**Version 3**



In this version of my project I will be focusing on 2 main segments:

1. Code Efficiency and Validation:

* I will try to remove repetitive code which I have a lot of by creating public sub-routines in a new module.
* For example, I will transfer my Refresh Button’s code to the module and pass in a number of variables into the parameters. When it comes to using the code in a form, I will simply call the sub-routine and pass in appropriate values.
* I will need to prevent any future errors that could exist by limiting what the user can input and what type of values they can input.

1. Adding in extra features to improve usability requested by my stakeholder Ben in my previous version of development. These features are:

* Create an option to make sure if the use wants to log out
* Create an option to make sure if the user wants to exit the program
* Fix the edit button
* Make the search button more efficient
* Add a feature that creates two different logins that restrict access to specific features
* Add a feature that counts the number of vehicles that an employee sells
* I have also decided to add an extra column to the vehicle database that tells the customer whether the vehicle is manual or automatic.
* Add an extra textbox to the Customer/Vehicle/Staff Database Forms to ease searching for a customer.

I will first focus on code efficiency (parameter passing) because this will ease tracking the contents of the program and it will reduce the memory consumption of the program.

All the code will be in a module called “ModuleCode”.

During the development of the Add Button, I encountered a problem in which I could not pass parameters for lines 3-10

1 TextBoxID.Text = NumOfCustomers + 1

2 With CustomerDatabase(NumOfCustomers)

3 'Puts values from all textboxes into the record elements

4 .ID = TextBoxID.Text

5 .Forename = TextBoxForename.Text

6 .Surname = TextBoxSurname.Text

7 .Contact = TextBoxContactNo.Text

8 .Address = TextBoxAddress.Text

9 .Email = TextBoxEmail.Text

10 End With

I have 3 structures in my program and all 3 structures have a different number of variables in them, meaning I’d have to pass in a large number of variables in the parameters and therefore would be too time consuming. My final decision for this is to have 3 different Add buttons for the 3 different structures like in my previous versions but they will now be public. I will also add some validation to this button to avoid future errors. The validation that I will add will be an if statement to check whether any of the variables are empty to avoid problems when it comes to deleting, searching and refreshing the data-grid-view.

The final code for the Add Button:

Public Sub AddCustomer()

CustomerDatabaseForm.TextBoxID.Text = NumOfCustomers + 1

'Increments the number of records by 1

With CustomerDatabase(NumOfCustomers)

.ID = CustomerDatabaseForm.TextBoxID.Text

.Forename = CustomerDatabaseForm.TextBoxForename.Text

.Surname = CustomerDatabaseForm.TextBoxSurname.Text

.Contact = CustomerDatabaseForm.TextBoxContactNo.Text

.Address = CustomerDatabaseForm.TextBoxAddress.Text

.Email = CustomerDatabaseForm.TextBoxEmail.Text

End With

'Puts values from all textboxes into the record elements

Dim File As StreamWriter

File = My.Computer.FileSystem.OpenTextFileWriter(VehicleFilename, True)

With CustomerDatabase(NumOfCustomers)

If .ID = "" Or .Forename = "" Or .Surname = "" Or .Contact = "" Or .Address = "" Or .Email = "" Then

'Do Nothing

Else

Dim oneline As String = .ID & sep & .Forename & sep & .Surname & sep & .Contact & sep & .Address & sep & .Email

File.WriteLine(oneline)

'write the line to file

End If

End With

File.Close()

End Sub

The development of the Delete Button was successfully done.

Public Sub Delete(Filename As String, TextBox As String, numofrecords As Integer)

Dim list As New List(Of String)()

Using Deletereader As New StreamReader(Filename)

While Not Deletereader.EndOfStream

Dim FullLine As String = Deletereader.ReadLine()

Dim Item() As String = Split(FullLine, sep)

If Item(0) = TextBox Then

'Do nothing

Else

list.Add(FullLine)

End If

End While

End Using

Dim NewNumOfRecords As Integer = list.Count

Using writer As New StreamWriter(Filename, False)

For i = 0 To NewNumOfRecords - 1

writer.Write(list(i) & vbCrLf)

Next

numofrecords = NewNumOfRecords

End Using

MessageBox.Show(numofrecords)

End Sub

To demonstrate how this would work, I would pass in 3 values for the 3 variables.

E.g. Private Sub cmdDeleteRow\_Click(sender As Object, e As EventArgs) Handles cmdDeleteRow.Click

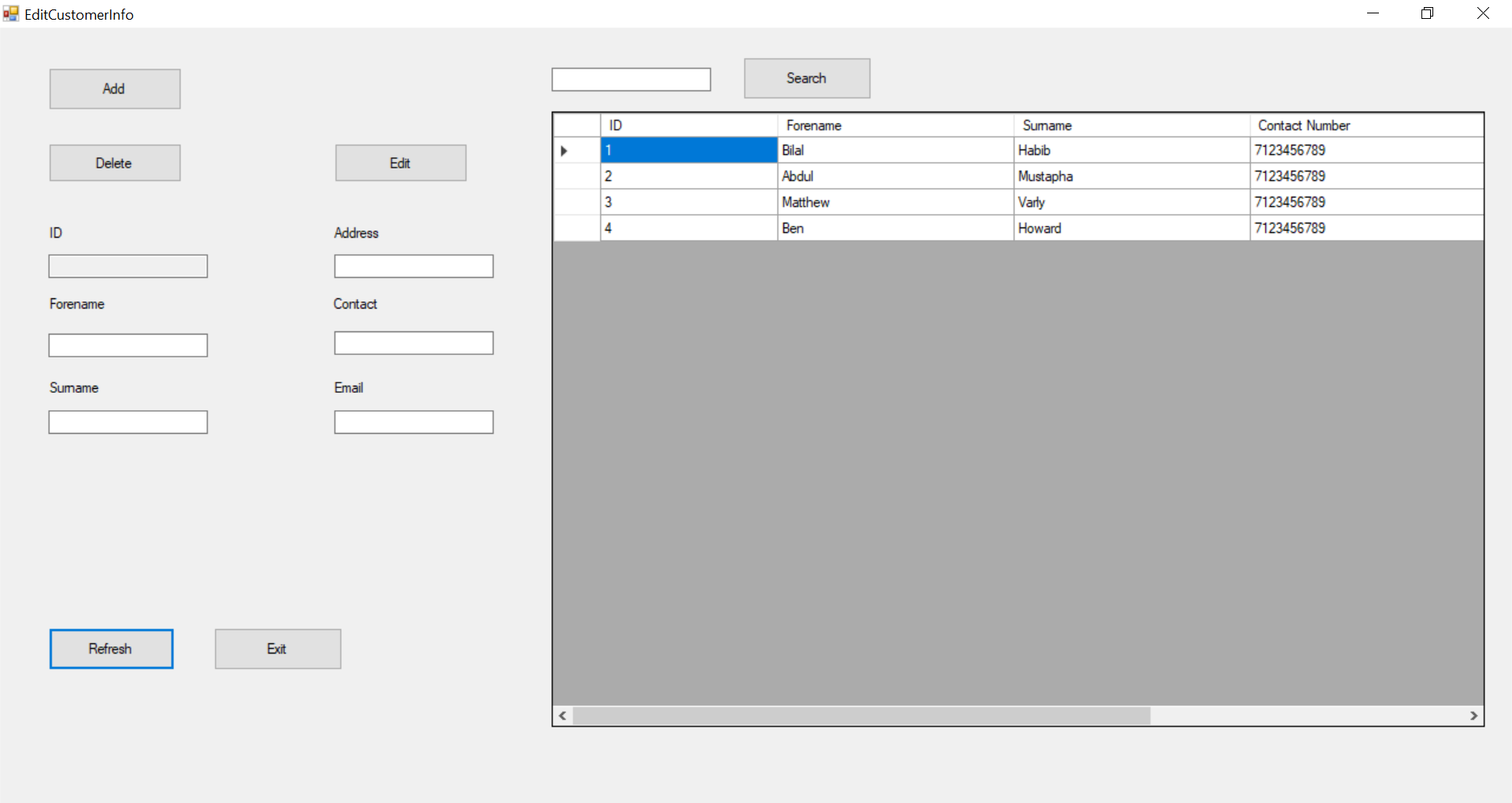
Delete(Customerfilename, TextBoxID.Text, NumOfCustomers)

End Sub

As you can see, instead of re-writing the whole delete button again whenever I need to use it, I can simply pass in 3 values for the variables that I passed in and all it took was a line of code.

Due to the stakeholder’s response from the previous version of development, I added an extra Textbox for Searching for a record to create ease for the user.

Due to this new textbox, I changed the location of the button so that it is right next to the new textbox. This shows that the button was made for that specific textbox and reduces confusion. I put the textbox on top of the data-grid-view so that it is not to be confused with any of the other textboxes.



The search button does not require any validation, so all I need to do is write its code in a module and pass in parameters.

Public Sub SearchRecord(filename As String, DataGrid As DataGridView, Textbox As String)

Dim columns\_expected As Integer = DataGrid.Columns.Count

Dim condition As Boolean

DataGrid.Rows.Clear()

Using SearchReader As New StreamReader(filename)

While Not SearchReader.EndOfStream

Dim FullLine As String = SearchReader.ReadLine()

If FullLine.Contains(Textbox) Then

Dim SeparatedLine() As String = FullLine.Split(",")

DataGrid.Rows.Add("")

If SeparatedLine.Length = columns\_expected Then

For ix As Integer = 0 To columns\_expected - 1

DataGrid.Rows(DataGrid.Rows.Count - 1).Cells(ix).Value = SeparatedLine(ix)

Next

End If

condition = True

End If

End While

End Using

If condition = True Then

MessageBox.Show("Record Found!")

Else

MessageBox.Show("No Results!")

End If

End Sub

The code below is an example of the search being called in the Customer Database Form and is made sure to be passed in the correct order.

Private Sub cmdSearchCustomer\_Click(sender As Object, e As EventArgs) Handles cmdSearchCustomer.Click

SearchRecord(Customerfilename, CustomerDataGridView, TextBoxSearch.Text)

End Sub

During the development of the Edit Button in version 2, I encountered a problem where the record wasn’t getting edited. Although I could not edit the record on the same index, I came up with a different solution. The new Edit Button is my Delete Button followed by the Add Button.

I have not created a sub-routine for the Edit Button because I will still need to pass in parameters.

E.g. Customer Database Form

Private Sub cmdEdit\_Click(sender As Object, e As EventArgs) Handles cmdEdit.Click

Delete(Customerfilename, TextBoxID.Text, NumOfCustomers)

AddCustomer()

End Sub

The Refresh Button was successfully written in the module and there were no problems and it does not require any validation.

Public Sub RefreshDatabase(Datagrid As DataGridView, filename As String, columns\_expected As Integer)

Datagrid.Rows.Clear()

'Acts as refresh button where we keep importing the customer database

Dim reader As New StreamReader(filename)

Dim FullLine As String = ""

Do

FullLine = reader.ReadLine

If FullLine Is Nothing Then Exit Do

Dim SeparatedLine() As String = FullLine.Split(",")

Datagrid.Rows.Add("")

If SeparatedLine.Length = columns\_expected Then

For ix As Integer = 0 To columns\_expected - 1

Datagrid.Rows(Datagrid.Rows.Count - 1).Cells(ix).Value = SeparatedLine(ix)

Next

End If

Loop

reader.Close()

End Sub