

**MS806 – Business Application Programming**

**Student Name: Rajit Patel**

**NUI Galway ID Number: 17233634**

**Assignment:** 5

**Date of Submission:** 22/11/2017

CODE:

/\* Student Name: Rajit Patel

\* Student ID: 17233634

\* Date: 22/11/2017

\* Assignment: 3

\* Task: Create a well-designed application for Mr. BagelInc regional chain of ten gourmet bagel bars, whose rapid rate of expansion has meant that the head office management team wishes to become more systematic and systems orientated in their business approach. Designing an application for EPOS system which will be handled by the counter staff.

using System;

using System.Collections;/\*used for genearting array list\*/

using System.IO;/\*used for input output file operations\*/

using System.Net;

using System.Net.Mail;

using System.Windows.Forms;

namespace Assignment\_5

{

public partial class Form1 : Form

{

int BagelTypeIndex, BagelSizeIndex;

/\*Declaring global variable as string for calculation purpose\*/

string pricing, TransactionNumber;

decimal priceLine;

/\*Declared Constant value which wont change in entire programme\*/

const int BagelRow = 5, BagelColumn = 16;

decimal TotalPrice = 0m;

double AvgSalesValue;

double TotalSalesvalue = 0;

decimal BagelCost, stock = 0m;

/\*Declaring global variable as string to store the value in date format \*/

string uniquetextfile = string.Format("{0:yyyy-MM-dd}\_{1}", DateTime.Now, "Mr.BagelStockDetails.txt");

string uniquefilepath = @"Mr.BagelSummaryDetails.txt" ;

/\*Declaring global variable as string to for calculation purpose \*/

string transactionfilepath = string.Format("{0:yyyy-MM-dd}\_{1}", DateTime.Now, "EntireTransactionsDetails.txt");

/\*Initializing the int variable \*/

int TotalsummaryTransaction = 0, NumberofBagelSold = 0, Quantity;

string ReportStock=@"stockReport.txt" ;

String ReportStockfilename = string.Format("{0:yyyy-MM-dd}\_{1}", DateTime.Now, "stockReport.txt");

/\*Declaring 2 dimensional array and initializing it \*/

decimal[,] BagelType = new decimal[BagelRow, BagelColumn]{ {2.45m,2.50m,2.55m, 2.60m ,4.55m,2.40m,3.49m,2.19m,3.99m,7.89m,3.45m,3.67m,4.55m,5.00m,1.23m,3.45m },

{2.95m,3.00m,3.05m,2.99m,5.69m,2.90m,3.99m,3.12m,4.49m,8.89m,3.75m,3.97m,5.00m,5.50m,2.23m,3.95m },

{3.45m,3.50m,3.55m, 3.38m ,6.83m,3.40m,4.49m,4.05m,5.50m,8.50m,4.25m,4.50m,5.36m,6.00m,3.23m,4.45m },

{3.99m,3.99m,4.00m ,3.99m,6.69m,3.59m,4.99m,4.49m,6.51m,8.11m,4.75m,5.03m,5.72m,6.50m, 4.23m,4.49m},

{4.53m,4.48m,4.45m, 4.60m ,9.99m,3.78m,5.49m,4.93m,7.52m,7.72m,5.25m,5.56m,6.08m,7.00m,5.23m,5.45m } };

decimal[,] BagelStock = new decimal[BagelRow, BagelColumn] { { 2,4,6,8,12,12,8,4,0,3,6,9,12,10,55,6},

{ 3,4,5,6,45,8,5,2,7,8,9,10,11,7,3,5},

{ 4,7,10,13,34,19,12,5,4,6,8,10,12,8,4,0},

{5,13,11,12,23,12,4,7,10,8,6,4,21,4,6,8 },

{6,2,0,2,4,6,3,4,5,6,7,8,9,6,23,8 } };

/\*Creating Arraylist to hold the data\*/

ArrayList BagelDetails = new ArrayList();

ArrayList Alltransactions = new ArrayList();

/\*Event handler for clear button\*/

private void buttonClear\_Click(object sender, EventArgs e)

{

try

{

/\*foreach loop to execute the below code in array list to update the stock array if order is cancelled\*/

foreach (Bagel infoDetails in BagelDetails)

{

BagelStock[infoDetails.BagelTypeIndex, infoDetails.BagelSizeIndex] = BagelStock[infoDetails.BagelTypeIndex, infoDetails.BagelSizeIndex] + infoDetails.QuantityofBagel;

}

TotalPrice = 0;

dataGridViewBagel.Rows.Clear();/\*Clear the rows in datagridView\*/

textBoxQuantity.Clear();

labelTotalPrice.Text = "";/\*Clear the label\*/

labelBagelPrice.Text = "";

listBoxBagelType.ClearSelected();

listBoxSize.ClearSelected();/\*Clear the index of listbox\*/

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/\*Event handler for selecting Bagel type\*/

private void listBoxSize\_SelectedIndexChanged(object sender, EventArgs e)

{

/\*Statement to handle exception and prevent the programme from crashing\*/

try

{

/\*User should select atleast one bagel type\*/

if (listBoxBagelType.SelectedIndex != -1)

{

/\*Storing the value of index in variable of datatype int\*/

BagelSizeIndex = listBoxSize.SelectedIndex;

pricing = BagelType[BagelSizeIndex, BagelTypeIndex].ToString();

labelBagelPrice.Text = "€"+pricing;

}

/\*If listbox not selected the below message box will be displayed\*/

else

{

MessageBox.Show("Please select the type of Bagel");

}

}

/\*Statement will catch any exception thrown by try statement\*/

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

/\*Event handler for complete order button\*/

private void buttonCompleteOrder\_Click(object sender, EventArgs e)

{

try

{

/\*Clear the pre values of datagrid initially\*/

dataGridViewBagel.Rows.Clear();

dataGridViewBagel.Visible = true;

/\*Generate a Random Number for transaction Id\*/

Random rnd = new Random();

/\*Storing the random number geberated to variable \*/

TransactionNumber = rnd.Next(100000, 1000000000).ToString();

/\*To add the elements as row in datagrid everytime the order is completed\*/

foreach (Bagel info in BagelDetails)

{

dataGridViewBagel.Rows.Add(info.BagelName, "€"+ info.priceofBagel, "€" + info.TotalPriceofBagel, info.SizeofBagel, info.QuantityofBagel);

/\*Calculating and storing the values of numbers of bagel sold \*/

NumberofBagelSold = info.QuantityofBagel;

Alltransactions.Add(TransactionNumber);

Alltransactions.Add(Quantity);

Alltransactions.Add(listBoxBagelType.SelectedItem.ToString());

Alltransactions.Add(listBoxSize.SelectedItem.ToString());

Alltransactions.Add(priceLine);

Alltransactions.Add(TotalPrice);

}

dataGridViewBagel.Rows.Add("", "", "", "", "");

/\*Display the total price and transactionId in datagrid\*/

dataGridViewBagel.Rows.Add("", "", "Total", "€" + TotalPrice, "");

dataGridViewBagel.Rows.Add("", "", "Transaction Id", TransactionNumber, "");

/\*Initializing the global variable to one for calculation purpose \*/

TotalsummaryTransaction = 1;

/\*Casting the values and then storing \*/

TotalSalesvalue = (double)TotalPrice;

/\*Clearing the value of textbox\*/

textBoxQuantity.Text = " ";

listBoxBagelType.ClearSelected();

listBoxSize.ClearSelected();/\*Clear the index of listbox\*/

/\*Do not perform any action if file is not present \*/

if (!File.Exists(uniquefilepath))

{

}

else

{

/\*\*/

StreamReader inputfile = new StreamReader(uniquefilepath);

/\*Reading the input file and parsing the value ,storing it in a variable \*/

TotalsummaryTransaction += int.Parse(inputfile.ReadLine());

NumberofBagelSold += int.Parse(inputfile.ReadLine());

TotalSalesvalue += double.Parse(inputfile.ReadLine());

/\*Closing the file after accessing it \*/

inputfile.Close();

}

/\*Calculating the average sales value \*/

AvgSalesValue = TotalSalesvalue / TotalsummaryTransaction;

/\*Declaring a variable outputfile to reference a Streamwriter object\*/

StreamWriter outputfile;

/\*Open the files in the path and get a StreamWriter object\*/

outputfile = File.CreateText(uniquefilepath);

/\*Write the first line with the value passed of variables\*/

outputfile.WriteLine(TotalsummaryTransaction);

outputfile.WriteLine(NumberofBagelSold);

outputfile.WriteLine(TotalSalesvalue);

/\*Close the file \*/

outputfile.Close();

/\*Initializing the value of total running price\*/

TotalPrice = 0;

labelTotalPrice.Text = TotalPrice.ToString("c2");

pricing = 0.ToString();

labelBagelPrice.Text = pricing;

/\*Clearing the array list \*/

BagelDetails.Clear();

}

catch(Exception ex)

{

/\*show the messsage of exception catched\*/

MessageBox.Show(ex.Message);

}

}

/\*Event handler for summary button \*/

private void buttonSummary\_Click(object sender, EventArgs e)

{

groupBoxSummary.Visible = true;

StreamReader inputfile = new StreamReader(uniquefilepath);

TotalsummaryTransaction = int.Parse(inputfile.ReadLine());

NumberofBagelSold = int.Parse(inputfile.ReadLine());

TotalSalesvalue = double.Parse(inputfile.ReadLine());

inputfile.Close();

/\*Displaying the value in textbpoxes\*/

AvgSalesValue = TotalSalesvalue / TotalsummaryTransaction;

labelTransaction.Text = TotalsummaryTransaction.ToString();

labelBagelSold.Text = NumberofBagelSold.ToString();

/\*using currency formatting to display the values\*/

labelTotalRevenue.Text = TotalSalesvalue.ToString("c2");

labelAverageRevenue.Text = AvgSalesValue.ToString("c2");

}

/\*Event handler for close button\*/

private void buttonClose\_Click(object sender, EventArgs e)

{

groupBoxSummary.Visible = false;

}

/\*Event handler for stock report\*/

private void buttonStock\_Click(object sender, EventArgs e)

{

/\*Initializing the value of variable for calculation purpose\*/

int k=0;

try

{

/\*Write all the text to a file \*/

File.WriteAllText(ReportStock, string.Empty);

StreamWriter outputfile;

outputfile = File.AppendText(ReportStock);

/\*write the below text in file\*/

outputfile.WriteLine("&&&&&&&&&&&&&&&&&&& Mr.Bagel &&&&&&&&&&&&&&&&&");

outputfile.WriteLine("&&&&&&&&&&&&&&&&&&& Stock Report &&&&&&&&&&&&&&&&&");

outputfile.WriteLine("&&&&&&&&&&&&&&&&&&&"+ DateTime.Now.ToString()+ "&&&&&&&&&&&&&&&&&");

outputfile.WriteLine("\t\t\t\t" + "Small" + "\t\t\t\t" + "Regular" + "\t\t\t\t" + "Medium" + "\t\t\t\t" + "Large"+ "\t\t\t" + "Extlarge");

/\*Create new line\*/

outputfile.WriteLine(Environment.NewLine);

/\*using two nested forloop to genrate formatted stock report\*/

for (int i = 0; i < 16; i++)

{

outputfile.WriteLine(Environment.NewLine);

/\*Write the element from the listbox\*/

outputfile.WriteLine(listBoxBagelType.Items[k]);

for (int j=0; j < 5; j++)

{

/\*Write the stock value from stock array in file\*/

outputfile.Write("\t\t\t\t"+BagelStock[j,i]);

}

/\*Incrementing the value of variable to write every next element from listbox to file \*/

k++;

}

/\*Closing the file \*/

outputfile.Close();

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

panelStockReport.Visible = true;

textBoxEmail.Enabled = false;

buttonEmailDone.Enabled = false;

}

/\*Event handler for selecting email as an option\*/

private void radioButtonEmail\_CheckedChanged(object sender, EventArgs e)

{

/\*Enabling the textbox and email button\*/

textBoxEmail.Enabled = true;

buttonEmailDone.Enabled = true;

}

/\*Event handler for selecting download as an option\*/

private void radioButtonDownload\_CheckedChanged(object sender, EventArgs e)

{

textBoxEmail.Enabled = false;

buttonEmailDone.Enabled = false;

try

{

Stream Reportstream = File.OpenRead(ReportStock);

SaveFileDialog DialogueBox = new SaveFileDialog();

DialogueBox.Filter ="txt files(\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

DialogueBox.FilterIndex = 1;

DialogueBox.FileName = string.Format("{0:yyyy-MM-dd}\_{1}", DateTime.Now, "stockReport.txt");

if(DialogueBox.ShowDialog()== DialogResult.OK)

{

if((Reportstream = DialogueBox.OpenFile())!=null)

{

Reportstream.Close();

}

}

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/\*Event handler for email to be sent \*/

private void buttonEmailDone\_Click(object sender, EventArgs e)

{

try

{

if (validationEmail(textBoxEmail.Text))

{

var fromAddress = new MailAddress("rajitpatel32@gmail.com");

var toAddress = new MailAddress(textBoxEmail.Text, "To");

const string fromPassword = "integration1994";

const string subject = "stock Report";

const string body = "Please find the attached stock report";

System.Net.Mail.Attachment attachmentfile;

attachmentfile = new System.Net.Mail.Attachment(String.Format("{0:yyyy-MM-dd}\_{1}", DateTime.Now, "stockReport.txt"));

var smtp = new SmtpClient

{

Host = "smtp.gmail.com",

Port = 25,

EnableSsl = true,

DeliveryMethod = SmtpDeliveryMethod.Network,

Credentials = new NetworkCredential(fromAddress.Address, fromPassword),

Timeout = 20000

};

using (var message = new MailMessage(fromAddress, toAddress)

{

Subject = subject,

Body = body,

})

{

message.Attachments.Add(attachmentfile);

smtp.Send(message);

MessageBox.Show("Email sent successfully");

}

}

}

catch

{

MessageBox.Show("Issues");

}

}

/\*Event handler for Exit button to save all the transactions to a text file\*/

private void buttonExit\_Click(object sender, EventArgs e)

{

try

{

/\*Write all the text in file \*/

File.WriteAllText(uniquetextfile,String.Empty);

/\*using nested loops to write all the value of stock in textfile\*/

for (int i=0; i <5; i++)

{

for (int j=0; j <16; j++)

{

StreamWriter outputfile;

outputfile = File.AppendText(uniquetextfile);

outputfile.WriteLine(BagelStock[i, j]);

outputfile.Close();

}

this.Close();

}

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

try

{

StreamWriter outputfiletransaction;

outputfiletransaction = File.AppendText(transactionfilepath);

foreach (var h in Alltransactions)

{

outputfiletransaction.WriteLine(h.ToString());

}

outputfiletransaction.WriteLine(Environment.NewLine);

outputfiletransaction.Close();

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void buttonCancel\_Click(object sender, EventArgs e)

{

panelStockReport.Visible = false;

}

public Form1()

{

InitializeComponent();

}

/\*Event handler for selecting Bagel type\*/

private void listBoxBagelType\_SelectedIndexChanged(object sender, EventArgs e)

{

try

{

/\*User should select atleast one activity\*/

if (listBoxBagelType.SelectedIndex != -1)

{

BagelTypeIndex = listBoxBagelType.SelectedIndex;

if (listBoxSize.SelectedIndex != -1)

{

pricing = BagelType[BagelSizeIndex, BagelTypeIndex].ToString();

labelBagelPrice.Text ="€"+(pricing);

}

}

}

/\*Statement will catch any exception thrown by try statement\*/

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/\*Created a class for email validation\*/

private bool validationEmail(String email)

{

try

{

MailAddress h = new MailAddress(email);

return true;

}

catch

{

return false;

}

}

/\*Event handler for Add to order button\*/

private void buttonOrder\_Click(object sender, EventArgs e)

{

decimal Price;

if (textBoxQuantity.Text == "" )

{

MessageBox.Show("Please enter the quantity");

}

else

{

/\*Taking input for quantity and storing it in a variable\*/

Quantity = int.Parse(textBoxQuantity.Text);

}

try

{

/\*Taking input for quantity and storing it in a variable\*/

/\*parsing the value of variable\*/

BagelCost = decimal.Parse(pricing);

/\*storing the value of element at that index and storing it in variable\*/

stock = BagelStock[BagelSizeIndex, BagelTypeIndex];

/\*calculating the priceline for each items\*/

priceLine = (Quantity \* BagelCost);

/\*validating if inputed text is not a zero\*/

if (Quantity != 0)

{

/\*if stock is greater then quantity updating the stock array \*/

if (stock >= Quantity)

{

Price = (Quantity \* BagelCost);

TotalPrice = (Price) + (TotalPrice);

labelTotalPrice.Text = TotalPrice.ToString("c2");

stock -= Quantity;

BagelStock[listBoxSize.SelectedIndex, listBoxBagelType.SelectedIndex] = stock;

/\*creating an object of a class\*/

Bagel info = new Bagel();

/\*Accessing each value entered by the user\*/

info.BagelName = listBoxBagelType.SelectedItem.ToString();

info.SizeofBagel = listBoxSize.SelectedItem.ToString();

info.QuantityofBagel = Quantity;

info.priceofBagel = priceLine;

info.TotalPriceofBagel = TotalPrice;

info.BagelSizeIndex = listBoxSize.SelectedIndex;

info.BagelTypeIndex = listBoxBagelType.SelectedIndex;

/\*Adding all the details which was accessed in array list\*/

BagelDetails.Add(info);

}

else

{

MessageBox.Show("Remaining stock for selected Bagel is :" + stock);

}

}

else

{

MessageBox.Show("Please enter the valid amount");

}

/\*cursor will be set to textbox to get input for quantity\*/

textBoxQuantity.Focus();

buttonCompleteOrder.Enabled = true;

}

catch(Exception ex)

{

MessageBox.Show("Please select Bagel Type and Size");

}

}

/\*Created a public Class to declare methods which can be handled by the object \*/

public class Bagel

{

/\*Declared\*/

public string BagelName { set; get; }

public string SizeofBagel { set; get; }

public int QuantityofBagel {set;get;}

public decimal priceofBagel { set; get; }

public decimal TotalPriceofBagel { set; get; }

public int BagelSizeIndex { set; get; }

public int BagelTypeIndex { set; get; }

}

/\*Event hanlder for Initial form of load\*/

private void Form1\_Load(object sender, EventArgs e)

{

/\*Groupbox will not be visible on initial form load\*/

groupBoxSummary.Visible = false;

panelStockReport.Visible = false;

dataGridViewBagel.Visible = false;

buttonCompleteOrder.Enabled = false;

}

}

}

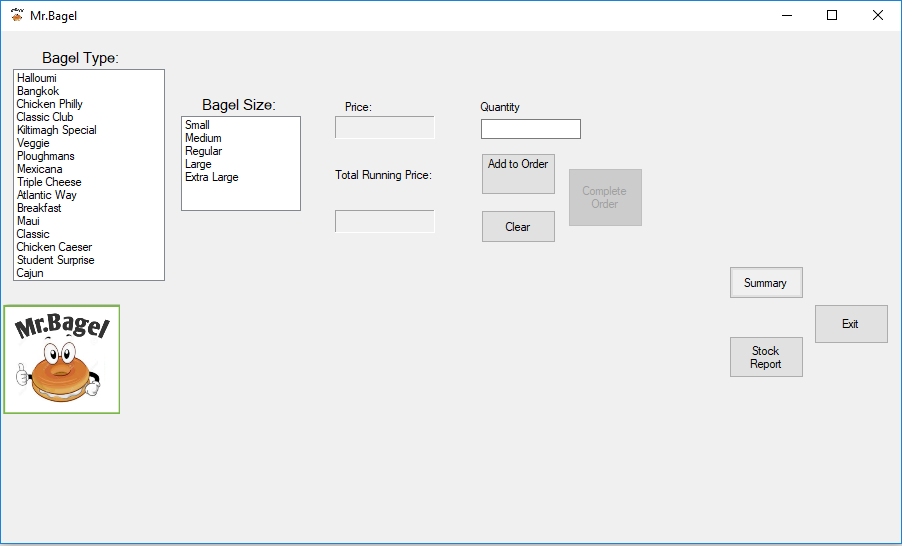


Figure Initial form load

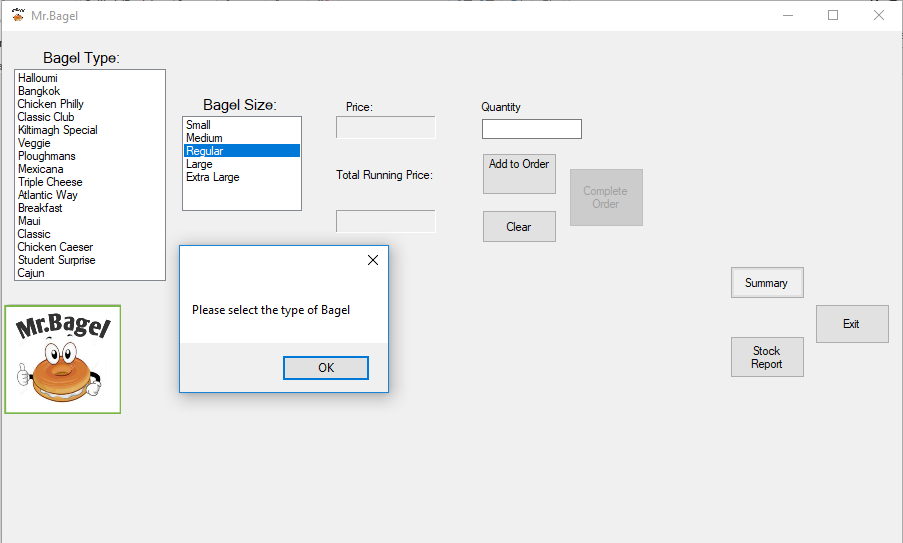


Figure Must select Bagel type and bagel size to view price

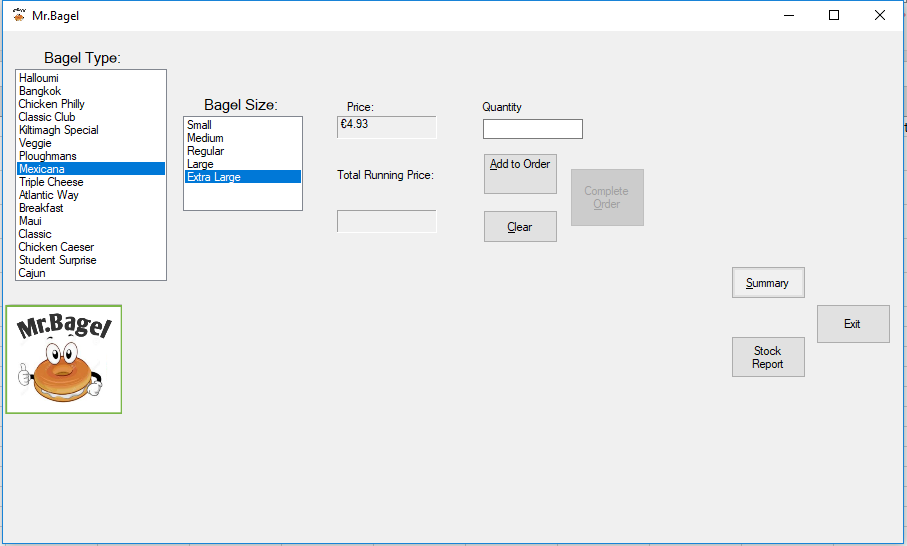


Figure View Price of selected Bagel Type and Size

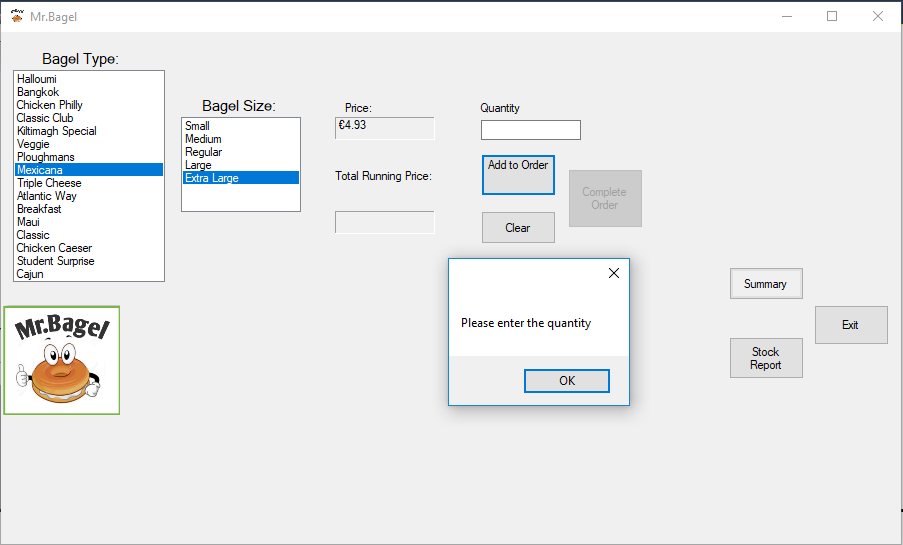


Figure Quantity textbox validation

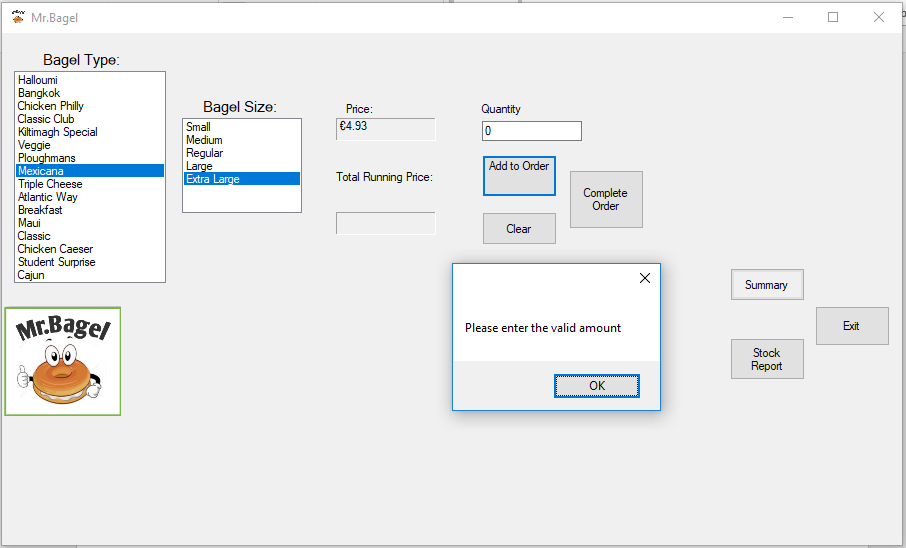


Figure Quantity textbox validation for zero quantity

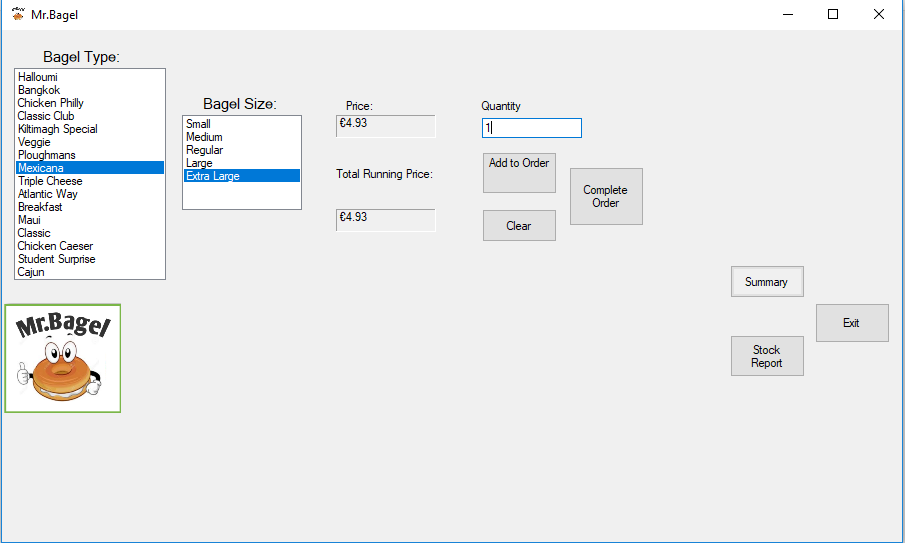


Figure Entering the quantity and pressing add to order button

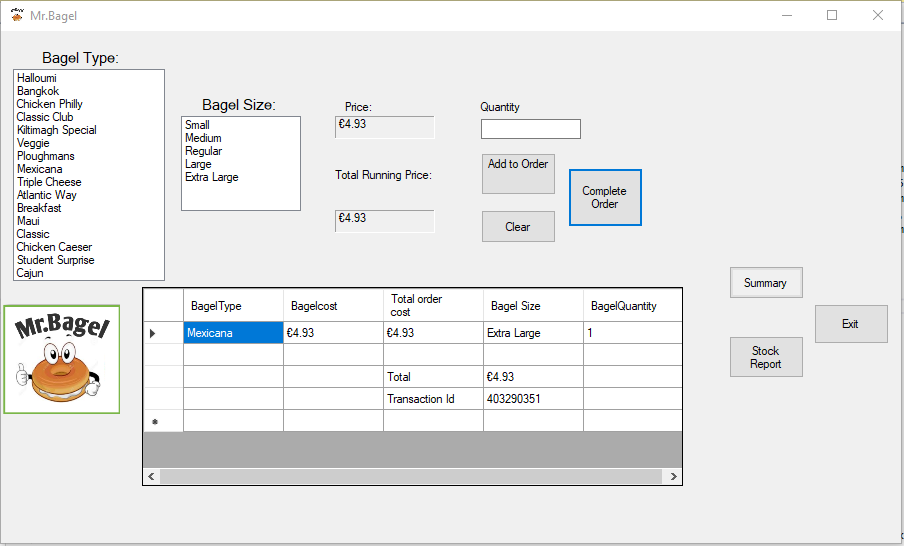


Figure After Pressing Complete order button ,data grid box appears mentioning all the details

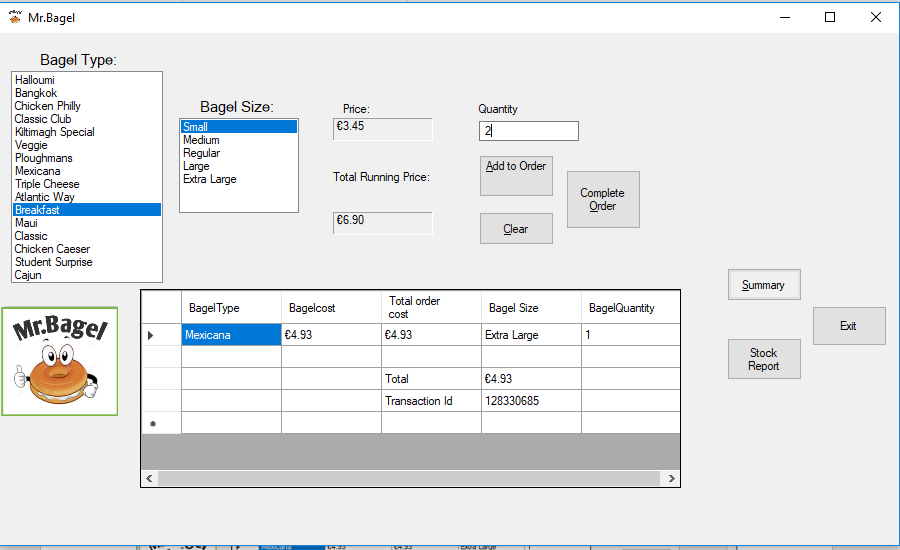


Figure Choosing other bagel type, sizes and entering quantity

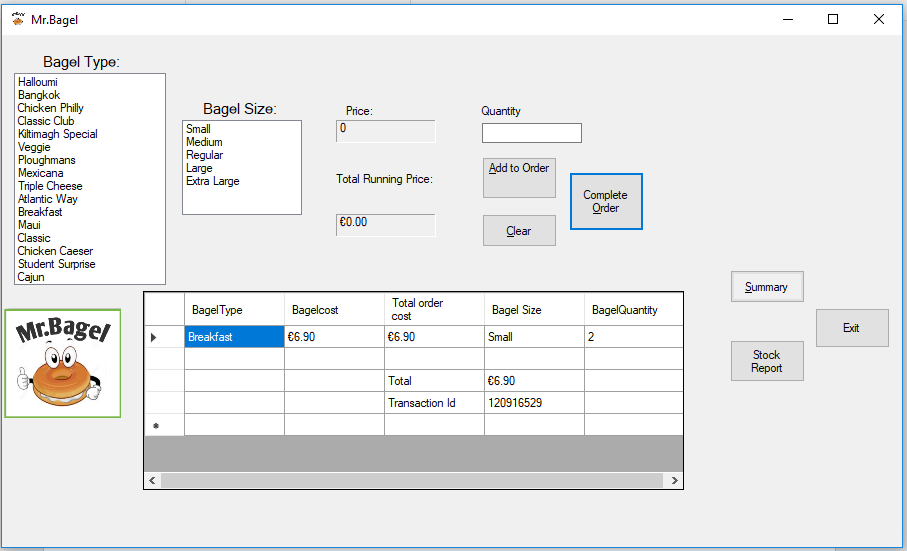


Figure Adding to order and then confirm

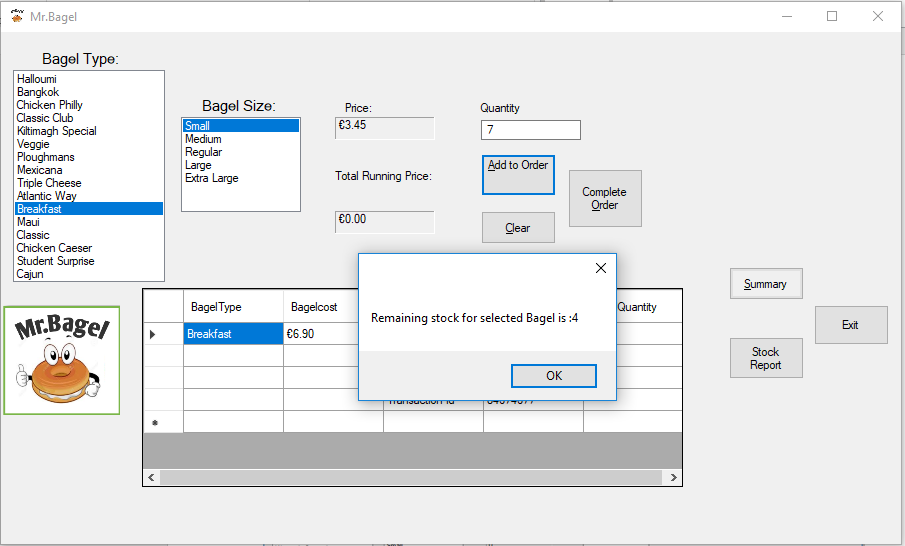


Figure Inserted quantity amount is out of stock

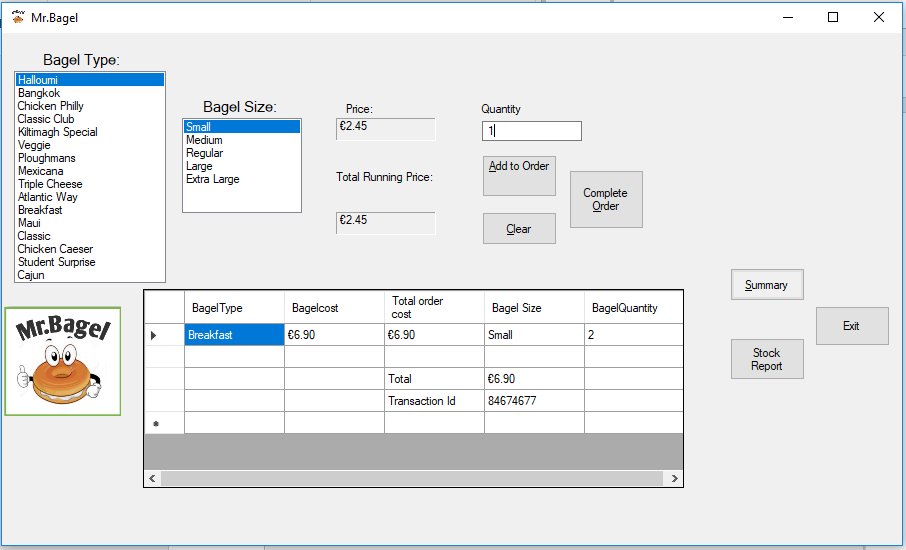


Figure Making again a different order

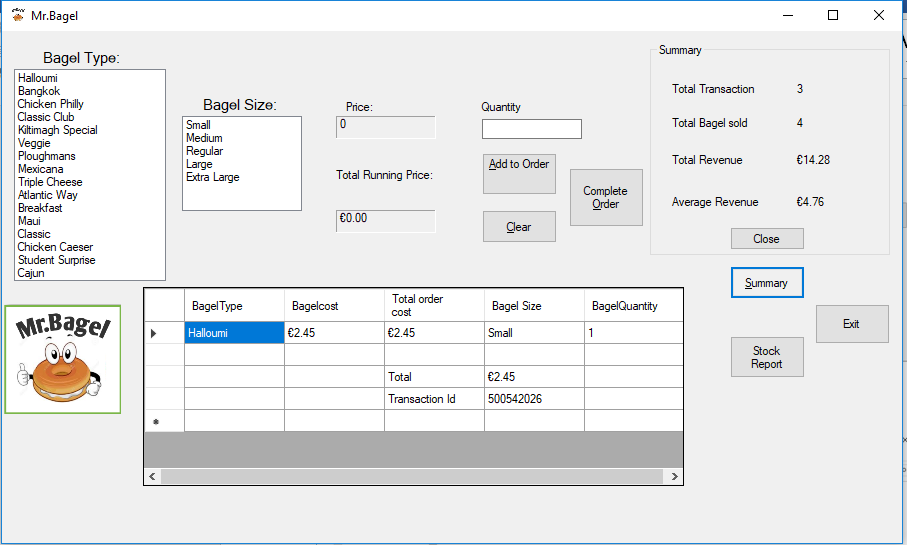


Figure To view the transaction details till now

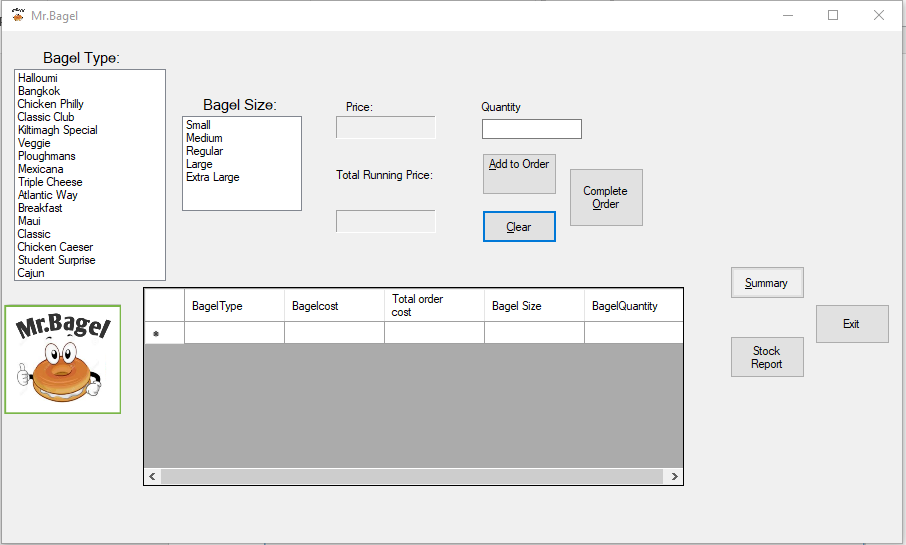
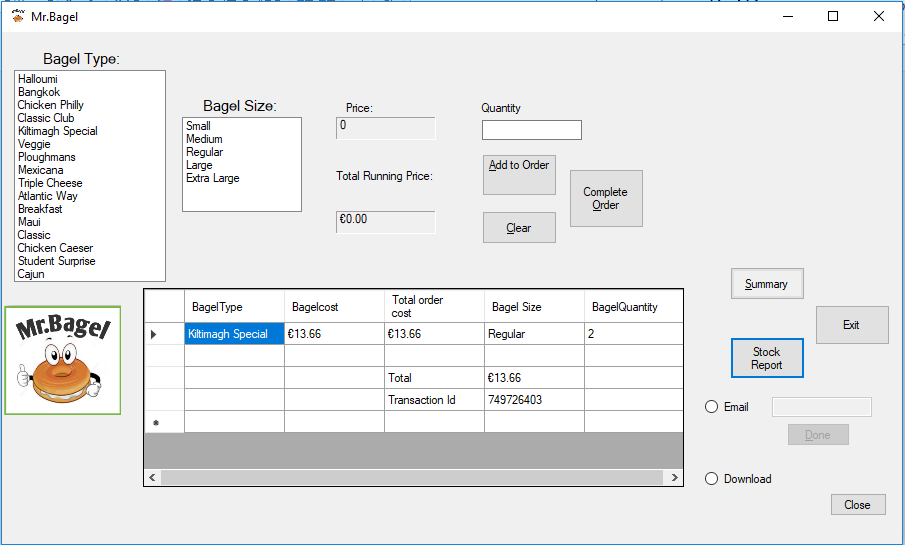


Figure On pressing Clear button ,all the data are cleared on the interface



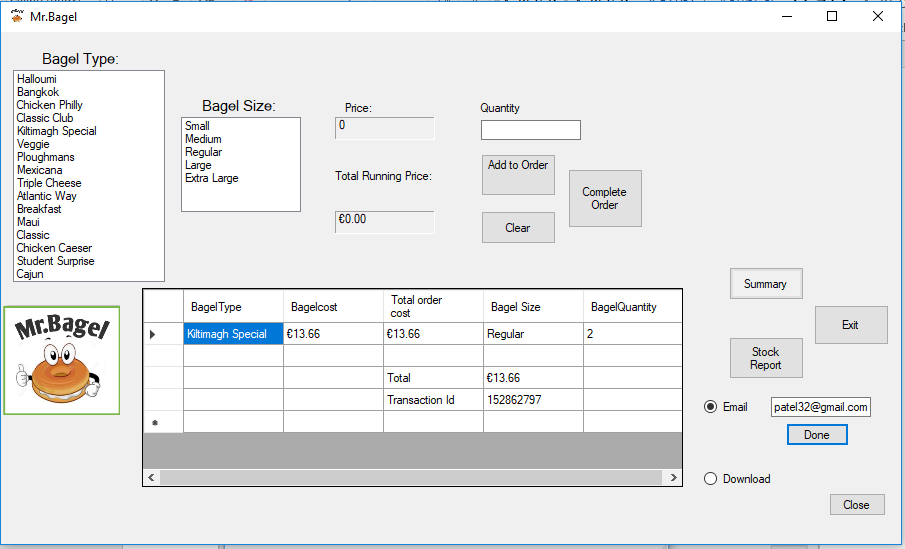


Figure Choosing Email as an option, Entering the valid email Id and clicking Done

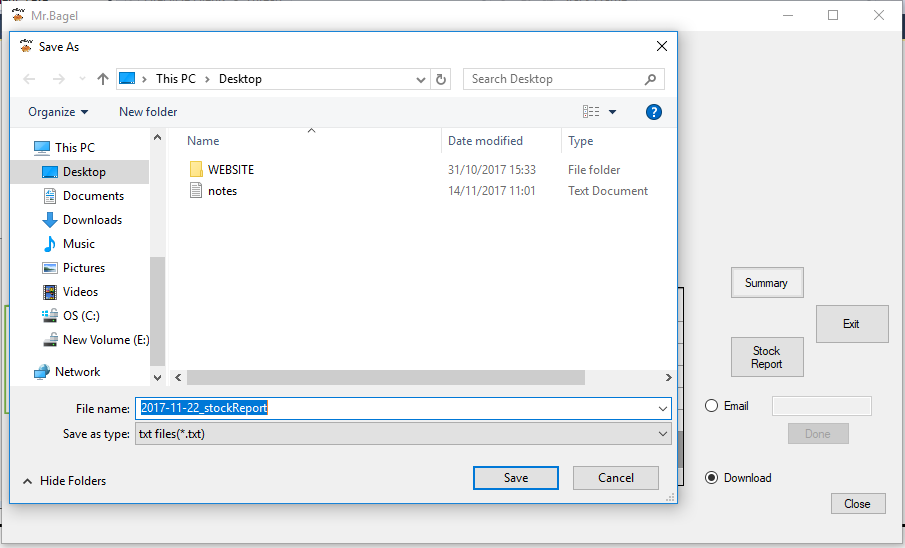


Figure If choosen downloaded option stock report is downloaded to the system

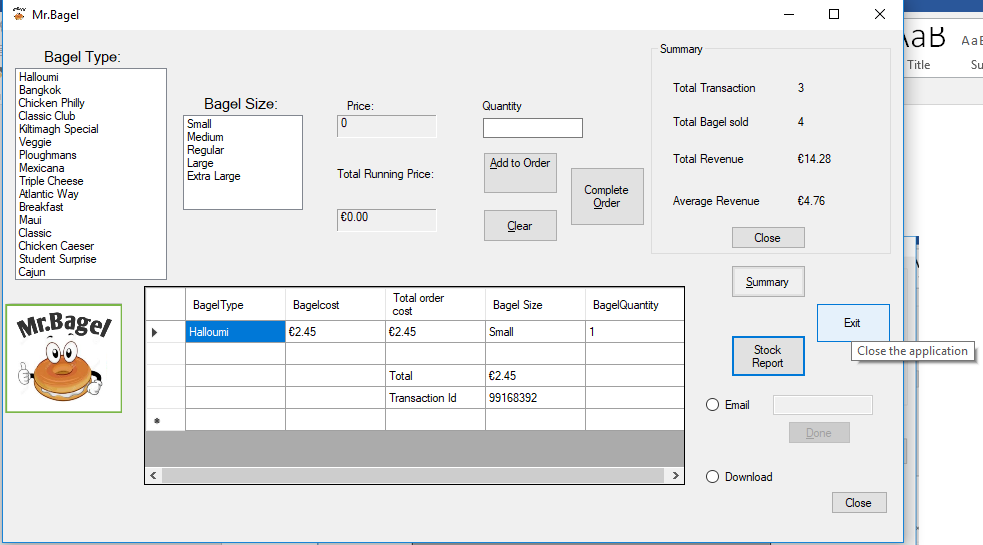


Figure Click on Exit to close the application .

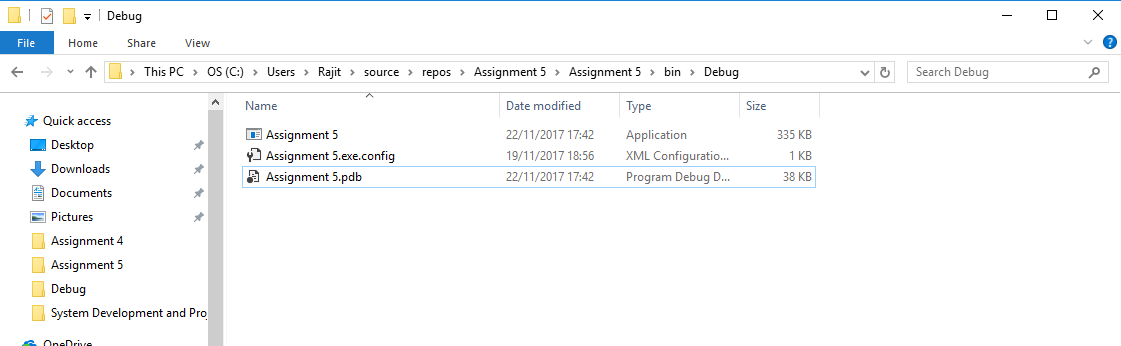


Figure 17Initially No files are present in Bin directory

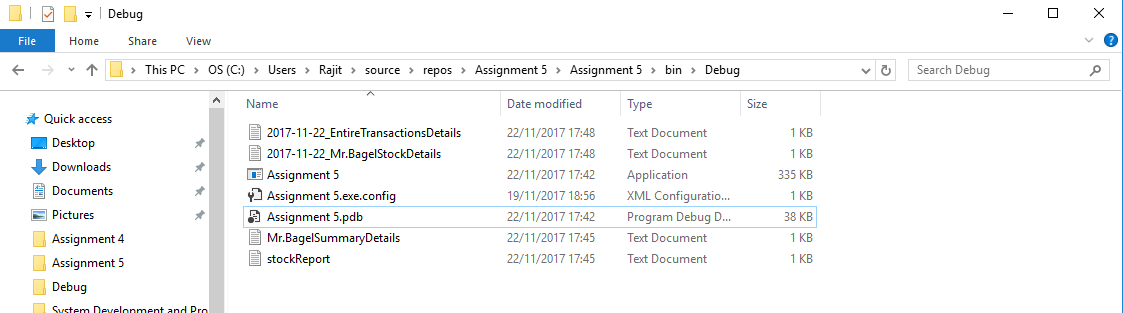


Figure As soon as stock button and exit button is pressed, reports are generated

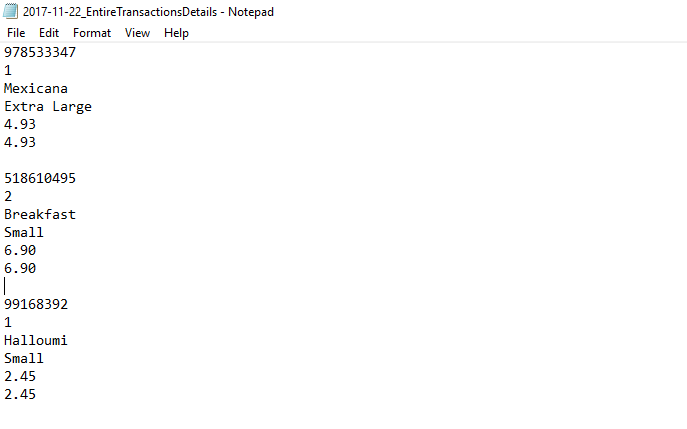


Figure Transaction Details



Figure Stock Details

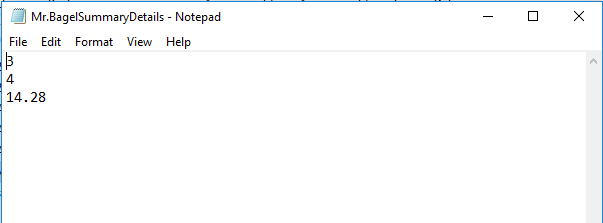


Figure Summary Details

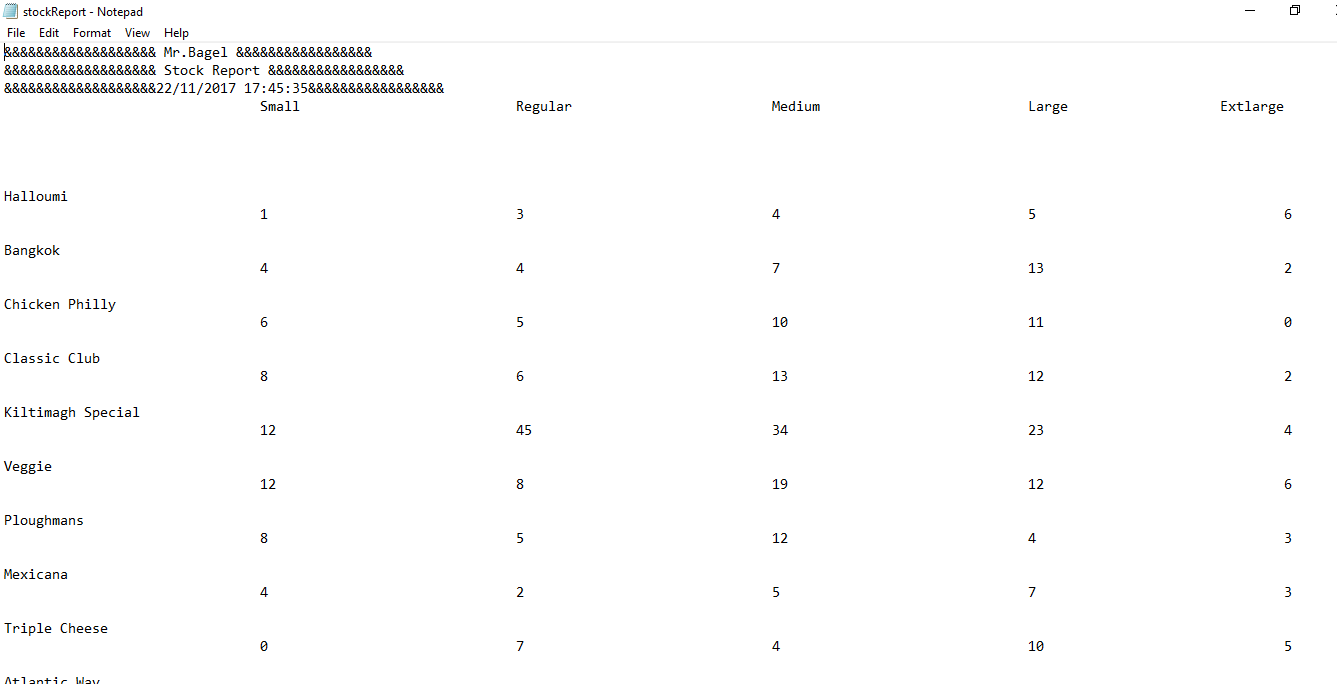


Figure Stock Report

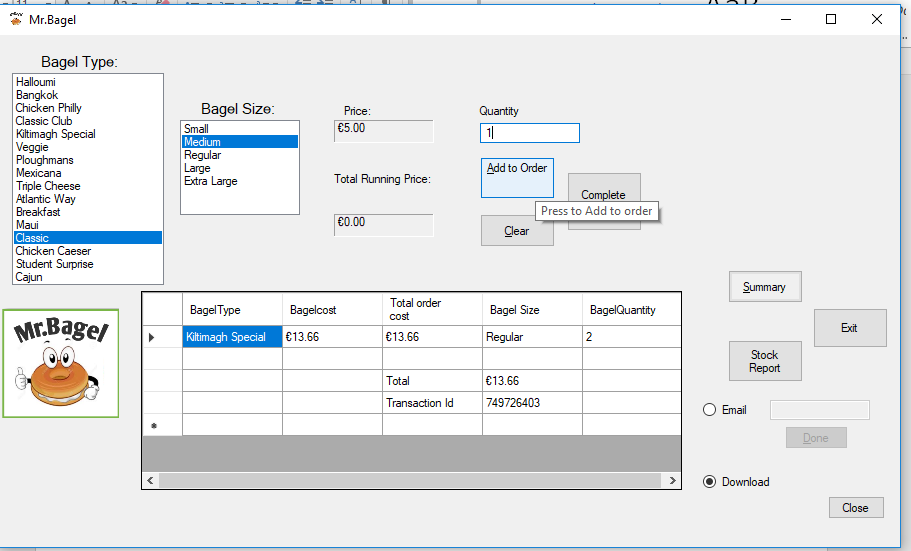


Figure Use of tooltips

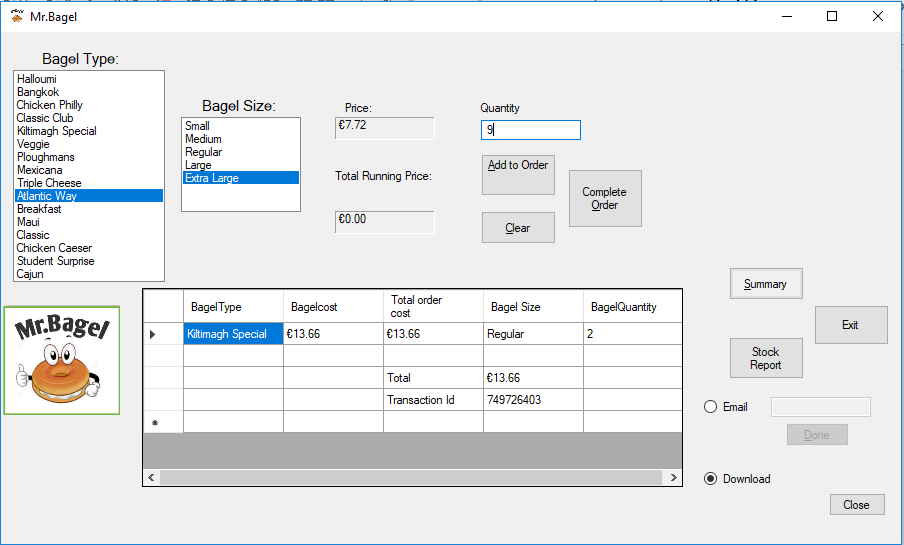


Figure Use of Access keys