TESTING OF A BAKERY DESKTOP APPLICAION

Abstract

In this document we will be discussing the desktop app that was built for tasty delights and we will be running comprehensive tests on it.

Akhil Montrose (81788)

Akhil.montrose788@we.utt.edu.tt

Table of Contents

Introduction

Purpose

This document will be dealing with a desktop app that was developed for a company named Tasty Delights. It is a bakery who at the time did not have an application of the sort for doing their normal day-to-day transactions. The goal of this document is to:

- Identify what information that we have on this application that can be tested.
- We will be listing the tests that are to be carried out on the system.
- Recommend the types of testing that will be employed on the system.
- Identify where there may be errors and give a feasible solutions for the problems.

Scope

This test plan will be looking at how the app was integrated and how well it will function for the company.

When the app was being developed it is safe to mention that it will be possible to run black/white box testing as well as testing for other various errors/problems such as GUI, etc (which will be discussed in the further chapters.)

The interfaces that will be tested/documented in this project are:

- Home page
- Customer interface (Adding, Viewing, Searching, Deleting)
- Pastry interface (Adding, Viewing, Updating, Searching, Deleting)
- Order interface (Adding, Viewing, Searching, Deleting)
- Employee interface (Adding, Viewing, Updating, Searching, Deleting)

This application for specifically built for the use of the employees to make record of the interaction/transactions of the system. So the tests would be done on remote PCs at the company.

When testing the performance, we will be focusing on:

- The response time of the application when opening up.
- The response time of the application while performing transactions.

Test Plan

Testing that will be performed

In the previous chapter we discussed what are the interfaces that are going to be tested. In this chapter we will discussing the type of tests that will be done on those said interfaces.

- Security and Access Control Testing.
- Function Testing (Does the system meet the requirements that were needed by the company?)
- User Interface Testing (checking the ease of navigation as well as visualizations of the system.)
- Data and Integrity Testing (How data is entered and if there are any problems occurring)
- Blackbox testing (in other words, if the end user will be receiving the expected outputs from the system.)
- Whitebox testing (This will be highlighting the internal workings of the application)
- Performance Testing (checking to see how long does it take to perform actions such as login, add/search/delete/update/retrieve information.)

Testing that will not be performed:

These are the tests that will not be performed on the web application:

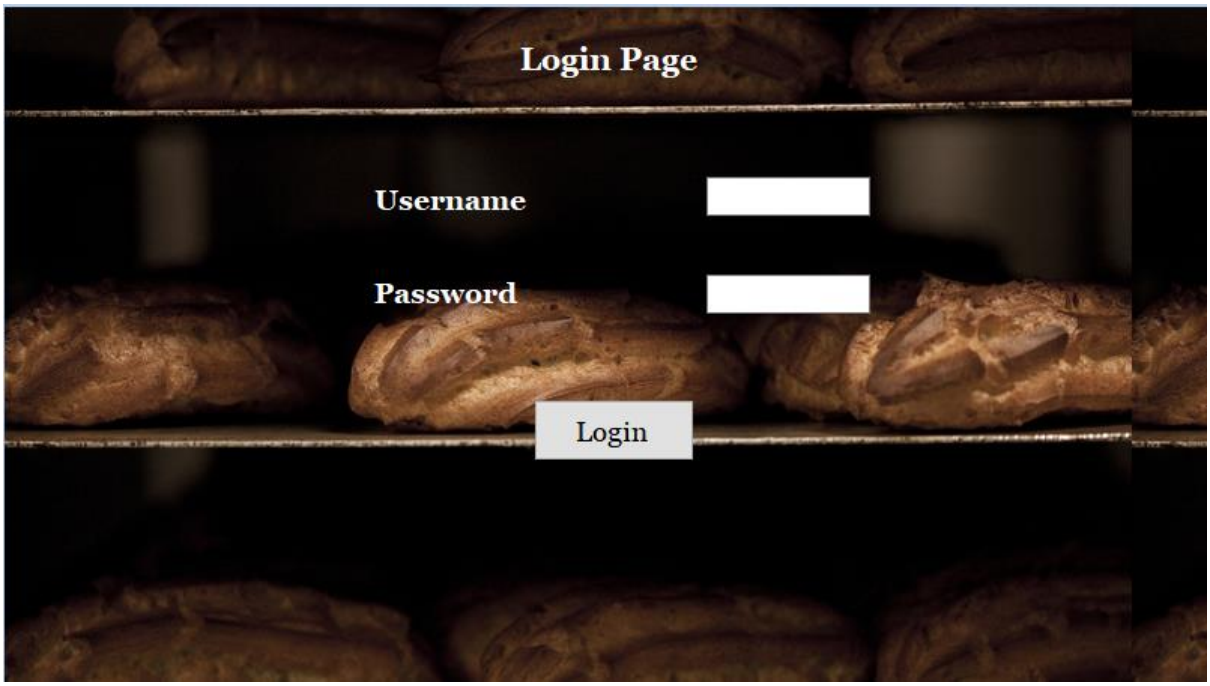
- Business Cycle Testing
- Load Testing
- Stress Testing
- Volume Testing
- Failover / Recovery Testing
- Installation Testing
- Automation Testing

Testing Phase

Security and Access Control Testing

Aim:

Testing to see if there are features that allow certain users to gain access to the system (logging into the system.



Login Page

The current page only accepts a hard coded username and password. It does not connect directly to a database. Below is the code that highlights how a username and password was hardcoded into the system

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load

    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        If TextBox1.Text = "monty" And TextBox2.Text = "password" Then
            Home.Show()
        Else MsgBox("Sorry, Incorrect Username or Password!", MsgBoxStyle.OkOnly, "Invalid")
        End If
    End Sub

    Private Sub Label1_Click(sender As Object, e As EventArgs) Handles Label1.Click

    End Sub

    Private Sub Label3_Click(sender As Object, e As EventArgs) Handles Label3.Click

    End Sub
End Class
```

Code For Login Page

Problem:

The problem that can arise:

- If someone gains access to the passwords, they will get full access to the entire system.
- If there are multiple PCs, then the username and password would have to be hardcoded.

Solution:

What can be done to alleviate the solution is to create a database that can hold the credentials of the employees using the system.

Problem 2: Employee Section on home screen.

A normal employee should not be able to have access to anything from the Employee Class. An employee can go and make unnecessary changes to any existing users in the system.

Solution:

What can be done to mitigate that problem is to make a different login and home screen for a manager so that they will have ultimate access to all the classes. Also for normal employees, the employee section should be removed.

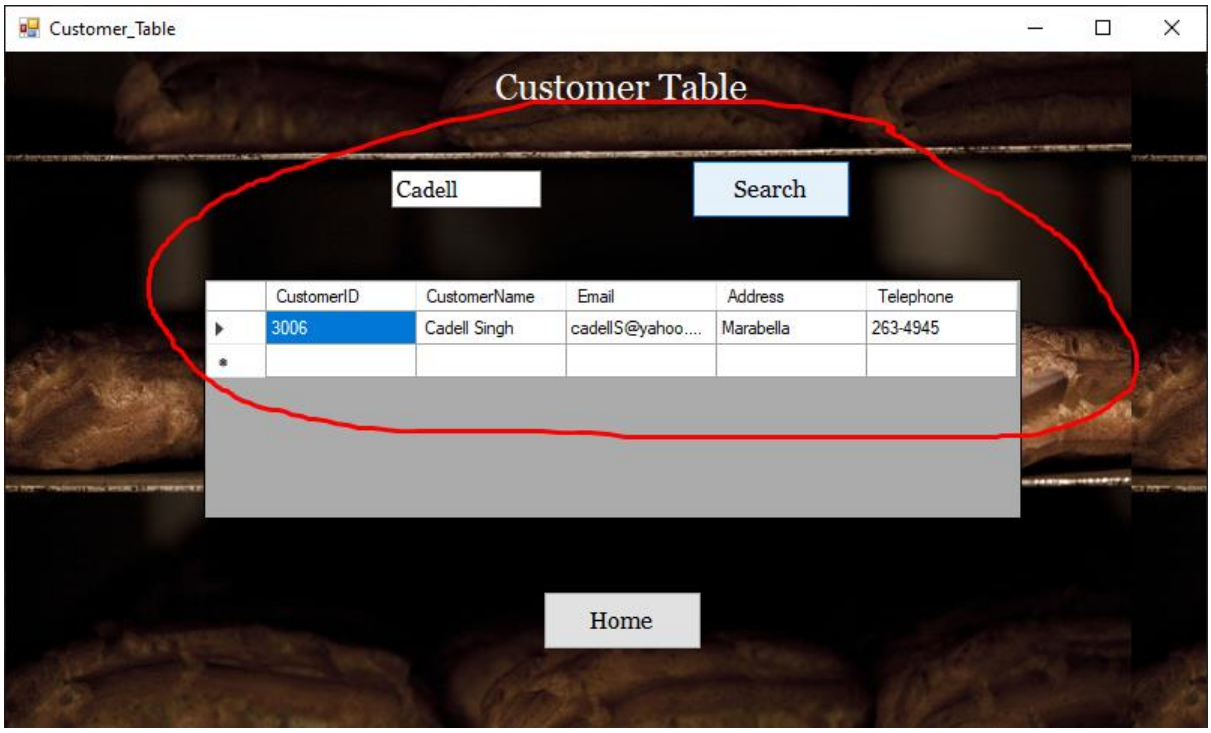
Function Testing

In this type of testing we will be checking to see if the system fulfils all of the requirements that the company, Tasty Delights wanted. The following functionalities include:

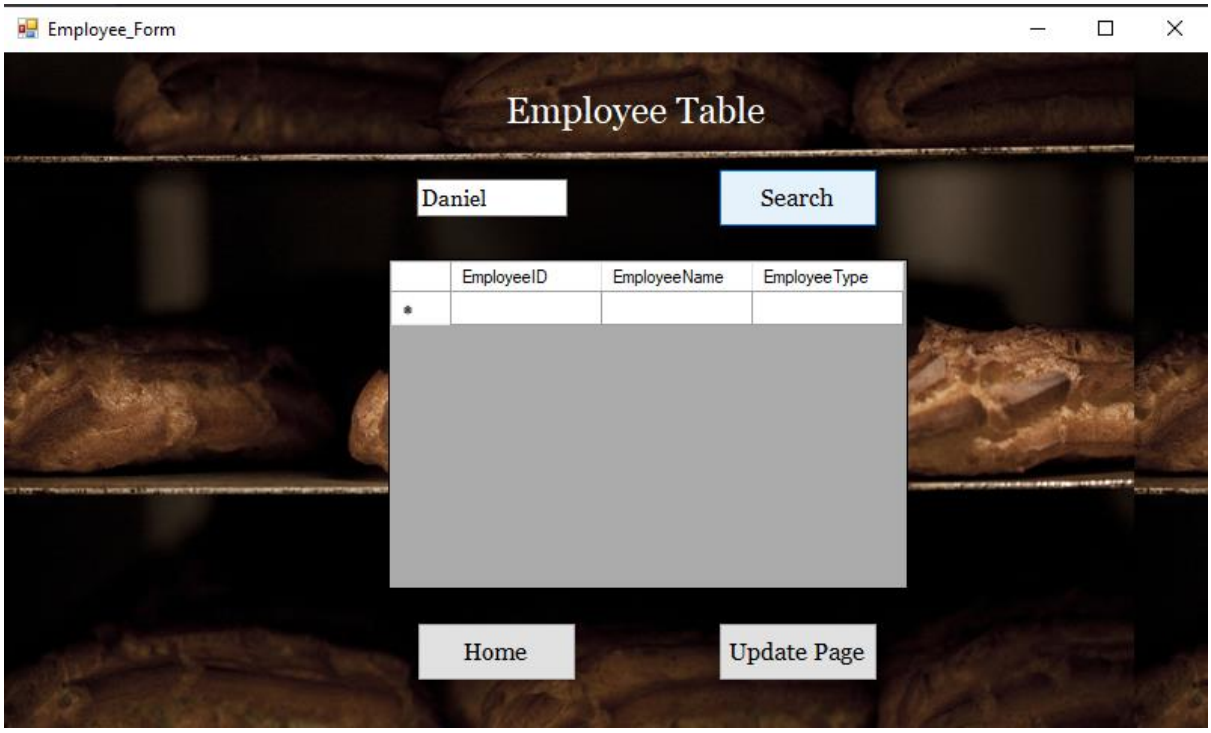
- **Search** the website to see what they would like to order.
- **Update** data on specific tables.
- **(Create)** make an order for what they want to purchase. The Employee will also have the ability to add customers, pastries, employees as well as create an order if they wish.
- **Retrieve** data from the Customer, Employee, Pastry and Order tables.
- **(Delete)** remove Customer, Employees, Orders and Pastries from the system.

Search:

The search feature is present in all of the entities in the web application. This test will be performed on all tables to see if the search feature is working as it should. Here are snippets showing search features of Customers and Employee:



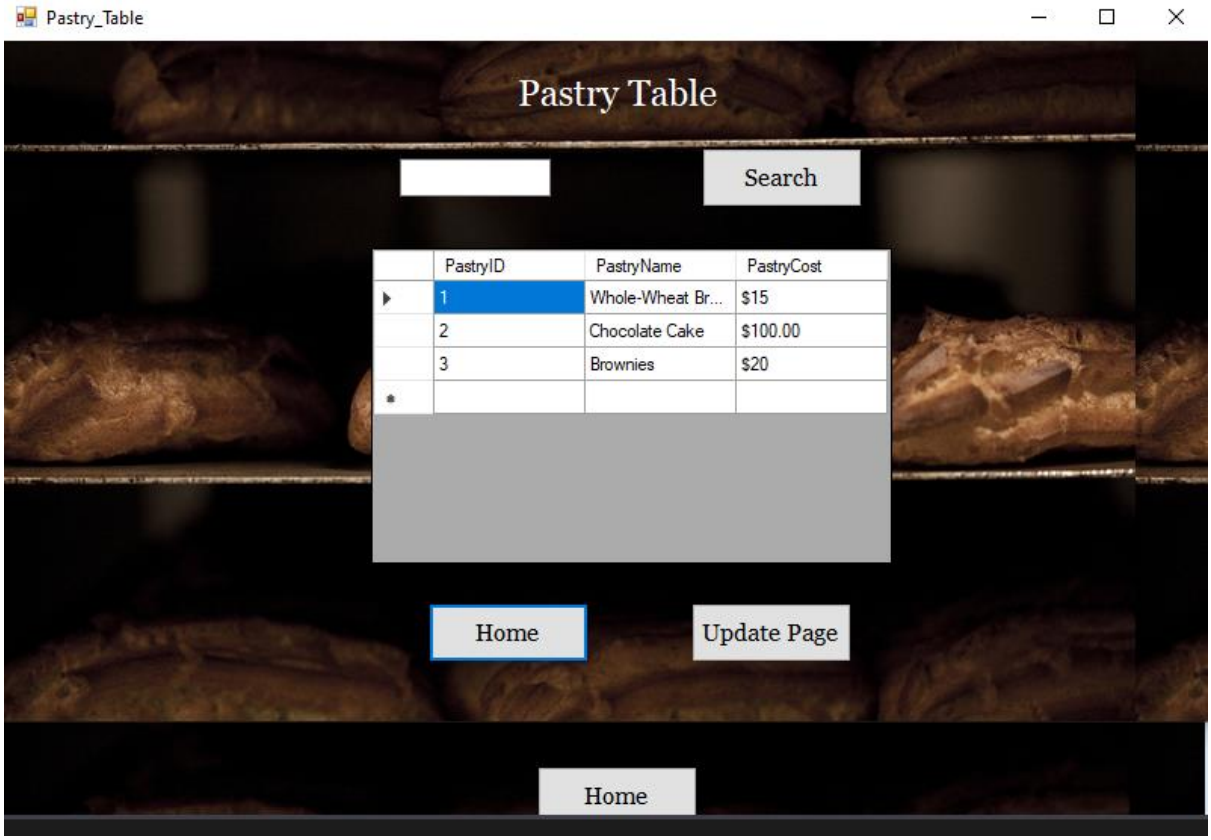
Customer Table



Employee Table

For the search functionality, it was utilized across all classes (Employee, Customer, Pastry and Order). The same type of functionality was used throughout each class so one example was given to show that it works. If it didn't work at any point of the system, the document would take note of that.

Retrieve:

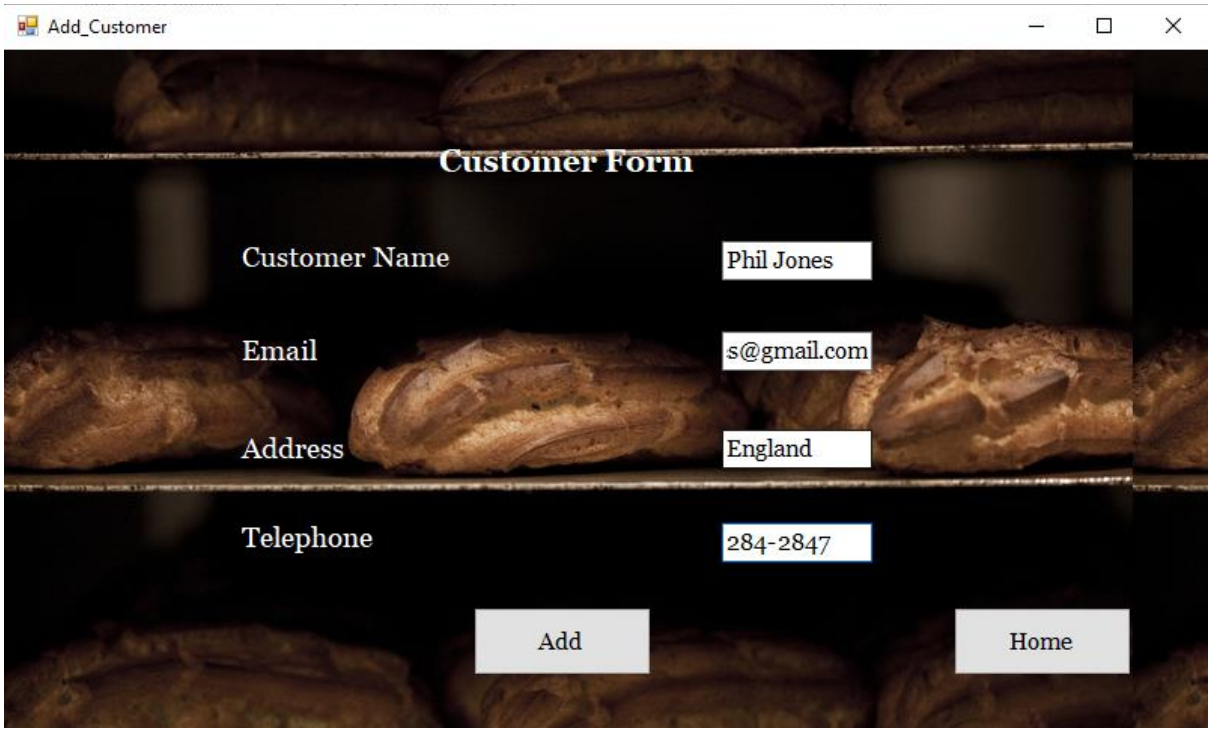


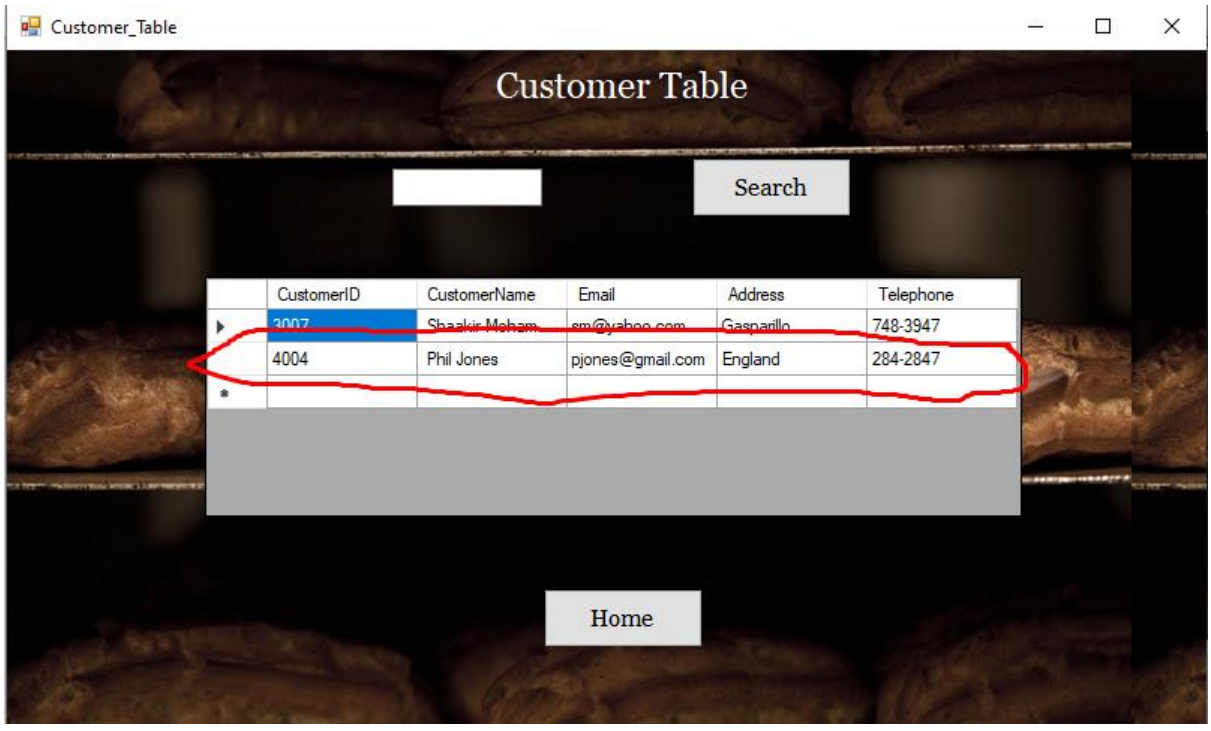
Information that was retrieved from the pastry table:

For the Retrieve functionality, it was utilized across all classes (Employee, Customer, Pastry and Order). The same type of functionality was used throughout each class so one example was given to show that it works. If it didn't work at any point of the system, the document would take note of that.

Add:

For this test we will be checking to see if the functionality is available for all of the entities in the application. Below is snippet of the create a customer form:

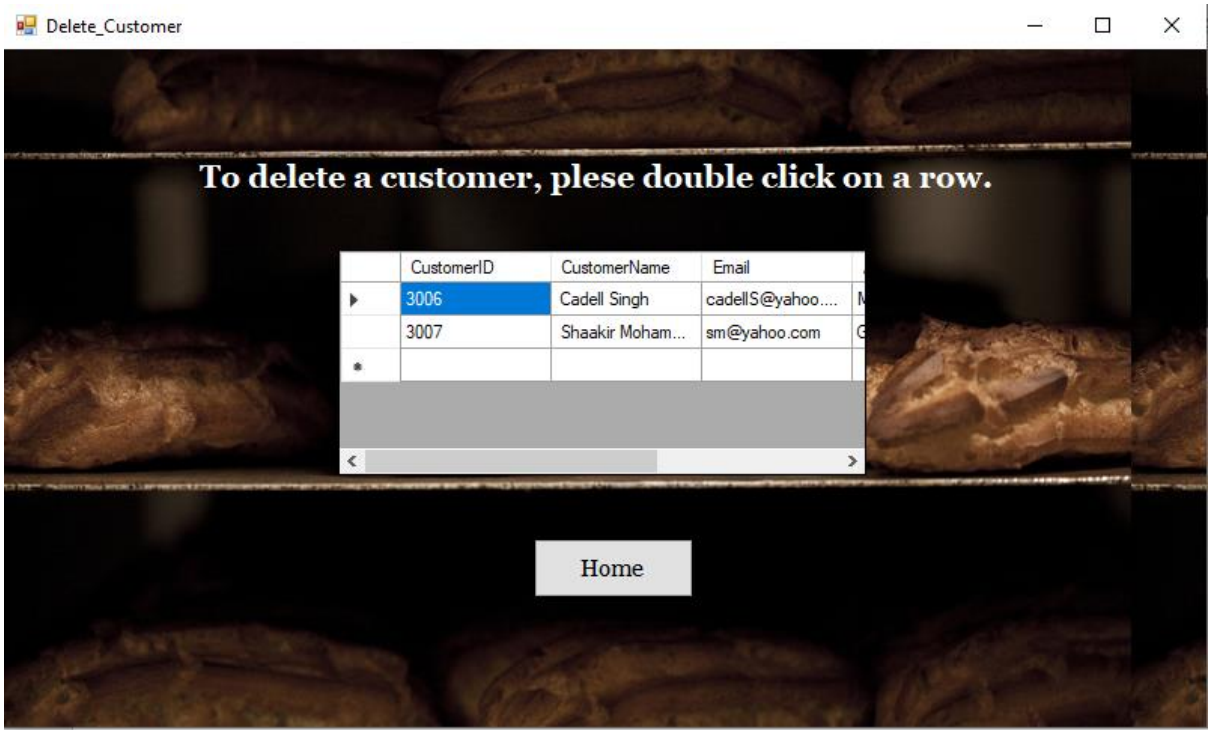


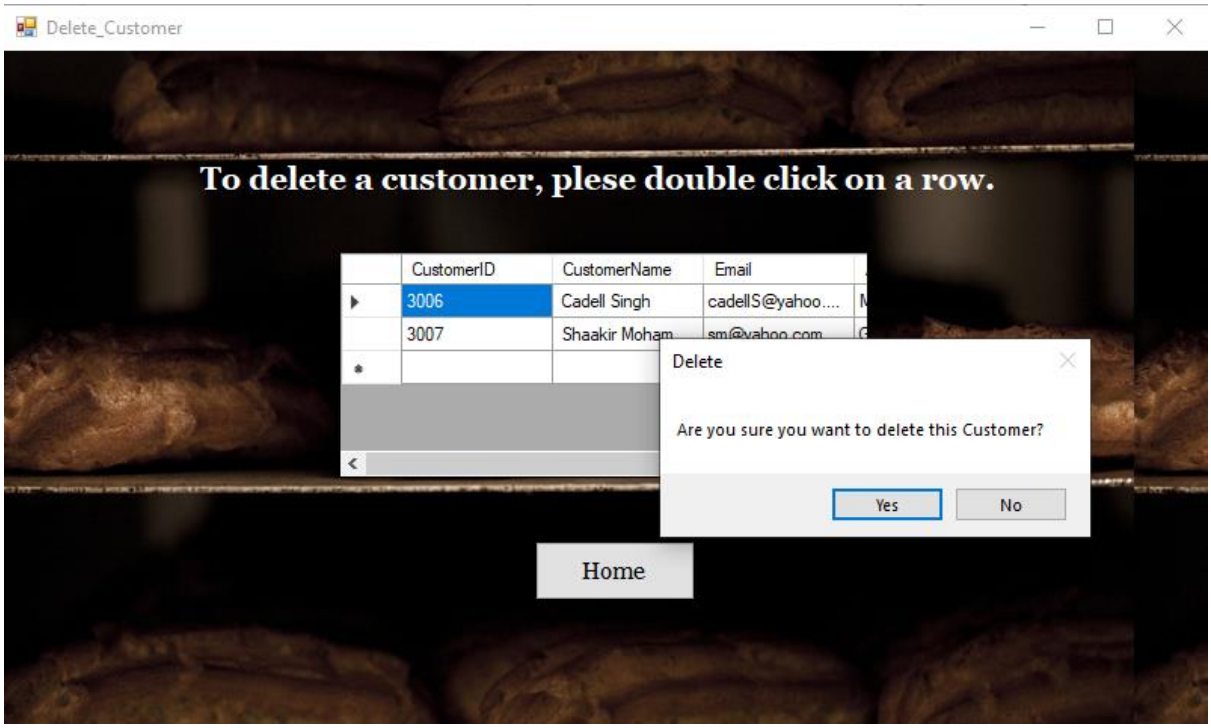


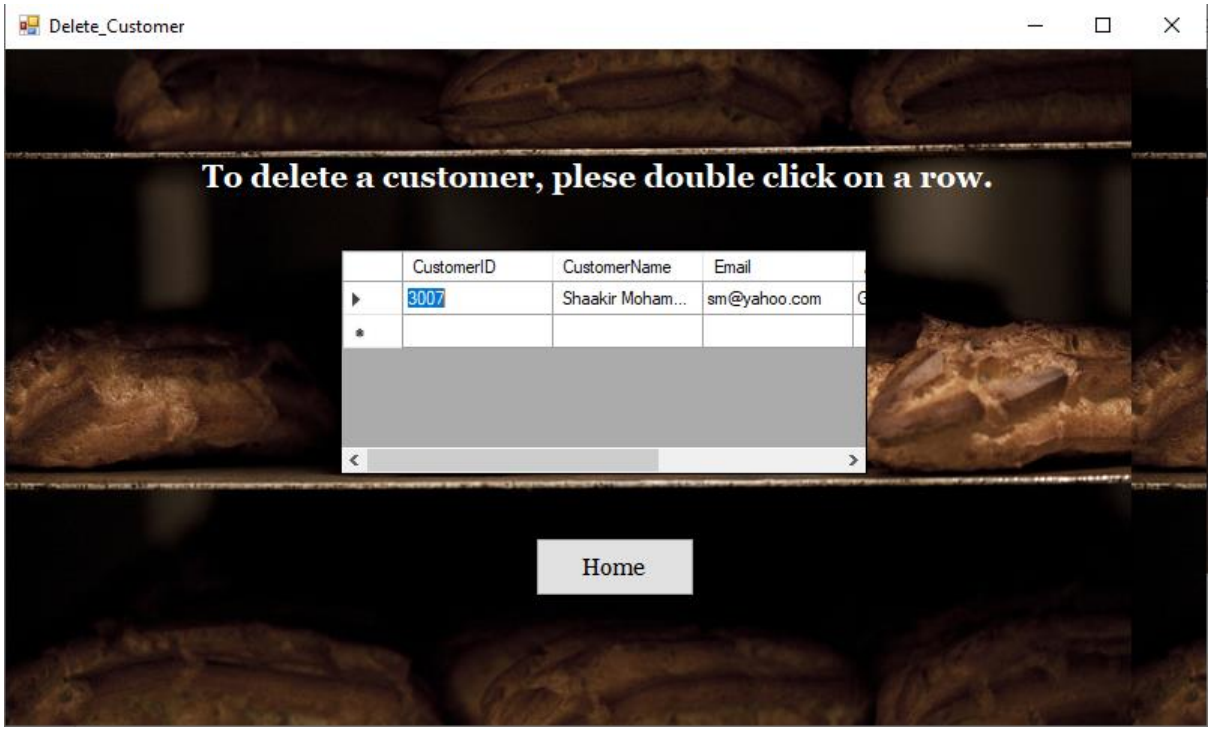
Customer table displaying information entered from the customer form.

For the Create functionality, it was utilized across all classes (Employee, Customer, Pastry and Order). The same type of functionality was used throughout each class so one example was given to show that it works. If it didn't work at any point of the system, the document would take note of that.

Delete:

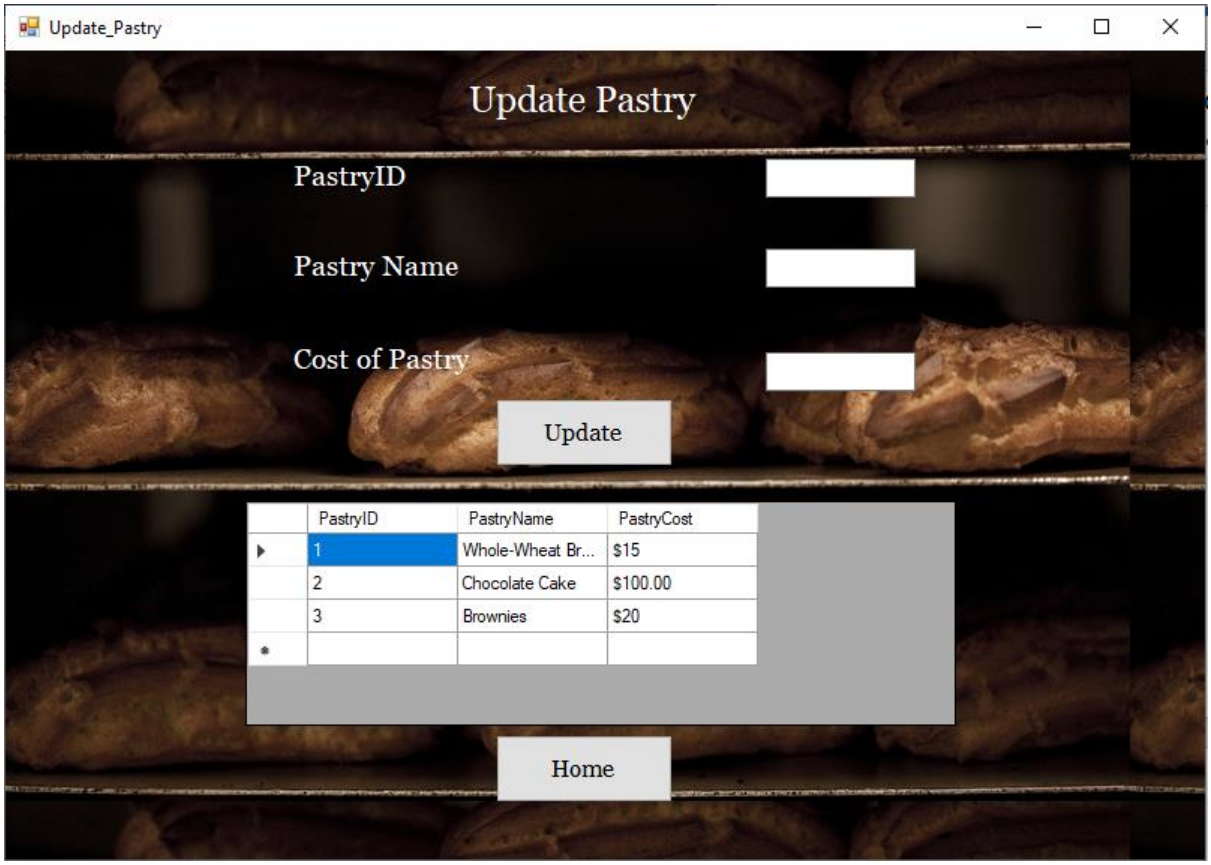

Delete_Customer
To delete a customer, plese double click on a row.
CustomerID
CustomerName
Email
3006
Cadell Singh
cadellS@yahoo....
3007
Shaakir Moham...
sm@yahoo.com
Home

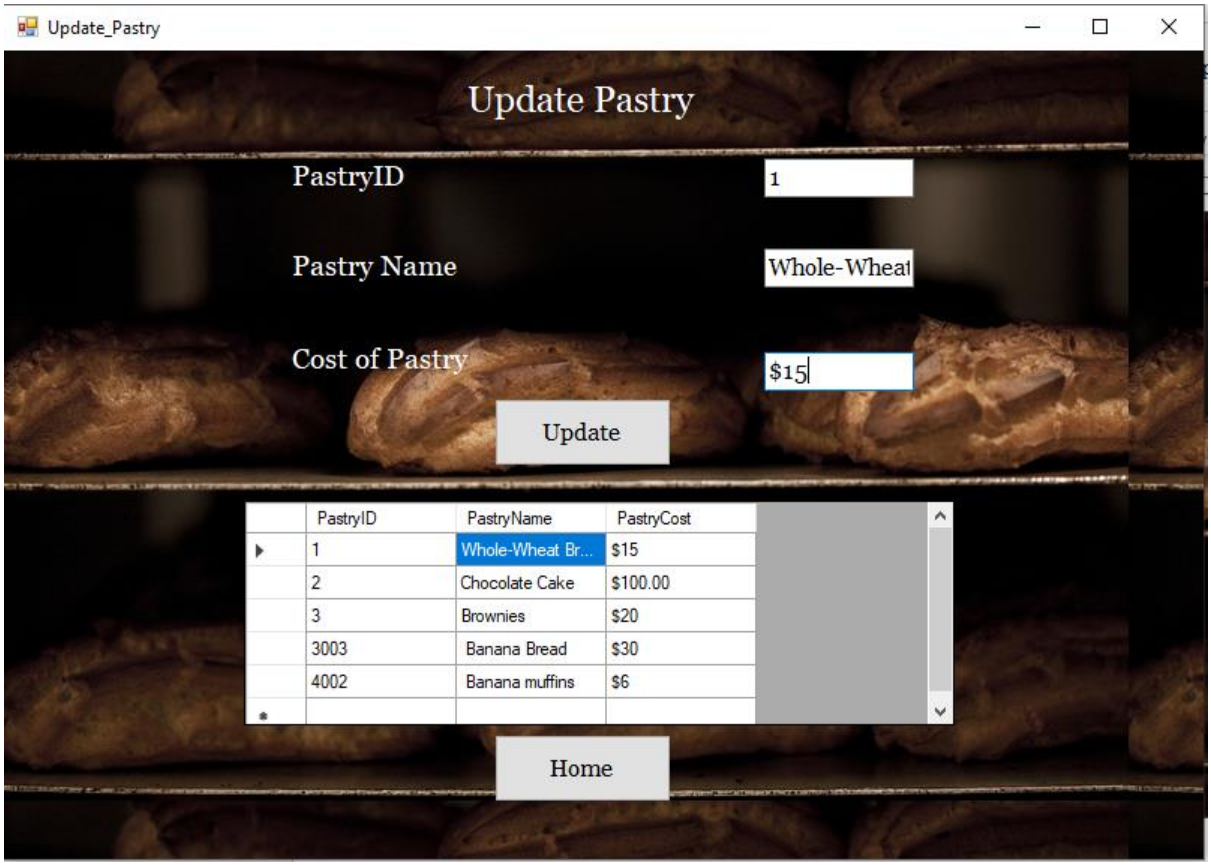




For the Delete functionality, it was utilized across all classes (Employee, Customer, Pastry and Order). The same type of functionality was used throughout each class so one example was given to show that it works. If it didn't work at any point of the system, the document would take note of that.

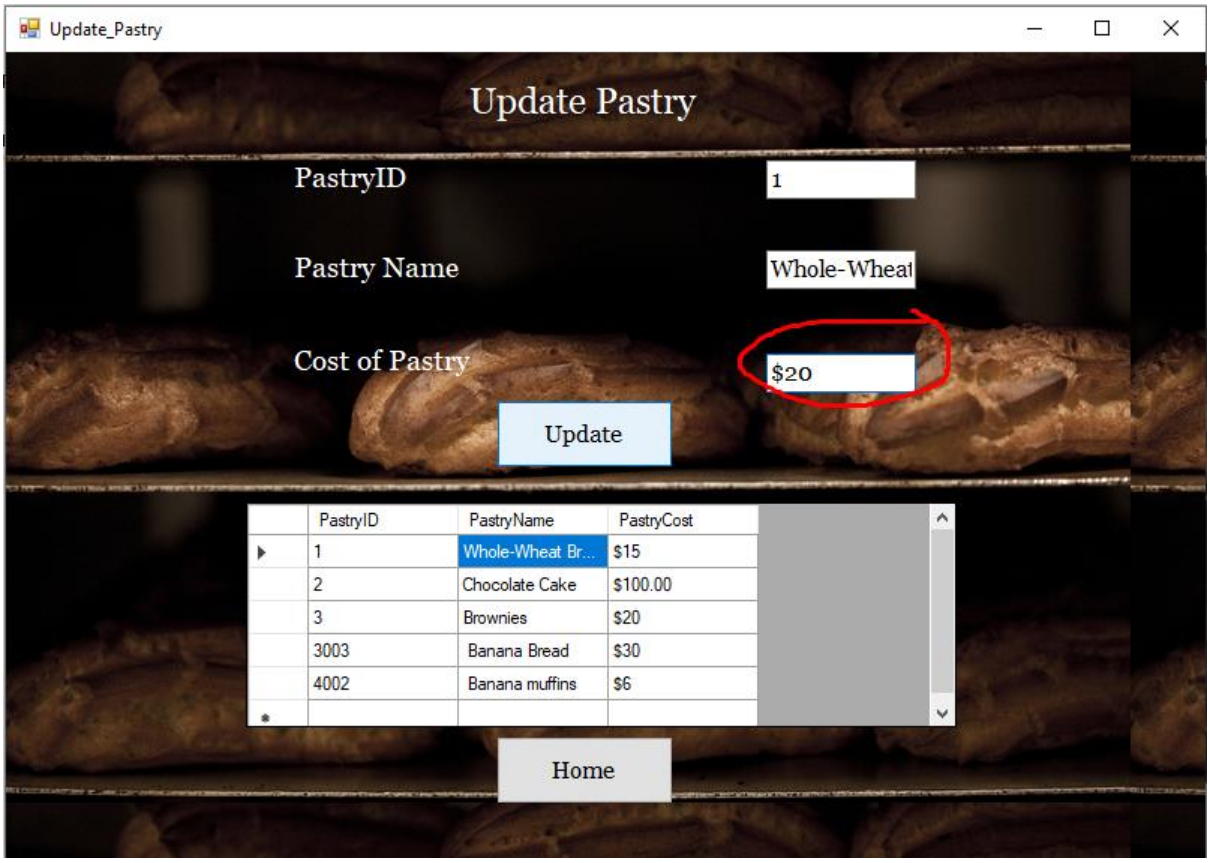
Update:

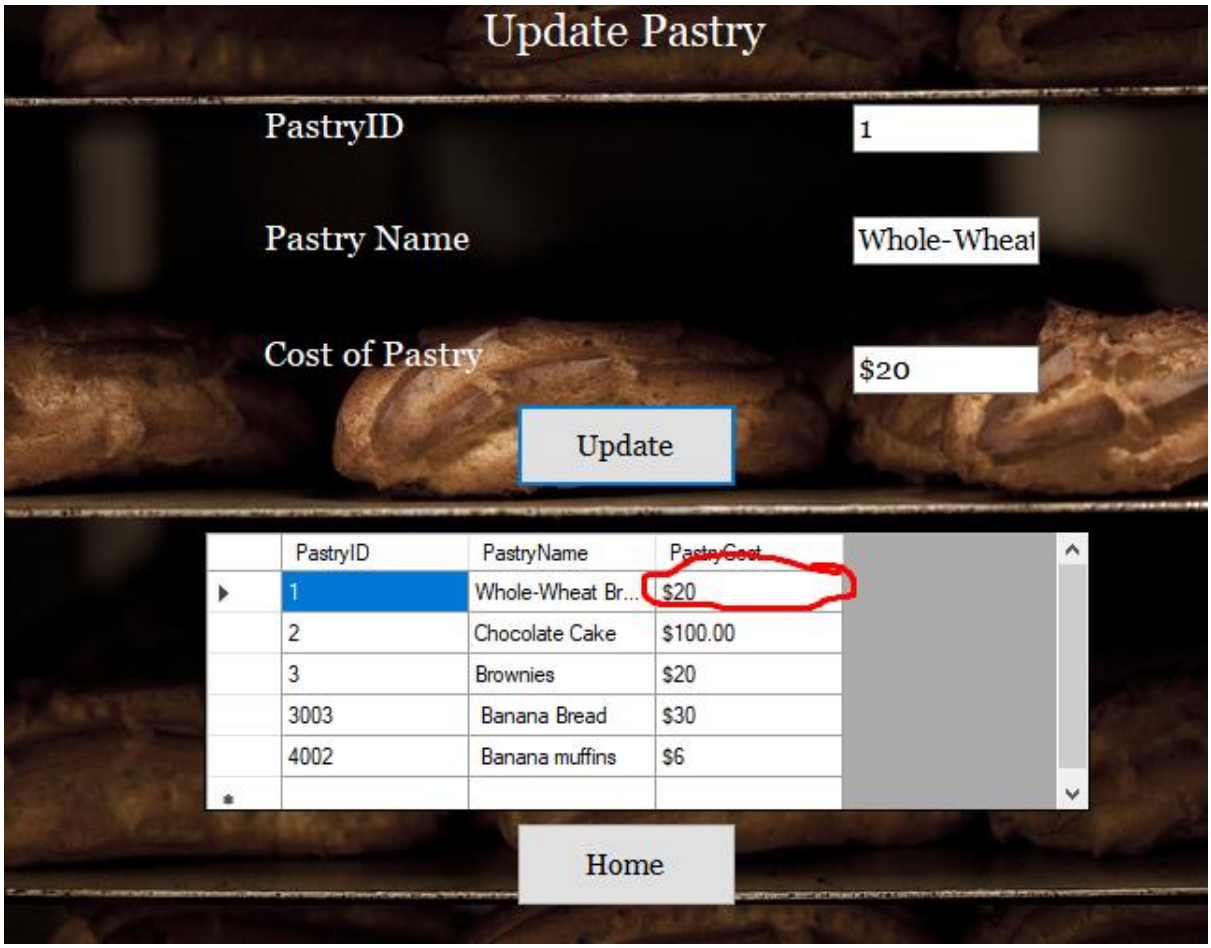

Update_Pastry
Update Pastry
PastryID
Pastry Name
Cost of Pastry
Update
PastryID
PastryName
PastryCost
1
Whole-Wheat Br...
$15
2
Chocolate Cake
$100.00
3
Brownies
$20
Home



Update Pastry (before)

For the update, we will be changing the price of the item from $15 to $20. The Pastry Name is also able to change but for the sake of this example, we are only changing the Cost. *The pastryID, although it shows up that it could change, the functionality for it to do so isn't there on purpose.*





Update Pastry (After)

For the Update functionality, it was utilized only in 2 classes (Employee and Pastry). The same type of functionality was used throughout each class so one example was given to show that it works. If it didn't work at any point of the system, the document would take note of that.

Summary:

Here is a table that summarizes the test done and if the system passes those tests.

Entity	Insert	Update	Search	Delete	Retrieve
Customer	Yes	No	Yes	Yes	Yes
Employee	Yes	No	Yes	Yes	Yes
Pastry	Yes	Yes	Yes	Yes	Yes
Order	Yes	Yes	Yes	Yes	Yes

User Interface Testing

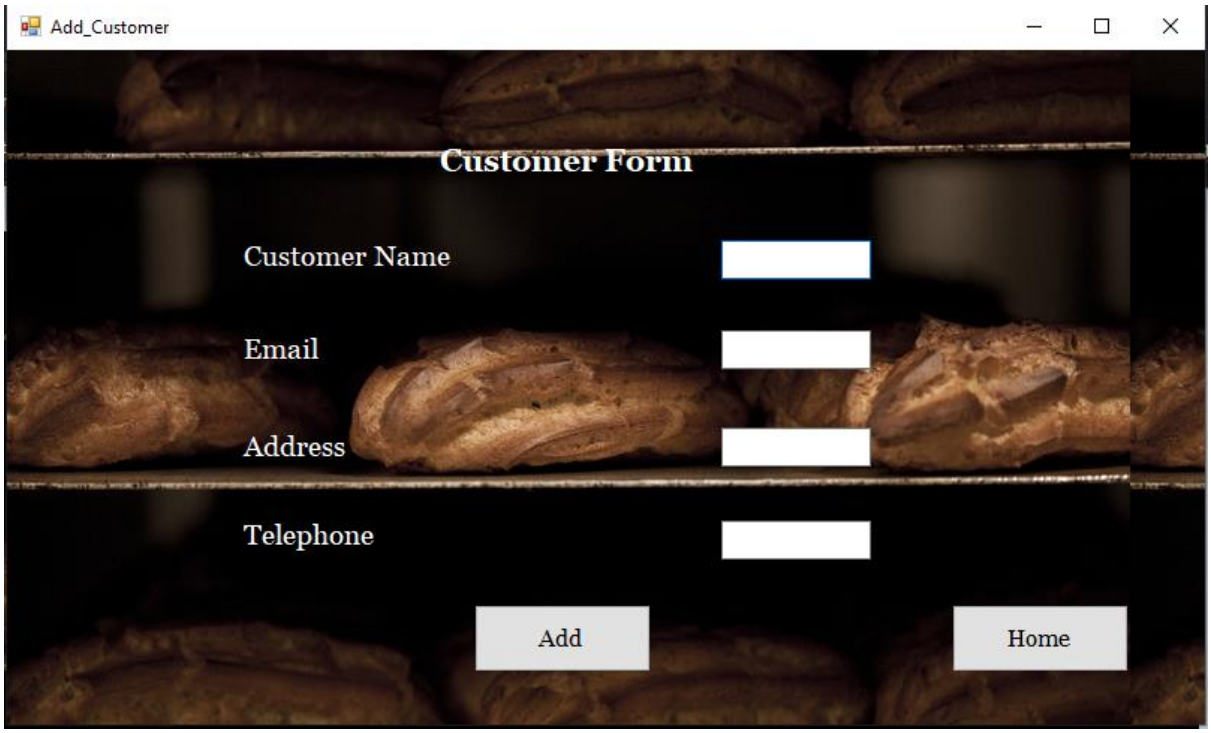
In this test we will be checking ease in which people can view the user interface as well as the ease of which a user can navigate through the application.

We will be checking the home menu of the application first.

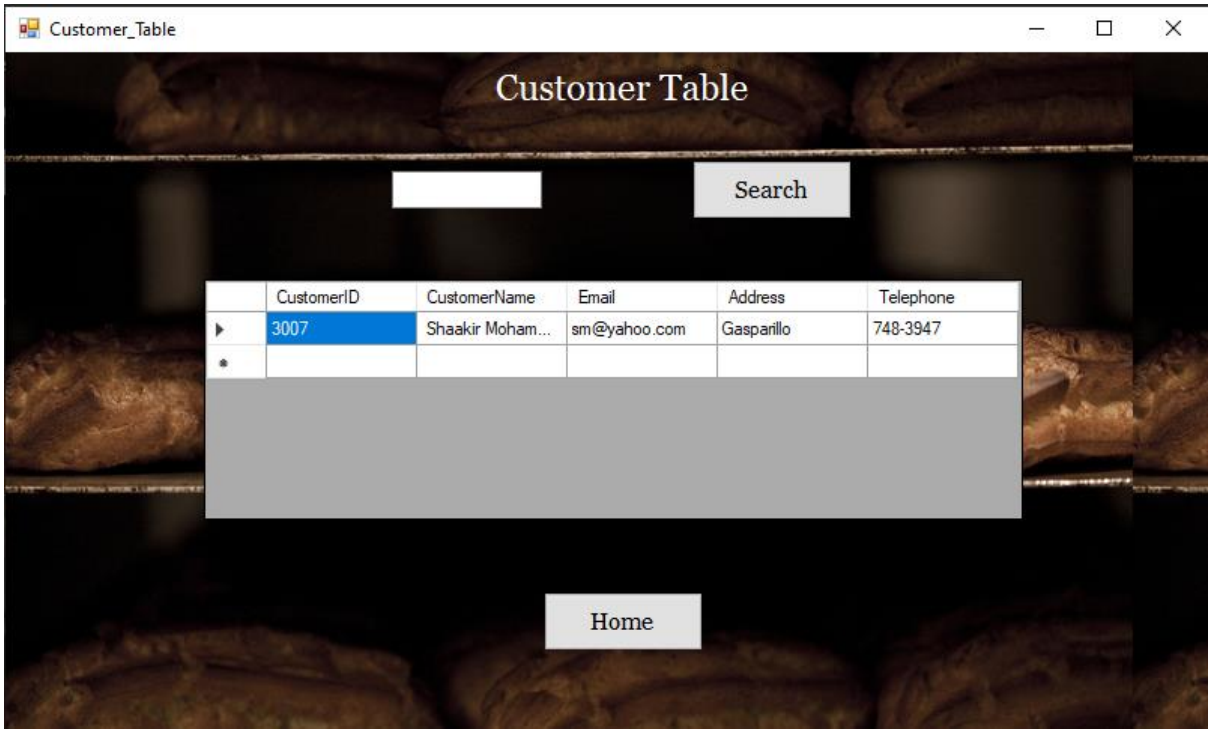
Home Menu

The above image is the home page and from above the font size as very viewable and it is very well labelled in that a person would know what which button does as where it would lead them.

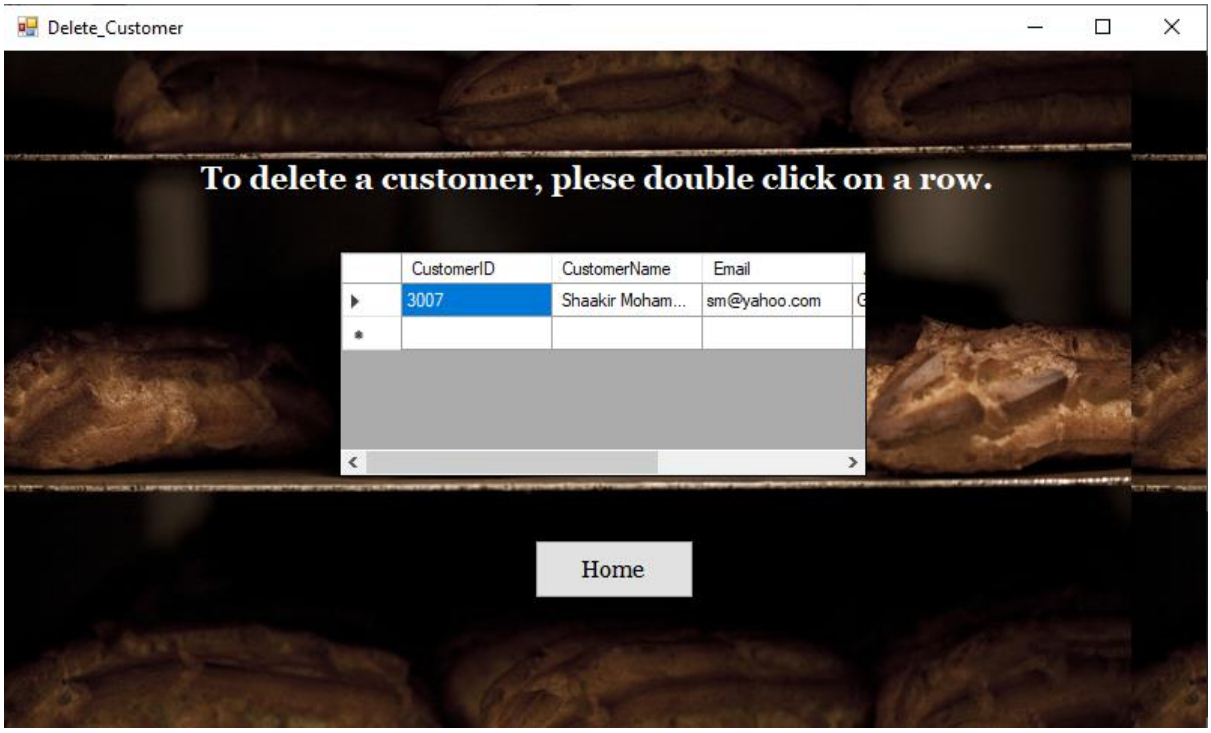
Button such as “Add……”, “View……..”, “Delete……” have the same layout. Below are some screenshots of how it looks.



Customer Form



Customer Table

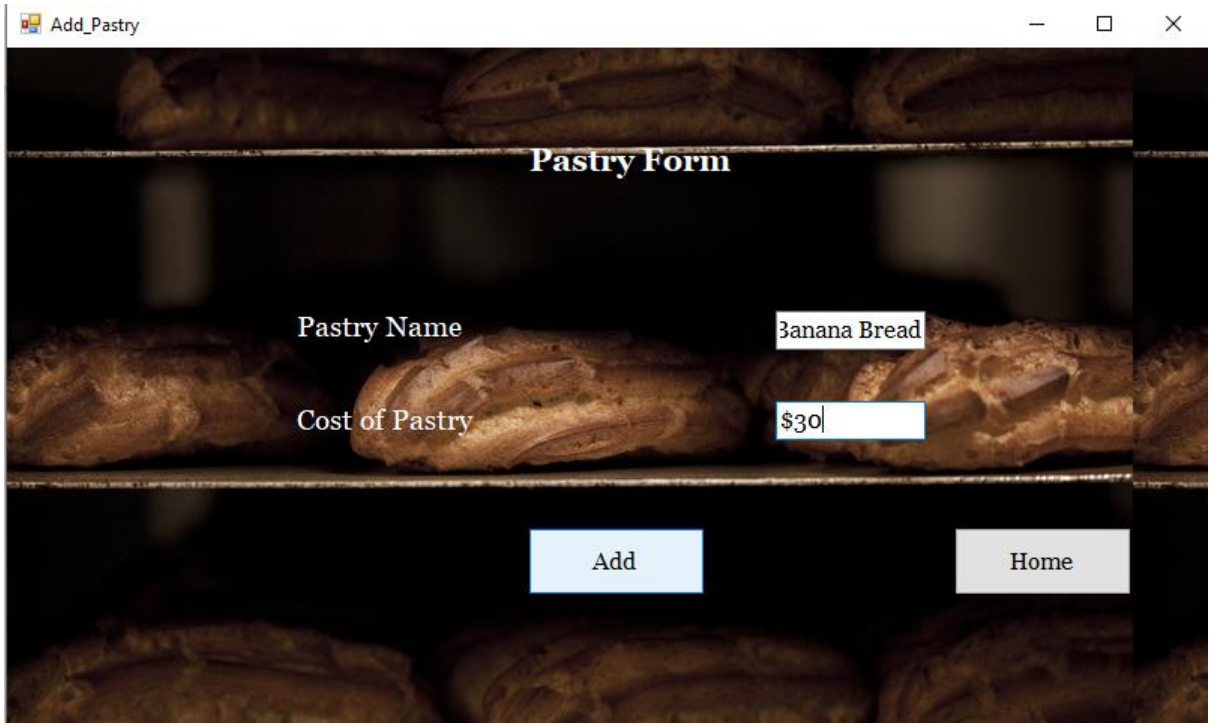


Delete Customer

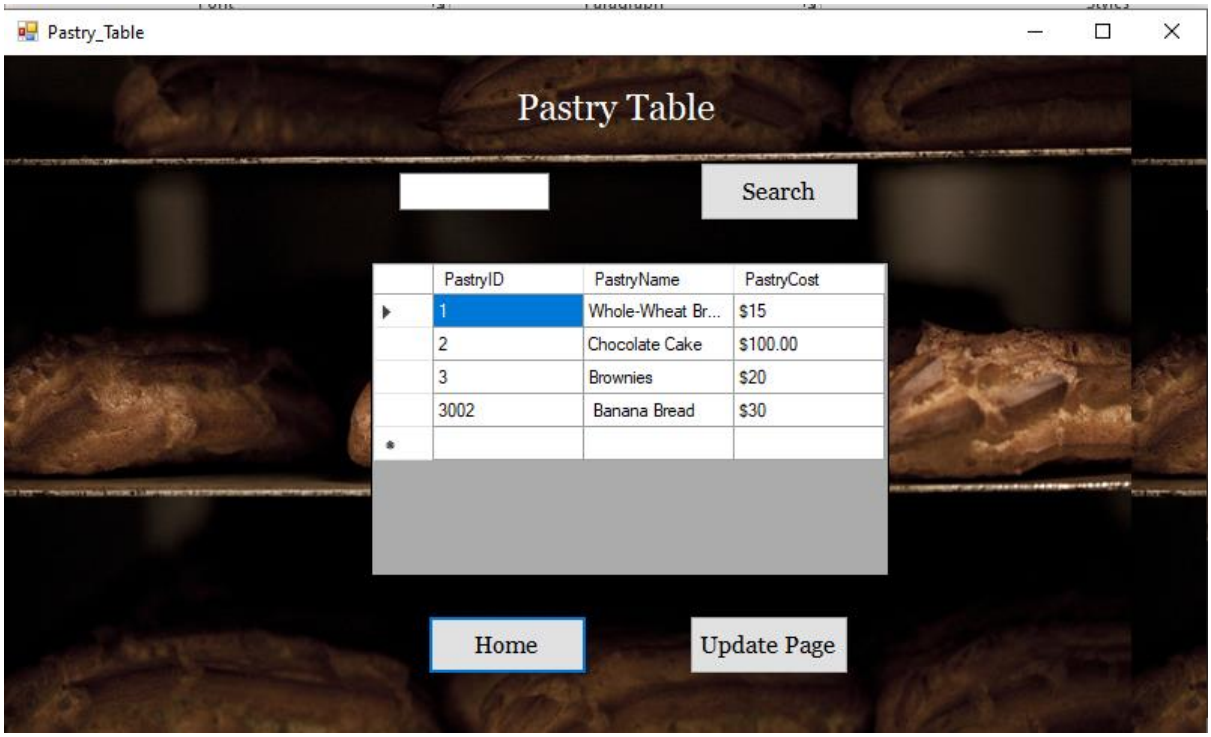
Data and Integrity Testing

In this test, we will be checking how data is entered into the database and how it appears as well as if that is what the user expects to see when entering the data.

Add Pastry:

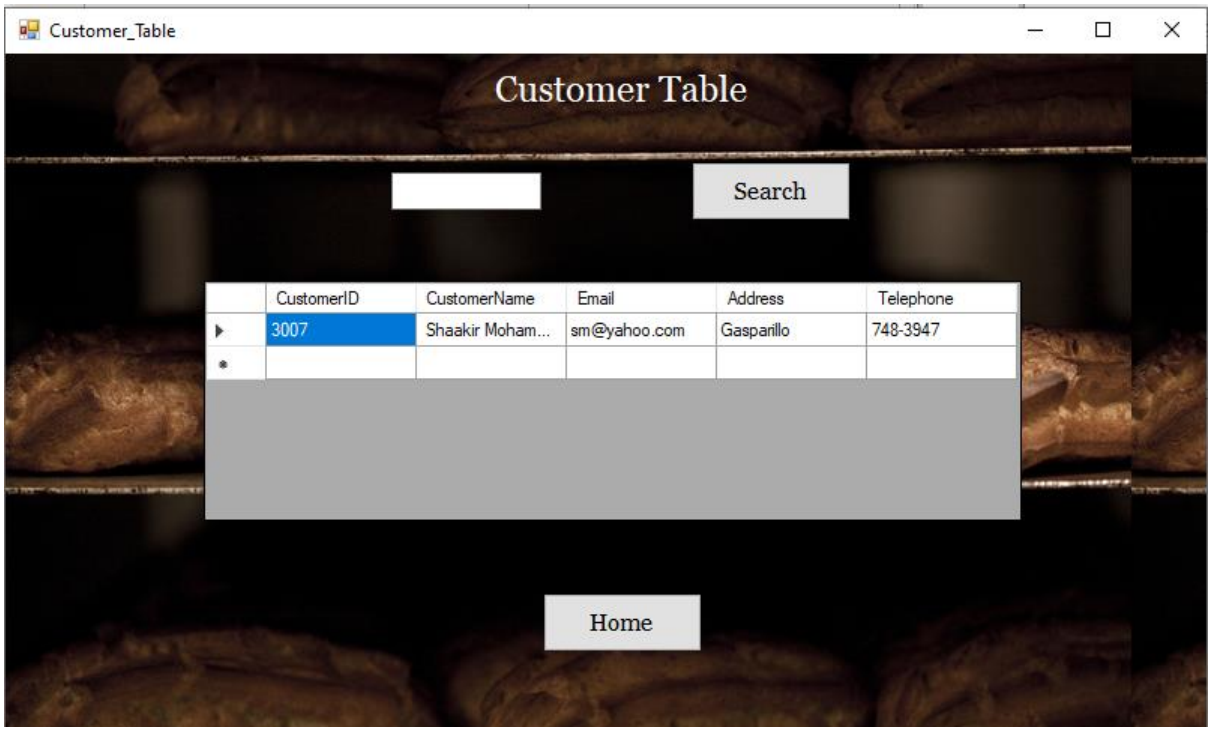


Pastry Form

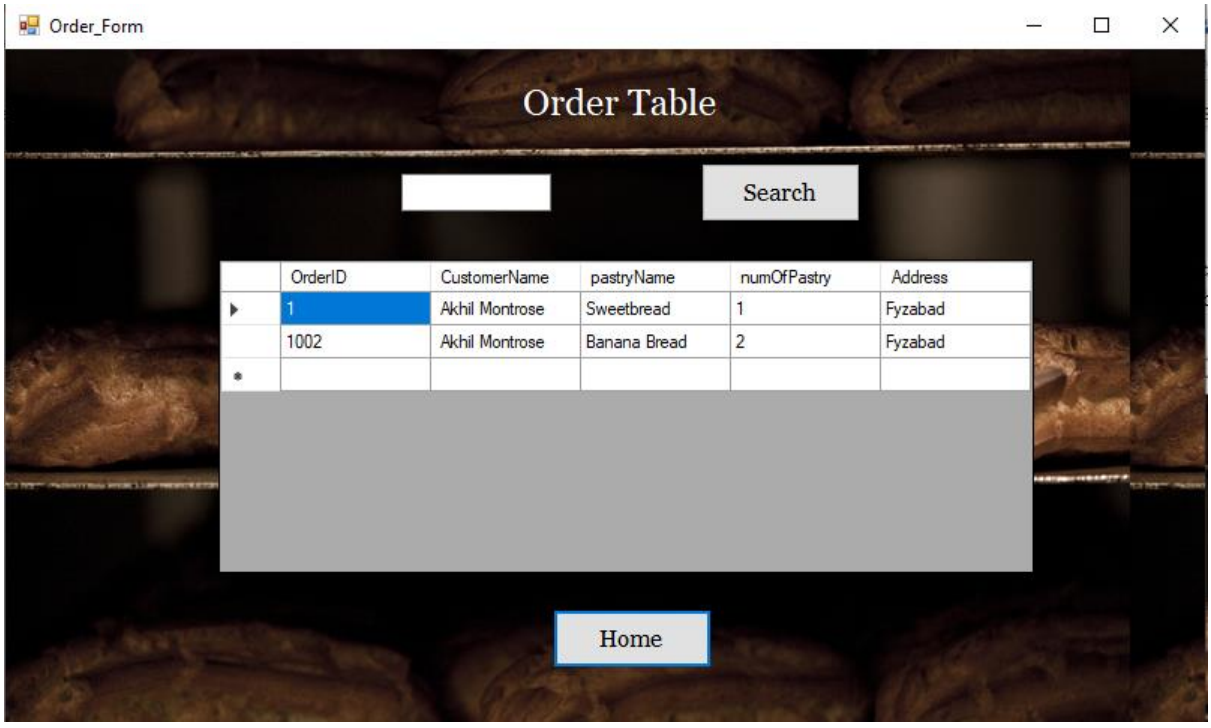


Pastry table

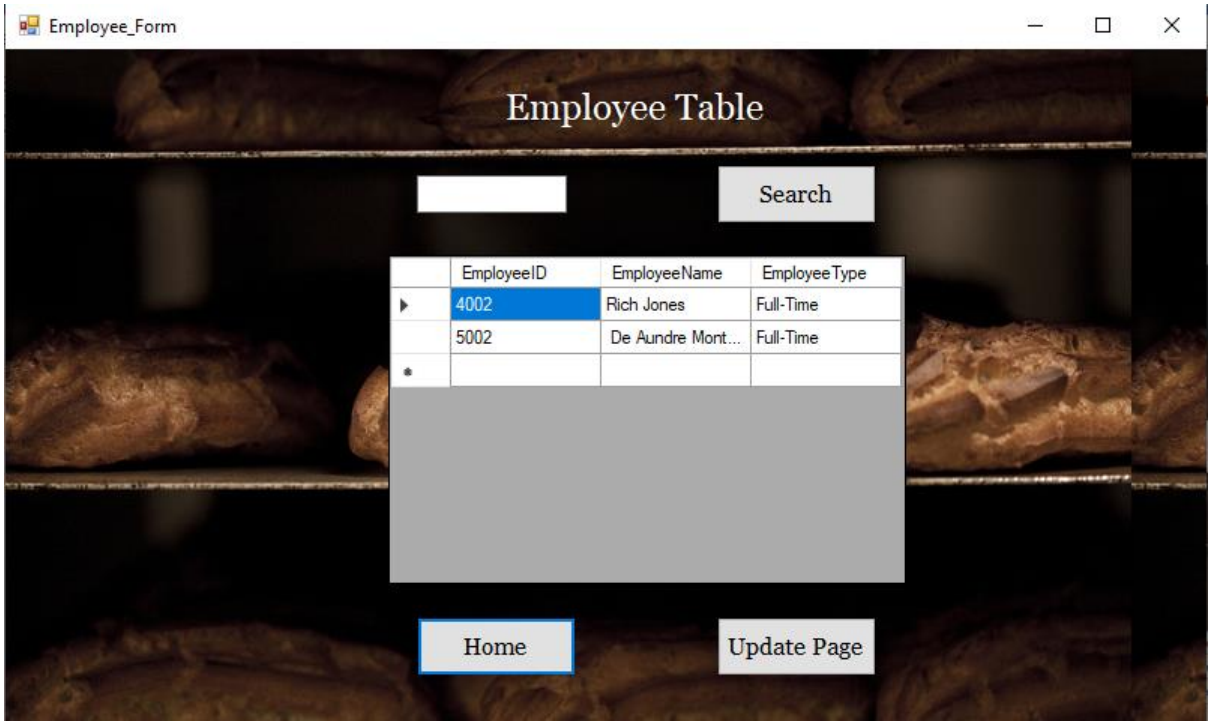
A problem that occurs is that in the table the first set of rows, the primary keys count form 1 – 3 which is correct then it skips to 3002. This is an error that occurs in the database when incrementing the value. This problem also occurs in other entities.



Customer Table



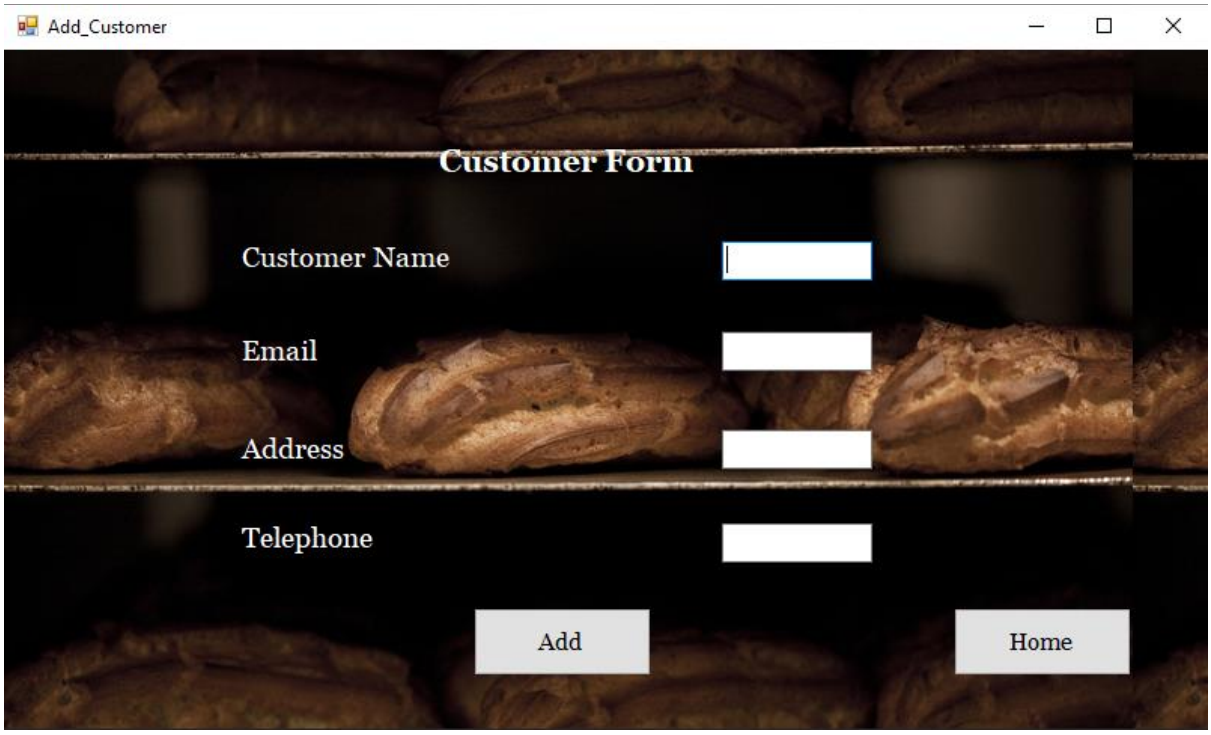
Order Table



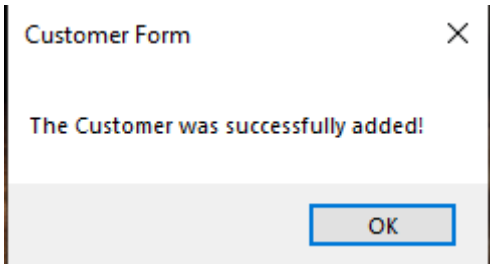
Employee Table

Test: To see if input is accepted if input fields are empty and if an error message shows up if a field is empty.

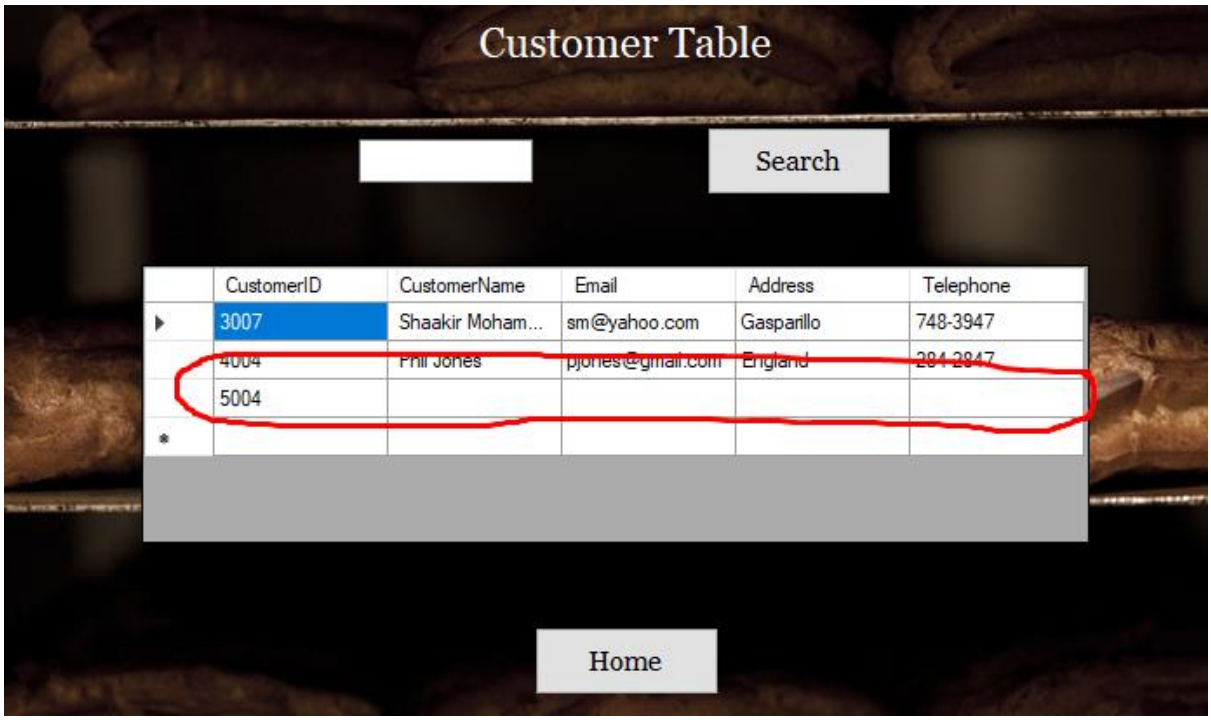
Customer:



Customer Form (empty)

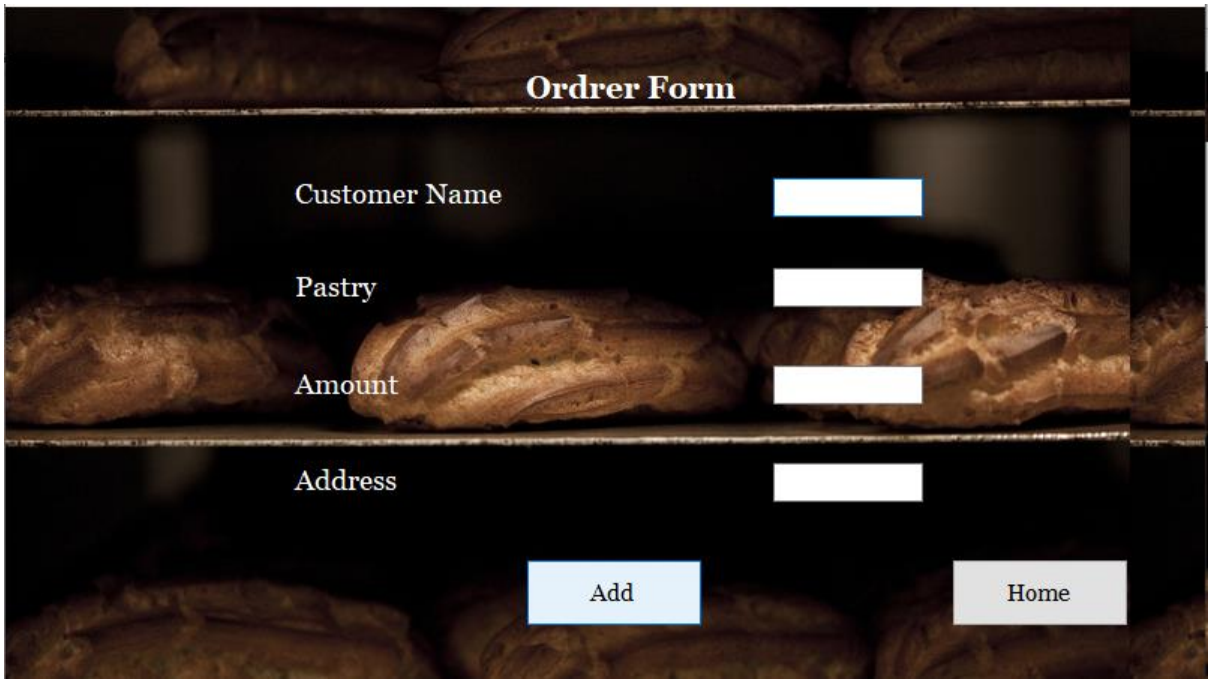


Customer Notification

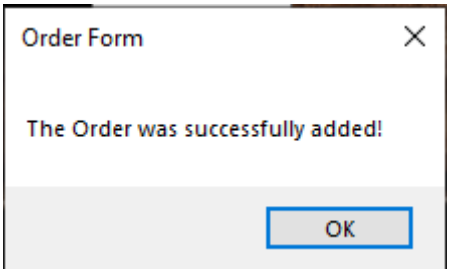


Customer Table showing empty row of data

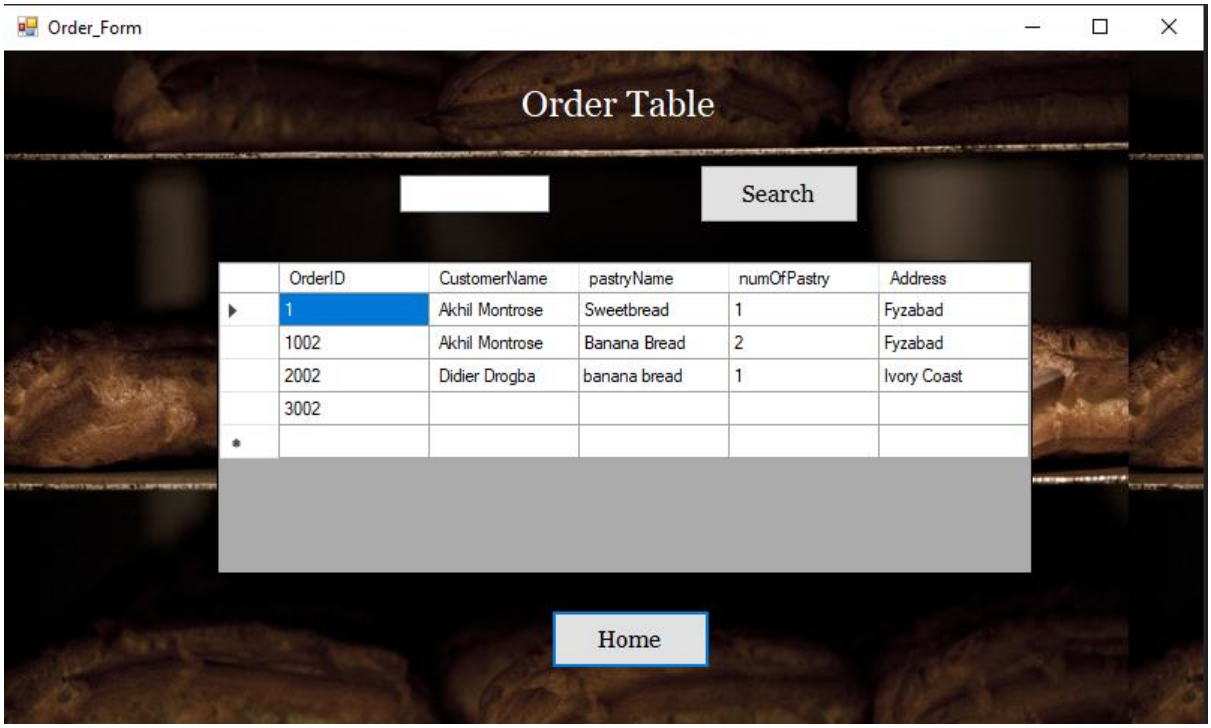
Order:



Order Form (empty)



Order Notification



Order Table showing empty row of data

Summary:

Here is a table that points out the errors if there are any.

Entities	**Does it accept input**	**Does it accept input when all/some fields are empty**	**Does it give an error message when a user makes an error/ leaves up a field**	**Does it display information**	**Is the information displayed correctly**
Customer	Yes	Yes	No	Yes	No (IDs)
Employee	Yes	Yes	No	Yes	No (IDs)
Pastry	Yes	Yes	No	Yes	No (IDs)
Orders	Yes	Yes	No	Yes	No (IDs)

Solution (For when it accepts input when all/some fields are empty):

What can be done to fix this issue is to implement:

- NOT NULLS in the database to ensure that no information will be entered if the fields are empty.
- Utilize form validation in the application so that if a user tries to enter information without completing some/all fields, it will give an error message.

Blackbox Testing

For this particular test we will be testing everything on the end user side. What is going to be observed is if the end user is getting the appropriate feedback that he/she expects to get. We will be testing navigation, input, as well as output.

Navigation:

When interacting with the navigation of the application, there were no problems going to the pages that was requested. All pages and buttons worked as it is intended.

Home Screen

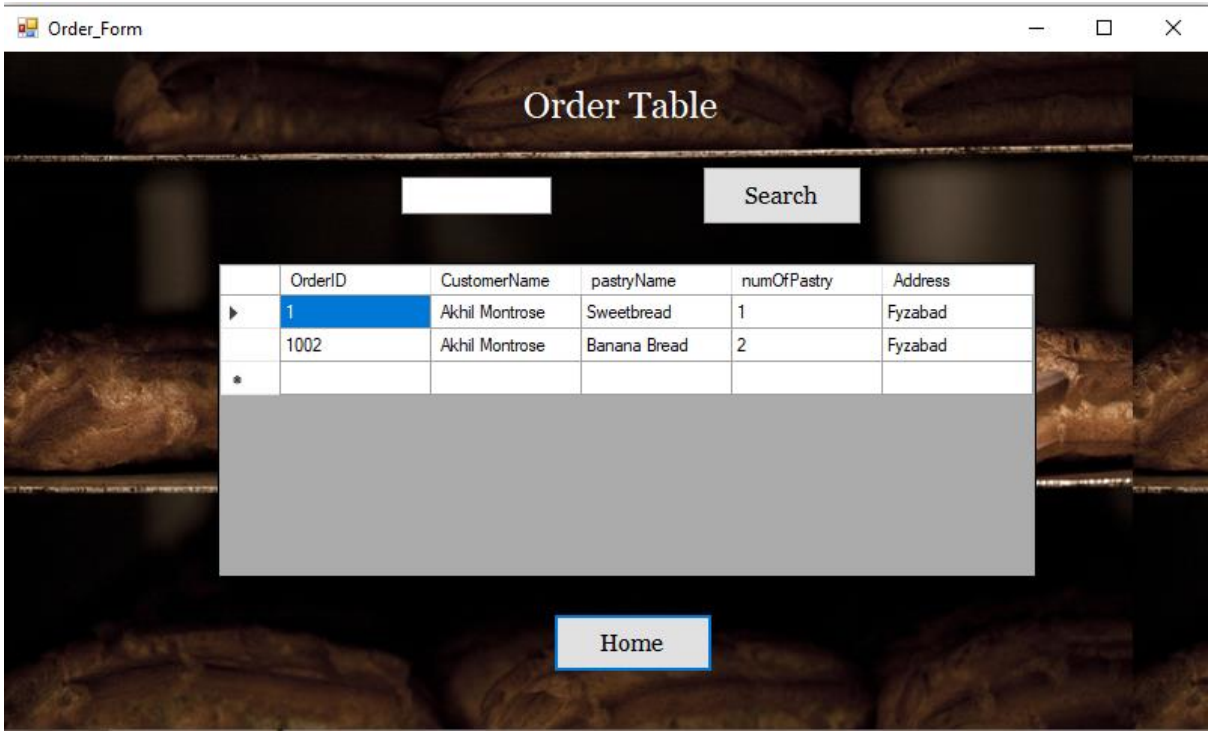
Input:

Input was tested to see if that the input that was taken by the system would be saved the way that the end users expects it to.

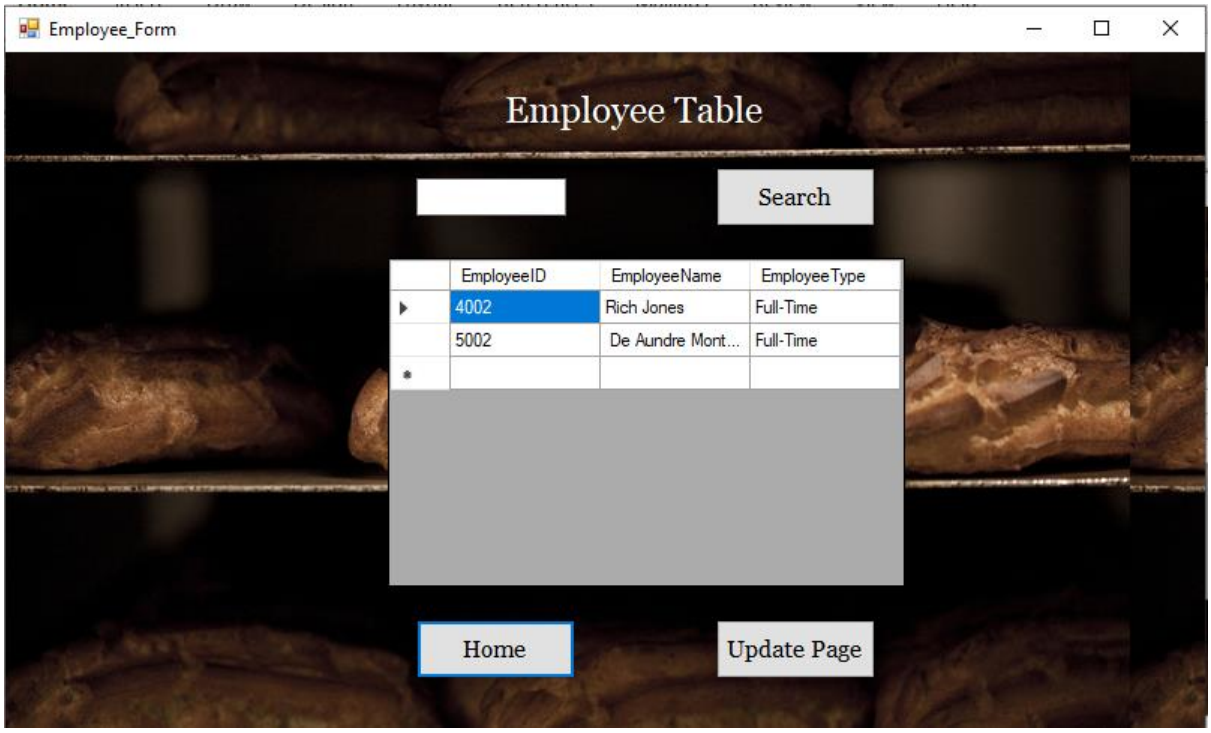
The tests were done and data was indeed being taken into the system.

Output:

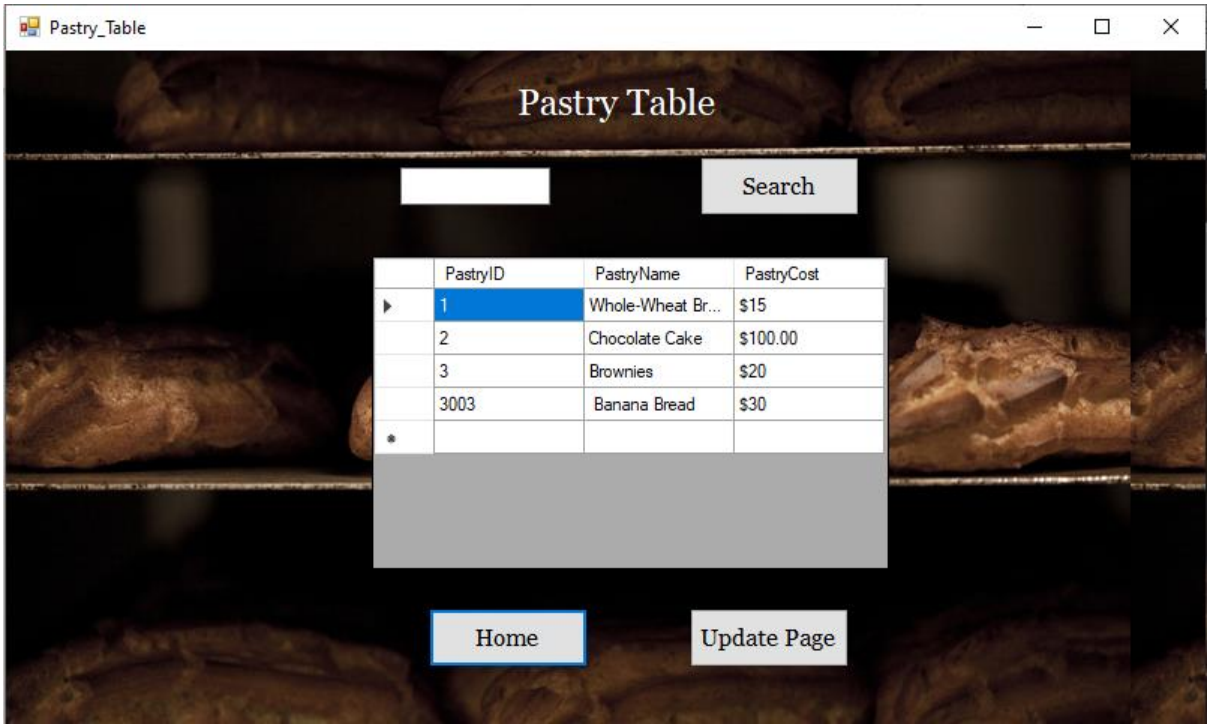
From the previous point, while data was taken into the system, the information being displayed is not always right, in particular, the IDs of the data. Below are some screenshots showing this error.



Order table



Employee Table



Pastry Table

Summary:

While there are no problems concerning the navigation through the website as well as what the user enters into the system, there is an error with what is being outputted.

Navigation	No Issue arose when trying to navigate the application from the end user side.
Input	No Issues arose when trying to input data into the fields
Output	A bug occurred with the IDs incrementing more than what is expected.

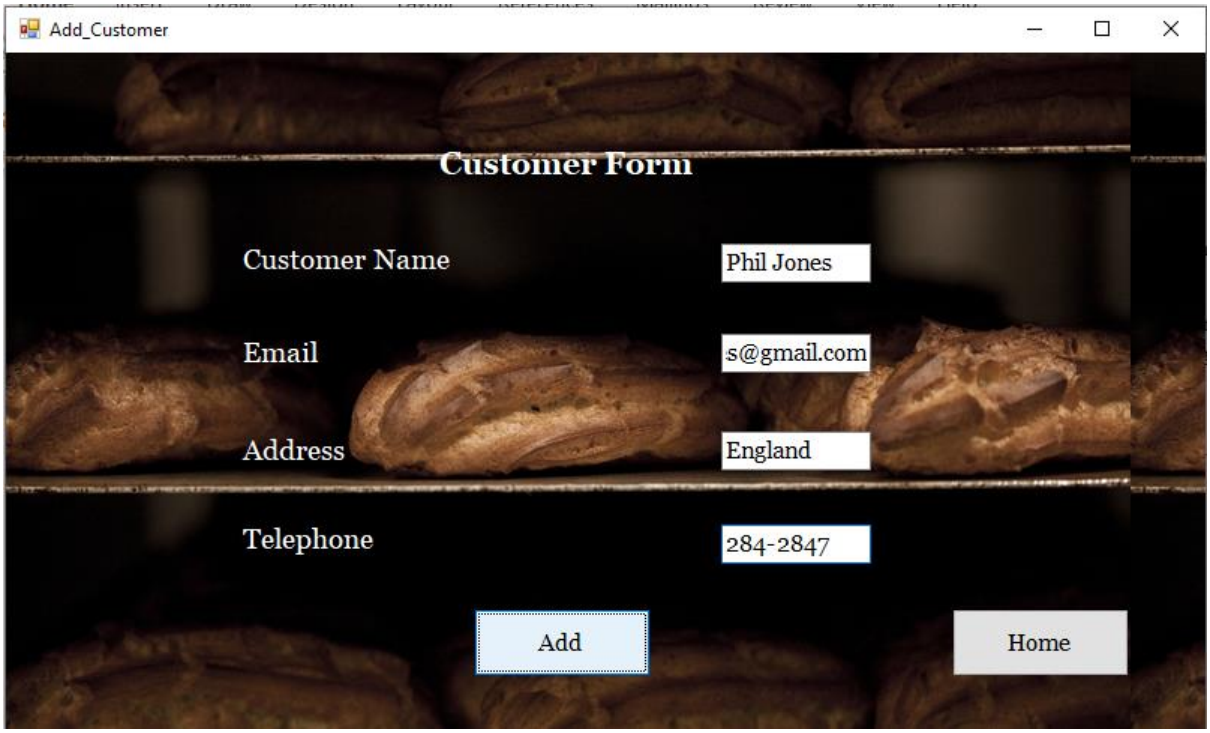
Whitebox Testing

In this type of test we will be observing the inner working so the application to address the issues that were present in blackbox testing.

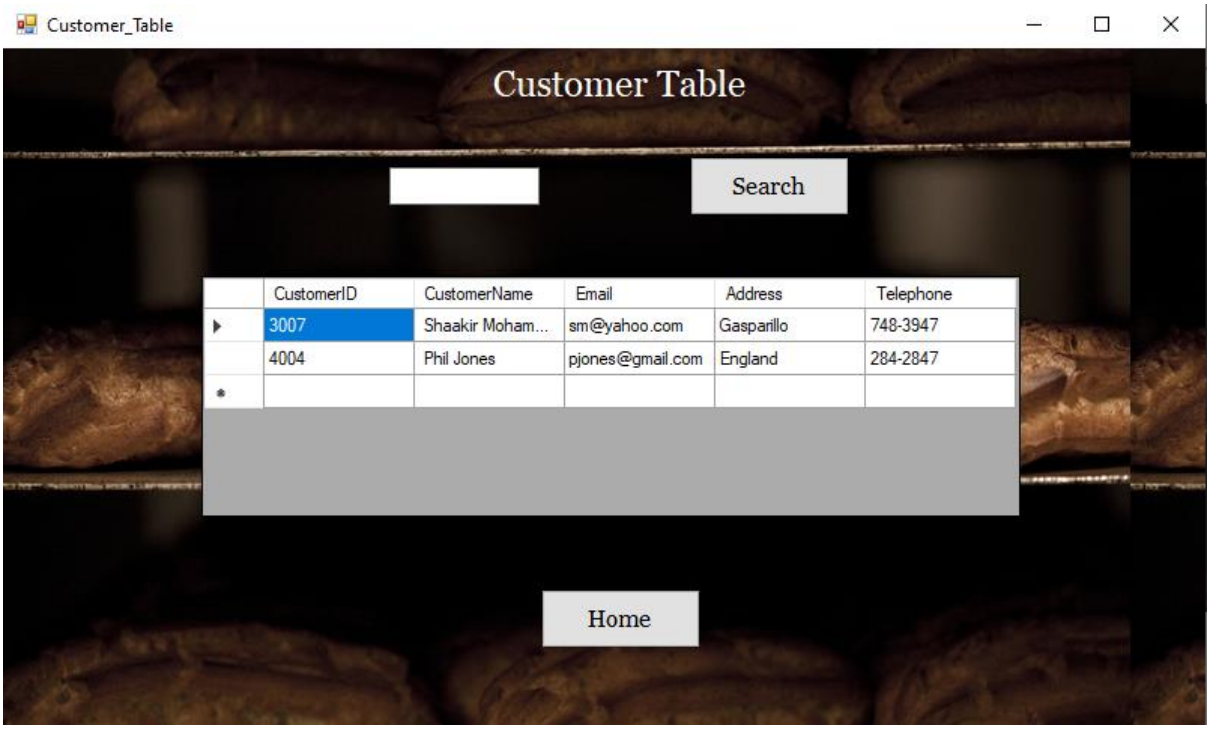
When we performed the blackbox testing, we noticed that there was an error with the IDs giving values higher than what the end user was expecting. We will be going deeper in whitebox testing by checking the internal working so of the application.

Customer

We will now provide the application with inputs and see what the outputs will be:



Customer Form

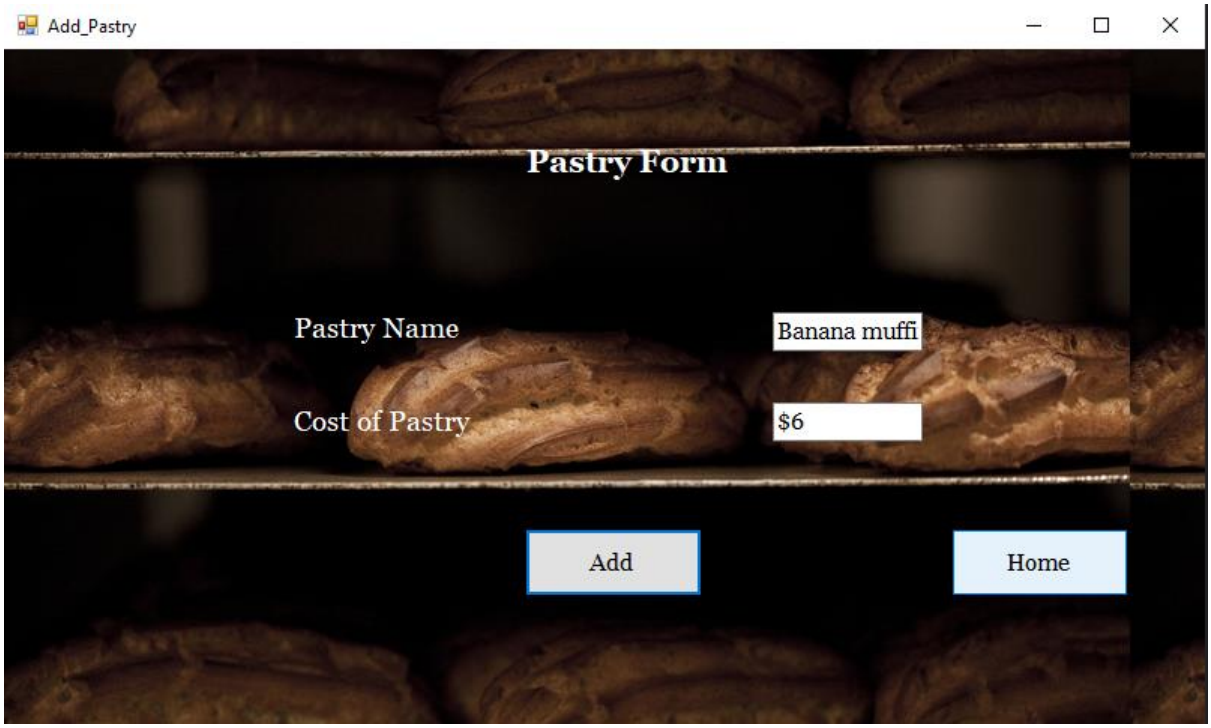


Please keep in mind, while testing, values were deleted from the tables but nonetheless the values shouldn't be so far apart.

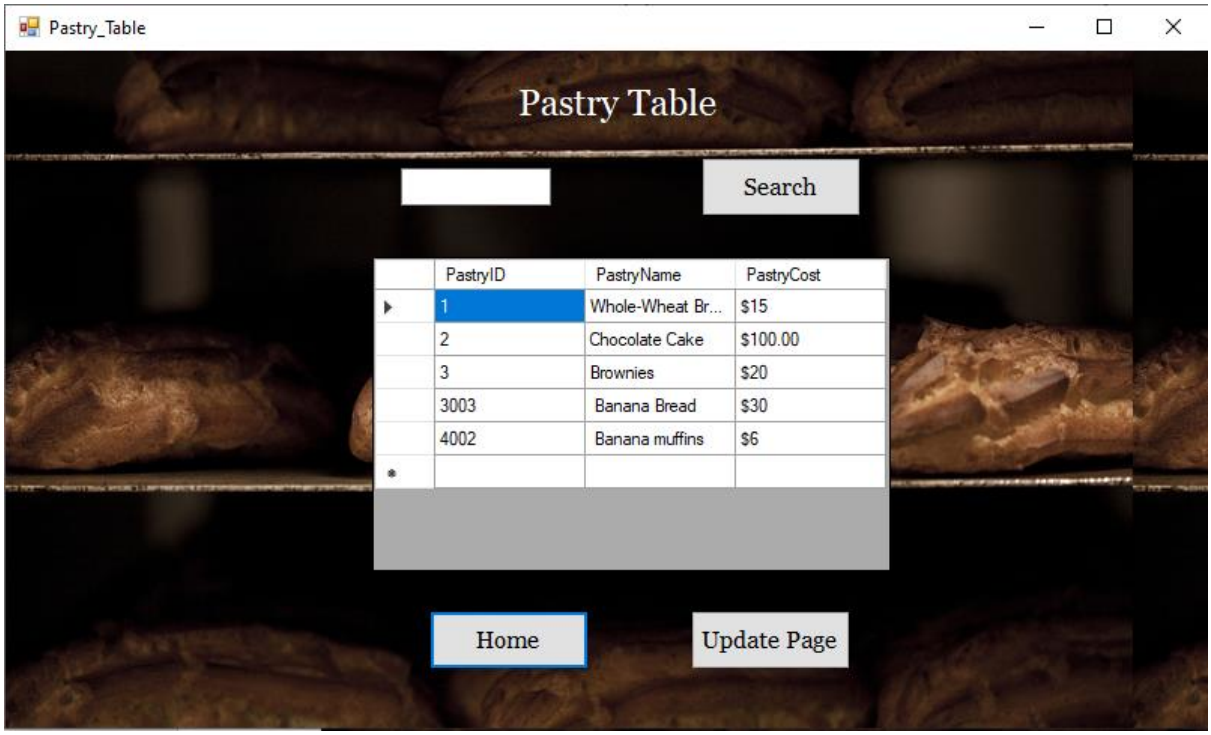
Below is the code for the associated insert Customer.

```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    Dim connection As New SqlConnection
    Dim command As New SqlCommand

    Try
        connection.ConnectionString = "Data Source=(LocalDb)\akhilmdb;Initial Catalog=Bakery;Integrated Security=True"
        connection.Open()
        command.Connection = connection
        command.CommandText = "insert into Customer values (' " & TextBox1.Text & "','" & TextBox2.Text & "','" & TextBox3.Text & "','" & TextBox4.Text & "')"
        command.ExecuteNonQuery()

        MsgBox("The Customer was successfully added!", Title:="Customer Form")

    Catch ex As Exception
        MessageBox.Show("Error while inserting record on table..." & ex.Message, "Insert Records")
    Finally
        connection.Close()
    End Try
```

Pastry:



Pastry From

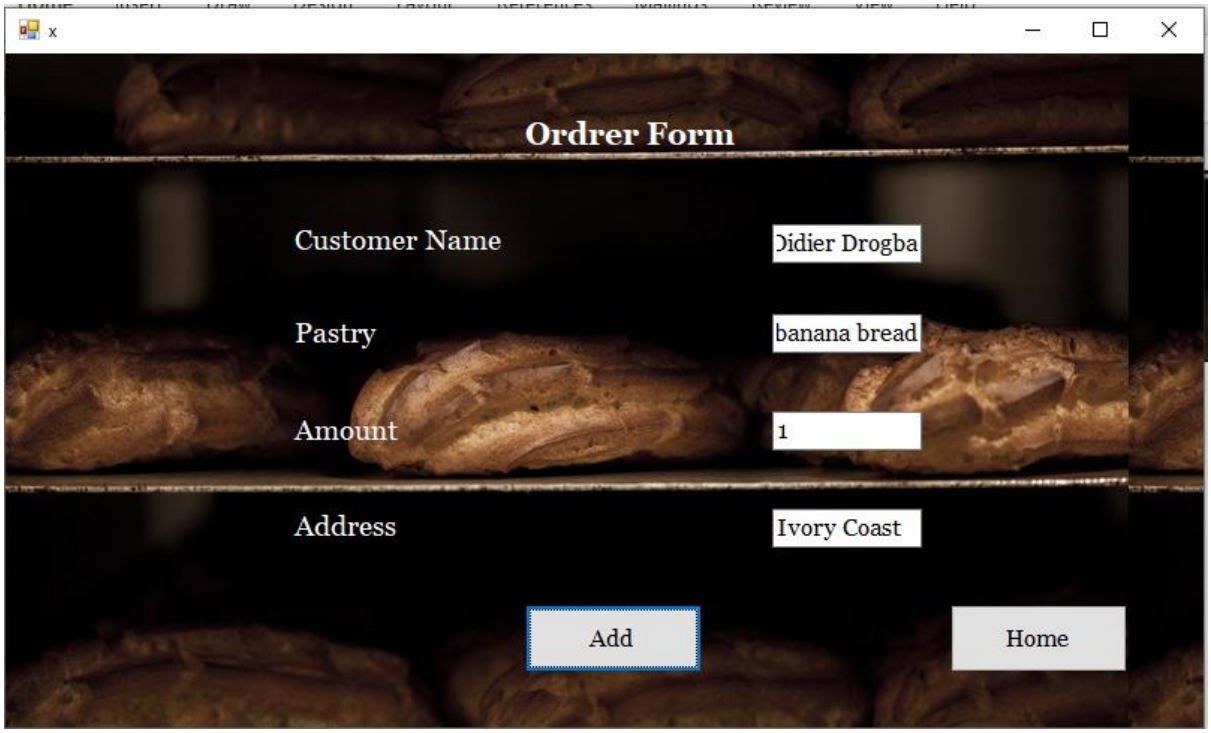


Pastry Table

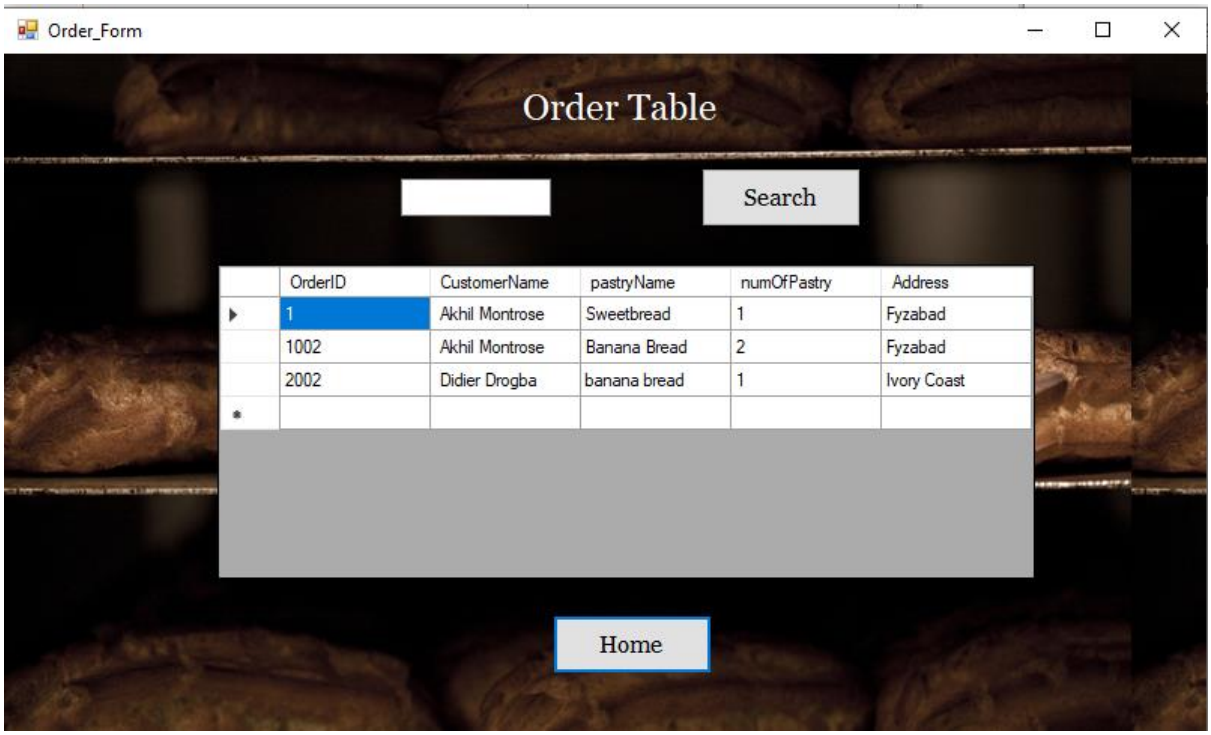
```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    Dim connection As New SqlConnection
    Dim command As New SqlCommand

    Try
        connection.ConnectionString = "Data Source=(LocalDb)\akhilmdb;Initial Catalog=Bakery;Integrated Security=True"
        connection.Open()
        command.Connection = connection
        command.CommandText = "insert into Pastry values (' " & TextBox1.Text & "','" & TextBox2.Text & "')"
        command.ExecuteNonQuery()

        MsgBox("The Pastry was successfully added!", Title:="Pastry Form")

    Catch ex As Exception
        MessageBox.Show("Error while inserting record on table..." & ex.Message, "Insert Records")
    Finally
        connection.Close()
    End Try
End Sub
```

Code for creating a Pastry

Order:



Order Form

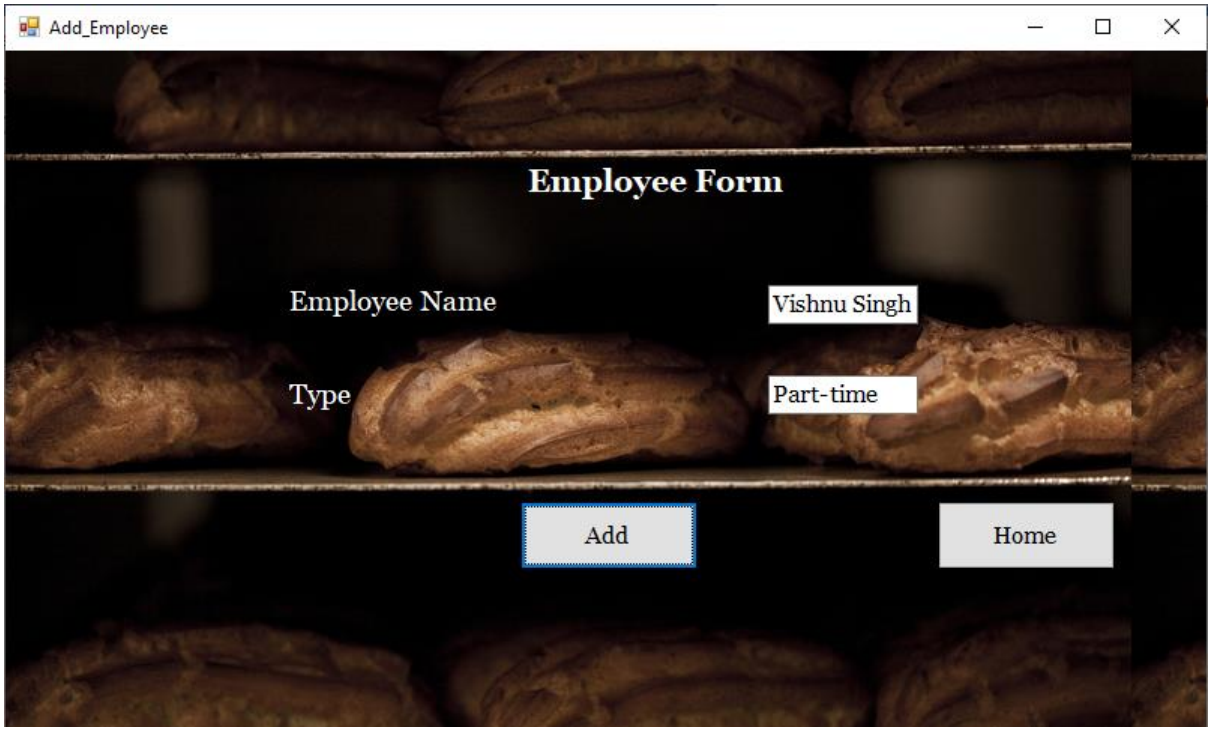


Order Table

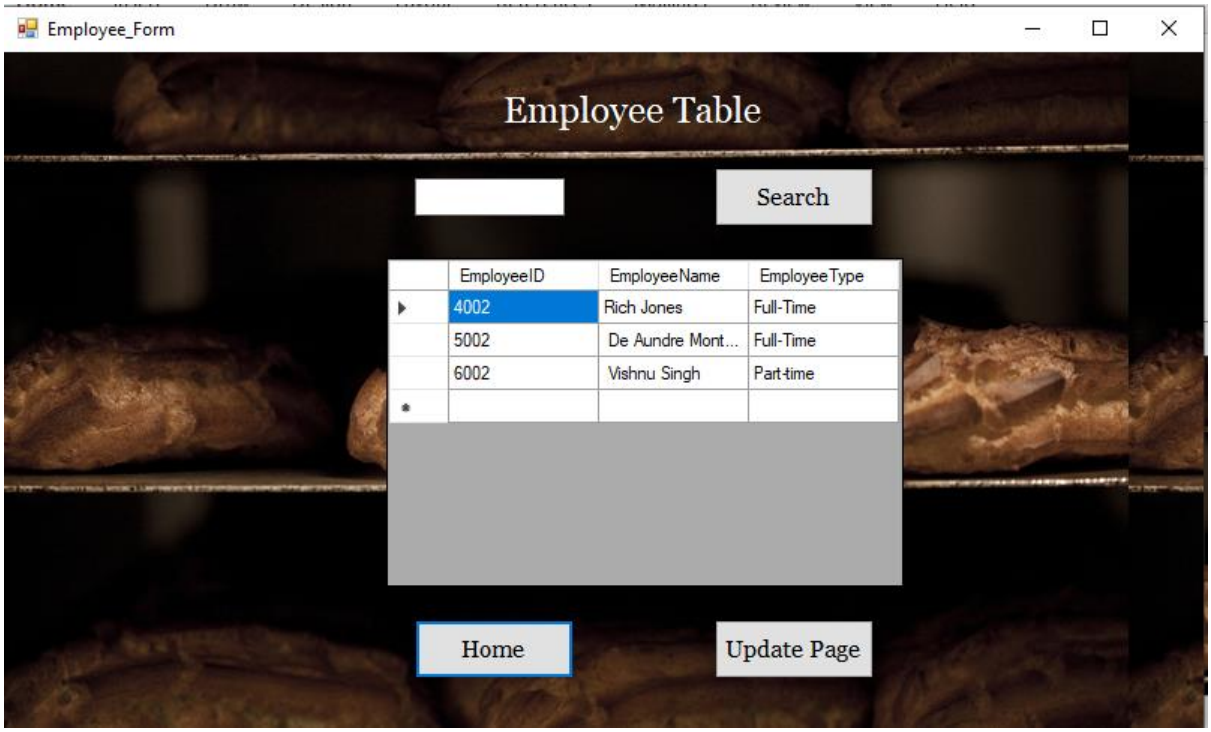
```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    Dim connection As New SqlConnection
    Dim command As New SqlCommand

    Try
        connection.ConnectionString = "Data Source=(LocalDb)\akhilmdb;Initial Catalog=Bakery;Integrated Security=True"
        connection.Open()
        command.Connection = connection
        command.CommandText = "insert into Orders values (' " & TextBox1.Text & "','" & TextBox2.Text & "','" & TextBox3.Text & "','" & TextBox4.Text & "')"
        command.ExecuteNonQuery()

        MsgBox("The Order was successfully added!", Title:="Order Form")

    Catch ex As Exception
        MessageBox.Show("Error while inserting record on table..." & ex.Message, "Insert Records")
    Finally
        connection.Close()
    End Try
End Sub
```

Code for Order Form

Employee:



Employee Form



Employee Table

```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    Dim connection As New SqlConnection
    Dim command As New SqlCommand

    Try
        connection.ConnectionString = "Data Source=(LocalDb)\akhilmdb;Initial Catalog=Bakery;Integrated Security=True"
        connection.Open()
        command.Connection = connection
        command.CommandText = "insert into Employee values (' " & TextBox1.Text & "','" & TextBox2.Text & "')"
        command.ExecuteNonQuery()

        MsgBox("The Employee was successfully added!", Title:="Employee Form")

    Catch ex As Exception
        MessageBox.Show("Error while inserting record on table..." & ex.Message, "Insert Records")
    Finally
        connection.Close()
    End Try
End Sub
```

Code For Order Form

Summary:

After testing the application it was observed that there is a problem but no problems was present in the code. So the problem that occurs with the primary key is within the database and how it increments the primary keys.

Solution:

What can be done to rectify the issues that arises with the IDs is to check the database information to see how the IDs are incrementing. There is no error apparent error in the code that performs the application itself so the problem arises in the Database Manager. Only fixing the problem there can rectify the issue.

Performance Testing

In this test we will be checking to see how long it takes to perform tasks in the application.

What will be tested	Observations
Login	Responds relatively quickly
Create (Customer, Pastry, Order, Employee)	Takes at least a 1-3 seconds before inserting info into the database
Retrieve (Customer, Pastry, Order, Employee)	Responds relatively quickly.
Update (Employee, Pastry)	Responds relatively quickly.
Delete (Customer, Pastry, Order, Employee)	Responds relatively quickly

Summary:

It was observed that for the insert (create) functionality, it took a little longer for the information to be processed. Some factors that may contribute to these problems are:

- Not enough RAM (Or RAM is being utilized elsewhere).
- The type of OS being used for the application.

References

No references were done for this document.