**Version 1**

In this project, I will have to break up the modules and classes into different sections. This makes it easier to control and coordinate my project.



In this version, I will be focusing on the main code that needs to be done for the program to successfully work and focus on extra features in later versions of the project. This means that the code in this version will be simple.

I will be focusing on creating: 1) An Add Button

2) A Delete Button

3) A Login

4) A Refresh Button

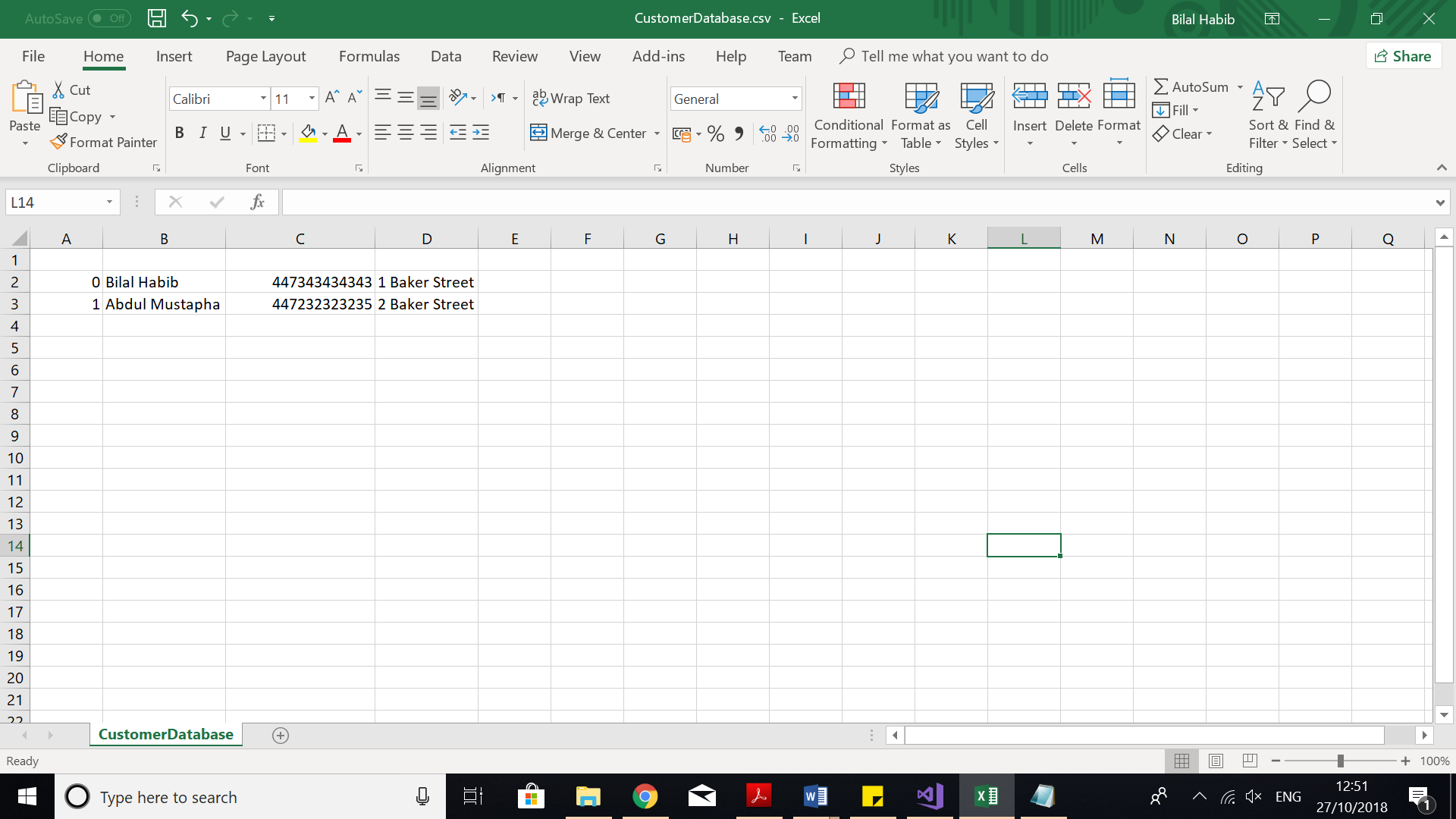
5) Databases for the forms

6) A Customer Database Form

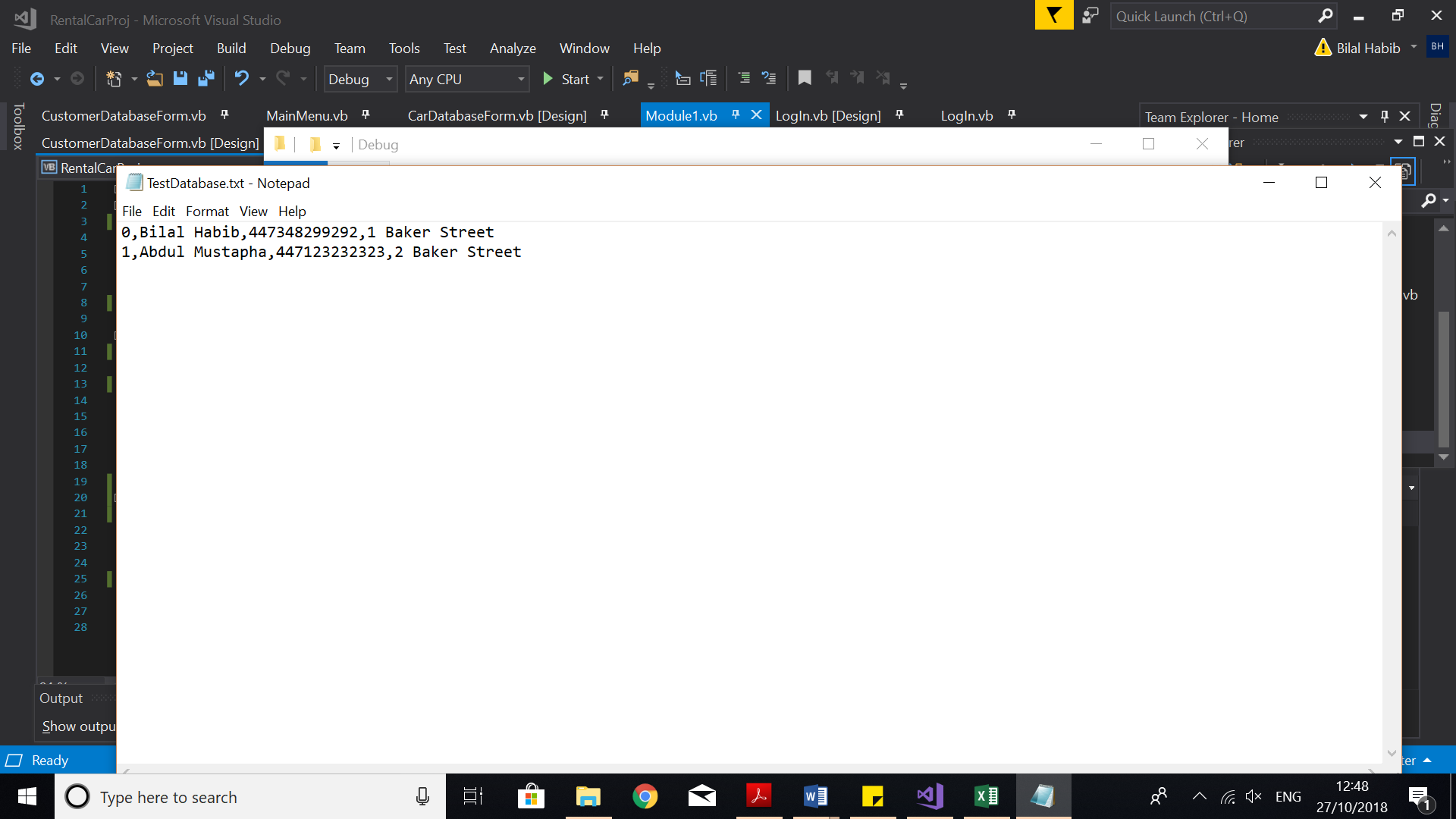
7) A Vehicle Database Form

8) Loading the data from the csv files

During this project, I had the decision to either use a text file or a csv file to act as a database and hold the details of the customers and vehicles. I made the decision of a csv file because the code would have been exactly the same as if I used a text file. This is because they both use the same method in terms of separating the values in a record which is simply putting a comma between each value. A csv file would provide a better user interface than a text file because it looks exactly like a database as it provides a data grid view.



If I were to make the text file look good, I would have to include extra lines of unnecessary code. This could potentially make more errors and waste my time on concentrating on more important and complex code.



In this project, I have created a module to hold all the global variables that my project needs.

Module Module1

Structure Customer

Dim ID As Integer

Dim Name As String

Dim Contact As String

Dim Address As String

End Structure

'Creates a structure for customers

Structure Vehicles

Dim ID As Integer

Dim Name As String

Dim CostPerDay As String

End Structure

'Creates a structure for Vehicles

Public NumOfCustomerRecords As Integer = 0

Public CustomerDatabase(9) As Customer

Public Customerfilename As String = "CustomerDatabase.csv"

'Setting the number of records that the database can hold

'Attaching the file name to a string to make it easier type up

Public NumOfVehicleRecords As Integer = 0

Public VehicleDatabase(9) As Vehicles

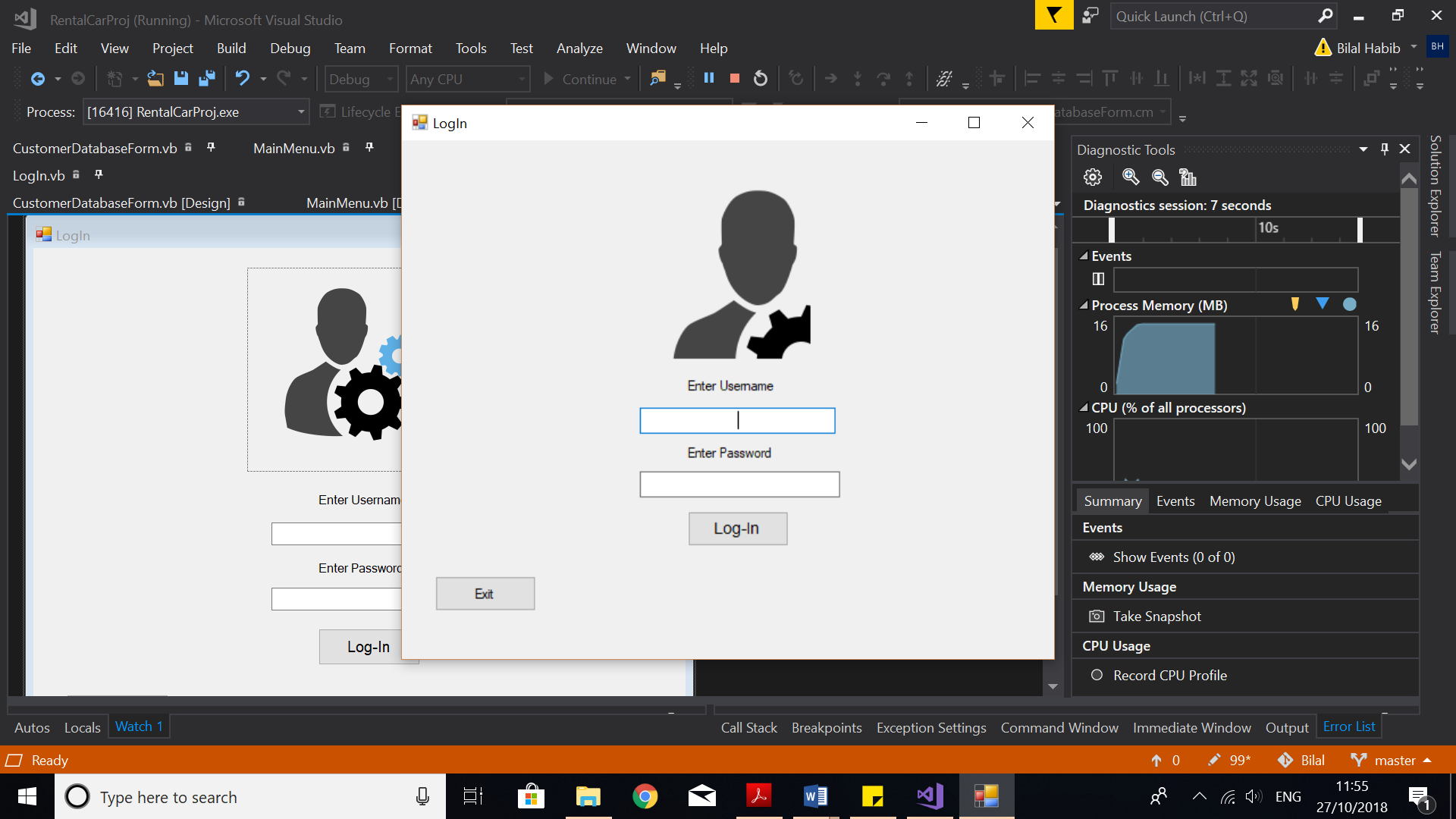
Public Vehiclefilename As String = "VehicleDatabase.csv"

End Module

I have used structures to hold the values of the vehicles and customers because a structure can contain an array as one or more of its elements. You can also declare an array of structures, E.g. CustomerDatabase(10) As Customer. Even though the array holds 10 records, I have adapted my code so that it is limitless. I have done this because I do not see why a company would want a limit of customers, plus it only contains text meaning it will not take up a lot of storage. I have also created the variables “NumOfCustomerRecords” and “NumOfVehicleRecords” to loop through the databases and from there add data to them.

**Log-In Form**

When the user loads my program, they will be greeted by this Login. As you can see, it is a very basic login with clear instructions to not confuse the user. I have chosen a similar colour for the background and buttons to not waste time on its design which will allow more time to focus on the code. For version 1, I have stuck to a very basic login in which there is only one person able to login.



I have used two variable naming conventions throughout all my code. E.g. cmdExitApp, CmdExitApp

Public Class LogOnForm

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

LabelLogin.Visible = False

If username.Text = "Admin" And password.Text = "AdminPass" Then

MainFormMenu.Show()

LabelLogin.Visible = True

Me.Hide() 'This hides the login form as there will be no need to login again and prevents errors

MainFormMenu.Show() 'If login is successful, then the main menu opens up

Else

LabelLogin.Visible = True

LabelLogin.Text = "Login Failure"

password.Text = "" 'Password Text cleared to ease retyping it

'Label of success/failure shown due to success criteria

End If

End SubPrivate Sub cmdExitApp\_Click(sender As Object, e As EventArgs) Handles cmdExitApp.Click

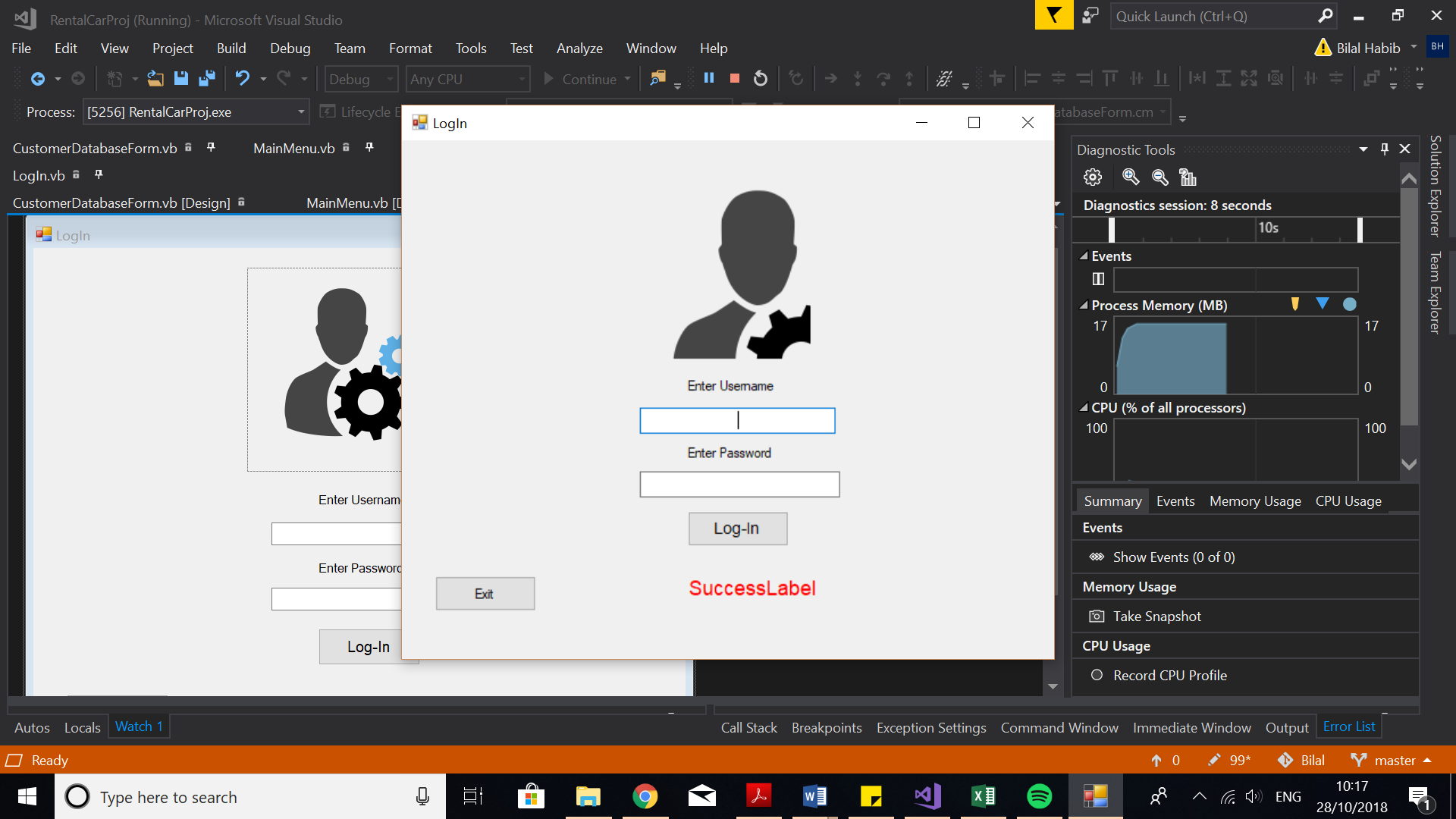
Close()

End Sub

End Class

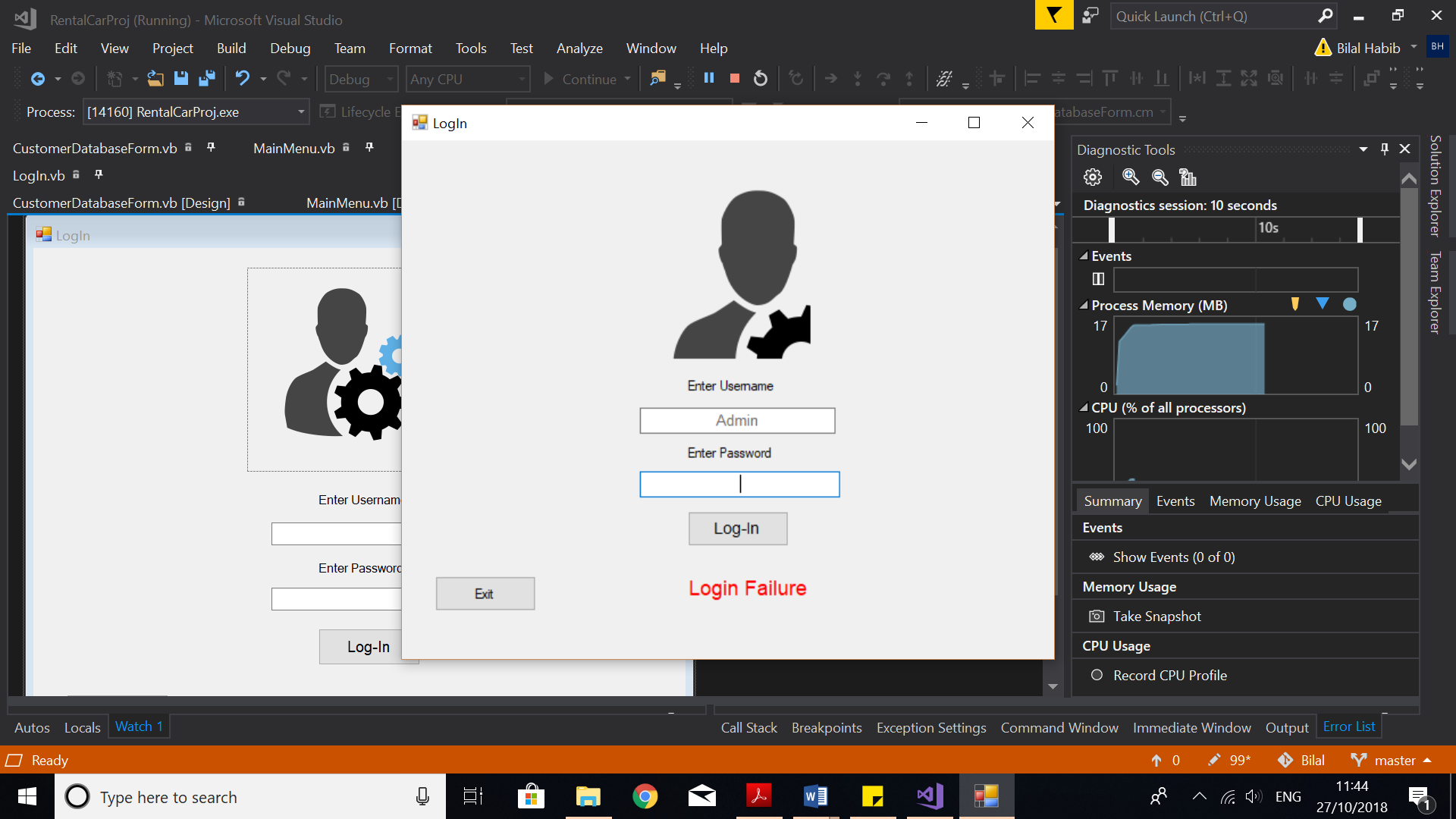
Here is the code for the login. As you can see, if the details that the user enters are correct, the main menu will be shown, and the login form will be hidden. Otherwise, the login form will still be shown, and the main menu will be hidden. I have set the success label to be ‘True’ in the code because I have it set as ‘False’ by default to provide a nicer user interface and not confuse the user with a Read-Only label. I have not included any validation as the code is simple and therefore does not require any. I have included an exit button to make it easier for the user to exit the application.

If the success label was always true, the user would have look at this screen:



As you can see, the user would be confused to see the label and therefore its visibility has been set as False.

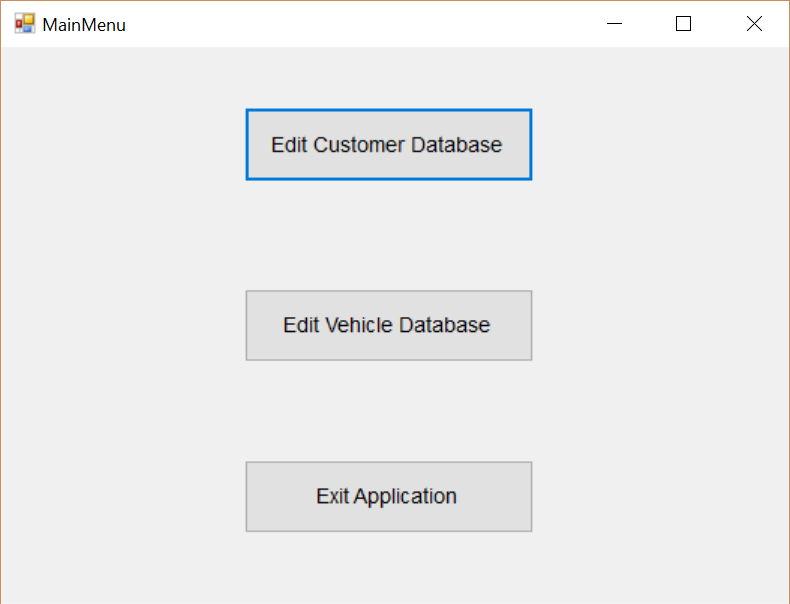
If the Login is unsuccessful, then the main menu will not be shown, and a label will pop up saying “Login Failure”. The textbox will be clear after a failed attempt to ease re-entering the password.



I have chosen the success label to be a different colour and a larger font then the normal text, so that it stands out and informs the user of an important message

If the Login is successful, then the Main Menu Form will be shown, and the Login Form will be hidden.

**Main Menu Form**



I have used two variable naming conventions throughout all my code. E.g. cmdExitApp, CmdExitApp

From there depending on which button you click, allows you to either edit the customer or vehicle database (or exiting the application). I have made the main menu basic with clear instructions as it does not need to be complicated. I have chosen the same colour for the background and buttons to not waste time on its design which will allow more time to focus on more important code.

Public Class MainFormMenu

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

CustomerDatabaseForm.Show()

Me.Hide()

End Sub

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

VehicleDatabaseForm.Show()

Me.Hide()

End Sub

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

Application.Exit()

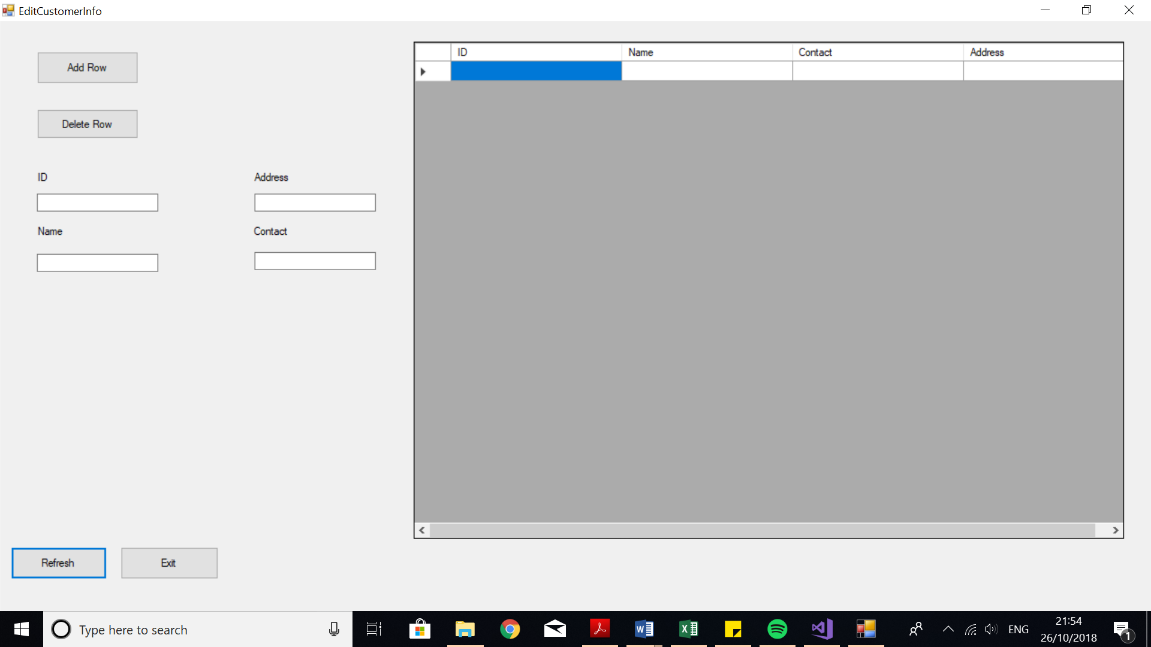
End Sub

End Class

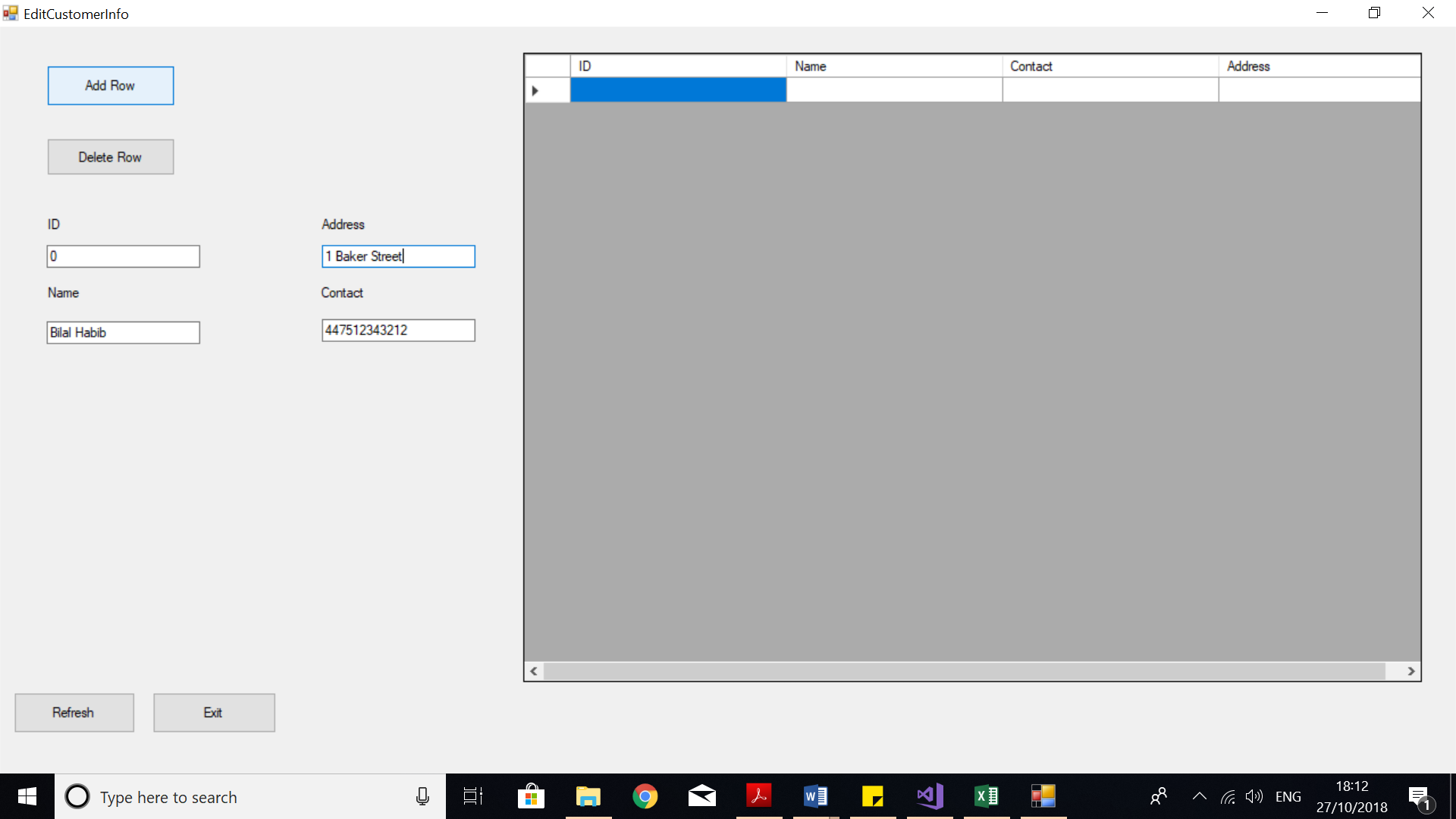
I have also taken advantage of the space that I have in the form as there are only three buttons by making the buttons larger and the font larger, to provide a nicer user interface and a more easily readable interface. I have not annotated this code as it is self-explanatory and therefore confusions will not occur in the future.

**Customer Database Form**

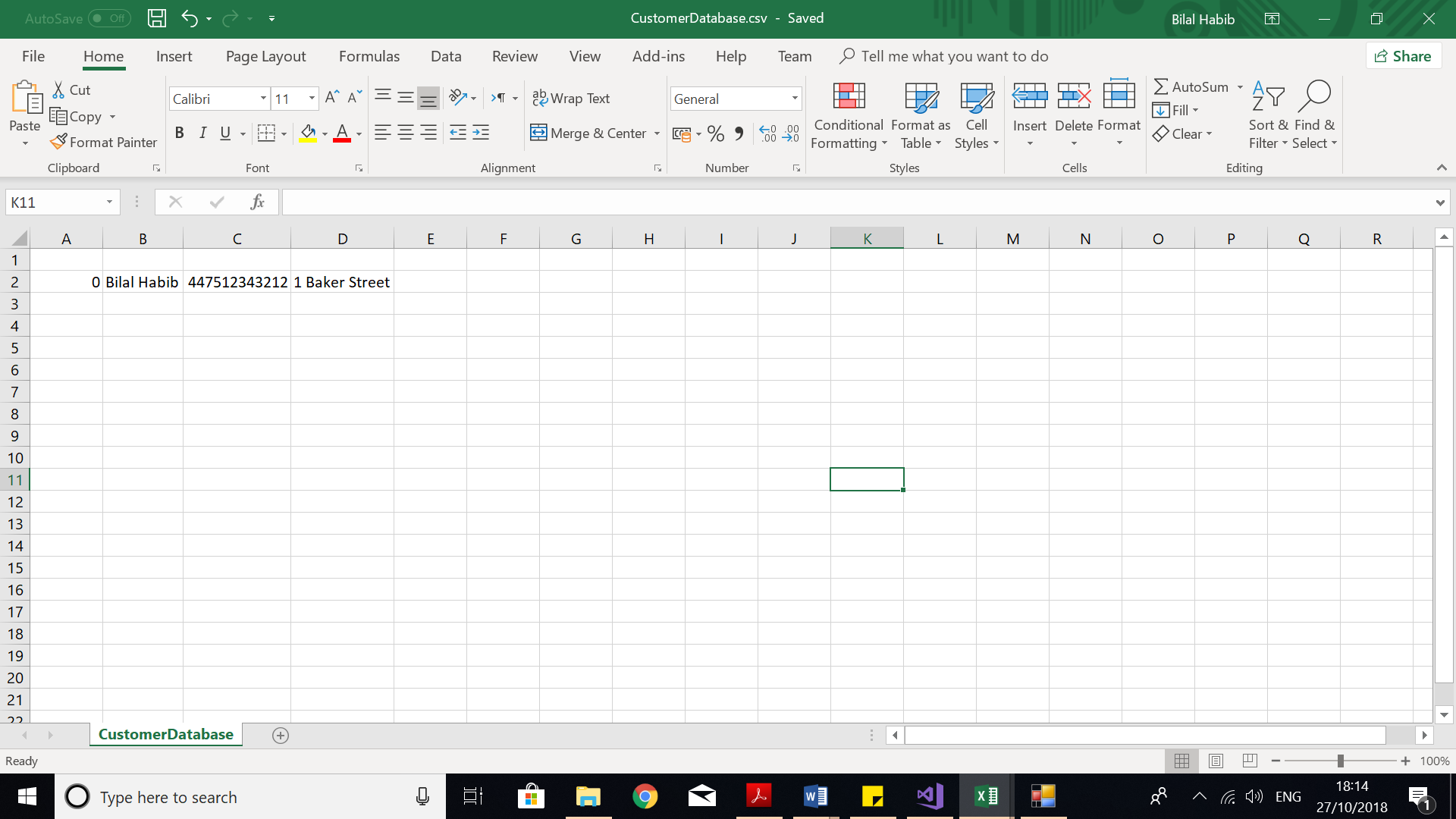
If the Edit Customer Database button is pressed, the customer database form will be shown, and the Main Menu Form will be hidden.



When details are entered into the textboxes, the user will have the option to add those details and make that a record in the database by clicking the Add button as shown below:



When the add button gets clicked, the details get added to the csv file (database).



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

With CustomerDatabase(NumOfCustomerRecords) 'Puts values from all textboxes into the record elements

.ID = TextBoxID.Text

I assigned the values from the textboxes into the variables created in the ‘Customer’ Structure.

.Name = TextBoxName.Text

.Contact = TextBoxContactNo.Text

.Address = TextBoxAddress.Text

End With

I then cleared all the textboxes to make it easier for the user to type in details into the textboxes again instead of manually deleting everything.

TextBoxID.Text = ""

TextBoxName.Text = ""

TextBoxContactNo.Text = ""

TextBoxAddress.Text = ""

Dim sep As String = "," ' the symbol to separate the fields on one line

Dim File As StreamWriter

StreamWriter is used to write text to files

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

For i As Integer = 0 To NumOfCustomerRecords 'loop through all the records

With CustomerDatabase(i)

Dim oneline As String = .ID + sep + .Name + sep + .Contact + sep + .Address

File.WriteLine(oneline) 'write the line to file

End With

Next

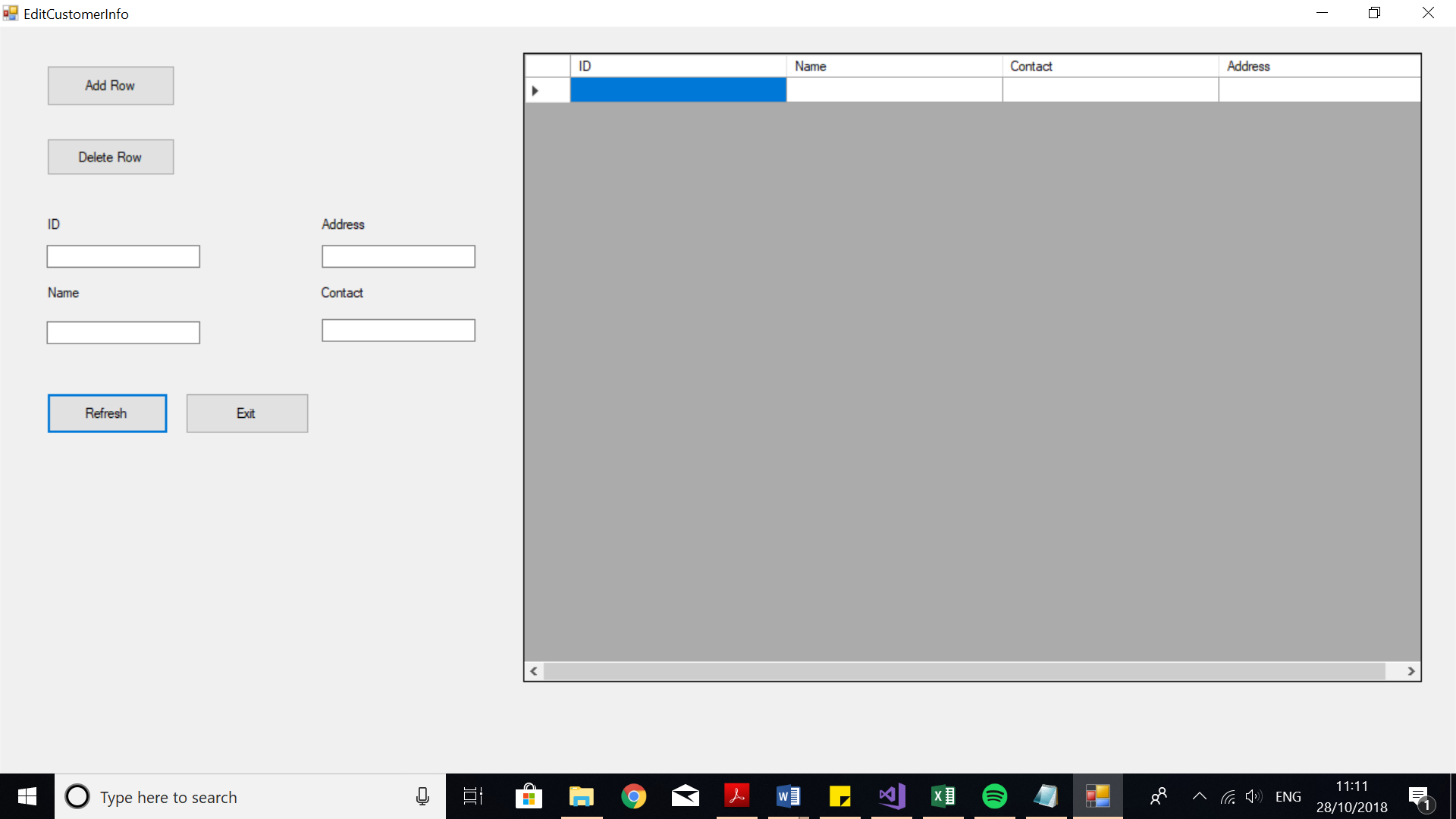
File.Close()

End Sub

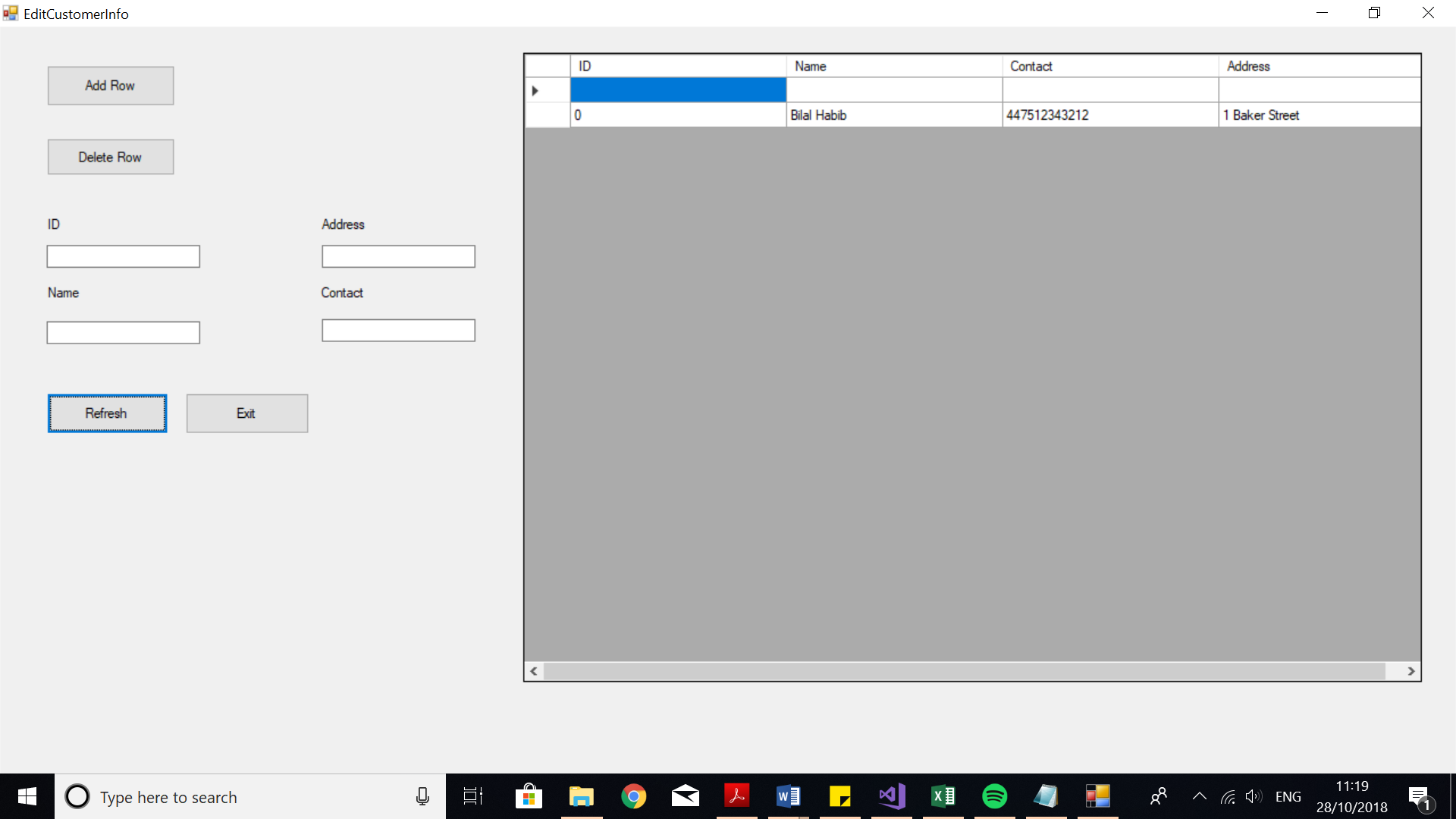
This line in the code needed a separator, so that they don’t get outputted into a single column in a row in the csv file. Hence why I have also used a Comma Separated Value file (csv file). This makes it easier to edit the data in every record which is what I will demonstrate in the version 2. I will also include validation in the later versions of the project.

I have only included four columns for the Customers’ Database as there is no use of more information, therefore it would be time consuming to add more details.

After the details are added to the database, in order to show those details in the data-grid-view in visual studio, I have created a Refresh button.



Taking the previous example of the add button, I am going to import that data from the csv file into the data-grid-view in visual studio.



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

DataGridView1.Rows.Clear()

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

Dim columns\_expected As Integer = 4

Dim reader As New StreamReader(Customerfilename)

StreamReader is used to read text form files

Dim sline As String = ""

Do

sline = reader.ReadLine

If sline Is Nothing Then Exit Do

Dim words() As String = sline.Split(",")

DataGridView1.Rows.Add("")

If words.Length = columns\_expected Then

For ix As Integer = 0 To 3

DataGridView1.Rows(DataGridView1.Rows.Count - 1).Cells(ix).Value = words(ix)

Next

End If

Loop

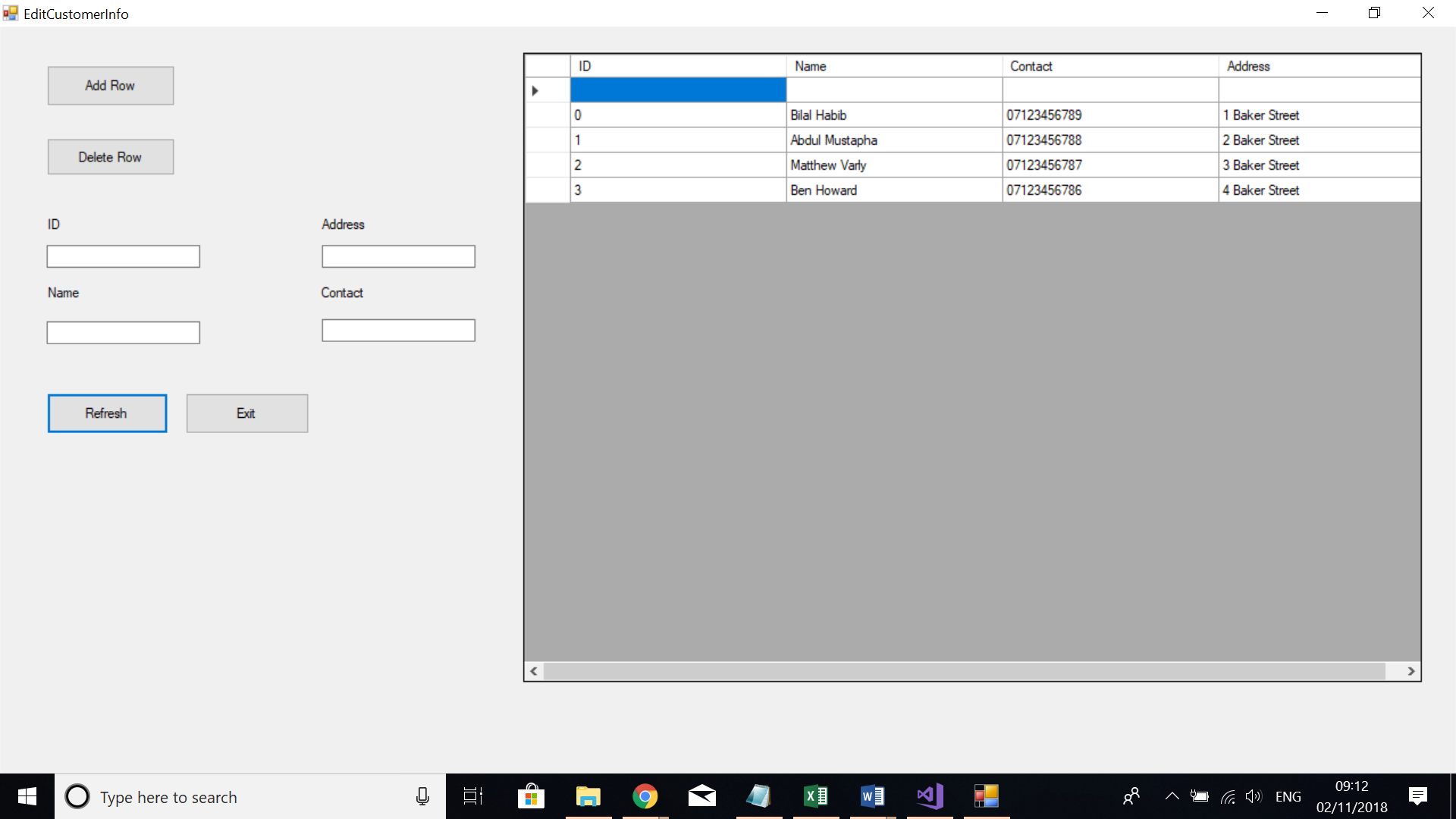
reader.Close()

'looping through the array and adding a comma between every value to easily locate it

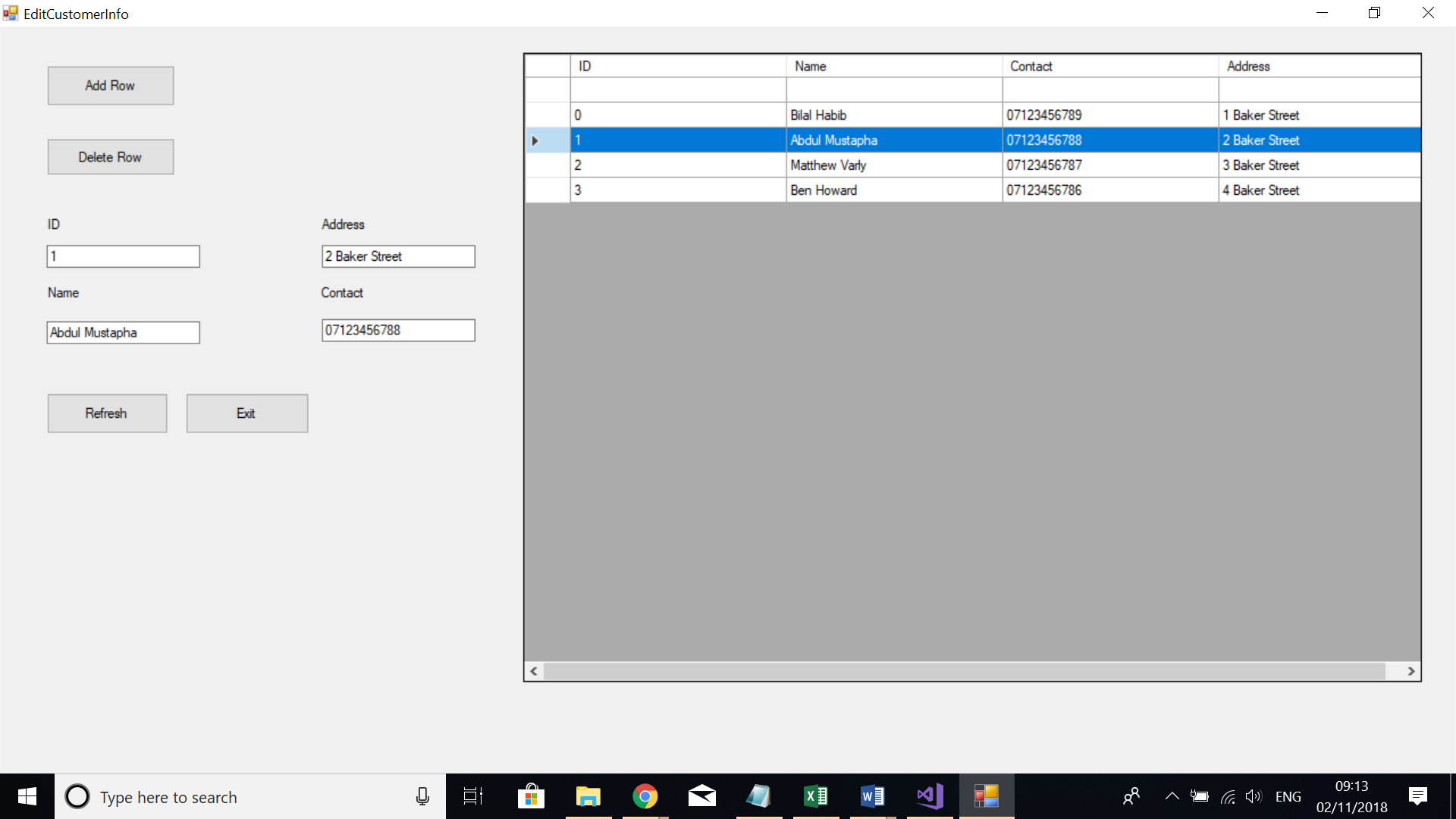
End Sub

In the code above, I needed to create sline which is a string variable in which to store each line of the file as I process it. Because I will be adding these lines to the list “words()”.

Another essential button that I have added is the Delete button. To delete a row the user must input the ID of the customer. I have only implemented my code so that it is the ID because, I have inserted another piece of code in which if the customer selects a row from the data-grid-view, all the details of the customer gets copied to the textboxes. This means that the ID will be filled in automatically and creates ease for the user’s experience.



If a row is selected, the data from the data-grid-view gets copied to the textboxes.



Private Sub CustomerDataGridView\_CellClick(sender As Object, e As DataGridViewCellEventArgs) Handles CustomerDataGridView.CellClick

' get the index of the selected datagridview row

index = e.RowIndex

Dim selectedRow As DataGridViewRow

' show data from the selected row to textboxes

selectedRow = CustomerDataGridView.Rows(index)

TextBoxID.Text = selectedRow.Cells(0).Value.ToString()

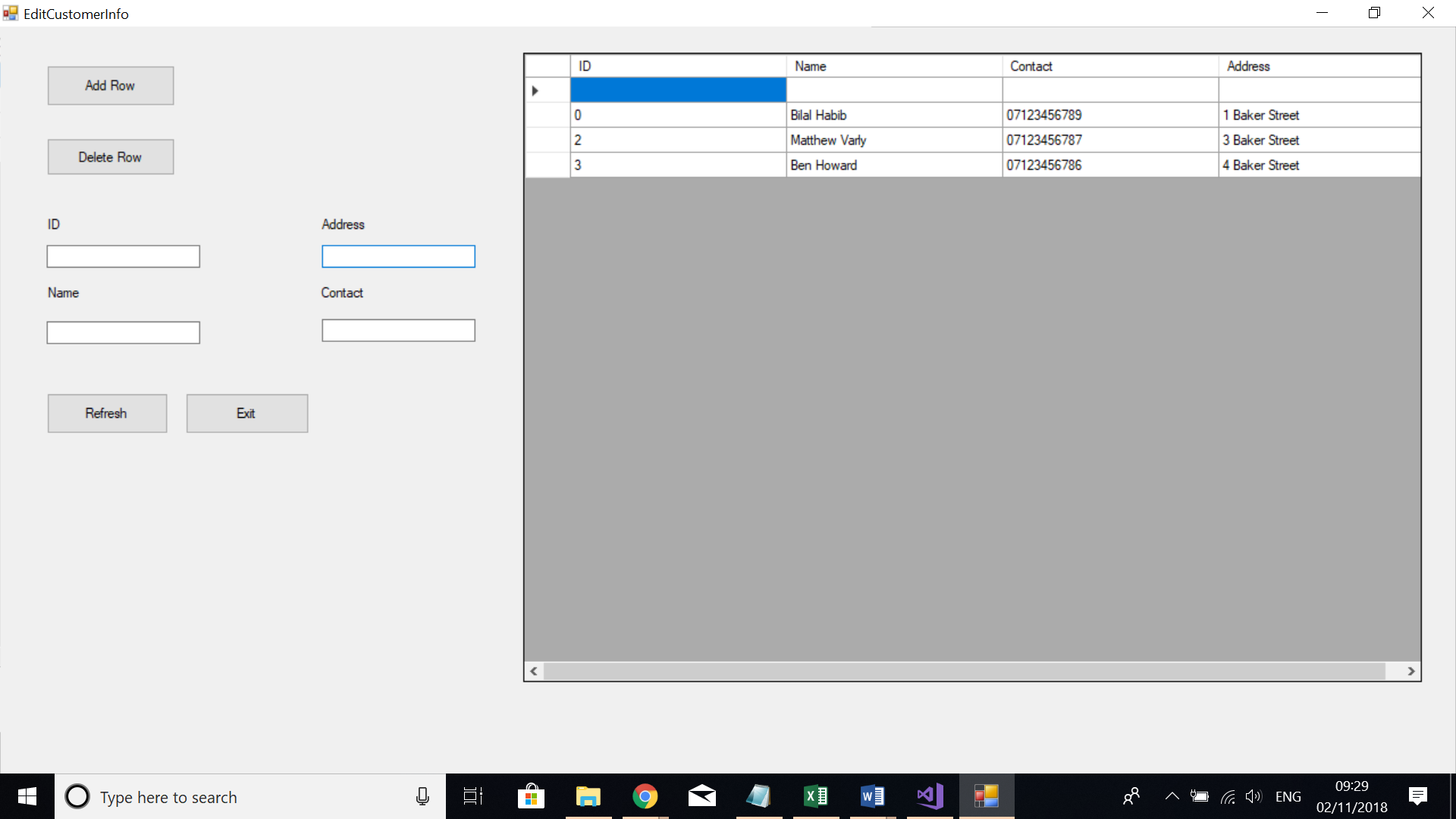
TextBoxName.Text = selectedRow.Cells(1).Value.ToString()

TextBoxContactNo.Text = selectedRow.Cells(2).Value.ToString()

TextBoxAddress.Text = selectedRow.Cells(3).Value.ToString()

End Sub

When the delete button gets clicked, the data in the database gets read by the StreamReader, then checks if the record contains the same data of what is in the ID textbox. It then writes every record that does not contain the ID into the CustomerList. (So it writes all the records other than the one that we want to delete). It then overwrites the file by adding what’s inside CustomerList into the Database.



In the form above, we deleted the record with ID 1. A problem that I want to resolve is re-ordering the ID’s of every record when they get deleted, so we don’t get random gaps.

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

Dim CustomerList As New List(Of String)()

Using Deletereader As New StreamReader(Customerfilename)

While Not Deletereader.EndOfStream

Dim line As String = Deletereader.ReadLine()

If line.Contains(TextBoxID.Text) Then

'Do nothing

Else

CustomerList.Add(line) ‘Adds all the records that we want

End If

End While

End Using

Using writer As New StreamWriter(Customerfilename, False)

For i = 0 To CustomerList.Count - 1

writer.Write(CustomerList(i) & vbCrLf)

Next

End Using

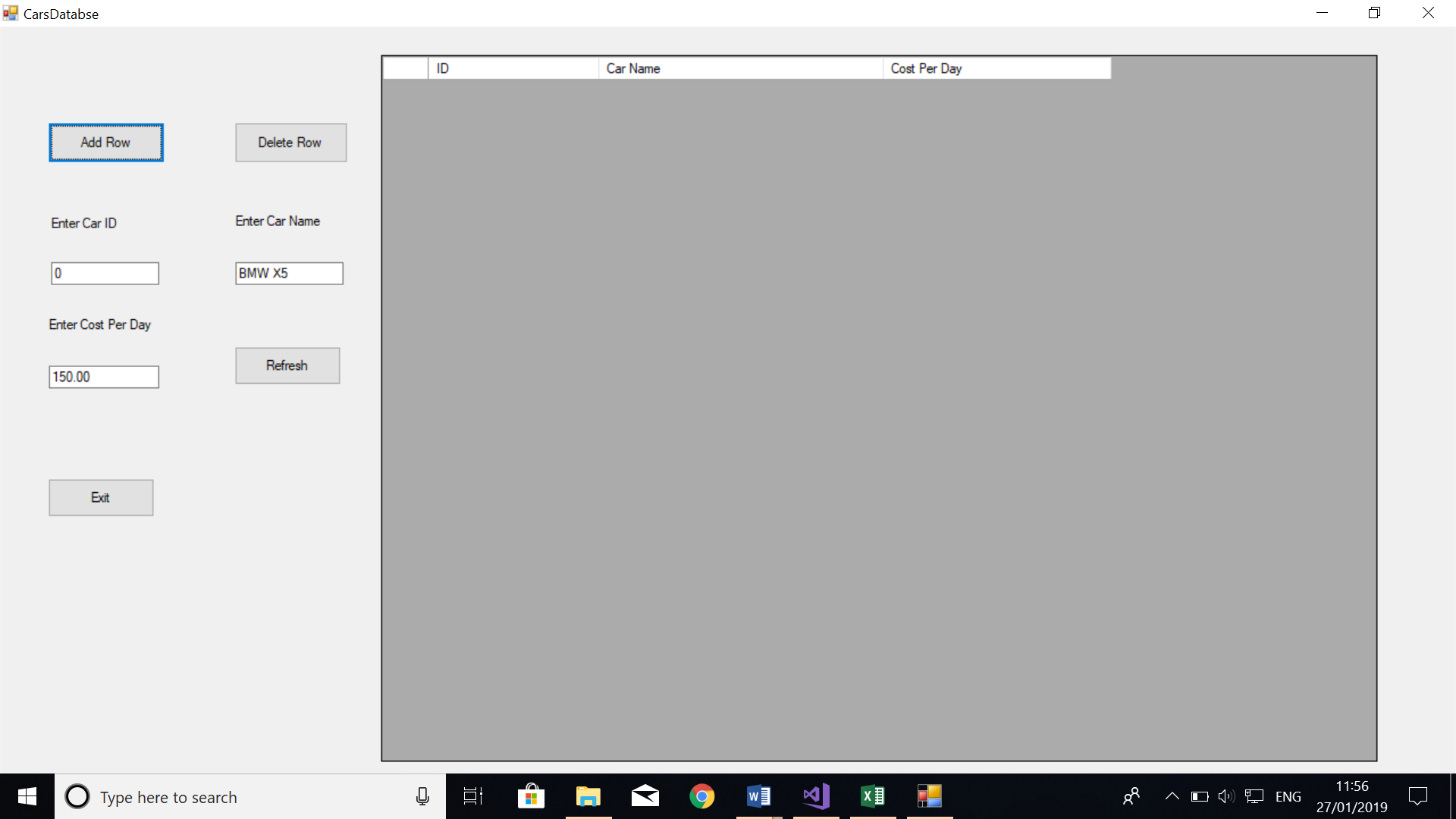
'Overwrites the current list

End Sub

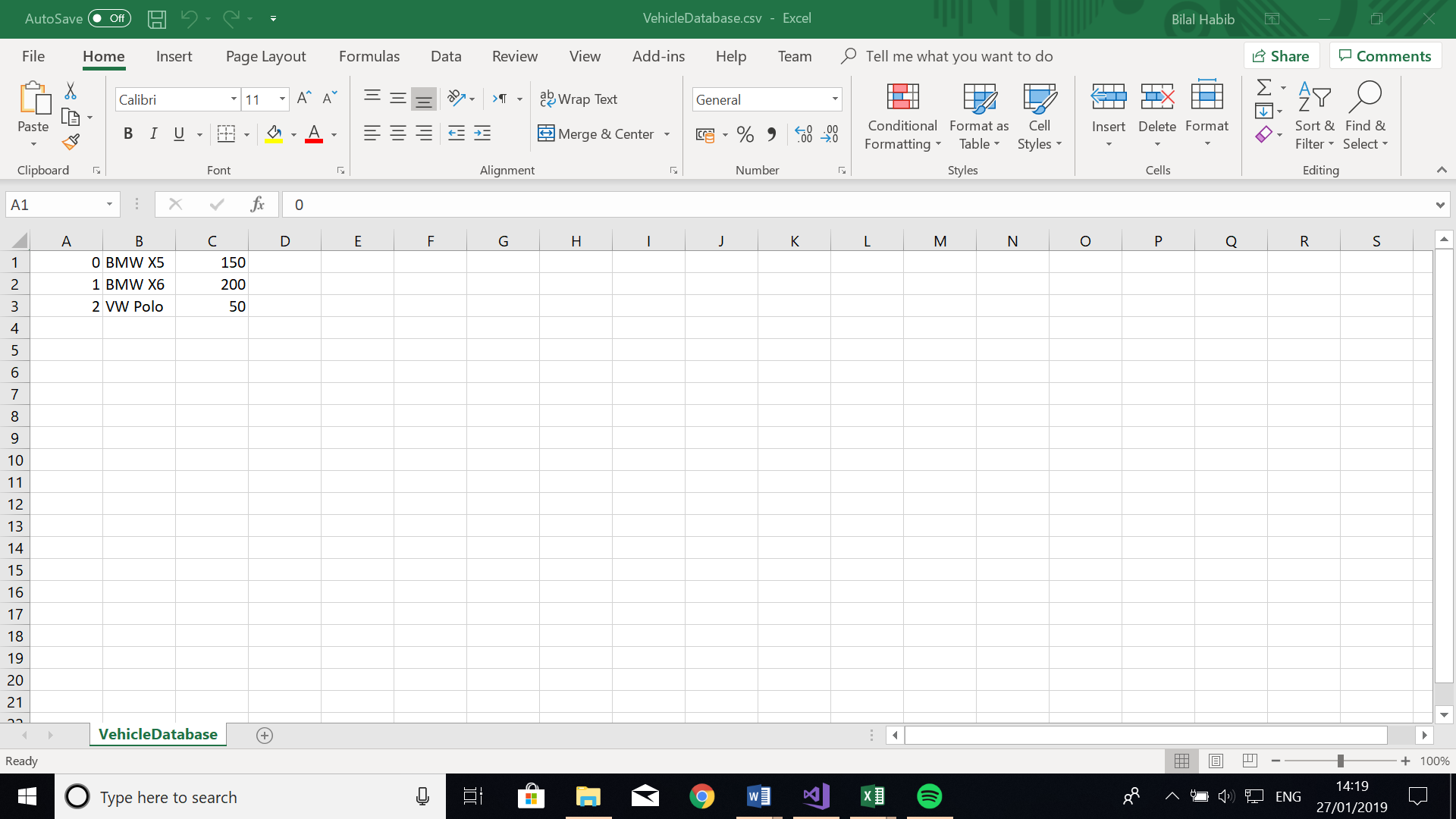
**Vehicle Database Form**

The Vehicle Database Form is very similar to the Customer Database Form in terms of the design of the form and the code that it contains. Both forms do exactly the same thing, the only difference is that the data that is added to csv file is different. (E.g Vehicle name and Cost per day)

For that reason, I am not going to demonstrate the code used as there are only minor differences such as the Structure name and the variables in the structure. But, I will show proof of the buttons working in the Vehicle Database Form.

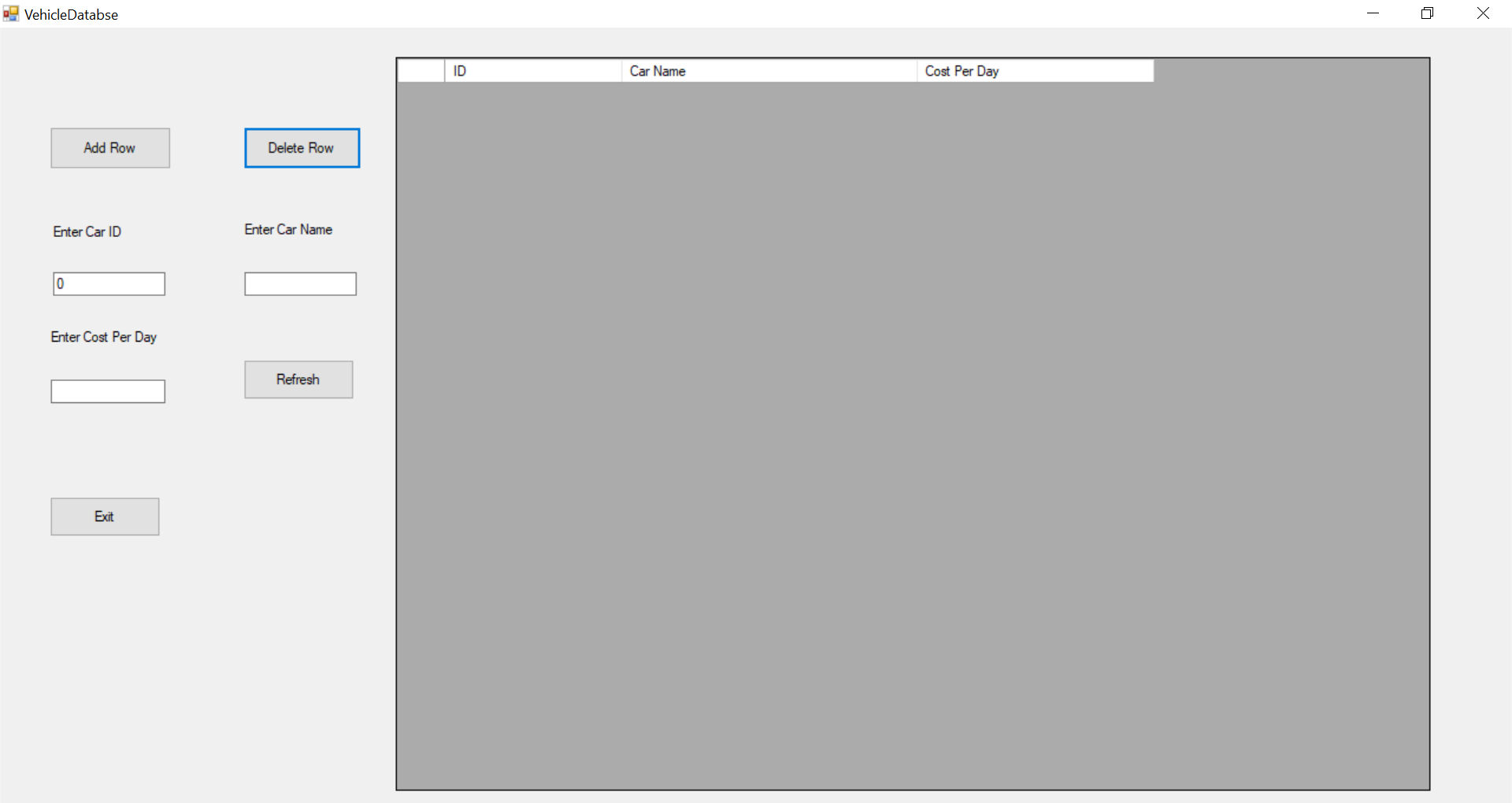


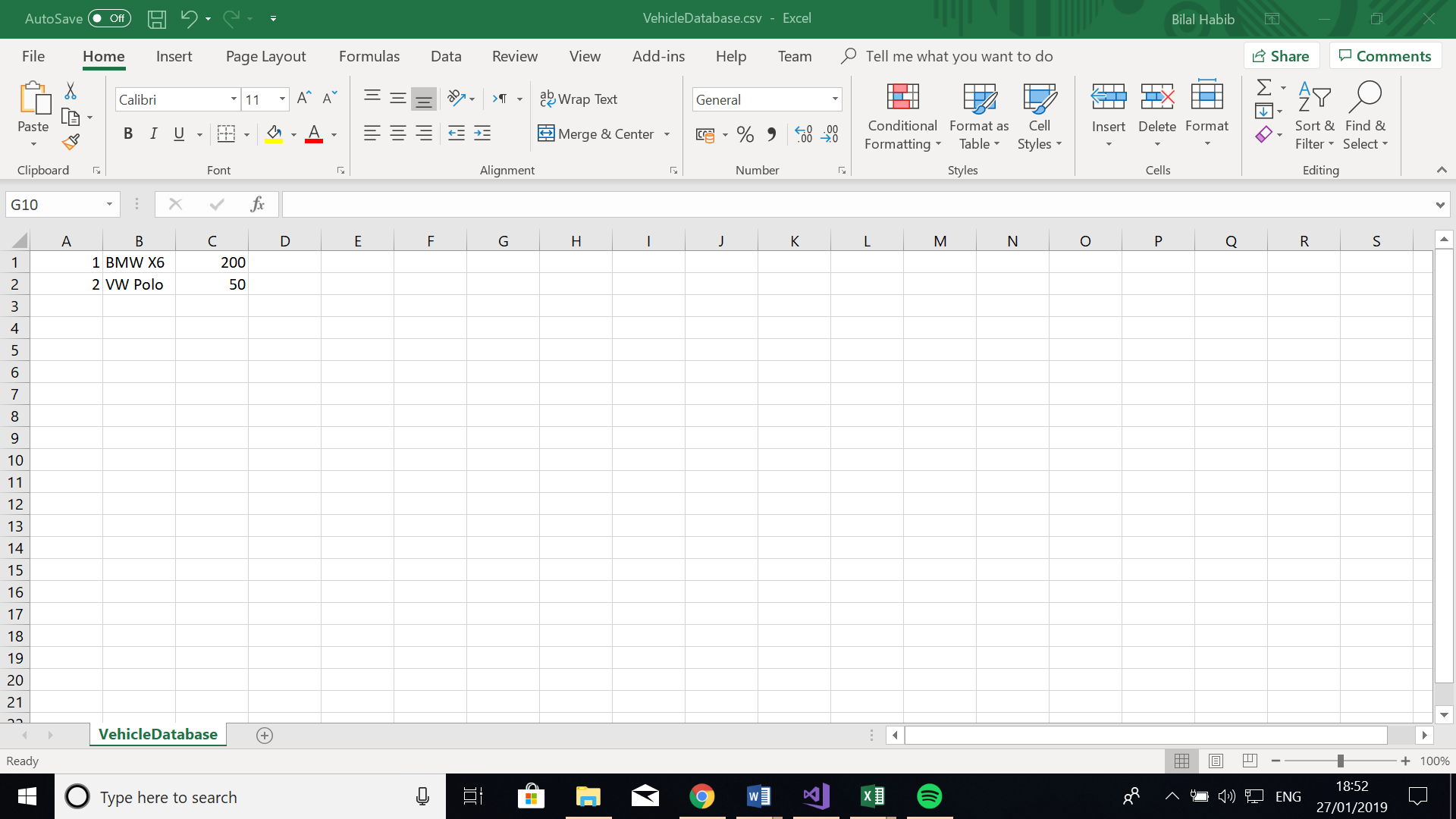
After entering the details of the vehicle into the textboxes and pressing the Add button, those details should be added to the csv file as a single record. (I added 2 more records to make sure it was working properly)



The details were added successfully as single records.

I am now going to test if the Delete Button works successfully.





The record added previously was successfully removed from the csv file by entering the ID number into the ID Textbox followed by clicking on the Delete Button.

One final test I need to make is to see if the Refresh Button works which will import the records from the csv file into the Vehicle Data-Gird-View.

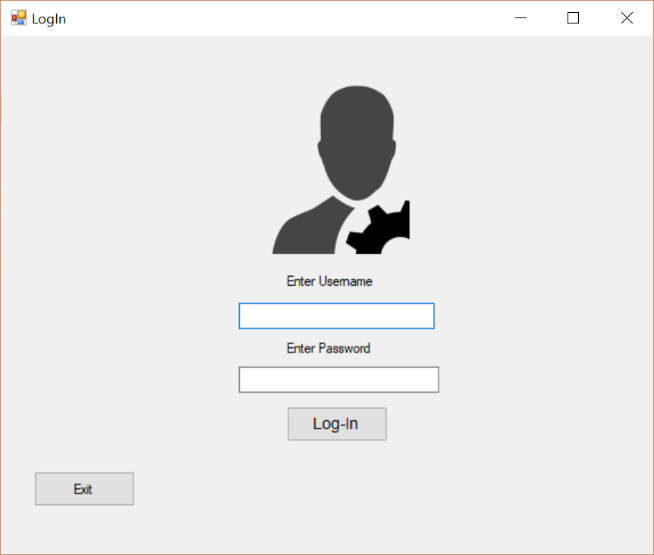
**Version 1 Testing:**

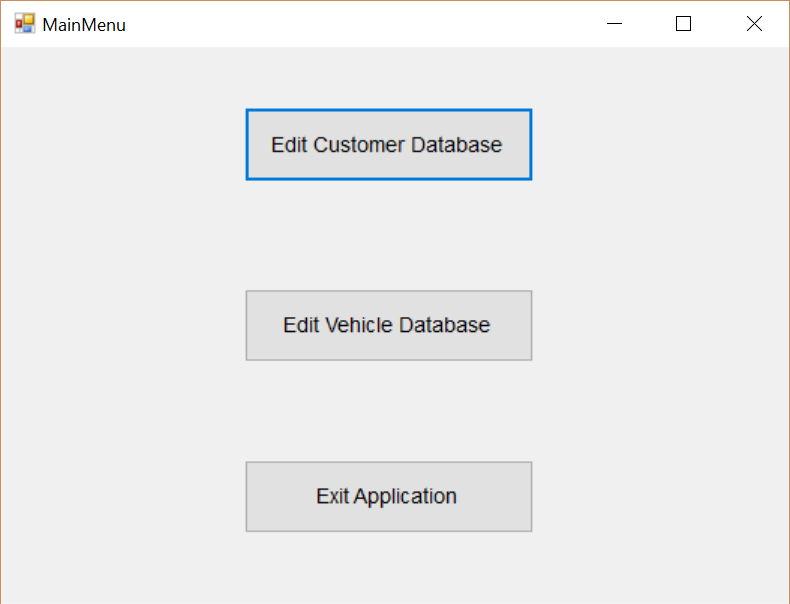
In order for the user to find the usability of the program enjoyable, I need to test all aspects of my program to see if they are full functioning. Due to this only being the first version of my project, not all forms have been created, deciding that it is not suitable to allow stakeholders to test the first version of my project as it will have minimal effect.

**Black Box Testing**

One of the main aspects of my program is navigation, this feature is necessary in order for the user to be able to access all parts of my program with ease. Therefore, this makes navigation an important usability feature as it will save time and overall make the use of the program a more enjoyable experience.

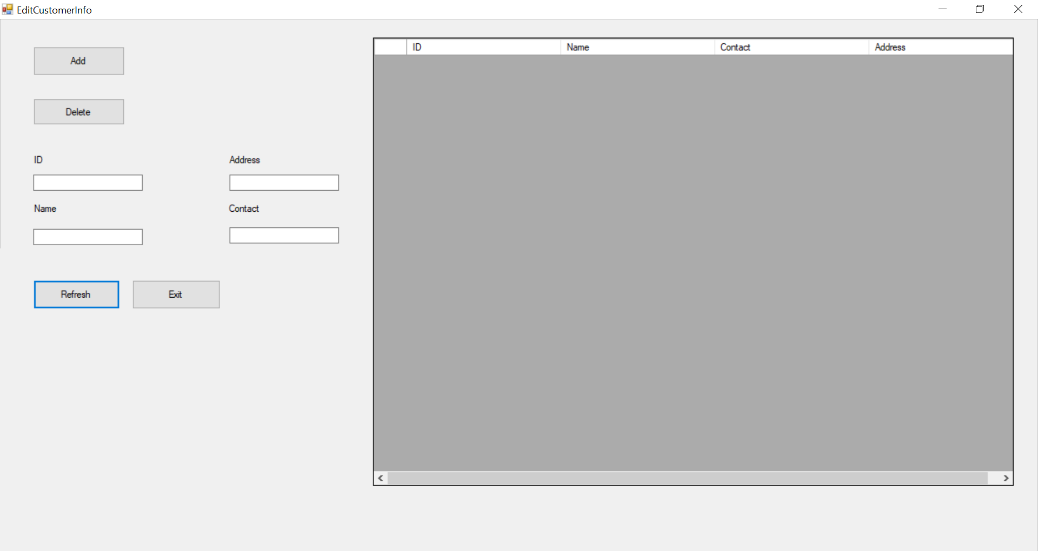
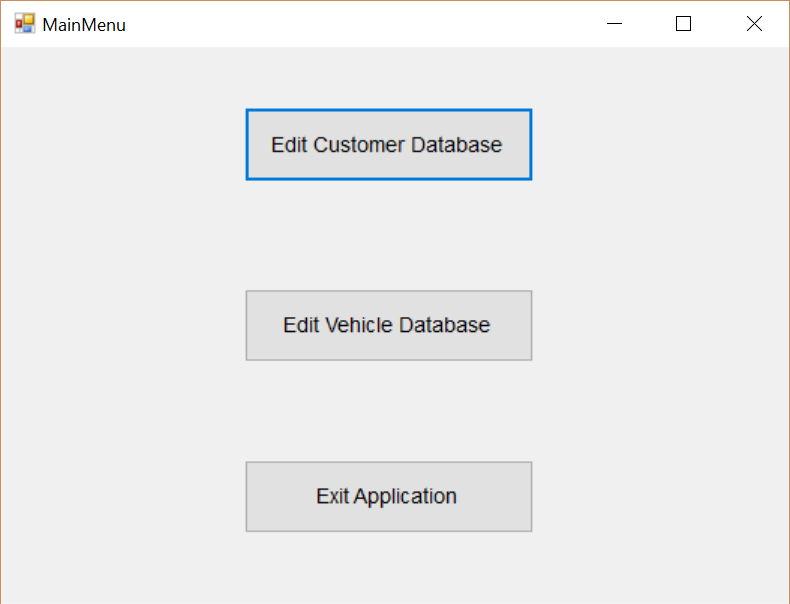
1. Login Menu to Main Menu:

If the user enters a valid username and password, they will be taken to the main menu

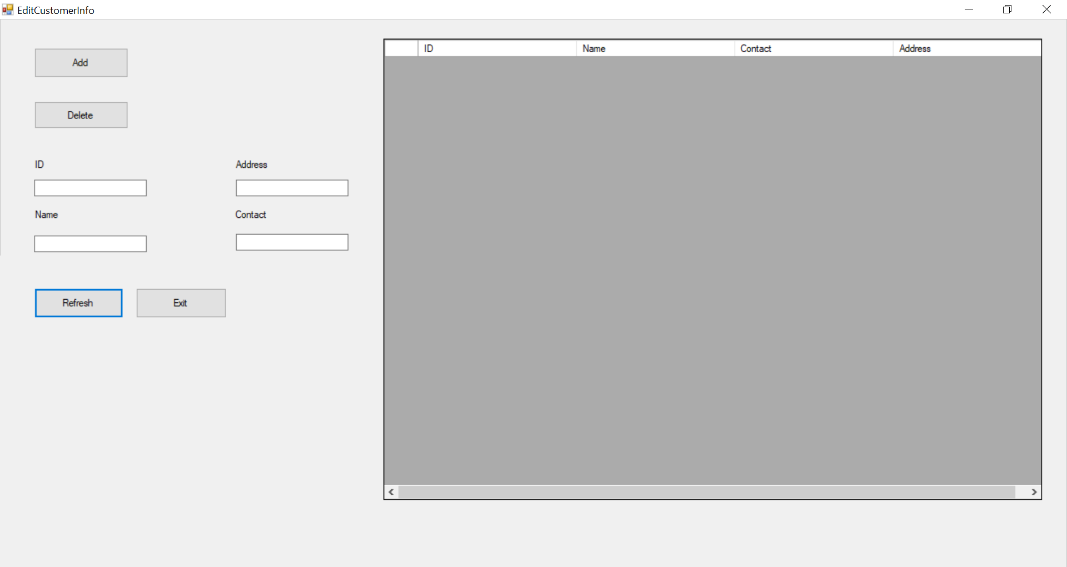
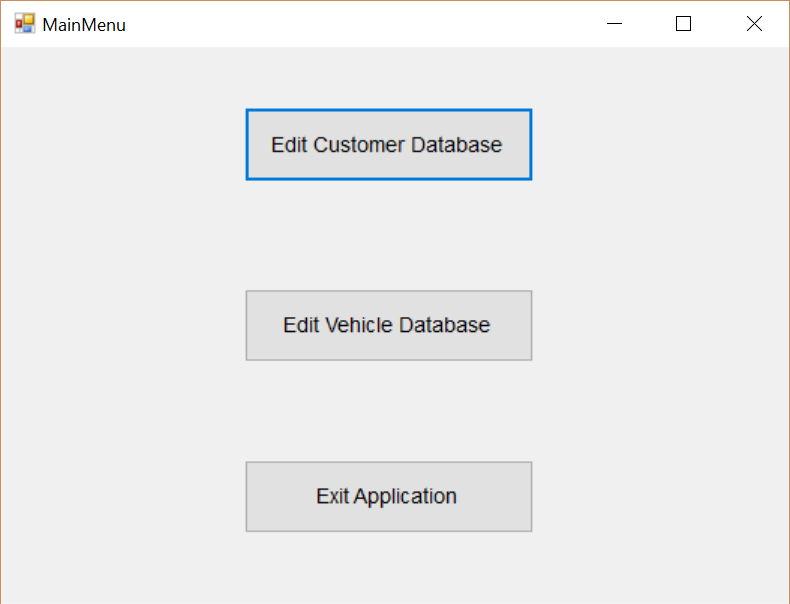


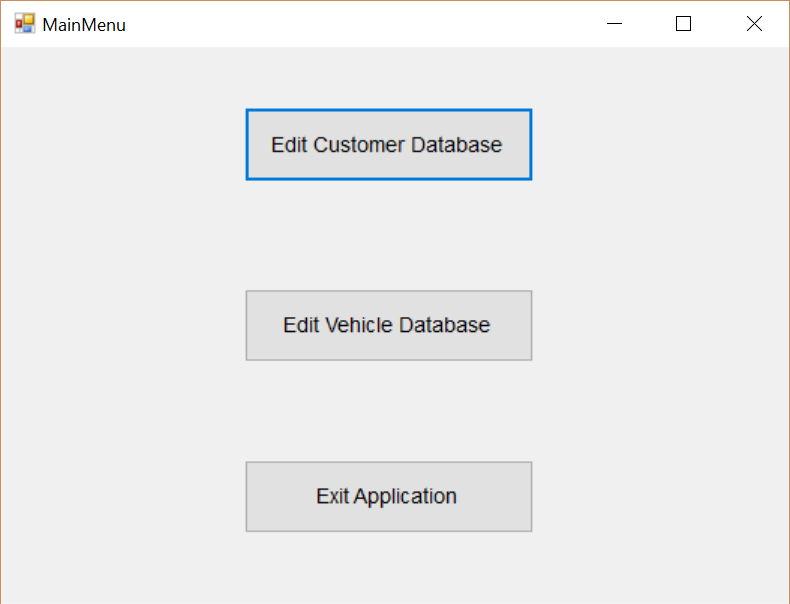
1. Main Menu to Customer Database Form

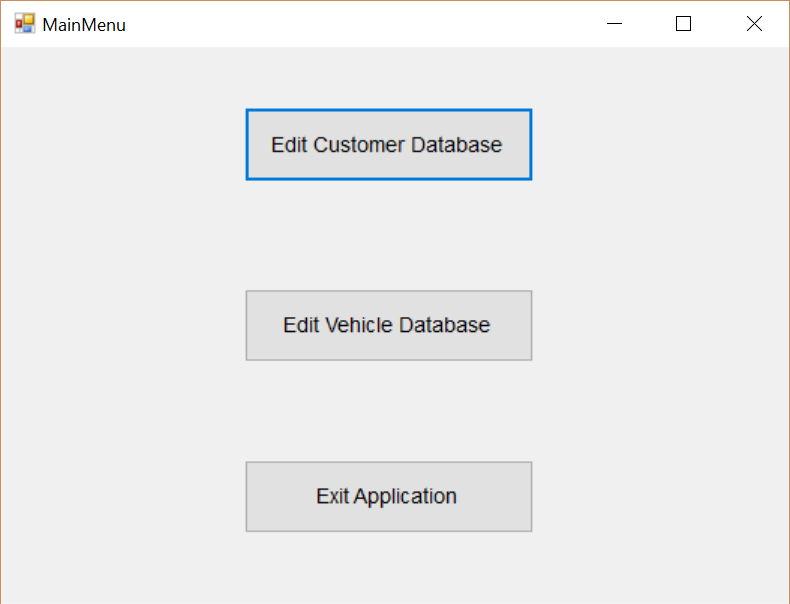
If the user clicks on the “Edit Customer Database” button



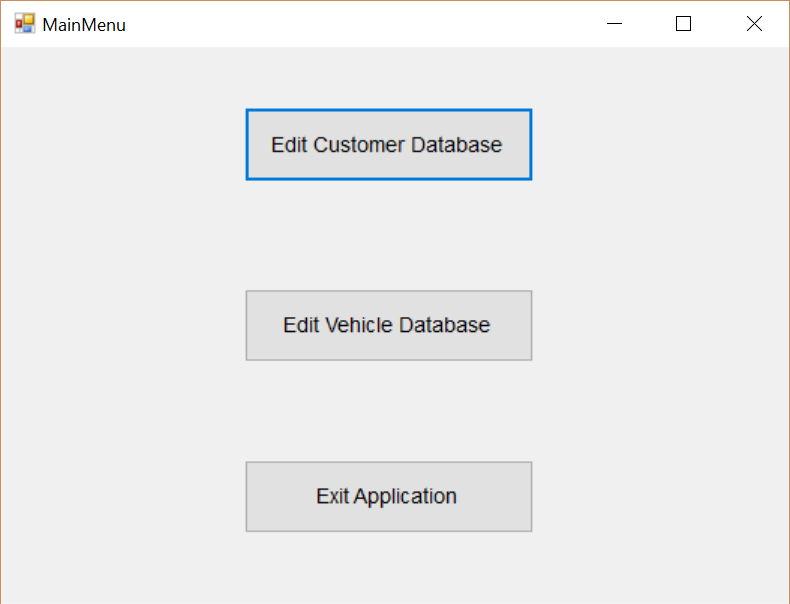
1. Customer Database Form back to Main Menu

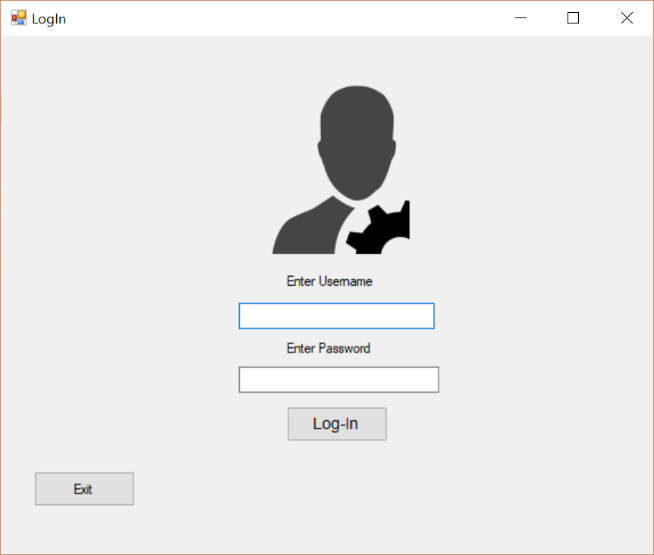
When the user clicks on “Exit”, they should return back to the Main Menu

1. Main Menu to Vehicle Database
2. Vehicle Database back to Main Menu

When the user clicks on “Exit”, they should return back to the Main Menu

1. Exiting the application

When the user clicks on exit in any of these two forms, the application closes and this is the expected result. Although, to make this program more usable, I should implement a feature where if the user clicks on the Exit button, they should be given an option that says “Are you sure you want to exit the program?”



**White Box Testing**

**Log-In Form**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Number | What Testing | Input | Expected Result | Actual Result | Actions Needed |
| 1 | Adding a Record | Username: “Admin”  Password: “AdminPass” | User is told they are logged in and taken to the main menu | User is told they are logged in and taken to the main menu | None |
| 2 | Adding a record | Username: “Admin” Password: “adminpaass” | User is told their username and password do not match | User is told their username and password do not match | None |
| 3 | Getting login details wrong and then entering them correctly | First Entry  Username: “aDmin”  Password: “adpass”  Second Entry:  Username: “Admin”  Password: “AdminPass” | User is told that their attempt to login was unsuccessful for first entry but when they enter it correctly the second time, they are told they have logged in successfully and taken to the main menu | User is told that their attempt to login was unsuccessful for first entry but when they enter it correctly the second time, they are told they have logged in successfully and taken to the main menu | None |

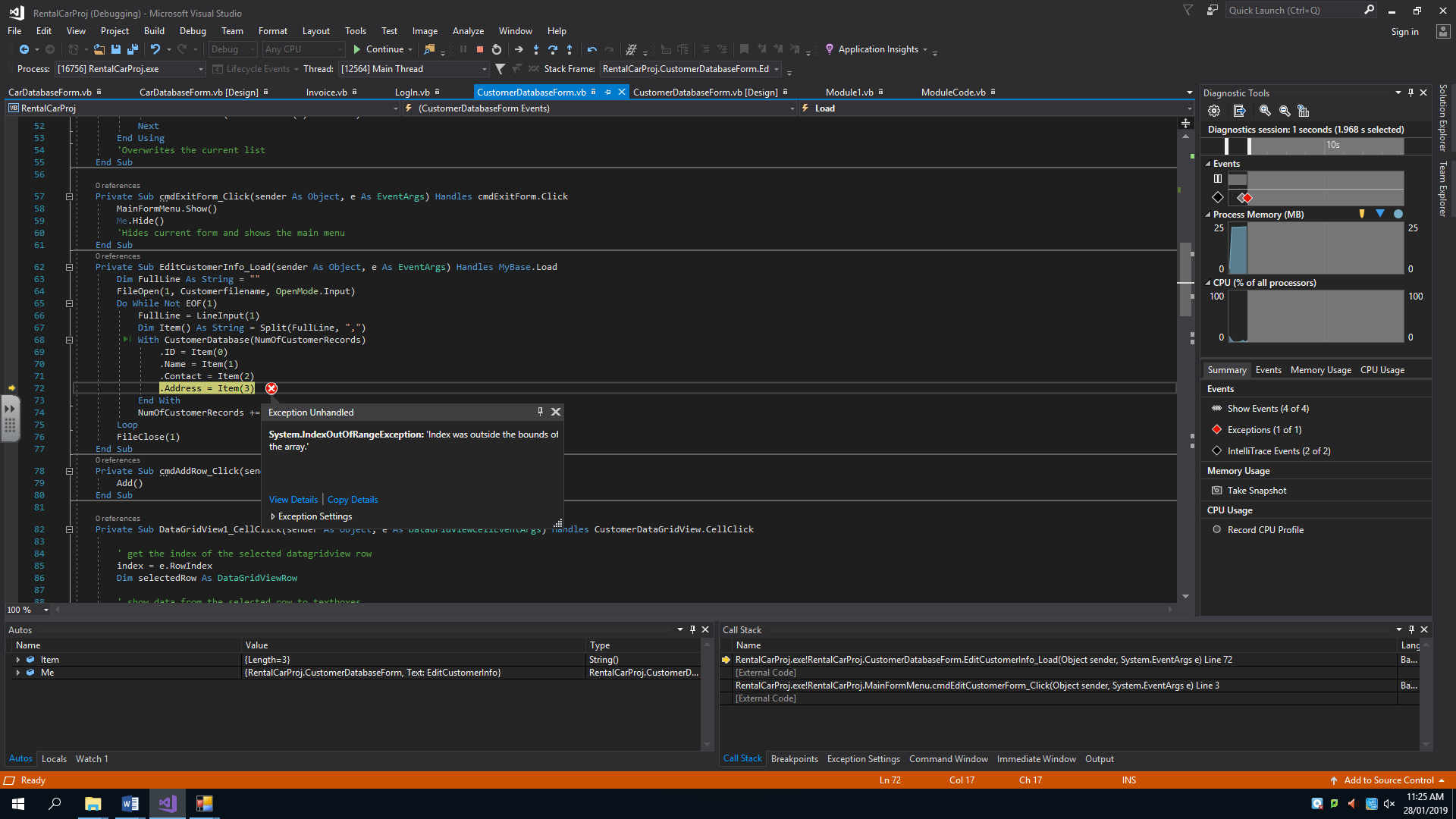
**Customer Database Form**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Number | What Testing | Input | Expected Result | Actual Result | Actions Needed |
| 4 | Adding a record | **ID:** 0  **Name**: Bilal Habib  **Contact:** 07123456789  **Address:** 1 Baker Street | The record (customer) gets added to the csv file | The record (customer) gets added to the csv file | None |
| 8 | Deleting a record | **ID:** 0 | The record in the csv file with ID number 0 should get removed from the file | The record in the csv file with ID number 0 should get removed from the file | None |
| 16 | Refreshing the data-grid-view | **none** | The data-grid-view should be filled with records from the csv file | The data-grid-view should be filled with records from the csv file | Fix the problem where the first record |
| 17 | Exit Button | **none** | The program should close the customer database form and open up the main menu | The program should close the customer database form and open up the main menu |  |

**Error Evidence and Resolution**

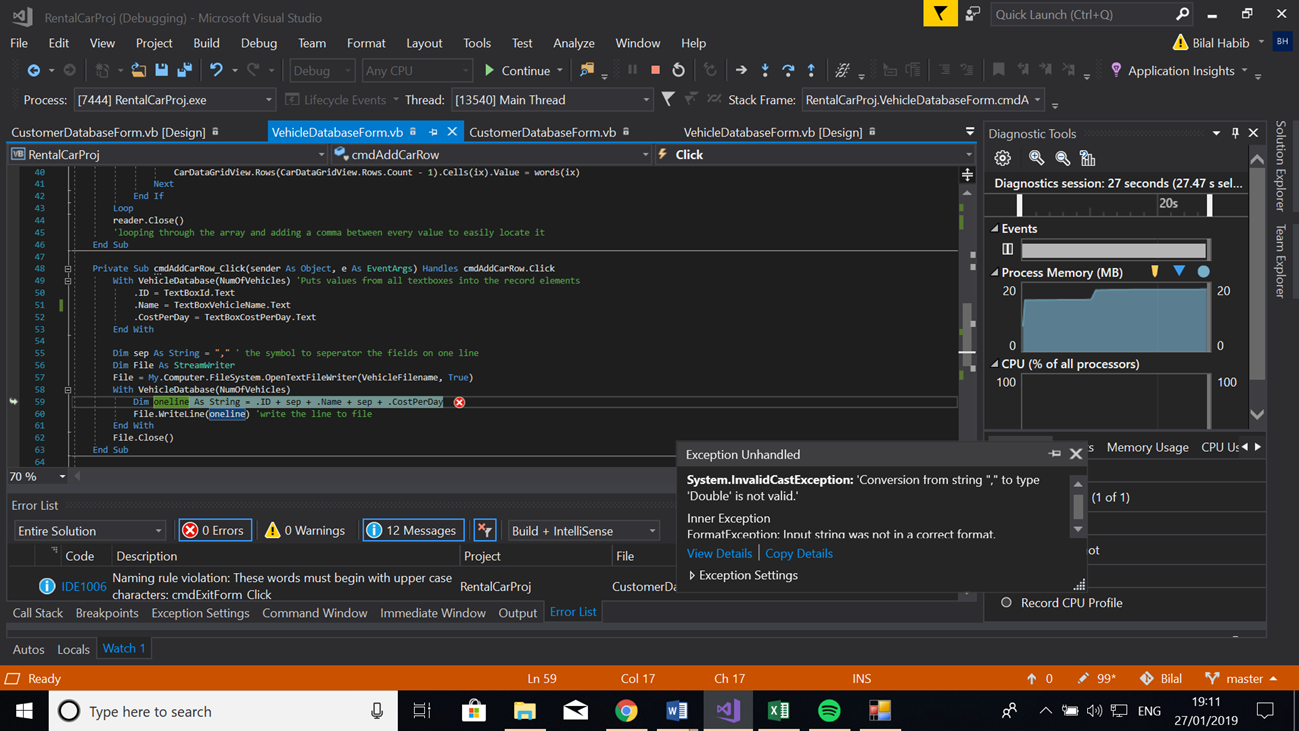
I encountered a few errors during the development of this version. I have decided to resolve them within this version as I have been able to resolve the errors and will therefore be concentrating on different segments of the project in the next version.

**Customer Database Load Errors:**



This error was caused by missing data in a record, a customer record has to hold 4 pieces of data but in this case, there was a record that only had 3 and this was identified by looking at the “Value” of the array “Item”. The array “Item” was supposed to have a length equal to 4 but it held 3 due to the missing value and therefore had an error when it came to reading the data from the csv file.

In order to prevent this, I will need to implement a form of validation where if a user is trying to add a record into the database, none of the textboxes can be blank and therefore force the user to input 4 pieces of data.

**Add Button Errors:**

This was an error caused by the data entered not being a string. “.ID” and “.CostPerDay” were defined as an integer and a double, “.Name” was the only String.

Oneline is a String therefore everything in Oneline must be a String.

I prevented this error by converting .ID and .CostPerDay into Strings by using the “.ToString” extension.

Code: Dim oneline As String = .ID.ToString + sep + .Name + sep + .CostPerDay.ToString

**Delete Button Errors:**

A problem with the Delete Button that I had was that if a record was empty and the user deleted the record, every record in the csv file would’ve got deleted as well and this is a massive issue as it would contribute to a huge loss of data.

To prevent this I must implement a form of validation where if any textbox is empty, the Delete Button should not perform anything. Although, none of the records should be empty due to a validation check that I need to implement where if any textbox is empty that record should not be added. I mentioned this validation on page “”.