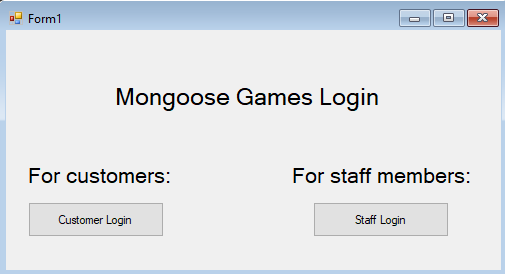
Software Development

Introduction: During this section of the project, I will provide screenshots of all of the forms in my project along with the corresponding bodies of code which relate to their functionality, proving the validity of my project. Furthermore, I will explain how data is entered into the system as well as showing how I have ensured my project to be user friendly and intuitive for both staff members and customers.

**Main Login form:**



This is the first thing any user sees when booting the system, the purpose of this part of the system is to allow the user to choose to login as a member of staff or as a customer, depending on which category they fall under.

They will then be directed to the login page of their choice, in order for them to access the system using their predefined credentials.

----------------------------------------------------------------------------------------------------------------

Public Class Form1

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

' Hides the MainLogin form

Hide()

' Shows the Member Login form when BtnMemberLogin is clicked

StaffLogin.Show()

End Sub

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

' Hides the MainLogin form

Hide()

' Shows the Customer Login form when BtnCustomerLogin is pressed

CustomerLogin.Show()

End Sub

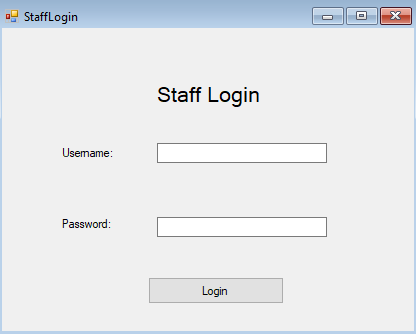
End Class

----------------------------------------------------------------------------------------------------------------

**Staff Login:**

This form allows a staff member to access the system using their valid credentials (provided by the boss, my client, Mongoose Hanks), if the wrong credentials are inputted, the text boxes are cleared and they are then informed of their 2 remaining attempts to access the system.

Validation has been added, ensuring that if any of the two text boxes are left blank, the user is informed that they need to enter both a username and password. This prevents the program trying to compute erroneous data which will cause crashes, it also aids the user in understanding what inputs are demanded.



----------------------------------------------------------------------------------------------------------------

' Allows database connections to be made

Imports System.Data.OleDb

Public Class StaffLogin

' Defines data reader which can be used anywhere in this form

Public dr As OleDbDataReader

' Defines attempts (used to store amount of attempts a user has made to access the system) which can be used anywhere in this form

Public attempts As Integer

Private Sub Button2\_Click(sender As Object, e As EventArgs) Handles BtnStaffLogin.Click

Try

' Check if username or password is empty and if so the user is informed that they must enter both before attempting to log in

If TxtStaffPassword.Text = "" Or TxtUsername.Text = "" Then

MessageBox.Show("Please complete the required fields..", "Authentication Error", MessageBoxButtons.OK, MessageBoxIcon.Error)

' Check if user exist in database

Else

' Connect to DB

Dim conn As New System.Data.OleDb.OleDbConnection()

' Connect with database on the current system

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\OneDrive\Desktop\MongooseGames.accdb"

' SQL statement that checks if entered credentials exist in the datanase

Dim sql As String = "SELECT \* FROM Staff WHERE StaffUser='" & TxtUsername.Text & "' AND StaffPass = '" & TxtStaffPassword.Text & "'"

' Allows sql statement to communicate with database

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

' Open connection between SQL command and database

sqlCom.Connection = conn

' Open database connection

conn.Open()

' Executes the SQL statement against the database

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' If they enter credentials that match to the database, they are then redirected to the customer menu

If sqlRead.Read() Then

MsgBox("Login successful")

' Hides current form

Me.Hide()

' Shows staff menu

StaffMenu.Show()

' if they expend their 3 attempts, they are then notified of this and the application is closed

ElseIf attempts = 2 Then

MessageBox.Show("You have expended your 3 attempts, system shutting down", "Security procedure")

Close()

Else

' If user enter wrong username and password combination

' Inform the user that they have entered incorrect credentials and that they now have x amount of attempts remaining

' Every time they get the credentials wrong, the attempts value increments by 1 to register that an attempt has been made

attempts += 1

MessageBox.Show("Username and Password do not match, you have expended " & attempts & "/3 retries", "Authentication Failure", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)

'Clear all fields

TxtStaffPassword.Text = ""

TxtUsername.Text = ""

'Focus on Username field (so that they can prompty retry without having to use their mouse)

TxtUsername.Focus()

' Closes the connection to the data source

conn.Close()

End If

End If

' Exception error is caught here, so that if there is an issue connecting to the database they are aware of this, instead of the program crashing

Catch ex As Exception

MessageBox.Show("Failed to connect to Database..", "Database Connection Error", MessageBoxButtons.OK, MessageBoxIcon.Error)

End Try

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**Staff Main Menu:**

****

This is the base of the system for the staff members, and allows them to access any function of the system that they want by clicking on the relevant button. The structure is intuitive and the labels are self-explanatory, to ensure that staff members will be able to use the system effectively and efficiently with little training required.

A return to main menu function has been included with each subsequent form that can be accessed from this menu, to allow staff members to carry out multiple tasks in one instance.

A logout button has also been implemented into the main menu, which ensures the changes in shifts are fluent as a staff member can log out of their account to allow another to access the system without having to restart the program, which saves a lot of time and effort. It can also be used as a safety feature, if the staff member is taking a break, they can log out to ensure that when they’re gone no unauthorised personnel can access the system and cause malicious damage.

----------------------------------------------------------------------------------------------------------------

Public Class StaffMenu

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

' Hides staff menu

Me.Hide()

' Accesses desired feature

QuotaCalculator.Show()

End Sub

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

' Hides staff menu

Me.Hide()

' Accesses desired feature

ResellCalculator.Show()

End Sub

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

' Hides staff menu

Me.Hide()

' Accesses desired feature

AddOrRemoveGame.Show()

End Sub

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

' Hides staff menu

Me.Hide()

' Accesses desired feature

ViewRecentGames.Show()

End Sub

Private Sub Button3\_Click(sender As Object, e As EventArgs) Handles BtnSearchGame.Click

' Hides staff menu

Me.Hide()

' Accesses desired feature

StaffSearchGames.Show()

End Sub

Private Sub Button7\_Click(sender As Object, e As EventArgs) Handles BtnUpdateDetails.Click

' Hides staff menu

Me.Hide()

' Accesses desired feature

UpdateCustDetails.Show()

End Sub

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

' Hides staff menu

Me.Hide()

' Accesses desired feature

ViewCustDetails.Show()

End Sub

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

' Hides staff menu

Me.Hide()

' Accesses desired feature

Form1.Show()

End Sub

Private Sub Button1\_Click\_1(sender As Object, e As EventArgs) Handles Button1.Click

' Hides staff menu

Me.Hide()

' Accesses desired feature

ViewCustomerRequests.Show()

End Sub

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

' Hides staff menu

Me.Hide()

' Accesses desired feature

RequestReply.Show()

End Sub

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

' Hides staff menu

Me.Hide()

' Accesses desired feature

ViewAllCustomers.Show()

End Sub

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

' Hides staff menu

Me.Hide()

' Accesses desired feature

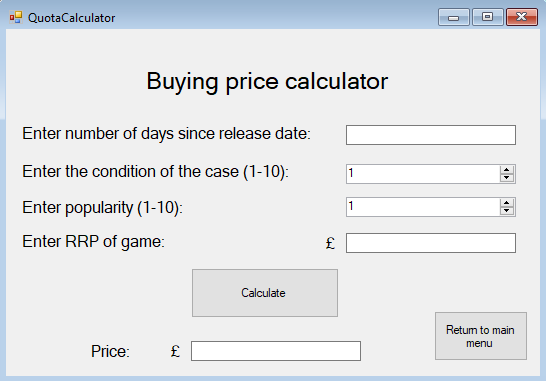
ViewAllGames.Show()

End Sub

End Class

---------------------------------------------------------------------------------------------------------------

**Buying price calculator:**

****

The buying price calculator (quote calculator) allows for a staff member to calculate the amount that they will pay a customer for their game based upon multiple factors that they will input in the corresponding text boxes such as the age of a game, condition of case, popularity and RRP.

The age of the game will be calculated by subtracting the release date from the current date.

The condition of the case will be determined by the staff member after a visual inspection.

The popularity will be determined by the staff member, after considering factors such as sales, reviews, scores and press coverage.

The RRP will be found out by searching how much the game was when it first game out, in brand new condition.

After all of these factors are inputted into the system, the algorithm will take all of the factors and produce a result and populate it into the price textbox.

----------------------------------------------------------------------------------------------------------------

Public Class QuotaCalculator

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

Try

' rules for games under a year old

' for every 1 condition value below 10, the price is subtracted by £1.5

' the less popular a game is, the more they are worth less than the RRP

If NumUpDownPopularity.Value >= 7 And txtGameAge.Text <= 365 Then

txtPrice.Text = (txtRRP.Text \* 0.75) - (10 - NumUpDownCondition.Value) \* 1.5

ElseIf NumUpDownPopularity.Value >= 5 And txtGameAge.Text <= 365 Then

txtPrice.Text = (txtRRP.Text \* 0.65) - (10 - NumUpDownCondition.Value) \* 1.5

ElseIf NumUpDownPopularity.Value >= 3 And txtGameAge.Text <= 365 Then

txtPrice.Text = (txtRRP.Text \* 0.5) - (10 - NumUpDownCondition.Value) \* 1.5

ElseIf NumUpDownPopularity.Value < 2 And txtGameAge.Text <= 365 Then

txtPrice.Text = (txtRRP.Text \* 0.3) - (10 - NumUpDownCondition.Value) \* 1.5

' now rules for games over a year old

' for every 1 condition value below 10, the price is subtracted by £1.5

' the less popular a game is, the more they are worth less than the RRP

ElseIf NumUpDownPopularity.Value >= 7 And txtGameAge.Text > 365 Then

txtPrice.Text = (txtRRP.Text \* 0.7) - (10 - NumUpDownCondition.Value) \* 1.5

ElseIf NumUpDownPopularity.Value >= 5 And txtGameAge.Text > 365 Then

txtPrice.Text = (txtRRP.Text \* 0.6) - (10 - NumUpDownCondition.Value) \* 1.5

ElseIf NumUpDownPopularity.Value >= 3 And txtGameAge.Text > 365 Then

txtPrice.Text = (txtRRP.Text \* 0.45) - (10 - NumUpDownCondition.Value) \* 1.5

ElseIf NumUpDownPopularity.Value < 2 And txtGameAge.Text > 365 Then

txtPrice.Text = (txtRRP.Text \* 0.25) - (10 - NumUpDownCondition.Value) \* 1.5

' If there is an error while trying to fit the inputted values into a category for a respective quote, the staff member is notified of this.

Else

MessageBox.Show("The game's characteristics do not match any field ", "Error while calculating quote")

End If

' If there is some sort of error, the staff member is made aware of it, without the program crashing.

Catch ex As Exception

MessageBox.Show("Please enter the required fields", "Error While compiling values")

End Try

End Sub

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

' Hide current form

Me.Hide()

' Show staff menu

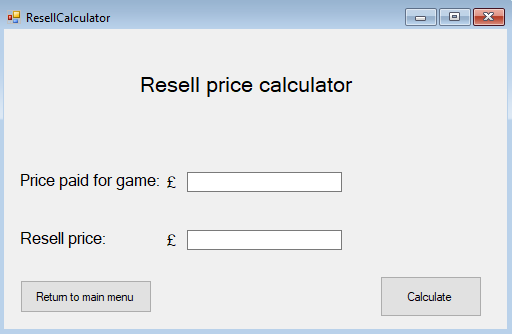
StaffMenu.Show()

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**Resell price calculator**

****

This part of the system allows a staff member to calculate the price that they will sell a game that has been bought from a customer for. This is important as it ensures the profitability of the business and eliminates human error and/or bias/subjective perspectives which will promote a healthy and thriving business. The staff member will enter the price they paid for the game and then the price that they should put the game back in stock for will be calculated for them. The algorithms increments the price paid by 25% for a worthy profit margin to be made.

Validation has been added ensuring that the value entered for Price Paid is numeric, that the Price Paid field isn’t left blank and that the Price Paid field value is not entered as a negative number. These forms of validations help ensure that staff members are guided into correct data entry as well as minimizing the impact erroneous data entry has on the system.

----------------------------------------------------------------------------------------------------------------

Public Class ResellCalculator

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

Try

' If they enter a price that is not numeric, they are informed that they need to enter a valid value

If IsNumeric(txtPricePaid.Text) = False Then

MessageBox.Show("Please ensure that the price paid for the game is a valid number, please try again.", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

txtPricePaid.Clear()

' If they leave the pricepaid textbox blank, they are then informed they are required to give a value

ElseIf txtPricePaid.Text = "" Then

MessageBox.Show("Please ensure that you enter the amount that was paid for the game.", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

txtPricePaid.Clear()

' If they enter a price that is negative, they are made aware that this is impossible and are encouraged to enter a valid value

ElseIf txtPricePaid.Text < 0 Then

MessageBox.Show("The price paid for the game cannot be negative, please try again.", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

txtPricePaid.Clear()

Else

' If the input has been successfully validated, the algorithm is then applied to it and used to populate the resell price textbox

txtResellPrice.Text = txtPricePaid.Text \* 1.25

End If

' If an unspecified error occurs, the system is kept from crashing by catching the error, the user is also informed of this occurence

Catch ex As Exception

MessageBox.Show("An unknown error has occured.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)

End Try

End Sub

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

' Hides current form

Me.Hide()

' Shows staff menu

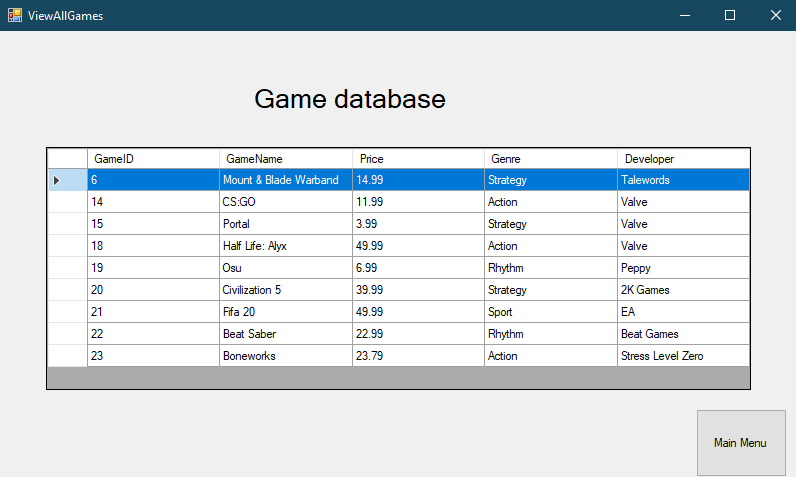
StaffMenu.Show()

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**View all available games on system**



This is a vital part of the system which allows the staff member to view all of the games that are currently available in store to be bought. This is a convenient part of the system as the staff member can use this feature to be able to verify that a game they have added to the system is now in the database and if a game that they have deleted has now been removed from the database.

Moreover, this feature can be used to confirm to customers that games are still in stock, if they are in a hurry or simply don’t have time to log onto a computer and will be given a prompt response on their query. This helps boost customer satisfaction as well as providing a major improvement from the prior system, where the staff member would then have to physically check whether the game was in the store or not.

Furthermore, this part of the system enables staff members to see the quantity of each game on the system, by ordering the records by GameName. This could prove useful when a customer wants to sell a game, and then after using this form, the staff member realises there are already 3 copies of that same game, and so decides to inform the customer that they are unable to purchase their game as it’s overstocked. This prevents a large amount of the same game being bought, which could then affect the business as the game would evidently have a low demand.

----------------------------------------------------------------------------------------------------------------

' Allows the database defined in the system configuration to be used

Imports System.Configuration

' Allows connection to be established with the connection string and the program

Imports System.Data.OleDb

Public Class ViewAllGames

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

' Hides current form

Me.Hide()

' Shows staff menu

StaffMenu.Show()

End Sub

Private Sub ViewAllGames\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

' Sets data source which will be the function returning the data table

GamesDataGridView.DataSource = GetGameInformation()

End Sub

' Creates function which returns the table to the staff member

Private Function GetGameInformation() As DataTable

' Defines the data table

Dim Games As New DataTable

' Fills data table with data from the database

Dim connString As String = ConfigurationManager.ConnectionStrings("Computing\_NEA.My.MySettings.MongooseGamesConnectionString").ConnectionString

' Allows sql statement to be performed on the database

Using conn As New OleDbConnection(connString)

' SQL Statement which retrieves all of the game data to allow the staff member to read it from the data grid view

Using cmd As New OleDbCommand("SELECT \* FROM Games", conn)

' Opens connection to the database

conn.Open()

' As SELECT statement is being used, reader is used, allows data to be read and sent to the form

Dim reader As OleDbDataReader = cmd.ExecuteReader()

' Loads the reader data into the form

Games.Load(reader)

End Using

End Using

' The function returns the data table

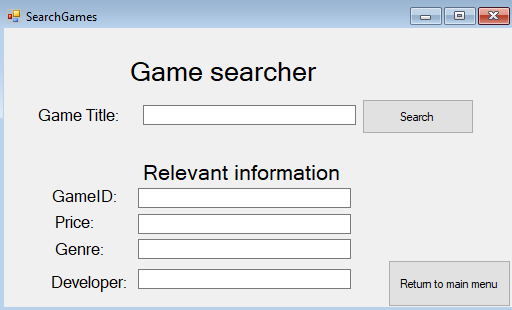
Return Games

End Function

End Class

----------------------------------------------------------------------------------------------------------------

**Search for a game on the system**

****

This feature allows the staff member to search for a certain game on the system by its name, where its corresponding information will then be given to them. They will need to input the game’s title, then click the button “Search”, they would then be notified if the game is found or not, and if it is, its information will be inputted in its corresponding text box, improving readability. This is important as it can be used to see if a game on a customer’s wishlist is in stock, and they could then subsequently report back to them, instead of having to search through the whole list of available games, or even worse, as how it used to be done, search through all of the notepads and paperwork, looking for the game. Overall, it’s an essential feature and a major improvement.

Validation has been added to ensure that the title of the game that is being searched is not left blank, to help the user understand that the game title is necessary for its information to be displayed.

----------------------------------------------------------------------------------------------------------------

Public Class StaffSearchGames

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

' If the user hasn't given a value for the game title of the game they wish to search for, they're informed that this must be filled out before their desired game is searched for

If txtGameTitle.Text = "" Then

MessageBox.Show("Please ensure that you've entered the title of the game you wish to search for", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

Else

' Connection is created

Dim conn As New System.Data.OleDb.OleDbConnection()

' Database location defined

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\OneDrive\Desktop\MongooseGames.accdb"

' SQL statement which returns all of the fields from the Games database where its name is equal to what the staff member searched

Dim sql As String = "SELECT GameID,GameName,Price,Genre,Developer FROM Games WHERE (GameName) = ('" & txtGameTitle.Text & "')"

' Allows the SQL command to communicate with the database

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

' Open SQL connection

sqlCom.Connection = conn

'Open Database Connection

conn.Open()

' Executes the SQL command to the database

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' If the game being searched for is found then this

If sqlRead.HasRows Then

' Reads through each field until all returned and then populates the relevant text boxes with each field value

While sqlRead.Read()

txtGameID.Text = sqlRead.Item("GameID")

txtGameTitle.Text = sqlRead.Item("GameName")

txtPrice.Text = sqlRead.Item("Price")

txtGenre.Text = sqlRead.Item("Genre")

txtDeveloper.Text = sqlRead.Item("Developer")

' User is notified of game being found

MessageBox.Show("Game found", "Searching of game successful")

End While

' Closes the read connection

sqlRead.Close()

Else

MessageBox.Show("No game called """ & txtGameTitle.Text & """" & " was found. Please try again", "Error while searching for game")

' All fields are cleared

txtGameID.Clear()

txtGameTitle.Clear()

txtPrice.Clear()

txtGenre.Clear()

txtDeveloper.Clear()

' Closes database connection

conn.Close()

End If

End If

End Sub

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

' Hides current form

Me.Hide()

' Shows staff menu

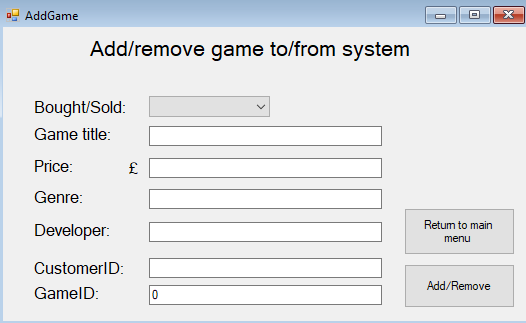
StaffMenu.Show()

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**Add/remove game from system**



This is one of the fundamental parts of the system, which enables staff members to add a game into the database after it’s bought from a customer, or remove a game from the database after it’s been sold to a customer. This will be used daily and is expected to be the most used function of the proposed system. The form is formatted intuitively, with the well-thought use of a bought/sold drop down box to avoid the implementation of two different functions on the system, which allows both of these tasks to be carried out through the same interface. Once the staff member enters all relative information, the database is subsequently manipulated by the system.

The “GameID” textbox will only have to be filled out when a game already exists in the database, and is being sold to a customer as it will have an assigned GameID, this would be explained to the staff member during training, or by hovering over the GameID label or related textbox, which through the use of a “ToolTip”, would inform them of this.

A staff member would be able to get all relevant information to be entered into the text box from the “View and search all available games” function for game information and “View customer details” function for the customerID, these will both be explained in detail later on in the project.

Validation has been added to ensure that fields are not left blank, price entered is a positive number, GameID is left as 0 if game is being bought (as game will not already exist in the database) as well as ensuring that when a game is being sold to a customer, that it actually exists in the system. All of these promote data integrity in the system and are vital to create a coherent and appropriate form that staff members utilise.

----------------------------------------------------------------------------------------------------------------

Public Class AddOrRemoveGame

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

Try

' If the staff member has left any essential field blank, they are notified that they must populate each field required

If txtTitle.Text = "" Or txtPrice.Text = "" Or txtGenre.Text = "" Or txtDeveloper.Text = "" Or txtCustomerID.Text = "" Then

MessageBox.Show("Please ensure that all fields are filled out", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

' If the staff member has entered a game that has a negative price, they are notified that this is impossible and they must review their input

ElseIf txtPrice.Text < 0 Or IsNumeric(txtPrice.Text) = False Then

MessageBox.Show("Please review the price value, it cannot be negative nor composed of characters", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

' If they want to sell a game, and have rightfully left the GameID value as 0, then this path is taken

ElseIf BoughtOrSold.Text = "Bought" And txtGameID.Text = 0 Then

' Create connection to the database

Dim conn As New System.Data.OleDb.OleDbConnection()

' Defines location of the database

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\OneDrive\Desktop\MongooseGames.accdb"

' SQL (Structured Query Lanugage) statement which inserts what the staff member input into the database in the corresponding table

Dim sql As String = "INSERT INTO Games (GameName,Price,Genre,Developer) VALUES ('" & txtTitle.Text & "', '" & txtPrice.Text & "', '" & txtGenre.Text & "', '" & txtDeveloper.Text & "')"

' Connects the SQL statement with the database

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

' Staff member is notified of their success of their required task.

MessageBox.Show("Game added to system", "Appending of game successful")

' Current form is hidden to allow the staff member to focus their attention on the AddToRecent form

Me.Hide()

' AddToRecent Form is shown, which asks the staff member if they wish to log the transaction into the system

AddToRecent.Show()

' Opens the connection between the SQL statement and the database

sqlCom.Connection = conn

' Opens the connection

conn.Open()

' Carry out the sql statement

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' Close the connection to the database

conn.Close()

' IF the staff member is buying a game from a customer and has populated the GameID text box they are infomred that this is not needed as it won't currently have a GameID since it's being introduced into the system

ElseIf BoughtOrSold.Text = "Bought" And txtGameID.Text <> 0 Then

MessageBox.Show("Please leave the GameID textbox as 0 since it is a game that is being introduced in the system and thus will not currently have a determined GameID", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

txtGameID.Text = "0"

' This is now for when the staff member wants to remove a game from the database due to it being sold

ElseIf BoughtOrSold.Text = "Sold" Then

Dim conn As New System.Data.OleDb.OleDbConnection()

' Define database location

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\OneDrive\Desktop\MongooseGames.accdb"

' SQL statement which ensures that the game that the staff member is selling to a customer actually exists in the system

Dim sqlValidation As String = "SELECT \* FROM Games WHERE GameID = " & txtGameID.Text & " AND GameName = '" & txtTitle.Text & "' AND Price = '" & txtPrice.Text & "' AND Genre = '" & txtGenre.Text & "' AND Developer = '" & txtDeveloper.Text & "'"

' Allows sql to communicate with database

Dim sqlComValidation As New System.Data.OleDb.OleDbCommand(sqlValidation)

' Opens the connection between the SQL statement and the database

sqlComValidation.Connection = conn

'Open Database Connection

conn.Open()

' Executes SQL statement against database

Dim sqlReadValidation As System.Data.OleDb.OleDbDataReader = sqlComValidation.ExecuteReader()

' If the game actually exists in the system, then the game is subsequently sold to the customer

If sqlReadValidation.Read() Then

' Create connection to the database

Dim conn1 As New System.Data.OleDb.OleDbConnection()

' Define location of the database

conn1.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\OneDrive\Desktop\MongooseGames.accdb"

' SQL (Structured Query Lanugage) statement which deletes the game that the staff member has referred to (using its ID) from the database

Dim sql As String = "DELETE \* FROM Games WHERE (GameID) = (" & txtGameID.Text & ")"

' Enables connection between SQL statement and database

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

' Staff member is notififed of their success of their required task.

MessageBox.Show("Game removed from system", "Removal of game successful")

' Current form is hidden to allow the staff member to focus their attention on the AddToRecent form

Me.Hide()

' AddToRecent Form is shown, which asks the staff member if they wish to log the transaction into the system

AddToRecent.Show()

' Opens the connection between the SQL statement and the database

sqlCom.Connection = conn1

' Opens database connection

conn1.Open()

' Carry out the sql statement

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' Close the connection to the database

conn.Close()

' If the GameID textbox is left blank, the staff member is notified that they are required to enter this seeing as they are selling a game to a customer and thus does have an existing gameID as it exists in the database currently

ElseIf txtGameID.Text = "" Or txtGameID.Text = 0 Then

MessageBox.Show("Please enter the GameID of the current game you wish to sell to the customer", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

Else

MessageBox.Show("The game you're trying to sell to a customer does not exist on the system, or you have inputted the game's details incorrectly. Please try again", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

End If

End If

' If any sort of error happens, this is caught to prevent the system from crashing, the staff member is then told to revise what they have input as it is likely they have entered erroneous data

Catch ex As Exception

MessageBox.Show("An error has occurred, please revise the inputs", "Data compilation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

End Try

End Sub

' When main menu button is clicked

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

' Hides current menu

Me.Hide()

' Displays the staff main menu

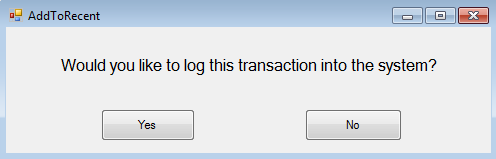
StaffMenu.Show()

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**Add to recent games**

****

After the staff member adds or removes a game, they will then be redirected to this form asking them if they wish to log this transaction into the system. If yes, it will be saved into the “RecentGames” table which stores all games that have been recently sold and bought from the system. If not, they will simply be redirected to the main menu.

----------------------------------------------------------------------------------------------------------------

Public Class AddToRecent

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

' Define connection

Dim conn As New System.Data.OleDb.OleDbConnection()

' Define location of database

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\OneDrive\Desktop\MongooseGames.accdb"

' Logs the transaction into the system

Dim sql As String = "INSERT INTO RecentGames (GameName,Price,Genre,Developer,BoughtOrSold,CustomerID,GameID) VALUES ('" & AddOrRemoveGame.txtTitle.Text & "', '" & AddOrRemoveGame.txtPrice.Text & "', '" & AddOrRemoveGame.txtGenre.Text & "', '" & AddOrRemoveGame.txtDeveloper.Text & "','" & AddOrRemoveGame.BoughtOrSold.Text & "', '" & AddOrRemoveGame.txtCustomerID.Text & "', '" & AddOrRemoveGame.txtGameID.Text & "')"

' Represents the SQL statement to be executed against the data base

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

' Staff member notified of successful logging of transaction

MessageBox.Show("Game transaction logged into the system, you will now be redirected to the main menu", "Recording of game transaction successful")

' Hides current form

Me.Hide()

' Shows staff menu

StaffMenu.Show()

' Opens connection of SQL statement and database

sqlCom.Connection = conn

' Opens connection to the database

conn.Open()

' Executes the SQL statement

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

End Sub

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

' Hides current form

Me.Hide()

' Shows staff menu

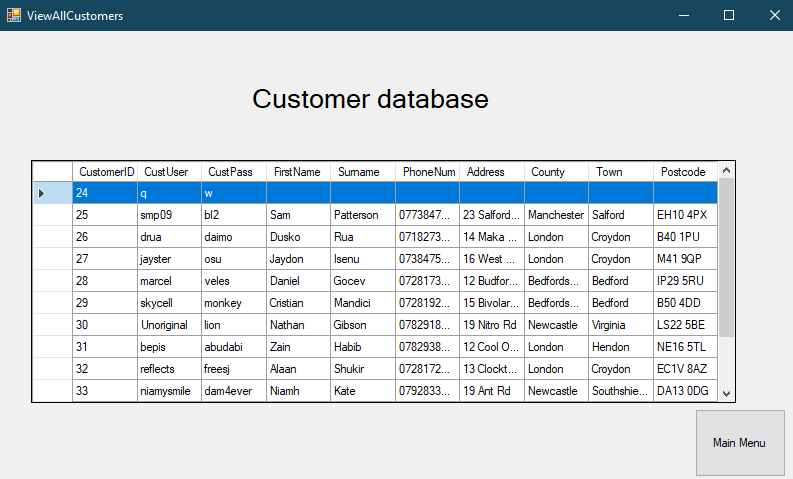
StaffMenu.Show()

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**Customer Information Viewer**

****

As a staff member, being able to view all of the customers registered to the system is essential. The information from the database is imported into the DataGridView which allows the staff member to familiarise themselves with everyone on the system and to also verify that changes that they have made have been applied to the system.

Moreover, if a customer should have issues logging in to their account, they could ask a staff member if their details are stored into the system, and once they verify their identity, they can be given new credentials. This provides exceptional customer service and generally enables the staff member to be able to view what is stored in the database directly.

----------------------------------------------------------------------------------------------------------------

' Allows the database defined in the system configuration to be used

Imports System.Configuration

' Allows connection to be established with the connection string and the program

Imports System.Data.OleDb

Public Class ViewAllCustomers

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

' Hides current form

Me.Hide()

' Shows staff menu form

StaffMenu.Show()

End Sub

Private Sub ViewAllCustomers\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

' Sets data source which will be the function returning the data table

CustomersDataGridView.DataSource = GetCustomerInformation()

End Sub

' Creates function which returns the table to the staff member

Private Function GetCustomerInformation() As DataTable

' Defines the data table

Dim Customers As New DataTable

' Fills data table with data from the database

Dim connString As String = ConfigurationManager.ConnectionStrings("Computing\_NEA.My.MySettings.MongooseGamesConnectionString").ConnectionString

' Allows sql statement to be performed on the database

Using conn As New OleDbConnection(connString)

' SQL Statement which shows all of the data to the staff member in the table

Using cmd As New OleDbCommand("SELECT \* FROM Customers", conn)

' Opens connection to the database

conn.Open()

' As SELECT statement is being used, reader is used, allows data to be read and sent to the form

Dim reader As OleDbDataReader = cmd.ExecuteReader()

' Loads the reader data into the form

Customers.Load(reader)

End Using

End Using

' The function returns the data table

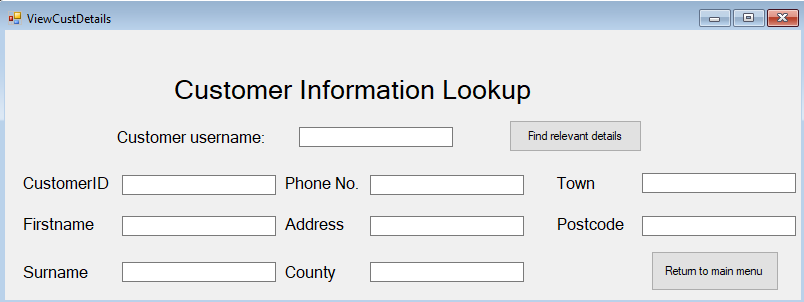
Return Customers

End Function

End Class

----------------------------------------------------------------------------------------------------------------

**Customer Information Lookup**



This form provides the staff members with the possibility of retrieving all of a certain customer’s details by providing their unique username. This will be used when a staff member buys a game from a customer, where they will need to know their customerID, they will simply ask the customer for their username and they’ll be able to retrieve their customerID from this form. Moreover, they will also be able to retrieve details such as their phone number and address, which will allow the staff member to contact the customer externally if need be.

Validation has been added to ensure that the “Customer username” field is populated - to prevent the system from trying to compute invalid data and to ensure that staff are informed of input requirements.

----------------------------------------------------------------------------------------------------------------

Public Class ViewCustDetails

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

' Hides current form

Me.Hide()

' Shows staff menu

StaffMenu.Show()

End Sub

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

' If the customer username field is left blank, the staff member is notified that they must enter a valid username

If TxtCustUser.Text = "" Then

MessageBox.Show("Please do not leave the customer username field blank", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

Else

' Define connection

Dim conn As New System.Data.OleDb.OleDbConnection()

' Define database location for connection

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\OneDrive\Desktop\MongooseGames.accdb"

' Retrieves the record of the customer with the username that the staff member inputs

Dim sql As String = "SELECT CustomerID,FirstName,Surname,PhoneNum,Address,County,Town,Postcode FROM Customers WHERE (CustUser) = ('" & TxtCustUser.Text & "')"

' Allows sql command to communicate with the data source

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

' Opens connection between SQL and database

sqlCom.Connection = conn

' Opens database connection

conn.Open()

' Provides a way of reading the data rows from the database

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' Executes following code if there are records found

If sqlRead.HasRows Then

' This while loop populates the relevant textbox with the corresponding information from the database until finished.

While sqlRead.Read()

TxtCustID.Text = sqlRead.Item("CustomerID")

TxtCustName.Text = sqlRead.Item("FirstName")

TxtCustSurname.Text = sqlRead.Item("Surname")

TxtCustPhone.Text = sqlRead.Item("PhoneNum")

TxtCustAddress.Text = sqlRead.Item("Address")

TxtCustCounty.Text = sqlRead.Item("County")

TxtCustTown.Text = sqlRead.Item("Town")

TxtCustPostcode.Text = sqlRead.Item("Postcode")

' Informs the staff member of their success of searching for a customer

MessageBox.Show("Customer found", "Searching of customer details successful")

End While

' Close connection between sql and database

sqlRead.Close()

Else

' If there are no customers with the username that the staff member inputted in the system they are informed of this and prompted to try again.

MessageBox.Show("No customer with the username of """ & TxtCustUser.Text & """ was found. Please try again", "Error while searching for customer details")

' The customer username field is then cleared to faciliate the re-population of the textbox

TxtCustUser.Clear()

' Connection of the database is closed

conn.Close()

End If

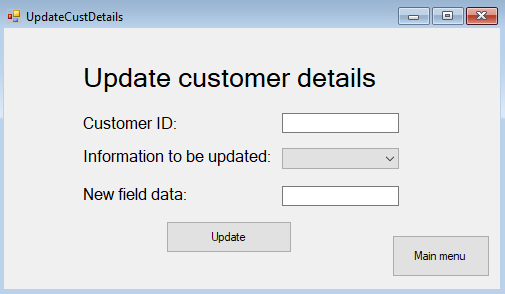
End If

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**Update customer Information:**



This form allows staff members to update a customer’s record by entering their CustomerID and then the data they would like to replace the certain field with. They are able to select, using the drop down menu what field they would like to update of the customer.

This form would be used when a customer would approach a staff member, informing them of a change in their information e.g. Address (moved house), phone number (bought a new phone) or surname (got married). This complies with GDPR rules, as it allows customers to request the data that is stored about them is to be changed in order to be accurate.

The UI uses very little text boxes and buttons i.e. avoids the use of using multiple text boxes that correspond to the different fields which can be edited separately, reducing confusion and clutter. This makes the interface a lot more digestible which can lead to more efficient and effective use of the program as more often than not, only a few fields will have to be updated rather than the entire customer record.

Validation has been added to ensure that only a value that is pre-existing in the “Information to be updated” drop down box can be chosen, that all fields are populated, that if the first name, surname, county or town of a customer is being changed, the new value cannot have digits, that a new phone number should consist of 11 digits and that a postcode is either 7 or 8 characters long (as per UK format) and contains a space (should be in the middle). These all guarantee data integrity to a certain extent as well as informing the staff member of the expectations of data entry

----------------------------------------------------------------------------------------------------------------

' Allows regex expressions to be utilised which can check if an input contains any letters or numbers to aid validation

Imports System.Text.RegularExpressions

Public Class UpdateCustDetails

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

' If the customerID of the customer they wish to edit their record is empty, or the desired field to be changed is left empty, or the new data to change the field value to is left empty,

If txtCustID.Text = "" Or FieldName.SelectedItem = "" Or txtNewData.Text = "" Then

' The staff member is informed that all text boxes require an input before they can update a customer's details

MessageBox.Show("Please ensure that all text boxes are filled before attempting to update a customer's field value", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

' If the staff member is trying to change a customer's field value that should only be composed of letters, to a value that contains digits,

ElseIf FieldName.SelectedItem = "FirstName" And Regex.Match(txtNewData.Text, "\d").Success Or FieldName.SelectedItem = "Surname" And Regex.Match(txtNewData.Text, "\d").Success Or FieldName.SelectedItem = "County" And Regex.Match(txtNewData.Text, "\d").Success Or FieldName.SelectedItem = "Town" And Regex.Match(txtNewData.Text, "\d").Success Then

' The staff member is informed that they cannot change that customer's field if it contains a digit

MessageBox.Show("You cannot change a customer's " & FieldName.SelectedItem & " to a value that contains digits ", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

txtNewData.Clear()

' If the staff member is trying to change the phone number of a customer, it checks if the phone number entered is less/more than 11 digits long

ElseIf FieldName.SelectedItem = "PhoneNum" And txtNewData.Text.Length <> 11 Or FieldName.SelectedItem = "PhoneNum" And IsNumeric(txtNewData.Text) = False Then

' If so, they are informed that they need to ensure their phone number exactly 11 characters long

MessageBox.Show("Please ensure that the phone number is 11 digits long", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

' Clears the text box to faciliate re-entering of data

txtNewData.Clear()

' If a staff member is trying to change the Postcode of a customer

' It is checked if its 7 or 8 characters long (UK format inc. space)

ElseIf FieldName.SelectedItem = "Postcode" And txtNewData.Text.Length < 7 Or FieldName.SelectedItem = "Postcode" And txtNewData.Text.Length > 8 Or FieldName.SelectedItem = "Postcode" And txtNewData.Text.Contains(" ") = False Then

' If not, they are informed that they need to enter a valid postcode

MessageBox.Show("Please ensure that your postcode is either 7 or 8 characters long, including a space in between the two parts ", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

Else

' If the inputs have been validated as per rules above, the customer details can be updated

' Create connection to the database

Dim conn As New System.Data.OleDb.OleDbConnection()

' Defines location of the database

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\OneDrive\Desktop\MongooseGames.accdb"

' SQL (Structured Query Lanugage) statement which changes the value of the desired field to the desired value in the record of the desired customer

Dim sql As String = "UPDATE Customers SET " & FieldName.SelectedItem & " = """ & txtNewData.Text & """ WHERE CustomerID = " & txtCustID.Text & ""

' Executes the statement against the database

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

' Connects the sql statement to the database

sqlCom.Connection = conn

' Opens the connection

conn.Open()

' Staff member is notified of their success of their required task.

MessageBox.Show("Field value updated", "Database record updation successful")

' Carry out the sql statement

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' Close the connection to the database

conn.Close()

End If

End Sub

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

' Hides the current form

Me.Hide()

' Shows the staff menu

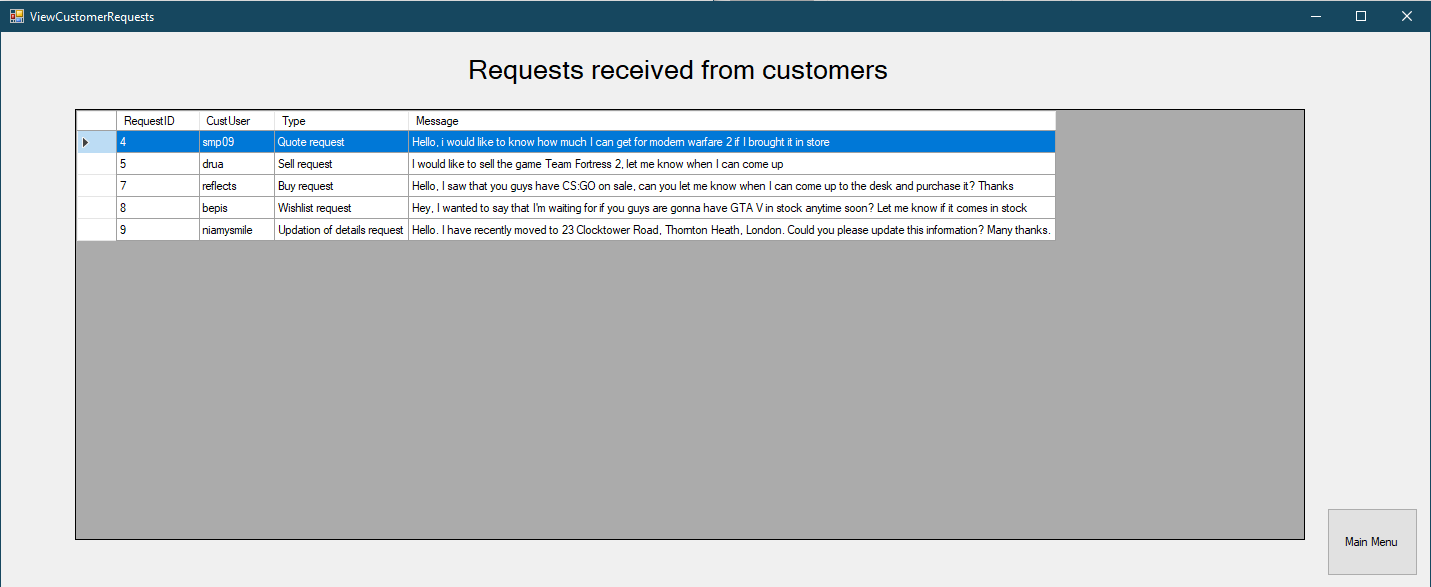
StaffMenu.Show()

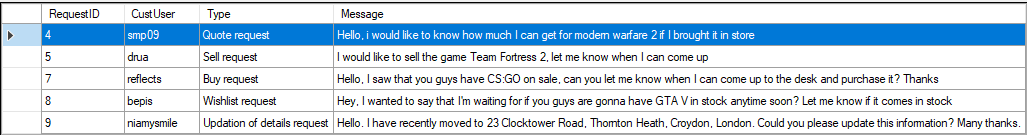
End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**View customer requests:**

****

****

This is a vital part of the program, which enables a plethora of actions to be completed. It allows customers to sell and buy games, as well as requesting games to be brought in stock. Crucially, it allows customers to request that the details stored about them are updated, which is necessary in order for Mongoose Games to comply with GDPR rules, that the data should always be accurate - to the customer’s liking.

As soon as the form opens, the information from the table is loaded into the DataViewGrid. I have ensured that the CustUser and Type of request is present, to enable the staff member to know what they’re requesting and then subsequently communicate back to the customer, informing them of an advance on their request. I have also changed some formatting options to remove the scroll bar from the form, as I believe it will cause delays when reading customer requests. I have also made it so when a staff member selects a record, that the whole record is highlighted not just one field, to enable the staff member to read the message more easily. In addition, this allows the entire record to be copied at once, which will prove very useful for when the staff member wants to reply to the user’s request, and will need the user’s username as well as the RequestID which they’ll be able to paste in with ease. Moreover, it would help them keep track of their request type and relevant message while they are writing a response.

The requests can also be ordered by any field e.g. RequestID this could be useful if the staff member wants to prioritise handling the requests that were sent first, to reduce waiting times for customers. Sorting by the Type could also be used if the staff member wishes to prioritise selling/buying requests rather than wishlist or updating of information requests as although they’re important, the economic transactions are valued higher to ensure maximum business profitability.

----------------------------------------------------------------------------------------------------------------

' Allows the database defined in the system configuration to be used

Imports System.Configuration

' Allows connection to be established with the connection string and the program

Imports System.Data.OleDb

Public Class ViewCustomerRequests

Private Sub ViewCustomerRequests\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

' Sets data source which will be the function returning the data table

CustomerRequestDataGridView.DataSource = GetCustomerRequests()

End Sub

' Creates function which returns the table to the staff member

Private Function GetCustomerRequests() As DataTable

' Defines the data table

Dim CustRequests As New DataTable

' Fills data table with data from the database

Dim connString As String = ConfigurationManager.ConnectionStrings("Computing\_NEA.My.MySettings.MongooseGamesConnectionString").ConnectionString

' Allows sql statement to be performed on the database

Using conn As New OleDbConnection(connString)

' SQL Statement which shows all of the data to the staff member in the table

Using cmd As New OleDbCommand("SELECT \* FROM CustRequests", conn)

' Opens connection to the database

conn.Open()

' As SELECT statement is being used, reader is used, allows data to be read and sent to the form

Dim reader As OleDbDataReader = cmd.ExecuteReader()

' Loads the reader data into the form

CustRequests.Load(reader)

End Using

End Using

' The function returns the data table

Return CustRequests

End Function

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

' Hides current form

Me.Hide()

' Shows staff menu

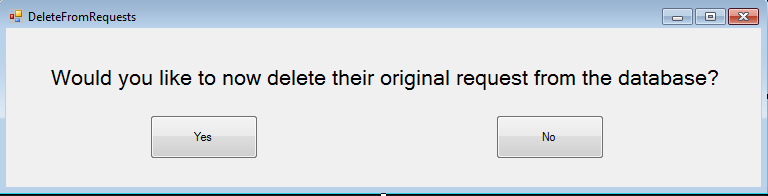
StaffMenu.Show()

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**Subsequent deletion of original customer request**



This is the form that will appear after the staff member has replied to a user request. If they choose to delete the original request (which will be done usually to avoid two staff members replying to one request and to reduce clutter) it will be removed from the database and then they will be redirected to the main menu. If not, they will simply be redirected to the main menu. This is useful as it could’ve been a wishlist request and the staff member would might want to keep it in the inbox so they don’t forget the customer has made that request, to then later on inform them of any updates relating to the availability of their game.

----------------------------------------------------------------------------------------------------------------

Public Class DeleteFromRequests

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

' Create connection to the database

Dim conn As New System.Data.OleDb.OleDbConnection()

' Define location of the database

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\OneDrive\Desktop\MongooseGames.accdb"

' SQL (Structured Query Lanugage) statement which deletes the request that the staff member has just dealt with from the system

Dim sql As String = "DELETE \* FROM CustRequests WHERE (RequestID) = (" & RequestReply.txtRequestID.Text & ")"

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

' Staff member is notififed of their success of their required task.

MessageBox.Show("Request removed from system, you will now be redirected to the main menu screen", "Removal of request successful")

' Current form is hidden

Me.Hide()

' redirected to main menu

StaffMenu.Show()

' Connect SQL to database

sqlCom.Connection = conn

' Open connection to database

conn.Open()

' Carry out the sql statement

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' Close the connection to the database

conn.Close()

End Sub

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

' Hides current form

Me.Hide()

' Shows menu

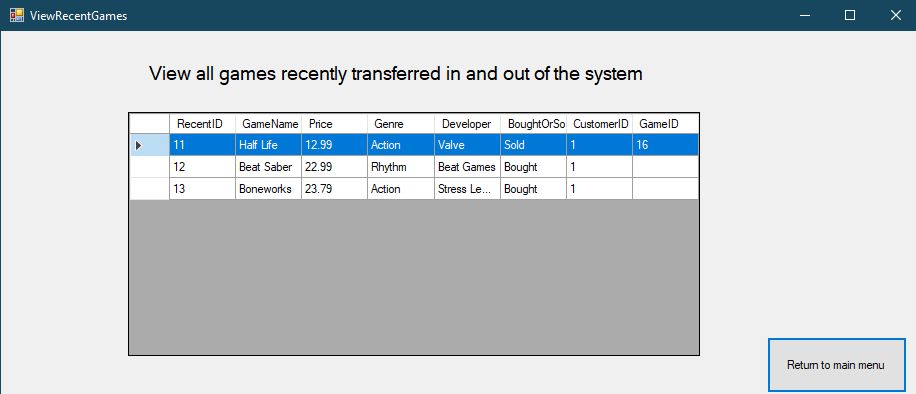
StaffMenu.Show()

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**View all recent transactions from the system**

****

Viewing all of the recently sold/bought games is an essential feature as it allows Mongoose Games to keep track of what games are being entered and removed from the database which is important as it could be used to identify customer purchase patterns.

As with the ViewCustomerRequests form, the layout and design is clear and the staff members are able to easily identify the details of each record. This feature could also be useful as it could provide a means of verifying that a customer has bought/sold a game, should this proof be needed.

----------------------------------------------------------------------------------------------------------------

' Allows the database defined in the system configuration to be used

Imports System.Configuration

' Allows connection to be established with the connection string and the program

Imports System.Data.OleDb

Public Class ViewRecentGames

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

' Hides current form

Me.Hide()

' Shows staff menu

StaffMenu.Show()

End Sub

Private Sub ViewRecentGames\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

' Sets data source which will be the function returning the data table

RecentGamesDataGridView.DataSource = GetRecentGames()

End Sub

' Creates function which returns the table to the staff member

Private Function GetRecentGames() As DataTable

' Defines the data table

Dim RecentGames As New DataTable

' Fills data table with data from the database

Dim connString As String = ConfigurationManager.ConnectionStrings("Computing\_NEA.My.MySettings.MongooseGamesConnectionString").ConnectionString

' Allows sql statement to be performed on the database

Using conn As New OleDbConnection(connString)

' SQL Statement which shows all of the data to the staff member in the table

Using cmd As New OleDbCommand("SELECT \* FROM RecentGames", conn)

' Opens connection to the database

conn.Open()

' As SELECT statement is being used, reader is used, allows data to be read and sent to the form

Dim reader As OleDbDataReader = cmd.ExecuteReader()

' Loads the reader data into the form

RecentGames.Load(reader)

End Using

End Using

' The function returns the data table

Return RecentGames

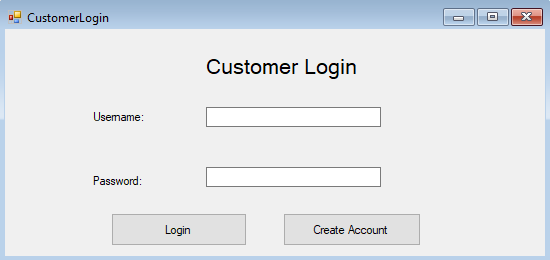
End Function

End Class

----------------------------------------------------------------------------------------------------------------

**Customer system features**

**Customer Login:**

****

This form is accessed after clicking the “Customer Login” button on the main form and allows a customer to access the system using their valid credentials (created prior using Create Account function shown below), if the wrong credentials are inputted, the text boxes are cleared and they are then informed of their 2 remaining tries to access the system.

Validation has been implemented as with the staff login, ensuring that both the username and password fields are populated before a customer can attempt their login. This prevents crashes and allows a customer to understand that both fields must be filled.

----------------------------------------------------------------------------------------------------------------

Public Class CustomerLogin

' Defines variable which holds the amount of attempts a user has made to login

Dim attempts As Integer

Private Sub Button2\_Click(sender As Object, e As EventArgs) Handles Button2.Click

' Check if username or password is empty

Try

' If password or username fields are empty, error is produced

If txtPassword.Text = "" Or txtUsername.Text = "" Then

MessageBox.Show("Please complete the required fields..", "Authentication Error", MessageBoxButtons.OK, MessageBoxIcon.Error)

Else

' Creates connection to database

Dim conn As New System.Data.OleDb.OleDbConnection()

' Define database connection

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\OneDrive\Desktop\MongooseGames.accdb"

' Searches for a customer which has the same username and password that was entered by the user

Dim sql As String = "SELECT \* FROM Customers WHERE CustUser='" & txtUsername.Text & "' AND CustPass = '" & txtPassword.Text & "'"

' Creates connection between

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

' Connects SQL statement with database

sqlCom.Connection = conn

'Open Database Connection

conn.Open()

' Executes SQL statement

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' If they enter the current combination of credentials, they are then redirected to the customer menu

If sqlRead.Read() Then

MsgBox("Login successful")

' Hides current form

Me.Hide()

' Shows menu for customers

CustomerMenu.Show()

' if they expend their 3 attempts, they are then notified of this and the application is closed

ElseIf attempts = 2 Then

MessageBox.Show("You have expended your 3 attempts, system shutting down", "Security procedure")

Close()

Else

' If user enter wrong username and password combination

' Inform them that their login attempt failed, and now they have 1 less attempt

' Every time they get the credentials wrong, the attempts value increments by 1 to register that an attempt has been made

attempts += 1

MessageBox.Show("Username and Password do not match, you have expended " & attempts & "/3 retries", "Authentication Failure", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)

'Clear all fields

txtPassword.Text = ""

txtUsername.Text = ""

'Focus on Username field (so that they can prompty retry without having to use their mouse)

txtUsername.Focus()

' Closes the connection to the data source

conn.Close()

End If

End If

' Exception caught here, so that if there is an issue connecting to the database they are aware of this, instead of the program crashing

Catch ex As Exception

MessageBox.Show("Failed to connect to Database..", "Database Connection Error", MessageBoxButtons.OK, MessageBoxIcon.Error)

End Try

End Sub

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

' if account create button is pressed the form is hidden

Me.Hide()

' Then the account creation screen is shown

AccCrea.Show()

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**Customer account creation**

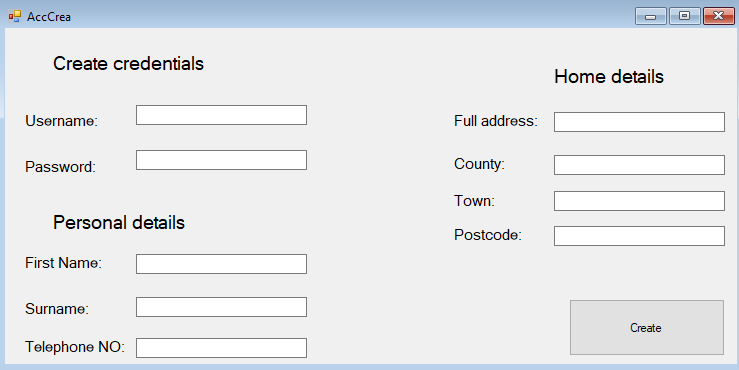
If it’s the customer’s first time accessing the system, they will need to create an account first along with their corresponding information, which will then grant their future access to the system. This is then handled by the form below which is presented when a customer presses the “Create account” button on the customer login (form shown above)

Validation has also been added to ensure that two people with the same username can’t exist, this is done by checking, before the account is saved to the database, if the desired username already exists in the database, if so they are notified of this and have to change the username, if not, it will go through normally. This is done in order to prevent any clashes of data and to ensure that messages are sent to the correct recipients.

Moreover, it is ensured that all fields are filled by the user, that the username and password are not the same and are both over 5 characters, that their telephone number is 11 digits, that their postcode is 7/8 characters (as per UK format) and contains a space (encouraged to place this in the middle) and that their first name, surname, county and town do not contain any numbers.

These extensive validation methods have been used for a variety of reasons. Ensuring that the username/password of a user is over 5 characters, and that they are both different, increase the security of their account - making them less prone to brute force attacks. Ensuring that their phone number and postcode are correct format promote data integrity by reducing the effect of erroneous data input. Ensuring that their first name, surname, county or town inputted do not contain any digits guarantees a certain sensible boundary for this data, although due to the many possibilities of these fields it’s difficult to produce more accurate validation.

Once all of the required information has been entered and validated, they are then able to create their account and are redirected back to the login screen to use their newly created credentials.



----------------------------------------------------------------------------------------------------------------

' Allows regex commands to be used to identify characters or numbers in inputs to aid validation

Imports System.Text.RegularExpressions

Public Class AccCrea

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

' Check if any field is empty and if any of them are empty

If txtPassword.Text = "" Or txtUsername.Text = "" Or txtFirstName.Text = "" Or txtSurname.Text = "" Or txtTelephone.Text = "" Or txtAddress.Text = "" Or txtCounty.Text = "" Or txtTown.Text = "" Or txtPostcode.Text = "" Then

' If so, then the user is notified of this and they are to then fill them in

MessageBox.Show("Please complete all required fields...", "Authentication Error", MessageBoxButtons.OK, MessageBoxIcon.Error)

' Checks if the password is the same as the username

ElseIf txtUsername.Text = txtPassword.Text Then

' If so, user is then to ensure that their username differs from their password

MessageBox.Show("Please ensure that your username is different from your password", "Security warning", MessageBoxButtons.OK, MessageBoxIcon.Error)

' Clears the text boxes to faciliate re-entering of data

txtUsername.Clear()

txtPassword.Clear()

' Checks if the length of the customer's username or password is less than 5

ElseIf txtUsername.Text.Length < 5 Or txtPassword.Text.Length < 5 Then

' If so, they are prompted to make their username/password longer than 5 characters so that their credentials are more sophisticated

MessageBox.Show("Please ensure that your username/password is longer than 5 characters", "Security warning", MessageBoxButtons.OK, MessageBoxIcon.Error)

' Clears the text boxes to faciliate re-entering of data

txtUsername.Clear()

txtPassword.Clear()

' Checks if the phone number entered is less/more than 11 digits long

ElseIf txtTelephone.Text.Length <> 11 Or IsNumeric(txtTelephone.Text) = False Then

' If so, they are informed that they need to ensure their phone number exactly 11 characters long

MessageBox.Show("Please ensure that your phone number is 11 digits long", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

' Clears the text box to faciliate re-entering of data

txtTelephone.Clear()

' Checks if the postcode entered is 7 or 8 characters long and if it contains a space

ElseIf txtPostcode.Text.Length < 7 Or txtPostcode.Text.Length > 8 Or txtPostcode.Text.Contains(" ") = False Then

' If not, they are informed that they need to enter a valid postcode

MessageBox.Show("Please ensure that your postcode is either 7 or 8 characters long, including a space in between the two parts ", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

' Checks if any field which shouldn't be composed of digits e.g. firstname contains any digits

ElseIf Regex.Match(txtFirstName.Text, "\d").Success Or Regex.Match(txtSurname.Text, "\d").Success Or Regex.Match(txtCounty.Text, "\d").Success Or Regex.Match(txtTown.Text, "\d").Success Then

' If so, they are prompted to reevaluate this

MessageBox.Show("Please ensure that there are no numbers in fields which don't require them", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

Else

' Define new connection

Dim conn As New System.Data.OleDb.OleDbConnection()

' Define database location

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\OneDrive\Desktop\MongooseGames.accdb"

' Find records that have the same username as someone else on the system

Dim sqlValidation As String = "SELECT \* FROM Customers WHERE CustUser='" & txtUsername.Text & "'"

' Allows sql to communicate with database

Dim sqlComValidation As New System.Data.OleDb.OleDbCommand(sqlValidation)

' Connect SQL statement to database

sqlComValidation.Connection = conn

'Open Database Connection

conn.Open()

' Executes SQL statement against database

Dim sqlReadValidation As System.Data.OleDb.OleDbDataReader = sqlComValidation.ExecuteReader()

' If username is already in use

If sqlReadValidation.Read() Then

' User is notified that username is already in use and they must change it

MsgBox("Username is already in use, please try another one")

Else

' Inserts all of the information that the user inputs into the database

Dim sqlInsert As String = "INSERT INTO customers (CustUser,CustPass,FirstName,Surname,PhoneNum,Address,County,Town,Postcode) VALUES ('" & txtUsername.Text & "', '" & txtPassword.Text & "', '" & txtFirstName.Text & "', '" & txtSurname.Text & "', '" & txtTelephone.Text & "', '" & txtAddress.Text & "', '" & txtCounty.Text & "', '" & txtTown.Text & "', '" & txtPostcode.Text.ToUpper & "')"

' Allows SQL statement to communicate with databasde

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sqlInsert)

' They are then informed of their account creation

MessageBox.Show("Account Created, and details saved, you will now be redirected to the login screen", "Account creation successful")

' Current form hidden

Me.Hide()

' Directed to the customer login form to now use their newly created credentials to gain access to the system

CustomerLogin.Show()

'Connects SQL statement with database

sqlCom.Connection = conn

' Executes SQL statement

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' Closes connection

conn.Close()

End If

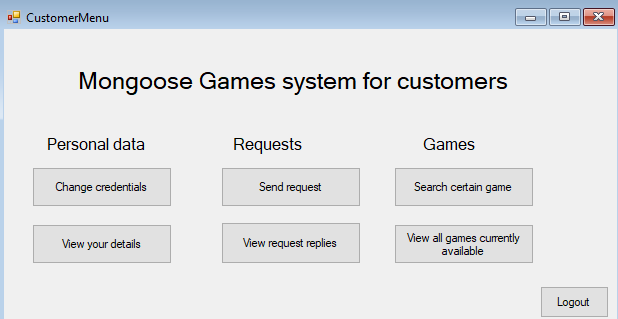
End If

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**Customer Main Menu:**



Once a customer has logged in successfully, they are redirected to the main menu, which is the hub of all the functionalities they will be able to carry out as a customer.

Depending on what they wish to do, they can click the corresponding button to be redirected to that feature.

The format and buttons are clear, with bold subheadings in order to help the customer locate their desired functionality

The most frequently used feature will be for customers to send requests to staff members and is subsequently placed in the middle of the form to reduce confusion and increase accessibility. Moreover, a ToolTip has been added for the button, as it’s fairly ambiguous, allowing the customer to understand what can be done with this function. This feature encompasses a plethora of requests including; buy game request, sell game request, quote request, wishlist request and updating of details request.

----------------------------------------------------------------------------------------------------------------

Public Class StaffMenu

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

' Hides staff menu

Me.Hide()

' Accesses desired feature

QuotaCalculator.Show()

End Sub

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

' Hides staff menu

Me.Hide()

' Accesses desired feature

ResellCalculator.Show()

End Sub

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

' Hides staff menu

Me.Hide()

' Accesses desired feature

AddOrRemoveGame.Show()

End Sub

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

' Hides staff menu

Me.Hide()

' Accesses desired feature

ViewRecentGames.Show()

End Sub

Private Sub Button3\_Click(sender As Object, e As EventArgs) Handles BtnSearchGame.Click

' Hides staff menu

Me.Hide()

' Accesses desired feature

StaffSearchGames.Show()

End Sub

Private Sub Button7\_Click(sender As Object, e As EventArgs) Handles BtnUpdateDetails.Click

' Hides staff menu

Me.Hide()

' Accesses desired feature

UpdateCustDetails.Show()

End Sub

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

' Hides staff menu

Me.Hide()

' Accesses desired feature

ViewCustDetails.Show()

End Sub

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

' Hides staff menu

Me.Hide()

' Accesses desired feature

Form1.Show()

End Sub

Private Sub Button1\_Click\_1(sender As Object, e As EventArgs) Handles Button1.Click

' Hides staff menu

Me.Hide()

' Accesses desired feature

ViewCustomerRequests.Show()

End Sub

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

' Hides staff menu

Me.Hide()

' Accesses desired feature

RequestReply.Show()

End Sub

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

' Hides staff menu

Me.Hide()

' Accesses desired feature

ViewAllCustomers.Show()

End Sub

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

' Hides staff menu

Me.Hide()

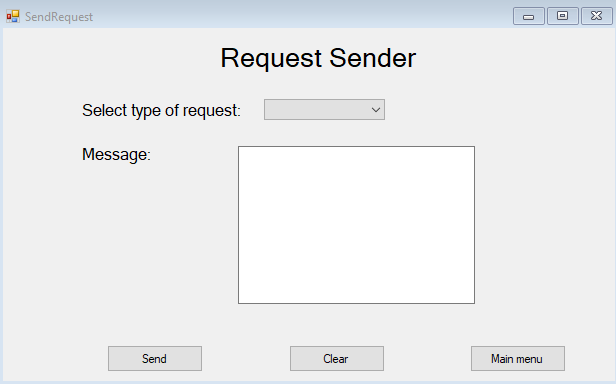
' Accesses desired feature

ViewAllGames.Show()

End Sub

End Class----------------------------------------------------------------------------------------------------------------

**Send request:**



This form allows the customers to make a range of requests for the staff member to then follow through on. They are required to select what type of request it will be (buy,sell,quote,wishlist,updating of details) along with a corresponding message to compliment their desired request. This will then be sent to the database, allowing a staff member to read and then reply to the request. This feature is essential as it aids the customers to be able to carry out fundamental tasks within just one form such as buying or selling a game. They will then receive a response from the staff member which they will then view under the “View Request Updates” form, shown below.

Overall, the ability to send messages regarding a large scope of topics, is essential in reducing customer waiting times, improving customer experience and improving the efficiency of the business.

In addition, it has been ensured that their request can contain an apostrophe in it, which would usually throw an error due to clashes with the SQL statement. Therefore, the SQL statement had to be catered specifically to ensure that the customer would be able to enter that piece of punctuation, which could be used by them in many ways e.g. I saw Osu in store and (I’d) like to buy it. Please let me know when I can come up and purchase it. Thanks.

Moreover, validation has been added which ensures the following:

- That if a customer is issuing an “Updating of details” request, their message input must not only contain digits (which could be the value they would like a field to be changed to) but also characters to be able to explain what field they would like to be updated.

- That their message is 20 characters long, to ensure that the send message isn’t accidentally pressed before the message is completed, and to ensure that enough information is given to the staff member to work with

- That if the customer isn’t sending an “Updating of details” request, their message should not contain any numbers as it’s unnecessary

All of these validation techniques ensure that the message is relevant to the request and that it is kept to a certain standard that is deemed a valid message. This aids staff members in being able to respond or act on their request as the message is kept to an informative and valid standard.

----------------------------------------------------------------------------------------------------------------

' Allows regex to be used, which is used to check if inputs contains numbers and letters to aid validation.

Imports System.Text.RegularExpressions

Public Class SendRequest

Private Sub SendRequest()

' Defines new connection

Dim conn As New System.Data.OleDb.OleDbConnection()

' Defines location of the database

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\OneDrive\Desktop\MongooseGames.accdb"

' Opens connection to the database

conn.Open()

' SQL (Structured Query Lanugage) statement which inserts what the staff member input into the database in the corresponding table

Dim sql As String = "INSERT INTO CustRequests(CustUser,RequestType,Message) VALUES ('" & CustomerLogin.txtUsername.Text & "', '" & RequestType.Text & "', """ & txtMsg.Text & """)"

' Allows the sql command to be executed against the database

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

' Customer is notififed of their success of their required task.

MessageBox.Show("Request sent", "Sending of request successful")

'Open Database Connection again

sqlCom.Connection = conn

' Carry out the sql statement

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' Close the connection to the database

conn.Close()

' Clears all fields

txtMsg.Clear()

RequestType.Text = ""

End Sub

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

' Hides current form

Me.Hide()

' Redirects them to customermenu

CustomerMenu.Show()

End Sub

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

' Clears all fields

txtMsg.Clear()

RequestType.Text = ""

End Sub

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

' If the user is sending an updating of details of request, and the message only consists of numbers, the user is notified this can't be done.

If RequestType.SelectedItem = "Updating of details request" And IsNumeric(txtMsg.Text) = True Then

MessageBox.Show("As you are requesting an updating of details, please ensure your message doesn't only consist of digits, as it is essential the staff member knows what field you want updated", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

' If the request is less than 20 characters long, the user is informed that it must be longer

ElseIf txtMsg.Text.Length < 20 Then

MessageBox.Show("Please ensure that your message is at least 20 characters long", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

' If the user hasn't specified what type of request they are sending, they are notified that this must be done

ElseIf RequestType.SelectedItem = "" Then

MessageBox.Show("Please ensure that you enter the type of request you want to send", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

' If the user isn't making an updating of details request and has included any digits in their request, they are informed that this is unnecessary

ElseIf RequestType.SelectedItem <> "Updating of details request" And Regex.Match(txtMsg.Text, "\d").Success And Regex.Match(txtMsg.Text, "\w").Success Then

MessageBox.Show("Since you are not sending an updating of details request, your message should only contain letters. Please try again", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

' message box is then cleared

txtMsg.Clear()

' If the request has been validated, it can be sent

Else

SendRequest()

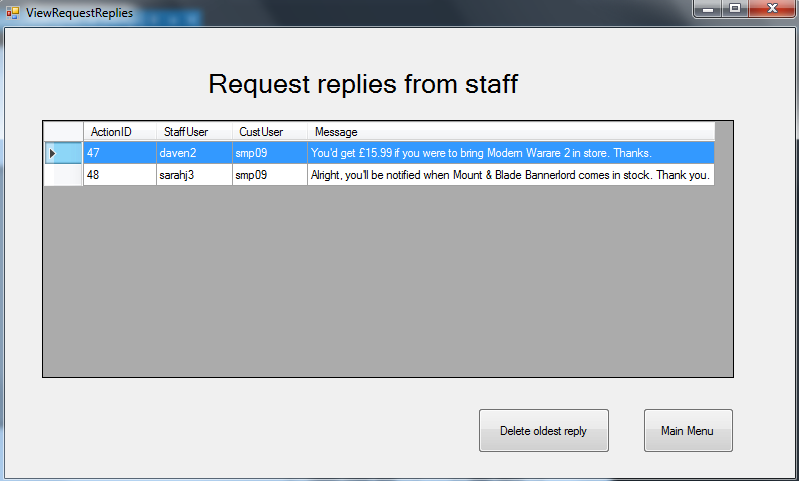
End If

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**View request replies:**

****

This form will be used by customers who wish to see any advances on the former requests that they made. Once they click the button, all of their replies are shown to them. If they have none, the DataGridView will be empty.

Moreover, they will be able to delete replies from the DataGridView (and consequently the database) if they have acknowledged the response and now wish to focus on other replies or simply to reduce clutter. This is done using the “Delete oldest reply” button, which deletes the reply that they received first in regards to all of the replies as a whole.

This extra feature promotes customers adhering to the oldest replies in order to consequently act on what they have been wanting to do/know for the longest time. Moreover, if they have a further query to the reply they will be able to make a new message using the request sender shown previously, ensuring minimum waiting time for both parties.

----------------------------------------------------------------------------------------------------------------

' Allows the database defined in the system configuration to be used

Imports System.Configuration

' Allows connection to be established with the connection string and the program

Imports System.Data.OleDb

Public Class ViewRequestReplies

' Fills data table with data from the database

Public connString As String = ConfigurationManager.ConnectionStrings("Computing\_NEA.My.MySettings.MongooseGamesConnectionString").ConnectionString

' Allows sql statement to be performed on the database

Public conn As New OleDbConnection(connString)

' Defines the data table

Public StaffReplies As New DataTable

Private Sub ViewRequestReplies\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

' Sets data source which will be the function returning the data table

StaffRepliesDataGridView.DataSource = GetCustomerRequests()

End Sub

' Creates function which returns the table to the staff member

Private Function GetCustomerRequests() As DataTable

Using conn

' SQL Statement which retrieves all of the replies that have been sent to the logged in customer so that they can be shown to them

Using cmd As New OleDbCommand("SELECT \* FROM StaffActions WHERE (CustUser) = ('" & CustomerLogin.txtUsername.Text & "')", conn)

' Opens connection to the database

conn.Open()

' As SELECT statement is being used, reader is used, allows data to be read and sent to the form

Dim reader As OleDbDataReader = cmd.ExecuteReader()

' Loads the reader data into the form

StaffReplies.Load(reader)

End Using

End Using

' The function returns the data table

Return StaffReplies

End Function

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

' Hides current form

Me.Hide()

' Shows menu

CustomerMenu.Show()

End Sub

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

' Defines variables that will be used to communicate with the database, allowing records to be deleted

Dim StaffRepliesDataSet As New DataSet

Dim StaffRepliesDataTable As New DataTable

' DataAdapter will be used to allow the SQL statement to communicate with the database

Dim StaffRepliesDataAdapter As New OleDbDataAdapter

' Loads all of the tables from the data set to the data table

StaffRepliesDataSet.Tables.Add(StaffRepliesDataTable)

' Define connection to the data source

Dim conn As New OleDbConnection(connString)

' Opens the connection

conn.Open()

' Fetches all records from the table

StaffRepliesDataAdapter = New OleDbDataAdapter("SELECT \* FROM StaffActions", conn)

' Refreshes the rows in the DataGridView to match the database

StaffRepliesDataAdapter.Fill(StaffRepliesDataTable)

' Deletes the datarow

StaffRepliesDataTable.Rows(0).Delete()

' Reconciles the changes made to the database

Dim cb As New OleDbCommandBuilder(StaffRepliesDataAdapter)

' Updates the datatable with changes made

StaffRepliesDataAdapter.Update(StaffRepliesDataTable)

' Refreshes the DataGridView to show changes made

StaffRepliesDataGridView.DataSource = StaffRepliesDataTable.DefaultView

' Closes the connection

conn.Close()

' Informs the customer that the record was deleted successfully

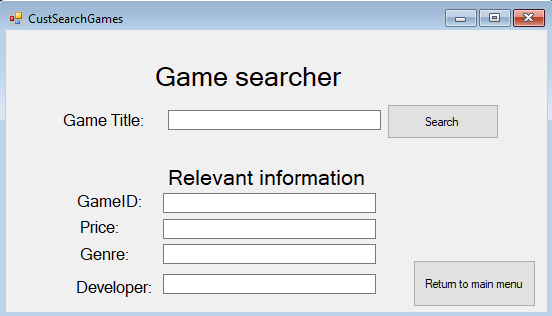
MessageBox.Show("Record deleted successfully", "Deletion of record successful")

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**Search for a game on the system**

****

This feature, although already exists for staff members, will also be available for users to utilise. As for staff members, it allows the user member to search for a certain game on the system by its name, where its corresponding information will then be given to them. They will need to input the game’s title, then click the button “Search”, they would then be notified if the game is found or not, and if it is, its information will be inputted in its corresponding text box, improving readability. This is essential for customers specifically as can allow them to search for a game that they would wish to buy, giving them a quick and informative response. This improves customer service and satisfaction, along with better business results, and more importantly, less clutter around the desk asking for if games are available and/or erroneous wishlist requests, for games already in stock, but weren’t aware of their availability.

Validation has been added to ensure that the title of the game that is being searched is not left blank, to help the user understand that the game title is necessary for its information to be displayed.

----------------------------------------------------------------------------------------------------------------

Public Class CustSearchGames

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

' If the user hasn't given a value for the game title of the game they wish to search for, they're informed that this must be filled out before their desired game is searched for

If txtGameTitle.Text = "" Then

MessageBox.Show("Please ensure that you've entered the title of the game you wish to search for", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

Else

' Connection is created

Dim conn As New System.Data.OleDb.OleDbConnection()

' Database location defined

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\OneDrive\Desktop\MongooseGames.accdb"

' SQL statement which returns all of the fields from the Games database where its name is equal to what the staff member searched

Dim sql As String = "SELECT GameID,GameName,Price,Genre,Developer FROM Games WHERE (GameName) = ('" & txtGameTitle.Text & "')"

' Allows the SQL command to communicate with the database

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

' Connects SQL command to database

sqlCom.Connection = conn

'Open Database Connection

conn.Open()

' Executes the SQL command to the database

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' If the game being searched for is found then this

If sqlRead.HasRows Then

' Reads through each field until all returned and then populates the relevant text boxes with each field value

While sqlRead.Read()

txtGameID.Text = sqlRead.Item("GameID")

txtGameTitle.Text = sqlRead.Item("GameName")

txtPrice.Text = sqlRead.Item("Price")

txtGenre.Text = sqlRead.Item("Genre")

txtDeveloper.Text = sqlRead.Item("Developer")

' User is notified of game being found

MessageBox.Show("Game found", "Searching of game successful")

End While

' Closes the read connection

sqlRead.Close()

Else

MessageBox.Show("No game called """ & txtGameTitle.Text & """" & " was found. Please try again", "Error while searching for game")

' All fields are cleared

txtGameID.Clear()

txtGameTitle.Clear()

txtPrice.Clear()

txtGenre.Clear()

txtDeveloper.Clear()

' Closes database connection

conn.Close()

End If

End If

End Sub

Private Sub BtnMainMenu\_Click(sender As Object, e As EventArgs) Handles BtnMainMenu.Clic

' Hides current form

Me.Hide()

' Shows customer menu

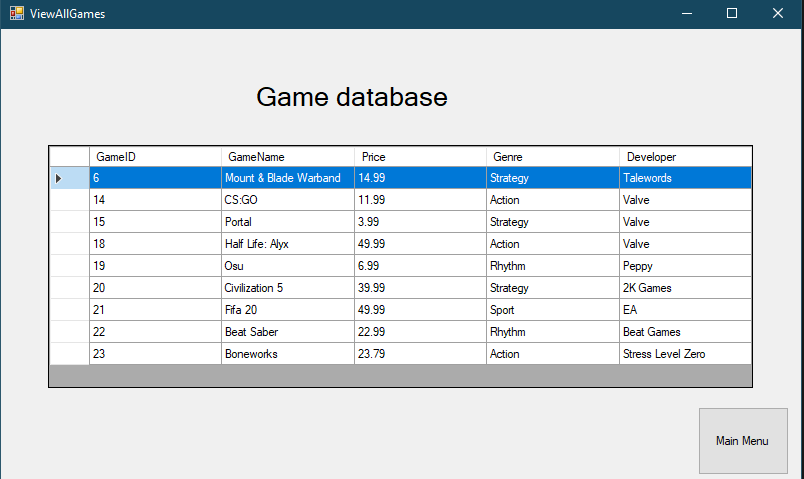
CustomerMenu.Show()

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**View all games on the system**

****

This feature will allow a customer to view all of the games on the system. This is the same function that is given to staff members but customers will also need to utilise this function to be able to browse all of the available games that they can buy, they can then use the game information to make a buy request.

----------------------------------------------------------------------------------------------------------------

' Allows the database defined in the system configuration to be used

Imports System.Configuration

' Allows connection to be established with the connection string and the program

Imports System.Data.OleDb

Public Class CustViewGames

Private Sub CustViewGames\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

' When the form loads, all of the games on the system are shown to them

GamesDataGridView.DataSource = GetGameInformation()

End Sub

' Creates function which returns the table to the staff member

Private Function GetGameInformation() As DataTable

' Defines the data table

Dim Games As New DataTable

' Fills data table with data from the database

Dim connString As String = ConfigurationManager.ConnectionStrings("Computing\_NEA.My.MySettings.MongooseGamesConnectionString").ConnectionString

' Allows sql statement to be performed on the database

Using conn As New OleDbConnection(connString)

' SQL Statement which shows all of the data to the staff member in the table

Using cmd As New OleDbCommand("SELECT \* FROM Games", conn)

' Opens connection to the database

conn.Open()

' As SELECT statement is being used, reader is used, allows data to be read and sent to the form

Dim reader As OleDbDataReader = cmd.ExecuteReader()

' Loads the reader data into the form

Games.Load(reader)

End Using

End Using

' The function returns the data table

Return Games

End Function

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

' Hides current form

Me.Hide()

' Shows customer menu

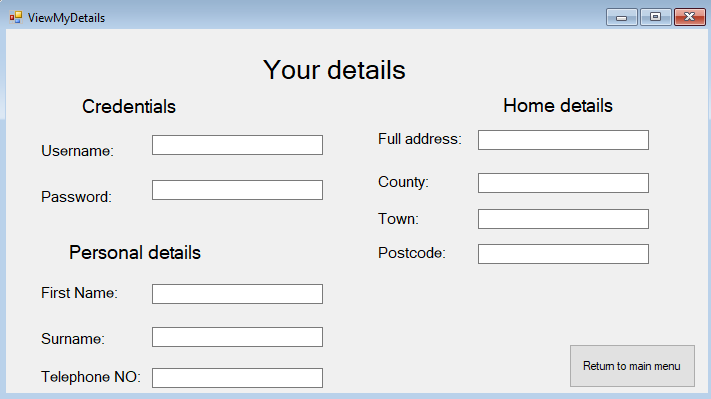
CustomerMenu.Show()

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**View current details**

****

This form allows a customer to be able to view all of the details that are stored about them. This works by recognising what account they are logged into and then displaying their information by populating the relevant text boxes. This is important as it complies with GDPR rules, which state that a customer has the right to view or request the information that is stored about them to ensure the information is up to date and not excessive.

Moreover, it will also allow a customer to detect any inaccuracies in the details that are stored about them which they can then report to a member of staff about utilising the request feature, to ensure that their details are accurate and up to date benefiting both them and the data integrity of the database as well as adhering to GDPR regulations.

----------------------------------------------------------------------------------------------------------------

Public Class ViewMyDetails

Private Sub ViewMyDetails\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

' Define connection

Dim conn As New System.Data.OleDb.OleDbConnection()

' Define database location for connection

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\OneDrive\Desktop\MongooseGames.accdb"

' Retrieves the record of the customer who is logged in

Dim sql As String = "SELECT CustUser,CustPass,FirstName,Surname,PhoneNum,Address,County,Town,Postcode FROM Customers WHERE (CustUser) = ('" & CustomerLogin.txtUsername.Text & "')"

' Allows sql command to communicate with the data source

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

' Connect SQL statement to database

sqlCom.Connection = conn

' Open connection to database

conn.Open()

' Provides a way of reading the data rows from the database

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' Executes following code if there are records found

If sqlRead.HasRows Then

' This while loop populates the relevant textbox with the corresponding information from the database until finished.

While sqlRead.Read()

txtUsername.Text = sqlRead.Item("CustUser")

txtPassword.Text = sqlRead.Item("CustPass")

txtFirstName.Text = sqlRead.Item("FirstName")

txtSurname.Text = sqlRead.Item("Surname")

txtTelephone.Text = sqlRead.Item("PhoneNum")

txtAddress.Text = sqlRead.Item("Address")

txtCounty.Text = sqlRead.Item("County")

txtTown.Text = sqlRead.Item("Town")

txtPostcode.Text = sqlRead.Item("Postcode")

End While

' Close connection between sql and database

sqlRead.Close()

Else

' If they have missing details or no details stored about them on the system, they are notified about this and advised to report this to a member of staff

MessageBox.Show("No details could be found about you, please report this to a member of staff")

End If

' Connection of the database is closed

conn.Close()

End Sub

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

' hides current form

Me.Hide()

' shows customer menu

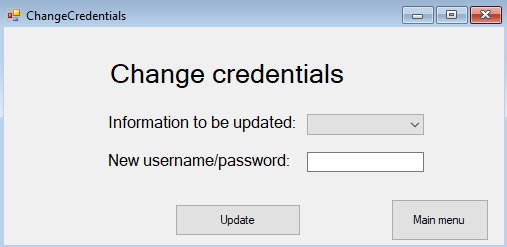
CustomerMenu.Show()

End Sub

End Class

----------------------------------------------------------------------------------------------------------------

**Change credentials**



This feature allows a customer to change their username/password by selecting their choice using the drop down menu, which has the values “CustUser” or “CustPass”, so it can be used directly within the SQL statement to communicate with the database. In case of confusion, a tool tip has been added to the textbox informing the user that “CustUser” is their Username and “CustPass” is their Password.

This feature is very similar to that of the “UpdateCustDetails” that the staff have at their disposal, however it’s based specifically on the username and password of the user. Since the username of a customer could be changed many times, they are given the function to change it themselves in order to decrease the influx of Updating of details requests to staff members. Since the password of a customer is a very personal piece of data, it would be inadequate and unprofessional for a customer to send a request to a staff member, stating they would like to change their password to x value and under the case of any malicious actions e.g. shoulder surfing, it could prove detrimental.

Validation has been added to adhere to the universal rule of the system that their new username/password must be over 5 characters, to comply with security standards.

----------------------------------------------------------------------------------------------------------------

Public Class ChangeCredentials

' The below procedure retrieves the customerID for the customer that is currently logged in, so this can be used for the SQL statement to change their username/password (as the user desires)

Sub RetrieveCustomerID()

' Defines new connection

Dim conn As New System.Data.OleDb.OleDbConnection()

' Define database location for connection

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\OneDrive\Desktop\MongooseGames.accdb"

' Retrieves the CustomerID of the customer who is logged in

Dim sql As String = "SELECT CustomerID FROM Customers WHERE (CustUser) = ('" & CustomerLogin.txtUsername.Text & "')"

' Allows sql command to communicate with the data source

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

' Connects SQL statement to database

sqlCom.Connection = conn

' Open Database Connection

conn.Open()

' Provides a way of reading the data rows from the database

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' Executes following code if there are records found

If sqlRead.HasRows Then

' This while loop populates the relevant textbox with the corresponding information from the database until finished.

While sqlRead.Read()

' Places the corresponding CustomerID in an invisible textbox so it can be used later

txtCustomerID.Text = sqlRead.Item("CustomerID")

End While

End If

' Close connection between sql and database

sqlRead.Close()

End Sub

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

' If they do not specify what they want to change their username/password to, they are informed that this is necessary

If txtNewData.Text = "" Then

MessageBox.Show("Please enter what you want your new username/password to be", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

' If their new username/password is under 5 characters, they are notified that it must be longer

ElseIf txtNewData.Text.Length < 5 Then

MessageBox.Show("Please ensure that your new username/password is longer than 5 characters", "Validation error", MessageBoxButtons.OK, MessageBoxIcon.Error)

Else

' Create connection to the database

Dim conn As New System.Data.OleDb.OleDbConnection()

' Defines location of the database

conn.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\ipr0z\OneDrive\Desktop\MongooseGames.accdb"

' SQL (Structured Query Lanugage) statement which changes the value of the desired field to the desired value in the record of the customer currently logged in

Dim sql As String = "UPDATE Customers SET " & FieldName.SelectedItem & " = """ & txtNewData.Text & """ WHERE CustomerID = " & txtCustomerID.Text & ""

' Executes the statement against the database

Dim sqlCom As New System.Data.OleDb.OleDbCommand(sql)

' Connects the sql statement to the database

sqlCom.Connection = conn

' Opens the connection

conn.Open()

' Staff member is notified of their success of their required task.

MessageBox.Show("Field value updated", "Database record updation successful")

' Carry out the sql statement

Dim sqlRead As System.Data.OleDb.OleDbDataReader = sqlCom.ExecuteReader()

' Close the connection to the database

conn.Close()

' Clears all text box values to enable them to change other information in rapid succession

FieldName.Text = ""

txtNewData.Clear()

End If

End Sub

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

' Hides current form

Me.Hide()

' Redirects to customer menu

CustomerMenu.Show()

End Sub

Private Sub ChangeCredentials\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

' When the form loads, the CustomerID is retrieved

RetrieveCustomerID()

End Sub

End Class

----------------------------------------------------------------------------------------------------------------