

**Business Application Development**

**Implementation**

**Team PLAB**

**Callum McLaughlin**

Contents

[Project Overview 3](#_Toc35423522)

[My Part of the Project 3](#_Toc35423523)

[Main Menu Form 5](#_Toc35423524)

[Customer Form 6](#_Toc35423525)

[Add Customer 7](#_Toc35423526)

[Edit Customer 8](#_Toc35423527)

[Delete Customer 9](#_Toc35423528)

[Cars Form 10](#_Toc35423529)

[Add Car 11](#_Toc35423530)

[Edit Car 12](#_Toc35423531)

[Delete Car 12](#_Toc35423532)

[Sales Form 13](#_Toc35423533)

[Add Sale 14](#_Toc35423534)

[Trade in Form (Add Sale) 15](#_Toc35423535)

[Finance Form 15](#_Toc35423536)

[Fig.1 15](#_Toc35423537)

[Fig.2 16](#_Toc35423538)

[Main Menu Code 18](#_Toc35423539)

[Customer Form Code 21](#_Toc35423540)

[Car Form Code 46](#_Toc35423541)

[Sales Form Code 82](#_Toc35423542)

[GlobalVar Class 127](#_Toc35423543)

[MyCar Class 128](#_Toc35423544)

[MyCustomer Class 132](#_Toc35423545)

[MySales Class 136](#_Toc35423546)

[MyFinanceAgreement Class 138](#_Toc35423547)

[MyValidation Class 140](#_Toc35423548)

[MyException Class 145](#_Toc35423549)

[Input Testing 146](#_Toc35423550)

[String Testing 146](#_Toc35423551)

[String Validation 147](#_Toc35423552)

[Integer Testing 148](#_Toc35423553)

[Integer Validation 149](#_Toc35423554)

[Functionality Testing 150](#_Toc35423555)

# Project Overview

The system we set out to make was for a car dealership, Desmond Motors. The company wanted a system which would allow customers, cars, upgrades and orders to be Created, Read, updated and deleted on the system. They also wanted us to add features to the system that would allow them to make an order for as customer, for a specific car. This order system would allow the customer to select the car they want to buy, add a car of their own to trade in against the new car which would take off some of the price and they could even choose whether to pay by cash, card, cheque or finance.

If the customer were to take the car out on finance the system shows them a detailed breakdown of the payment plan, including how many months, monthly cost and total cost including interest and excluding interest. The customer can also choose and plan length which will automatically update the finance breakdown when a new plan length is selected.

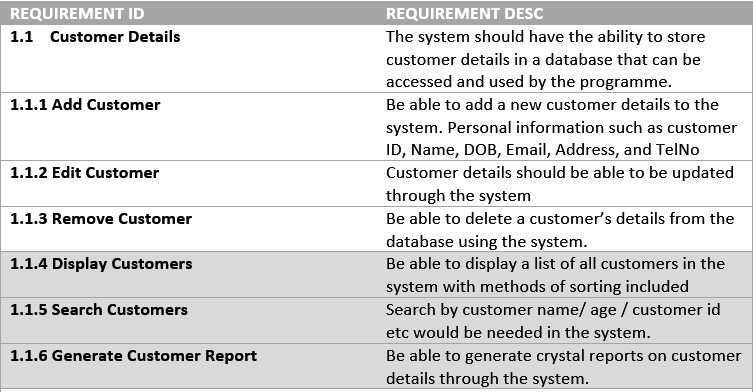
The systems design was all based around the core colour scheme that desmonds uses at their showrooms, blue and white. On the main menu we were also able to create a scrolling banner with the desmonds motors logo.

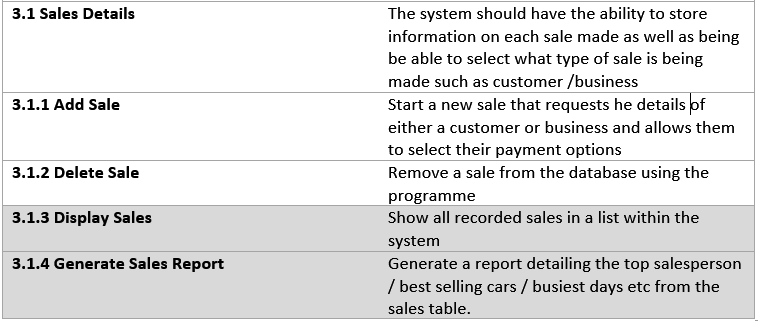
The system is created to be used by employees at the showroom. An example being, an employee sitting down with a customer and asking them for the details of the car they want to purchase, the car they may want to trade in, finance plan length etc. The employee will put in these details and will be able to show the customer on their monitor the plan breakdown.

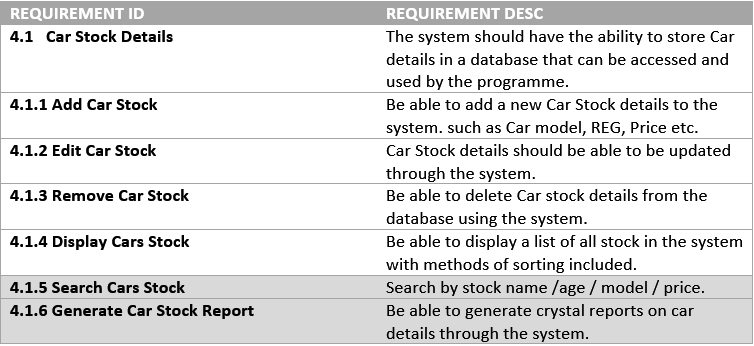
## My Part of the Project

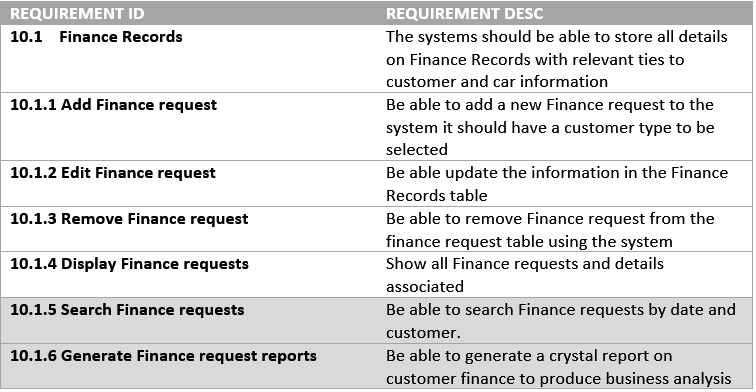
As part of a two-man team we had a lot of work split between the two of us in order to make up for the fact that other groups had three. We were challenged to create a system as good as the rest of the teams but strived to make it better. My task was to create the customer, cars and finance tables which would allow the user to Add, Edit, Delete and Read customer, car order and finance details.

The requirements that I was assigned with are listed below. All requirements were met.







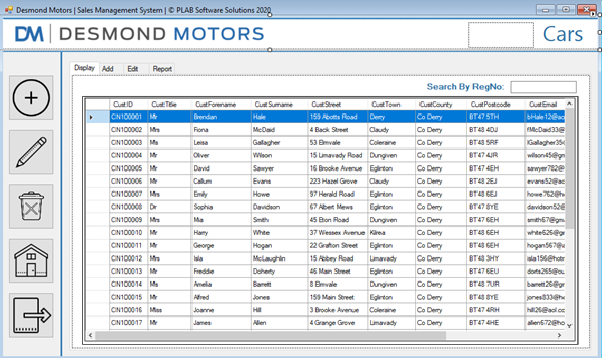


# Main Menu Form



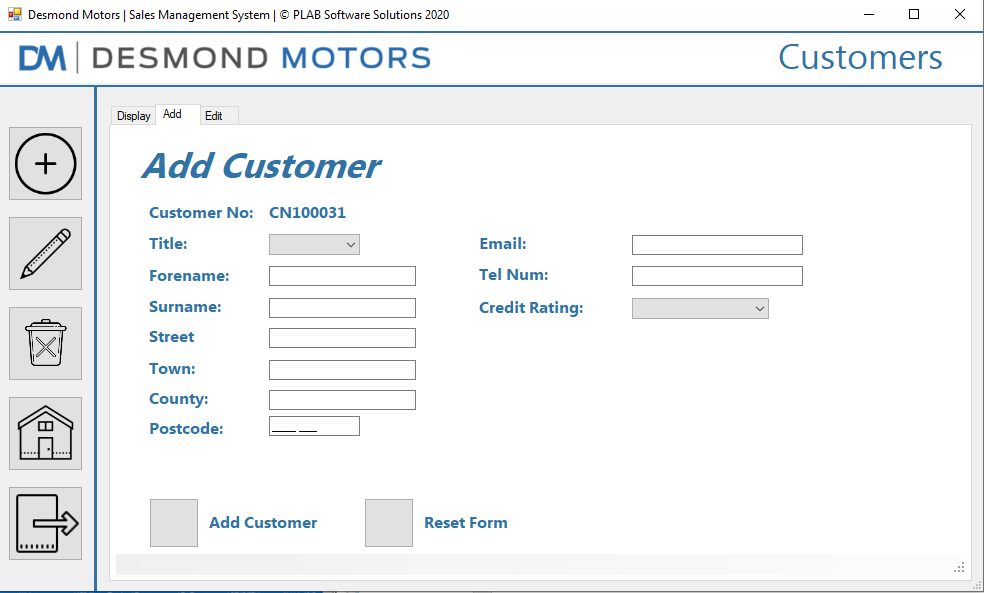
Above is the main menu form. This form acts as the home page for the entire application. From this page you can move to any other form. The buttons on the left allow the user to navigate to the customer form, maintenance/upgrades form, cars form, sales form, staff form and there’s an exit button at the bottom which exits the entire application. The banner at the top of the screen is also a scrolling banner, it moves across the screen to the left.

# Customer Form

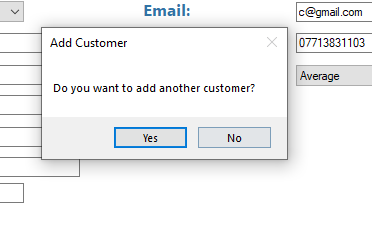
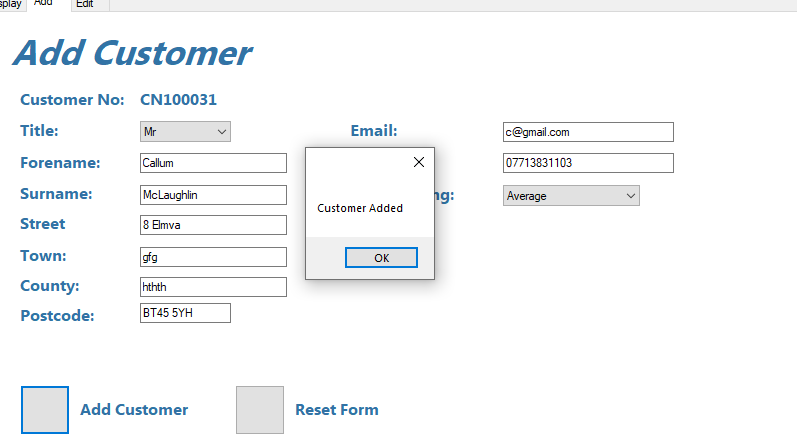
****

Shown above is the customer form. This form allows the user the create, read, update and delete customers from the system. The buttons on the left are for adding, editing, deleting customers as well as a home button and exit button. To add a customer, you simply click on the add button and this screen will be shown:

## Add Customer

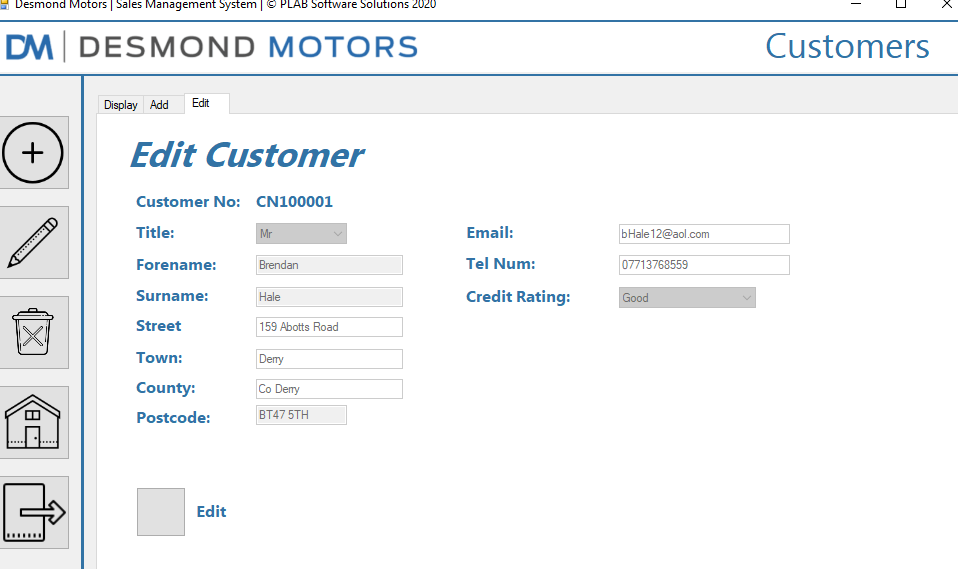


From here you can enter the customer details using each input field and the reset form button can be used to clear all input and start again if need be. The add customer button will add the customer to the system if all data entered is valid. When you add a customer, a message box will show stating that a customer has been added. After you click Ok on that box another message box will show asking if you want to dd another customer, if you click yes, it will reset the add form and allow you to add another customer, if you click no it will return you to the display customers tab.

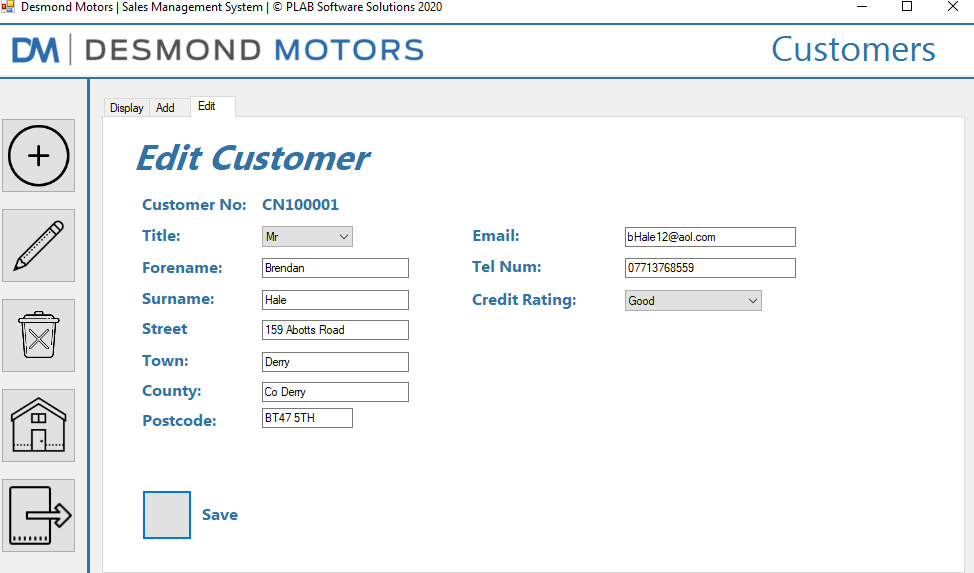


To edit a customer, you can simply click on the customer you wish to edit on the list and either click the edit button on the left, or the edit tab and it will pull that selected customers details into the edit screen as shown below:

## Edit Customer



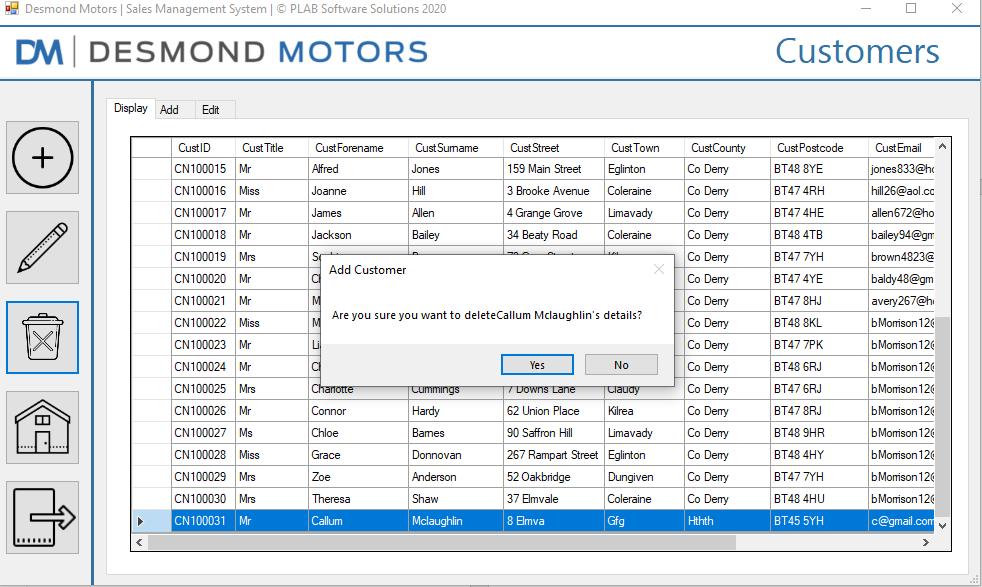
To begin editing the customers details you must enable the input fields by clicking the edit button, after you do this the edit button will change to a save button as shown below:



After you edit the customers details and have finished, click the save button and you will be shown a message box confirming the customers details have been updated successfully.

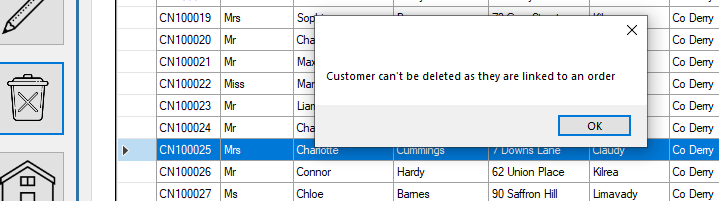
## Delete Customer

If you wish to delete a customer details, you select their name on the list and click the delete button on the left side of the screen and you will be prompted with a message like the one shown below:

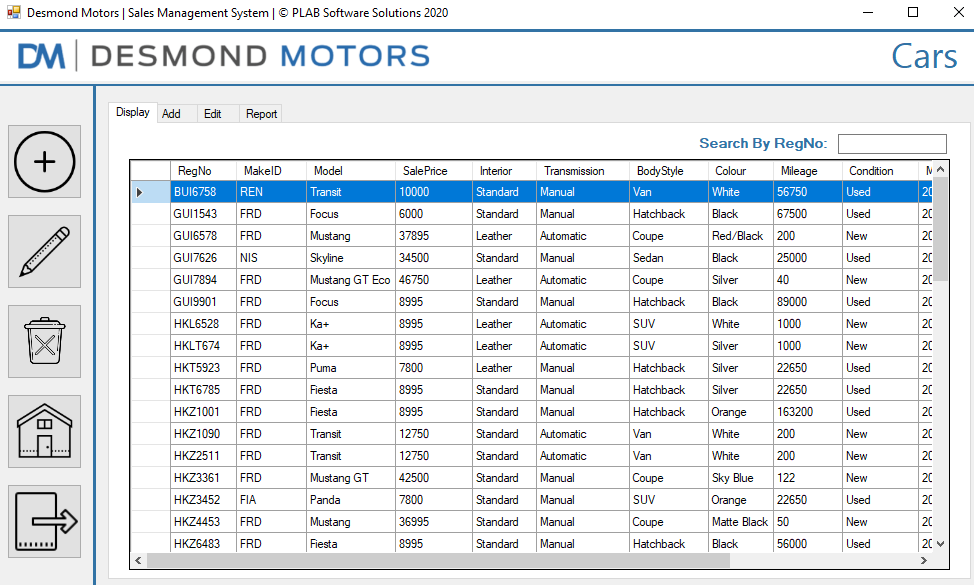


If you click yes, the customers details will be deleted, if you click no it will close the message box, and nothing further will happen.

N.B. If a customer has an order linked to their account you will not be able to delete the customer; an example of that error message is shown below:

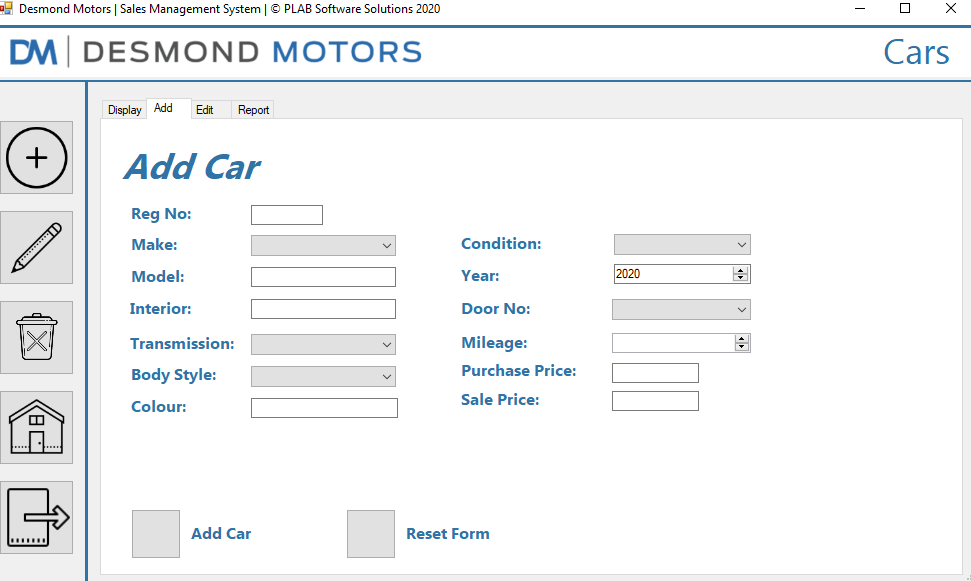


# Cars Form



Shown above is the cars form, like the customer form, this form allows the user the create, read, update and delete cars from the system. The buttons on the left are for adding, editing, deleting cars as well as a home button and exit button. To add a car, you simply click on the add button and this screen will be shown:

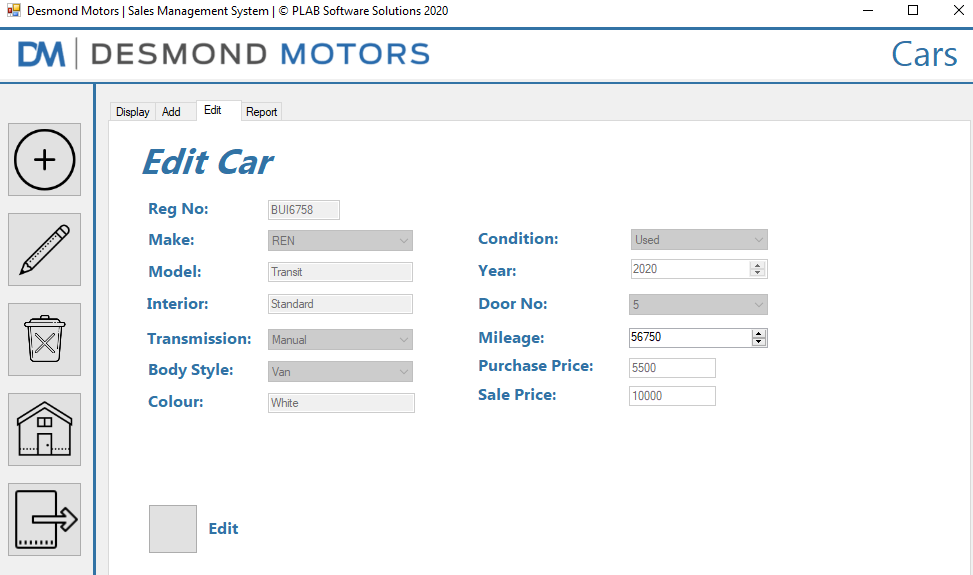
## Add Car



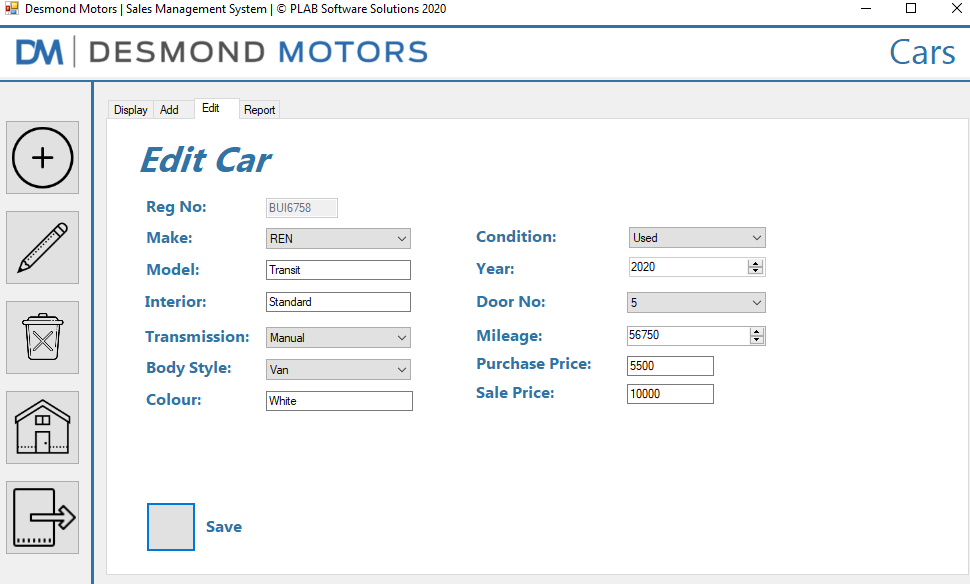
The same as the customer form, here you can enter the car details using each input field and the reset form button can be used to clear all input and start again if need be. The add car button will add the car to the system if all data entered is valid. When you add a car, a message box will show stating that a car has been added. After you click Ok on that box another message box will show asking if you want to dd another car, if you click yes, it will reset the add form and allow you to add another car, if you click no it will return you to the display cars tab.

To edit a car, you can simply click on the car you wish to edit on the list and either click the edit button on the left, or the edit tab and it will pull that selected cars details into the edit screen as shown below:

## Edit Car



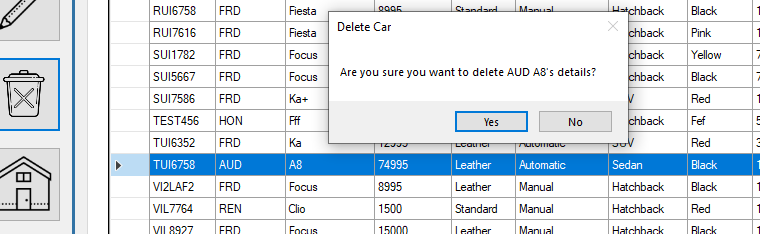
To begin editing the cars details you must enable the input fields by clicking the edit button, after you do this the edit button will change to a save button as shown below:



After you edit the cars details and have finished, click the save button and you will be shown a message box confirming the cars details have been changed successfully.

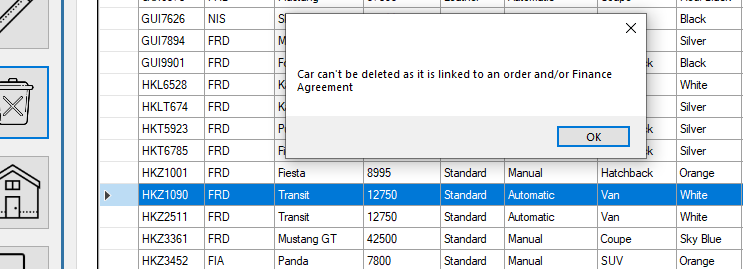
If you wish to delete a cars details, you select the car reg on the list and click the delete button on the left side of the screen and you will be prompted with a message like the one shown below:

## Delete Car



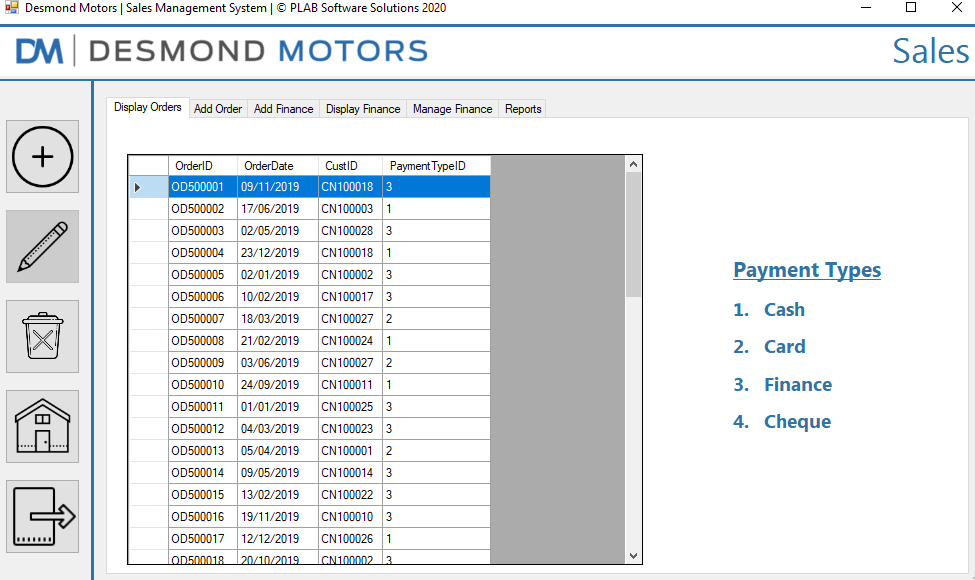
If you click yes, the cars details will be deleted, if you click no it will close the message box, and nothing further will happen.

N.B. If a car is linked to an order or finance agreement you will not be able to delete the car. An example of this error message is shown below:



# Sales Form

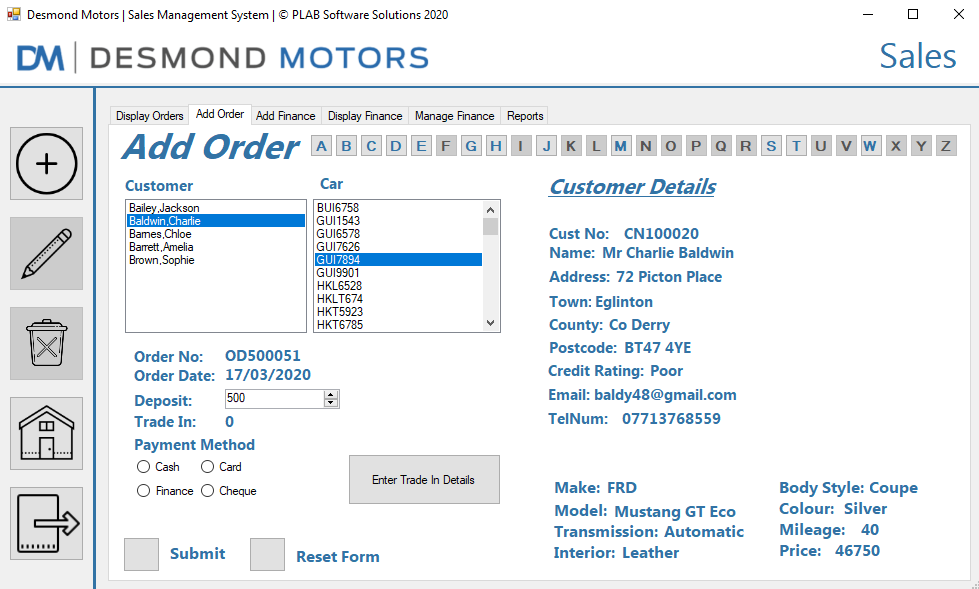
Shown below is the Sales form. This form handles every type of sale that goes through the dealership whether its finance or straight payment with cash or card. The user can add orders, add finance agreements and edit finance agreements as well as view reports from this form.



The display orders tab allows you to view all orders that have been made. You can also delete orders but only if they do not have a finance agreement attached to them.

Shown below if the add order tab here the user can place an order for the customer.

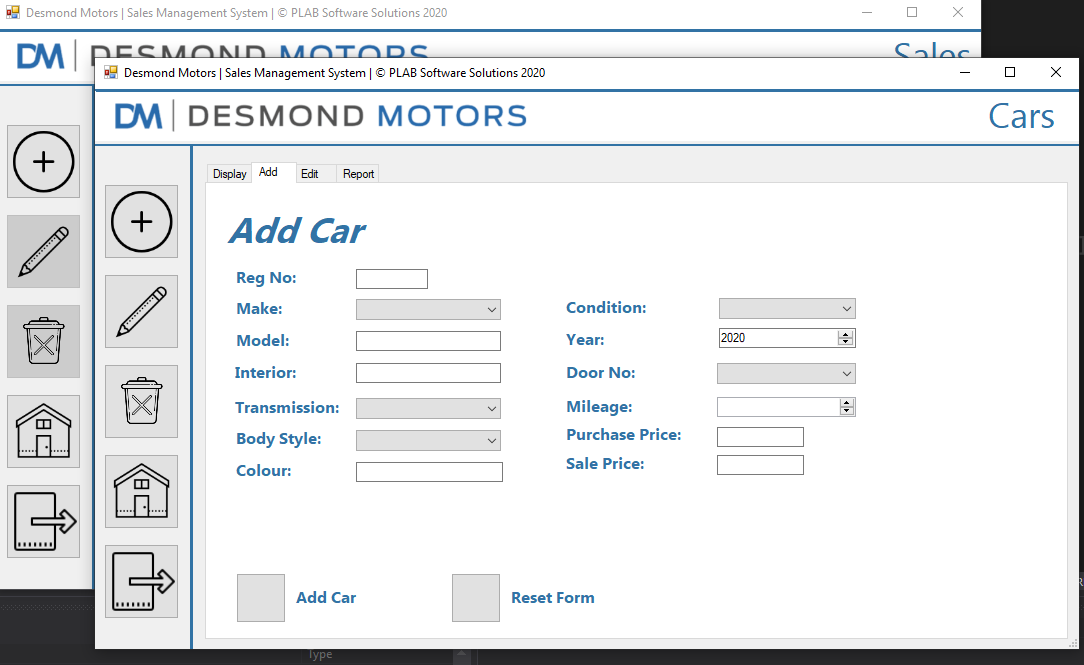
## Add Sale



The user can select the customer by filtering first by surname using the letters at the top of the form. The user can then select the customer from the list on the left and the car reg on the right, selecting these will fill out the details sections on the right to ensure that the user is selecting the right customer and car.

The user can then input the deposit, which is a minimum of 500, a payment method and they can also click on the “Enter trade in details “ button which will open a new form on top of the current form allowing them to add the customers trade in car to the database, once this is done the trade in value will be shown on the main form. The trade in form in shown below (it is an instance of the add car form):

## Trade in Form (Add Sale)

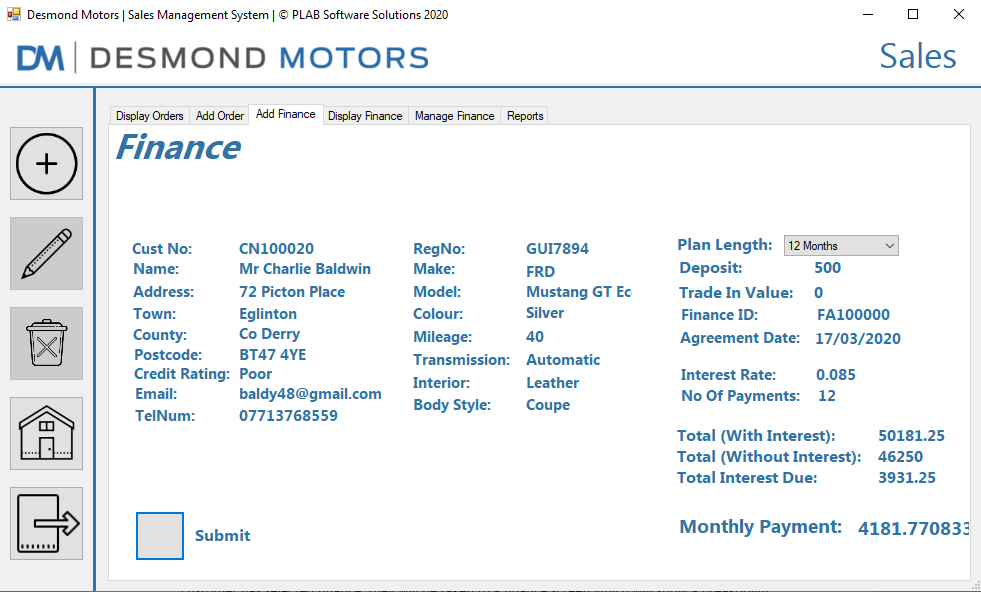


Once this form has been filled in and the car is added the trade in value will change to be equal to the purchase price of the added car, in other words, how much desmonds are willing to pay for the car. Once this had been done, the user clicks the submit button to complete the order.

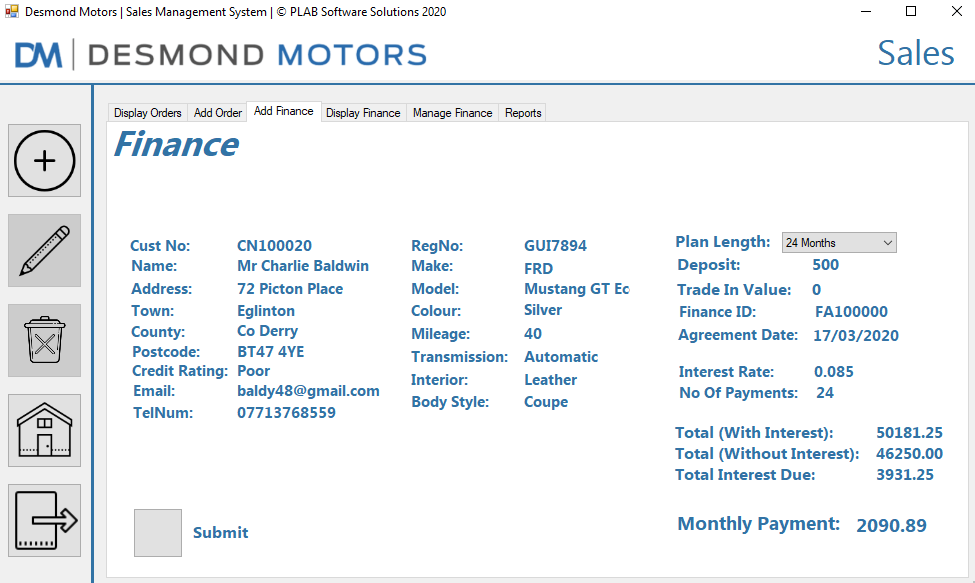
If the customer has selected finance, they will be taken to a finance screen which will show a breakdown of their payment plan and they can choose a plan length that updates the finance breakdown each time the plan length is changed. An example of this screen is shown below where fig.1 shows a 12 months plan and fig.2 shows a 24-month finance breakdown for the same car and customer:

## Finance Form

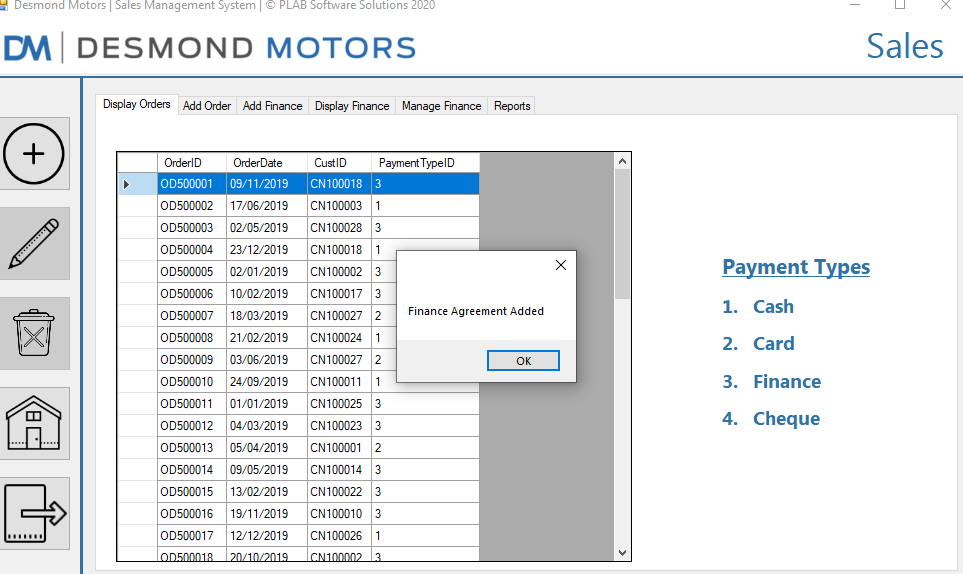
### Fig.1



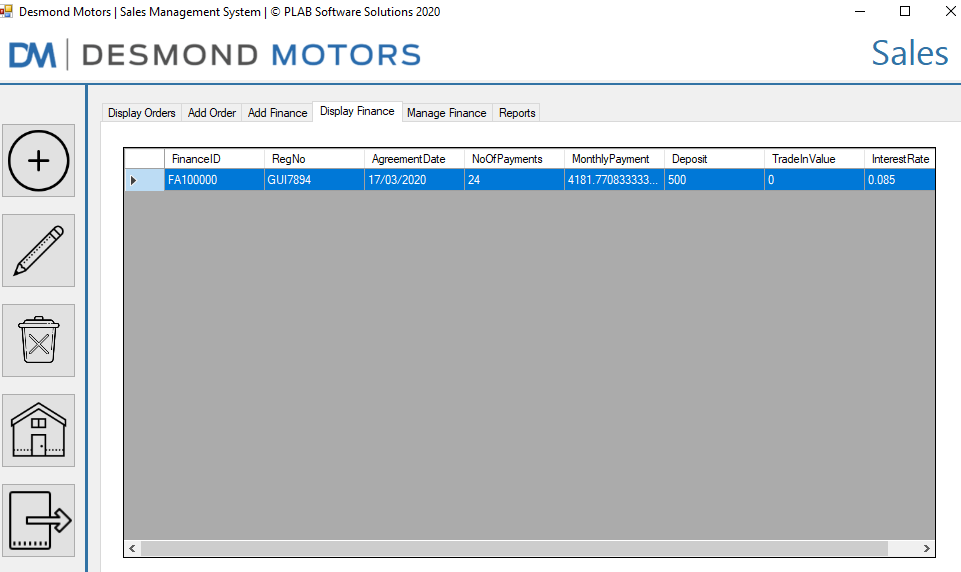
### Fig.2



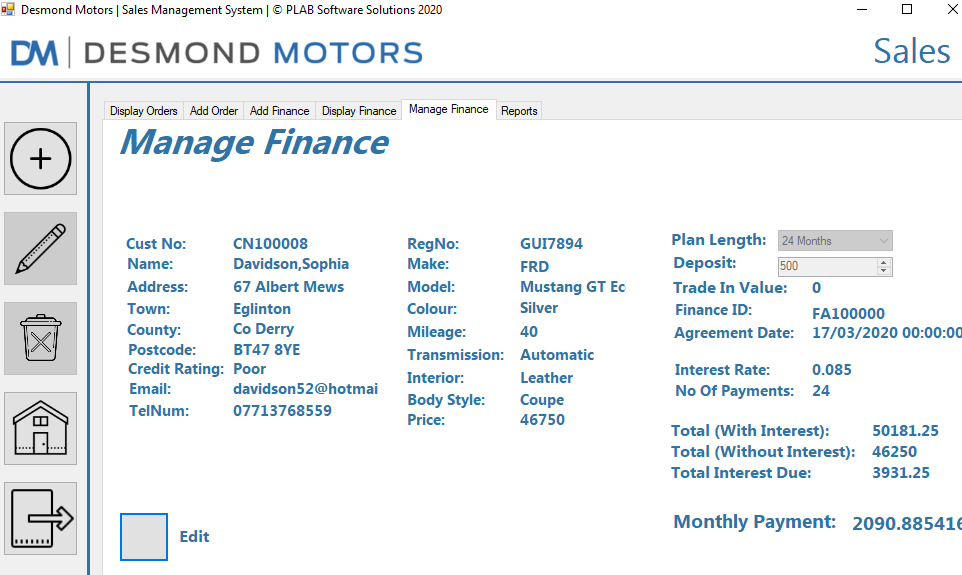
When a plan length is selected a message box will be displayed confirming that an order has been added, if it was a finance order it will show a 2nd message box as shown below:



The Display finance tab will show all finance agreements that have been created. An example is shown below



To edit a finance agreement, you select the finance ID from the list and either click the manage finance tab or the edit button on the left and you will be taken to this screen:



To edit the agreement, you click on the edit button which then changes to a save button, allowing you to edit the plan length and the deposit value. The reason I allowed the user to edit the deposit value was that if a customer wanted to come in a pay off a bulk sum off their finance plan, this would allow for it the be accounted for easily. An example being that if a customer purchased a car on finance for £10000 and paid a £2000 deposit, they would have £8000 left to pay on finance. With this feature a customer could come in and say they want to pay off another £3000 off their £8000 that they are currently paying on finance, meaning they only have £5000 left to pay on finance with interest.

Lastly is the reports tab which will show the invoices for each customer, filtering by customer number using a combo box as shown below:



# Main Menu Code

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ScreenDesigns

{

public partial class frmMainMenu : Form

{

public frmMainMenu()

{

InitializeComponent();

}

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

{

pnlTopMenuBar.BackColor = Color.FromArgb(27, 99, 155);

panel1.BackColor = Color.FromArgb(27, 99, 155);

}

private void bannerTimer\_Tick(object sender, EventArgs e)

{

pb1.Left -= 2;

pb2.Left -= 2;

if (pb1.Left == -100)

pb1.Left = 100;

if (pb2.Left <= -100)

pb2.Left = pb1.Left + 100;

}

private void btnCustomer\_MouseHover(object sender, EventArgs e)

{

lblCustBtn.Visible = true;

}

private void btnCustomer\_MouseLeave(object sender, EventArgs e)

{

lblCustBtn.Visible = false;

}

private void btnMaintenance\_MouseHover(object sender, EventArgs e)

{

lblMaintenanceBtn.Visible = true;

}

private void btnMaintenance\_MouseLeave(object sender, EventArgs e)

{

lblMaintenanceBtn.Visible = false;

}

private void btnCars\_MouseHover(object sender, EventArgs e)

{

lblCarsBtn.Visible = true;

}

private void btnCars\_MouseLeave(object sender, EventArgs e)

{

lblCarsBtn.Visible = false;

}

private void btnSales\_MouseHover(object sender, EventArgs e)

{

lblSalesBtn.Visible = true;

}

private void btnSales\_MouseLeave(object sender, EventArgs e)

{

lblSalesBtn.Visible = false;

}

private void btnAppointments\_MouseHover(object sender, EventArgs e)

{

lblStaffBtn.Visible = true;

}

private void btnAppointments\_MouseLeave(object sender, EventArgs e)

{

lblStaffBtn.Visible = false;

}

private void btnExit\_MouseHover(object sender, EventArgs e)

{

lblExitBtn.Visible = true;

}

private void btnExit\_MouseLeave(object sender, EventArgs e)

{

lblExitBtn.Visible = false;

}

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

{

this.Close();

Environment.Exit(1);

}

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

{

this.Hide();

frmCustomer frmCust = new frmCustomer();

frmCust.ShowDialog();

}

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

{

this.Hide();

frmMaintenance frmMain = new frmMaintenance();

frmMain.ShowDialog();

}

private void btnStaff\_MouseHover(object sender, EventArgs e)

{

lblStaffBtn.Show();

}

private void btnStaff\_MouseLeave(object sender, EventArgs e)

{

lblStaffBtn.Hide();

}

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

{

this.Hide();

frmStaff frmSt = new frmStaff();

frmSt.ShowDialog();

}

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

{

this.Hide();

frmCar frmCr = new frmCar();

frmCr.ShowDialog();

}

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

{

this.Hide();

frmSales frmSa = new frmSales();

frmSa.ShowDialog();

}

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

{

this.Hide();

frmTradeIn frmTr = new frmTradeIn();

frmTr.ShowDialog();

}

}

}

# Customer Form Code

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ScreenDesigns

{

public partial class frmCustomer : Form

{

SqlDataAdapter daCustomer, daOrders;

DataSet dsDesmonds = new DataSet();

SqlCommandBuilder cmdBCustomer, cmdBOrder;

DataRow drCustomer;

String connStr, sqlCustomer, sqlOrders;

int selectedTab = 0;

bool custSelected = false;

string custNoSelected = "";

public frmCustomer()

{

InitializeComponent();

}

private void frmCustomer\_Shown(object sender, EventArgs e)

{

tabCustomer.TabPages[0].CausesValidation = true;

tabCustomer.TabPages[0].Validating += new CancelEventHandler(AddTabValidate);

}

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

{

connStr = @"Data Source = .; Initial Catalog = desmonds; Integrated Security = true";

sqlCustomer = @"select \* from customer";

daCustomer = new SqlDataAdapter(sqlCustomer, connStr);

cmdBCustomer = new SqlCommandBuilder(daCustomer);

daCustomer.FillSchema(dsDesmonds, SchemaType.Source, "Customer");

daCustomer.Fill(dsDesmonds, "Customer"); // Naming the table in VS //

sqlOrders = @"SELECT \* FROM orders";

daOrders = new SqlDataAdapter(sqlOrders, connStr);

cmdBOrder = new SqlCommandBuilder(daOrders);

daOrders.FillSchema(dsDesmonds, SchemaType.Source, "Orders");

daOrders.Fill(dsDesmonds, "Orders");

dgvCustomer.DataSource = dsDesmonds.Tables["Customer"];

// Resize the dgv columns to fit the newly loaded content //

dgvCustomer.AutoResizeColumns(DataGridViewAutoSizeColumnsMode.AllCells);

tabCustomer.SelectedIndex = 1;

tabCustomer.SelectedIndex = 0;

}

// Form Button Click Events //

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

{

tabCustomer.SelectedIndex = 1;

}

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

{

tabCustomer.SelectedIndex = 2;

}

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

{

if (dgvCustomer.SelectedRows.Count == 0)

{

MessageBox.Show("Please select a customer from the list", "Customer");

}

else

{

bool okDelete = true;

foreach (DataRow dr in dsDesmonds.Tables["orders"].Rows)

{

if (dr["CustID"].ToString() == dgvCustomer.SelectedRows[0].Cells[0].Value.ToString())

{

okDelete = false;

}

}

if (okDelete == true)

{

drCustomer = dsDesmonds.Tables["Customer"].Rows.Find(dgvCustomer.SelectedRows[0].Cells[0].Value);

string tempName = drCustomer["CustForename"].ToString() + " " + drCustomer["CustSurname"].ToString() + "\'s";

if (MessageBox.Show("Are you sure you want to delete" + tempName + " details?", "Add Customer", MessageBoxButtons.YesNo) == System.Windows.Forms.DialogResult.Yes)

{

drCustomer.Delete();

daCustomer.Update(dsDesmonds, "Customer");

}

}

else

MessageBox.Show("Customer can't be deleted as they are linked to an order");

}

}

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

{

System.Windows.Forms.Application.Exit();

}

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

{

errP.Clear();

MyCustomer myCustomer = new MyCustomer();

bool Ok = true;

String invalMessage = "Invalid data entry for: ";

//errP.clear();

try

{

myCustomer.CustomerNo = lblAddCustNoShow.Text.Trim(); // passed to customer class to check //

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(lblAddCustNoShow, MyEx.toString());

}

try

{

myCustomer.Title = cmbAddCustTitle.Text.Trim();

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(cmbAddCustTitle, MyEx.toString());

invalMessage += " Title |";

}

try

{

myCustomer.Forename = txtAddCustForename.Text.Trim();

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtAddCustForename, MyEx.toString());

invalMessage += " Forename |";

}

try

{

myCustomer.Surname = txtAddCustSurname.Text.Trim();

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtAddCustSurname, MyEx.toString());

invalMessage += " Surname |";

}

try

{

myCustomer.Street = txtAddCustStreet.Text.Trim();

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtAddCustStreet, MyEx.toString());

invalMessage += " Street |";

}

try

{

myCustomer.Town = txtAddCustTown.Text.Trim();

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtAddCustTown, MyEx.toString());

invalMessage += " Town |";

}

try

{

myCustomer.County = txtAddCustCounty.Text.Trim();

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtAddCustCounty, MyEx.toString());

invalMessage += " County |";

}

try

{

myCustomer.Postcode = txtAddCustPostcode.Text.Trim();

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtAddCustPostcode, MyEx.toString());

invalMessage += " Postcode |";

}

try

{

myCustomer.TelNo = txtAddCustTelNum.Text.Trim();

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtAddCustTelNum, MyEx.toString());

invalMessage += " TelNum |";

}

try

{

myCustomer.Email = txtAddCustEmail.Text.Trim();

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtAddCustEmail, MyEx.toString());

invalMessage += " Email |";

}

try

{

myCustomer.CreditRating = cmbAddCustCreditRating.Text.Trim();

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(cmbAddCustCreditRating, MyEx.toString());

invalMessage += " Credit Rating |";

}

if (Ok == false)

stLabel.Text = invalMessage;

try

{

if (Ok)

{

drCustomer = dsDesmonds.Tables["Customer"].NewRow();

drCustomer["CustID"] = myCustomer.CustomerNo;

drCustomer["CustTitle"] = myCustomer.Title;

drCustomer["CustForename"] = myCustomer.Forename;

drCustomer["CustSurname"] = myCustomer.Surname;

drCustomer["CustStreet"] = myCustomer.Street;

drCustomer["CustTown"] = myCustomer.Town;

drCustomer["CustCounty"] = myCustomer.County;

drCustomer["CustPostcode"] = myCustomer.Postcode;

drCustomer["CustEmail"] = myCustomer.Email;

drCustomer["TelNo"] = myCustomer.TelNo;

drCustomer["CreditRating"] = myCustomer.CreditRating;

dsDesmonds.Tables["Customer"].Rows.Add(drCustomer);

daCustomer.Update(dsDesmonds, "Customer");

MessageBox.Show("Customer Added");

if (MessageBox.Show("Do you want to add another customer?", "Add Customer", MessageBoxButtons.YesNo) == System.Windows.Forms.DialogResult.Yes)

{

clearAddForm();

getTotalNum(dsDesmonds.Tables["Customer"].Rows.Count);

errP.Clear();

clearTxtError();

}

else

tabCustomer.SelectedIndex = 0;

}

}

catch (Exception ex)

{

MessageBox.Show("" + ex.TargetSite + "" + ex.Message, "Error!", MessageBoxButtons.AbortRetryIgnore, MessageBoxIcon.Error);

}

}

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

{

clearAddForm();

}

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

{

if (lblEditCustEdit.Text == "Edit")

{

cmbEditCustTitle.Enabled = true;

txtEditCustForename.Enabled = true;

txtEditCustSurname.Enabled = true;

txtEditCustStreet.Enabled = true;

txtEditCustTown.Enabled = true;

txtEditCustCounty.Enabled = true;

txtEditCustPostcode.Enabled = true;

txtEditCustEmail.Enabled = true;

txtEditCustTelNum.Enabled = true;

cmbEditCustCreditRating.Enabled = true;

lblEditCustEdit.Text = "Save";

}

else

{

MyCustomer myCustomer = new MyCustomer();

bool Ok = true;

errP.Clear();

try

{

myCustomer.CustomerNo = (lblEditCustNoShow.Text.Trim()); // passed to customer class to check //

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(lblEditCustNoShow, MyEx.toString());

}

try

{

myCustomer.Title = cmbEditCustTitle.Text.Trim();

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(cmbEditCustTitle, MyEx.toString());

}

try

{

myCustomer.Forename = txtEditCustForename.Text.Trim();

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtEditCustForename, MyEx.toString());

}

try

{

myCustomer.Surname = txtEditCustSurname.Text.Trim();

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtEditCustSurname, MyEx.toString());

}

try

{

myCustomer.Street = txtEditCustStreet.Text.Trim();

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtEditCustStreet, MyEx.toString());

}

try

{

myCustomer.Town = txtEditCustTown.Text.Trim();

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtEditCustTown, MyEx.toString());

}

try

{

myCustomer.County = txtEditCustCounty.Text.Trim();

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtEditCustCounty, MyEx.toString());

}

try

{

myCustomer.Postcode = txtEditCustPostcode.Text.Trim();

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtEditCustPostcode, MyEx.toString());

}

try

{

myCustomer.TelNo = txtEditCustTelNum.Text.Trim();

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtEditCustTelNum, MyEx.toString());

}

try

{

myCustomer.Email = txtEditCustEmail.Text.Trim();

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtEditCustEmail, MyEx.toString());

}

try

{

myCustomer.CreditRating = cmbEditCustCreditRating.Text.Trim();

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(cmbEditCustCreditRating, MyEx.toString());

}

try

{

if (Ok)

{

drCustomer.BeginEdit();

drCustomer["CustID"] = myCustomer.CustomerNo;

drCustomer["CustTitle"] = myCustomer.Title;

drCustomer["CustForename"] = myCustomer.Forename;

drCustomer["CustSurname"] = myCustomer.Surname;

drCustomer["CustStreet"] = myCustomer.Street;

drCustomer["CustTown"] = myCustomer.Town;

drCustomer["CustCounty"] = myCustomer.County;

drCustomer["CustPostcode"] = myCustomer.Postcode;

drCustomer["CustEmail"] = myCustomer.Email;

drCustomer["TelNo"] = myCustomer.TelNo;

drCustomer["CreditRating"] = myCustomer.CreditRating;

drCustomer.EndEdit();

daCustomer.Update(dsDesmonds, "Customer");

MessageBox.Show("Customer Details Updated", "Customer");

cmbEditCustTitle.Enabled = false;

txtEditCustForename.Enabled = false;

txtEditCustSurname.Enabled = false;

txtEditCustStreet.Enabled = false;

txtEditCustTown.Enabled = false;

txtEditCustCounty.Enabled = false;

txtEditCustPostcode.Enabled = false;

txtEditCustEmail.Enabled = false;

txtEditCustTelNum.Enabled = false;

cmbEditCustCreditRating.Enabled = false;

lblEditCustEdit.Text = "Edit";

tabCustomer.SelectedIndex = 0;

}

}

catch (Exception ex)

{

MessageBox.Show("" + ex.TargetSite + "" + ex.Message, "Error!", MessageBoxButtons.AbortRetryIgnore, MessageBoxIcon.Error);

}

}

}

// Custom Methods //

private void getTotalNum(int noRows)

{

drCustomer = dsDesmonds.Tables["Customer"].Rows[noRows - 1];

int temp = int.Parse(drCustomer["CustID"].ToString().Substring(2, 6)) + 1;

lblAddCustNoShow.Text = "CN" + temp;

}

void clearAddForm()

{

cmbAddCustTitle.SelectedIndex = -1;

txtAddCustForename.Clear();

txtAddCustSurname.Clear();

txtAddCustStreet.Clear();

txtAddCustTown.Clear();

txtAddCustCounty.Clear();

txtAddCustPostcode.Clear();

txtAddCustEmail.Clear();

txtAddCustTelNum.Clear();

cmbAddCustCreditRating.SelectedIndex = -1;

}

void AddTabValidate(object sender, CancelEventArgs e)

{

if (dgvCustomer.SelectedRows.Count == 0)

{

custSelected = false;

custNoSelected = "";

}

else if (dgvCustomer.SelectedRows.Count == 1)

{

custSelected = true;

custNoSelected = dgvCustomer.SelectedRows[0].Cells[0].Value.ToString();

}

}

void EditTabValidate(object sender, CancelEventArgs e)

{

if (custSelected == false && custNoSelected == "")

{

// have to do this bit //

// reset tab to display and put out a message to select customer

custSelected = false;

custNoSelected = "";

}

else if (dgvCustomer.SelectedRows.Count == 1)

{

custSelected = true;

custNoSelected = dgvCustomer.SelectedRows[0].Cells[0].ToString();

}

}

void clearTxtError()

{

cmbAddCustTitle.BackColor = Color.White;

txtAddCustForename.BackColor = Color.White;

txtAddCustSurname.BackColor = Color.White;

txtAddCustStreet.BackColor = Color.White;

txtAddCustTown.BackColor = Color.White;

txtAddCustCounty.BackColor = Color.White;

txtAddCustPostcode.BackColor = Color.White;

txtAddCustEmail.BackColor = Color.White;

txtAddCustTelNum.BackColor = Color.White;

cmbAddCustCreditRating.BackColor = Color.White;

}

// Add Form Error Feedback - TextBox color changes to red if invalid //

private void txtAddCustForename\_TextChanged(object sender, EventArgs e)

{

if (txtAddCustForename.Text.Length >= 2 && txtAddCustForename.Text.Length <= 15)

{

txtAddCustForename.BackColor = Color.White;

}

else

{

txtAddCustForename.BackColor = Color.LightCoral;

}

}

private void txtAddCustSurname\_TextChanged(object sender, EventArgs e)

{

if (txtAddCustSurname.Text.Length >= 2 && txtAddCustSurname.Text.Length <= 15)

{

txtAddCustSurname.BackColor = Color.White;

}

else

{

txtAddCustSurname.BackColor = Color.LightCoral;

}

}

private void txtAddCustStreet\_TextChanged(object sender, EventArgs e)

{

if (txtAddCustStreet.Text.Length >= 5 && txtAddCustStreet.Text.Length <= 40)

{

txtAddCustStreet.BackColor = Color.White;

}

else

{

txtAddCustStreet.BackColor = Color.LightCoral;

}

}

private void txtAddCustTown\_TextChanged(object sender, EventArgs e)

{

if (txtAddCustTown.Text.Length >= 2 && txtAddCustTown.Text.Length <= 20)

{

txtAddCustTown.BackColor = Color.White;

}

else

{

txtAddCustTown.BackColor = Color.LightCoral;

}

}

private void txtAddCustCounty\_TextChanged(object sender, EventArgs e)

{

if (txtAddCustCounty.Text.Length >= 2 && txtAddCustCounty.Text.Length <= 20)

{

txtAddCustCounty.BackColor = Color.White;

}

else

{

txtAddCustCounty.BackColor = Color.LightCoral;

}

}

private void txtAddCustPostcode\_TextChanged(object sender, EventArgs e)

{

if (txtAddCustPostcode.Text.Length >= 7 && txtAddCustPostcode.Text.Length <= 8)

{

txtAddCustPostcode.BackColor = Color.White;

}

else

{

txtAddCustPostcode.BackColor = Color.LightCoral;

}

}

private void txtAddCustEmail\_TextChanged(object sender, EventArgs e)

{

if (txtAddCustEmail.Text.Length >= 7 && txtAddCustPostcode.Text.Length <= 25)

{

txtAddCustEmail.BackColor = Color.White;

}

else

{

txtAddCustEmail.BackColor = Color.LightCoral;

}

}

private void txtAddCustTelNum\_TextChanged(object sender, EventArgs e)

{

if (txtAddCustTelNum.Text.Length >= 11 && txtAddCustTelNum.Text.Length <= 15)

{

txtAddCustTelNum.BackColor = Color.White;

}

else

{

txtAddCustTelNum.BackColor = Color.LightCoral;

}

}

// Edit Form Error Feedback - TextBox color changes to red if invalid //

private void txtEditCustForename\_TextChanged(object sender, EventArgs e)

{

if (txtEditCustForename.Text.Length >= 2 && txtEditCustForename.Text.Length <= 15)

{

txtEditCustForename.BackColor = Color.White;

}

else

{

txtEditCustForename.BackColor = Color.LightCoral;

}

}

private void txtEditCustSurname\_TextChanged\_1(object sender, EventArgs e)

{

if (txtEditCustSurname.Text.Length >= 2 && txtEditCustSurname.Text.Length <= 15)

{

txtEditCustSurname.BackColor = Color.White;

}

else

{

txtEditCustSurname.BackColor = Color.LightCoral;

}

}

private void txtEditCustStreet\_TextChanged\_1(object sender, EventArgs e)

{

if (txtEditCustStreet.Text.Length >= 5 && txtEditCustStreet.Text.Length <= 40)

{

txtEditCustStreet.BackColor = Color.White;

}

else

{

txtEditCustStreet.BackColor = Color.LightCoral;

}

}

private void txtEditCustTown\_TextChanged\_1(object sender, EventArgs e)

{

if (txtEditCustTown.Text.Length >= 2 && txtEditCustTown.Text.Length <= 20)

{

txtEditCustTown.BackColor = Color.White;

}

else

{

txtEditCustTown.BackColor = Color.LightCoral;

}

}

private void txtEditCustCounty\_TextChanged\_1(object sender, EventArgs e)

{

if (txtEditCustCounty.Text.Length >= 2 && txtEditCustCounty.Text.Length <= 20)

{

txtEditCustCounty.BackColor = Color.White;

}

else

{

txtEditCustCounty.BackColor = Color.LightCoral;

}

}

private void txtEditCustPostcode\_TextChanged\_1(object sender, EventArgs e)

{

if (txtEditCustPostcode.Text.Length >= 7 && txtEditCustPostcode.Text.Length <= 8)

{

txtEditCustPostcode.BackColor = Color.White;

}

else

{

txtEditCustPostcode.BackColor = Color.LightCoral;

}

}

private void txtEditCustEmail\_TextChanged\_1(object sender, EventArgs e)

{

if (txtEditCustEmail.Text.Length >= 7 && txtEditCustEmail.Text.Length <= 25)

{

txtEditCustEmail.BackColor = Color.White;

}

else

{

txtEditCustEmail.BackColor = Color.LightCoral;

}

}

private void txtEditCustTelNum\_TextChanged\_1(object sender, EventArgs e)

{

if (txtEditCustTelNum.Text.Length >= 11 && txtEditCustTelNum.Text.Length <= 15)

{

txtEditCustTelNum.BackColor = Color.White;

}

else

{

txtEditCustTelNum.BackColor = Color.LightCoral;

}

}

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

{

GlobalVar.tradeInCar = false;

this.Hide();

frmMainMenu menuForm = new frmMainMenu();

menuForm.ShowDialog();

}

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

{

selectedTab = tabCustomer.SelectedIndex;

tabCustomer.TabPages[tabCustomer.SelectedIndex].Focus();

tabCustomer.TabPages[tabCustomer.SelectedIndex].CausesValidation = true;

switch (tabCustomer.SelectedIndex)

{

case 0:

{

dsDesmonds.Tables["Customer"].Clear();

daCustomer.Fill(dsDesmonds, "Customer");

break;

}

case 1:

{

int noRows = dsDesmonds.Tables["Customer"].Rows.Count;

if (noRows == 0)

{

lblAddCustNoShow.Text = "CN100000";

}

else

{

getTotalNum(noRows);

}

txtAddCustPostcode.Enabled = true;

errP.Clear();

clearAddForm();

break;

}

case 2:

{

if (custNoSelected == "")

{

tabCustomer.SelectedIndex = 0;

}

else

{

lblEditCustNoShow.Text = custNoSelected.ToString();

drCustomer = dsDesmonds.Tables["Customer"].Rows.Find(lblEditCustNoShow.Text);

if (drCustomer["CustTitle"].ToString() == "Mr")

{

cmbEditCustTitle.SelectedIndex = 0;

}

else if (drCustomer["CustTitle"].ToString() == "Mrs")

{

cmbEditCustTitle.SelectedIndex = 1;

}

else if (drCustomer["CustTitle"].ToString() == "Miss")

{

cmbEditCustTitle.SelectedIndex = 2;

}

else if (drCustomer["CustTitle"].ToString() == "Ms")

{

cmbEditCustTitle.SelectedIndex = 3;

}

txtEditCustForename.Text = drCustomer["CustForename"].ToString();

txtEditCustSurname.Text = drCustomer["CustSurname"].ToString();

txtEditCustStreet.Text = drCustomer["CustStreet"].ToString();

txtEditCustTown.Text = drCustomer["CustTown"].ToString();

txtEditCustCounty.Text = drCustomer["CustCounty"].ToString();

txtEditCustPostcode.Text = drCustomer["CustPostcode"].ToString();

txtEditCustEmail.Text = drCustomer["CustEmail"].ToString();

txtEditCustTelNum.Text = drCustomer["TelNo"].ToString();

if (drCustomer["CreditRating"].ToString() == "Poor")

{

cmbEditCustCreditRating.SelectedIndex = 0;

}

else if (drCustomer["CreditRating"].ToString() == "Very Poor")

{

cmbEditCustCreditRating.SelectedIndex = 1;

}

else if (drCustomer["CreditRating"].ToString() == "Average")

{

cmbEditCustCreditRating.SelectedIndex = 2;

}

else if (drCustomer["CreditRating"].ToString() == "Good")

{

cmbEditCustCreditRating.SelectedIndex = 3;

}

else if (drCustomer["CreditRating"].ToString() == "Very Good")

{

cmbEditCustCreditRating.SelectedIndex = 4;

}

else if (drCustomer["CreditRating"].ToString() == "Excellent")

{

cmbEditCustCreditRating.SelectedIndex = 5;

}

}

break;

}

}

}

}

}

# Car Form Code

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ScreenDesigns

{

public partial class frmCar : Form

{

SqlDataAdapter daCar, daMake, daSearch, daOrderDetails, daFinanceAgreement2;

DataSet dsDesmonds = new DataSet();

DataTable dtDesmonds = new DataTable();

SqlCommandBuilder cmdBCar, cmdBMake, cmdBSearch, cmdBOrderDet, cmdBFinanceAgreement2;

DataRow drCar;

String connStr, sqlCar, sqlMake, sqlOrderDetails, sqlFinanceDetails2;

int selectedTab = 0, tabCount = 0;

bool carSelected = false;

string carRegSelected = "", connSearch;

Color clrTheme = Color.FromArgb(50, 115, 165);

public frmCar()

{

InitializeComponent();

dtpAddCarYear.Format = DateTimePickerFormat.Custom;

dtpAddCarYear.CustomFormat = "yyyy";

dtpAddCarYear.ShowUpDown = true;

dtpEditCarYear.Format = DateTimePickerFormat.Custom;

dtpEditCarYear.CustomFormat = "yyyy";

dtpEditCarYear.ShowUpDown = true;

}

private void frmCar\_Shown(object sender, EventArgs e)

{

tabCar.TabPages[0].CausesValidation = true;

tabCar.TabPages[0].Validating += new CancelEventHandler(AddTabValidate);

}

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

{

connStr = @"Data Source = .; Initial Catalog = desmonds; Integrated Security = true";

sqlCar = @"select \* from car";

daCar = new SqlDataAdapter(sqlCar, connStr);

cmdBCar = new SqlCommandBuilder(daCar);

daCar.FillSchema(dsDesmonds, SchemaType.Source, "Car");

daCar.Fill(dsDesmonds, "Car"); // Naming the table in VS //

sqlMake = @"select DISTINCT MakeID, MakeDesc from make";

daMake = new SqlDataAdapter(sqlMake, connStr);

cmdBMake = new SqlCommandBuilder(daMake);

daMake.FillSchema(dsDesmonds, SchemaType.Source, "Make");

daMake.Fill(dsDesmonds, "Make"); // Naming the table in VS //

sqlOrderDetails = @"SELECT \* FROM orderDetails";

daOrderDetails = new SqlDataAdapter(sqlOrderDetails, connStr);

cmdBOrderDet = new SqlCommandBuilder(daOrderDetails);

daOrderDetails.FillSchema(dsDesmonds, SchemaType.Source, "OrderDetails");

daOrderDetails.Fill(dsDesmonds, "OrderDetails");

sqlFinanceDetails2 = @"SELECT \* FROM financeAgreement";

daFinanceAgreement2 = new SqlDataAdapter(sqlFinanceDetails2, connStr);

cmdBFinanceAgreement2 = new SqlCommandBuilder(daFinanceAgreement2);

daFinanceAgreement2.FillSchema(dsDesmonds, SchemaType.Source, "financeAgreement2");

daFinanceAgreement2.Fill(dsDesmonds, "financeAgreement2");

dgvCar.DataSource = dsDesmonds.Tables["Car"];

// Resize the dgv columns to fit the newly loaded content //

dgvCar.AutoResizeColumns(DataGridViewAutoSizeColumnsMode.AllCells);

cmbAddCarMake.DataSource = dsDesmonds.Tables["Make"];

cmbAddCarMake.DisplayMember = "MakeDesc";

cmbAddCarMake.ValueMember = "MakeID";

tabCar.SelectedIndex = 1;

tabCar.SelectedIndex = 0;

cmbAddCarMake.DropDownStyle = ComboBoxStyle.DropDownList;

cmbAddCarTransmission.DropDownStyle = ComboBoxStyle.DropDownList;

cmbAddCarBody.DropDownStyle = ComboBoxStyle.DropDownList;

cmbAddCarCondition.DropDownStyle = ComboBoxStyle.DropDownList;

cmbAddCarDoorNo.DropDownStyle = ComboBoxStyle.DropDownList;

cmbEditCarMake.DropDownStyle = ComboBoxStyle.DropDownList;

cmbEditCarTransmission.DropDownStyle = ComboBoxStyle.DropDownList;

cmbEditCarBody.DropDownStyle = ComboBoxStyle.DropDownList;

cmbEditCarCondition.DropDownStyle = ComboBoxStyle.DropDownList;

cmbEditCarDoorNo.DropDownStyle = ComboBoxStyle.DropDownList;

}

// For Button Click events //

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

{

tabCar.SelectedIndex = 1;

}

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

{

tabCar.SelectedIndex = 2;

}

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

{

if (dgvCar.SelectedRows.Count == 0)

{

MessageBox.Show("Please select a car from the list", "Car");

}

else

{

bool okDelete = true;

foreach (DataRow dr in dsDesmonds.Tables["orderDetails"].Rows)

{

if (dr["regNo"].ToString() == dgvCar.SelectedRows[0].Cells[0].Value.ToString())

{

okDelete = false;

}

}

foreach (DataRow dr in dsDesmonds.Tables["financeAgreement2"].Rows)

{

if (dr["regNo"].ToString() == dgvCar.SelectedRows[0].Cells[0].Value.ToString())

{

okDelete = false;

}

}

if (okDelete == true)

{

drCar = dsDesmonds.Tables["Car"].Rows.Find(dgvCar.SelectedRows[0].Cells[0].Value);

string tempName = drCar["MakeID"].ToString() + " " + drCar["Model"].ToString() + "\'s";

if (MessageBox.Show("Are you sure you want to delete " + tempName + " details?", "Delete Car", MessageBoxButtons.YesNo) == System.Windows.Forms.DialogResult.Yes)

{

drCar.Delete();

daCar.Update(dsDesmonds, "Car");

}

}

else

MessageBox.Show("Car can't be deleted as it is linked to an order and/or Finance Agreement");

}

}

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

{

System.Windows.Forms.Application.Exit();

}

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

{

if (lblEditCarEdit.Text == "Edit")

{

txtAddCarReg.Enabled = true;

cmbEditCarMake.Enabled = true;

txtEditCarModel.Enabled = true;

txtEditCarInterior.Enabled = true;

cmbEditCarTransmission.Enabled = true;

cmbEditCarBody.Enabled = true;

txtEditCarColour.Enabled = true;

cmbEditCarCondition.Enabled = true;

dtpEditCarYear.Enabled = true;

cmbEditCarDoorNo.Enabled = true;

txtEditCarMileage.Enabled = true;

txtEditCarPurchasePrice.Enabled = true;

txtEditCarSalePrice.Enabled = true;

lblEditCarEdit.Text = "Save";

}

else

{

MyCar myCar = new MyCar();

bool Ok = true;

errP.Clear();

try

{

myCar.RegNo = txtEditCarRegNo.Text.Trim(); // passed to car class to check //

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtEditCarRegNo, MyEx.toString());

}

try

{

myCar.MakeID = cmbEditCarMake.Text.Trim(); // passed to car class to check //

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(cmbEditCarMake, MyEx.toString());

}

try

{

myCar.Model = txtEditCarModel.Text.Trim(); // passed to car class to check //

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtEditCarModel, MyEx.toString());

}

try

{

myCar.Interior = txtEditCarInterior.Text.Trim(); // passed to car class to check //

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtEditCarInterior, MyEx.toString());

}

try

{

myCar.Transmission = cmbEditCarTransmission.Text.Trim(); // passed to car class to check //

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(cmbEditCarTransmission, MyEx.toString());

}

try

{

myCar.BodyStyle = cmbEditCarBody.Text.Trim(); // passed to car class to check //

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(cmbEditCarBody, MyEx.toString());

}

try

{

myCar.Colour = txtEditCarColour.Text.Trim(); // passed to car class to check //

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtEditCarColour, MyEx.toString());

}

try

{

myCar.Condition = cmbEditCarCondition.Text.Trim(); // passed to car class to check //

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(cmbEditCarCondition, MyEx.toString());

}

try

{

myCar.ModelYear = dtpEditCarYear.Text.Trim(); // passed to car class to check //

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(dtpEditCarYear, MyEx.toString());

}

try

{

myCar.DoorNo = cmbEditCarDoorNo.Text.Trim(); // passed to car class to check //

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(cmbEditCarDoorNo, MyEx.toString());

}

try

{

myCar.Mileage = int.Parse(txtEditCarMileage.Text.Trim());

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtEditCarMileage, MyEx.toString());

}

try

{

myCar.PurchasePrice = Double.Parse(txtEditCarPurchasePrice.Text.Trim()); // passed to car class to check //

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtEditCarPurchasePrice, MyEx.toString());

}

try

{

myCar.SalePrice = Double.Parse(txtEditCarSalePrice.Text.Trim()); // passed to car class to check //

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtEditCarSalePrice, MyEx.toString());

}

try

{

if (Ok)

{

drCar.BeginEdit();

drCar["RegNo"] = myCar.RegNo;

drCar["MakeID"] = myCar.MakeID;

drCar["Model"] = myCar.Model;

drCar["SalePrice"] = myCar.SalePrice;

drCar["Interior"] = myCar.Interior;

drCar["Transmission"] = myCar.Transmission;

drCar["BodyStyle"] = myCar.BodyStyle;

drCar["Colour"] = myCar.Colour;

drCar["Mileage"] = myCar.Mileage;

drCar["Condition"] = myCar.Condition;

drCar["ModelYear"] = myCar.ModelYear;

drCar["PurchasePrice"] = myCar.PurchasePrice;

drCar["DoorNo"] = myCar.DoorNo;

drCar.EndEdit();

daCar.Update(dsDesmonds, "Car");

MessageBox.Show("Car Details Updated", "Car");

MessageBox.Show("Car Edit Complete");

cmbEditCarMake.Enabled = false;

txtEditCarModel.Enabled = false;

txtEditCarInterior.Enabled = false;

cmbEditCarTransmission.Enabled = false;

cmbEditCarBody.Enabled = false;

txtEditCarColour.Enabled = false;

cmbEditCarCondition.Enabled = false;

dtpEditCarYear.Enabled = false;

cmbEditCarDoorNo.Enabled = false;

txtEditCarMileage.Enabled = false;

txtEditCarPurchasePrice.Enabled = false;

txtEditCarSalePrice.Enabled = false;

lblEditCarEdit.Text = "Edit";

tabCar.SelectedIndex = 0;

}

}

catch (Exception ex)

{

MessageBox.Show("" + ex.TargetSite + "" + ex.Message, "Error!", MessageBoxButtons.AbortRetryIgnore, MessageBoxIcon.Error);

}

}

}

// Custom Methods //

void clearAddForm()

{

txtAddCarReg.Clear();

cmbAddCarMake.SelectedIndex = -1;

txtAddCarModel.Clear();

txtAddCarInterior.Clear();

cmbAddCarTransmission.SelectedIndex = -1;

cmbAddCarBody.SelectedIndex = -1;

txtAddCarColour.Clear();

cmbAddCarCondition.SelectedIndex = -1;

dtpAddCarYear.ResetText();

cmbAddCarDoorNo.SelectedIndex = -1;

txtAddCarMileage.ResetText();

txtAddCarPurchasePrice.Clear();

txtAddCarSalePrice.Clear();

}

void AddTabValidate(object sender, CancelEventArgs e)

{

if (dgvCar.SelectedRows.Count == 0)

{

carSelected = false;

carRegSelected = "";

}

else if (dgvCar.SelectedRows.Count == 1)

{

carSelected = true;

carRegSelected = dgvCar.SelectedRows[0].Cells[0].Value.ToString();

}

}

void EditTabValidate(object sender, CancelEventArgs e)

{

if (carSelected == false && carRegSelected == "")

{

// have to do this bit //

// reset tab to display and put out a message to select customer

carSelected = false;

carRegSelected = "";

}

else if (dgvCar.SelectedRows.Count == 1)

{

carSelected = true;

carRegSelected = dgvCar.SelectedRows[0].Cells[0].ToString();

}

}

void clearTxtError()

{

txtAddCarReg.BackColor = Color.White;

cmbAddCarMake.BackColor = Color.White;

txtAddCarModel.BackColor = Color.White;

txtAddCarInterior.BackColor = Color.White;

cmbAddCarTransmission.BackColor = Color.White;

cmbAddCarBody.BackColor = Color.White;

txtAddCarColour.BackColor = Color.White;

cmbAddCarCondition.BackColor = Color.White;

dtpAddCarYear.BackColor = Color.White;

cmbAddCarDoorNo.BackColor = Color.White;

txtAddCarMileage.BackColor = Color.White;

txtAddCarPurchasePrice.BackColor = Color.White;

txtAddCarSalePrice.BackColor = Color.White;

}

void searchString()

{

connSearch = (@"select \* from car where RegNo LIKE '%" + txtSearchMake.Text + "%'");

daSearch = new SqlDataAdapter(connSearch, connStr);

cmdBSearch = new SqlCommandBuilder(daSearch);

dsDesmonds.Tables["Car"].Clear();

daSearch.FillSchema(dsDesmonds, SchemaType.Source, "Car");

daSearch.Fill(dsDesmonds, "Car"); // Naming the table in VS /

dgvCar.DataSource = dsDesmonds.Tables["Car"];

}

// Add Form Error Feedback - TextBox color changes to red if invalid //

private void txtAddCarReg\_TextChanged(object sender, EventArgs e)

{

if (txtAddCarReg.Text.Length == 7)

{

txtAddCarReg.BackColor = Color.White;

}

else

{

txtAddCarReg.BackColor = clrTheme;

}

}

private void cmbAddCarMake\_TextChanged(object sender, EventArgs e)

{

if (cmbAddCarMake.Text.Length == 0)

{

cmbAddCarMake.BackColor = Color.White;

}

else

{

cmbAddCarMake.BackColor = clrTheme;

}

}

private void txtAddCarModel\_TextChanged(object sender, EventArgs e)

{

if (txtAddCarModel.Text.Length >= 2 && txtAddCarModel.Text.Length <= 20)

{

txtAddCarModel.BackColor = Color.White;

}

else

{

txtAddCarModel.BackColor = clrTheme;

}

}

private void txtAddCarInterior\_TextChanged(object sender, EventArgs e)

{

if (txtAddCarInterior.Text.Length >= 3 && txtAddCarInterior.Text.Length <= 20)

{

txtAddCarInterior.BackColor = Color.White;

}

else

{

txtAddCarInterior.BackColor = clrTheme;

}

}

private void cmbAddCarTransmission\_TextChanged(object sender, EventArgs e)

{

if (cmbAddCarTransmission.Text.Length < 6 && cmbAddCarTransmission.Text.Length > 9)

{

cmbAddCarTransmission.BackColor = Color.White;

}

else

{

cmbAddCarTransmission.BackColor = clrTheme;

}

}

private void cmbAddCarBody\_TextChanged(object sender, EventArgs e)

{

if (cmbAddCarBody.Text.Length < 3 && cmbAddCarBody.Text.Length > 20)

{

cmbAddCarBody.BackColor = Color.White;

}

else

{

cmbAddCarBody.BackColor = clrTheme;

}

}

private void txtAddCarColour\_TextChanged(object sender, EventArgs e)

{

if (txtAddCarColour.Text.Length >= 3 && txtAddCarColour.Text.Length <= 20)

{

txtAddCarColour.BackColor = Color.White;

}

else

{

txtAddCarColour.BackColor = clrTheme;

}

}

private void cmbAddCarCondition\_TextChanged(object sender, EventArgs e)

{

if (cmbAddCarCondition.Text.Length >= 3 && cmbAddCarCondition.Text.Length <= 5)

{

cmbAddCarCondition.BackColor = Color.White;

}

else

{

cmbAddCarCondition.BackColor = clrTheme;

}

}

private void cmbAddCarDoorNo\_TextChanged(object sender, EventArgs e)

{

if (cmbAddCarDoorNo.SelectedIndex == -1)

{

cmbAddCarDoorNo.BackColor = Color.White;

}

else

{

cmbAddCarDoorNo.BackColor = clrTheme;

}

}

private void txtAddCarMileage\_TextChanged(object sender, EventArgs e)

{

if (int.Parse(txtAddCarMileage.Text) > 0)

{

txtAddCarMileage.BackColor = Color.White;

}

else

{

txtAddCarMileage.BackColor = clrTheme;

}

}

private void txtAddCarPurchasePrice\_TextChanged(object sender, EventArgs e)

{

if (int.Parse(txtAddCarPurchasePrice.Text) > 0)

{

txtAddCarPurchasePrice.BackColor = Color.White;

}

else

{

txtAddCarPurchasePrice.BackColor = clrTheme;

}

}

private void txtAddCarSalePrice\_TextChanged(object sender, EventArgs e)

{

if (int.Parse(txtAddCarSalePrice.Text) > 0)

{

txtAddCarSalePrice.BackColor = Color.White;

}

else

{

txtAddCarSalePrice.BackColor = clrTheme;

}

}

// Edit Form Error Feedback - TextBox color changes to red if invalid //

private void txtEditCarReg\_TextChanged(object sender, EventArgs e)

{

if (txtAddCarReg.Text.Length == 7)

{

txtAddCarReg.BackColor = Color.White;

}

else

{

txtAddCarReg.BackColor = clrTheme;

}

}

private void cmbEditCarMake\_TextChanged(object sender, EventArgs e)

{

if (cmbAddCarMake.Text.Length == 3)

{

cmbAddCarMake.BackColor = Color.White;

}

else

{

cmbAddCarMake.BackColor = clrTheme;

}

}

private void txtEditCarModel\_TextChanged(object sender, EventArgs e)

{

if (txtAddCarModel.Text.Length >= 1 || txtAddCarModel.Text.Length <= 20)

{

txtAddCarModel.BackColor = Color.White;

}

else

{

cmbAddCarMake.BackColor = clrTheme;

}

}

private void txtEditCarInterior\_TextChanged(object sender, EventArgs e)

{

if (txtAddCarInterior.Text.Length >= 3 || txtAddCarInterior.Text.Length <= 20)

{

txtAddCarInterior.BackColor = Color.White;

}

else

{

txtAddCarInterior.BackColor = clrTheme;

}

}

private void cmbEditCarTransmission\_TextChanged(object sender, EventArgs e)

{

if (cmbAddCarTransmission.Text.Length >= 6 || cmbAddCarTransmission.Text.Length <= 9)

{

cmbAddCarTransmission.BackColor = Color.White;

}

else

{

cmbAddCarTransmission.BackColor = clrTheme;

}

}

private void cmbEditCarBody\_TextChanged(object sender, EventArgs e)

{

if (cmbAddCarBody.Text.Length >= 3 || cmbAddCarBody.Text.Length <= 20)

{

cmbAddCarBody.BackColor = Color.White;

}

else

{

cmbAddCarBody.BackColor = clrTheme;

}

}

private void txtEditCarColour\_TextChanged(object sender, EventArgs e)

{

if (txtAddCarColour.Text.Length >= 3 || txtAddCarColour.Text.Length <= 20)

{

txtAddCarColour.BackColor = Color.White;

}

else

{

txtAddCarColour.BackColor = clrTheme;

}

}

private void cmbEditCarCondition\_TextChanged(object sender, EventArgs e)

{

if (cmbAddCarCondition.Text.Length >= 3 || cmbAddCarCondition.Text.Length <= 5)

{

cmbAddCarCondition.BackColor = Color.White;

}

else

{

cmbAddCarCondition.BackColor = clrTheme;

}

}

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

{

GlobalVar.tradeInCar = false;

this.Hide();

frmMainMenu menuForm = new frmMainMenu();

menuForm.ShowDialog();

}

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

{

MyCar myCar = new MyCar();

bool Ok = true;

String invalMessage = "Invalid data entry for: ";

errP.Clear();

foreach (DataRow dr in dsDesmonds.Tables["Car"].Rows)

{

if (dr["regNo"].ToString() == txtAddCarReg.Text.Trim())

{

Ok = false;

errP.SetError(txtAddCarReg, ("RegNo already exists, please enter another"));

}

}

try

{

myCar.RegNo = txtAddCarReg.Text.Trim(); // passed to car class to check //

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtAddCarReg, MyEx.toString());

}

try

{

if (cmbAddCarMake.SelectedIndex != -1)

{

myCar.MakeID = cmbAddCarMake.SelectedValue.ToString(); // passed to car class to check //

}

else

throw new MyException("Please select a car Make");

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(cmbAddCarMake, MyEx.toString());

}

try

{

myCar.Model = txtAddCarModel.Text.Trim(); // passed to car class to check //

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtAddCarModel, MyEx.toString());

}

try

{

myCar.Interior = txtAddCarInterior.Text.Trim(); // passed to car class to check //

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtAddCarInterior, MyEx.toString());

}

try

{

myCar.Transmission = cmbAddCarTransmission.Text.Trim(); // passed to car class to check //

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(cmbAddCarTransmission, MyEx.toString());

}

try

{

myCar.BodyStyle = cmbAddCarBody.Text.Trim(); // passed to car class to check //

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(cmbAddCarBody, MyEx.toString());

}

try

{

myCar.Colour = txtAddCarColour.Text.Trim(); // passed to car class to check //

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtAddCarColour, MyEx.toString());

}

try

{

myCar.Condition = cmbAddCarCondition.Text.Trim(); // passed to car class to check //

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(cmbAddCarCondition, MyEx.toString());

try

{

if (int.Parse(dtpAddCarYear.Text.Trim()) > 1900 || int.Parse(dtpAddCarYear.Text.Trim()) < DateTime.Now.Year)

{

myCar.ModelYear = dtpAddCarYear.Text.Trim(); // passed to car class to check //

}

else

{

throw new MyException("Please enter a year between 1900 and " + DateTime.Now.Year.ToString());

}

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(dtpAddCarYear, MyEx.toString());

}

try

{

myCar.DoorNo = cmbAddCarDoorNo.Text.Trim(); // passed to car class to check //

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(cmbAddCarDoorNo, MyEx.toString());

}

try

{

myCar.Mileage = int.Parse(txtAddCarMileage.Text.Trim()); // passed to car class to check //

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtAddCarMileage, MyEx.toString());

}

catch (System.FormatException)

{

Ok = false;

errP.SetError(txtAddCarMileage, "Invalid Mileage");

}

try

{

myCar.PurchasePrice = Double.Parse(txtAddCarPurchasePrice.Text.Trim()); // passed to car class to check //

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtAddCarPurchasePrice, MyEx.toString());

}

catch (System.FormatException)

{

Ok = false;

errP.SetError(txtAddCarPurchasePrice, "Invalid Purchase Price");

}

try

{

myCar.SalePrice = Double.Parse(txtAddCarSalePrice.Text); // passed to car class to check //

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(txtAddCarSalePrice, MyEx.toString());

}

catch (System.FormatException)

{

Ok = false;

errP.SetError(txtAddCarSalePrice, "Invalid Sale Price");

}

try

{

if (Ok)

{

drCar = dsDesmonds.Tables["Car"].NewRow();

drCar["RegNo"] = myCar.RegNo;

drCar["MakeID"] = myCar.MakeID;

drCar["Model"] = myCar.Model;

drCar["SalePrice"] = myCar.SalePrice;

drCar["Interior"] = myCar.Interior;

drCar["Transmission"] = myCar.Transmission;

drCar["BodyStyle"] = myCar.BodyStyle;

drCar["Colour"] = myCar.Colour;

drCar["Mileage"] = myCar.Mileage;

drCar["Condition"] = myCar.Condition;

drCar["ModelYear"] = myCar.ModelYear;

drCar["PurchasePrice"] = myCar.PurchasePrice;

drCar["DoorNo"] = myCar.DoorNo;

drCar["Sold"] = 0;

dsDesmonds.Tables["Car"].Rows.Add(drCar);

daCar.Update(dsDesmonds, "Car");

if (GlobalVar.tradeInCar == false)

{

MessageBox.Show("Car Added");

}

else if ((GlobalVar.tradeInCar == true))

{

MessageBox.Show("Trade In Value Saved and Car has been Added");

}

if (GlobalVar.tradeInCar == false)

{

if (MessageBox.Show("Do you want to add another Car?", "Add Car", MessageBoxButtons.YesNo) == System.Windows.Forms.DialogResult.Yes)

{

errP.Clear();

clearAddForm();

clearTxtError();

}

else

tabCar.SelectedIndex = 0;

}

else if (GlobalVar.tradeInCar == true)

{

this.Close();

GlobalVar.tradeInFormClose = true;

}

}

}

catch (Exception ex)

{

MessageBox.Show("" + ex.TargetSite + "" + ex.Message, "Error!", MessageBoxButtons.AbortRetryIgnore, MessageBoxIcon.Error);

}

if (GlobalVar.tradeInCar == true)

{

GlobalVar.purchasePriceGlobal = int.Parse(txtAddCarPurchasePrice.Text);

this.Close();

}

}

private void txtSearchSurname\_TextChanged(object sender, EventArgs e)

{

searchString();

}

private void cmbEditCarDoorNo\_TextChanged(object sender, EventArgs e)

{

if (cmbAddCarDoorNo.Text == "3" || cmbAddCarDoorNo.Text == "5")

{

cmbAddCarDoorNo.BackColor = Color.White;

}

else

{

cmbAddCarDoorNo.BackColor = clrTheme;

}

}

private void txtEditCarMileage\_TextChanged(object sender, EventArgs e)

{

if (txtAddCarMileage.Text.Length > 0)

{

if (int.Parse(txtAddCarMileage.Text) < 1)

{

txtAddCarMileage.BackColor = Color.White;

}

else

{

txtAddCarMileage.BackColor = clrTheme;

}

}

}

private void txtEditCarPurchasePrice\_TextChanged(object sender, EventArgs e)

{

if (txtEditCarPurchasePrice.Text.Length > 0)

{

if (double.Parse(txtEditCarPurchasePrice.Text) > 0)

{

txtEditCarPurchasePrice.BackColor = Color.White;

}

else

{

txtEditCarPurchasePrice.BackColor = clrTheme;

}

}

}

private void txtEditCarSalePrice\_TextChanged(object sender, EventArgs e)

{

if (double.Parse(txtEditCarSalePrice.Text) > 0)

{

txtEditCarSalePrice.BackColor = Color.White;

}

else

{

txtEditCarSalePrice.BackColor = clrTheme;

}

}

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

{

if (GlobalVar.tradeInCar == true)

{

if (tabCar.SelectedIndex == 0 || tabCar.SelectedIndex == 2)

{

tabCar.SelectedIndex = 1;

if (tabCount >= 1)

{

MessageBox.Show("Error: You cannot view or edit other entries at this time. Please enter trade in details", "Enter Trade In Details");

}

tabCount++;

}

}

selectedTab = tabCar.SelectedIndex;

tabCar.TabPages[tabCar.SelectedIndex].Focus();

tabCar.TabPages[tabCar.SelectedIndex].CausesValidation = true;

switch (tabCar.SelectedIndex)

{

case 0:

{

txtSearchMake.Text = "";

dsDesmonds.Tables["Car"].Clear();

daCar.Fill(dsDesmonds, "Car");

break;

}

case 1:

{

txtSearchMake.Text = "";

int noRows = dsDesmonds.Tables["Car"].Rows.Count;

errP.Clear();

clearAddForm();

clearTxtError();

break;

}

case 2:

{

txtSearchMake.Text = "";

if (carRegSelected == "")

{

tabCar.SelectedIndex = 0;

}

else

{

// Sets RegNo textbox on edit page equal to the selected records //

txtEditCarRegNo.Text = carRegSelected.ToString();

// Finds the car using the selected records regNo //

drCar = dsDesmonds.Tables["Car"].Rows.Find(txtEditCarRegNo.Text);

// Checks to see which Make ID code matches the selected record and changes the combo box to the appropriate index //

if (drCar["MakeID"].ToString() == "FRD")

{

cmbEditCarMake.SelectedIndex = 0;

}

else if (drCar["MakeID"].ToString() == "REN")

{

cmbEditCarMake.SelectedIndex = 1;

}

else if (drCar["MakeID"].ToString() == "AUD")

{

cmbEditCarMake.SelectedIndex = 2;

}

else if (drCar["MakeID"].ToString() == "NIS")

{

cmbEditCarMake.SelectedIndex = 3;

}

else if (drCar["MakeID"].ToString() == "HON")

{

cmbEditCarMake.SelectedIndex = 4;

}

else if (drCar["MakeID"].ToString() == "FIA")

{

cmbEditCarMake.SelectedIndex = 5;

}

// Sets model textbox on edit page equal to the selected records //

txtEditCarModel.Text = drCar["Model"].ToString();

// Sets interior textbox on edit page equal to the selected records //

txtEditCarInterior.Text = drCar["Interior"].ToString();

// Checks to see which transmission matches the selected record and changes the combo box to the appropriate index //

if (drCar["Transmission"].ToString() == "Manual")

{

cmbEditCarTransmission.SelectedIndex = 0;

}

else if (drCar["Transmission"].ToString() == "Automatic")

{

cmbEditCarTransmission.SelectedIndex = 1;

}

// Checks to see which Body Style matches the selected record and changes the combo box to the appropriate index //

if (drCar["BodyStyle"].ToString() == "Sedan")

{

cmbEditCarBody.SelectedIndex = 0;

}

else if (drCar["BodyStyle"].ToString() == "Coupe")

{

cmbEditCarBody.SelectedIndex = 1;

}

else if (drCar["BodyStyle"].ToString() == "SUV")

{

cmbEditCarBody.SelectedIndex = 2;

}

else if (drCar["BodyStyle"].ToString() == "Hatchback")

{

cmbEditCarBody.SelectedIndex = 3;

}

else if (drCar["BodyStyle"].ToString() == "Sport")

{

cmbEditCarBody.SelectedIndex = 4;

}

else if (drCar["BodyStyle"].ToString() == "Van")

{

cmbEditCarBody.SelectedIndex = 5;

}

// Sets colour textbox on edit page equal to the selected records //

txtEditCarColour.Text = drCar["Colour"].ToString();

// Checks to see which Conidition matches the selected record and changes the combo box to the appropriate index //

if (drCar["Condition"].ToString() == "New")

{

cmbEditCarCondition.SelectedIndex = 0;

}

else if (drCar["Condition"].ToString() == "Used")

{

cmbEditCarCondition.SelectedIndex = 1;

}

// How to fill year on edit page when select a car //

// Checks to see which DoorNo matches the selected record and changes the combo box to the appropriate index //

if (drCar["DoorNo"].ToString() == "3")

{

cmbEditCarDoorNo.SelectedIndex = 0;

}

else if (drCar["DoorNo"].ToString() == "5")

{

cmbEditCarDoorNo.SelectedIndex = 1;

}

// Sets Mileage textbox on edit page equal to the selected records //

txtEditCarMileage.Text = drCar["Mileage"].ToString();

// Sets Purchase Price textbox on edit page equal to the selected records //

txtEditCarPurchasePrice.Text = drCar["PurchasePrice"].ToString();

// Sets Sale Price textbox on edit page equal to the selected records //

txtEditCarSalePrice.Text = drCar["SalePrice"].ToString();

}

break;

}

}

}

}

}

# Sales Form Code

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ScreenDesigns

{

public partial class frmSales : Form

{

double carPrice = 0, carDeposit = 0, tradeValue = 0, interestRate = 0, costNoInterest = 0, costWithInterest = 0, totalInterest = 0, numPayments = 0, monthlyPayment = 0;

Button[] btns = new Button[26];

String sqlNames, sqlCustomerDetails, sqlCustomerDetails2, sqlCarDetails, sqlOrders, sqlOrderDetails, sqlFinanceDetails, sqlFinanceDetails2;

String strCarReg = "";

string financeNoSelected = "";

bool financeSelected = false;

SqlDataAdapter daNames, daCustomers, daCustomers2, daCars, daOrders, daOrderDetails, daFinanceAgreement, daFinanceAgreement2;

DataSet dsDesmonds = new DataSet();

SqlConnection conn;

String connStr;

SqlCommand cmdCustomerDetails, cmdCustomerDetails2;

SqlCommandBuilder cmdBCustomer2, cmdBOrder, cmdBOrderDet, cmdBFinanceAgreement, cmdBFinanceAgreement2, cmdBCars;

DataRow drOrder, drCustomer, drCar, drFinanceAgreement;

private void cmbFinancePlanLengthEdit\_SelectedIndexChanged\_1(object sender, EventArgs e)

{

if (cmbFinancePlanLengthEdit.SelectedIndex == 0)

{

lblFinanceNumPaymentsEdit.Text = "12";

}

else if (cmbFinancePlanLengthEdit.SelectedIndex == 1)

{

lblFinanceNumPaymentsEdit.Text = "24";

}

else if (cmbFinancePlanLengthEdit.SelectedIndex == 2)

{

lblFinanceNumPaymentsEdit.Text = "36";

}

else if (cmbFinancePlanLengthEdit.SelectedIndex == 3)

{

lblFinanceNumPaymentsEdit.Text = "48";

}

else if (cmbFinancePlanLengthEdit.SelectedIndex == 4)

{

lblFinanceNumPaymentsEdit.Text = "60";

}

object[] pk = new object[2];

pk[0] = dgvFinanceAgreements.SelectedRows[0].Cells[0].Value;

pk[1] = dgvFinanceAgreements.SelectedRows[0].Cells[1].Value;

drCar = dsDesmonds.Tables["Car"].Rows.Find(dgvFinanceAgreements.CurrentRow.Cells[1].Value.ToString());

drFinanceAgreement = dsDesmonds.Tables["FinanceAgreement2"].Rows.Find(pk);

carPrice = double.Parse(drCar["SalePrice"].ToString());

lblFinancePriceEdit.Text = carPrice.ToString();

carDeposit = double.Parse(drFinanceAgreement["Deposit"].ToString());

tradeValue = double.Parse(drFinanceAgreement["TradeInValue"].ToString());

interestRate = double.Parse(drFinanceAgreement["InterestRate"].ToString()) + 1;

costNoInterest = ((carPrice - carDeposit) - tradeValue);

costWithInterest = ((carPrice - carDeposit) - tradeValue) \* interestRate;

totalInterest = costWithInterest - costNoInterest;

numPayments = int.Parse(lblFinanceNumPaymentsEdit.Text);

monthlyPayment = costWithInterest / numPayments;

lblFinanceTotalInterestEdit.Text = costWithInterest.ToString();

lblFinanceTotalNoInterestEdit.Text = costNoInterest.ToString();

lblFinanceInterestDueEdit.Text = totalInterest.ToString();

lblFinanceMonthlyPaymentEdit.Text = monthlyPayment.ToString();

}

private void frmSales\_Activated(object sender, EventArgs e)

{

lblCarTradeValue.Text = GlobalVar.purchasePriceGlobal.ToString();

}

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

{

Environment.Exit(0);

}

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

{

tabSales.SelectedIndex = 4;

}

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

{

tabSales.SelectedIndex = 1;

}

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

{

if (dgvFinanceAgreements.SelectedRows.Count == 0)

{

MessageBox.Show("Please select a Finance Agreement from the list", "Car");

}

else

{

financeNoSelected = dgvFinanceAgreements.SelectedRows[0].Cells[0].Value.ToString();

strCarReg = dgvFinanceAgreements.SelectedRows[0].Cells[1].Value.ToString();

Object[] args = new object[2];

args[0] = financeNoSelected;

args[1] = strCarReg;

drFinanceAgreement = dsDesmonds.Tables["financeAgreement2"].Rows.Find(args);

string tempName = drFinanceAgreement["FinanceID"].ToString();

if (MessageBox.Show("Are you sure you want to delete Finance Agreement : " + tempName, "Delete FinanceAgreement", MessageBoxButtons.YesNo) == System.Windows.Forms.DialogResult.Yes)

{

drFinanceAgreement.Delete();

daFinanceAgreement2.Update(dsDesmonds, "Car");

}

}

}

private void txtFinanceDepositEdit\_ValueChanged(object sender, EventArgs e)

{

financeNoSelected = dgvFinanceAgreements.SelectedRows[0].Cells[0].Value.ToString();

strCarReg = dgvFinanceAgreements.SelectedRows[0].Cells[1].Value.ToString();

Object[] args = new object[2];

args[0] = financeNoSelected;

args[1] = strCarReg;

drCar = dsDesmonds.Tables["Car"].Rows.Find(dgvFinanceAgreements.CurrentRow.Cells[1].Value.ToString());

drFinanceAgreement = dsDesmonds.Tables["financeAgreement2"].Rows.Find(args);

carPrice = double.Parse(drCar["SalePrice"].ToString());

lblFinancePriceEdit.Text = carPrice.ToString();

carDeposit = double.Parse(txtFinanceDepositEdit.Value.ToString());

tradeValue = double.Parse(drFinanceAgreement["TradeInValue"].ToString());

interestRate = double.Parse(drFinanceAgreement["InterestRate"].ToString()) + 1;

costNoInterest = ((carPrice - carDeposit) - tradeValue);

costWithInterest = ((carPrice - carDeposit) - tradeValue) \* interestRate;

totalInterest = costWithInterest - costNoInterest;

numPayments = GlobalVar.noOfMonths;

monthlyPayment = costWithInterest / numPayments;

lblFinanceTotalInterestEdit.Text = costWithInterest.ToString("0.00");

lblFinanceTotalNoInterestEdit.Text = costNoInterest.ToString("0.00");

lblFinanceInterestDueEdit.Text = totalInterest.ToString("0.00");

lblFinanceMonthlyPaymentEdit.Text = monthlyPayment.ToString("0.00");

double check1 = 0, check2 = 0, check3 = 0, check4 = 0;

check1 = double.Parse(lblFinanceTotalInterestEdit.Text);

if (check1 <= 0)

{

MessageBox.Show("CANNOT BE 0 OR BELOW");

}

}

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

{

if (lblEditFinanceEdit.Text == "Edit")

{

cmbFinancePlanLengthEdit.Enabled = true;

txtFinanceDepositEdit.Enabled = true;

lblEditFinanceEdit.Text = "Save";

}

else

{

MyFinanceAgreement myFinanceAgreement = new MyFinanceAgreement();

bool okFinance = true;

errP.Clear();

try

{

myFinanceAgreement.FinanceID = lblFinanceFinanceIDEdit.Text.Trim(); // passed to Finance agreement class to check //

}

catch (MyException MyEx)

{

okFinance = false;

errP.SetError(lblFinanceFinanceIDEdit, MyEx.toString());

}

try

{

myFinanceAgreement.RegNo = lblFinanceRegNoEdit.Text.Trim(); // passed to Finance agreement class to check //

}

catch (MyException MyEx)

{

okFinance = false;

errP.SetError(lblFinanceRegNoEdit, MyEx.toString());

}

try

{

myFinanceAgreement.AgreementDate = Convert.ToDateTime(DateTime.Now.ToShortDateString()); // passed to Finance agreement class to check //

}

catch (MyException MyEx)

{

okFinance = false;

errP.SetError(lblFinanceAgreementDateEdit, MyEx.toString());

}

try

{

myFinanceAgreement.NoOfPayments = int.Parse(lblFinanceNumPaymentsEdit.Text); // passed to Finance agreement class to check //

}

catch (MyException MyEx)

{

okFinance = false;

errP.SetError(lblFinanceNumPaymentsEdit, MyEx.toString());

}

try

{

myFinanceAgreement.MonthlyPayment = double.Parse(lblFinanceMonthlyPaymentEdit.Text); // passed to Finance agreement class to check //

}

catch (MyException MyEx)

{

okFinance = false;

errP.SetError(lblFinanceMonthlyPaymentEdit, MyEx.toString());

}

try

{

myFinanceAgreement.Deposit = double.Parse(txtFinanceDepositEdit.Text); // passed to Finance agreement class to check //

}

catch (MyException MyEx)

{

okFinance = false;

errP.SetError(txtFinanceDepositEdit, MyEx.toString());

}

try

{

myFinanceAgreement.TradeInValue = double.Parse(lblFinanceTradeInEdit.Text); // passed to Finance agreement class to check //

}

catch (MyException MyEx)

{

okFinance = false;

errP.SetError(lblFinanceTradeInEdit, MyEx.toString());

}

try

{

myFinanceAgreement.InterestRate = double.Parse(lblFinanceInterestRateEdit.Text); // passed to Finance agreement class to check //

}

catch (MyException MyEx)

{

okFinance = false;

errP.SetError(lblFinanceInterestRateEdit, MyEx.toString());

}

try

{

if (okFinance)

{

drFinanceAgreement.BeginEdit();

drFinanceAgreement["FinanceID"] = myFinanceAgreement.FinanceID;

drFinanceAgreement["RegNo"] = myFinanceAgreement.RegNo;

drFinanceAgreement["AgreementDate"] = myFinanceAgreement.AgreementDate;

drFinanceAgreement["NoOfPayments"] = myFinanceAgreement.NoOfPayments;

drFinanceAgreement["MonthlyPayment"] = myFinanceAgreement.MonthlyPayment;

drFinanceAgreement["Deposit"] = myFinanceAgreement.Deposit;

drFinanceAgreement["TradeInValue"] = myFinanceAgreement.TradeInValue;

drFinanceAgreement["InterestRate"] = myFinanceAgreement.InterestRate;

drFinanceAgreement.EndEdit();

daFinanceAgreement2.Update(dsDesmonds, "FinanceAgreement2");

MessageBox.Show("Finance Agreement Updated", "Finance Edit");

cmbFinancePlanLengthEdit.Enabled = false;

txtFinanceDepositEdit.Enabled = false;

lblEditFinanceEdit.Text = "Edit";

tabSales.SelectedIndex = 0;

GlobalVar.noOfMonths = 0;

}

}

catch (Exception ex)

{

MessageBox.Show("" + ex.TargetSite + "" + ex.Message, "Error!", MessageBoxButtons.AbortRetryIgnore, MessageBoxIcon.Error);

}

}

}

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

{

bool repFound = false;

foreach (DataRow dr in dsDesmonds.Tables["customer2"].Rows)

{

if (dr["CustID"].ToString() == cmbRepOrderSearch.SelectedValue.ToString())

{

OrdersRep rep = new OrdersRep();

rep.SetParameterValue("searchID", cmbRepOrderSearch.SelectedValue.ToString());

crystalReportViewer1.ReportSource = rep;

}

}

}

// Form Events //

public frmSales()

{

InitializeComponent();

}

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

{

btnSalesEdit.Enabled = false;

btnSalesDelete.Enabled = true;

cmbFinancePlanLength.DropDownStyle = ComboBoxStyle.DropDownList;

lstCar.Enabled = false;

clearSale();

GlobalVar.tradeInFormClose = false;

btnAddSaleReset.Enabled = true;

lblSaleOrderDateShow.Text = DateTime.Now.ToShortDateString();

pnlListBoxes.Enabled = true;

for (int count = 0; count < 26; count++)

{

btns[count] = (Button)pnlButtons.Controls[count];

btns[count].Text = "" + (char)(65 + count);

btns[count].Enabled = false;

btns[count].Click += new EventHandler(btnA\_Click);

}

connStr = @"Data Source = .; Initial Catalog = desmonds; Integrated Security = true";

// Get Surnames for Letter Buttons //

sqlNames = @"Select CustSurname from customer order by CustSurname";

daNames = new SqlDataAdapter(sqlNames, connStr);

daNames.Fill(dsDesmonds, "Names");

int no;

// Enable relevant alpha buttons //

foreach (DataRow dr in dsDesmonds.Tables["Names"].Rows)

{

no = (int)dr["CustSurname"].ToString()[0] - 65; // -65 to get the index of the button, enables and disables buttons and changes colour //

btns[no].Enabled = true;

btns[no].BackColor = Color.Transparent;

btns[no].ForeColor = Color.FromArgb(50, 115, 165);

}

sqlCustomerDetails = @"Select CustID, CustTitle, CustForename, CustSurname, CustSurname + ',' + CustForename as name, CustStreet, CustTown, CustCounty, CustPostcode, CreditRating, TelNo, CustEmail FROM customer WHERE CustSurname LIKE @Letter order by CustSurname, CustForename";

conn = new SqlConnection(connStr);

cmdCustomerDetails = new SqlCommand(sqlCustomerDetails, conn);

cmdCustomerDetails.Parameters.Add("@Letter", SqlDbType.VarChar);

daCustomers = new SqlDataAdapter(cmdCustomerDetails);

daCustomers.FillSchema(dsDesmonds, SchemaType.Source, "customer");

sqlCustomerDetails2 = @"SELECT \* FROM customer";

daCustomers2 = new SqlDataAdapter(sqlCustomerDetails2, conn);

cmdBCustomer2 = new SqlCommandBuilder(daCustomers2);

daCustomers2.FillSchema(dsDesmonds, SchemaType.Source, "customer2");

daCustomers2.Fill(dsDesmonds, "customer2");

sqlOrders = @"SELECT \* FROM orders";

daOrders = new SqlDataAdapter(sqlOrders, conn);

cmdBOrder = new SqlCommandBuilder(daOrders);

daOrders.FillSchema(dsDesmonds, SchemaType.Source, "Orders");

daOrders.Fill(dsDesmonds, "Orders");

sqlOrderDetails = @"SELECT \* FROM orderDetails";

daOrderDetails = new SqlDataAdapter(sqlOrderDetails, conn);

cmdBOrderDet = new SqlCommandBuilder(daOrderDetails);

daOrderDetails.FillSchema(dsDesmonds, SchemaType.Source, "OrderDetails");

daOrderDetails.Fill(dsDesmonds, "OrderDetails");

sqlCarDetails = @"SELECT \* FROM car";

daCars = new SqlDataAdapter(sqlCarDetails, conn);

cmdBCars = new SqlCommandBuilder(daCars);

daCars.FillSchema(dsDesmonds, SchemaType.Source, "Car");

daCars.Fill(dsDesmonds, "Car");

sqlFinanceDetails2 = @"SELECT \* FROM financeAgreement";

daFinanceAgreement2 = new SqlDataAdapter(sqlFinanceDetails2, conn);

cmdBFinanceAgreement2 = new SqlCommandBuilder(daFinanceAgreement2);

daFinanceAgreement2.FillSchema(dsDesmonds, SchemaType.Source, "financeAgreement2");

daFinanceAgreement2.Fill(dsDesmonds, "financeAgreement2");

sqlFinanceDetails = @"SELECT fa.FinanceID, fa.RegNo, cst.CustID, cst.CustSurname + ',' + cst.CustForename as name, cst.CustStreet, cst.CustTown, cst.CustCounty, cst.CustPostcode, cst.CreditRating, cst.TelNo, cst.CustEmail, fa.AgreementDate, fa.NoOfPayments, fa.MonthlyPayment, fa.Deposit, fa.TradeInValue, fa.InterestRate, c.MakeID, c.Model, c.Colour, c. Mileage, c.Transmission, c.Interior, c.BodyStyle, c.SalePrice FROM financeAgreement fa

JOIN car c ON fa.RegNo = c.RegNo

JOIN orderDetails od ON c.RegNo = od.RegNo

JOIN orders o ON od.OrderID = o.OrderID

JOIN customer cst ON o.CustID = cst.CustID order by fa.FinanceID ASC";

daFinanceAgreement = new SqlDataAdapter(sqlFinanceDetails, conn);

cmdBFinanceAgreement = new SqlCommandBuilder(daFinanceAgreement);

daFinanceAgreement.FillSchema(dsDesmonds, SchemaType.Source, "financeAgreement");

daFinanceAgreement.Fill(dsDesmonds, "financeAgreement");

int noRows = dsDesmonds.Tables["Orders"].Rows.Count;

if (noRows == 0)

{

lblSaleOrderNoShow.Text = "OD100000";

}

else

{

getTotalNumOrders(noRows);

}

dgvOrders.DataSource = dsDesmonds.Tables["Orders"];

// Resize the dgv columns to fit the newly loaded content //

dgvOrders.AutoResizeColumns(DataGridViewAutoSizeColumnsMode.AllCells);

dgvFinanceAgreements.DataSource = dsDesmonds.Tables["FinanceAgreement2"];

// Resize the dgv columns to fit the newly loaded content //

dgvFinanceAgreements.AutoResizeColumns(DataGridViewAutoSizeColumnsMode.AllCells);

}

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

{

tabSales.SelectedIndex = 4;

}

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

{

GlobalVar.addFinance = false;

GlobalVar.noOfMonths = int.Parse(lblFinanceNumPaymentsShow.Text);

MySales myOrders = new MySales();

bool OkOrder = true;

String invalMessage = "Invalid data entry for: ";

errP.Clear();

try

{

myOrders.OrderID = lblSaleOrderNoShow.Text.Trim(); // passed to car class to check //

}

catch (MyException MyEx)

{

OkOrder = false;

errP.SetError(lblSaleOrderNoShow, MyEx.toString());

}

try

{

myOrders.OrderDate = Convert.ToDateTime(DateTime.Now.ToShortDateString()); // passed to car class to check //

}

catch (MyException MyEx)

{

OkOrder = false;

errP.SetError(lblSaleOrderDateShow, MyEx.toString());

}

try

{

myOrders.CustID = lblSaleCustNoShow.Text.Trim(); // passed to car class to check //

}

catch (MyException MyEx)

{

OkOrder = false;

errP.SetError(lblSaleCustNoShow, MyEx.toString());

}

try

{

if (rbCash.Checked == true)

{

myOrders.PaymentTypeID = 1;

}

if (rbCard.Checked == true)

{

myOrders.PaymentTypeID = 2;

}

if (rbFinance.Checked == true)

{

myOrders.PaymentTypeID = 3;

}

if (rbCheque.Checked == true)

{

myOrders.PaymentTypeID = 4;

}

}

catch (MyException MyEx)

{

OkOrder = false;

errP.SetError(rbCash, MyEx.toString());

errP.SetError(rbCard, MyEx.toString());

errP.SetError(rbFinance, MyEx.toString());

errP.SetError(rbCheque, MyEx.toString());

}

try

{

if (OkOrder)

{

drCar.BeginEdit();

drCar["Sold"] = 1;

drCar.EndEdit();

daCars.Update(dsDesmonds, "Car");

drOrder = dsDesmonds.Tables["Orders"].NewRow();

drOrder["OrderID"] = myOrders.OrderID;

drOrder["OrderDate"] = myOrders.OrderDate;

drOrder["CustID"] = myOrders.CustID;

drOrder["PaymentTypeID"] = myOrders.PaymentTypeID;

dsDesmonds.Tables["Orders"].Rows.Add(drOrder);

daOrders.Update(dsDesmonds, "Orders");

MessageBox.Show("Order Added");

clearSale();

tabSales.SelectedIndex = 0;

}

}

catch (Exception ex)

{

MessageBox.Show("" + ex.TargetSite + "" + ex.Message, "Error!", MessageBoxButtons.AbortRetryIgnore, MessageBoxIcon.Error);

}

/////////////////////// ADDING FINANCE AGREEMENT ///////////////////////

bool okAgreement = true;

MyFinanceAgreement myFinanceAgreement = new MyFinanceAgreement();

errP.Clear();

try

{

myFinanceAgreement.FinanceID = lblFinanceAgreementIDShow.Text.Trim(); // passed to Finance agreement class to check //

}

catch (MyException MyEx)

{

okAgreement = false;

errP.SetError(lblFinanceAgreementIDShow, MyEx.toString());

}

try

{

myFinanceAgreement.RegNo = lblFinanceRegNoShow.Text.Trim(); // passed to Finance agreement class to check //

}

catch (MyException MyEx)

{

okAgreement = false;

errP.SetError(lblFinanceRegNoShow, MyEx.toString());

}

try

{

myFinanceAgreement.AgreementDate = Convert.ToDateTime(DateTime.Now.ToShortDateString()); // passed to Finance agreement class to check //

}

catch (MyException MyEx)

{

okAgreement = false;

errP.SetError(lblFinanceAgreeDateShow, MyEx.toString());

}

try

{

myFinanceAgreement.NoOfPayments = GlobalVar.noOfMonths; // passed to Finance agreement class to check //

}

catch (MyException MyEx)

{

okAgreement = false;

errP.SetError(lblFinanceNumPaymentsShow, MyEx.toString());

}

try

{

myFinanceAgreement.MonthlyPayment = double.Parse(lblFinanceMonthlyPaymentShow.Text); // passed to Finance agreement class to check //

}

catch (MyException MyEx)

{

okAgreement = false;

errP.SetError(lblFinanceMonthlyPaymentShow, MyEx.toString());

}

try

{

myFinanceAgreement.Deposit = double.Parse(lblFinanceDepositShow.Text); // passed to Finance agreement class to check //

}

catch (MyException MyEx)

{

okAgreement = false;

errP.SetError(lblFinanceDepositShow, MyEx.toString());

}

try

{

myFinanceAgreement.TradeInValue = double.Parse(lblFinanceTradeValueShow.Text); // passed to Finance agreement class to check //

}

catch (MyException MyEx)

{

okAgreement = false;

errP.SetError(lblFinanceTradeValueShow, MyEx.toString());

}

try

{

myFinanceAgreement.InterestRate = double.Parse(lblFinanceInterestRateShow.Text); // passed to Finance agreement class to check //

}

catch (MyException MyEx)

{

okAgreement = false;

errP.SetError(lblFinanceInterestRateShow, MyEx.toString());

}

try

{

if (okAgreement)

{

drFinanceAgreement = dsDesmonds.Tables["FinanceAgreement2"].NewRow();

drFinanceAgreement["FinanceID"] = myFinanceAgreement.FinanceID;

drFinanceAgreement["RegNo"] = myFinanceAgreement.RegNo;

drFinanceAgreement["AgreementDate"] = myFinanceAgreement.AgreementDate;

drFinanceAgreement["NoOfPayments"] = myFinanceAgreement.NoOfPayments;

drFinanceAgreement["MonthlyPayment"] = myFinanceAgreement.MonthlyPayment;

drFinanceAgreement["Deposit"] = myFinanceAgreement.Deposit;

drFinanceAgreement["TradeInValue"] = myFinanceAgreement.TradeInValue;

drFinanceAgreement["InterestRate"] = myFinanceAgreement.InterestRate;

dsDesmonds.Tables["FinanceAgreement2"].Rows.Add(drFinanceAgreement);

daFinanceAgreement2.Update(dsDesmonds, "FinanceAgreement2");

MessageBox.Show("Finance Agreement Added");

clearSale();

tabSales.SelectedIndex = 0;

GlobalVar.noOfMonths = 0;

}

}

catch (Exception ex)

{

MessageBox.Show("" + ex.TargetSite + "" + ex.Message, "Error!", MessageBoxButtons.AbortRetryIgnore, MessageBoxIcon.Error);

}

}

private void frmSales\_Shown(object sender, EventArgs e)

{

tabSales.TabPages[0].CausesValidation = true;

tabSales.TabPages[0].Validating += new CancelEventHandler(AddTabValidate);

tabSales.TabPages[2].CausesValidation = true;

tabSales.TabPages[2].Validating += new CancelEventHandler(EditTabValidate);

}

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

{

if (cmbFinancePlanLength.SelectedIndex == 0)

{

lblFinanceNumPaymentsShow.Text = "12";

}

else if (cmbFinancePlanLength.SelectedIndex == 1)

{

lblFinanceNumPaymentsShow.Text = "24";

}

else if (cmbFinancePlanLength.SelectedIndex == 2)

{

lblFinanceNumPaymentsShow.Text = "36";

}

else if (cmbFinancePlanLength.SelectedIndex == 3)

{

lblFinanceNumPaymentsShow.Text = "48";

}

else if (cmbFinancePlanLength.SelectedIndex == 4)

{

lblFinanceNumPaymentsShow.Text = "60";

}

if (GlobalVar.counter > 0)

{

carPrice = double.Parse(lblSaleCarPriceShow.Text);

carDeposit = double.Parse(txtSaleDeposit.Text);

tradeValue = double.Parse(lblCarTradeValue.Text);

interestRate = double.Parse(lblFinanceInterestRateShow.Text) + 1;

costNoInterest = ((carPrice - carDeposit) - tradeValue);

costWithInterest = ((carPrice - carDeposit) - tradeValue) \* interestRate;

totalInterest = costWithInterest - costNoInterest;

numPayments = int.Parse(lblFinanceNumPaymentsShow.Text);

monthlyPayment = costWithInterest / numPayments;

lblFinanceInterestPriceShow.Text = costWithInterest.ToString("0.00");

lblFinancePriceShow.Text = costNoInterest.ToString("0.00");

lblFinanceTotalInterest.Text = totalInterest.ToString("0.00");

lblFinanceMonthlyPaymentShow.Text = monthlyPayment.ToString("0.00");

}

GlobalVar.counter++;

}

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

{

switch (cmbFinancePlanLengthEdit.SelectedIndex)

{

case 0:

{

lblFinanceNumPaymentsEdit.Text = "12";

break;

}

case 1:

{

lblFinanceNumPaymentsEdit.Text = "24";

break;

}

case 2:

{

lblFinanceNumPaymentsEdit.Text = "36";

break;

}

case 3:

{

lblFinanceNumPaymentsEdit.Text = "48";

break;

}

case 4:

{

lblFinanceNumPaymentsEdit.Text = "60";

break;

}

}

if (GlobalVar.counter > 0)

{

try

{

carPrice = double.Parse(lblSaleCarPriceShow.Text);

carDeposit = double.Parse(txtSaleDeposit.Text);

tradeValue = double.Parse(lblCarTradeValue.Text);

interestRate = double.Parse(lblFinanceInterestRateShow.Text) + 1;

costNoInterest = ((carPrice - carDeposit) - tradeValue);

costWithInterest = ((carPrice - carDeposit) - tradeValue) \* interestRate;

totalInterest = costWithInterest - costNoInterest;

numPayments = int.Parse(lblFinanceNumPaymentsShow.Text);

monthlyPayment = costWithInterest / numPayments;

lblFinanceInterestPriceShow.Text = costWithInterest.ToString();

lblFinancePriceShow.Text = costNoInterest.ToString();

lblFinanceTotalInterest.Text = totalInterest.ToString();

lblFinanceMonthlyPaymentShow.Text = monthlyPayment.ToString();

}

catch (Exception)

{

}

}

GlobalVar.counter++;

}

private void dgvFinanceAgreements\_SelectionChanged(object sender, EventArgs e)

{

try

{

financeNoSelected = dgvFinanceAgreements.SelectedRows[0].Cells[0].Value.ToString();

strCarReg = dgvFinanceAgreements.SelectedRows[0].Cells[1].Value.ToString();

}

catch (Exception)

{

}

}

// Radio Button Click Events //

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

{

txtSaleDeposit.Enabled = false;

}

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

{

txtSaleDeposit.Enabled = false;

}

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

{

txtSaleDeposit.Enabled = false;

}

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

{

txtSaleDeposit.Enabled = true;

}

// TabControl Events //

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

{

cmbFinancePlanLength.SelectedIndex = 0;

lstCar.SelectedIndex = -1;

lstCustomer.SelectedIndex = -1;

if (tabSales.SelectedIndex == 0)

{

lstCar.Items.Clear();

btnSalesEdit.Enabled = true;

btnSalesDelete.Enabled = true;

tabSales.TabPages[tabSales.SelectedIndex].Focus();

tabSales.TabPages[tabSales.SelectedIndex].CausesValidation = true;

dsDesmonds.Tables["Orders"].Clear();

daOrders.Fill(dsDesmonds, "Orders");

}

else if (tabSales.SelectedIndex == 1)

{

int noRows = dsDesmonds.Tables["Orders"].Rows.Count;

if (noRows == 0)

{

lblSaleOrderNoShow.Text = "OD100000";

}

else

{

getTotalNumOrders(noRows);

}

btnSalesEdit.Enabled = false;

btnSalesDelete.Enabled = false;

}

if (tabSales.SelectedIndex == 2)

{

lstCar.Items.Clear();

btnSalesEdit.Enabled = false;

btnSalesDelete.Enabled = false;

}

if (tabSales.SelectedIndex == 2 && GlobalVar.addFinance == false)

{

MessageBox.Show("Cannot add finance agreement until an order is added, select the Add Order Tab ");

tabSales.SelectedIndex = 0;

}

else

{

btnSalesEdit.Enabled = false;

btnSalesDelete.Enabled = false;

lblFinanceAgreeDateShow.Text = DateTime.Now.ToShortDateString();

lblFinanceDepositShow.Text = txtSaleDeposit.Text;

lblFinanceTradeValueShow.Text = lblCarTradeValue.Text;

lblFinancePriceShow.Text = lblSaleCarPriceShow.Text;

if (txtSaleDeposit.Value <= 0)

{

lblFinanceDepositShow.Text = "No Deposit";

}

else

{

lblFinanceDepositShow.Text = txtSaleDeposit.Text;

}

lblFinanceTradeValueShow.Text = lblFinanceTradeValueShow.Text;

lblFinanceMake.Text = lblSaleCarMakeShow.Text;

lblFinanceModel.Text = lblSaleCarModelShow.Text;

lblFinanceColour.Text = lblSaleCarColourShow.Text;

lblFinanceMileage.Text = lblSaleCarMileageShow.Text;

lblFinanceTransmission.Text = lblSaleCarTransmissionShow.Text;

lblFinanceInterior.Text = lblSaleCarInteriorShow.Text;

lblFinanceBodyStyle.Text = lblSaleCarBodyShow.Text;

lblFinanceCustNo.Text = lblSaleCustNoShow.Text;

lblFinanceCustName.Text = lblSaleCustNameShow.Text;

lblFinanceCustAddress.Text = lblSaleCustAddressShow.Text;

lblFinanceCustTown.Text = lblSaleCustTownShow.Text;

lblFinanceCustCounty.Text = lblSaleCustCountyShow.Text;

lblFinancePostcode.Text = lblSaleCustPostcodeShow.Text;

lblFinanceCustCreditRating.Text = lblSaleCustCreditRatingShow.Text;

lblFinanceCustEmail.Text = lblSaleCustEmailShow.Text;

lblFinanceTelNum.Text = lblSaleCustTelNumShow.Text;

if (lblFinanceCustCreditRating.Text == "Very Poor")

{

lblFinanceInterestRateShow.Text = "0.09";

}

else if (lblFinanceCustCreditRating.Text == "Poor")

{

lblFinanceInterestRateShow.Text = "0.085";

}

else if (lblFinanceCustCreditRating.Text == "Average")

{

lblFinanceInterestRateShow.Text = "0.08";

}

else if (lblFinanceCustCreditRating.Text == "Good")

{

lblFinanceInterestRateShow.Text = "0.075";

}

else if (lblFinanceCustCreditRating.Text == "Very Good")

{

lblFinanceInterestRateShow.Text = "0.07";

}

else if (lblFinanceCustCreditRating.Text == "Excellent")

{

lblFinanceInterestRateShow.Text = "0.065";

}

int noRows = dsDesmonds.Tables["financeAgreement2"].Rows.Count;

if (noRows == 0)

{

lblFinanceAgreementIDShow.Text = "FA100000";

}

else

{

getTotalNumFinanceAgreements(noRows);

}

try

{

carPrice = double.Parse(lblFinancePriceShow.Text);

carDeposit = double.Parse(lblFinanceDepositShow.Text);

tradeValue = double.Parse(lblFinanceTradeValueShow.Text);

interestRate = double.Parse(lblFinanceInterestRateShow.Text) + 1;

costNoInterest = ((carPrice - carDeposit) - tradeValue);

costWithInterest = ((carPrice - carDeposit) - tradeValue) \* interestRate;

totalInterest = costWithInterest - costNoInterest;

numPayments = int.Parse(lblFinanceNumPaymentsShow.Text);

monthlyPayment = costWithInterest / numPayments;

lblFinanceInterestPriceShow.Text = costWithInterest.ToString();

lblFinancePriceShow.Text = costNoInterest.ToString();

lblFinanceTotalInterest.Text = totalInterest.ToString();

lblFinanceMonthlyPaymentShow.Text = monthlyPayment.ToString();

}

catch (Exception ex1)

{

// nothing in here purposefully, just to handle crash //

}

}

if (tabSales.SelectedIndex == 3)

{

btnSalesEdit.Enabled = true;

btnSalesDelete.Enabled = true;

lstCar.Items.Clear();

}

if (tabSales.SelectedIndex == 4)

{

btnSalesEdit.Enabled = false;

btnSalesDelete.Enabled = false;

lstCar.Items.Clear();

daFinanceAgreement.Fill(dsDesmonds, "FinanceAgreement");

if (financeNoSelected == "")

{

tabSales.SelectedIndex = 0;

MessageBox.Show("Please select a finance agreement from the Display Finanace tab in order to use the manage finance tab");

}

else

{

foreach (DataRow dr in dsDesmonds.Tables["FinanceAgreement"].Rows)

{

if (dr["FinanceID"].ToString() == dgvFinanceAgreements.CurrentRow.Cells[0].Value.ToString())

{

string tempNum = dr["NoOfPayments"].ToString();

lblFinanceCustNoEdit.Text = dr["CustID"].ToString();

lblFinanceRegNoEdit.Text = dr["RegNo"].ToString();

lblFinanceMakeEdit.Text = dr["MakeID"].ToString();

lblFinanceModelEdit.Text = dr["Model"].ToString();

lblFinanceColourEdit.Text = dr["Colour"].ToString();

lblFinanceMileageEdit.Text = dr["Mileage"].ToString();

lblFinanceTransmissionEdit.Text = dr["Transmission"].ToString();

lblFinanceInteriorEdit.Text = dr["Interior"].ToString();

lblFinanceBodyEdit.Text = dr["BodyStyle"].ToString();

lblFinanceCustNameEdit.Text = dr["name"].ToString();

lblFinanceCustAddressEdit.Text = dr["CustStreet"].ToString();

lblFinanceCustTownEdit.Text = dr["CustTown"].ToString();

lblFinanceCustCountyEdit.Text = dr["CustCounty"].ToString();

lblFinanceCustPostcodeEdit.Text = dr["CustPostcode"].ToString();

lblFinanceCustEmailEdit.Text = dr["CustEmail"].ToString();

lblFinanceCustCreditEdit.Text = dr["CreditRating"].ToString();

lblFinanceTelNumEdit.Text = dr["TelNo"].ToString();

if (tempNum == "12")

{

cmbFinancePlanLengthEdit.SelectedIndex = 0;

}

else if (tempNum == "24")

{

cmbFinancePlanLengthEdit.SelectedIndex = 1;

}

else if (tempNum == "36")

{

cmbFinancePlanLengthEdit.SelectedIndex = 2;

}

else if (tempNum == "48")

{

cmbFinancePlanLengthEdit.SelectedIndex = 3;

}

else if (tempNum == "60")

{

cmbFinancePlanLengthEdit.SelectedIndex = 4;

}

txtFinanceDepositEdit.Value = int.Parse(dr["Deposit"].ToString());

lblFinanceTradeInEdit.Text = dr["TradeInValue"].ToString();

lblFinanceFinanceIDEdit.Text = dr["FinanceID"].ToString();

lblFinanceAgreementDateEdit.Text = dr["AgreementDate"].ToString();

lblFinanceInterestRateEdit.Text = dr["InterestRate"].ToString();

if (cmbFinancePlanLengthEdit.SelectedIndex == 0)

{

lblFinanceNumPaymentsEdit.Text = "12";

}

else if (cmbFinancePlanLengthEdit.SelectedIndex == 1)

{

lblFinanceNumPaymentsEdit.Text = "24";

}

else if (cmbFinancePlanLengthEdit.SelectedIndex == 2)

{

lblFinanceNumPaymentsEdit.Text = "36";

}

else if (cmbFinancePlanLengthEdit.SelectedIndex == 3)

{

lblFinanceNumPaymentsEdit.Text = "48";

}

else if (cmbFinancePlanLengthEdit.SelectedIndex == 4)

{

lblFinanceNumPaymentsEdit.Text = "60";

}

try

{

carPrice = double.Parse(lblFinancePriceEdit.Text);

carDeposit = double.Parse(txtFinanceDepositEdit.Text);

tradeValue = double.Parse(lblFinanceTradeInEdit.Text);

interestRate = double.Parse(lblFinanceInterestRateEdit.Text) + 1;

costNoInterest = ((carPrice - carDeposit) - tradeValue);

costWithInterest = ((carPrice - carDeposit) - tradeValue) \* interestRate;

totalInterest = costWithInterest - costNoInterest;

numPayments = int.Parse(lblFinanceNumPaymentsEdit.Text);

monthlyPayment = costWithInterest / numPayments;

lblFinanceTotalInterestEdit.Text = costWithInterest.ToString();

lblFinanceTotalNoInterestEdit.Text = costNoInterest.ToString();

lblFinanceInterestDueEdit.Text = totalInterest.ToString();

lblFinanceMonthlyPaymentEdit.Text = monthlyPayment.ToString();

}

catch (Exception ex1)

{

}

}

}

}

}

if (tabSales.SelectedIndex == 5)

{

btnSalesEdit.Enabled = false;

btnSalesDelete.Enabled = false;

lstCar.Items.Clear();

cmbRepOrderSearch.DataSource = dsDesmonds.Tables["customer2"];

cmbRepOrderSearch.DisplayMember = "CustID";

cmbRepOrderSearch.ValueMember = "CustID";

cmbRepOrderSearch.SelectedIndex = 0;

}

}

// List Events //

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

{

lstCar.Items.Clear();

lstCar.SelectedIndex = -1;

if (lstCar.SelectedIndex == -1)

{

pnlSaleCarDetails.Visible = false;

pnlSaleCustDetails.Visible = false;

}

if (lstCustomer.SelectedIndex == -1)

{

var confirmResult = MessageBox.Show("Please select a customer", "Select Customer", MessageBoxButtons.OK);

}

else

{

String title = "";

drCustomer = dsDesmonds.Tables["customer"].Rows.Find(lstCustomer.SelectedValue);

if (drCustomer["CustTitle"].ToString() == "Mr")

{

title = "Mr";

}

if (drCustomer["CustTitle"].ToString() == "Mrs")

{

title = "Mrs";

}

if (drCustomer["CustTitle"].ToString() == "Miss")

{

title = "Miss";

}

if (drCustomer["CustTitle"].ToString() == "Ms")

{

title = "Ms";

}

lblSaleCustNoShow.Text = drCustomer["CustID"].ToString();

lblSaleCustNameShow.Text = title + " " + drCustomer["CustForename"].ToString() + " " + drCustomer["CustSurname"].ToString();

lblSaleCustAddressShow.Text = drCustomer["CustStreet"].ToString();

lblSaleCustTownShow.Text = drCustomer["CustTown"].ToString();

lblSaleCustCountyShow.Text = drCustomer["CustCounty"].ToString();

lblSaleCustPostcodeShow.Text = drCustomer["CustPostcode"].ToString();

lblSaleCustCreditRatingShow.Text = drCustomer["CreditRating"].ToString();

lblSaleCustEmailShow.Text = drCustomer["CustEmail"].ToString();

lblSaleCustTelNumShow.Text = drCustomer["TelNo"].ToString();

pnlSaleCarDetails.Visible = true;

pnlSaleCustDetails.Visible = true;

daCars.Fill(dsDesmonds, "Car");

// fill list box

foreach (DataRow dr in dsDesmonds.Tables["Car"].Rows)

{

if (Convert.ToBoolean(dr["Sold"]) == false)

{

lstCar.Items.Add(dr["RegNo"].ToString());

}

}

lstCar.SelectedIndex = -1;

lstCar.Enabled = true;

}

}

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

{

if (lstCustomer.SelectedIndex == -1)

{

var confirmResult = MessageBox.Show("Please select a Car Registration", "Select Car Registration", MessageBoxButtons.OK);

}

else

{

drCar = dsDesmonds.Tables["Car"].Rows.Find(lstCar.SelectedItem.ToString());

lblFinanceRegNoShow.Text = drCar["RegNo"].ToString();

lblSaleCarMakeShow.Text = drCar["MakeID"].ToString();

lblSaleCarModelShow.Text = drCar["Model"].ToString();

lblSaleCarTransmissionShow.Text = drCar["Transmission"].ToString();

lblSaleCarInteriorShow.Text = drCar["Interior"].ToString();

lblSaleCarBodyShow.Text = drCar["BodyStyle"].ToString();

lblSaleCarColourShow.Text = drCar["Colour"].ToString();

lblSaleCarMileageShow.Text = drCar["Mileage"].ToString();

lblSaleCarPriceShow.Text = drCar["SalePrice"].ToString();

pnlSaleCarDetails.Visible = true;

pnlSaleCustDetails.Visible = true;

}

}

// Button Events //

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

{

pnlListBoxes.Enabled = true;

Button b = (Button)sender;

// get customer details for listbox - use selected button letter for parameter //

String str = b.Text;

// empty dataset table customer

dsDesmonds.Tables["customer"].Clear();

fillListBoxCustomers(str);

// clear any previously selected cars/orders by emptying the dataset tables //

dsDesmonds.Tables["Car"].Clear();

dsDesmonds.Tables["Orders"].Clear();

clearSale();

}

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

{

this.Hide();

frmMainMenu menuForm = new frmMainMenu();

menuForm.ShowDialog();

}

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

{

if (rbFinance.Checked == false)

{

MySales myOrders = new MySales();

bool Ok = true;

String invalMessage = "Invalid data entry for: ";

errP.Clear();

try

{

myOrders.OrderID = lblSaleOrderNoShow.Text.Trim(); // passed to car class to check //

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(lblSaleOrderNoShow, MyEx.toString());

}

try

{

myOrders.OrderDate = Convert.ToDateTime(DateTime.Now.ToShortDateString()); // passed to car class to check //

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(lblSaleOrderDateShow, MyEx.toString());

}

try

{

myOrders.CustID = lblSaleCustNoShow.Text.Trim(); // passed to car class to check //

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(lblSaleCustNoShow, MyEx.toString());

}

try

{

if (rbCash.Checked == true)

{

myOrders.PaymentTypeID = 1;

}

if (rbCard.Checked == true)

{

myOrders.PaymentTypeID = 2;

}

if (rbFinance.Checked == true)

{

myOrders.PaymentTypeID = 3;

}

if (rbCheque.Checked == true)

{

myOrders.PaymentTypeID = 4;

}

}

catch (MyException MyEx)

{

Ok = false;

errP.SetError(rbCash, MyEx.toString());

errP.SetError(rbCard, MyEx.toString());

errP.SetError(rbFinance, MyEx.toString());

errP.SetError(rbCheque, MyEx.toString());

invalMessage += " Payment Type |";

}

try

{

if (Ok)

{

drCar.BeginEdit();

drCar["Sold"] = 1;

drCar.EndEdit();

daCars.Update(dsDesmonds, "Car");

drOrder = dsDesmonds.Tables["Orders"].NewRow();

drOrder["OrderID"] = myOrders.OrderID;

drOrder["OrderDate"] = myOrders.OrderDate;

drOrder["CustID"] = myOrders.CustID;

drOrder["PaymentTypeID"] = myOrders.PaymentTypeID;

dsDesmonds.Tables["Orders"].Rows.Add(drOrder);

MessageBox.Show("Order Added");

clearSale();

tabSales.SelectedIndex = 0;

}

}

catch (Exception ex)

{

MessageBox.Show("" + ex.TargetSite + "" + ex.Message, "Error!", MessageBoxButtons.AbortRetryIgnore, MessageBoxIcon.Error);

}

}

else

{

GlobalVar.addFinance = true;

tabSales.SelectedIndex = 2;

}

}

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

{

lstCustomer.SelectedIndex = -1;

lstCar.SelectedIndex = -1;

lblSaleOrderDateShow.Text = DateTime.Now.ToShortDateString();

txtSaleDeposit.Value = 0;

rbCard.Checked = false;

rbCash.Checked = false;

rbCheque.Checked = false;

rbFinance.Checked = false;

}

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

{

GlobalVar.tradeInCar = true;

frmCar tradeInForm = new frmCar();

tradeInForm.ShowDialog(this);

// frmTradeIn TradeInForm = new frmTradeIn();

// TradeInForm.ShowDialog(this);

}

// Custom Methods //

private void clearSale()

{

lstCustomer.SelectedIndex = -1;

lblSaleCustNameShow.Text = "";

lblSaleCustAddressShow.Text = "";

lblSaleCustTownShow.Text = "";

lblSaleCustCountyShow.Text = "";

lblSaleCustPostcodeShow.Text = "";

lblSaleCustCreditRatingShow.Text = "";

lblSaleCustEmailShow.Text = "";

lblSaleCustTelNumShow.Text = "";

lblSaleCarBodyShow.Text = "";

lblSaleCarColourShow.Text = "";

lblSaleCarDetailsTitle.Text = "";

lblSaleCarInteriorShow.Text = "";

lblSaleCarMakeShow.Text = "";

lblSaleCarMileageShow.Text = "";

lblSaleCarModelShow.Text = "";

lblSaleCarTransmissionShow.Text = "";

rbCard.Checked = false;

rbCash.Checked = false;

rbFinance.Checked = false;

rbCheque.Checked = false;

}

private void getTotalNumOrders(int noRows)

{

drOrder = dsDesmonds.Tables["Orders"].Rows[noRows - 1];

int temp = int.Parse(drOrder["OrderID"].ToString().Substring(2, 6)) + 1;

lblSaleOrderNoShow.Text = "OD" + temp;

}

private void getTotalNumFinanceAgreements(int noRows)

{

drOrder = dsDesmonds.Tables["financeAgreement2"].Rows[noRows - 1];

int temp = int.Parse(drOrder["FinanceID"].ToString().Substring(2, 6)) + 1;

lblFinanceAgreementIDShow.Text = "FA" + temp;

}

private void fillListBoxCustomers(String str)

{

// get all customer details for listbox - use wildcard for parameter //

cmdCustomerDetails.Parameters["@Letter"].Value = str + "%";

daCustomers.Fill(dsDesmonds, "customer");

// fill listbox //

lstCustomer.DataSource = dsDesmonds.Tables["customer"];

lstCustomer.DisplayMember = "name";

lstCustomer.ValueMember = "CustID";

}

void EditTabValidate(object sender, CancelEventArgs e)

{

if (financeSelected == false) // && financeNoSelected == "")

{

// have to do this bit //

// reset tab to display and put out a message to select customer

financeSelected = false;

financeNoSelected = "";

}

else if (dgvFinanceAgreements.SelectedRows.Count == 1)

{

financeSelected = true;

financeNoSelected = dgvFinanceAgreements.SelectedRows[0].Cells[0].Value.ToString();

}

}

void AddTabValidate(object sender, CancelEventArgs e)

{

if (dgvFinanceAgreements.SelectedRows.Count == 0)

{

financeSelected = false;

financeNoSelected = "";

}

else if (dgvFinanceAgreements.SelectedRows.Count == 1)

{

financeSelected = true;

financeNoSelected = dgvFinanceAgreements.SelectedRows[0].Cells[0].ToString();

}

}

}

}

# GlobalVar Class

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ScreenDesigns

{

class GlobalVar

{

public static int purchasePriceGlobal = 0;

public static Boolean tradeInCar = false, tradeInFormClose = false, addFinance = false;

public static int counter = 0;

public static int noOfMonths = 0;

}

}

# MyCar Class

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ScreenDesigns

{

class MyCar

{

private string regNo, makeID, model, interior, transmission, bodyStyle, colour, condition, modelYear, doorNo;

private double salePrice, purchasePrice;

private int mileage;

public MyCar()

{

this.regNo = ""; this.makeID = ""; this.model = "";

this.interior = ""; this.transmission = ""; this.bodyStyle = ""; this.colour = "";

this.condition = ""; this.modelYear = ""; this.doorNo = "";

this.salePrice = 0.0; this.purchasePrice = 0.0;

this.mileage = 0;

}

public string RegNo

{

get { return regNo; }

set

{

if (MyValidation.validLength(value, 1, 7))

{

regNo = MyValidation.EachLetterToUpper(value);

}

else

throw new MyException("Please enter a valid RegNo");

}

}

public string MakeID

{

get { return makeID; }

set

{

if (MyValidation.validLength(value, 3, 20) && MyValidation.validLetterWhiteSpace(value))

{

makeID = MyValidation.EachLetterToUpper(value);

}

else

throw new MyException("Please Select a Make from the Dropdown List");

}

}

public string Model

{

get { return model; }

set

{

if (MyValidation.validLength(value, 1, 20) && MyValidation.validLetterWhiteSpace(value))

{

model = MyValidation.firstLetterEachWordUpper(value);

}

else

throw new MyException("Model must be 1-20 characters");

}

}

public string Interior

{

get { return interior; }

set

{

if (MyValidation.validLength(value, 3, 20) && MyValidation.validLetterWhiteSpace(value))

{

interior = MyValidation.firstLetterEachWordUpper(value);

}

else

throw new MyException("Interior must be 3-20 characters");

}

}

public string Transmission

{

get { return transmission; }

set

{

if (MyValidation.validLength(value, 6, 9) && MyValidation.validLetterWhiteSpace(value))

{

transmission = MyValidation.firstLetterEachWordUpper(value);

}

else

throw new MyException("Please Select a Transmission type from the Dropdown List"); ;

}

}

public string BodyStyle

{

get { return bodyStyle; }

set

{

if (MyValidation.validLength(value, 3, 20) && MyValidation.validLetterWhiteSpace(value))

{

bodyStyle = MyValidation.firstLetterEachWordUpper(value);

}

else

throw new MyException("Please Select a Body Type from the Dropdown List");

}

}

public string Colour

{

get { return colour; }

set

{

if (MyValidation.validLength(value, 3, 20) && MyValidation.validLetterWhiteSpace(value))

{

colour = MyValidation.firstLetterEachWordUpper(value);

}

else

throw new MyException("Colour must be 3-20 characters");

}

}

public string Condition

{

get { return condition; }

set

{

if (MyValidation.validLength(value, 3, 5) && MyValidation.validLetterWhiteSpace(value))

{

condition = MyValidation.firstLetterEachWordUpper(value);

}

else

throw new MyException("Please Select a Condition from the Dropdown List");

}

}

public string ModelYear

{

get { return modelYear; }

set

{

if (MyValidation.validLength(value, 4, 4))

{

modelYear = value;

}

else

throw new MyException("Please Enter the Model Year");

}

}

public string DoorNo

{

get { return doorNo; }

set

{

if (MyValidation.validLength(value, 1, 1))

{

doorNo = value;

}

else

throw new MyException("Please Select a Door No from the Dropdown List");

}

}

public double SalePrice

{

get { return salePrice; }

set

{

if (value > 0)

{

salePrice = value;

}

else

throw new MyException("Sale Price must be greater than £0");

}

}

public double PurchasePrice

{

get { return purchasePrice; }

set

{

if (value > 0)

{

purchasePrice = value;

}

else

throw new MyException("Purchase Price must be greater than £0");

}

}

public int Mileage

{

get { return mileage; }

set

{

if (value > 0)

{

mileage = value;

}

else

throw new MyException("Mileage must be at least 1");

}

}

}

}

# MyCustomer Class

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ScreenDesigns

{

class MyCustomer

{

private string customerNo, title, forename, surname, street, town, county, postcode, telNo, email, creditRating;

public MyCustomer()

{

this.customerNo = "";

this.title = ""; this.forename = ""; this.surname = "";

this.street = ""; this.town = ""; this.county = "";

this.postcode = ""; this.telNo = "";

this.email = ""; this.creditRating = "";

}

public MyCustomer(string customerNo, string title, string forename, string surname, string street, string town, string county, string postcode, string telNo, string email, string creditRating)

{

this.customerNo = customerNo;

this.title = title; this.forename = forename; this.surname = surname;

this.street = street; this.town = town; this.county = county; this.postcode = postcode;

this.telNo = telNo; this.email = email;

this.creditRating = creditRating;

}

public string CustomerNo

{

get { return customerNo; }

set { customerNo = value; }

}

public string Title

{

get { return title; }

set

{

if (value.ToUpper() != "MR" && value.ToUpper() != "MRS" && value.ToUpper() != "MISS" && value.ToUpper() != "MS")

{

throw new MyException("Title must Mr, Mrs, Miss or Ms");

}

else

title = MyValidation.firstLetterEachWordUpper(value);

}

}

public string Surname

{

get { return surname; }

set

{

if (MyValidation.validLength(value, 2, 15) && MyValidation.validSurname(value))

{

surname = MyValidation.firstLetterEachWordUpper(value);

}

else

throw new MyException("Surname must be 2-15 letters");

}

}

public string Forename

{

get { return forename; }

set

{

if (MyValidation.validLength(value, 2, 15) && MyValidation.validForename(value))

{

forename = MyValidation.firstLetterEachWordUpper(value);

}

else

throw new MyException("Forename must be 2 - 15 letters");

}

}

public string Street

{

get { return street; }

set

{

if (MyValidation.validLength(value, 5, 40)/\* && MyValidation.validLetterNumberWhiteSpace(value)\*/)

{

street = MyValidation.firstLetterEachWordUpper(value);

}

else

throw new MyException("Street must be 5 - 40 letters");

}

}

public string Town

{

get { return town; }

set

{

if (MyValidation.validLength(value, 2, 20)) //&& MyValidation.validLetterWhiteSpace(value))

{

town = MyValidation.firstLetterEachWordUpper(value);

}

else

{

throw new MyException("Town must be 2-20 letters");

}

}

}

public string County

{

get { return county; }

set

{

if (MyValidation.validLength(value, 2, 20)) // && MyValidation.validLetter(value))

{

county = MyValidation.firstLetterEachWordUpper(value);

}

else

{

throw new MyException("County must be 2-20 letters");

}

}

}

public string Postcode

{

get { return postcode; }

set

{

if (MyValidation.validLength(value, 7, 8)) /\*&& MyValidation.validLetterNumberWhiteSpace(value)\*/

{

postcode = MyValidation.EachLetterToUpper(value);

}

else

{

throw new MyException("Post Code must be between 7-8 characters and be alphanumeric only.");

}

}

}

public string TelNo

{

get { return telNo; }

set

{

if (MyValidation.validLength(value, 11, 15) && MyValidation.validNumber(value))

{

telNo = value;

}

else

{

throw new MyException("Telephone number must be 11-15 digits");

}

}

}

public string Email

{

get { return email; }

set

{

if (MyValidation.validEmail(value) == true)

{

email = value;

}

else

{

throw new MyException("Must enter a valid email address");

}

}

}

public string CreditRating

{

get { return creditRating; }

set

{

if (value.ToUpper() != "VERY POOR" && value.ToUpper() != "POOR" && value.ToUpper() != "AVERAGE" && value.ToUpper() != "GOOD" && value.ToUpper() != "VERY GOOD" && value.ToUpper() != "EXCELLENT")

{

throw new MyException("Credit Rating must be Very Poor, Poor, Average, Good, Very Good or Excellent");

}

else

creditRating = MyValidation.firstLetterEachWordUpper(value);

}

}

}

}

# MySales Class

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ScreenDesigns

{

class MySales

{

private string orderID = "", custID = "";

private DateTime orderDate = DateTime.Now;

private int paymentTypeID = 0;

public MySales()

{

this.orderID = "";

this.custID = "";

this.orderDate = DateTime.Now;

this.paymentTypeID = 0;

}

public MySales(string OrderID, string CustID, DateTime OrderDate, int PaymentTypeID)

{

this.orderID = OrderID;

this.custID = CustID;

this.orderDate = OrderDate;

this.paymentTypeID = PaymentTypeID;

}

public string OrderID

{

get { return orderID; }

set { orderID = value; }

}

public string CustID

{

get { return custID; }

set { custID = value; }

}

public DateTime OrderDate

{

get { return orderDate; }

set

{

if (value.Date >= DateTime.Now.Date)

{

orderDate = value;

}

else

throw new MyException("Order Date cannot be in the past");

}

}

public int PaymentTypeID

{

get { return paymentTypeID; }

set

{

if (value >= 1 || value <= 4)

{

paymentTypeID = value;

}

else

throw new MyException("PaymentTypeID must be 1,2,3 or 4");

}

}

}

}

# MyFinanceAgreement Class

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ScreenDesigns

{

class MyFinanceAgreement

{

string financeID, regNo;

DateTime agreementDate = DateTime.Now;

int noOfPayments = 0;

double monthlyPayment = 0, deposit = 0, tradeInValue = 0, interestRate = 0;

public MyFinanceAgreement()

{

this.financeID = ""; this.regNo = "";

this.agreementDate = DateTime.Now;

this.noOfPayments = 0;

this.monthlyPayment = 0; this.deposit = 0; this.tradeInValue = 0; this.interestRate = 0;

}

public MyFinanceAgreement(string financeID, string regNo, DateTime agreementDate, int noOfPayments, double monthlyPayment, double deposit, double tradeInValue, double interestRate)

{

this.financeID = financeID;

this.regNo = regNo;

this.agreementDate = agreementDate;

this.noOfPayments = noOfPayments;

this.monthlyPayment = monthlyPayment;

this.deposit = deposit;

this.tradeInValue = tradeInValue;

this.interestRate = interestRate;

}

public string FinanceID

{

get { return financeID; }

set { financeID = value; }

}

public string RegNo

{

get { return regNo; }

set { regNo = value; }

}

public DateTime AgreementDate

{

get { return agreementDate; }

set

{

if (value.Date >= DateTime.Now.Date)

{

agreementDate = value;

}

else

throw new MyException("Agreement Date cannot be in the past");

}

}

public int NoOfPayments

{

get { return noOfPayments; }

set { noOfPayments = value; }

}

public double MonthlyPayment

{

get { return monthlyPayment; }

set { monthlyPayment = value; }

}

public double Deposit

{

get { return deposit; }

set { deposit = value; }

}

public double TradeInValue

{

get { return tradeInValue; }

set { tradeInValue = value; }

}

public double InterestRate

{

get { return interestRate; }

set { interestRate = value; }

}

}

}

# MyValidation Class

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

namespace ScreenDesigns

{

class MyValidation

{

public static bool validLength(string txt, int min, int max)

{

bool ok = true;

if (string.IsNullOrEmpty(txt))

ok = false;

else if (txt.Length < min || txt.Length > max)

ok = false;

return ok;

}

public static bool validNumber(string txt)

{

bool ok = true;

for (int count = 0; count < txt.Length; count++)

{

if (!(char.IsNumber(txt[count])))

{

}

}

return ok;

}

public static bool validDouble(string txt)

{

bool ok = true;

for (int i = 0; i < txt.Length; i++)

{

if (!(char.IsNumber(txt[i]) || (!(char.IsDigit(txt[i])))))

{

ok = false;

}

}

return ok;

}

public static bool validLetter(string txt) //allows alphabetic characters//

{

bool ok = true;

if (txt.Trim().Length == 0)

{

ok = false;

}

else

{

for (int count = 0; count < txt.Length; count++)

{

if (!(char.IsLetter(txt[count])) && !(char.IsWhiteSpace(txt[count])))

ok = false;

}

}

return ok;

}

public static bool validForename(string txt) //allows alphabetic, dash and whitepsace for double barrel names//

{

bool ok = true;

if (txt.Trim().Length == 0)

{

ok = false;

}

else

{

for (int count = 0; count < txt.Length; count++)

{

if (!(char.IsLetter(txt[count])) && !(char.IsWhiteSpace(txt[count])) && !(txt[count].Equals('-')))

{

ok = false;

}

}

}

return ok;

}

public static bool validLetterWhiteSpace(String txt)

{

bool ok = true;

if (txt.Trim().Length == 0)

{

ok = false;

}

else

{

for (int x = 0; x < txt.Length; x++)

{

if (!(char.IsLetter(txt[x])) && !char.IsWhiteSpace(txt[x]))

{

ok = false;

}

}

}

return ok;

}

public static bool validSurname(string txt) //allows alphabetic, dash and whitepsace for double barrel names//

{

bool ok = true;

if (txt.Trim().Length == 0)

{

ok = false;

}

else

{

for (int count = 0; count < txt.Length; count++)

{

if (!(char.IsLetter(txt[count])) && !(char.IsWhiteSpace(txt[count])) && !(txt[count].Equals('-')))

{

ok = false;

}

}

}

return ok;

}

public static bool validPostcode(String txt)

{

bool ok = true;

if (txt.Trim().Length == 0)

{

ok = false;

}

else

{

bool isPostcode = Regex.IsMatch(txt, "[A-Z][A-Z][0-9][0-9] [0-9][A-Z][A-Z]", RegexOptions.IgnoreCase);

if (!isPostcode)

{

ok = false;

}

}

return ok;

}

public static bool validMake(string txt)

{

bool ok = true;

if (txt.Trim().Length == 0)

{

ok = false;

}

else

{

string[] makes = new string[] {"AUD", "FIA","FRD","HON","NIS","REN"};

for (int i = 0; i < makes.Length; i++)

{

if(makes[i].Equals(txt))

{

ok = true;

break;

}

ok = false;

}

}

return ok;

}

public static bool validTelNo(String txt)

{

bool ok = true;

if (txt.Trim().Length == 0 || txt.Trim().Length > 11)

{

ok = false;

}

else

{

bool isTelNo = Regex.IsMatch(txt, "[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]", RegexOptions.IgnoreCase);

if (!isTelNo)

{

ok = false;

}

}

return ok;

}

public static bool validEmail(string txt) //allows alphanumeric and whitespace

{

bool ok = true;

if (txt.Trim().Length == 0)

{

ok = false;

}

else

{

bool isEmail = Regex.IsMatch(txt, @"\A(?:[a-z0-9!#$%&'\*+/=?^\_`{|}~-]+(?:\.[a-z0-9!#$%&'\*+/=?^\_`{|}~-]+)\*@(?:[a-z0-9](?:[a-z0-9-]\*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]\*[a-z0-9])?)\Z", RegexOptions.IgnoreCase);

if (!isEmail)

{

ok = false;

}

}

return ok;

}

public static bool validDogDOB(DateTime dogDOB)

{

DateTime currentDate = DateTime.Now;

TimeSpan t = currentDate - dogDOB;

double NoOfDays = t.TotalDays;

bool ok = true;

if (NoOfDays <= 56)

{

ok = false;

}

return ok;

}

public static String firstLetterEachWordUpper(String word)

{

Char[] array = word.ToCharArray();

if (Char.IsLower(array[0]))

{

array[0] = Char.ToUpper(array[0]);

}

for (int count = 1; count < array.Length; count++)

{

if (array[count - 1] == ' ')

{

if (Char.IsLower(array[count]))

{

array[count] = Char.ToUpper(array[count]);

}

}

else

array[count] = Char.ToLower(array[count]);

}

return new string(array);

}

public static String EachLetterToUpper(String word)

{

Char[] array = word.ToCharArray();

for (int count = 0; count < array.Length; count++)

{

if (Char.IsLower(array[count]))

{

array[count] = char.ToUpper(array[count]);

}

}

return new string(array);

}

public static bool validLetterNumberWhiteSpace(String txt)

{

bool ok = true;

if (txt.Trim().Length == 0)

{

ok = false;

}

else

{

for (int x = 0; x < txt.Length; x++)

{

if (!(char.IsNumber(txt[x])) && !char.IsWhiteSpace(txt[x]) && !(char.IsLetter(txt[x])))

{

ok = false;

}

}

}

return ok;

}

}

}

# MyException Class

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ScreenDesigns

{

class MyException : Exception

{

private String message;

public MyException()

{

}

public MyException(String message)

{

this.message = message;

}

public String toString()

{

return String.Format("Error: {0} ", message);

}

}

}

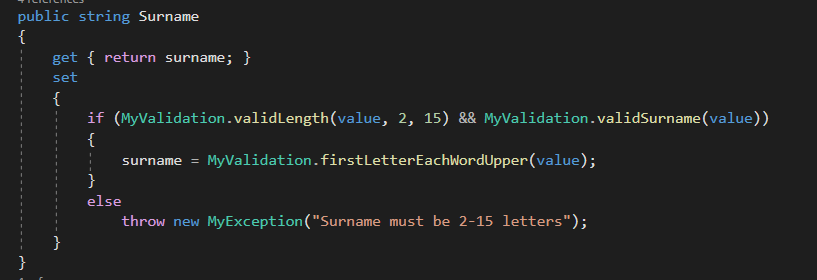
# Input Testing

## String Testing

|  |  |  |  |
| --- | --- | --- | --- |
| Test No. | Test Description | Expected Outcome | Actual Outcome |
| 1. | Entering a valid string | String should be accepted with no error | As Expected |
| 2. | Entering invalid input in a string field (Special Characters) | String should not be accepted, and text box background colour should turn red | As Expected |
| 3. | Entering a string above maximum character count | String should not be accepted; text box background colour should turn red and error provider icon should show with tooltip message. | As Expected |
| 4. | Entering a string below minimum character count | Input should not be accepted; text box background colour should turn red and error provider icon should show with tooltip message. | As Expected |
| 5. | Enter numbers only | Input should not be accepted, and error provider icon should show with tooltip message. | As Expected |
| 6. | Only enter spaces | Input should not be accepted, and error provider icon should show with tooltip message. | As Expected |
| 7. | Test no input | Input should not be accepted, and error provider icon should show with tooltip message. | As Expected |

## String Validation

I have written most of my validation in the set accessor of each attribute in each class, e.g. the MyCustomer class for customers. The methods I used for validating strings can be seen in the MyValidation class, this class was combined with the MyException class allowing me to create my own custom error providers by using try/catches. An example of how I used these methods to validate the length and ensuring that the data entered matches that of a surname string, accounting for ‘-‘ and “double barrel” surnames is shown below:

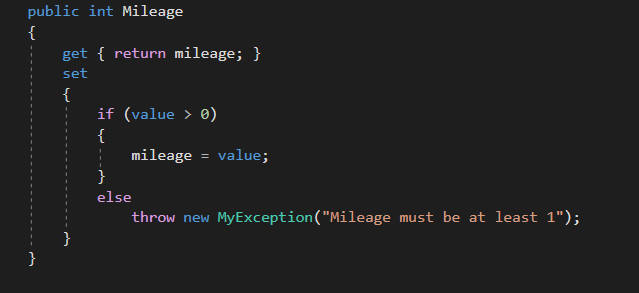


## Integer Testing

|  |  |  |  |
| --- | --- | --- | --- |
| Test No. | Test Description | Expected Outcome | Actual Outcome |
| 1. | Enter valid integer into mileage field | Input should be accepted | As Expected |
| 2. | Enter invalid input into mileage field (Special Characters) | Should not be able to as it is a masked text field, input should only accept integers | As Expected |
| 3. | Enter integer above maximum allowed | Should reset to the maximum when the threshold is exceeded. | As Expected |
| 4. | Enter integer below minimum allowed (1) | Should reset to 0 with error provider icon. | As Expected |
| 5. | Enter string value | Should not be able to as it is a masked text field, input should only accept integers | As Expected |
| 6. | Enter only spaces | Should not be able to as it is a masked text field, input should only accept integers | As Expected |
| 7. | Try and submit with field blank | Should not be able to as it is a masked text field, input should only accept integers | As Expected |

## Integer Validation

As with my string validation I written most of my validation for integers in the set accessor of each attribute in each class, e.g. the MySales class for sales. The methods I used for validating integers can also be seen in the MyValidation class, this class was also combined with the MyException class. An example is shown below of how I validated the mileage input, making sure that it input was greater than 0:



# Functionality Testing

|  |  |  |  |
| --- | --- | --- | --- |
| Test No. | Test Description | Expected Outcome | Actual Outcome |
| **Customer Form Functionality** | | | |
|  | Add new customer to database | Customer should be added if input is valid and message box shown to confirm. Second message box shown to ask if the user wants to add another customer | As Expected |
|  | Click Yes to add another customer on message box | Customer should be added if input is valid and message box shown to confirm. Second message box shown to ask if the user wants to add another customer | As Expected |
|  | Click No to add another customer on message box | System should revert to display tab of customer form | As Expected |
|  | Edit Existing customer | System should update selected customers details when save button is clicked | As Expected |
|  | Check that edit details verification | When customer details are edited, the user must click edit to unlock the fields |  |
|  | Delete existing customer from database | Message box asking user if they are sure they want to delete the customer, if user clicks yes, details are deleted, if they click no, return to display tab. | As Expected |
|  | Customer cannot be deleted if they are linked to finance agreement/order. | Message box should be shown notifying the user that the customer cannot be deleted as they are linked to an order. | As Expected |
|  | Click home button | Take user back to the home screen of the system | As Expected |
|  | Click exit button | System should close | As Expected |
|  | Click on the reports tab to ensure customer report is working | Customer report should load on crystal report viewer | As Expected |
|  | Click the reset button on the add customer tab | Should reset all fields to blank | As Expected |
|  | Change to another tab while an edit is ongoing and change back to edit tab | When the user goes back to edit tab the details should reload, any changes made before this will be lost. | As Expected – Should have put a message to confirm that the user wants to leave and that changes will be lost. |
|  | Test that Customer list is displayed | List of customers should load when tab display is selected | As Expected |
| **Car Form Functionality** | | | |
|  | Test that car list is displayed | List of cars should load when tab display is selected | As Expected |
|  | Search by RegNo filter | When text is entered search field it updates each time a character us entered and filters by RegNo | As Expected |
|  | Add new car to database | car should be added if input is valid and message box shown to confirm. Second message box shown to ask if the user wants to add another car | As Expected |
|  | Click Yes to add another car on message box | car should be added if input is valid and message box shown to confirm. Second message box shown to ask if the user wants to add another car | As Expected |
|  | Click No to add another car on message box | System should revert to display tab of car form | As Expected |
|  | Edit Existing car | System should update selected cars details when save button is clicked | As Expected |
|  | Check that edit details verification | When the car details are edited, the user must click edit to unlock the fields | As Expected |
|  | Delete existing car from database | Message box asking user if they are sure they want to delete the car, if user clicks yes, details are deleted, if they click no, return to display tab. | As Expected |
|  | Car cannot be deleted if it is linked to finance agreement/order. | Message box should be shown notifying the user that the car cannot be deleted as it is linked to an order/finance agreement. | As Expected |
|  | Test that reset button works on the add car form | When clicked the input fields should all be reset to blank | As Expected |
|  | Test home button on car form | When clicked the user should be taken to the main menu form | As Expected |
|  | Test Exit button on car form | When clicked the system should close | As Expected – should have confirmation message on this button to make it more user friendly |
| **Sales Form Functionality** | | | |
|  | Check that letter buttons to filter by surname work | When a “Letter button” is clicked, it should filter the customer listview with only surnames beginning with that letter | As Expected |
|  | Check that letter buttons that should be disabled | If there are no surnames in the database beginning with a specific letter, that button should be disabled | As Expected  Circled buttons are disabled |
|  | Check that car list view loads | When a customer is selected from the list view, the available cars should be shown on the car list view. | As Expected |
|  | Check trade in details button | When trade in details button is clicked, it opens an instance of the add car form. | As Expected |
|  | Check trade in value is taken from trade in form | When trade in form is completed and car is added to database, the purchase value of the car is set to the trade in value on the add order form. | As Expected |
|  | Check cash payment method | If the user selects cash, the sale is completed, the order is added to the database | As Expected |
|  | Check card payment method | If the user selects card, the sale is completed, the order is added to the database | As Expected |
|  | Check cheque payment method | If the user selects cheque, the sale is completed, the order is added to the database | As Expected |
|  | Check finance payment method | The user is taken to the finance screen | As Expected |
|  | Check car details on finance form | Check that the car details taken from the order form are correct | As Expected |
|  | Check customer details on finance form | Check that the customer details taken from the order form are correct | As Expected |
|  | Check plan length combo box updates finance breakdown. | Check that when the plan length is changed, the calculation is run again and the totals update | As Expected |
|  | Check that Finance agreement ID is auto incremented | When a new finance agreement is created, the id should be automatically incremented by one. | As Expected |
|  | Check agreement date | The agreement date should be todays date, taken from the system date. | As Expected |
|  | Check that number of payments matches the plan length | No. of payments should be the same as the number of months in the finance plan length combo box | As Expected |
|  | Check that the total with interest is correct | Check that total with interest is correct, this is interest rate \* remaining balance after deposit and trade in value | As Expected |
|  | Check that the total without interest is correct | Check that the total cost without interest, after trade in value and deposit is subtracted, is correct. | As Expected |
|  | Check that the total interest due is correct | Check that the total interest due total is correct, total cost \* interest rate – total cost. | As Expected |
|  | Check total monthly payment is correct | Ensure that monthly payment calculation is accurate, total with interest divided by total plan length (no. of payments) | As Expected |
|  | Submit finance message boxes | When the user submits the order finance, there should be two message boxes telling the user the order has been added and the finance agreement has been added | As Expected |
|  | Check finance agreement edit | When a finance agreement is selected and the edit button is clicked, the user should be taken to the manage finance screen | As Expected |
|  | Check the edit verification | The user should not be able to edit the plan length or deposit until they click the edit button. | As Expected |
|  | Check edit button | When the user clicks he edit button it should unlock the 2 editable fields and change to a save button. | As Expected |
|  | Check that finance agreement delete works | When the user selects a finance agreement and clicks delete it should prompt the message box asking the user are they sure they want to delete the finance agreement+ | As Expected |
|  | Check message box delete option yes | The finance agreement will be deleted from the database and no longer appear on the list | As Expected |
|  | Check message box delete option no | The message box will close, and the user will be taken back to the main from | As Expected |
|  | Test that delete button doesn’t allow the user to delete orders | Message box will ask user to select a finance agreement and not an order | As Expected |
|  | Buttons will be disabled on the appropriate screens | Edit and delete button will be disabled on certain forms | As Expected |
|  | Check calculation updates for plan length | When the user selects a new plan length the calculation should be updated accordingly | As Expected |
|  | Check calculation updates for deposit | When the user enters a new deposit length the calculation should be updated accordingly | Not as Expected  The calculation doesn’t update, this is because I haven’t placed any code to handle this int eh text changed event of the deposit text box. |
|  | Save button on edit finance tab | When the user clicks save, a message should be displayed telling the user the finance agreement has been updated. | As Expected |
|  | Display reports works | When the user clicks on reports tab, the invoice reports should be shown | Not as Expected - When creating the report, I used the customerID as the search key when creating the query but I should have used finance ID, because of this I was unable to show the report I had originally wanted to display as it needed that parameter. There was no quick way around it, and I didn’t have the time to fix it so I used another report I had already previously created in its place. |
|  | Test home button on car form | When clicked the user should be taken to the main menu form | As Expected |
|  | Test Exit button on car form | When clicked the system should close | As Expected – should have confirmation message on this button to make it more user friendly |
|  | Check display orders list | When the display orders tab is clicked, a list of orders should be shown in the data grid view | As Expected |
|  | Check display finance list | When the display finance tab is clicked, a list of finance agreements should be shown in the data grid view | As Expected |