.DataSource = dtStateTotals
            stateGrid.DataBind()
            'Updates the grand totals data table'
            With dtGrandTotals.Rows(0)
                .Item("Total Transactions") += 1
                .Item("Total Units Sold") += purchaseUnitsTxt.Text
                .Item("Total Revenue Generated") += beforeTax
            End With
            totalsGrid.DataSource = dtGrandTotals
            totalsGrid.DataBind()
            'Clear the form after a transaction is recorded'
            itemList.SelectedIndex = -1
            purchaseUnitsTxt.Text = Nothing
            stateList.SelectedIndex = -1
            dateTxt.Text = Nothing
        End If
    End Sub

    'Handles the restocking of inventory'
    Protected Sub restockBtn_Click(sender As Object, e As EventArgs) Handles
restockBtn.Click
        'User error check"
        If restockItemList.SelectedIndex = -1 Then
            restockOutputTxt.Text = "Please select an item to restock"
        ElseIf restockUnitsTxt.Text < 0 Or IsNumeric(restockUnitsTxt.Text) = False Then
            restockOutputTxt.Text = "Please enter a positive numerical value"
        Else
            With dtInventory.Rows(restockItemList.SelectedIndex)
                .Item("Units") += restockUnitsTxt.Text
            End With
            inventoryGrid.DataSource = dtInventory
            inventoryGrid.DataBind()
            restockOutputTxt.Text = restockItemList.SelectedItem.Text & " restocked! " &
restockUnitsTxt.Text & " units added"
        End If
        'Clears the form after a restock is performed'
        restockItemList.SelectedIndex = -1
        restockUnitsTxt.Text = Nothing
    End Sub

    'Handles the addition of items to inventory list'
    Protected Sub inventoryAddBtn_Click(sender As Object, e As EventArgs) Handles
inventoryAddBtn.Click
        'User error check'
        If itemAddTxt.Text = Nothing Then
            inventoryMgmtTxt.Text = "Please enter a name for the item"
        ElseIf itemPriceTxt.Text = Nothing Then
            inventoryMgmtTxt.Text = "Please enter a price for the item"
        ElseIf itemPriceTxt.Text < 0 Or IsNumeric(itemPriceTxt.Text) = False Then
            inventoryMgmtTxt.Text = "Please enter a positive numeric value for the item
price"
        Else
            'Add the new item to lists'
            manageItemList.Items.Add(itemAddTxt.Text)
            itemList.Items.Add(itemAddTxt.Text)
            restockItemList.Items.Add(itemAddTxt.Text)
            manageItemList.Items.FindByText(itemAddTxt.Text).Value =
Convert.ToDecimal(itemPriceTxt.Text)
```

```
            itemList.Items.FindByText(itemAddTxt.Text).Value =
Convert.ToDecimal(itemPriceTxt.Text)

            'Add the new item to data tables'
            Dim drNewItem As DataRow = dtInventory.NewRow
            Dim drNewItem2 As DataRow = dtItemTotals.NewRow
            drNewItem.Item("Item Name") = itemAddTxt.Text
            drNewItem2.Item("Item Name") = itemAddTxt.Text
            dtInventory.Rows.Add(drNewItem)
            dtItemTotals.Rows.Add(drNewItem2)
            inventoryGrid.DataSource = dtInventory
            inventoryGrid.DataBind()
            itemGrid.DataSource = dtItemTotals
            itemGrid.DataBind()

            inventoryMgmtTxt.Text = itemAddTxt.Text & " added to inventory list." &
vbNewLine & FormatCurrency(itemPriceTxt.Text, 2) &
                " assigned as price for " & itemAddTxt.Text
            itemAddTxt.Text = Nothing
            itemPriceTxt.Text = Nothing
            manageItemList.SelectedIndex = -1
        End If
    End Sub

    'Handles the addition/updating of price of the an item'
    Protected Sub itemValueAddBtn_Click(sender As Object, e As EventArgs) Handles
itemValueAddBtn.Click
        'User error check'
        If manageItemList.SelectedIndex = -1 Then
            inventoryMgmtTxt.Text = "Please select an item to update the price of"
        ElseIf itemPriceTxt.Text = Nothing Then
            inventoryMgmtTxt.Text = "Please enter a price for the item"
        ElseIf itemPriceTxt.Text < 0 Or IsNumeric(itemPriceTxt.Text) = False Then
            inventoryMgmtTxt.Text = "Please enter a positive numeric value for the item
price"
        Else
            'Updates/adds price of an item'
            manageItemList.SelectedItem.Value = Convert.ToDecimal(itemPriceTxt.Text)
            itemList.Items.FindByText(manageItemList.SelectedItem.Text).Value =
Convert.ToDecimal(itemPriceTxt.Text)

            inventoryMgmtTxt.Text = FormatCurrency(itemPriceTxt.Text, 2) & " assigned as
price for " & manageItemList.SelectedItem.Text
            itemAddTxt.Text = Nothing
            itemPriceTxt.Text = Nothing
            manageItemList.SelectedIndex = -1
        End If
    End Sub

    Protected Sub itemRemoveBtn_Click(sender As Object, e As EventArgs) Handles
itemRemoveBtn.Click
        If manageItemList.SelectedIndex = -1 Then
            inventoryMgmtTxt.Text = "Please select an item from list to remove from
inventory"
        Else
            'Removes the selected item from inventory'
            Dim removedItem As ListItem
            removedItem = manageItemList.SelectedItem
            manageItemList.Items.Remove(manageItemList.SelectedItem)
            restockItemList.Items.Remove(removedItem)
            itemList.Items.Remove(removedItem)

            manageItemList.SelectedIndex = -1
            inventoryMgmtTxt.Text = manageItemList.SelectedItem.Text & " removed from
inventory"
        End If
    End Sub
End Class
```