Text

Description automatically generated

**MS806 – Business Application Programming**

**Student Name: Dilip Venkatesan Sankar**

**NUI Galway ID Number: 22225743**

**Assignment:** **4**

**Date of Submission:** **11/27/2022**

“In submitting this work, I confirm that it is entirely my own. I acknowledge that I may be invited to undertake an online interview if there is any concern in relation to the integrity of my submission.”

**/\* Student Name: Dilip Venkatesan Sankar**

**\* Student ID: 22225743**

**\* Date: 27/11/2022**

**\* Assignment: 4**

**\* Assignment: Create an EPOS app for the MBSI staff**

**\* To process customer orders and store transaction details,**

**\* Stock details and stock price**

**\*/**

**using** System**;**

**using** System**.**Data**;**

**namespace** MyBagelShopAssignment

**{**

//Initializing global variables

**public** **partial** class MBSIMainForm **:** Form

**{**

string**[]** BagelTypes **=** **new** string**[]** **{** "Plain"**,** "Cinnamon"**,** "Blueberry"**,** "Chocolate"**,** "Onion"**,** "Asiago"**,** "French"**,**

"Egg"**,** "Pumpernickel"**,** "Wheat"**,** "Garlic"**,** "Sesame"**,**"Cheddar"**};**

string**[]** BagelSize **=** **new** string**[]** **{** "Small"**,** "Medium"**,** "Regular"**,** "Large"**,** "Extra-Large" **};**

const int ROWS\_IN\_FILE **=** 13**,** COLS\_IN\_FILE **=** 5**;**

const string STOCKININVENTORYFILE **=** "Stock\_In\_Inventory\_Details.txt"**;**

**public** static int**[,]** StockInventoryArray **=** **new** int**[**ROWS\_IN\_FILE**,** COLS\_IN\_FILE**];** //Load stock details to original array

**public** static int**[,]** TempStockInventoryArray **=** **new** int**[**ROWS\_IN\_FILE**,** COLS\_IN\_FILE**];** //Load stock details to temporary array

**public** static int**[,]** DupicateStockInventoryArray **=** **new** int**[**ROWS\_IN\_FILE**,** COLS\_IN\_FILE**];** //Make a copy of orginal array

**public** static int**[,]** NewSalesArray **=** **new** int**[**ROWS\_IN\_FILE**,** COLS\_IN\_FILE**];** //Empty array to show sale values

const string BAGELPRICELISTFILE **=** "Bagel\_Price\_List.txt"**;** //Bagel price list file

**public** static int**[,]** BagelPriceListArray **=** **new** int**[**ROWS\_IN\_FILE**,** COLS\_IN\_FILE**];**

decimal BagelPrice **=** 0m**,** ItemTotal **=** 0m**,** FinalPrice **=** 0m**;**

int QuantityStock**=**0**,** TransactionNumberOutput**;**

List**<**string**>** TotalTransactionCreated **=** **new** List**<**string**>();**

string TransactionSummary **=** "Bagel\_Shop\_Transaction\_Details.txt"**;** //Bagel Transaction file

DateTime Currentdate **=** DateTime**.**Now**;**

string MessageDisplay**;** string comma **=** ""**;**

bool CheckNum**;**

int SummaryTransactions**,** SummaryQuantity**;**

decimal SummaryTotal**,** SummaryAvg**;**

int SummaryQuantity1**;**

//Ops to perform when Form loads

**public** MBSIMainForm**()**

**{**

InitializeComponent**();**

BagelTypePopulate**(**BagelTypes**);**

BagelSizePopulate**(**BagelSize**);**

//Loading stock values to array

StockInventoryArray**=** StockInventoryArrayValues**(**STOCKININVENTORYFILE**);**

DupicateStockInventoryArray **=** StockInventoryArrayValues**(**STOCKININVENTORYFILE**);**

Array**.**Copy**(**StockInventoryArray**,** 0**,** TempStockInventoryArray**,** 0**,** StockInventoryArray**.**Length**);**

BagelPriceListArray **=** BagelPriceListArrayValues**(**BAGELPRICELISTFILE**);**

**}**

//Event handler to populate bagel type list box

**private** void BagelTypePopulate**(**String**[]** BagelTypes**)**

**{**

**for** **(**int BagelTypeCount **=** 0**;** BagelTypeCount **<** BagelTypes**.**Length**;** BagelTypeCount**++)**

**{**

BagelTypeListBox**.**Items**.**Add**(**BagelTypes**[**BagelTypeCount**]);**

**}**

**}**

//Event handler to populate bagel size list box

**private** void BagelSizePopulate**(**String**[]** BagelSize**)**

**{**

**for** **(**int BagelSizeCount **=** 0**;** BagelSizeCount **<** BagelSize**.**Length**;** BagelSizeCount**++)**

**{**

BagelSizeListBox**.**Items**.**Add**(**BagelSize**[**BagelSizeCount**]);**

**}**

**}**

//Assigning array values from the sales inventory file

**public** int**[,]** StockInventoryArrayValues**(**string StockFileName**)**

**{**

int**[,]** StockInInventoryArray **=** **new** int**[**ROWS\_IN\_FILE**,** COLS\_IN\_FILE**];** //empty array

StreamReader FileRead**;**

int RowCount **=** 0**;**

**try**

**{**

FileRead **=** File**.**OpenText**(**STOCKININVENTORYFILE**);** //Open Stock file to read data

**while** **(!**FileRead**.**EndOfStream**)**

**{**

string**[]** DataRead **=** FileRead**.**ReadLine**().**Split**(**','**);** //read line and store in array as common seperated

**for** **(**int ColCount **=** 0**;** ColCount **<** DataRead**.**Length**;** ColCount**++)**

**{**

StockInInventoryArray**[**RowCount**,** ColCount**]** **=** int**.**Parse**(**DataRead**[**ColCount**]);** //loop and read each column and store in array

**}**

RowCount**++;**

**}**

FileRead**.**Close**();**

**}**

**catch** **(**Exception Excep**)**

**{**

MessageBox**.**Show**(**Excep**.**Message**);**

**}**

**return** StockInInventoryArray**;**

**}**

//Assigning array values from price list file

**public** int **[,]** BagelPriceListArrayValues**(**string PriceListFileName**)**

**{**

int**[,]** BagelPriceListArray **=** **new** int**[**ROWS\_IN\_FILE**,** COLS\_IN\_FILE**];** //empty array

StreamReader FileRead**;**

int RowCount **=** 0**;**

**try**

**{**

FileRead **=** File**.**OpenText**(**BAGELPRICELISTFILE**);** //Open Price file to read price list

**while** **(!**FileRead**.**EndOfStream**)**

**{**

string**[]** DataRead **=** FileRead**.**ReadLine**().**Split**(**','**);** //read line and store in array as common seperated

**for** **(**int ColCount **=** 0**;** ColCount **<** DataRead**.**Length**;** ColCount**++)**

**{**

BagelPriceListArray**[**RowCount**,** ColCount**]** **=** int**.**Parse**(**DataRead**[**ColCount**]);** //loop and read each column and store in array

**}**

RowCount**++;**

**}**

FileRead**.**Close**();**

**}**

**catch** **(**Exception Excep**)**

**{**

MessageBox**.**Show**(**Excep**.**Message**);**

**}**

**return** BagelPriceListArray**;**

**}**

//Event Handler to check bagel type and related size if avaiable

**public** void BagelSizeListBox\_SelectedIndexChanged**(object** sender**,** EventArgs e**)**

**{**

**if** **(**BagelSizeListBox**.**SelectedIndex **!=** **-**1**)**

**{**

**if(**StockInventoryArray**[**BagelTypeListBox**.**SelectedIndex**,** BagelSizeListBox**.**SelectedIndex**]** **==** 0**)** //condition to check if stock is available

**{**

MessageBox**.**Show**(**"OOPS! Product Out of Stock" **+** "\n" **+** "Please select another Item"**,** "Stock Alert"**,**

MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Exclamation**);**

BagelSizeListBox**.**ClearSelected**();**

PriceLabel2**.**Text **=** ""**;**

**}**

**else** // popluate price box if stock is avaliables

**{**

BagelPrice **=** BagelPriceListArray**[**BagelTypeListBox**.**SelectedIndex**,** BagelSizeListBox**.**SelectedIndex**];**

PriceLabel2**.**Text **=** BagelPrice**.**ToString**(**"C2"**);**

**}**

**}**

**}**

//Event handler to add items to cart based on available stock

**public** void AddToCartButton\_Click**(object** sender**,** EventArgs e**)**

**{**

QuantityStock **=** Convert**.**ToInt32**(**UpDownOperator**.**Value**);** //store quantity in variable

SummaryQuantity1 **+=** QuantityStock**;** //increment quantity on each order

**if** **(**BagelTypeListBox**.**SelectedIndex **!=** **-**1**)**

**{**

**if(**BagelSizeListBox**.**SelectedIndex **!=** **-**1 **)**

**{**

**if(**QuantityStock **>** 0**)** //condition to check for available stocks

**{**

**if** **(**QuantityStock **>** StockInventoryArray**[**BagelTypeListBox**.**SelectedIndex**,** BagelSizeListBox**.**SelectedIndex**])**

**{**

MessageBox**.**Show**(**"OOPS! Only "**+**

StockInventoryArray**[**BagelTypeListBox**.**SelectedIndex**,** BagelSizeListBox**.**SelectedIndex**]+**

" Avaiable in Stock Currently"**+**"\n"**+**"SORRY :("**,** "Stock Update"**,**

MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Warning**);**

UpDownOperator**.**Value **=** StockInventoryArray**[**BagelTypeListBox**.**SelectedIndex**,** BagelSizeListBox**.**SelectedIndex**];**

**}**

**else** //add items to cart list view and receipt

**{**

ItemTotal **=** BagelPrice **\*** QuantityStock**;**

ListViewItem cart **=** **new** ListViewItem**(**BagelTypes**[**BagelTypeListBox**.**SelectedIndex**]);**

cart**.**SubItems**.**Add**(**BagelSize**[**BagelSizeListBox**.**SelectedIndex**]);**

cart**.**SubItems**.**Add**(**QuantityStock**.**ToString**());**

cart**.**SubItems**.**Add**(**ItemTotal**.**ToString**(**"C2"**));**

BagelShowCartView**.**Items**.**Add**(**cart**);**

FinalPrice **=** ItemTotal **+** FinalPrice**;**

TotalPriceLabel2**.**Text **=** FinalPrice**.**ToString**(**"C2"**);**

TotalTransactionCreated**.**Add**(**BagelTypeListBox**.**SelectedIndex**.**ToString**()** **+** "|" **+**

BagelSizeListBox**.**SelectedIndex**.**ToString**()** **+** "|" **+** QuantityStock **+** "|" **+** ItemTotal**);**

MessageDisplay **+=** QuantityStock **+** " " **+** BagelTypes**[**BagelTypeListBox**.**SelectedIndex**]** **+**

" " **+** BagelSize**[**BagelSizeListBox**.**SelectedIndex**]** **+** " = $" **+** ItemTotal**+**"\n"**;**

//updating stock details

StockInventoryArray**[**BagelTypeListBox**.**SelectedIndex**,** BagelSizeListBox**.**SelectedIndex**]** **=**

StockInventoryArray**[**BagelTypeListBox**.**SelectedIndex**,** BagelSizeListBox**.**SelectedIndex**]** **-** QuantityStock**;**

BagelTypeListBox**.**SelectedItems**.**Clear**();**

BagelSizeListBox**.**SelectedItems**.**Clear**();**

BagelSizeListBox**.**Enabled **=** **false;**

PriceLabel2**.**Text **=** ""**;**

UpDownOperator**.**Value **=** 0**;**

QuantityStock **=** 0**;**

**}**

**}**

**else**

**{**

MessageBox**.**Show**(**"Add Atleast one doughnut to Proceed"**,** "Proceed Error"**,**

MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Error**);**

**}**

**}**

**else**

**{**

MessageBox**.**Show**(**"Select any Doughnut Size to Proceed"**,** "Selection Error"**,**

MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Error**);**

**}**

**}**

**else**

**{**

MessageBox**.**Show**(**"Select any Doughnut Type to Proceed"**,** "Selection Error"**,**

MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Error**);**

**}**

**}**

//Event habdler to exit the application

**private** void ExitButton\_Click**(object** sender**,** EventArgs e**)**

**{**

DialogResult iExit**;**

iExit **=** MessageBox**.**Show**(**"Confirm if you want to exit the system"**,**

"BagelShop system"**,** MessageBoxButtons**.**YesNo**,** MessageBoxIcon**.**Question**);**

**if** **(**iExit **==** DialogResult**.**Yes**)**

**{**

Application**.**Exit**();**

**}**

**}**

//Event handler to generate sales report

**private** void SalesReportButton\_Click**(object** sender**,** EventArgs e**)**

**{**

SalesForm NewForm **=** **new** SalesForm**();** //call sales form3

NewForm**.**ShowDialog**();**

**}**

//Event handler to generate stock report file

**private** void StockReportButton\_Click**(object** sender**,** EventArgs e**)**

**{**

StreamWriter StockReportGenerator **=** File**.**CreateText**(**"Bagel\_Shop\_Stock\_Report.txt"**);** //Create a file in debug folder

//add below lines in the file

StockReportGenerator**.**WriteLine**(**" Stock report generated date: " **+** Currentdate**.**ToShortDateString**()**

**+**"Time: "**+** Currentdate**.**ToShortTimeString**());**

StockReportGenerator**.**WriteLine**(**"\n"**);**

StockReportGenerator**.**WriteLine**(**"UPDATED STOCKS ARE SHOWN BELOW:\n"**);**

string FileInfo **=** string**.**Format**(**"{0,-20}{1,-15}{2,-20}{3,-15}{4,-15}{5,-15}\n"**,**

"Food Items"**,** "Small"**,** "Medium"**,** "Regular"**,** "Large"**,** "Extra-large"**);** //formatting of the file with spacing

**for(**int count**=**0**;**count**<**BagelTypes**.**Length**;**count**++)** //store the remaning stock with names in the varibles

**{**

FileInfo **+=** String**.**Format**(**"{0,-20}{1,-15}{2,-20}{3,-15}{4,-15}{5,-15}\n"**,** BagelTypes**[**count**],**

StockInventoryArray**[**count**,** 0**],** StockInventoryArray**[**count**,** 1**],** StockInventoryArray**[**count**,** 2**]**

**,** StockInventoryArray**[**count**,** 3**],** StockInventoryArray**[**count**,** 4**]);**

**}**

StockReportGenerator**.**WriteLine**(**$"{FileInfo}"**);** //Write to file

StockReportGenerator**.**Close**();**

MessageBox**.**Show**(**"Stock Report Generated Successsfully"**,** "Stock Report Generation"**,**

MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Information**);**

**}**

//Main Form Load

**private** void MBSIMainForm\_Load**(object** sender**,** EventArgs e**)**

**{**

TrxnLabel**.**Visible **=** **false;**

BagelSizeListBox**.**Enabled **=** **false;**

**}**

//Method to generate random ID and return value

**private** int RandomNumberGeneration**()**

**{**

Random RandomTN **=** **new** Random**();**

int NewTransactionNumber **=** RandomTN**.**Next**(**10000**,** 99999**);**

**return** NewTransactionNumber**;** //storing in varible to diplay in message box

**}**

//Event handler to display summary form and details

**private** void SummaryButton\_Click**(object** sender**,** EventArgs e**)**

**{**

SummaryMainForm NewForm **=** **new** SummaryMainForm**();**

NewForm**.**SummaryValues**(**SummaryQuantity**,** SummaryTotal**,** SummaryTransactions**,** SummaryAvg**);**

NewForm**.**Show**();**

**}**

//Event handler to check transaction number is unique

**private** void CheckRandomNumberIsUnique**(**int NewTrxnNumber**,** string TransactionSummary**)**

**{**

string ValCheck **=** NewTrxnNumber**.**ToString**();**

string**[]** CheckFile **=** File**.**ReadAllLines**(**TransactionSummary**);** //Read lines from the transaction file

**for(**int each**=**0**;**each**<**CheckFile**.**Length**;**each**++)**

**{**

**if** **(**CheckFile**[**each**].**Contains**(**ValCheck**))** //condition to check if same number exists

**{**

CheckNum **=** **true;**

TrxnLabel**.**Visible**=true;**

TransactionNumberOutput **=** RandomNumberGeneration**();** //regenerate number

**}**

**else**

**{**

CheckNum **=** **false;**

TrxnLabel**.**Visible **=** **false;**

**}**

**}**

**}**

//Event handler to clear the order and cart

**private** void ClearButton\_Click**(object** sender**,** EventArgs e**)**

**{**

FinalPrice **=** 0**;**

TotalTransactionCreated**.**Clear**();**

BagelTypeListBox**.**ClearSelected**();**

BagelSizeListBox**.**ClearSelected**();**

UpDownOperator**.**Value **=** 0**;**

PriceLabel2**.**Text **=** ""**;**

Array**.**Copy**(**TempStockInventoryArray**,** 0**,** StockInventoryArray**,** 0**,** StockInventoryArray**.**Length**);**

BagelShowCartView**.**Items**.**Clear**();**

TotalPriceLabel2**.**Text **=** ""**;**

MessageDisplay **=** ""**;**

SummaryQuantity1 **=** SummaryQuantity**;** **;**

**}**

//Event handler to confirm the order

**private** void ConfirmButton\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**File**.**Exists**(**TransactionSummary**))**

**{**

**if(**BagelShowCartView**.**Items**.**Count **!=** 0**)**

**{**

TransactionNumberOutput **=** RandomNumberGeneration**();** //random numner generation

CheckRandomNumberIsUnique**(**TransactionNumberOutput**,** TransactionSummary**);** //check random number is unique

DialogResult ProceedResult **=** MessageBox**.**Show**(**"\n" **+** "Do you wish to confirm your details? " **+**

"\n" **+** "Transaction Number: " **+** TransactionNumberOutput **+**

"\n" **+** "Transaction Timestamp: " **+** Currentdate**.**ToString**()** **+**

"\n" **+** MessageDisplay **+**

"\n" **+** "Total Price: $" **+** FinalPrice**,** "Confirmation"**,** MessageBoxButtons**.**OKCancel**,**

MessageBoxIcon**.**Information**);**

**if** **(**ProceedResult **==** DialogResult**.**OK**)**

**{**

**try**

**{**

//Storing and incr values for summary details

SummaryTransactions**++;**

SummaryTotal **+=** FinalPrice**;**

SummaryAvg **=** SummaryTotal **/** SummaryTransactions**;**

SummaryQuantity **=** SummaryQuantity1**;**

Array**.**Copy**(**StockInventoryArray**,** 0**,** TempStockInventoryArray**,** 0**,** StockInventoryArray**.**Length**);** //copy array elements if OK

StreamWriter TransactionSummaryDetails**;** //store transaction in file

TransactionSummaryDetails **=** File**.**AppendText**(**TransactionSummary**);**

**for** **(**int count **=** 0**;** count **<** TotalTransactionCreated**.**Count**;** count**++)**

**{**

int BagelTypeIndex **=** int**.**Parse**((**TotalTransactionCreated**[**count**].**Split**(**'|'**))[**0**]);**

int BagelSizeIndex **=** int**.**Parse**((**TotalTransactionCreated**[**count**].**Split**(**'|'**))[**1**]);**

int BagelQuantityIndex **=** int**.**Parse**((**TotalTransactionCreated**[**count**].**Split**(**'|'**))[**2**]);**

int TotalPriceIndex **=** int**.**Parse**((**TotalTransactionCreated**[**count**].**Split**(**'|'**))[**3**]);**

TransactionSummaryDetails**.**WriteLine**(**TransactionNumberOutput **+** " | " **+** Currentdate**.**ToString**()** **+** " | " **+**

BagelTypes**[**BagelTypeIndex**]** **+** " | " **+** BagelSize**[**BagelSizeIndex**]** **+** " | "

**+** BagelQuantityIndex **+** " | " **+** TotalPriceIndex**);**

**}**

TransactionSummaryDetails**.**Close**();**

//update stock array after confirm order

File**.**Delete**(**STOCKININVENTORYFILE**);**

**if** **(!**File**.**Exists**(**STOCKININVENTORYFILE**))**

**{**

var \_File **=** File**.**CreateText**(**STOCKININVENTORYFILE**);**

\_File**.**Dispose**();**

**}**

StreamWriter LatestSalesReport **=** File**.**AppendText**(**STOCKININVENTORYFILE**);**

**for** **(**int i **=** 0**;** i **<** ROWS\_IN\_FILE**;** i**++)**

**{**

comma **=** ""**;**

**for** **(**int j **=** 0**;** j **<** COLS\_IN\_FILE**;** j**++)**

**{**

LatestSalesReport**.**Write**(**comma**);**

LatestSalesReport**.**Write**(**StockInventoryArray**[**i**,** j**]);**

comma **=** ","**;**

**}**

LatestSalesReport**.**Write**(**Environment**.**NewLine**);**

**}**

LatestSalesReport**.**Close**();**

BagelTypeListBox**.**ClearSelected**();**

BagelSizeListBox**.**ClearSelected**();**

UpDownOperator**.**Value **=** 0**;**

PriceLabel2**.**Text **=** ""**;**

FinalPrice **=** 0**;**

BagelShowCartView**.**Items**.**Clear**();**

TotalPriceLabel2**.**Text **=** ""**;**

MessageDisplay **=** ""**;**

TotalTransactionCreated**.**Clear**();**

//SummaryQuantity1 = 0;

**}**

**catch** **(**Exception Excep**)**

**{**

MessageBox**.**Show**(**Excep**.**Message**);**

**}**

**}**

**else** //ops to perform when cancel order

**{**

Array**.**Copy**(**TempStockInventoryArray**,** 0**,** StockInventoryArray**,** 0**,** StockInventoryArray**.**Length**);** //revert array elements

BagelShowCartView**.**Items**.**Clear**();**

TotalPriceLabel2**.**Text **=** ""**;**

FinalPrice **=** 0**;**

MessageDisplay **=** ""**;**

SummaryQuantity1 **=** SummaryQuantity**;**

**}**

**}**

**else**

**{**

DialogResult ProceedResult **=** MessageBox**.**Show**(**"\n" **+** "Please add Items to Cart to Proceed "**,**

"Add Items to Cart"**,** MessageBoxButtons**.**OK**,**

MessageBoxIcon**.**Information**);**

**}**

**}**

**else**

**{**

StreamWriter NewFile **=** File**.**CreateText**(**TransactionSummary**);**

NewFile**.**Close**();**

**}**

**}**

//Event handler to search transaction details

**private** void SearchButton\_Click**(object** sender**,** EventArgs e**)**

**{**

SearchForm NewForm **=** **new** SearchForm**();**

NewForm**.**Show**();**

**}**

**private** void BagelTypeListBox\_SelectedIndexChanged**(object** sender**,** EventArgs e**)**

**{**

BagelSizeListBox**.**Enabled **=** **true;**

**}**

**private** void QuantityLabel1\_Click**(object** sender**,** EventArgs e**)**

**{**

**}**

**}**

**}**

**===================================================================================================**

//SALES FORM OPS

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**;**

**using** System**.**Windows**.**Forms**;**

//SALES FORM OPS

**namespace** MyBagelShopAssignment

**{**

**public** **partial** class SalesForm **:** Form

**{**

**public** SalesForm**()**

**{**

InitializeComponent**();**

**}**

//initialize sales form global variables

const int ROWS\_IN\_FILE **=** 13**,** COLS\_IN\_FILE **=** 5**;**

**public** static int**[,]** NewSalesArray **=** **new** int**[**ROWS\_IN\_FILE**,** COLS\_IN\_FILE**];**

DateTime CurrentTime **=** DateTime**.**Now**;**

string**[]** BagelTypes **=** **new** string**[]** **{** "Plain"**,** "Cinnamon"**,** "Blueberry"**,** "Chocolate"**,** "Onion"**,** "Asiago"**,** "French"**,**

"Egg"**,** "Pumpernickel"**,** "Wheat"**,** "Garlic"**,** "Sesame"**,**"Cheddar"**};**

//Event handler to exit the sales form and return to MBSI form1

**private** void ExitButton\_Click**(object** sender**,** EventArgs e**)**

**{**

**this.**Close**();**

**}**

**private** void listView2\_SelectedIndexChanged**(object** sender**,** EventArgs e**)**

**{**

**}**

//Event handler to display sales form

**private** void SalesForm\_Load**(object** sender**,** EventArgs e**)**

**{**

DateLabel**.**Text **=** CurrentTime**.**ToShortDateString**();**

TimeLabel**.**Text **=** CurrentTime**.**ToShortTimeString**();**

**for** **(**int j **=** 0**;** j **<** 13**;** j**++)** // using listview to display the stock names

**{**

ListViewItem item2 **=** **new** ListViewItem**(**BagelTypes**[**j**].**ToString**());**

listView2**.**Items**.**Add**(**item2**);**

**}**

**for** **(**int a **=** 0**;** a **<** ROWS\_IN\_FILE**;** a**++)** //calculate remaining stock

**{**

**for** **(**int j **=** 0**;** j **<** COLS\_IN\_FILE**;** j**++)**

**{**

NewSalesArray**[**a**,** j**]** **=** MBSIMainForm**.**DupicateStockInventoryArray**[**a**,** j**]** **-** MBSIMainForm**.**StockInventoryArray**[**a**,** j**];**

**}**

**}**

**for** **(**int i **=** 0**;** i **<** 13**;** i**++)** //Dispaly sales values in the list view

**{**

ListViewItem item **=** **new** ListViewItem**(**NewSalesArray**[**i**,** 0**].**ToString**());**

item**.**SubItems**.**Add**(**NewSalesArray**[**i**,** 1**].**ToString**());**

item**.**SubItems**.**Add**(**NewSalesArray**[**i**,** 2**].**ToString**());**

item**.**SubItems**.**Add**(**NewSalesArray**[**i**,** 3**].**ToString**());**

item**.**SubItems**.**Add**(**NewSalesArray**[**i**,** 4**].**ToString**());**

listView1**.**Items**.**Add**(**item**);**

**}**

**}**

**}**

**}**

**===================================================================================================**

//SEARCH FORM OPS

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**;**

**using** System**.**Windows**.**Forms**;**

**namespace** MyBagelShopAssignment

**{**

**public** **partial** class SearchForm **:** Form

**{**

**public** SearchForm**()**

**{**

InitializeComponent**();**

**}**

bool CheckStatus**;** int IncrNum**;**

string TransactionSummary **=** "Bagel\_Shop\_Transaction\_Details.txt"**;**

**private** void SearchForm\_Load**(object** sender**,** EventArgs e**)**

**{**

**}**

//Event handler to perform search ops

**private** void SearchButton\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**SearchTextBox**.**Text **==** ""**)** //Execute if condition when textbox is empty

**{**

MessageBox**.**Show**(**"Please Enter Transaction date or Transaction number"**,** "Details"**,** MessageBoxButtons**.**OK**,**

MessageBoxIcon**.**Error**);**

**}**

**else**

**{**

**try**

**{**

**if** **((**TransactionNumberRadioButton**.**Checked**)** **&&** **!(**SearchTextBox**.**Text**.**Contains**(**"/"**))** **&&** **(**SearchTextBox**.**Text**.**Length**==**5**))**

**{**

CheckStatus **=** **true;** // return true when entered and file email address matchs for search condition

**}**

**else** **if** **((**TransactionDateRadioButton**.**Checked**)** **&&** **(**SearchTextBox**.**Text**.**Contains**(**"/"**)))**

**{**

CheckStatus **=** **true;** //return true when entered and file transaction number match for search condition

**}**

**if(**CheckStatus**==true)**

**{**

string**[]** TransactionFile **=** File**.**ReadAllLines**(**TransactionSummary**);**

**foreach** **(**string each **in** TransactionFile**)**

**{**

**if** **(**each**.**Contains**(**SearchTextBox**.**Text**))**

**{**

SearchListBox**.**Items**.**Add**(**each**);** //iterate and check for matching records

IncrNum **=** 1**;**

**}**

**else**

**{**

IncrNum **=** 0**;**

**}**

**}**

CheckStatus **=** **false;**

**}**

**else**

**{**

MessageBox**.**Show**(**"No records found"**,** "No records"**,** MessageBoxButtons**.**OK**,**

MessageBoxIcon**.**Information**);**

**}**

**}**

**catch**

**{**

**}**

**}**

**}**

////Event handler for clear ops

**private** void ClearButton\_Click**(object** sender**,** EventArgs e**)**

**{**

SearchTextBox**.**Clear**();**

SearchListBox**.**Items**.**Clear**();**

SearchButton**.**Enabled **=** **true;**

TransactionNumberRadioButton**.**Checked **=** **false;**

TransactionDateRadioButton**.**Checked**=** **false;**

**}**

//Event handler to exit search form

**private** void ExitButton\_Click**(object** sender**,** EventArgs e**)**

**{**

**this.**Close**();**

**}**

**}**

**}**

**===================================================================================================**

//SUMMARY FORM OPS

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**;**

**using** System**.**Windows**.**Forms**;**

**namespace** MyBagelShopAssignment

**{**

**public** **partial** class SummaryMainForm **:** Form

**{**

**public** SummaryMainForm**()**

**{**

InitializeComponent**();**

**}**

//Event handler to exit the summary form

**private** void ExitButton\_Click**(object** sender**,** EventArgs e**)**

**{**

**this.**Close**();**

**}**

**private** void SummaryMainForm\_Load**(object** sender**,** EventArgs e**)**

**{**

**}**

//Event handler to display summary values

**public** void SummaryValues**(**int val1**,** decimal val2**,** int val3**,** decimal val4**)**

**{**

TotalBagelSoldLabel1**.**Text **=** val1**.**ToString**();**

TotalSaleValueLabel1**.**Text **=** val2**.**ToString**(**"C2"**);**

TotalTnxlabel1**.**Text**=**val3**.**ToString**();**

AvgTransactiosnLabel1**.**Text **=** val4**.**ToString**(**"C2"**);**

**}**

**}**

**}**

**===================================================================================================**