**A picture containing text, screenshot, font, number

Description automatically generated**



Assignment 2

**Programming 3 –** RoadTrip Rentals

Jordan McElwee & David Green

**Contents**

[**Database Diagram** 5](#_Toc135013121)

[**Jordan Forms** 6](#_Toc135013122)

[Main Menu 6](#_Toc135013123)

[Main Customer 11](#_Toc135013124)

[Add Customer 14](#_Toc135013125)

[Edit Customer 19](#_Toc135013126)

[Main Rentals 24](#_Toc135013127)

[Add Rentals 27](#_Toc135013128)

[Edit / Delete Rentals 39](#_Toc135013129)

[Main Car Details 54](#_Toc135013130)

[Add Car details 57](#_Toc135013131)

[Edit car details 61](#_Toc135013132)

[Main Models 65](#_Toc135013133)

[Add Models 68](#_Toc135013134)

[Edit Model 71](#_Toc135013135)

[Main Rental Cost 74](#_Toc135013136)

[Add Rental Cost 77](#_Toc135013137)

[Edit Rental Cost 80](#_Toc135013138)

[Main Payments 83](#_Toc135013139)

[Add Payments 86](#_Toc135013140)

[Edit Payments 89](#_Toc135013141)

[**David Forms** 92](#_Toc135013142)

[Suppliers Main 92](#_Toc135013143)

[Edit Supplier 94](#_Toc135013144)

[Delete Supplier 98](#_Toc135013145)

[Add Supplier 100](#_Toc135013146)

[Stock Order 104](#_Toc135013147)

[Order Details 105](#_Toc135013148)

[Delete Order 107](#_Toc135013149)

[Order Delivered 108](#_Toc135013150)

[Add Order 110](#_Toc135013151)

[Reports 115](#_Toc135013152)

[Customer Rental car report - Jordan 115](#_Toc135013153)

[Customer List report - Jordan 116](#_Toc135013154)

[Supplier List report – David 117](#_Toc135013155)

[Supplier Stock order - David 118](#_Toc135013156)

[Testing 119](#_Toc135013157)

[Input – Jordan 119](#_Toc135013158)

[Add/ Edit Customer 119](#_Toc135013159)

[Add/ Edit Car details 124](#_Toc135013160)

[Add / Edit models 128](#_Toc135013161)

[Add / Edit Rental Cost 129](#_Toc135013162)

[Add Rental / Edit Rental 130](#_Toc135013163)

[Navigation – Jordan 132](#_Toc135013164)

[Main Menu 132](#_Toc135013165)

[Main Customer 134](#_Toc135013166)

[Add Customer 135](#_Toc135013167)

[Edit Customer 135](#_Toc135013168)

[Main Rental 136](#_Toc135013169)

[Add Rental 137](#_Toc135013170)

[Edit rental 140](#_Toc135013171)

[General navigation 141](#_Toc135013172)

[Processes / Outputs - Jordan 142](#_Toc135013173)

[Input Testing - David 143](#_Toc135013174)

[Processes/ Outputs – David 153](#_Toc135013175)

# **Database Diagram**

**A picture containing text, diagram, plan, parallel

Description automatically generated**

# **Jordan Forms**

## Main Menu

**A screenshot of a dashboard

Description automatically generated with medium confidence**

Main menu which is the first form upon loading the application. Window size is set to a minimum amount to avoid content cutoff. Everything apart from the Panel (the one containing the logo) will be changed depending on which subform button is clicked.

Main Menu

using RoadTripRentals.Forms.Jordan;

using RoadTripRentals.Properties;

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 RoadTripRentals

{

public partial class MainMenu : Form

{

Boolean vis;

private Button currentButton;

private Form activeForm;

public MainMenu()

{

InitializeComponent();

this.Text = string.Empty;

startSubClosed();

btnCloseSubForm.Visible = false;

}

//Submenu code start

private void startSubClosed()

{

panelCustomerSubmenu.Visible = false;

panelRentalsSubmenu.Visible = false;

panelSupplierSubmenu.Visible = false;

panelStockSubmenu.Visible = false;

panelCarSubmenu.Visible = false;

panelPaymentSubmenu.Visible = false;

}

private void HideAllSubmenus(params Panel[] panels)

{

foreach (var panel in panels)

{

if (panel.Visible)

{

panel.Visible = false;

}

}

}

private void hideSubMenu()

{

HideAllSubmenus(panelCustomerSubmenu, panelRentalsSubmenu, panelSupplierSubmenu, panelStockSubmenu, panelCarSubmenu, panelPaymentSubmenu);

}

private void showSubMenu(Panel subMenu)

{

if (subMenu.Visible == false)

{

hideSubMenu();

subMenu.Visible = true;

}

else

subMenu.Visible = false;

}

//Submenu End

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

{

vis = false;

tmrStop.Enabled = true;

}

private void ActivateButton(object btnSender) //Method for changeing the colour of the button to a color to highlight what has been selected

{

if (btnSender != null)

{

if (currentButton != (Button)btnSender)

{

DisableButton();

Color color = Color.FromArgb(0, 180, 126); //Setting the colour variable

currentButton = (Button)btnSender;

currentButton.BackColor = color;

currentButton.Font = new System.Drawing.Font("Segoe UI", 13.0F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point); //Changing font slightly to contrast the text

}

}

}

private void DisableButton() //Changes the previous button back to stock properties

{

foreach (Control previousBtn in panelMenu.Controls)

{

if (previousBtn.GetType() == typeof(Button))

{

previousBtn.BackColor = Color.FromArgb(26, 26, 26);

previousBtn.ForeColor = Color.White;

previousBtn.Font = new System.Drawing.Font("Segoe UI", 12F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point);

}

}

}

private void btnCloseSubForm\_Click(object sender, EventArgs e) //"X" button to close current form

{

if (activeForm != null)

activeForm.Close();

Reset();

}

private void Reset() //Back to home, removing x button

{

DisableButton();

lblTitle.Text = "Main Menu";

currentButton = null;

btnCloseSubForm.Visible = false;

}

public void openSubForm(Form childForm, object btnSender) //Opens sub forms in panel

{

if (activeForm != null)

{

activeForm.Close();

}

//ActivateButton(btnSender);

activeForm = childForm;

childForm.TopLevel = false;

childForm.FormBorderStyle = FormBorderStyle.None;

childForm.Dock = DockStyle.Fill;

this.panelForm.Controls.Add(childForm);

this.panelForm.Tag = childForm;

childForm.BringToFront();

childForm.Show();

lblTitle.Text = childForm.Text;

btnCloseSubForm.Visible = true;

}

//Main Form buttons START

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

{

ActivateButton(sender);

showSubMenu(panelCustomerSubmenu);

}

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

{

ActivateButton(sender);

showSubMenu(panelSupplierSubmenu);

}

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

{

ActivateButton(sender);

showSubMenu(panelRentalsSubmenu);

}

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

{

ActivateButton(sender);

showSubMenu(panelCarSubmenu);

}

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

{

ActivateButton(sender);

showSubMenu(panelPaymentSubmenu);

}

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

{

ActivateButton(sender);

showSubMenu(panelStockSubmenu);

}

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

{

openSubForm(new RoadTripRentals.frmMainCustomer(), sender);

}

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

{

Close();

}

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

{

frmSettings settingsForm = new frmSettings();

openSubForm(settingsForm, sender);

}

//Main form buttons END

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

{

frmAddCustomer addCustomerForm = new frmAddCustomer();

addCustomerForm.OpenSubFormRequest += OpenSubFormRequest;

openSubForm(addCustomerForm, sender);

}

private void OpenSubFormRequest(Form subForm)

{

openSubForm(subForm, null);

}

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

{

frmMainRentals mainRentalsForm = new frmMainRentals();

openSubForm(mainRentalsForm, sender);

}

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

{

frmAddRentals AddRentalsForm = new frmAddRentals();

openSubForm(AddRentalsForm, sender);

}

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

{

frmMainCar mainCarForm = new frmMainCar();

openSubForm(mainCarForm, sender);

}

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

{

frmAddCar addCarForm = new frmAddCar();

openSubForm(addCarForm, sender);

}

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

{

frmMainModel mainModelForm = new frmMainModel();

openSubForm(mainModelForm, sender);

}

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

{

frmMainRentalCost mainRentalCostForm = new frmMainRentalCost();

openSubForm(mainRentalCostForm, sender);

}

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

{

frmMainPayments mainPaymentsForm = new frmMainPayments();

openSubForm(mainPaymentsForm, sender);

}

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

{

frmAddPayments addPaymentsForm = new frmAddPayments();

openSubForm(addPaymentsForm, sender);

}

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

{

frmOrderStock addStockForm = new frmOrderStock();

openSubForm(addStockForm, sender);

}

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

{

frmSupplierMain displayForm = new frmSupplierMain();

openSubForm(displayForm, sender);

}

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

{

frmAddSupplier displayForm = new frmAddSupplier();

openSubForm(displayForm, sender);

}

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

{

frmOrderMain displayForm = new frmOrderMain();

openSubForm(displayForm, sender);

}

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

{

frmEditRental displayForm = new frmEditRental();

openSubForm(displayForm, sender);

}

}

}

## Main Customer

**A screenshot of a computer

Description automatically generated**

Customer menu which displays the database containing each of the customers. Can navigate to the add and edit forms from here which will replace the panel with the relevant form. Edit and Delete will only work if a row has been selected. The close sub form button also becomes visible, as an extra option to reset back to the main menu.

using RoadTripRentals.Forms.Jordan;

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 RoadTripRentals

{

public partial class frmMainCustomer : Form

{

SqlDataAdapter daCustomer;

DataSet dsRoadTripRentals = new DataSet();

SqlCommandBuilder cmdBCustomer;

String connStr, sqlCustomer;

public delegate void OpenSubFormRequestHandler(Form subForm);

public frmMainCustomer()

{

InitializeComponent();

}

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

{

// TODO: This line of code loads data into the 'roadTripRentalsDataSet.Customer' table. You can move, or remove it, as needed.

connStr = @"Data Source = DESKTOP-ASEMACC\INTHEDOGHOUSE; Initial Catalog = RoadTripRentals; Integrated Security = true";

sqlCustomer = @"select \* from Customer";

daCustomer = new SqlDataAdapter(sqlCustomer, connStr);

cmdBCustomer = new SqlCommandBuilder(daCustomer);

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

daCustomer.Fill(dsRoadTripRentals, "Customer");

dgvCustomers.DataSource = dsRoadTripRentals.Tables["Customer"];

//Resize the DataGridView columns to fit the newly loaded content.

dgvCustomers.AutoResizeColumns(DataGridViewAutoSizeColumnsMode.AllCells);

btnDelCustomer.Click += btnDelCustomer\_Click;

}

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

{

if (dgvCustomers.SelectedRows.Count > 0)

{

int customerID = Convert.ToInt32(dgvCustomers.SelectedRows[0].Cells["CustomerID"].Value);

// Ask for confirmation before deleting the customer

DialogResult result = MessageBox.Show("Are you sure you want to delete the selected customer?", "Confirm Delete", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (result == DialogResult.Yes)

{

// Delete the customer from the dataset and update the database

DataRow customerRow = dsRoadTripRentals.Tables["Customer"].Rows.Find(customerID);

if (customerRow != null)

{

customerRow.Delete();

daCustomer.Update(dsRoadTripRentals, "Customer");

MessageBox.Show("Customer Deleted");

}

else

{

MessageBox.Show("Customer not found.");

}

}

}

}

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

{

frmAddCustomer newSubForm = new frmAddCustomer();

OpenSubFormInPanel(newSubForm);

}

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

{

if (dgvCustomers.SelectedRows.Count > 0)

{

int customerID = Convert.ToInt32(dgvCustomers.SelectedRows[0].Cells["CustomerID"].Value);

frmEditCustomer newSubForm = new frmEditCustomer(customerID);

OpenSubFormInPanel(newSubForm);

}

else

{

MessageBox.Show("Please select a customer to edit.");

}

}

private void OpenSubFormInPanel(Form subForm)

{

subForm.TopLevel = false;

subForm.FormBorderStyle = FormBorderStyle.None;

subForm.Dock = DockStyle.Fill;

// Replace 'panel1' with the name of the Panel control in your main form.

panelSub.Controls.Clear();

panelSub.Controls.Add(subForm);

subForm.Show();

}

}

}

## Add Customer

**A screenshot of a computer

Description automatically generated with medium confidence**

Form where the a new Customer can be added to the database. Cancel will not push through the changes and returns back to the main customer menu. The customer no is auto populated reducing the input error and potential unique key conflicts.

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 RoadTripRentals.Forms.Jordan

{

public partial class frmAddCustomer : Form

{

SqlDataAdapter daCustomer;

DataSet dsRoadTripRentals = new DataSet();

SqlCommandBuilder cmdBCustomer;

DataRow drCustomer;

SqlConnection conn;

String connStr, sqlCustomer;

public delegate void OpenSubFormRequestHandler(Form subForm);

public event OpenSubFormRequestHandler OpenSubFormRequest;

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

{

// TODO: This line of code loads data into the 'roadTripRentalsDataSet.Customer' table. You can move, or remove it, as needed.

connStr = @"Data Source = DESKTOP-ASEMACC\INTHEDOGHOUSE; Initial Catalog = RoadTripRentals; Integrated Security = true";

sqlCustomer = @"select \* from Customer";

daCustomer = new SqlDataAdapter(sqlCustomer, connStr);

cmdBCustomer = new SqlCommandBuilder(daCustomer);

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

daCustomer.Fill(dsRoadTripRentals, "Customer");

//dgvCustomers.DataSource = dsRoadTripRentals.Tables["Customer"];

//Resize the DataGridView columns to fit the newly loaded content.

//dgvCustomers.AutoResizeColumns(DataGridViewAutoSizeColumnsMode.AllCells);

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

if (noRows == 0)

lblAddCustNoValue.Text = "10000";

else

{

getNumber();

}

errP.Clear();

clearAddForm();

}

public frmAddCustomer()

{

InitializeComponent();

}

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

{

frmMainCustomer newSubForm = new frmMainCustomer();

OpenSubFormInPanel(newSubForm);

}

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

{

MyCustomer myCustomer = new MyCustomer();

bool ok = true;

errP.Clear();

//Exception catching

//CUSTOMER ID

try

{

myCustomer.CustomerID = Convert.ToInt32(lblAddCustNoValue.Text.Trim()); //passed to Customer class to check

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(lblAddCustomerNo, MyEx.validate());

}

//TITLE

try

{

myCustomer.Title = cmbAddTitle.Text.Trim(); //passed to Customer class to check

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(cmbAddTitle, MyEx.validate());

}

//SURNAME

try

{

myCustomer.Surname = txtAddSurname.Text.Trim(); //passed to Customer class to check

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtAddSurname, MyEx.validate());

}

//FORENAME

try

{

myCustomer.Forename = txtAddForename.Text.Trim(); //passed to Customer class to check

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtAddForename, MyEx.validate());

}

//STREET NAME

try

{

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

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtAddStreet, MyEx.validate());

}

//CITY/TOWN

try

{

myCustomer.Town = txtAddTown.Text.Trim(); //passed to Customer class to check

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtAddTown, MyEx.validate());

}

//COUNTY

try

{

myCustomer.County = txtAddCounty.Text.Trim(); //passed to Customer class to check

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtAddCounty, MyEx.validate());

}

//POSTCODE

try

{

myCustomer.Postcode = txtAddPostcode.Text.Trim(); //passed to Customer class to check

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtAddPostcode, MyEx.validate());

}

//EMAIL

try

{

myCustomer.Email = txtAddEmail.Text.Trim(); //passed to Customer class to check

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtAddEmail, MyEx.validate());

}

//TELNO

try

{

myCustomer.TelNo = txtAddTelNo.Text.Trim(); //passed to Customer class to check

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtAddTelNo, MyEx.validate());

}

try

{

if (ok)

{

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

drCustomer["CustomerID"] = myCustomer.CustomerID;

drCustomer["Title"] = myCustomer.Title;

drCustomer["Forename"] = myCustomer.Forename;

drCustomer["Surname"] = myCustomer.Surname;

drCustomer["Street"] = myCustomer.Street;

drCustomer["City"] = myCustomer.Town;

drCustomer["County"] = myCustomer.County;

drCustomer["Postcode"] = myCustomer.Postcode;

drCustomer["EmailAddress"] = myCustomer.Email;

drCustomer["TelephoneNo"] = myCustomer.TelNo;

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

daCustomer.Update(dsRoadTripRentals, "Customer");

MessageBox.Show("Customer Added");

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

{

clearAddForm();

getNumber();

}

else

{

frmMainCustomer newSubForm = new frmMainCustomer();

OpenSubFormInPanel(newSubForm);

}

}

}

catch (Exception ex)

{

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

}

}

private void getNumber()

{

try

{

using (SqlConnection conn = new SqlConnection(connStr))

{

conn.Open();

SqlCommand cmd = new SqlCommand("SELECT MAX(CustomerID) FROM Customer", conn);

int maxCustomerId = (int)cmd.ExecuteScalar();

lblAddCustNoValue.Text = (maxCustomerId + 1).ToString();

conn.Close();

}

}

catch (Exception ex)

{

MessageBox.Show("Error fetching the next CustomerID: " + ex.Message);

}

}

void clearAddForm()

{

cmbAddTitle.SelectedIndex = -1;

txtAddForename.Clear();

txtAddSurname.Clear();

txtAddStreet.Clear();

txtAddTown.Clear();

txtAddCounty.Clear();

txtAddPostcode.Clear();

txtAddEmail.Clear();

txtAddTelNo.Clear();

}

private void OpenSubFormInPanel(Form subForm)

{

subForm.TopLevel = false;

subForm.FormBorderStyle = FormBorderStyle.None;

subForm.Dock = DockStyle.Fill;

// Replace 'panel1' with the name of the Panel control in your main form.

panelSub.Controls.Clear();

panelSub.Controls.Add(subForm);

subForm.Show();

}

}

}

## Edit Customer

**A screenshot of a computer

Description automatically generated with medium confidence**

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 RoadTripRentals.Forms.Jordan

{

public partial class frmEditCustomer : Form

{

SqlDataAdapter daCustomer, daCustomers;

DataSet dsRoadTripRentals = new DataSet();

SqlCommandBuilder cmdBCustomer;

SqlCommand cmdCustomerDetails;

DataRow drCustomer;

String connStr, sqlCustomer;

bool custSelected = false;

int custNoSelected = 0;

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

{

// TODO: This line of code loads data into the 'roadTripRentalsDataSet.Customer' table. You can move, or remove it, as needed.

connStr = @"Data Source = DESKTOP-ASEMACC\INTHEDOGHOUSE; Initial Catalog = RoadTripRentals; Integrated Security = true";

sqlCustomer = @"select \* from Customer";

daCustomer = new SqlDataAdapter(sqlCustomer, connStr);

cmdBCustomer = new SqlCommandBuilder(daCustomer);

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

daCustomer.Fill(dsRoadTripRentals, "Customer");

//dgvCustomers.DataSource = dsRoadTripRentals.Tables["Customer"];

//Resize the DataGridView columns to fit the newly loaded content.

//dgvCustomers.AutoResizeColumns(DataGridViewAutoSizeColumnsMode.AllCells);

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

LoadCustomerData(custNoSelected);

}

public frmEditCustomer(int customerID)

{

InitializeComponent();

custNoSelected = customerID;

}

private void LoadCustomerData(int customerID)

{

DataRow customerRow = dsRoadTripRentals.Tables["Customer"].Rows.Find(customerID);

if (customerRow != null)

{

lblEditCustNoValue.Text = customerRow["CustomerID"].ToString();

cmbEditTitle.Text = customerRow["Title"].ToString();

txtEditForename.Text = customerRow["Forename"].ToString();

txtEditSurname.Text = customerRow["Surname"].ToString();

txtEditStreet.Text = customerRow["Street"].ToString();

txtEditTown.Text = customerRow["City"].ToString();

txtEditCounty.Text = customerRow["County"].ToString();

txtEditPostcode.Text = customerRow["Postcode"].ToString();

txtEditEmail.Text = customerRow["EmailAddress"].ToString();

txtEditTelNo.Text = customerRow["TelephoneNo"].ToString();

}

else

{

MessageBox.Show("Customer not found.");

}

}

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

{

frmMainCustomer newSubForm = new frmMainCustomer();

OpenSubFormInPanel(newSubForm);

}

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

{

MyCustomer myCustomer = new MyCustomer();

bool ok = true;

errP.Clear();

//Exception catching

//CUSTOMER ID

try

{

myCustomer.CustomerID = Convert.ToInt32(lblEditCustNoValue.Text.Trim()); //passed to Customer class to check

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(lblEditCustomerNo, MyEx.validate());

}

//TITLE

try

{

myCustomer.Title = cmbEditTitle.Text.Trim(); //passed to Customer class to check

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(cmbEditTitle, MyEx.validate());

}

//SURNAME

try

{

myCustomer.Surname = txtEditSurname.Text.Trim(); //passed to Customer class to check

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtEditSurname, MyEx.validate());

}

//FORENAME

try

{

myCustomer.Forename = txtEditForename.Text.Trim(); //passed to Customer class to check

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtEditForename, MyEx.validate());

}

//STREET NAME

try

{

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

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtEditStreet, MyEx.validate());

}

//CITY/TOWN

try

{

myCustomer.Town = txtEditTown.Text.Trim(); //passed to Customer class to check

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtEditTown, MyEx.validate());

}

//COUNTY

try

{

myCustomer.County = txtEditCounty.Text.Trim(); //passed to Customer class to check

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtEditCounty, MyEx.validate());

}

//POSTCODE

try

{

myCustomer.Postcode = txtEditPostcode.Text.Trim(); //passed to Customer class to check

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtEditPostcode, MyEx.validate());

}

//EMAIL

try

{

myCustomer.Email = txtEditEmail.Text.Trim(); //passed to Customer class to check

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtEditEmail, MyEx.validate());

}

//TELNO

try

{

myCustomer.TelNo = txtEditTelNo.Text.Trim(); //passed to Customer class to check

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtEditTelNo, MyEx.validate());

}

try

{

if (ok)

{

int customerID = Convert.ToInt32(lblEditCustNoValue.Text);

DataRow customerRow = dsRoadTripRentals.Tables["Customer"].Rows.Find(customerID);

if (customerRow != null)

{

customerRow["Title"] = myCustomer.Title;

customerRow["Forename"] = myCustomer.Forename;

customerRow["Surname"] = myCustomer.Surname;

customerRow["Street"] = myCustomer.Street;

customerRow["City"] = myCustomer.Town;

customerRow["County"] = myCustomer.County;

customerRow["Postcode"] = myCustomer.Postcode;

customerRow["EmailAddress"] = myCustomer.Email;

customerRow["TelephoneNo"] = myCustomer.TelNo;

daCustomer.Update(dsRoadTripRentals, "Customer");

MessageBox.Show("Customer Updated");

}

else

{

MessageBox.Show("Customer not found.");

}

}

}

catch (Exception ex)

{

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

}

}

void clearAddForm()

{

cmbEditTitle.SelectedIndex = -1;

txtEditForename.Clear();

txtEditSurname.Clear();

txtEditStreet.Clear();

txtEditTown.Clear();

txtEditCounty.Clear();

txtEditPostcode.Clear();

txtEditEmail.Clear();

txtEditTelNo.Clear();

}

private void OpenSubFormInPanel(Form subForm)

{

subForm.TopLevel = false;

subForm.FormBorderStyle = FormBorderStyle.None;

subForm.Dock = DockStyle.Fill;

// Replace 'panelSub' with the name of the Panel control in your main form.

panelSub.Controls.Clear();

panelSub.Controls.Add(subForm);

subForm.Show();

}

}

}

## Main Rentals

**A screenshot of a rental application

Description automatically generated with medium confidence**

Rentals menu which displays the database containing each of the rentals. Can navigate to the add and edit forms from here which will replace the panel with the relevant form. Edit and Delete will only work if a row has been selected. The close sub form button also becomes visible, as an extra option to reset back to the main menu.

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 RoadTripRentals.Forms.Jordan

{

public partial class frmMainRentals : Form

{

SqlDataAdapter daRental, daRentals;

DataSet dsRoadTripRentals = new DataSet();

SqlCommandBuilder cmdBRental;

SqlCommand cmdRentalDetails;

DataRow drRental;

SqlConnection conn;

String connStr, sqlRental, sqlRentalDetails;

int selectedTab = 0;

bool custSelected = false;

int custNoSelected = 0;

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

{

// TODO: This line of code loads data into the 'roadTripRentalsDataSet.Rental' table. You can move, or remove it, as needed.

connStr = @"Data Source = DESKTOP-ASEMACC\INTHEDOGHOUSE; Initial Catalog = RoadTripRentals; Integrated Security = true";

sqlRental = @"select \* from Rental";

daRental = new SqlDataAdapter(sqlRental, connStr);

cmdBRental = new SqlCommandBuilder(daRental);

daRental.FillSchema(dsRoadTripRentals, SchemaType.Source, "Rental");

daRental.Fill(dsRoadTripRentals, "Rental");

dgvRentals.DataSource = dsRoadTripRentals.Tables["Rental"];

//Resize the DataGridView columns to fit the newly loaded content.

dgvRentals.AutoResizeColumns(DataGridViewAutoSizeColumnsMode.AllCells);

//btnDelRental.Click += btnDelRental\_Click;

}

public delegate void OpenSubFormRequestHandler(Form subForm);

public event OpenSubFormRequestHandler OpenSubFormRequest;

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

{

frmEditRental newSubForm = new frmEditRental();

OpenSubFormInPanel(newSubForm);

}

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

{

if (dgvRentals.SelectedRows.Count > 0)

{

int RentalID = Convert.ToInt32(dgvRentals.SelectedRows[0].Cells["RentalID"].Value);

// Ask for confirmation before deleting the Rental

DialogResult result = MessageBox.Show("Are you sure you want to delete the selected Rental type?", "Confirm Delete", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (result == DialogResult.Yes)

{

// Find the Rental in the dataset using LINQ

DataRow RentalRow = dsRoadTripRentals.Tables["Rental"].AsEnumerable().SingleOrDefault(row => row.Field<int>("RentalID") == RentalID);

if (RentalRow != null)

{

// Delete the Rental from the dataset and update the database

RentalRow.Delete();

daRental.Update(dsRoadTripRentals, "Rental");

MessageBox.Show("Rental Deleted");

}

else

{

MessageBox.Show("Rental not found.");

}

}

}

else

{

MessageBox.Show("Please select a Rental to delete.");

}

}

public frmMainRentals()

{

InitializeComponent();

}

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

{

frmAddRentals addRentalForm = new frmAddRentals();

OpenSubFormInPanel(addRentalForm);

}

private void OpenSubFormInPanel(Form subForm)

{

subForm.TopLevel = false;

subForm.FormBorderStyle = FormBorderStyle.None;

subForm.Dock = DockStyle.Fill;

// Replace 'panel1' with the name of the Panel control in your main form.

panelSub.Controls.Clear();

panelSub.Controls.Add(subForm);

subForm.Show();

}

}

}

## Add Rentals

**A screenshot of a car rental application

Description automatically generated with medium confidence**

The add rentals form brings in all of the tables Jordan has created.The buttons at the top are used to sort the lstCustomer list by surname. Once the customer is clicked the search parameter panel and Cars table becomes enabled. The user can either narrow their search down by model or cost (these fields are populated only with cars that actually have these traits. The no seats further reduces the details in the table. The user has to then click on the row for the car they want, and then hit the confirm check box to enable the booking date. Total cost is calculated by NoDays \* Rental cost. Once these fields are populated the Payment panel becomes available. This is the same as the Add Payment form (except it is requiring a payment date, which is tied to the Rental table ). The add car becomes enabled and populates the lvwRental with the relevant info. The confirm booking button then becomes enabled, and allows the user to add a rental with an associated customer, car and payment type. If there are multiple cars added for one rental, the details are still populated in the RentalCar table aswell.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Data.SqlTypes;

using System.Drawing;

using System.Drawing.Text;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace RoadTripRentals.Forms.Jordan

{

public partial class frmAddRentals : Form

{

Button[] btns = new Button[26];

SqlDataAdapter daNames, daCustomers, daRental, daRentalCost, daRentalCar;

DataSet dsRoadTripRentals = new DataSet();

SqlConnection conn;

SqlCommand cmdCustomerDetails;

DataRow drCustomer;

String sqlNames, sqlCustomerDetails, sqlRental;

String connStr;

SqlDataAdapter daCarDetails, daCarDetailss;

SqlCommandBuilder cmdBCarDetails, cmdBRentalCar;

SqlCommand cmdCarDetails;

DataRow drCarDetails;

String sqlCarDetails, sqlCarDetailsDetails;

SqlDataAdapter daPayment, daPaymentss;

SqlCommandBuilder cmdBPayment, cmdBRental, cmdBRentalCost;

SqlCommand cmdPaymentsDetails;

DataRow drPayment;

String sqlRentalCar, sqlRentalCost, sqlPayment, sqlPaymentsDetails;

private int previousSelectedCustomer = -1;

private bool isMessageBoxShown = false;

public frmAddRentals()

{

InitializeComponent();

Load += frmAddRentals\_Load;

}

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

{

btnAdd.Enabled = false;

ClearCustomer();

btnAddItem.Enabled = false;

int no;

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

dtpStartDate.MinDate = DateTime.Now;

dtpPaymentDate.MinDate = DateTime.Now;

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

{

var btn = (Button)pnlButtons.Controls["btn" + (char)(65 + i)]; // access buttons by name

btn.Text = "" + (char)(65 + i);

btn.Enabled = false;

btn.Click += new EventHandler(button1\_Click);

btns[i] = btn;

}

connStr = @"Data Source = DESKTOP-ASEMACC\INTHEDOGHOUSE; Initial Catalog = RoadTripRentals; Integrated Security = true";

//connStr = Properties.Resources.connectionStr;

//get surnames for alphabet buttons

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

daNames = new SqlDataAdapter(sqlNames, connStr);

daNames.Fill(dsRoadTripRentals, "Names");

//enable relevant alpha buttons

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

{

no = (int)dr["Surname"].ToString()[0] - 65;

btns[no].Enabled = true;

btns[no].BackColor = Color.Black;

btns[no].BackColor = Color.White;

}

// setup dataAdapter for customer details for the listbox

sqlCustomerDetails = @"Select customerID, title, surname, forename, surname +', ' + Forename as name, street, city, county, postcode, postcode, telephoneno from customer where surname LIKE @Letter order by surname, forename ";

conn = new SqlConnection(connStr);

cmdCustomerDetails = new SqlCommand(sqlCustomerDetails, conn);

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

daCustomers = new SqlDataAdapter(cmdCustomerDetails);

daCustomers.FillSchema(dsRoadTripRentals, SchemaType.Source, "Customer");

//CarDetails DGV

sqlCarDetails = @"select \* from CarDetails";

daCarDetails = new SqlDataAdapter(sqlCarDetails, connStr);

cmdBCarDetails = new SqlCommandBuilder(daCarDetails);

daCarDetails.FillSchema(dsRoadTripRentals, SchemaType.Source, "CarDetails");

daCarDetails.Fill(dsRoadTripRentals, "CarDetails");

dgvCars.DataSource = dsRoadTripRentals.Tables["CarDetails"];

//Resize the DataGridView columns to fit the newly loaded content.

dgvCars.AutoResizeColumns(DataGridViewAutoSizeColumnsMode.AllCells);

//Rental Cost

sqlRentalCost = @"select \* from RentalCost";

daRentalCost = new SqlDataAdapter(sqlRentalCost, connStr);

cmdBRentalCost = new SqlCommandBuilder(daRentalCost);

daRentalCost.FillSchema(dsRoadTripRentals, SchemaType.Source, "RentalCost");

daRentalCost.Fill(dsRoadTripRentals, "RentalCost");

// Rental table

sqlRental = @"select \* from Rental";

daRental = new SqlDataAdapter(sqlRental, connStr);

cmdBRental = new SqlCommandBuilder(daRental);

daRental.FillSchema(dsRoadTripRentals, SchemaType.Source, "Rental");

daRental.Fill(dsRoadTripRentals, "Rental");

// RentalCar table

sqlRentalCar = @"select \* from RentalCar";

daRentalCar = new SqlDataAdapter(sqlRentalCar, connStr);

cmdBRentalCar = new SqlCommandBuilder(daRentalCar);

daRentalCar.FillSchema(dsRoadTripRentals, SchemaType.Source, "RentalCar");

daRentalCar.Fill(dsRoadTripRentals, "RentalCar");

// Payment table

sqlPayment = @"select \* from PaymentType";

daPayment = new SqlDataAdapter(sqlPayment, connStr);

cmdBPayment = new SqlCommandBuilder(daPayment);

daPayment.FillSchema(dsRoadTripRentals, SchemaType.Source, "PaymentType");

daPayment.Fill(dsRoadTripRentals, "Payment");

int noRows = dsRoadTripRentals.Tables["PaymentType"].Rows.Count;

getNumber();

using (SqlConnection conn = new SqlConnection(connStr))

{

conn.Open();

using (SqlDataAdapter da = new SqlDataAdapter("SELECT DISTINCT Model.ModelID, Model.ModelDesc FROM Model INNER JOIN CarDetails ON Model.ModelID = CarDetails.ModelID", conn))

{

DataTable dt = new DataTable();

da.Fill(dt);

cmbAddModelID.DataSource = dt;

cmbAddModelID.DisplayMember = "ModelDesc";

cmbAddModelID.ValueMember = "ModelID";

}

using (SqlDataAdapter da = new SqlDataAdapter(@"

SELECT RentalCostID, RentalCost

FROM RentalCost

WHERE RentalCostID IN (

SELECT DISTINCT RentalCostID

FROM CarDetails

)

", conn))

{

DataTable dt = new DataTable();

da.Fill(dt);

cmbAddRentalCostID.DataSource = dt;

cmbAddRentalCostID.DisplayMember = "RentalCost";

cmbAddRentalCostID.ValueMember = "RentalCostID";

}

}

// Clear the car details labels

lblCarReg.Text = "";

lblModelID.Text = "";

lblColour.Text = "";

lblMileage.Text = "";

lblFuelType.Text = "";

lblNoSeats.Text = "";

lblYear.Text = "";

lblRentalCostID.Text = "";

v.Enabled = false;

v.Visible = false;

pnlBooking.Enabled = false;

pnlPayment.Enabled = false;

btnResetFilters.Visible = false;

}

private void button1\_Click(object sender, EventArgs e)

{

panelCarDetails.Enabled = false;

Button b = (Button)sender;

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

String str = b.Text;

//empty dataset table customer

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

fillListBoxCustomers(str);

ClearCustomer();

pnlBooking.Enabled = false;

}

private void fillListBoxCustomers(String str)

{

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

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

daCustomers.Fill(dsRoadTripRentals, "Customer");

//fill listbox

lstCustomer.DataSource = dsRoadTripRentals.Tables["Customer"];

lstCustomer.DisplayMember = "name";

lstCustomer.ValueMember = "CustomerID";

}

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

{

btnResetFilters.Visible = true;

if (cmbAddModelID.SelectedItem != null)

{

string modelID = cmbAddModelID.SelectedValue.ToString();

int noSeats = (int)txtAddNoSeats.Value;

string rentalCostID = cmbAddRentalCostID.SelectedValue != null ? cmbAddRentalCostID.SelectedValue.ToString() : null;

sqlCarDetails = "SELECT \* FROM CarDetails WHERE ModelID = @ModelID AND NoSeats = @NoSeats" + (rentalCostID != null ? " AND RentalCostID = @RentalCostID" : "");

daCarDetails = new SqlDataAdapter(sqlCarDetails, conn);

daCarDetails.SelectCommand.Parameters.AddWithValue("@ModelID", modelID);

daCarDetails.SelectCommand.Parameters.AddWithValue("@NoSeats", noSeats);

if (rentalCostID != null)

{

daCarDetails.SelectCommand.Parameters.AddWithValue("@RentalCostID", rentalCostID);

}

// Ensure the DataSet and DataTable are not null before trying to clear it.

if (dsRoadTripRentals != null && dsRoadTripRentals.Tables.Contains("CarDetails"))

{

dsRoadTripRentals.Tables["CarDetails"].Clear();

daCarDetails.Fill(dsRoadTripRentals, "CarDetails");

dgvCars.DataSource = dsRoadTripRentals.Tables["CarDetails"];

}

else

{

// Log an error message or throw an exception.

Console.WriteLine("DataSet or DataTable is null.");

}

}

}

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

{

btnResetFilters.Visible = true;

if (cmbAddRentalCostID.SelectedItem != null)

{

if (cmbAddRentalCostID.SelectedValue is DataRowView) return;

string rentalCostID = cmbAddRentalCostID.SelectedValue.ToString();

string modelID = cmbAddModelID.SelectedValue.ToString();

int noSeats = (int)txtAddNoSeats.Value;

sqlCarDetails = @"

SELECT \*

FROM CarDetails

WHERE RentalCostID = @RentalCostID AND ModelID = @ModelID AND NoSeats = @NoSeats";

daCarDetails = new SqlDataAdapter(sqlCarDetails, conn);

daCarDetails.SelectCommand.Parameters.AddWithValue("@RentalCostID", rentalCostID);

daCarDetails.SelectCommand.Parameters.AddWithValue("@ModelID", modelID);

daCarDetails.SelectCommand.Parameters.AddWithValue("@NoSeats", noSeats);

// Ensure the DataSet and DataTable are not null before trying to clear it.

if (dsRoadTripRentals != null && dsRoadTripRentals.Tables.Contains("CarDetails"))

{

dsRoadTripRentals.Tables["CarDetails"].Clear();

daCarDetails.Fill(dsRoadTripRentals, "CarDetails");

dgvCars.DataSource = dsRoadTripRentals.Tables["CarDetails"];

}

else

{

// Log an error message or throw an exception.

Console.WriteLine("DataSet or DataTable is null.");

}

}

}

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

{

btnResetFilters.Visible = true;

cmbModelID\_SelectedIndexChanged(sender, e);

}

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

{

String title = "";

drCustomer = dsRoadTripRentals.Tables["Customer"].Rows.Find(lstCustomer.SelectedValue);

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

title = "Mr";

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

title = "Mrs";

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

title = "Miss";

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

title = "Ms";

lblCust0.Text = drCustomer["CustomerID"].ToString();

lblCust1.Text = title + " " + drCustomer["Forename"].ToString() + " " + drCustomer["Surname"].ToString();

lblCust2.Text = drCustomer["Street"].ToString();

lblCust3.Text = drCustomer["City"].ToString();

lblCust4.Text = drCustomer["County"].ToString();

lblCust5.Text = drCustomer["Postcode"].ToString();

panelCarDetails.Enabled = true;

dgvCars.ClearSelection();

v.Enabled = false;

v.Visible = false;

pnlBooking.Enabled = false;

// Populate dgvCars and display the Confirm Selection checkbox only after a customer is selected

if (lstCustomer.SelectedItems.Count > 0)

{

ResetCarDetails(); // Make sure this method populates dgvCars with all cars and doesn't clear it

// Enable the Confirm Selection checkbox

v.Enabled = true;

}

}

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

{

if (dgvCars.SelectedRows.Count > 0) // Make sure there is a selected row

{

DataGridViewRow row = dgvCars.SelectedRows[0];

// Assume you have labels or other controls to display the selected car details

lblCarReg.Text = row.Cells["CarReg"].Value.ToString();

lblModelID.Text = row.Cells["ModelID"].Value.ToString();

lblColour.Text = row.Cells["Colour"].Value.ToString();

lblMileage.Text = Convert.ToDecimal(row.Cells["Mileage"].Value).ToString();

lblFuelType.Text = row.Cells["FuelType"].Value.ToString();

lblNoSeats.Text = Convert.ToDecimal(row.Cells["NoSeats"].Value).ToString();

lblYear.Text = Convert.ToDecimal(row.Cells["Year"].Value).ToString();

lblRentalCostID.Text = row.Cells["RentalCostID"].Value.ToString();

// Enable the confirm selection checkbox

v.Enabled = true;

v.Visible = true;

CalculateTotalCost();

}

}

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

{

if (isMessageBoxShown)

{

isMessageBoxShown = false; // Reset the flag

return; // Skip the event handler

}

if (lvwRental.Items.Count > 0 && MessageBox.Show("If you proceed with a new customer, the bookings will be cleared. Do you want to proceed?", "Clear bookings", MessageBoxButtons.YesNo) == DialogResult.No)

{

isMessageBoxShown = true; // Set the flag

if (previousSelectedCustomer != -1)

{

lstCustomer.SelectedIndex = previousSelectedCustomer;

}

else

{

lstCustomer.ClearSelected();

}

}

else

{

btnAdd.Enabled = false;

lvwRental.Items.Clear();

previousSelectedCustomer = lstCustomer.SelectedIndex;

}

}

private void ResetCarDetails()

{

// If the DataSet and DataTable are not null, fill the DataGridView with all cars

if (dsRoadTripRentals != null && dsRoadTripRentals.Tables.Contains("CarDetails"))

{

dsRoadTripRentals.Tables["CarDetails"].Clear();

daCarDetails.SelectCommand.CommandText = "SELECT \* FROM CarDetails";

daCarDetails.Fill(dsRoadTripRentals, "CarDetails");

dgvCars.DataSource = dsRoadTripRentals.Tables["CarDetails"];

btnResetFilters.Visible = false;

}

else

{

// Log an error message or throw an exception.

Console.WriteLine("DataSet or DataTable is null.");

}

}

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

{

frmMainRentals newSubForm = new frmMainRentals();

OpenSubFormInPanel(newSubForm);

}

private void chkConfirmCarSelection\_CheckedChanged(object sender, EventArgs e)

{

if (v.Checked)

{

// If the selection is confirmed, enable the date booked DateTimePicker

pnlBooking.Enabled = true;

dgvCars.Enabled = false;

dgvCars.Visible = false;

panelCarDetails.Enabled = false;

pnlPayment.Enabled = false;

btnResetFilters.Visible = false;

UpdateAddItemButtonState();

lblSelectCar.Text = "Selected Car";

}

else

{

// If the selection is not confirmed, disable the date booked DateTimePicker

pnlBooking.Enabled = false;

dgvCars.Enabled = true;

dgvCars.Visible = true;

panelCarDetails.Enabled = true;

ResetCarDetails();

lblSelectCar.Text = "Select Car";

}

}

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

{

ResetCarDetails();

}

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

{

CalculateTotalCost();

pnlPayment.Enabled = true;

UpdateAddItemButtonState();

}

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

{

if (lvwRental.SelectedItems.Count != 0)

{

var item = lvwRental.SelectedItems[0];

lvwRental.Items.Remove(item);

}

}

public class MyPayments

{

public int PaymentID { get; set; }

public string PaymentType { get; set; }

}

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

{

MyPayments myPayments = new MyPayments();

// Generate a new PaymentID by finding a unique value not already present in the PaymentType table

int paymentId = GenerateUniquePaymentId();

if (paymentId == -1)

{

// Unable to generate a unique paymentId

MessageBox.Show("Failed to generate a unique PaymentID. Please try again.");

return;

}

DataRow drPayments = dsRoadTripRentals.Tables["PaymentType"].NewRow();

drPayments["PaymentID"] = paymentId;

drPayments["PaymentType"] = cmbPaymentType.Text.Trim();

dsRoadTripRentals.Tables["PaymentType"].Rows.Add(drPayments);

daPayment.Update(dsRoadTripRentals, "PaymentType");

DataRow drRental;

int rentalId;

// Get the next RentalID

int noRows = dsRoadTripRentals.Tables["Rental"].Rows.Count;

drRental = dsRoadTripRentals.Tables["Rental"].Rows[noRows - 1];

rentalId = (int.Parse(drRental["RentalID"].ToString()) + 1);

// Create new Rental row

drRental = dsRoadTripRentals.Tables["Rental"].NewRow();

drRental["RentalID"] = rentalId;

drRental["CustomerID"] = int.Parse(lblCust0.Text);

drRental["StartDatetime"] = DateTime.Now;

drRental["NoDays"] = int.Parse(cmbNoOfDays.Text);

drRental["PaymentID"] = paymentId; // Assign the generated paymentId

drRental["PaymentDate"] = DateTime.Now; // Set the PaymentDate column to the current date and time

// Get the customer ID, car registration, and start date you're about to insert

int customerID = int.Parse(lblCust0.Text);

string carReg = dgvCars.SelectedRows[0].Cells["CarReg"].Value.ToString();

DateTime startDate = DateTime.Now; // If you're using another date, use that instead

foreach (ListViewItem item in lvwRental.Items)

{

carReg = item.Text;

int noDays = int.Parse(cmbNoOfDays.Text);

var existingRentals = dsRoadTripRentals.Tables["Rental"].AsEnumerable()

.Where(r => r.Field<int>("CustomerID") == customerID)

.Join(dsRoadTripRentals.Tables["RentalCar"].AsEnumerable(),

rental => rental.Field<int>("RentalID"),

rentalCar => rentalCar.Field<int>("RentalID"),

(rental, rentalCar) => new { Rental = rental, RentalCar = rentalCar })

.Where(joined => joined.RentalCar.Field<string>("CarReg") == carReg);

foreach (var existingRental in existingRentals)

{

DateTime existingStartDate = existingRental.Rental.Field<DateTime>("StartDatetime");

int existingNoDays = existingRental.Rental.Field<int>("NoDays");

DateTime existingEndDate = existingStartDate.AddDays(existingNoDays);

// Check if the new rental date overlaps with the existing rental period

if ((startDate >= existingStartDate && startDate <= existingEndDate) ||

(startDate.AddDays(noDays) >= existingStartDate && startDate.AddDays(noDays) <= existingEndDate))

{

MessageBox.Show("You cannot book the same car on overlapping dates.");

return;

}

}

}

dsRoadTripRentals.Tables["Rental"].Rows.Add(drRental);

daRental.Update(dsRoadTripRentals, "Rental");

// Create new RentalCar row for each car in the lvwRental ListView

foreach (ListViewItem item in lvwRental.Items)

{

DataRow drRentalCar = dsRoadTripRentals.Tables["RentalCar"].NewRow();

drRentalCar["RentalID"] = drRental["RentalID"];

drRentalCar["CarReg"] = item.Text;

//// Code to assign TotalCost

//string costText = item.SubItems[1].Text.Replace("£", string.Empty).Trim();

//drRentalCar["TotalCost"] = decimal.Parse(costText);

dsRoadTripRentals.Tables["RentalCar"].Rows.Add(drRentalCar);

daRentalCar.Update(dsRoadTripRentals, "RentalCar");

}

MessageBox.Show("Rental ID: " + drRental["RentalID"].ToString() + " added to the system");

DialogResult result = MessageBox.Show("Do you want to add another rental?", "Add Another Rental", MessageBoxButtons.YesNo);

if (result == DialogResult.Yes)

{

// Reset the form to its initial state

ResetForm();

btnAdd.Enabled = false;

}

else

{

frmMainRentals subForm = new frmMainRentals();

OpenSubFormInPanel(subForm);

}

}

private void ResetForm()

{

// Reset all the controls

cmbNoOfDays.SelectedIndex = -1;

cmbPaymentType.SelectedIndex = -1;

lblCust0.Text = "";

dgvCars.ClearSelection();

lvwRental.Items.Clear();

lblTotalCost.Text = "";

// ... etc for all other controls that need resetting

}

private void OpenSubFormInPanel(Form subForm)

{

subForm.TopLevel = false;

subForm.FormBorderStyle = FormBorderStyle.None;

subForm.Dock = DockStyle.Fill;

// Replace 'panel1' with the name of the Panel control in your main form.

panelSub.Controls.Clear();

panelSub.Controls.Add(subForm);

subForm.Show();

}

private int GenerateUniquePaymentId()

{

try

{

using (SqlConnection conn = new SqlConnection(connStr))

{

conn.Open();

SqlCommand cmd = new SqlCommand("SELECT MAX(PaymentID) FROM PaymentType", conn);

int maxPaymentID = (int)cmd.ExecuteScalar();

int paymentId = maxPaymentID + 1;

// Check if the generated paymentId already exists in the PaymentType table

while (CheckPaymentIdExists(paymentId))

{

paymentId++;

}

return paymentId;

}

}

catch (Exception ex)

{

MessageBox.Show("Error generating unique PaymentID: " + ex.Message);

return -1;

}

}

private bool CheckPaymentIdExists(int paymentId)

{

return dsRoadTripRentals.Tables["PaymentType"]

.AsEnumerable()

.Any(row => row.Field<int>("PaymentID") == paymentId);

}

private void getNumber()

{

try

{

using (SqlConnection conn = new SqlConnection(connStr))

{

conn.Open();

SqlCommand cmd = new SqlCommand("SELECT MAX(PaymentID) FROM PaymentType", conn);

int maxPaymentID = (int)cmd.ExecuteScalar();

txtPaymentID.Text = (maxPaymentID + 1).ToString();

conn.Close();

}

}

catch (Exception ex)

{

MessageBox.Show("Error fetching the next PaymentID: " + ex.Message);

}

}

private void CalculateTotalCost()

{

if (cmbNoOfDays.SelectedItem != null && dgvCars.SelectedRows.Count > 0)

{

decimal duration = Convert.ToDecimal(cmbNoOfDays.SelectedItem);

// Obtain the RentalCostID from the selected row in dgvCars

int rentalCostID = Convert.ToInt32(dgvCars.SelectedRows[0].Cells["RentalCostID"].Value);

// Lookup the corresponding RentalCost in the RentalCost

DataRow[] foundRows = dsRoadTripRentals.Tables["RentalCost"].Select("RentalCostID = " + rentalCostID);

if (foundRows.Length > 0)

{

decimal dailyRentalCost = Convert.ToDecimal(foundRows[0]["RentalCost"]);

decimal totalCost = duration \* dailyRentalCost;

lblTotalCost.Text = totalCost.ToString("C");

}

else

{

// Handle the case where no corresponding RentalCost is found for the given RentalCostID

MessageBox.Show("No corresponding Rental Cost found for the selected car.");

}

}

}

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

{

// Make sure all necessary data is selected

if (dgvCars.SelectedRows.Count > 0 && cmbNoOfDays.SelectedItem != null && cmbPaymentType.SelectedItem != null)

{

// Get the selected car's registration number

string carReg = dgvCars.SelectedRows[0].Cells["CarReg"].Value.ToString();

// Check if this car is already added to the ListView

foreach (ListViewItem item in lvwRental.Items)

{

if (item.SubItems[0].Text == carReg)

{

MessageBox.Show("This car is already added.", "Booking");

return; // Exit the method here

}

}

// Calculate the total cost

decimal duration = Convert.ToDecimal(cmbNoOfDays.SelectedItem);

DataRow[] carDetails = dsRoadTripRentals.Tables["CarDetails"].Select($"CarReg = '{carReg}'");

int rentalCostID = (int)carDetails[0]["RentalCostID"];

DataRow[] rentalCosts = dsRoadTripRentals.Tables["RentalCost"].Select($"RentalCostID = {rentalCostID}");

decimal dailyRentalCost = (decimal)rentalCosts[0]["RentalCost"];

decimal totalCost = duration \* dailyRentalCost;

// Get the selected payment type

string paymentType = cmbPaymentType.SelectedItem.ToString();

// Get the selected booking date

DateTime bookingDate = dtpStartDate.Value;

// Calculate the end date of the booking with a 1-day grace period

DateTime endDate = bookingDate.AddDays((double)duration + 1);

DateTime paymentDate = dtpPaymentDate.Value;

// Create a new ListViewItem with these details

ListViewItem listItem = new ListViewItem(new[] { carReg, totalCost.ToString("C"), paymentType, bookingDate.ToString("d"), duration.ToString(), paymentDate.ToString("d") });

// Add the item to the ListView

lvwRental.Items.Add(listItem);

pnlPayment.Enabled = false;

pnlBooking.Enabled = false;

// Reset car details after adding a new item

ResetCarDetails();

btnResetFilters.Visible = false;

// Clear selected car in dgvCars

dgvCars.ClearSelection();

// Reset labels

lblCarReg.Text = "";

lblModelID.Text = "";

lblColour.Text = "";

lblMileage.Text = "";

lblFuelType.Text = "";

lblNoSeats.Text = "";

lblYear.Text = "";

lblRentalCostID.Text = "";

// Disable the confirm selection checkbox

v.Checked = false;

v.Enabled = false;

v.Visible = false;

// Enable the cars DataGridView

dgvCars.Enabled = true;

dgvCars.Visible = true;

panelCarDetails.Enabled = true;

//Rental button confirmation enabled

btnAdd.Enabled = true;

}

else

{

MessageBox.Show("Please select a car, rental duration, payment type, and booking date.");

}

}

private void UpdateAddItemButtonState()

{

// Enable the Add Item button only if both panelBooking and pnlPayment are enabled

btnAddItem.Enabled = pnlBooking.Enabled && pnlPayment.Enabled;

}

private void ClearCustomer()

{

lstCustomer.SelectedIndex = -1;

lblCust0.Text = "";

lblCust1.Text = "";

lblCust2.Text = "";

lblCust3.Text = "";

lblCust4.Text = "";

lblCust5.Text = "";

}

}

}

## Edit / Delete Rentals

**A screenshot of a computer

Description automatically generated with medium confidence**

The edit / delete rentals form brings in all of the tables Jordan has created. This form is used to grab the existing rentals associated with a customer (and if the selected customer doesn’t have a rental, error message boxes appear, redirecting the user and resetting the form). The buttons at the top are used to sort the lstCustomer list by surname. The rentals box then becomes available for the user to select a desired rental which populates each of the fields which are grabbed from Rental and RentalCar respectively. The user can still choose a new car to add or remove, and includes all the functionality of the add form. A selected rental can also be deleted, which removes data from the Rentals, RentalCar, and payments tables as that info is no longer need.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Data.SqlTypes;

using System.Drawing;

using System.Drawing.Text;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace RoadTripRentals.Forms.Jordan

{

public partial class frmEditRental : Form

{

Button[] btns = new Button[26];

SqlDataAdapter daNames, daCustomers, daRental, daRentalCost, daRentalCar;

DataSet dsRoadTripRentals = new DataSet();

SqlConnection conn;

SqlCommand cmdCustomerDetails;

DataRow drCustomer;

String sqlNames, sqlCustomerDetails, sqlRental;

String connStr;

SqlDataAdapter daCarDetails, daCarDetailss;

SqlCommandBuilder cmdBCarDetails, cmdBRentalCar;

SqlCommand cmdCarDetail, cmdRental, cmdRentalCar, cmdRentalCars;

DataRow drCarDetails;

String sqlCarDetails, sqlCarDetailsDetails;

SqlDataAdapter daPayment, daPaymentType, daPaymentss;

SqlCommandBuilder cmdBPayment, cmdBRental, cmdBRentalCost;

SqlCommand cmdPaymentType;

DataRow drPayment, drRental;

String sqlRentalCar, sqlRentalCost, sqlPayment, sqlPaymentsDetails;

private int previousSelectedCustomer = -1;

private bool isMessageBoxShown = false;

public frmEditRental()

{

InitializeComponent();

Load += frmEditRental\_Load;

}

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

{

btnAdd.Enabled = false;

btnDelete.Enabled = false;

cmdRentalCar = new SqlCommand(sqlRentalCar, conn);

cmdRentalCar.Parameters.Add("@RentalID", SqlDbType.Int);

ClearCustomer();

btnAddItem.Enabled = false;

int no;

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

{

var btn = (Button)pnlButtons.Controls["btn" + (char)(65 + i)]; // access buttons by name

btn.Text = "" + (char)(65 + i);

btn.Enabled = false;

btn.Click += new EventHandler(button1\_Click);

btns[i] = btn;

}

connStr = @"Data Source = DESKTOP-ASEMACC\INTHEDOGHOUSE; Initial Catalog = RoadTripRentals; Integrated Security = true";

//connStr = Properties.Resources.connectionStr;

//get surnames for alphabet buttons

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

daNames = new SqlDataAdapter(sqlNames, connStr);

daNames.Fill(dsRoadTripRentals, "Names");

//enable relevant alpha buttons

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

{

no = (int)dr["Surname"].ToString()[0] - 65;

btns[no].Enabled = true;

btns[no].BackColor = Color.Black;

btns[no].BackColor = Color.White;

}

// setup dataAdapter for customer details for the listbox

sqlCustomerDetails = @"Select customerID, title, surname, forename, surname +', ' + Forename as name, street, city, county, postcode, postcode, telephoneno from customer where surname LIKE @Letter order by surname, forename ";

conn = new SqlConnection(connStr);

cmdCustomerDetails = new SqlCommand(sqlCustomerDetails, conn);

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

daCustomers = new SqlDataAdapter(cmdCustomerDetails);

daCustomers.FillSchema(dsRoadTripRentals, SchemaType.Source, "Customer");

//CarDetails DGV

sqlCarDetails = @"select \* from CarDetails";

daCarDetails = new SqlDataAdapter(sqlCarDetails, connStr);

cmdBCarDetails = new SqlCommandBuilder(daCarDetails);

daCarDetails.FillSchema(dsRoadTripRentals, SchemaType.Source, "CarDetails");

daCarDetails.Fill(dsRoadTripRentals, "CarDetails");

dgvCars.DataSource = dsRoadTripRentals.Tables["CarDetails"];

//Resize the DataGridView columns to fit the newly loaded content.

dgvCars.AutoResizeColumns(DataGridViewAutoSizeColumnsMode.AllCells);

//Rental Cost

sqlRentalCost = @"select \* from RentalCost";

daRentalCost = new SqlDataAdapter(sqlRentalCost, connStr);

cmdBRentalCost = new SqlCommandBuilder(daRentalCost);

daRentalCost.FillSchema(dsRoadTripRentals, SchemaType.Source, "RentalCost");

daRentalCost.Fill(dsRoadTripRentals, "RentalCost");

// Rental table

sqlRental = @"select \* from Rental";

daRental = new SqlDataAdapter(sqlRental, connStr);

cmdBRental = new SqlCommandBuilder(daRental);

daRental.FillSchema(dsRoadTripRentals, SchemaType.Source, "Rental");

daRental.Fill(dsRoadTripRentals, "Rental");

// RentalCar table

sqlRentalCar = @"select \* from RentalCar";

daRentalCar = new SqlDataAdapter(sqlRentalCar, connStr);

cmdBRentalCar = new SqlCommandBuilder(daRentalCar);

daRentalCar.FillSchema(dsRoadTripRentals, SchemaType.Source, "RentalCar");

daRentalCar.Fill(dsRoadTripRentals, "RentalCar");

// Payment table

sqlPayment = @"select \* from PaymentType";

daPayment = new SqlDataAdapter(sqlPayment, connStr);

cmdBPayment = new SqlCommandBuilder(daPayment);

daPayment.FillSchema(dsRoadTripRentals, SchemaType.Source, "PaymentType");

daPayment.Fill(dsRoadTripRentals, "Payment");

int noRows = dsRoadTripRentals.Tables["PaymentType"].Rows.Count;

getNumber();

using (SqlConnection conn = new SqlConnection(connStr))

{

conn.Open();

using (SqlDataAdapter da = new SqlDataAdapter("SELECT DISTINCT Model.ModelID, Model.ModelDesc FROM Model INNER JOIN CarDetails ON Model.ModelID = CarDetails.ModelID", conn))

{

DataTable dt = new DataTable();

da.Fill(dt);

cmbAddModelID.DataSource = dt;

cmbAddModelID.DisplayMember = "ModelDesc";

cmbAddModelID.ValueMember = "ModelID";

}

using (SqlDataAdapter da = new SqlDataAdapter(@"

SELECT RentalCostID, RentalCost

FROM RentalCost

WHERE RentalCostID IN (

SELECT DISTINCT RentalCostID

FROM CarDetails

)

", conn))

{

DataTable dt = new DataTable();

da.Fill(dt);

cmbAddRentalCostID.DataSource = dt;

cmbAddRentalCostID.DisplayMember = "RentalCost";

cmbAddRentalCostID.ValueMember = "RentalCostID";

}

}

// Clear the car details labels

lblCarReg.Text = "";

lblModelID.Text = "";

lblColour.Text = "";

lblMileage.Text = "";

lblFuelType.Text = "";

lblNoSeats.Text = "";

lblYear.Text = "";

lblRentalCostID.Text = "";

chkConfirmCarSelection.Enabled = false;

chkConfirmCarSelection.Visible = false;

pnlBooking.Enabled = false;

pnlPayment.Enabled = false;

btnResetFilters.Visible = false;

sqlRental = @"select \* from Rental where CustomerID = @CustomerID";

cmdRental = new SqlCommand(sqlRental, conn);

cmdRental.Parameters.Add("@CustomerID", SqlDbType.Int); // Assuming CustomerID is of type Int

daRental = new SqlDataAdapter(cmdRental);

daRental.FillSchema(dsRoadTripRentals, SchemaType.Source, "Rental");

// After initializing cmdRentalCar and cmdRental

cmdPaymentType = new SqlCommand("SELECT \* FROM PaymentType WHERE PaymentID = @PaymentID", conn);

cmdPaymentType.Parameters.Add("@PaymentID", SqlDbType.Int); // Assuming PaymentID is of type Int

daPaymentType = new SqlDataAdapter(cmdPaymentType);

daPaymentType.FillSchema(dsRoadTripRentals, SchemaType.Source, "PaymentType");

}

private void button1\_Click(object sender, EventArgs e)

{

btnAdd.Enabled = false;

btnDelete.Enabled = false;

panelCarDetails.Enabled = false;

Button b = (Button)sender;

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

String str = b.Text;

//empty dataset table customer

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

fillListBoxCustomers(str);

//clear any previously selected dogs/kennels by emptying the dataset tables

//dsRoadTripRentals.Tables["Dog"].Clear();

//dsRoadTripRentals.Tables["Kennel"].Clear();

ClearCustomer();

panel3.Enabled = true;

pnlBooking.Enabled = false;

}

private void PopulateRentalListbox()

{

cmdRental.Parameters.Clear();

cmdRental.Parameters.AddWithValue("@CustomerID", lstCustomer.SelectedValue);

daRental.SelectCommand = cmdRental;

dsRoadTripRentals.Tables["Rental"].Clear();

daRental.Fill(dsRoadTripRentals, "Rental");

lstRental.DataSource = dsRoadTripRentals.Tables["Rental"];

lstRental.DisplayMember = "RentalID";

lstRental.ValueMember = "RentalID";

lstRental.SelectedIndex = -1;

}

private void fillListBoxCustomers(String str)

{

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

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

daCustomers.Fill(dsRoadTripRentals, "Customer");

//fill listbox

lstCustomer.DataSource = dsRoadTripRentals.Tables["Customer"];

lstCustomer.DisplayMember = "name";

lstCustomer.ValueMember = "CustomerID";

}

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

{

btnResetFilters.Visible = true;

if (cmbAddModelID.SelectedItem != null)

{

string modelID = cmbAddModelID.SelectedValue.ToString();

int noSeats = (int)txtAddNoSeats.Value;

string rentalCostID = cmbAddRentalCostID.SelectedValue != null ? cmbAddRentalCostID.SelectedValue.ToString() : null;

sqlCarDetails = "SELECT \* FROM CarDetails WHERE ModelID = @ModelID AND NoSeats = @NoSeats" + (rentalCostID != null ? " AND RentalCostID = @RentalCostID" : "");

daCarDetails = new SqlDataAdapter(sqlCarDetails, conn);

daCarDetails.SelectCommand.Parameters.AddWithValue("@ModelID", modelID);

daCarDetails.SelectCommand.Parameters.AddWithValue("@NoSeats", noSeats);

if (rentalCostID != null)

{

daCarDetails.SelectCommand.Parameters.AddWithValue("@RentalCostID", rentalCostID);

}

// Ensure the DataSet and DataTable are not null before trying to clear it.

if (dsRoadTripRentals != null && dsRoadTripRentals.Tables.Contains("CarDetails"))

{

dsRoadTripRentals.Tables["CarDetails"].Clear();

daCarDetails.Fill(dsRoadTripRentals, "CarDetails");

dgvCars.DataSource = dsRoadTripRentals.Tables["CarDetails"];

}

else

{

// Log an error message or throw an exception.

Console.WriteLine("DataSet or DataTable is null.");

}

}

}

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

{

btnResetFilters.Visible = true;

if (cmbAddRentalCostID.SelectedItem != null)

{

if (cmbAddRentalCostID.SelectedValue is DataRowView) return;

string rentalCostID = cmbAddRentalCostID.SelectedValue.ToString();

string modelID = cmbAddModelID.SelectedValue.ToString();

int noSeats = (int)txtAddNoSeats.Value;

sqlCarDetails = @"

SELECT \*

FROM CarDetails

WHERE RentalCostID = @RentalCostID AND ModelID = @ModelID AND NoSeats = @NoSeats";

daCarDetails = new SqlDataAdapter(sqlCarDetails, conn);

daCarDetails.SelectCommand.Parameters.AddWithValue("@RentalCostID", rentalCostID);

daCarDetails.SelectCommand.Parameters.AddWithValue("@ModelID", modelID);

daCarDetails.SelectCommand.Parameters.AddWithValue("@NoSeats", noSeats);

// Ensure the DataSet and DataTable are not null before trying to clear it.

if (dsRoadTripRentals != null && dsRoadTripRentals.Tables.Contains("CarDetails"))

{

dsRoadTripRentals.Tables["CarDetails"].Clear();

daCarDetails.Fill(dsRoadTripRentals, "CarDetails");

dgvCars.DataSource = dsRoadTripRentals.Tables["CarDetails"];

}

else

{

// Log an error message or throw an exception.

Console.WriteLine("DataSet or DataTable is null.");

}

}

}

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

{

btnResetFilters.Visible = true;

cmbModelID\_SelectedIndexChanged(sender, e);

}

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

{

btnAdd.Enabled = false;

btnDelete.Enabled = false;

String title = "";

drCustomer = dsRoadTripRentals.Tables["Customer"].Rows.Find(lstCustomer.SelectedValue);

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

title = "Mr";

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

title = "Mrs";

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

title = "Miss";

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

title = "Ms";

lblCust0.Text = drCustomer["CustomerID"].ToString();

lblCust1.Text = title + " " + drCustomer["Forename"].ToString() + " " + drCustomer["Surname"].ToString();

lblCust2.Text = drCustomer["Street"].ToString();

lblCust3.Text = drCustomer["City"].ToString();

lblCust4.Text = drCustomer["County"].ToString();

lblCust5.Text = drCustomer["Postcode"].ToString();

panelCarDetails.Enabled = true;

dgvCars.ClearSelection();

chkConfirmCarSelection.Enabled = false;

chkConfirmCarSelection.Visible = false;

pnlBooking.Enabled = false;

// Populate dgvCars and display the Confirm Selection checkbox only after a customer is selected

if (lstCustomer.SelectedItems.Count > 0)

{

ResetCarDetails(); // Make sure this method populates dgvCars with all cars and doesn't clear it

// Enable the Confirm Selection checkbox

chkConfirmCarSelection.Enabled = true;

// Call the method to populate rentals for the selected customer

PopulateRentalListbox();

// Hide the customer if there are no bookings

if (lstRental.Items.Count == 0)

{

MessageBox.Show("No bookings found for this customer.", "Error");

lstCustomer.ClearSelected();

panel3.Enabled = false;

panelMajor.Enabled = false;

return;

}

}

}

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

{

if (dgvCars.SelectedRows.Count > 0) // Make sure there is a selected row

{

DataGridViewRow row = dgvCars.SelectedRows[0];

// Assume you have labels or other controls to display the selected car details

lblCarReg.Text = row.Cells["CarReg"].Value.ToString();

lblModelID.Text = row.Cells["ModelID"].Value.ToString();

lblColour.Text = row.Cells["Colour"].Value.ToString();

lblMileage.Text = Convert.ToDecimal(row.Cells["Mileage"].Value).ToString();

lblFuelType.Text = row.Cells["FuelType"].Value.ToString();

lblNoSeats.Text = Convert.ToDecimal(row.Cells["NoSeats"].Value).ToString();

lblYear.Text = Convert.ToDecimal(row.Cells["Year"].Value).ToString();

lblRentalCostID.Text = row.Cells["RentalCostID"].Value.ToString();

// Enable the confirm selection checkbox

chkConfirmCarSelection.Enabled = true;

chkConfirmCarSelection.Visible = true;

CalculateTotalCost();

}

}

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

{

if (isMessageBoxShown)

{

isMessageBoxShown = false;

return; // Skip the event handler

}

if (lvwRental.Items.Count > 0 && MessageBox.Show("If you proceed with a new customer, the bookings will be cleared. Do you want to proceed?", "Clear bookings", MessageBoxButtons.YesNo) == DialogResult.No)

{

isMessageBoxShown = true;

if (previousSelectedCustomer != -1)

{

lstCustomer.SelectedIndex = previousSelectedCustomer;

}

else

{

lstCustomer.ClearSelected();

btnAdd.Enabled = false;

btnDelete.Enabled = false;

}

}

else

{

lvwRental.Items.Clear();

previousSelectedCustomer = lstCustomer.SelectedIndex;

}

}

private void ResetCarDetails()

{

// If the DataSet and DataTable are not null, fill the DataGridView with all cars

if (dsRoadTripRentals != null && dsRoadTripRentals.Tables.Contains("CarDetails"))

{

dsRoadTripRentals.Tables["CarDetails"].Clear();

daCarDetails.SelectCommand.CommandText = "SELECT \* FROM CarDetails";

daCarDetails.Fill(dsRoadTripRentals, "CarDetails");

dgvCars.DataSource = dsRoadTripRentals.Tables["CarDetails"];

btnResetFilters.Visible = false;

}

else

{

// Log an error message or throw an exception.

Console.WriteLine("DataSet or DataTable is null.");

}

}

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

{

int rentalId = Convert.ToInt32(lstRental.SelectedValue);

// Delete RentalCar rows for the specified RentalID

DataRow[] rentalCarRows = dsRoadTripRentals.Tables["RentalCar"].Select($"RentalID = {rentalId}");

foreach (DataRow rentalCarRow in rentalCarRows)

{

rentalCarRow.Delete();

}

// Delete Rental row for the specified RentalID

DataRow rentalRow = dsRoadTripRentals.Tables["Rental"].Rows.Find(rentalId);

if (rentalRow != null)

{

rentalRow.Delete();

}

// Delete PaymentType rows for the specified PaymentID

int paymentId = Convert.ToInt32(txtPaymentID.Text);

DataRow[] paymentRows = dsRoadTripRentals.Tables["PaymentType"].Select($"PaymentID = {paymentId}");

foreach (DataRow paymentRow in paymentRows)

{

paymentRow.Delete();

}

// Create the delete commands for the data adapter

SqlCommand deleteRentalCarCommand = new SqlCommand("DELETE FROM RentalCar WHERE RentalID = @RentalID");

deleteRentalCarCommand.Parameters.AddWithValue("@RentalID", rentalId);

SqlCommand deleteRentalCommand = new SqlCommand("DELETE FROM Rental WHERE RentalID = @RentalID");

deleteRentalCommand.Parameters.AddWithValue("@RentalID", rentalId);

SqlCommand deletePaymentCommand = new SqlCommand("DELETE FROM PaymentType WHERE PaymentID = @PaymentID");

deletePaymentCommand.Parameters.AddWithValue("@PaymentID", paymentId);

// Assign the delete commands to the data adapter

daRentalCar.DeleteCommand = deleteRentalCarCommand;

daRental.DeleteCommand = deleteRentalCommand;

daPayment.DeleteCommand = deletePaymentCommand;

// Assign the connection object to the delete commands

daRentalCar.DeleteCommand.Connection = conn;

daRental.DeleteCommand.Connection = conn;

daPayment.DeleteCommand.Connection = conn;

// Perform the updates on the dataset

daRentalCar.Update(dsRoadTripRentals, "RentalCar");

daRental.Update(dsRoadTripRentals, "Rental");

daPayment.Update(dsRoadTripRentals, "PaymentType");

using (SqlConnection conn = new SqlConnection(connStr))

{

// Assign the connection object to the UpdateCommand if it is not null

if (daRentalCar.UpdateCommand != null)

{

daRentalCar.UpdateCommand.Connection = conn;

}

// Open the connection

conn.Open();

// Update the dataset with the changes

daRental.Update(dsRoadTripRentals, "Rental");

daRentalCar.Update(dsRoadTripRentals, "RentalCar");

daPayment.Update(dsRoadTripRentals, "PaymentType");

// Clear the dataset and refill it with fresh data

dsRoadTripRentals.Clear();

daRental.Fill(dsRoadTripRentals, "Rental");

daRentalCar.Fill(dsRoadTripRentals, "RentalCar");

daPayment.Fill(dsRoadTripRentals, "PaymentType");

// Show a message indicating successful deletion

MessageBox.Show("Rental has been deleted successfully.");

}

}

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

{

frmMainRentals newSubForm = new frmMainRentals();

OpenSubFormInPanel(newSubForm);

}

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

{

panelMajor.Enabled = true;

dtpStartDate.Enabled = true;

cmbNoOfDays.Enabled = true;

lvwRental.Enabled = true;

lvwRental.Items.Clear();

if (lstRental.Items.Count != 0)

{

DataRow drRental = dsRoadTripRentals.Tables["Rental"].Rows.Find(lstRental.SelectedValue);

txtPaymentID.Text = drRental["PaymentID"].ToString();

cmbNoOfDays.Text = drRental["NoDays"].ToString();

cmbNoOfDays.Text = drRental["NoDays"].ToString();

// Set the PaymentDate from the selected rental

if (drRental.Table.Columns.Contains("PaymentDate")) // Check if the PaymentDate column exists

{

if (!DBNull.Value.Equals(drRental["PaymentDate"])) // Check if the PaymentDate value is not null

{

dtpPaymentDate.Value = Convert.ToDateTime(drRental["PaymentDate"]);

}

else

{

dtpPaymentDate.Value = DateTime.Now;

}

}

// Get the paymentID from the selected rental

int paymentID = Convert.ToInt32(drRental["PaymentID"]);

// Retrieve PaymentType from the database based on PaymentID

string paymentQuery = "SELECT \* FROM PaymentType WHERE PaymentID = @PaymentID";

using (SqlConnection connection = new SqlConnection(connStr))

{

connection.Open();

using (SqlCommand command = new SqlCommand(paymentQuery, connection))

{

command.Parameters.AddWithValue("@PaymentID", paymentID);

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.Read())

{

cmbPaymentType.Text = reader["PaymentType"].ToString();

}

}

}

}

lblBookingDate.Text = (Convert.ToDateTime(drRental["StartDatetime"].ToString())).ToShortDateString();

dtpStartDate.MinDate = DateTime.MinValue; // Set MinDate to the earliest possible date

dtpStartDate.Value = Convert.ToDateTime(drRental["StartDatetime"].ToString());

dsRoadTripRentals.Tables["RentalCar"].Clear();

cmdRentalCar.Parameters["@RentalID"].Value = lstRental.SelectedValue;

daRentalCar.Fill(dsRoadTripRentals, "RentalCar");

foreach (DataRow dr in dsRoadTripRentals.Tables["RentalCar"].Rows)

{

if (dr["RentalID"].ToString() == lstRental.Text)

{

// Get car registration

string carReg = dr["CarReg"].ToString();

// Get rental duration

decimal duration = Convert.ToDecimal(cmbNoOfDays.Text);

// Get RentalCostID and daily rental cost for this car

DataRow[] carDetails = dsRoadTripRentals.Tables["CarDetails"].Select($"CarReg = '{carReg}'");

int rentalCostID = (int)carDetails[0]["RentalCostID"];

DataRow[] rentalCosts = dsRoadTripRentals.Tables["RentalCost"].Select($"RentalCostID = {rentalCostID}");

decimal dailyRentalCost = (decimal)rentalCosts[0]["RentalCost"];

// Calculate total cost

decimal totalCost = duration \* dailyRentalCost;

// Create a ListViewItem with the car registration and total cost

ListViewItem item = new ListViewItem(new[] { carReg, totalCost.ToString("C") });

lvwRental.Items.Add(item);

}

}

btnAdd.Enabled = true;

btnDelete.Enabled = true;

}

}

private void chkConfirmCarSelection\_CheckedChanged(object sender, EventArgs e)

{

if (chkConfirmCarSelection.Checked)

{

// If the selection is confirmed, enable the date booked DateTimePicker

pnlBooking.Enabled = true;

dgvCars.Enabled = false;

dgvCars.Visible = false;

panelCarDetails.Enabled = false;

pnlPayment.Enabled = false;

btnResetFilters.Visible = false;

UpdateAddItemButtonState();

lblSelectCar.Text = "Selected Car";

}

else

{

// If the selection is not confirmed, disable the date booked DateTimePicker

pnlBooking.Enabled = false;

dgvCars.Enabled = true;

dgvCars.Visible = true;

panelCarDetails.Enabled = true;

ResetCarDetails();

lblSelectCar.Text = "Select Car";

}

}

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

{

ResetCarDetails();

}

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

{

CalculateTotalCost();

pnlPayment.Enabled = true;

UpdateAddItemButtonState();

}

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

{

if (lvwRental.SelectedItems.Count != 0)

{

var item = lvwRental.SelectedItems[0];

lvwRental.Items.Remove(item);

}

}

public class MyPayments

{

public int PaymentID { get; set; }

public string PaymentType { get; set; }

}

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

{

int rentalId = Convert.ToInt32(lstRental.SelectedValue);

DataRow drRental = dsRoadTripRentals.Tables["Rental"].Rows.Find(rentalId);

// Update the Rental details

drRental["NoDays"] = int.Parse(cmbNoOfDays.Text);

// Get the customer ID, car registration, and start date

int customerId = int.Parse(lblCust0.Text);

DateTime startDate = DateTime.Now;

// Remove rental cars that are not in the updated lvwRental list

var existingRentalCars = dsRoadTripRentals.Tables["RentalCar"].Select($"RentalID = {rentalId}");

foreach (DataRow existingRentalCar in existingRentalCars)

{

string carReg = existingRentalCar["CarReg"].ToString();

bool found = false;

foreach (ListViewItem item in lvwRental.Items)

{

if (item.Text == carReg)

{

found = true;

break;

}

}

if (!found)

{

existingRentalCar.Delete();

}

}

// Update the RentalCar table based on the items in lvwRental

foreach (ListViewItem item in lvwRental.Items)

{

string carReg = item.Text;

var existingRentalCar = dsRoadTripRentals.Tables["RentalCar"].AsEnumerable()

.FirstOrDefault(row => row.Field<int>("RentalID") == rentalId && row.Field<string>("CarReg") == carReg);

if (existingRentalCar == null)

{

// If the rental car doesn't exist, create a new DataRow and add it to the DataTable

DataRow newRentalCar = dsRoadTripRentals.Tables["RentalCar"].NewRow();

newRentalCar["RentalID"] = rentalId;

newRentalCar["CarReg"] = carReg;

dsRoadTripRentals.Tables["RentalCar"].Rows.Add(newRentalCar);

}

}

// Create the update command for the Rental table

var updateRentalCommand = new SqlCommand("UPDATE Rental SET NoDays = @NoDays WHERE RentalID = @RentalID");

updateRentalCommand.Parameters.Add("@NoDays", SqlDbType.Int, 4, "NoDays");

updateRentalCommand.Parameters.Add("@RentalID", SqlDbType.Int, 4, "RentalID");

// Assign the update command to the data adapter

daRental.UpdateCommand = updateRentalCommand;

// Create a SqlConnection object and assign it to the UpdateCommand

using (SqlConnection conn = new SqlConnection(connStr))

{

daRental.UpdateCommand.Connection = conn;

// Open the connection

conn.Open();

// Update the dataset with the changes

daRental.Update(dsRoadTripRentals, "Rental");

daRentalCar.Update(dsRoadTripRentals, "RentalCar");

// Show a message indicating successful update

MessageBox.Show("Rental details have been updated successfully.");

}

}

private void ResetForm()

{

// Reset all the controls

cmbNoOfDays.SelectedIndex = -1;

cmbPaymentType.SelectedIndex = -1;

lblCust0.Text = "";

dgvCars.ClearSelection();

lvwRental.Items.Clear();

lblTotalCost.Text = "";

}

private void OpenSubFormInPanel(Form subForm)

{

subForm.TopLevel = false;

subForm.FormBorderStyle = FormBorderStyle.None;

subForm.Dock = DockStyle.Fill;

// Replace 'panel1' with the name of the Panel control in your main form.

panelSub.Controls.Clear();

panelSub.Controls.Add(subForm);

subForm.Show();

}

private void getNumber()

{

try

{

using (SqlConnection conn = new SqlConnection(connStr))

{

conn.Open();

SqlCommand cmd = new SqlCommand("SELECT MAX(PaymentID) FROM PaymentType", conn);

int maxPaymentID = (int)cmd.ExecuteScalar();

txtPaymentID.Text = (maxPaymentID + 1).ToString();

conn.Close();

}

}

catch (Exception ex)

{

MessageBox.Show("Error fetching the next PaymentID: " + ex.Message);

}

}

private void CalculateTotalCost()

{

if (cmbNoOfDays.SelectedItem != null && dgvCars.SelectedRows.Count > 0)

{

decimal duration = Convert.ToDecimal(cmbNoOfDays.SelectedItem);

// Obtain the RentalCostID from the selected row in dgvCars

int rentalCostID = Convert.ToInt32(dgvCars.SelectedRows[0].Cells["RentalCostID"].Value);

// Lookup the corresponding RentalCost in the RentalCost

DataRow[] foundRows = dsRoadTripRentals.Tables["RentalCost"].Select("RentalCostID = " + rentalCostID);

if (foundRows.Length > 0)

{

decimal dailyRentalCost = Convert.ToDecimal(foundRows[0]["RentalCost"]);

decimal totalCost = duration \* dailyRentalCost;

lblTotalCost.Text = totalCost.ToString("C");

}

else

{

// Handle the case where no corresponding RentalCost is found for the given RentalCostID

MessageBox.Show("No corresponding RentalCost found for the selected car.");

}

}

}

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

{

// Make sure all necessary data is selected

if (dgvCars.SelectedRows.Count > 0 && cmbNoOfDays.SelectedItem != null && cmbPaymentType.SelectedItem != null)

{

// Get the selected car's registration number

string carReg = dgvCars.SelectedRows[0].Cells["CarReg"].Value.ToString();

// Check if this car is already added to the ListView

foreach (ListViewItem item in lvwRental.Items)

{

if (item.SubItems[0].Text == carReg)

{

MessageBox.Show("This car is already added.", "Booking");

return; // Exit the method here

}

}

// Get the selected rental ID

int rentalId = Convert.ToInt32(lstRental.SelectedValue);

// Find the corresponding rental car row based on the car registration and rental ID

DataRow rentalCarRow = dsRoadTripRentals.Tables["RentalCar"].AsEnumerable()

.FirstOrDefault(row => row.Field<int>("RentalID") == rentalId && row.Field<string>("CarReg") == carReg);

// Find the corresponding car details row based on the car registration

DataRow carDetailsRow = dsRoadTripRentals.Tables["CarDetails"].AsEnumerable()

.FirstOrDefault(row => row.Field<string>("CarReg") == carReg);

// Retrieve the rental cost ID from the car details row

int rentalCostId = Convert.ToInt32(carDetailsRow["RentalCostID"]);

// Find the corresponding rental cost row based on the rental cost ID

DataRow rentalCostRow = dsRoadTripRentals.Tables["RentalCost"].AsEnumerable()

.FirstOrDefault(row => row.Field<int>("RentalCostID") == rentalCostId);

// Retrieve the rental cost from the rental cost row

decimal dailyRentalCost = Convert.ToDecimal(rentalCostRow["RentalCost"]);

// Calculate the total cost

decimal duration = Convert.ToDecimal(cmbNoOfDays.SelectedItem);

decimal totalCost = duration \* dailyRentalCost;

// Get the selected payment type

string paymentType = cmbPaymentType.SelectedItem.ToString();

// Get the selected booking date

DateTime bookingDate = dtpStartDate.Value;

// Calculate the end date of the booking with a 1-day grace period

DateTime endDate = bookingDate.AddDays((double)duration + 1);

DateTime paymentDate = dtpPaymentDate.Value;

// Create a new ListViewItem with these details

ListViewItem listItem = new ListViewItem(new[] { carReg, totalCost.ToString("C"), paymentType, bookingDate.ToString("d"), duration.ToString(), paymentDate.ToString("d") });

// Add the item to the ListView

lvwRental.Items.Add(listItem);

pnlPayment.Enabled = false;

pnlBooking.Enabled = false;

// Reset car details after adding a new item

ResetCarDetails();

btnResetFilters.Visible = false;

// Clear selected car in dgvCars

dgvCars.ClearSelection();

// Reset labels

lblCarReg.Text = "";

lblModelID.Text = "";

lblColour.Text = "";

lblMileage.Text = "";

lblFuelType.Text = "";

lblNoSeats.Text = "";

lblYear.Text = "";

lblRentalCostID.Text = "";

// Disable the confirm selection checkbox

chkConfirmCarSelection.Checked = false;

chkConfirmCarSelection.Enabled = false;

chkConfirmCarSelection.Visible = false;

// Enable the cars DataGridView

dgvCars.Enabled = true;

dgvCars.Visible = true;

panelCarDetails.Enabled = true;

}

else

{

MessageBox.Show("Please select a car, rental duration, payment type, and booking date.");

}

}

private void UpdateAddItemButtonState()

{

// Enable the Add Item button only if both panelBooking and pnlPayment are enabled

btnAddItem.Enabled = pnlBooking.Enabled && pnlPayment.Enabled;

}

private void ClearCustomer()

{

lstCustomer.SelectedIndex = -1;

lblCust0.Text = "";

lblCust1.Text = "";

lblCust2.Text = "";

lblCust3.Text = "";

lblCust4.Text = "";

lblCust5.Text = "";

}

}

}

## Main Car Details

**A screenshot of a computer

Description automatically generated with medium confidence**

Car details menu which displays the database containing each of the cars. Can navigate to the add and edit forms from here which will replace the panel with the relevant form. Edit and Delete will only work if a row has been selected. The close sub form button also becomes visible, as an extra option to reset back to the main menu.

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 RoadTripRentals.Forms.Jordan

{

public partial class frmMainCar : Form

{

SqlDataAdapter daCarDetails, daCarDetailss;

DataSet dsRoadTripRentals = new DataSet();

SqlCommandBuilder cmdBCarDetails;

SqlCommand cmdCarDetailsDetails;

DataRow drCarDetails;

SqlConnection conn;

String connStr, sqlCarDetails, sqlCarDetailsDetails;

int selectedTab = 0;

bool custSelected = false;

int custNoSelected = 0;

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

{

// TODO: This line of code loads data into the 'roadTripRentalsDataSet.Car' table. You can move, or remove it, as needed.

connStr = @"Data Source = DESKTOP-ASEMACC\INTHEDOGHOUSE; Initial Catalog = RoadTripRentals; Integrated Security = true";

sqlCarDetails = @"select \* from CarDetails";

daCarDetails = new SqlDataAdapter(sqlCarDetails, connStr);

cmdBCarDetails = new SqlCommandBuilder(daCarDetails);

daCarDetails.FillSchema(dsRoadTripRentals, SchemaType.Source, "CarDetails");

daCarDetails.Fill(dsRoadTripRentals, "CarDetails");

dgvCars.DataSource = dsRoadTripRentals.Tables["CarDetails"];

//Resize the DataGridView columns to fit the newly loaded content.

dgvCars.AutoResizeColumns(DataGridViewAutoSizeColumnsMode.AllCells);

//btnDelCar.Click += btnDelCar\_Click;

}

public delegate void OpenSubFormRequestHandler(Form subForm);

public event OpenSubFormRequestHandler OpenSubFormRequest;

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

{

if (dgvCars.SelectedRows.Count > 0)

{

string carReg = Convert.ToString(dgvCars.SelectedRows[0].Cells["CarReg"].Value);

// Ask for confirmation before deleting the car

DialogResult result = MessageBox.Show("Are you sure you want to delete the selected car?", "Confirm Delete", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (result == DialogResult.Yes)

{

// Find the car in the dataset using LINQ

DataRow carRow = dsRoadTripRentals.Tables["CarDetails"].AsEnumerable().SingleOrDefault(row => row.Field<string>("CarReg") == carReg);

if (carRow != null)

{

// Delete the car from the dataset and update the database

carRow.Delete();

daCarDetails.Update(dsRoadTripRentals, "CarDetails");

MessageBox.Show("Car Deleted");

}

else

{

MessageBox.Show("Car not found.");

}

}

}

else

{

MessageBox.Show("Please select a car to delete.");

}

}

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

{

if (dgvCars.SelectedRows.Count > 0)

{

string carReg = Convert.ToString(dgvCars.SelectedRows[0].Cells["CarReg"].Value);

frmEditCar newSubForm = new frmEditCar(carReg);

OpenSubFormInPanel(newSubForm);

}

else

{

MessageBox.Show("Please select a car to edit.");

}

}

public frmMainCar()

{

InitializeComponent();

}

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

{

frmAddCar addCarForm = new frmAddCar();

OpenSubFormInPanel(addCarForm);

}

private void OpenSubFormInPanel(Form subForm)

{

subForm.TopLevel = false;

subForm.FormBorderStyle = FormBorderStyle.None;

subForm.Dock = DockStyle.Fill;

// Replace 'panel1' with the name of the Panel control in your main form.

panelSub.Controls.Clear();

panelSub.Controls.Add(subForm);

subForm.Show();

}

}

}

## Add Car details

**A screenshot of a car details

Description automatically generated with medium confidence**

Form where a new Car can be added to the database. Cancel will not push through the changes and returns back to the main car menu. The car reg text box will remove any spaces which is used to avoid ABC1234 and ABC 1234 being technically different Car regs. A lot of Boxes are using combo boxes with pre populating data to avoid user input error. Model combo box grabs from Models, and the same with the rental cost

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 RoadTripRentals.Forms.Jordan

{

public partial class frmAddCar : Form

{

SqlDataAdapter daCarDetails, daCarDetailss;

DataSet dsRoadTripRentals = new DataSet();

SqlCommandBuilder cmdBCarDetails;

SqlCommand cmdCarDetailsDetails;

DataRow drCarDetails;

SqlConnection conn;

String sqlCarDetails, sqlCarDetailsDetails;

int selectedTab = 0;

bool custSelected = false;

int custNoSelected = 0;

private string connStr = @"Data Source = DESKTOP-ASEMACC\INTHEDOGHOUSE; Initial Catalog = RoadTripRentals; Integrated Security = true";

public frmAddCar()

{

InitializeComponent();

}

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

{

frmMainCar newSubForm = new frmMainCar();

OpenSubFormInPanel(newSubForm);

}

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

{

sqlCarDetails = @"select \* from CarDetails";

daCarDetails = new SqlDataAdapter(sqlCarDetails, connStr);

cmdBCarDetails = new SqlCommandBuilder(daCarDetails);

daCarDetails.FillSchema(dsRoadTripRentals, SchemaType.Source, "CarDetails");

daCarDetails.Fill(dsRoadTripRentals, "CarDetails");

cmbAddColour.Items.Add("Red");

cmbAddColour.Items.Add("Blue");

cmbAddColour.Items.Add("Green");

using (SqlConnection conn = new SqlConnection(connStr))

{

conn.Open();

using (SqlDataAdapter da = new SqlDataAdapter("SELECT ModelID, ModelDesc FROM Model", conn))

{

DataTable dt = new DataTable();

da.Fill(dt);

cmbAddModelID.DataSource = dt;

cmbAddModelID.DisplayMember = "ModelDesc";

cmbAddModelID.ValueMember = "ModelID";

}

using (SqlDataAdapter da = new SqlDataAdapter("SELECT RentalCostID, RentalCost FROM RentalCost", conn))

{

DataTable dt = new DataTable();

da.Fill(dt);

cmbAddRentalCostID.DataSource = dt;

cmbAddRentalCostID.DisplayMember = "RentalCost";

cmbAddRentalCostID.ValueMember = "RentalCostID";

}

}

}

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

{

MyCar myCar = new MyCar();

bool ok = true;

errP.Clear();

//CAR REG

try

{

myCar.CarReg = txtAddCarReg.Text.Trim().Replace(" ", "");

}

catch (MyException MyEx)

{

ok = false;

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

}

try

{

myCar.ModelID = cmbAddModelID.SelectedValue.ToString();

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(cmbAddModelID, MyEx.validate());

}

//COLOUR

try

{

myCar.Colour = cmbAddColour.Text.Trim();

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(cmbAddColour, MyEx.validate());

}

//MILEAGE

try

{

myCar.Mileage = (int)txtAddMileage.Value;

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtAddMileage, MyEx.validate());

}

//FUEL TYPE

try

{

myCar.FuelType = cmbAddFuelType.Text.Trim();

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(cmbAddFuelType, MyEx.validate());

}

//NO SEATS

try

{

myCar.NoSeats = (int)txtAddNoSeats.Value;

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtAddNoSeats, MyEx.validate());

}

//YEAR

try

{

myCar.Year = (int)txtAddYear.Value;

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtAddYear, MyEx.validate());

}

//RENTAL COST ID

try

{

myCar.RentalCostID = (int)cmbAddRentalCostID.SelectedValue;

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(cmbAddRentalCostID, MyEx.validate());

}

try

{

if (ok)

{

DataRow drCar = dsRoadTripRentals.Tables["CarDetails"].NewRow();

drCar["CarReg"] = myCar.CarReg;

drCar["ModelID"] = myCar.ModelID;

drCar["Colour"] = myCar.Colour;

drCar["Mileage"] = myCar.Mileage;

drCar["FuelType"] = myCar.FuelType;

drCar["NoSeats"] = myCar.NoSeats;

drCar["Year"] = myCar.Year;

drCar["RentalCostID"] = myCar.RentalCostID;

try

{

dsRoadTripRentals.Tables["CarDetails"].Rows.Add(drCar);

daCarDetails.Update(dsRoadTripRentals, "CarDetails");

// If no exceptions are thrown, show the "Car Added" message

MessageBox.Show("Car Added");

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

{

clearAddForm();

}

else

{

frmMainCar newSubForm = new frmMainCar();

OpenSubFormInPanel(newSubForm);

}

}

catch (ConstraintException)

{

MessageBox.Show("The car registration number '" + myCar.CarReg + "' already exists. Please enter a unique registration number.", "Duplicate Car Registration", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

catch (Exception ex)

{

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

}

}

}

catch (Exception ex)

{

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

}

}

void clearAddForm()

{

txtAddCarReg.Clear();

cmbAddModelID.SelectedIndex = -1;

cmbAddColour.SelectedIndex = -1;

txtAddMileage.Value = 1;

cmbAddFuelType.SelectedIndex = -1;

txtAddNoSeats.Value = 2;

txtAddYear.Value = 1950;

cmbAddRentalCostID.SelectedIndex = -1;

}

private void OpenSubFormInPanel(Form subForm)

{

subForm.TopLevel = false;

subForm.FormBorderStyle = FormBorderStyle.None;

subForm.Dock = DockStyle.Fill;

// Replace 'panel1' with the name of the Panel control in your main form.

panelSub.Controls.Clear();

panelSub.Controls.Add(subForm);

subForm.Show();

}

}

}

## Edit car details

**A screenshot of a car details

Description automatically generated with medium confidence**

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 RoadTripRentals.Forms.Jordan

{

public partial class frmEditCar : Form

{

SqlDataAdapter daCarDetails, daCarDetailss;

DataSet dsRoadTripRentals = new DataSet();

SqlCommandBuilder cmdBCarDetails;

SqlCommand cmdCarDetailsDetails;

DataRow drCarDetails;

SqlConnection conn;

String sqlCarDetails, sqlCarDetailsDetails;

int selectedTab = 0;

bool custSelected = false;

int custNoSelected = 0;

private string connStr = @"Data Source = DESKTOP-ASEMACC\INTHEDOGHOUSE; Initial Catalog = RoadTripRentals; Integrated Security = true";

private string carReg;

public frmEditCar(string carReg)

{

InitializeComponent();

this.carReg = carReg;

}

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

{

// TODO: This line of code loads data into the 'roadTripRentalsDataSet.Car' table. You can move, or remove it, as needed.

connStr = @"Data Source = DESKTOP-ASEMACC\INTHEDOGHOUSE; Initial Catalog = RoadTripRentals; Integrated Security = true";

sqlCarDetails = @"select \* from CarDetails";

daCarDetails = new SqlDataAdapter(sqlCarDetails, connStr);

cmdBCarDetails = new SqlCommandBuilder(daCarDetails);

daCarDetails.FillSchema(dsRoadTripRentals, SchemaType.Source, "CarDetails");

daCarDetails.Fill(dsRoadTripRentals, "CarDetails");

// Load data into ModelID ComboBox

using (SqlConnection conn = new SqlConnection(connStr))

{

conn.Open();

using (SqlDataAdapter da = new SqlDataAdapter("SELECT ModelID, ModelDesc FROM Model", conn))

{

DataTable dt = new DataTable();

da.Fill(dt);

cmbEditModelID.DataSource = dt;

cmbEditModelID.DisplayMember = "ModelDesc";

cmbEditModelID.ValueMember = "ModelID";

}

}

// Load data into RentalCostID ComboBox

using (SqlConnection conn = new SqlConnection(connStr))

{

conn.Open();

using (SqlDataAdapter da = new SqlDataAdapter("SELECT RentalCostID, RentalCost FROM RentalCost", conn))

{

DataTable dt = new DataTable();

da.Fill(dt);

cmbEditRentalCostID.DataSource = dt;

cmbEditRentalCostID.DisplayMember = "RentalCost";

cmbEditRentalCostID.ValueMember = "RentalCostID";

}

}

PopulateColourComboBox(cmbEditColour); // in frmEditCar\_Load

// Load the car details.

DataRow carRow = dsRoadTripRentals.Tables["CarDetails"].Rows.Find(carReg);

if (carRow != null)

{

lblEditCarReg.Text = carRow["CarReg"].ToString();

cmbEditModelID.SelectedValue = carRow["ModelID"];

cmbEditColour.SelectedItem = carRow["Colour"].ToString();

txtEditMileage.Value = Convert.ToDecimal(carRow["Mileage"]);

cmbEditFuelType.Text = carRow["FuelType"].ToString();

txtEditNoSeats.Value = Convert.ToDecimal(carRow["NoSeats"]);

txtEditYear.Value = Convert.ToDecimal(carRow["Year"]);

cmbEditRentalCostID.SelectedValue = carRow["RentalCostID"];

}

else

{

MessageBox.Show("Car not found.");

}

// Now you can set the SelectedValue

cmbEditModelID.SelectedValue = carRow["ModelID"];

cmbEditRentalCostID.SelectedValue = carRow["RentalCostID"];

}

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

{

frmMainCar newSubForm = new frmMainCar();

OpenSubFormInPanel(newSubForm);

}

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

{

MyCar myCar = new MyCar();

bool ok = true;

errP.Clear();

try

{

myCar.ModelID = cmbEditModelID.SelectedValue.ToString();

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(cmbEditModelID, MyEx.validate());

}

//COLOUR

try

{

myCar.Colour = cmbEditColour.Text.Trim();

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(cmbEditColour, MyEx.validate());

}

//MILEAGE

try

{

myCar.Mileage = (int)txtEditMileage.Value;

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtEditMileage, MyEx.validate());

}

//FUEL TYPE

try

{

myCar.FuelType = cmbEditFuelType.Text.Trim();

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(cmbEditFuelType, MyEx.validate());

}

//NO SEATS

try

{

myCar.NoSeats = (int)txtEditNoSeats.Value;

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtEditNoSeats, MyEx.validate());

}

//YEAR

try

{

myCar.Year = (int)txtEditYear.Value;

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtEditYear, MyEx.validate());

}

//RENTAL COST ID

try

{

myCar.RentalCostID = (int)cmbEditRentalCostID.SelectedValue;

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(cmbEditRentalCostID, MyEx.validate());

}

if (ok)

{

DataRow drCar = dsRoadTripRentals.Tables["CarDetails"].Rows.Find(myCar.CarReg);

if (drCar != null)

{

drCar["ModelID"] = myCar.ModelID;

drCar["Colour"] = myCar.Colour;

drCar["Mileage"] = myCar.Mileage;

drCar["FuelType"] = myCar.FuelType;

drCar["NoSeats"] = myCar.NoSeats;

drCar["Year"] = myCar.Year;

drCar["RentalCostID"] = myCar.RentalCostID;

daCarDetails.Update(dsRoadTripRentals, "CarDetails");

MessageBox.Show("Car Updated");

}

else

{

MessageBox.Show("Car not found.");

}

}

}

private void PopulateColourComboBox(ComboBox comboBox)

{

string[] colours = new string[]

{

"Red",

"Blue",

"Green",

"Yellow",

"Black",

"White",

"Gray",

"Silver",

"Purple",

"Orange",

"Brown",

"Pink"

// Add more colours as needed

};

foreach (string colour in colours)

{

comboBox.Items.Add(colour);

}

}

private void OpenSubFormInPanel(Form subForm)

{

subForm.TopLevel = false;

subForm.FormBorderStyle = FormBorderStyle.None;

subForm.Dock = DockStyle.Fill;

// Replace 'panel1' with the name of the Panel control in your main form.

panelSub.Controls.Clear();

panelSub.Controls.Add(subForm);

subForm.Show();

}

}

}

## Main Models

**A screenshot of a computer

Description automatically generated**

Models menu which displays the database containing each of the models. Can navigate to the add and edit forms from here which will replace the panel with the relevant form. Edit and Delete will only work if a row has been selected. The close sub form button also becomes visible, as an extra option to reset back to the main menu.

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 RoadTripRentals.Forms.Jordan

{

public partial class frmMainModel : Form

{

SqlDataAdapter daModel, daModels;

DataSet dsRoadTripRentals = new DataSet();

SqlCommandBuilder cmdBModel;

SqlCommand cmdModelDetails;

DataRow drModel;

SqlConnection conn;

String connStr, sqlModel, sqlModelDetails;

int selectedTab = 0;

bool custSelected = false;

int custNoSelected = 0;

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

{

// TODO: This line of code loads data into the 'roadTripRentalsDataSet.Model' table. You can move, or remove it, as needed.

connStr = @"Data Source = DESKTOP-ASEMACC\INTHEDOGHOUSE; Initial Catalog = RoadTripRentals; Integrated Security = true";

sqlModel = @"select \* from Model";

daModel = new SqlDataAdapter(sqlModel, connStr);

cmdBModel = new SqlCommandBuilder(daModel);

daModel.FillSchema(dsRoadTripRentals, SchemaType.Source, "Model");

daModel.Fill(dsRoadTripRentals, "Model");

dgvModels.DataSource = dsRoadTripRentals.Tables["Model"];

//Resize the DataGridView columns to fit the newly loaded content.

dgvModels.AutoResizeColumns(DataGridViewAutoSizeColumnsMode.AllCells);

//btnDelModel.Click += btnDelModel\_Click;

}

public delegate void OpenSubFormRequestHandler(Form subForm);

public event OpenSubFormRequestHandler OpenSubFormRequest;

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

{

if (dgvModels.SelectedRows.Count > 0)

{

string ModelID = Convert.ToString(dgvModels.SelectedRows[0].Cells["ModelID"].Value);

// Check if any car is associated with this model.

string sqlCheck = "SELECT COUNT(\*) FROM CarDetails WHERE ModelID = @ModelID";

// Open the SQL connection

conn = new SqlConnection(connStr);

conn.Open();

SqlCommand cmdCheck = new SqlCommand(sqlCheck, conn);

cmdCheck.Parameters.AddWithValue("@ModelID", ModelID);

int count = (int)cmdCheck.ExecuteScalar();

// Close the SQL connection

conn.Close();

if (count > 0)

{

MessageBox.Show("This model is assigned to one or more cars and cannot be deleted.", "Cannot Delete");

return;

}

// Ask for confirmation before deleting the Model

DialogResult result = MessageBox.Show("Are you sure you want to delete the selected Model?", "Confirm Delete", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (result == DialogResult.Yes)

{

// Find the Model in the dataset using LINQ

DataRow ModelRow = dsRoadTripRentals.Tables["Model"].AsEnumerable().SingleOrDefault(row => row.Field<string>("ModelID") == ModelID);

if (ModelRow != null)

{

// Delete the Model from the dataset and update the database

ModelRow.Delete();

daModel.Update(dsRoadTripRentals, "Model");

MessageBox.Show("Model Deleted");

}

else

{

MessageBox.Show("Model not found.");

}

}

}

else

{

MessageBox.Show("Please select a Model to delete.");

}

}

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

{

if (dgvModels.SelectedRows.Count > 0)

{

string ModelID = Convert.ToString(dgvModels.SelectedRows[0].Cells["ModelID"].Value);

frmEditModel newSubForm = new frmEditModel(ModelID);

OpenSubFormInPanel(newSubForm);

}

else

{

MessageBox.Show("Please select a Model to edit.");

}

}

public frmMainModel()

{

InitializeComponent();

}

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

{

frmAddModel addModelForm = new frmAddModel();

OpenSubFormInPanel(addModelForm);

}

private void OpenSubFormInPanel(Form subForm)

{

subForm.TopLevel = false;

subForm.FormBorderStyle = FormBorderStyle.None;

subForm.Dock = DockStyle.Fill;

// Replace 'panel1' with the name of the Panel control in your main form.

panelSub.Controls.Clear();

panelSub.Controls.Add(subForm);

subForm.Show();

}

}

}

## Add Models

**A screenshot of a computer

Description automatically generated**

Form where a new model for a car can be added to the database. Cancel will not push through the changes and returns back to the main car menu.

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 RoadTripRentals.Forms.Jordan

{

public partial class frmAddModel : Form

{

SqlDataAdapter daModel, daModels;

DataSet dsRoadTripRentals = new DataSet();

SqlCommandBuilder cmdBModel;

SqlCommand cmdModelDetails;

DataRow drModel;

SqlConnection conn;

String connStr, sqlModel, sqlModelDetails;

int selectedTab = 0;

bool custSelected = false;

int custNoSelected = 0;

public frmAddModel()

{

InitializeComponent();

}

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

{

// TODO: This line of code loads data into the 'roadTripRentalsDataSet.Model' table. You can move, or remove it, as needed.

connStr = @"Data Source = DESKTOP-ASEMACC\INTHEDOGHOUSE; Initial Catalog = RoadTripRentals; Integrated Security = true";

sqlModel = @"select \* from Model";

daModel = new SqlDataAdapter(sqlModel, connStr);

cmdBModel = new SqlCommandBuilder(daModel);

daModel.FillSchema(dsRoadTripRentals, SchemaType.Source, "Model");

daModel.Fill(dsRoadTripRentals, "Model");

//dgvModels.DataSource = dsRoadTripRentals.Tables["Model"];

//Resize the DataGridView columns to fit the newly loaded content.

// dgvModels.AutoResizeColumns(DataGridViewAutoSizeColumnsMode.AllCells);

//btnDelModel.Click += btnDelModel\_Click;

}

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

{

MyModel myModel = new MyModel();

bool ok = true;

errP.Clear();

// ModelID

try

{

myModel.ModelID = txtModel.Text.Trim();

}

catch (Exception ex)

{

ok = false;

errP.SetError(txtModel, ex.Message);

}

// Make

try

{

myModel.Make = txtMake.Text.Trim();

}

catch (Exception ex)

{

ok = false;

errP.SetError(txtMake, ex.Message);

}

// Description

try

{

myModel.Description = txtDesc.Text.Trim();

}

catch (Exception ex)

{

ok = false;

errP.SetError(txtDesc, ex.Message);

}

try

{

if (ok)

{

DataRow drModel = dsRoadTripRentals.Tables["Model"].NewRow();

drModel["ModelID"] = myModel.ModelID;

drModel["ModelDesc"] = myModel.Description;

drModel["Make"] = myModel.Make;

try

{

dsRoadTripRentals.Tables["Model"].Rows.Add(drModel);

daModel.Update(dsRoadTripRentals, "Model");

// If no exceptions are thrown, show the "Model Added" message

MessageBox.Show("Model Added");

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

{

clearAddForm();

}

else

{

frmMainModel newSubForm = new frmMainModel();

OpenSubFormInPanel(newSubForm);

}

}

catch (ConstraintException)

{

MessageBox.Show("The model ID '" + myModel.ModelID + "' already exists. Please enter a unique model ID.", "Duplicate Model ID", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

catch (Exception ex)

{

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

}

}

}

catch (Exception ex)

{

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

}

}

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

{

frmMainModel newSubForm = new frmMainModel();

OpenSubFormInPanel(newSubForm);

}

void clearAddForm()

{

txtModel.Clear();

txtDesc.Clear();

txtMake.Clear();

}

private void OpenSubFormInPanel(Form subForm)

{

subForm.TopLevel = false;

subForm.FormBorderStyle = FormBorderStyle.None;

subForm.Dock = DockStyle.Fill;

// Replace 'panel1' with the name of the Panel control in your main form.

panelSub.Controls.Clear();

panelSub.Controls.Add(subForm);

subForm.Show();

}

}

}

## Edit Model

**A screenshot of a computer

Description automatically generated**

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 RoadTripRentals.Forms.Jordan

{

public partial class frmEditModel : Form

{

SqlDataAdapter daModel, daModels;

DataSet dsRoadTripRentals = new DataSet();

SqlCommandBuilder cmdBModel;

SqlCommand cmdModelDetails;

DataRow drModel;

SqlConnection conn;

String connStr, sqlModel, sqlModelDetails;

int selectedTab = 0;

bool custSelected = false;

int custNoSelected = 0;

private string ModelID;

public frmEditModel(string ModelID)

{

InitializeComponent();

this.ModelID = ModelID;

}

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

{

// TODO: This line of code loads data into the 'roadTripRentalsDataSet.Model' table. You can move, or remove it, as needed.

connStr = @"Data Source = DESKTOP-ASEMACC\INTHEDOGHOUSE; Initial Catalog = RoadTripRentals; Integrated Security = true";

sqlModel = @"select \* from Model";

daModel = new SqlDataAdapter(sqlModel, connStr);

cmdBModel = new SqlCommandBuilder(daModel);

daModel.FillSchema(dsRoadTripRentals, SchemaType.Source, "Model");

daModel.Fill(dsRoadTripRentals, "Model");

DataRow drModel = dsRoadTripRentals.Tables["Model"].Rows.Find(ModelID);

if (drModel != null)

{

txtModel.Text = drModel["ModelID"].ToString();

txtMake.Text = drModel["Make"].ToString();

txtDesc.Text = drModel["ModelDesc"].ToString();

}

}

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

{

MyModel myModel = new MyModel();

bool ok = true;

errP.Clear();

// ModelID

try

{

myModel.ModelID = txtModel.Text.Trim();

}

catch (Exception ex)

{

ok = false;

errP.SetError(txtModel, ex.Message);

}

// Make

try

{

myModel.Make = txtMake.Text.Trim();

}

catch (Exception ex)

{

ok = false;

errP.SetError(txtMake, ex.Message);

}

// Description

try

{

myModel.Description = txtDesc.Text.Trim();

}

catch (Exception ex)

{

ok = false;

errP.SetError(txtDesc, ex.Message);

}

try

{

if (ok)

{

DataRow drModel = dsRoadTripRentals.Tables["Model"].Rows.Find(myModel.ModelID);

if (drModel != null)

{

drModel["ModelDesc"] = myModel.Description;

drModel["Make"] = myModel.Make;

try

{

daModel.Update(dsRoadTripRentals, "Model");

// If no exceptions are thrown, show the "Model Updated" message

MessageBox.Show("Model Updated");

frmMainModel newSubForm = new frmMainModel();

OpenSubFormInPanel(newSubForm);

}

catch (Exception ex)

{

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

}

}

else

{

MessageBox.Show("Model not found", "Error!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

catch (Exception ex)

{

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

}

}

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

{

frmMainModel newSubForm = new frmMainModel();

OpenSubFormInPanel(newSubForm);

}

void clearAddForm()

{

txtModel.Clear();

txtDesc.Clear();

txtMake.Clear();

}

private void OpenSubFormInPanel(Form subForm)

{

subForm.TopLevel = false;

subForm.FormBorderStyle = FormBorderStyle.None;

subForm.Dock = DockStyle.Fill;

// Replace 'panel1' with the name of the Panel control in your main form.

panelSub.Controls.Clear();

panelSub.Controls.Add(subForm);

subForm.Show();

}

}

}

## Main Rental Cost

**A screenshot of a computer

Description automatically generated with medium confidence**

Rental cost menu which displays the database containing each of the rental costs . Can navigate to the add and edit forms from here which will replace the panel with the relevant form. Edit and Delete will only work if a row has been selected. The close sub form button also becomes visible, as an extra option to reset back to the main menu.

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 RoadTripRentals.Forms.Jordan

{

public partial class frmMainRentalCost : Form

{

SqlDataAdapter daRentalCost, daRentalCosts;

DataSet dsRoadTripRentals = new DataSet();

SqlCommandBuilder cmdBRentalCost;

SqlCommand cmdRentalCostDetails;

DataRow drRentalCost;

SqlConnection conn;

String connStr, sqlRentalCost, sqlRentalCostDetails;

int selectedTab = 0;

bool custSelected = false;

int custNoSelected = 0;

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

{

// TODO: This line of code loads data into the 'roadTripRentalsDataSet.RentalCost' table. You can move, or remove it, as needed.

connStr = @"Data Source = DESKTOP-ASEMACC\INTHEDOGHOUSE; Initial Catalog = RoadTripRentals; Integrated Security = true";

sqlRentalCost = @"select \* from RentalCost";

daRentalCost = new SqlDataAdapter(sqlRentalCost, connStr);

cmdBRentalCost = new SqlCommandBuilder(daRentalCost);

daRentalCost.FillSchema(dsRoadTripRentals, SchemaType.Source, "RentalCost");

daRentalCost.Fill(dsRoadTripRentals, "RentalCost");

dgvRentalCosts.DataSource = dsRoadTripRentals.Tables["RentalCost"];

//Resize the DataGridView columns to fit the newly loaded content.

dgvRentalCosts.AutoResizeColumns(DataGridViewAutoSizeColumnsMode.AllCells);

//btnDelRentalCost.Click += btnDelRentalCost\_Click;

}

public delegate void OpenSubFormRequestHandler(Form subForm);

public event OpenSubFormRequestHandler OpenSubFormRequest;

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

{

if (dgvRentalCosts.SelectedRows.Count > 0)

{

string RentalCostID = Convert.ToString(dgvRentalCosts.SelectedRows[0].Cells["RentalCostID"].Value);

// Check if any car is associated with this RentalCost.

string sqlCheck = "SELECT COUNT(\*) FROM CarDetails WHERE RentalCostID = @RentalCostID";

// Open the SQL connection

conn = new SqlConnection(connStr);

conn.Open();

SqlCommand cmdCheck = new SqlCommand(sqlCheck, conn);

cmdCheck.Parameters.AddWithValue("@RentalCostID", RentalCostID);

int count = (int)cmdCheck.ExecuteScalar();

// Close the SQL connection

conn.Close();

if (count > 0)

{

MessageBox.Show("This Rental Cost is assigned to one or more cars and cannot be deleted. Please reassign or this remove this Rental Cost from the Car(s) assocaitd with it.", "Cannot Delete");

return;

}

// Ask for confirmation before deleting the RentalCost

DialogResult result = MessageBox.Show("Are you sure you want to delete the selected Rental Cost?", "Confirm Delete", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (result == DialogResult.Yes)

{

int rentalCostIdInt = Convert.ToInt32(RentalCostID);

DataRow RentalCostRow = dsRoadTripRentals.Tables["RentalCost"].AsEnumerable().SingleOrDefault(row => row.Field<int>("RentalCostID") == rentalCostIdInt);

if (RentalCostRow != null)

{

// Delete the RentalCost from the dataset and update the database

RentalCostRow.Delete();

daRentalCost.Update(dsRoadTripRentals, "RentalCost");

MessageBox.Show("Rental Cost Deleted");

}

else

{

MessageBox.Show("Rental Cost not found.");

}

}

}

else

{

MessageBox.Show("Please select a RentalCost to delete.");

}

}

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

{

if (dgvRentalCosts.SelectedRows.Count > 0)

{

int RentalCostID = Convert.ToInt32(dgvRentalCosts.SelectedRows[0].Cells["RentalCostID"].Value);

frmEditRentalCost newSubForm = new frmEditRentalCost(RentalCostID);

OpenSubFormInPanel(newSubForm);

}

else

{

MessageBox.Show("Please select a RentalCost to edit.");

}

}

public frmMainRentalCost()

{

InitializeComponent();

//frmSettings.ApplyColorScheme(this);

}

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

{

frmAddRentalCost addRentalCostForm = new frmAddRentalCost();

OpenSubFormInPanel(addRentalCostForm);

}

private void OpenSubFormInPanel(Form subForm)

{

subForm.TopLevel = false;

subForm.FormBorderStyle = FormBorderStyle.None;

subForm.Dock = DockStyle.Fill;

// Replace 'panel1' with the name of the Panel control in your main form.

panelSub.Controls.Clear();

panelSub.Controls.Add(subForm);

subForm.Show();

}

}

}

## Add Rental Cost

**A screenshot of a computer

Description automatically generated with medium confidence**

Form where a new cost can be added to the database which is then used to assigned a cost to car. Cancel will not push through the changes and returns back to the main car menu. The rental cost box is a numericupdown box which increments in 0.5, this is also marked as read only so the custonmer cannot input anything they want, or even have the field blank.

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 RoadTripRentals.Forms.Jordan

{

public partial class frmAddRentalCost : Form

{

SqlDataAdapter daRentalCost, daRentalCosts;

DataSet dsRoadTripRentals = new DataSet();

SqlCommandBuilder cmdBRentalCost;

SqlCommand cmdRentalCostDetails;

DataRow drRentalCost;

SqlConnection conn;

String connStr, sqlRentalCost, sqlRentalCostDetails;

int selectedTab = 0;

bool custSelected = false;

int custNoSelected = 0;

public frmAddRentalCost()

{

InitializeComponent();

}

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

{

// TODO: This line of code loads data into the 'roadTripRentalsDataSet.RentalCost' table. You can move, or remove it, as needed.

connStr = @"Data Source = DESKTOP-ASEMACC\INTHEDOGHOUSE; Initial Catalog = RoadTripRentals; Integrated Security = true";

sqlRentalCost = @"select \* from RentalCost";

daRentalCost = new SqlDataAdapter(sqlRentalCost, connStr);

cmdBRentalCost = new SqlCommandBuilder(daRentalCost);

daRentalCost.FillSchema(dsRoadTripRentals, SchemaType.Source, "RentalCost");

daRentalCost.Fill(dsRoadTripRentals, "RentalCost");

int noRows = dsRoadTripRentals.Tables["RentalCost"].Rows.Count;

if (noRows == 0)

txtRentalCostID.Text = "10000";

else

{

getNumber();

}

}

public class MyRentalCost

{

public int RentalCostID { get; set; }

public decimal RentalCost { get; set; }

}

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

{

MyRentalCost myRentalCost = new MyRentalCost();

bool ok = true;

errP.Clear();

// RentalCostID

try

{

myRentalCost.RentalCostID = Convert.ToInt32(txtRentalCostID.Text.Trim());

// drRentalCost["RentalCostID"] = myRentalCost.RentalCostID; // Remove this line

}

catch (Exception ex)

{

ok = false;

errP.SetError(txtRentalCostID, ex.Message);

}

// RentalCost

try

{

myRentalCost.RentalCost = txtRentalCost.Value; // NumericUpDown.Value returns a decimal

}

catch (Exception ex)

{

ok = false;

errP.SetError(txtRentalCost, ex.Message);

}

if (ok)

{

DataRow drRentalCost = dsRoadTripRentals.Tables["RentalCost"].NewRow();

drRentalCost["RentalCostID"] = myRentalCost.RentalCostID;

drRentalCost["RentalCost"] = myRentalCost.RentalCost;

// RentalCostID is auto-incremented in the database, so no need to set it here

try

{

dsRoadTripRentals.Tables["RentalCost"].Rows.Add(drRentalCost);

daRentalCost.Update(dsRoadTripRentals, "RentalCost");

// If no exceptions are thrown, show the "Rental Cost Added" message

MessageBox.Show("Rental Cost Added");

if (MessageBox.Show("Do you wish to add another rental cost?", "Add Rental Cost", MessageBoxButtons.YesNo) == System.Windows.Forms.DialogResult.Yes)

{

clearAddForm();

getNumber();

}

else

{

frmMainRentalCost newSubForm = new frmMainRentalCost();

OpenSubFormInPanel(newSubForm);

}

}

catch (Exception ex)

{

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

}

}

}

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

{

frmMainRentalCost newSubForm = new frmMainRentalCost();

OpenSubFormInPanel(newSubForm);

}

void clearAddForm()

{

txtRentalCost.Value = txtRentalCost.Minimum;

}

private void getNumber()

{

try

{

using (SqlConnection conn = new SqlConnection(connStr))

{

conn.Open();

SqlCommand cmd = new SqlCommand("SELECT MAX(RentalCostID) FROM RentalCost", conn);

int maxRentalCostId = (int)cmd.ExecuteScalar();

txtRentalCostID.Text = (maxRentalCostId + 1).ToString();

conn.Close();

}

}

catch (Exception ex)

{

MessageBox.Show("Error fetching the next RentalCostID: " + ex.Message);

}

}

private void OpenSubFormInPanel(Form subForm)

{

subForm.TopLevel = false;

subForm.FormBorderStyle = FormBorderStyle.None;

subForm.Dock = DockStyle.Fill;

// Replace 'panel1' with the name of the Panel control in your main form.

panelSub.Controls.Clear();

panelSub.Controls.Add(subForm);

subForm.Show();

}

}

}

## Edit Rental Cost

**A screenshot of a computer

Description automatically generated with medium confidence**

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 RoadTripRentals.Forms.Jordan

{

public partial class frmEditRentalCost : Form

{

SqlDataAdapter daRentalCost, daRentalCosts;

DataSet dsRoadTripRentals = new DataSet();

SqlCommandBuilder cmdBRentalCost;

SqlCommand cmdRentalCostDetails;

DataRow drRentalCost;

SqlConnection conn;

String connStr, sqlRentalCost, sqlRentalCostDetails;

int selectedTab = 0;

bool custSelected = false;

int custNoSelected = 0;

public int RentalCostIdToEdit { get; set; }

public frmEditRentalCost(int rentalCostID)

{

InitializeComponent();

RentalCostIdToEdit = rentalCostID;

}

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

{

// TODO: This line of code loads data into the 'roadTripRentalsDataSet.RentalCost' table. You can move, or remove it, as needed.

connStr = @"Data Source = DESKTOP-ASEMACC\INTHEDOGHOUSE; Initial Catalog = RoadTripRentals; Integrated Security = true";

sqlRentalCost = @"select \* from RentalCost";

daRentalCost = new SqlDataAdapter(sqlRentalCost, connStr);

cmdBRentalCost = new SqlCommandBuilder(daRentalCost);

daRentalCost.FillSchema(dsRoadTripRentals, SchemaType.Source, "RentalCost");

daRentalCost.Fill(dsRoadTripRentals, "RentalCost");

LoadRentalCost(RentalCostIdToEdit);

}

public class MyRentalCost

{

public int RentalCostID { get; set; }

public decimal RentalCost { get; set; }

}

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

{

MyRentalCost myRentalCost = new MyRentalCost();

bool ok = true;

errP.Clear();

// RentalCost

try

{

myRentalCost.RentalCost = txtRentalCost.Value; // NumericUpDown.Value returns a decimal

}

catch (Exception ex)

{

ok = false;

errP.SetError(txtRentalCost, ex.Message);

}

if (ok)

{

DataRow drRentalCost = dsRoadTripRentals.Tables["RentalCost"].Rows.Find(RentalCostIdToEdit);

drRentalCost["RentalCost"] = myRentalCost.RentalCost;

try

{

daRentalCost.Update(dsRoadTripRentals, "RentalCost");

// If no exceptions are thrown, show the "Rental Cost Updated" message

MessageBox.Show("Rental Cost Updated");

// Go directly to the MainRentalCost form after editing

frmMainRentalCost newSubForm = new frmMainRentalCost();

OpenSubFormInPanel(newSubForm);

}

catch (Exception ex)

{

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

}

}

}

private void LoadRentalCost(int rentalCostId)

{

DataRow drRentalCost = dsRoadTripRentals.Tables["RentalCost"].Rows.Find(rentalCostId);

if (drRentalCost != null)

{

txtRentalCostID.Text = drRentalCost["RentalCostID"].ToString();

txtRentalCost.Value = Convert.ToDecimal(drRentalCost["RentalCost"]);

}

}

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

{

frmMainRentalCost newSubForm = new frmMainRentalCost();

OpenSubFormInPanel(newSubForm);

}

void clearEditForm()

{

txtRentalCost.Value = txtRentalCost.Minimum;

}

private void OpenSubFormInPanel(Form subForm)

{

subForm.TopLevel = false;

subForm.FormBorderStyle = FormBorderStyle.None;

subForm.Dock = DockStyle.Fill;

// Replace 'panel1' with the name of the Panel control in your main form.

panelSub.Controls.Clear();

panelSub.Controls.Add(subForm);

subForm.Show();

}

}

}

## Main Payments

**A screenshot of a computer

Description automatically generated with medium confidence**

Payments menu which displays the database containing each of the payments. Can navigate to the add and edit forms from here which will replace the panel with the relevant form. Edit and Delete will only work if a row has been selected. The close sub form button also becomes visible, as an extra option to reset back to the main menu.

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 RoadTripRentals.Forms.Jordan

{

public partial class frmMainPayments : Form

{

SqlDataAdapter daPayments, daPaymentss;

DataSet dsRoadTripRentals = new DataSet();

SqlCommandBuilder cmdBPayments;

SqlCommand cmdPaymentsDetails;

DataRow drPayments;

SqlConnection conn;

String connStr, sqlPayments, sqlPaymentsDetails;

int selectedTab = 0;

bool custSelected = false;

int custNoSelected = 0;

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

{

// TODO: This line of code loads data into the 'roadTripRentalsDataSet.PaymentType' table. You can move, or remove it, as needed.

connStr = @"Data Source = DESKTOP-ASEMACC\INTHEDOGHOUSE; Initial Catalog = RoadTripRentals; Integrated Security = true";

sqlPayments = @"select \* from PaymentType";

daPayments = new SqlDataAdapter(sqlPayments, connStr);

cmdBPayments = new SqlCommandBuilder(daPayments);

daPayments.FillSchema(dsRoadTripRentals, SchemaType.Source, "PaymentType");

daPayments.Fill(dsRoadTripRentals, "PaymentType");

dgvPayments.DataSource = dsRoadTripRentals.Tables["PaymentType"];

//Resize the DataGridView columns to fit the newly loaded content.

dgvPayments.AutoResizeColumns(DataGridViewAutoSizeColumnsMode.AllCells);

//btnDelPayments.Click += btnDelPayments\_Click;

}

public delegate void OpenSubFormRequestHandler(Form subForm);

public event OpenSubFormRequestHandler OpenSubFormRequest;

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

{

if (dgvPayments.SelectedRows.Count > 0)

{

int PaymentID = Convert.ToInt32(dgvPayments.SelectedRows[0].Cells["PaymentID"].Value);

// Ask for confirmation before deleting the Payment

DialogResult result = MessageBox.Show("Are you sure you want to delete the selected Payment type?", "Confirm Delete", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (result == DialogResult.Yes)

{

// Find the Payment in the dataset using LINQ

DataRow PaymentRow = dsRoadTripRentals.Tables["PaymentType"].AsEnumerable().SingleOrDefault(row => row.Field<int>("PaymentID") == PaymentID);

if (PaymentRow != null)

{

// Delete the Payment from the dataset and update the database

PaymentRow.Delete();

daPayments.Update(dsRoadTripRentals, "PaymentType");

MessageBox.Show("Payment type Deleted");

}

else

{

MessageBox.Show("Payment type not found.");

}

}

}

else

{

MessageBox.Show("Please select a Payment type to delete.");

}

}

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

{

if (dgvPayments.SelectedRows.Count > 0)

{

int PaymentID = Convert.ToInt32(dgvPayments.SelectedRows[0].Cells["PaymentID"].Value);

// Assuming dsRoadTripRentals is accessible from this method

DataRow[] rows = dsRoadTripRentals.Tables["PaymentType"].Select($"PaymentID = {PaymentID}");

if (rows.Length > 0)

{

frmEditPayments newSubForm = new frmEditPayments(rows[0]);

OpenSubFormInPanel(newSubForm);

}

else

{

MessageBox.Show("No corresponding data found.");

}

}

else

{

MessageBox.Show("Please select a Payment type to edit.");

}

}

public frmMainPayments()

{

InitializeComponent();

}

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

{

frmAddPayments addPaymentsForm = new frmAddPayments();

OpenSubFormInPanel(addPaymentsForm);

}

private void OpenSubFormInPanel(Form subForm)

{

subForm.TopLevel = false;

subForm.FormBorderStyle = FormBorderStyle.None;

subForm.Dock = DockStyle.Fill;

// Replace 'panel1' with the name of the Panel control in your main form.

panelSub.Controls.Clear();

panelSub.Controls.Add(subForm);

subForm.Show();

}

}

}

## Add Payments

**A screenshot of a computer

Description automatically generated with medium confidence**

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 RoadTripRentals.Forms.Jordan

{

public partial class frmAddPayments : Form

{

SqlDataAdapter daPayments, daRental, daPaymentss;

DataSet dsRoadTripRentals = new DataSet();

SqlCommandBuilder cmdBPayments, cmdBRental;

SqlCommand cmdPaymentsDetails ;

DataRow drPayments;

SqlConnection conn;

String connStr, sqlPayments, sqlRental, sqlPaymentsDetails;

int selectedTab = 0;

bool custSelected = false;

int custNoSelected = 0;

public frmAddPayments()

{

InitializeComponent();

}

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

{

// TODO: This line of code loads data into the 'roadTripRentalsDataSet.Payments' table. You can move, or remove it, as needed.

connStr = @"Data Source = DESKTOP-ASEMACC\INTHEDOGHOUSE; Initial Catalog = RoadTripRentals; Integrated Security = true";

sqlPayments = @"select \* from PaymentType";

daPayments = new SqlDataAdapter(sqlPayments, connStr);

cmdBPayments = new SqlCommandBuilder(daPayments);

daPayments.FillSchema(dsRoadTripRentals, SchemaType.Source, "PaymentType");

daPayments.Fill(dsRoadTripRentals, "PaymentType");

int noRows = dsRoadTripRentals.Tables["PaymentType"].Rows.Count;

// Rental table

sqlRental = @"select \* from Rental";

daRental = new SqlDataAdapter(sqlRental, connStr);

cmdBRental = new SqlCommandBuilder(daRental);

daRental.FillSchema(dsRoadTripRentals, SchemaType.Source, "Rental");

daRental.Fill(dsRoadTripRentals, "Rental");

if (noRows == 0)

txtPaymentID.Text = "10000";

else

{

getNumber();

}

}

public class MyPayments

{

public int PaymentID { get; set; }

public string PaymentType { get; set; }

public DateTime PaymentDate { get; set; }

}

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

{

MyPayments myPayments = new MyPayments();

bool ok = true;

errP.Clear();

// PaymentID

try

{

myPayments.PaymentID = Convert.ToInt32(txtPaymentID.Text.Trim());

// drPayments["PaymentID"] = myPayments.PaymentID; // Remove this line

}

catch (Exception ex)

{

ok = false;

errP.SetError(txtPaymentID, ex.Message);

}

// PaymentType

try

{

myPayments.PaymentType = cmbPaymentType.Text.Trim(); // NumericUpDown.Value returns a decimal

}

catch (Exception ex)

{

ok = false;

errP.SetError(cmbPaymentType, ex.Message);

}

if (ok)

{

DataRow drPayments = dsRoadTripRentals.Tables["PaymentType"].NewRow();

drPayments["PaymentID"] = myPayments.PaymentID;

drPayments["PaymentType"] = myPayments.PaymentType;

// PaymentID is auto-incremented in the database, so no need to set it here

try

{

dsRoadTripRentals.Tables["PaymentType"].Rows.Add(drPayments);

daPayments.Update(dsRoadTripRentals, "PaymentType");

// If no exceptions are thrown, show the "Rental Cost Added" message

MessageBox.Show("Rental Cost Added");

if (MessageBox.Show("Do you wish to add another payment?", "Add Rental Cost", MessageBoxButtons.YesNo) == System.Windows.Forms.DialogResult.Yes)

{

clearAddForm();

getNumber();

}

else

{

frmMainPayments newSubForm = new frmMainPayments();

OpenSubFormInPanel(newSubForm);

}

}

catch (Exception ex)

{

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

}

}

}

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

{

frmMainPayments newSubForm = new frmMainPayments();

OpenSubFormInPanel(newSubForm);

}

void clearAddForm()

{

cmbPaymentType.SelectedIndex = -1;

}

private void getNumber()

{

try

{

using (SqlConnection conn = new SqlConnection(connStr))

{

conn.Open();

SqlCommand cmd = new SqlCommand("SELECT MAX(PaymentID) FROM PaymentType", conn);

int maxPaymentID = (int)cmd.ExecuteScalar();

txtPaymentID.Text = (maxPaymentID + 1).ToString();

conn.Close();

}

}

catch (Exception ex)

{

MessageBox.Show("Error fetching the next PaymentID: " + ex.Message);

}

}

private void OpenSubFormInPanel(Form subForm)

{

subForm.TopLevel = false;

subForm.FormBorderStyle = FormBorderStyle.None;

subForm.Dock = DockStyle.Fill;

// Replace 'panel1' with the name of the Panel control in your main form.

panelSub.Controls.Clear();

panelSub.Controls.Add(subForm);

subForm.Show();

}

}

}

## Edit Payments

**A screenshot of a computer

Description automatically generated with medium confidence**

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 RoadTripRentals.Forms.Jordan

{

public partial class frmEditPayments : Form

{

SqlDataAdapter daPayments, daPaymentss;

DataSet dsRoadTripRentals = new DataSet();

SqlCommandBuilder cmdBPayments;

SqlCommand cmdPaymentsDetails;

DataRow drPayments;

SqlConnection conn;

String connStr, sqlPayments, sqlPaymentsDetails;

int selectedTab = 0;

bool custSelected = false;

int custNoSelected = 0;

DataRow selectedRow;

public frmEditPayments(DataRow row)

{

InitializeComponent();

selectedRow = row;

}

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

{

// TODO: This line of code loads data into the 'roadTripRentalsDataSet.Payments' table. You can move, or remove it, as needed.

connStr = @"Data Source = DESKTOP-ASEMACC\INTHEDOGHOUSE; Initial Catalog = RoadTripRentals; Integrated Security = true";

sqlPayments = @"select \* from PaymentType";

daPayments = new SqlDataAdapter(sqlPayments, connStr);

cmdBPayments = new SqlCommandBuilder(daPayments);

daPayments.FillSchema(dsRoadTripRentals, SchemaType.Source, "PaymentType");

daPayments.Fill(dsRoadTripRentals, "PaymentType");

int noRows = dsRoadTripRentals.Tables["PaymentType"].Rows.Count;

if (noRows == 0)

txtPaymentID.Text = "10000";

else

{

getNumber();

}

if (selectedRow != null)

{

txtPaymentID.Text = selectedRow["PaymentID"].ToString();

cmbPaymentType.Text = selectedRow["PaymentType"].ToString();

}

}

public class MyPayments

{

public int PaymentID { get; set; }

public string PaymentType { get; set; }

public DateTime PaymentDate { get; set; }

}

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

{

MyPayments myPayments = new MyPayments();

bool ok = true;

errP.Clear();

// PaymentID

try

{

myPayments.PaymentID = Convert.ToInt32(txtPaymentID.Text.Trim());

}

catch (Exception ex)

{

ok = false;

errP.SetError(txtPaymentID, ex.Message);

}

// PaymentType

try

{

myPayments.PaymentType = cmbPaymentType.Text.Trim();

}

catch (Exception ex)

{

ok = false;

errP.SetError(cmbPaymentType, ex.Message);

}

if (ok)

{

// First, find the row in the datatable

DataRow[] foundRows = dsRoadTripRentals.Tables["PaymentType"].Select("PaymentID = " + myPayments.PaymentID);

// If the row was found, update it

if (foundRows.Length > 0)

{

foundRows[0]["PaymentType"] = myPayments.PaymentType;

try

{

daPayments.Update(dsRoadTripRentals, "PaymentType");

// If no exceptions are thrown, show the "Payment Updated" message

MessageBox.Show("Payment Updated");

frmMainPayments newSubForm = new frmMainPayments();

OpenSubFormInPanel(newSubForm);

}

catch (Exception ex)

{

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

}

}

}

}

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

{

frmMainPayments newSubForm = new frmMainPayments();

OpenSubFormInPanel(newSubForm);

}

void clearAddForm()

{

cmbPaymentType.SelectedIndex = -1;

}

private void getNumber()

{

try

{

using (SqlConnection conn = new SqlConnection(connStr))

{

conn.Open();

SqlCommand cmd = new SqlCommand("SELECT MAX(PaymentID) FROM PaymentType", conn);

int maxPaymentID = (int)cmd.ExecuteScalar();

txtPaymentID.Text = (maxPaymentID + 1).ToString();

conn.Close();

}

}

catch (Exception ex)

{

MessageBox.Show("Error fetching the next PaymentID: " + ex.Message);

}

}

private void OpenSubFormInPanel(Form subForm)

{

subForm.TopLevel = false;

subForm.FormBorderStyle = FormBorderStyle.None;

subForm.Dock = DockStyle.Fill;

// Replace 'panel1' with the name of the Panel control in your main form.

panelSub.Controls.Clear();

panelSub.Controls.Add(subForm);

subForm.Show();

}

}

}

# **David Forms**

## Suppliers Main

A screenshot of a computer

Description automatically generated

Click on the Suppliers button on the side panel will display 2 options – Details and Add. when clicking on the Details option the above screen will appear on the main panel, Supplier Details button will display the list of suppliers in a data grid view, the edit button will display the details of the current highlighted supplier in the data grid view in a new form to amend and save. The Delete button will display a data grid view and allow the user to select a supplier and click delete to remove from the database. The quit will close the form and return the user to the main menu, the ‘X’ in the top right will also do the same.

public partial class frmSupplierMain : Form

{

int noMenuItems = 0;

Label[] menuItems;

public delegate void OpenSubFormRequestHandler(Form subForm);

public event OpenSubFormRequestHandler OpenSubFormRequest;

public frmSupplierMain()

{

InitializeComponent();

}

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

{

frmDisplaySupplier frm1 = new frmDisplaySupplier();

frm1.TopLevel = false;

frm1.FormBorderStyle = FormBorderStyle.None;

frm1.WindowState = FormWindowState.Maximized;

pnlMain.Controls.Add(frm1);

frm1.Show();

}

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

{

int startIndex = 0;

Label lbl = (Label)sender;

MyGlobals.frmClosing = false;

MyGlobals.frmEditForm = false;

startIndex = Convert.ToInt32(lbl.Tag.ToString().Substring(1, 1));

switch (startIndex)

{

case 1:

frmDisplaySupplier frm1 = new frmDisplaySupplier();

frm1.TopLevel = false;

frm1.FormBorderStyle = FormBorderStyle.None;

frm1.WindowState = FormWindowState.Maximized;

pnlMain.Controls.Add(frm1);

frm1.Show();

break;

case 2:

frmAddSupplier frm2 = new frmAddSupplier();

frm2.TopLevel = false;

frm2.FormBorderStyle = FormBorderStyle.None;

frm2.WindowState = FormWindowState.Maximized;

pnlMain.Controls.Add(frm2);

frm2.Show();

break;

case 3:

break;

case 4:

frmDeleteSupplier frm4 = new frmDeleteSupplier();

frm4.TopLevel = false;

frm4.FormBorderStyle = FormBorderStyle.None;

frm4.WindowState = FormWindowState.Maximized;

pnlMain.Controls.Add(frm4);

frm4.Show();

break;

case 5:

this.Close();

break;

}

}

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

{

Close();

}

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

{

if (MyGlobals.selectedSupplierNo != 0)

{

frmEditSupplier frm3 = new frmEditSupplier();

frm3.TopLevel = false;

frm3.FormBorderStyle = FormBorderStyle.None;

frm3.WindowState = FormWindowState.Maximized;

pnlMain.Controls.Add(frm3);

frm3.Show();

}

else

MessageBox.Show("No Supplier selected for edit!", "Select a Supplier");

}

}

## Edit Supplier

A screenshot of a computer

Description automatically generated with medium confidence

Edit supplier allows user to update existing supplier details, text boxes were used for each field with validation behind each as shown in the code below.

public partial class frmEditSupplier : Form

{

SqlDataAdapter daSupplier;

DataSet dsRoadTripRentals = new DataSet();

SqlCommandBuilder cmdBSupplier;

DataRow drSupplier;

String connStr, sqlSupplier;

public frmEditSupplier()

{

InitializeComponent();

}

private void frmEditSupplier\_FormClosing(object sender, FormClosingEventArgs e)

{

MyGlobals.frmClosing = true;

}

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

{

if (btnEditSupplier.Text == "Edit")

{

txtSupplierName.Enabled = true;

txtStreet.Enabled = true;

txtTown.Enabled = true;

txtCounty.Enabled = true;

txtPostcode.Enabled = true;

txtTelNo.Enabled = true;

txtEmail.Enabled = true;

btnEditSupplier.Text = "Save";

}

else

{

MySupplier mySupplier = new MySupplier();

bool ok = true;

errP.Clear();

try

{

mySupplier.SupplierNo = Convert.ToInt32(lblSupplierNoValue.Text.Trim());

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(lblSupplierNoValue, MyEx.validate());

}

try

{

mySupplier.SupplierName = txtSupplierName.Text.Trim();

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtSupplierName, MyEx.validate());

}

try

{

mySupplier.Street = txtStreet.Text.Trim();

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtStreet, MyEx.validate());

}

try

{

mySupplier.Town = txtTown.Text.Trim();

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtTown, MyEx.validate());

}

try

{

mySupplier.County = txtCounty.Text.Trim();

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtCounty, MyEx.validate());

}

try

{

mySupplier.Postcode = txtPostcode.Text.Trim();

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtPostcode, MyEx.validate());

}

try

{

mySupplier.SupplierTelNo = txtTelNo.Text.Trim();

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtTelNo, MyEx.validate());

}

try

{

mySupplier.SupplierEmail = txtEmail.Text.Trim();

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtEmail, MyEx.validate());

}

try

{

if (ok)

{

drSupplier.BeginEdit();

drSupplier["SupplierNo"] = mySupplier.SupplierNo;

drSupplier["SupplierName"] = mySupplier.SupplierName;

drSupplier["SupplierStreet"] = mySupplier.Street;

drSupplier["SupplierTown"] = mySupplier.Town;

drSupplier["SupplierCounty"] = mySupplier.County;

drSupplier["SupplierPostCode"] = mySupplier.Postcode;

drSupplier["SupplierTelNo"] = mySupplier.SupplierTelNo;

drSupplier["SupplierEmail"] = mySupplier.SupplierEmail;

drSupplier.EndEdit();

daSupplier.Update(dsRoadTripRentals, "Supplier");

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

txtSupplierName.Enabled = false;

txtStreet.Enabled = false;

txtTown.Enabled = false;

txtCounty.Enabled = false;

txtPostcode.Enabled = false;

txtTelNo.Enabled = false;

txtEmail.Enabled = false;

btnEditSupplier.Text = "Edit";

}

}

catch (Exception ex)

{

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

}

}

}

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

{

if (MessageBox.Show("Cancel the edit of Supplier No: " + lblSupplierNoValue.Text + "?", "Edit Supplier", MessageBoxButtons.YesNo) == System.Windows.Forms.DialogResult.Yes)

Close();

}

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

{

connStr = @"Data Source = .\sqlExpress; Initial Catalog = RoadTripRentals; Integrated Security = true";

sqlSupplier = @"select \* from Supplier";

daSupplier = new SqlDataAdapter(sqlSupplier, connStr);

cmdBSupplier = new SqlCommandBuilder(daSupplier);

daSupplier.FillSchema(dsRoadTripRentals, SchemaType.Source, "Supplier");

daSupplier.Fill(dsRoadTripRentals, "Supplier");

lblSupplierNoValue.Text = MyGlobals.selectedSupplierNo.ToString();

drSupplier = dsRoadTripRentals.Tables["Supplier"].Rows.Find(lblSupplierNoValue.Text);

txtSupplierName.Text = drSupplier["SupplierName"].ToString();

txtStreet.Text = drSupplier["SupplierStreet"].ToString();

txtTown.Text = drSupplier["SupplierTown"].ToString();

txtCounty.Text = drSupplier["SupplierCounty"].ToString();

txtPostcode.Text = drSupplier["SupplierPostCode"].ToString();

txtTelNo.Text = drSupplier["SupplierTelNo"].ToString();

txtEmail.Text = drSupplier["SupplierEmail"].ToString();

}

}

## Delete Supplier

A screenshot of a computer

Description automatically generated with medium confidence

Delete supplier will load a separate data grid view and allow the user to select a supplier and press the delete button with a confirmation message appearing to confirm the deletion.

public partial class frmDeleteSupplier : Form

{

SqlDataAdapter daSupplier;

DataSet dsRoadTripRentals = new DataSet();

SqlCommandBuilder cmdBSupplier;

DataRow drSupplier;

String connStr, sqlSupplier;

public frmDeleteSupplier()

{

InitializeComponent();

}

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

{

if (dgvSupplier.SelectedRows.Count == 0)

{

MessageBox.Show("Please select a supplier from the list.", "Select Supplier");

}

else

{

drSupplier = dsRoadTripRentals.Tables["Supplier"].Rows.Find(dgvSupplier.SelectedRows[0].Cells[0].Value);

string tempName = drSupplier["SupplierName"].ToString();

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

{

drSupplier.Delete();

daSupplier.Update(dsRoadTripRentals, "Supplier");

}

}

}

private void frmDeleteSupplier\_FormClosing(object sender, FormClosingEventArgs e)

{

MyGlobals.frmClosing = true;

}

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

{

connStr = @"Data Source = .\sqlExpress; Initial Catalog = RoadTripRentals; Integrated Security = true";

sqlSupplier = @"select \* from Supplier";

daSupplier = new SqlDataAdapter(sqlSupplier, connStr);

cmdBSupplier = new SqlCommandBuilder(daSupplier);

daSupplier.FillSchema(dsRoadTripRentals, SchemaType.Source, "Supplier");

daSupplier.Fill(dsRoadTripRentals, "Supplier");

dgvSupplier.DataSource = dsRoadTripRentals.Tables["Supplier"];

dgvSupplier.AutoResizeColumns(DataGridViewAutoSizeColumnsMode.AllCells);

}

}

}

## Add Supplier

A screenshot of a computer

Description automatically generated with medium confidence

Clicking on the Add button on the side panel in the Suppliers sub menu will display the Add Supplier form, it has identical validation to the Edit Supplier form only it will add a new row to the database instead of editing an existing row. The cancel button will close the form once the pop-up message is confirmed

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

{

connStr = @"Data Source = .\sqlExpress; Initial Catalog = RoadTripRentals; Integrated Security = true";

sqlSupplier = @"select \* from Supplier";

daSupplier = new SqlDataAdapter(sqlSupplier, connStr);

cmdBSupplier = new SqlCommandBuilder(daSupplier);

daSupplier.FillSchema(dsRoadTripRentals, SchemaType.Source, "Supplier");

daSupplier.Fill(dsRoadTripRentals, "Supplier");

int noRows = dsRoadTripRentals.Tables["Supplier"].Rows.Count;

if (noRows == 0)

lblSupplierNoValue.Text = "1001";

else

{

getNumber(noRows);

}

errP.Clear();

clearAddForm();

}

private void getNumber(int noRows)

{

drSupplier = dsRoadTripRentals.Tables["Supplier"].Rows[noRows - 1];

lblSupplierNoValue.Text = (int.Parse(drSupplier["SupplierNo"].ToString()) + 1).ToString();

}

void clearAddForm()

{

txtSupplierName.Clear();

txtStreet.Clear();

txtTown.Clear();

txtCounty.Clear();

txtPostcode.Clear();

txtTelNo.Clear();

txtEmail.Clear();

}

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

{

MySupplier mySupplier = new MySupplier();

bool ok = true;

errP.Clear();

try

{

mySupplier.SupplierNo = Convert.ToInt32(lblSupplierNoValue.Text.Trim());

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(lblSupplierNoValue, MyEx.validate());

}

try

{

mySupplier.SupplierName = txtSupplierName.Text.Trim();

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtSupplierName, MyEx.validate());

}

try

{

mySupplier.Street = txtStreet.Text.Trim();

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtStreet, MyEx.validate());

}

try

{

mySupplier.Town = txtTown.Text.Trim();

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtTown, MyEx.validate());

}

try

{

mySupplier.County = txtCounty.Text.Trim();

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtCounty, MyEx.validate());

}

try

{

mySupplier.Postcode = txtPostcode.Text.Trim();

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtPostcode, MyEx.validate());

}

try

{

mySupplier.SupplierTelNo = txtTelNo.Text.Trim();

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtTelNo, MyEx.validate());

}

try

{

mySupplier.SupplierEmail = txtEmail.Text.Trim();

}

catch (MyException MyEx)

{

ok = false;

errP.SetError(txtEmail, MyEx.validate());

}

try

{

if (ok)

{

drSupplier = dsRoadTripRentals.Tables["Supplier"].NewRow();

drSupplier["SupplierNo"] = mySupplier.SupplierNo;

drSupplier["SupplierName"] = mySupplier.SupplierName;

drSupplier["SupplierStreet"] = mySupplier.Street;

drSupplier["SupplierTown"] = mySupplier.Town;

drSupplier["SupplierCounty"] = mySupplier.County;

drSupplier["SupplierPostCode"] = mySupplier.Postcode;

drSupplier["SupplierTelNo"] = mySupplier.SupplierTelNo;

drSupplier["SupplierEmail"] = mySupplier.SupplierEmail;

dsRoadTripRentals.Tables["Supplier"].Rows.Add(drSupplier);

daSupplier.Update(dsRoadTripRentals, "Supplier");

MessageBox.Show("Supplier Added");

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

{

clearAddForm();

getNumber(dsRoadTripRentals.Tables["Supplier"].Rows.Count);

}

else

Close();

}

}

catch (Exception ex)

{

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

}

}

private void frmAddSupplier\_FormClosing(object sender, FormClosingEventArgs e)

{

MyGlobals.frmClosing = true;

}

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

{

if (MessageBox.Show("Cancel the addition of Supplier No: " + lblSupplierNoValue.Text + "?", "Add Supplier", MessageBoxButtons.YesNo) == System.Windows.Forms.DialogResult.Yes)

Close();

}

}

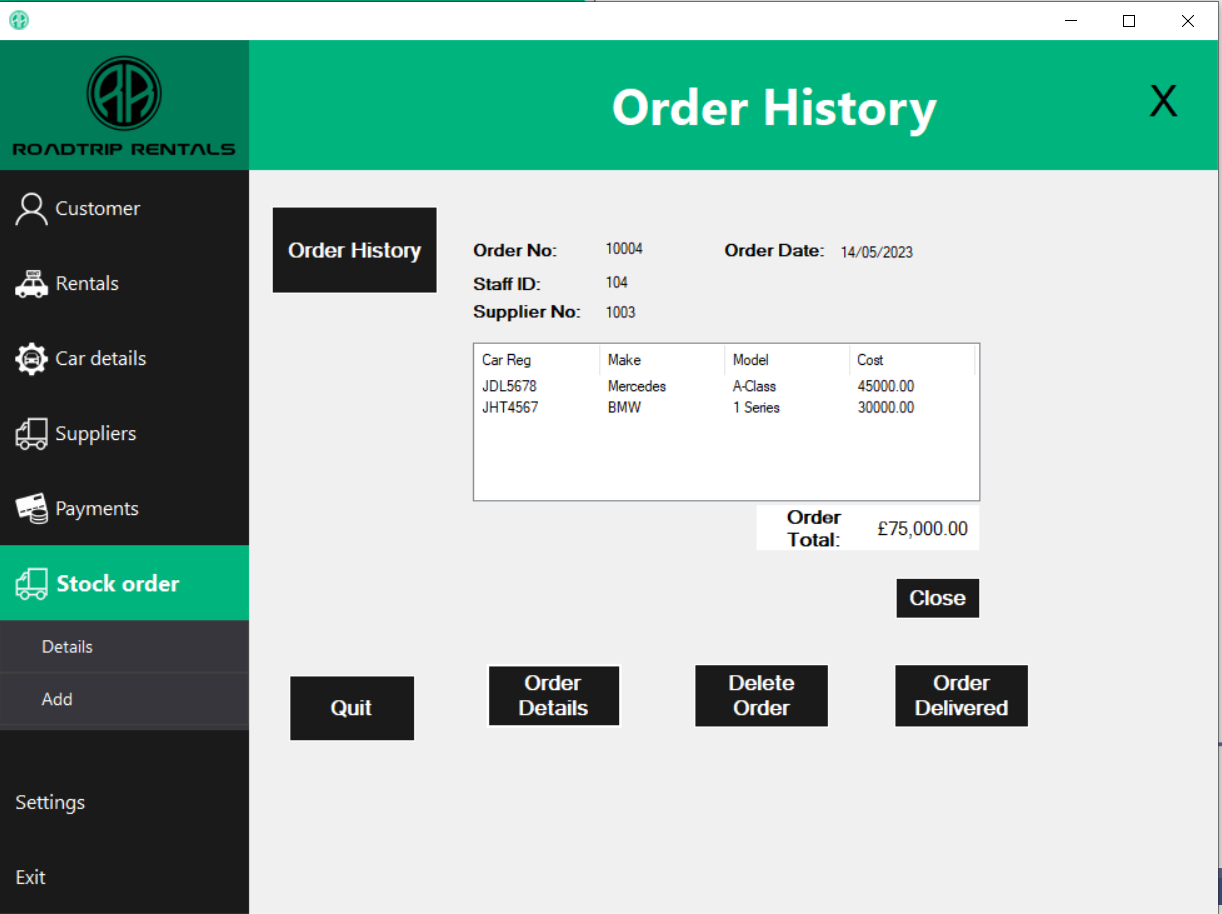
## Stock Order

A screenshot of a computer

Description automatically generated with medium confidence

Clicking on the Stock order button will open the sub menu and display 2 options - Details and Add. Details will load up a data grid view with the order history, 3 buttons below the data grid view, Order Details will display the details of the selected order, Delete Order will delete the selected order from the database along with associated details. Order Delivered will confirm the cars have been received and update the Details into the Car Details table to be used for Rentals.

## Order Details



When the Order Details button is clicked on a selected row of the Order History, the data grid view is hidden and the order details information is made visible with labels showing all the details and a list view showing all the cars in the order with the overall cost. The close button will hide the order details and make the data grid view visible again.

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

{

//Hide data grid view

dgvOrderHistory.Visible = false;

//display order details

lblOrderNo.Visible = true;

lblOrderNoValue.Visible = true;

lblOrderDate.Visible = true;

lblOrderDateValue.Visible = true;

lblStaffName.Visible = true;

lblStaffNameValue.Visible = true;

lblSupplier.Visible = true;

lblSupplierValue.Visible = true;

lvwOrderDetails.Visible = true;

lblOrderTotal.Visible = true;

lblOrderTotalValue.Visible = true;

btnClose.Visible = true;

//set up dataAdapter for Order Details in listview

sqlOrderDetails = @"SELECT sso.OrderNo, sso.SupplierCarReg, ss.Price, m.ModelDesc, m.Make

FROM SupplierStockOrder sso

INNER JOIN SupplierStock ss ON sso.SupplierCarReg = ss.SupplierCarReg

INNER JOIN Model m ON ss.ModelID = m.ModelID

WHERE sso.OrderNo = @OrderNo";

cmdOrderDetails = new SqlCommand(sqlOrderDetails, conn);

cmdOrderDetails.Parameters.AddWithValue("@OrderNo", dgvOrderHistory.SelectedRows[0].Cells[0].Value.ToString());

daOrderDetails = new SqlDataAdapter(cmdOrderDetails);

DataTable orderDetails = new DataTable();

daOrderDetails.Fill(orderDetails);

if (dgvOrderHistory.SelectedRows.Count == 0)

MessageBox.Show("Please select an order to view", "Order Details");

else

{

DataGridViewRow selectedRow = dgvOrderHistory.SelectedRows[0];

//Get the orderNo from the row that is selected

int orderNo = Convert.ToInt32(selectedRow.Cells[0].Value);

drStockOrder = dsRoadTripRentals.Tables["StockOrder"].Rows.Find(orderNo);

lblOrderNoValue.Text = drStockOrder["OrderNo"].ToString();

DateTime orderDate = (DateTime)drStockOrder["OrderDate"];

lblOrderDateValue.Text = orderDate.ToString("d");

lblStaffNameValue.Text = drStockOrder["StaffID"].ToString();

lblSupplierValue.Text = drStockOrder["SupplierNo"].ToString();

foreach (DataRow dr in orderDetails.Rows)

{

ListViewItem item = new ListViewItem(dr["SupplierCarReg"].ToString());

item.SubItems.Add(dr["Make"].ToString());

item.SubItems.Add(dr["ModelDesc"].ToString());

item.SubItems.Add(Convert.ToDecimal(dr["Price"]).ToString("0.00"));

lvwOrderDetails.Items.Add(item);

decimal carPrice = decimal.Parse(dr["Price"].ToString());

orderTotal += carPrice;

}

lblOrderTotalValue.Text = orderTotal.ToString("c");

}

}

## Delete Order

A screenshot of a computer

Description automatically generated with medium confidence

The Delete Order button will ask the user to confirm the deletion and then delete the selected row. The data in the SupplierStockOrder and StockOrder tables will then be deleted respectively.

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

{

if (dgvOrderHistory.SelectedRows.Count != 0)

{

DataGridViewRow selectedRow = dgvOrderHistory.SelectedRows[0];

int orderNo = Convert.ToInt32(selectedRow.Cells["OrderNo"].Value);

if (MessageBox.Show("Are you sure you want to delete Order No: " + orderNo + "?", "Delete Booking", MessageBoxButtons.YesNo) == System.Windows.Forms.DialogResult.Yes)

{

foreach (DataRow row in dsRoadTripRentals.Tables["SupplierStockOrder"].Rows)

{

int orderNoValue = Convert.ToInt32(row["OrderNo"]);

if (orderNoValue == orderNo)

{

row.Delete();

}

}

daOrderDetails.Update(dsRoadTripRentals, "SupplierStockOrder");

DataRow drStockOrder = dsRoadTripRentals.Tables["StockOrder"].Rows.Find(orderNo);

drStockOrder.Delete();

daStockOrder.Update(dsRoadTripRentals, "StockOrder");

}

}

else

MessageBox.Show("Please select an order from the list to delete.", "Delete Order");

}

## Order Delivered

A screenshot of a computer

Description automatically generated with medium confidence

The Order Delivered button checks if the order has already been delivered, if it has then it will give a message advising it has already been delivered. If the order has not been delivered, it will update the Car Details table with the car information so it can be used for Rentals, it then changes the OrderDelivered status to true in the StockOrder table.

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

{

if (dgvOrderHistory.SelectedRows.Count != 0)

{

DataGridViewRow selectedRow = dgvOrderHistory.SelectedRows[0];

int orderNo = Convert.ToInt32(selectedRow.Cells["OrderNo"].Value);

Boolean orderDelivered = Convert.ToBoolean(selectedRow.Cells["OrderDelivered"].Value);

if (!orderDelivered)

{

if (MessageBox.Show("Confirm Order No: " + orderNo + " has been delivered", "Confirm Delivery", MessageBoxButtons.YesNo) == System.Windows.Forms.DialogResult.Yes)

{

foreach (DataRow row in dsRoadTripRentals.Tables["SupplierStockOrder"].Rows)

{

sqlUpdateCars = @"INSERT INTO CarDetails(CarReg, ModelID, Colour, Mileage, FuelType, NoSeats, [Year], RentalCostID)

SELECT DISTINCT s.SupplierCarReg, s.ModelID, s.Colour, s.Mileage, s.FuelType, s.NoOfSeats, s.CarYear, RentalCostID = 1

FROM SupplierStock s INNER JOIN SupplierStockOrder so ON s.SupplierCarReg = so.SupplierCarReg

INNER JOIN StockOrder o ON so.OrderNo = o.OrderNo

WHERE o.OrderNo = @OrderNo

AND NOT EXISTS(SELECT 1 FROM CarDetails cd WHERE cd.CarReg = s.SupplierCarReg); ";

using (SqlConnection conn = new SqlConnection(connStr))

{

SqlCommand cmdCarDetails = new SqlCommand(sqlUpdateCars, conn);

cmdCarDetails.Parameters.AddWithValue("@OrderNo", orderNo);

conn.Open();

cmdCarDetails.ExecuteNonQuery();

}

}

sqlUpdateOrderDelivered = @"UPDATE StockOrder SET OrderDelivered = 1 WHERE OrderNo = @OrderNo;";

using (SqlConnection conn = new SqlConnection(connStr))

{

SqlCommand cmdOrderDelivered = new SqlCommand(sqlUpdateOrderDelivered, conn);

cmdOrderDelivered.Parameters.AddWithValue("@OrderNo", orderNo);

conn.Open();

cmdOrderDelivered.ExecuteNonQuery();

}

daCarDetails.Update(dsRoadTripRentals, "CarDetails");

MessageBox.Show("Order Delivery Confirmed");

}

}

else

MessageBox.Show("Order has already been delivered", "Order Delivered");

}

else

MessageBox.Show("Please select an order from the list", "Order Delivered");

}

}

## Add Order

A screenshot of a computer

Description automatically generated with medium confidence

The Add button in the sub menu of Stock Order opens the order form used to order cars from the SupplierStock table. The user must select a staff ID to enable the Supplier dropdown, then they must select a supplier to load the cars into the table from that supplier, this then activates the Add to Order to allow users to add cars to the order in the list view box as shown below.

A screenshot of a computer

Description automatically generated with medium confidence

The user can select a car in the list view and click Remove From Order to take the car out and the Order Total will update to reflect the cars added or removed from the order. The user can click Cancel Order to return to the home screen or if they click Complete Order the details will transfer to the Order History section.

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

{

if (cmbStaff.SelectedIndex != -1)

{

cmbSupplier.Enabled = true;

}

}

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

{

if (cmbSupplier.SelectedIndex != -1)

{

dgvSupplierStock.Enabled = true;

lblAddToOrder.Enabled = true;

if (!formLoading)

{

int supplierNo = (int)cmbSupplier.SelectedValue;

sqlSupplierStock = "SELECT \* FROM SupplierStock WHERE SupplierNo = @SupplierNo";

daSupplierStock = new SqlDataAdapter(sqlSupplierStock, conn);

daSupplierStock.SelectCommand.Parameters.AddWithValue("@SupplierNo", supplierNo);

dsRoadTripRentals.Tables["SupplierStock"].Clear();

daSupplierStock.Fill(dsRoadTripRentals, "SupplierStock");

dgvSupplierStock.DataSource = dsRoadTripRentals.Tables["SupplierStock"];

}

}

}

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

{

//disable combo boxes so different staff or supplier cannot be added to order

cmbStaff.Enabled = false;

cmbSupplier.Enabled = false;

//set up dataAdapter for car details in listview

sqlCarDetails = @"SELECT s.SupplierCarReg, s.Price, m.ModelDesc, m.Make FROM SupplierStock s INNER JOIN Model m ON s.ModelID = m.ModelID WHERE s.SupplierCarReg = @SupplierCarReg";

cmdCarDetails = new SqlCommand(sqlCarDetails, conn);

cmdCarDetails.Parameters.AddWithValue("@SupplierCarReg", dgvSupplierStock.SelectedRows[0].Cells[0].Value.ToString());

daCarDetails = new SqlDataAdapter(cmdCarDetails);

DataTable carDetails = new DataTable();

daCarDetails.Fill(carDetails);

bool exits = false;

if (dgvSupplierStock.SelectedRows.Count == 0)

MessageBox.Show("Please select a Car", "Car");

else

{

foreach (ListViewItem item in lvwOrderDetails.Items)

{

string selectedCar = item.SubItems[0].Text;

foreach (DataGridViewRow row in dgvSupplierStock.SelectedRows)

{

string supplierCar = row.Cells[0].Value.ToString();

if (selectedCar == supplierCar)

{

MessageBox.Show("Car already added to the order.", "Order");

exits = true;

break;

}

}

}

if (!exits)

{

foreach (DataRow dr in carDetails.Rows)

{

if (dr["SupplierCarReg"].ToString() == dgvSupplierStock.SelectedRows[0].Cells[0].Value.ToString())

{

ListViewItem item = new ListViewItem(dr["SupplierCarReg"].ToString());

item.SubItems.Add(dr["Make"].ToString());

item.SubItems.Add(dr["ModelDesc"].ToString());

item.SubItems.Add(Convert.ToDecimal(dr["Price"]).ToString("0.00"));

lvwOrderDetails.Items.Add(item);

decimal carPrice = decimal.Parse(dr["Price"].ToString());

orderTotal += carPrice;

lblOrderTotalValue.Text = orderTotal.ToString("c");

break;

}

}

}

}

}

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

{

if (lvwOrderDetails.SelectedItems.Count != 0)

{

var item = lvwOrderDetails.SelectedItems[0];

decimal carPrice = Decimal.Parse(item.SubItems[3].Text);

orderTotal -= carPrice;

lblOrderTotalValue.Text = orderTotal.ToString("c");

lvwOrderDetails.Items.Remove(item);

if (lvwOrderDetails.Items.Count == 0)

{

cmbStaff.Enabled = true;

cmbSupplier.Enabled = true;

}

}

}

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

{

DataRow drStockOrder, drStockOrderDetails;

int OrderNo;

int noRows = dsRoadTripRentals.Tables["StockOrder"].Rows.Count;

DateTime orderDeliveryDate = DateTime.Parse(lblOrderDate.Text.Trim());

drStockOrder = dsRoadTripRentals.Tables["StockOrder"].Rows[noRows - 1];

OrderNo = (int.Parse(drStockOrder["OrderNo"].ToString()) + 1);

if (lvwOrderDetails.Items.Count == 0)

MessageBox.Show("Please add a car to the order", "Order Details");

else

{

drStockOrder = dsRoadTripRentals.Tables["StockOrder"].NewRow();

drStockOrder["OrderNo"] = OrderNo;

drStockOrder["OrderDate"] = DateTime.Parse(lblOrderDate.Text.Trim());

drStockOrder["OrderDeliveryDate"] = orderDeliveryDate.AddDays(7);

drStockOrder["OrderDelivered"] = false;

drStockOrder["StaffID"] = cmbStaff.SelectedValue;

drStockOrder["SupplierNo"] = cmbSupplier.SelectedValue;

dsRoadTripRentals.Tables["StockOrder"].Rows.Add(drStockOrder);

daStockOrder.Update(dsRoadTripRentals, "StockOrder");

foreach (ListViewItem item in lvwOrderDetails.Items)

{

drStockOrderDetails = dsRoadTripRentals.Tables["SupplierStockOrder"].NewRow();

drStockOrderDetails["OrderNo"] = drStockOrder["OrderNo"];

drStockOrderDetails["SupplierCarReg"] = item.SubItems[0].Text;

dsRoadTripRentals.Tables["SupplierStockOrder"].Rows.Add(drStockOrderDetails);

daStockOrderDetails.Update(dsRoadTripRentals, "SupplierStockOrder");

}

MessageBox.Show("Order No: " + drStockOrder["OrderNo"].ToString() + " complete");

ClearOrder();

}

}

private void ClearOrder()

{

lvwOrderDetails.Items.Clear();

lblOrderTotalValue.Text = "";

cmbStaff.SelectedIndex = -1;

cmbSupplier.SelectedIndex = -1;

orderTotal = 0;

cmbStaff.Enabled = true;

cmbSupplier.Enabled = true;

dgvSupplierStock.DataSource = null;

dgvSupplierStock.Rows.Clear();

lblAddToOrder.Enabled = false;

}

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

{

if (MessageBox.Show("Do you want to cancel the order?", "Cancel Order", MessageBoxButtons.YesNo) == System.Windows.Forms.DialogResult.Yes)

{

Close();

}

}

# Reports

## Customer Rental car report - Jordan

A screenshot of a computer screen

Description automatically generated with low confidence

## Customer List report - Jordan

A screenshot of a computer

Description automatically generated with low confidence

## Supplier List report – David

A screenshot of a website

Description automatically generated with low confidence

## Supplier Stock order - David

A screenshot of a computer

Description automatically generated with low confidence

# Testing

## Input – Jordan

### Add/ Edit Customer

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Area Tested | Test No. | | Test value | | | Expected outcome | | Actual outcome |
| Customer Title | 1 | | "Mr" (Selected from Combo) | | | Accepted | | As expected |
|  | 2 | | "123" (User attempts to input text) | | | Rejected (Input not allowed in combo box) | | As expected |
|  | 3 | | "Random" (User attempts to input text) | | | Rejected (Input not allowed in combo box) | | As expected |
|  | 4 | | ““ (no value selected) | | | Rejected (not allowed a blank value) | | As expected |
| Customer Forename | 1 | | "John" | | | Accepted | | As expected |
|  | 2 | | "J" | | | Rejected (Too short) | | As expected |
|  | 3 | | "JohnathanJamesonWilliamsWonderlandJimmyJoo" | | | Rejected (Too long) | | As expected |
|  | 4 | | ““ | | | Rejected (not allowed a blank value) | | As expected |
|  | 5 | | Special characters @”-+”£ | | | Rejected | |  |
| Customer Surname | 1 | | "Doe" | | | Accepted | | As expected |
|  | 2 | | "D" | | | Rejected (Too short) | | As expected |
|  | 3 | | "DoeDoeJohnsonSmithwilliamwonderapplepear" | | | Rejected (Too long) | | As expected |
|  | 4 | | ““ | | | Rejected (not allowed a blank value) | | As expected |
|  | 5 | | Special characters @”-+”£ | | | Rejected | | As expected |
| Street Name | 1 | | 22 Anderson Crescent | | | Accepted | | As expected |
|  | 2 | | 22 | | | Rejected (Too short) | | As expected |
|  | 3 | | 22 Anderson Crescent 22 Anderson Crescent | | | Rejected (Too long) | | As expected |
|  | 4 | | ““ | | | Rejected (not allowed a blank value) | | As expected |
|  | 5 | | @”-+”£ | | | Rejected | | As expected |
| Town | 1 | | Limavady | | | Accepted | | As expected |
|  | 2 | | L | | | Rejected (Too short) | | As expected |
|  | 3 | | Limavadyuwhejfiehiuerhgoi | | | Rejected (Too long) | | As expected |
|  | 4 | | ““ | | | Rejected (not allowed a blank value) | | As expected |
|  | 5 | | @”-+”£ | | | Rejected | | As expected |
| County | 1 | | Derry | | | Accepted | | As expected |
|  | 2 | | D | | | Rejected (Too short) | | As expected |
|  | 3 | | DerryDerryDerry | | | Rejected (Too long) | | As expected |
|  | 4 | | ““ | | | Rejected (not allowed a blank value) | | As expected |
|  | 5 | | @”-+”£@”-+” | | | Rejected | | As expected |
| Postcode | 1 | BT490SY | | Accepted | Not as expected, CONSTRAINT within database stops no spaced postcodes from being entered, ideally want both spaces and no spaces working. | | **FIX**  **Database schema updated to accept both spaces and no spaces.** | | |
|  | 1.1 | | BT49 0SY | | | Accepted | | As expected |
|  | 2 | | BT3697 | | | Rejected (Too short) | | As expected |
|  | 3 | | BT490SY12 | | | Rejected (Too long) | | As expected |
|  | 4 | | ““ | | | Rejected (not allowed a blank value) | | As expected |
|  | 5 | | @”-+”£\* | | | Rejected | | As expected |
| TelNo | 1 | | 07543802994 | | | Accepted | | As expected |
|  | 2 | | 07543702 | | | Rejected (Too short) | | As expected |
|  | 3 | | 0754380299456721 | | | Rejected (Too long) | | As expected |
|  | 4 | | ““ | | | Rejected (not allowed a blank value) | | As expected |
|  | 5 | | @”-+”£@”-+”£ | | | Rejected | | As expected |
|  | 6 | | ABCDEFGHIJKL | | | Rejected (only numbers allowed) | | As expected |
| Email | 1 | | jordanmcelwee@hotmail.co.uk | | | Accepted | | As expected |
|  | 2 | | jordanmcelwee@hotmail.com | | | Accepted | | As expected |
|  | 3 | | jordanmcelweehotmail.co.uk | | | Rejected (Missing '@' character) | | As expected |
|  | 4 | | jordanmcelwee@ | | | Rejected (Invalid email format) | | As expected |
|  | 5 | | ““ | | | Rejected (not allowed a blank value) | | As expected |
|  | 6 | | @”-+”£\* | | | Rejected | | As expected |
|  | 7 | | @hotmail.com | | | Rejected | | As expected |

### Add/ Edit Car details

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Area Tested | Test No. | Test value | Expected outcome | Actual outcome | FIX (If applicable) |
| Car Reg | 1 | HFZ2711 | Accepted | As expected |  |
|  | 2 | HFZ27 | Rejected (Too short) | As expected |  |
|  | 3 | HFZ27111234 | Rejected (Too long) | As expected |  |
|  | 4 | “” | Rejected (not allowed a blank value) | As expected |  |
|  | 5 | +=@[]) | Rejected | Not as expected, its excepting special characters | MyValidation for validCarReg updated  OLD  public static bool validCarReg(string carReg)  {  return carReg.Length >= 6 && carReg.Length <= 10;  }  FIX  public static bool validCarReg(string carReg)  {  // Use regular expression pattern to match alphanumeric characters only  string pattern = @"^[a-zA-Z0-9]+$";  // Check if the carReg matches the pattern and the length is within the specified range  return Regex.IsMatch(carReg, pattern) && carReg.Length >= 6 && carReg.Length <= 10;  }  Now only accepting letters |
| Combo boxes (Model, Colour, Fuel Type, RentalCost) | 1 | Any input | Rejected (no inputs can be made, list is pre populated) | As expected |  |
|  | 2 | “” | Rejected, a value needs selected | As expected |  |
| Mileage | 1 | 0 | Rejected and auto changes to 1 (the min value allowed) | As expected |  |
|  | 2 | 9999999999 | Rejected and auto changes to 500000 (the max value allowed) | As expected |  |
|  | 3 | 500001 | Rejected and auto changes to 500000 (the max value allowed) | As expected |  |
|  | 4 | Up increment +500 onto value | Accepted, and doesn’t go above max value | As expected |  |
|  | 5 | Down increment  – 500 value | Accepted, and doesn’t go above min value | As expected |  |
|  | 6 | “” (delete value that’s populated) | Rejected | Not as expected, deleting the value doesn’t flag as an error. Upon further testing, this is the same case with all other numeric boxes | Mileage doesn’t have a perfect fix. Didn’t want to mark as read only as it takes away accuracy (temporary workaround is that the last value entered is saved, even if it was removed)      No.Seats and year have been marked as read only and only allowing the up down arrows to be used, which is ok since the ranges are less extreme.  Fixed in both Add and Edit forms |
| No.Seats | 1 | 0 | Rejected input not allowed | As expected |  |
|  | 2 | 99 | Rejected input not allowed | As expected |  |
|  | 3 | 9 | Rejected input not allowed | As expected |  |
|  | 4 | Up increment +1 onto value | Accepted, and doesn’t go above max value | As expected |  |
|  | 5 | Down increment  – 1 value | Accepted, and doesn’t go above min value | As expected |  |
|  | 6 | “” (delete value that’s populated) | Rejected (read only) | As expected |  |
| Year | 1 | Any Input | Rejected input not allowed | As expected |  |
|  | 2 | Up increment +1 onto value | Accepted, and doesn’t go above max value | As expected |  |
|  | 3 | Down increment  – 1 value | Accepted, and doesn’t go above min value | As expected |  |
|  | 4 | “” (delete value that’s populated) | Rejected (read only) | As expected |  |
|  | 5 | 2050 | Rejected as system doesn’t accept years higher than 2023 | As expected, the max value is 2050 which is used for future proofing the system if the current year we live in does go to 2050 |  |

### Add / Edit models

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Area Tested | Test No. | Test value | Expected outcome | Actual outcome | Fix (if applicable) |
| Model ID | 1 | A | Accepted | As expected |  |
|  | 2 | A4 | Accepted | As expected |  |
|  | 3 | A123456789 | Rejected (too long) | As expected |  |
|  | 4 | “” | Rejected (needs an input) | As expected |  |
|  | 5 | A (again) | Rejected on add click (unique key should not be duplicated) | As expected |  |
| Description | 1 | A | Accepted | As expected |  |
|  | 2 | A4 Audi | Accepted | As expected |  |
|  | 3 | Audi Description that is too big | Rejected (too long) | As expected |  |
|  | 4 | “” | Rejected (needs an input) |  |  |
| Make | 1 | A | Accepted | As expected |  |
|  | 2 | Audi | Accepted | As expected |  |
|  | 3 | Audi make that is too big | Rejected (too long) | As expected |  |
|  | 4 | “” | Rejected (needs an input) | As expected |  |

### Add / Edit Rental Cost

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Area Tested | Test No. | Test value | Expected outcome | Actual outcome | Fix (if applicable) |
| Rental Cost | 1 | Any input | Rejected (marked as read only) | As expected |  |
|  | 2 | Up increment +0.50 onto value | Accepted, and doesn’t go above max value (1000) | As expected |  |
|  | 3 | Down increment  – 0.50 value | Accepted, and doesn’t go above min value (10) | As expected |  |
|  | 4 | “” | Rejected (needs an input and impossible to be blank) | As expected |  |

### Add Rental / Edit Rental

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Area Tested | Test No. | Test Value | Expected Outcome | Actual Outcome |
| Rental Cost | 1 | **str** value is "A" | ListBox should display all customers whose names start with "A" | As Expected |
|  | 2 | **str** value is "" | ListBox should display all customers as "" is a wildcard | As Expected | |
|  | 3 | **str** value is "Z" | ListBox should display all customers whose names start with "Z" | As Expected |
| Populate Rental ListBox | 1 | **lstCustomer.SelectedValue** is a valid CustomerID | ListBox should display all rentals associated with the given CustomerID | As Expected | |
|  | 2 | **lstCustomer.SelectedValue** is a non-existing CustomerID | ListBox should be empty | As Expected |
| Filter Car Details | 1 | **cmbAddModelID.SelectedValue**, **txtAddNoSeats.Value** and **cmbAddRentalCostID.SelectedValue** are valid and match some cars | DataGridView should display all cars matching the selected ModelID, NoSeats, and RentalCostID | As Expected | |
|  | 2 | **cmbAddModelID.SelectedValue**, **txtAddNoSeats.Value** and **cmbAddRentalCostID.SelectedValue** do not match any cars | DataGridView should be empty | As Expected |
|  | 3 | One of **cmbAddModelID.SelectedValue**, **txtAddNoSeats.Value** or **cmbAddRentalCostID.SelectedValue** is null or invalid | DataGridView should be empty or show an error message | As Expected | |
| Confirm Car Selection Checkbox | 1 | Checkbox checked | pnlBooking should be enabled, dgvCars, panelCarDetails, pnlPayment should be disabled, btnResetFilters should not be visible, lblSelectCar text should be "Selected Car" | As Expected |
|  | 2 | Checkbox unchecked | pnlBooking should be disabled, dgvCars, panelCarDetails should be enabled, lblSelectCar text should be "Select Car" | As Expected | |
| Reset filters button | **1** | Button clicked | The car details should be reset | As Expected |
| Number of days combo box | **1** | Value selected | Total cost should be calculated, pnlPayment should be enabled | As Expected | |
| Remove Item Button | **1** | Item selected in lvwRental | Selected item should be removed | As Expected |
|  | **2** | No item selected in lvwRental | Nothing should happen | As Expected | |
| Add Button | **1** | Button clicked with valid rental ID, number of days, customer ID, and updated lvwRental items | Rental details should be updated successfully in the database and in lvwRental | As Expected |
| ResetForm | **1** | Method called | All controls should be reset to their initial states | As Expected | |

## Navigation – Jordan

### Main Menu

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Area Tested | Test No. | Test value | Expected outcome | Actual outcome |
| Main Menu Navigation | 1 | Click on btnCustomer | **panelCustomerSubmenu** should become visible, other submenus should hide | As expected |
|  | 2 | Click on btnRentals | **panelRentalsSubmenu** should become visible, other submenus should hide | As expected |
|  | 3 | Click on btnCar | **panelCarSubmenu** should become visible, other submenus should hide | As expected |
|  | 4 | Click on btnPayments | **panelPaymentSubmenu** should become visible, other submenus should hide | As expected |
|  | 5 | Click on btnExit | Application should close | As expected |
|  | 6 | panelMenu scrollable | Menu panel content should become scrollable if content become cut off | As expected |
|  | 7 | Main menu button colour change | All above main menu buttons should change colour upon click. Colour change is only in effect on active Submenu | As expected |
| Submenu Navigation | 8 | Click on btnMainCustomer | **frmMainCustomer** form should open within panelForm | As expected |
|  | 9 | Click on btnAddCustomer | **frmAddCustomer** form should open within panelForm | As expected |
|  | 10 | Click on btnMainRental | **frmMainRentals** form should open within panelForm | As expected |
|  | 11 | Click on btnAddRental | **frmAddRentals** form should open within panelForm | As expected |
|  | 12 | Click on btnMainCars | **frmMainCar** form should open within panelForm | As expected |
|  | 13 | Click on btnAddCar | **frmAddCar** form should open within panelForm | As expected |
|  | 14 | Click on btnMainModels | **frmMainModel** form should open within panelForm | As expected |
|  | 15 | Click on btnMainRentalCost | **frmMainRentalCost** form should open within panelForm | As expected |
|  | 16 | Click on btnMainPayment | **frmMainPayments** form should open within panelForm | As expected |
|  | 17 | Click on btnAddPayment | **frmAddPayments** form should open within panelForm | As expected |
|  | 18 | Click on btnEditRental | **frmEditRental** form should open within panelForm | As expected |
| Close Subforms | 19 | “X” icon button (named btnCloseSubForm) is disabled | **btnCloseSubForm** is not visible on the main menu of the application, therefore not enabled. | As expected |
|  |  | “X” icon button (named btnCloseSubForm) is enabled | **btnCloseSubForm** is visible on any of the subforms of the application, allowing immediate redirect to the main menu. | As expected |
| Home button | 20 | Company logo on click | **pictureBox2** on click redirects back to main menu | As expected |

### Main Customer

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Area Tested | Test No. | Test value | Expected outcome | Actual outcome |
| Main Customer | 1 | **Click on btnAddCustomer** | **frmAddCustomer** form should open within the **panelSub** | As expected |
|  | 2 | **btnEditCustomer** with a selected customer | **frmEditCustomer** form should open within the **panelSub** for the selected customer | As expected |
|  | 3 | **btnEditCustomer** with no selected customer | A MessageBox should appear with the message "Please select a customer to edit." | As expected |
|  | 4 | **btnDelCustomer** with a selected customer and confirm delete | Selected customer should be deleted from the database and DataGridView should be updated | As expected |
|  | 5 | **btnDelCustomer** with a selected customer and cancel delete | No change should occur in the database or DataGridView | As expected |
|  | 6 | **btnDelCustomer** with no selected customer | No action should occur as there is no customer to delete | As expected |

### Add Customer

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Area Tested | Test No. | Test value | Expected outcome | Actual outcome |
| Add Customer | **1** | Click on **btnAddCancel** | frmMainCustomer form should open within the panelSub | As expected |
|  | **2** | **Click btnAddAdd** with valid inputs and user chooses to add another customer | The form should clear and be ready for the next customer addition (with a prompt advising “Do you want to add another customer” in YesNo msgBox. | As expected |
|  | **3** | **Click btnAddAdd** with valid inputs and user chooses not to add another customer | frmMainCustomer form should open within the panelSub | As expected |
|  | **4** | **btnAddAdd** with invalid inputs | Error prompts should be displayed, and the form should not navigate further | As expected |

### Edit Customer

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Area Tested | Test No. | Test Value | Expected Outcome | Actual Outcome |
| Edit Customer | 1 | Click on **btnEditCancel** | **frmMainCustomer** form should open within the **panelSub** | As expected |
|  | 2 | Click on **btnEditAdd** with valid inputs | Customer data should be updated, and a confirmation message should appear | As expected |
|  | 3 | Click on **btnEditAdd** with invalid inputs (e.g., empty fields, incorrect data types, data that doesn't satisfy the constraints in the database) | Appropriate error prompts should be displayed | As expected |
|  | 4 | Click on **btnEditAdd** with non-existing customer ID | "Customer not found" error message should be displayed | As expected |

### Main Rental

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Area Tested | Test No. | Test Value | Expected Outcome | Actual Outcome |
| Rental Navigation | 1 | Click on **btnAddRental** | **frmAddRentals** form should open within the **panelSub** | As expected |
|  | 2 | Click on **btnEditRental** | **frmEditRental** form should open within the **panelSub** | As expected |
|  | 3 | Click on **btnDelRental** with a rental selected in **dgvRentals** | Confirmation message should appear, upon confirmation rental should be deleted from the grid and database | As expected |
|  | 4 | Click on **btnDelRental** without a rental selected in **dgvRentals** | "Please select a Rental to delete." message should appear | As expected |

### Add Rental

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Area Tested | | Test No. | | Test Value | | Expected Outcome | | | Actual Outcome |
| Alphabet Buttons | | 2 | | Click on Alphabet button (e.g., **btnA**, **btnB**, **btnC**, etc.) | | Retrieve customer details (based on surname) starting with the selected alphabet letter and populate the customer listbox with the retrieved data | | | As expected |
| Customer Selection | | 3 | | Customer from the listbox **lstCustomer** | | Enable the "Confirm Selection" checkbox, populate the car details in the data grid view, clear car selection, disable the "Confirm Selection" checkbox, and disable the booking and payment panels | | | As expected |
| Car Selection | | 4 | | Car from the data grid view **dgvCars** | | Display the selected car details in the respective labels, enable the "Confirm Selection" checkbox, and calculate the total cost | | | As expected |
| Reset Filters Button Click | | 5 | | N/A | | Clear the selected model, rental cost, and number of seats filters and display all cars in the data grid view | | | Somewhat as expected, while search parameters reset, the values in the boxes stay the same |
| Confirm Selection Checkbox | | 7 | | **chkConfirmCarSelection** state (checked/unchecked) | | Enable/disable the booking and payment panels based on the state of the checkbox | | | As expected |
| Model ID Selection | | 8 | | Model ID from the **cmbAddModelID** list | | Filter the cars in the data grid view based on the selected model ID | | | As expected |
| Rental Cost ID Selection | | 9 | | Rental Cost ID from the **cmbAddRentalCostID** list | | Filter the cars in the data grid view based on the selected rental cost ID. Displays rental cost assigned with the id. | | | As expected |
| Number of Seats Value Change | | 10 | | Number of seats value in the **txtAddNoSeats** numeric up down | | Filter the cars in the data grid view based on the selected number of seats | | | As expected |
| lstCustomer Click | | 11 | | Customer from the listbox | | Display the selected customer details in the respective labels, enable the car details panel, and enable the "Confirm Selection" checkbox | | | As expected |
| dgvCars SelectionChanged | | 12 | | Car from the data grid view | | Display the selected car details in the respective labels, enable the "Confirm Selection" checkbox, and calculate the total cost | | | As expected |
| Customer Selection Change | | 13 | | Customer selection in the listbox **lstCustomer changed midway** | | If there are existing items in list view, display a confirmation message asking if the user wants to proceed with a new customer. If the user selects "No," restore the previous selected customer. If the user selects "Yes," clear the existing bookings, update the previous selected customer, and allow the selection of a new customer. | | | As expected |
| Confirm Car Selection Check | | 15 | | Checkbox state (checked/unchecked) | | Enable/disable the booking and payment panels and the "Reset Filters" button based on the state of the checkbox | | | As expected |
| No of Days Selection Change | | 17 | | No of days selection in the dropdown list | | Recalculate the total cost and enable the payment panel | | | As expected |
| Remove Item Button Click | | 18 | | N/A | | Remove the selected item from the rental list view | | | As expected  # |
| Add Button Click | 19 | | Click on **btnAdd** | | Generate a unique payment ID, add the payment details to the PaymentType table, create a new Rental row with the customer and payment details, create new RentalCar rows for each selected car, display a success message, and prompt for adding another rental or returning to the main rentals form. Button disabled until an item has been added. | | Not as expected, button is enabled and available right from the start of the sub form, causing a break and an uncaught exception very early, since there is no data | **FIX**  Add button is disabled until an item has been populated in lvwRental, button disables again if customer selection is changed.      //other code// | | |
| Add Item Button Click | | 26 | | Click on **btnAddItem** | | Validate the selected car, number of days, and payment type, calculate the total cost, create a new lvwRentals with the car reg details and total cost, add the item to the rental list view. | | | As expected |

### Edit rental

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Area Tested | | Test No. | | Test Value | | Expected Outcome | | Actual Outcome |
| Hide Customer | | 27 | | No bookings for the customer | | Display an error message indicating no bookings are found for the customer, clear the selected customer, disable the "panel3" and "panelButtons" panels, and return to the initial state | | As expected |
| Rental Selection Change | | 28 | | Rental selection in the "**lstRental**" ListBox | | Enable the "panelButtons" panel, enable the "dtpStartDate" DateTimePicker, enable the "cmbNoOfDays" ComboBox, enable the "lvwRental" ListView, clear the "lvwRental" ListView, populate the selected rental's details (payment ID, number of days, payment type, payment date, booking date), and load the corresponding rental cars into the "lvwRental" ListView | | As expected |
| Update Rental Car Details | | 30 | | Edited rental car details  **PROCESS?** | | Update the selected rental car's details (total cost) in the dataset and database | | As expected |
| Remove Rental Car | | 31 | | Rental car selected for removal | | Remove the selected rental car from the "**lvwRental**" ListView | | As expected |
| Cancel Edit | | 32 | | Click on **btnAddCancel** | | Discard any changes made in the edit form and return to the main rentals form ("frmMainRentals") | | As expected |
| Save Edit | | 33 | | Click on **btnAdd** | | Save the edited rental and rental car details to the dataset and database, display a success message, and return to the main rentals form ("frmMainRentals") | | As expected |
| Delete Rental | | 36 | | **btnDelete** clicked | | Delete the selected rental, including associated rental car and payment records, from the dataset and database. Update the database with the delete commands assigned to the data adapters. Show a message indicating successful deletion. Clear and refill the dataset with fresh data. | | As expected |
| Get Payment Details | | 35 | | Rental's payment ID and details **PROCESS TESTING** | | Retrieve the payment type and payment date from the database based on the rental's payment ID and populate the corresponding controls | | As expected |

### General navigation

#### Forms for each section (“Customer, Rental, Car Details etc”) – Same code to process, with slight altercations

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Area Tested | Test No. | Test value | Expected outcome | Actual outcome |
| Main | 1 | **Click on btnAdd** | **frmAdd** form should open within the **panelSub** | As expected |
|  | 2 | **btnEdit** with a selected value from the dgv | **frmEdit** form should open within the **panelSub** for the selected value | As expected |
|  | 3 | **btnEdit** with no selected value | A MessageBox should appear with the message "Please select a -------- to edit." | As expected |
|  | 4 | **btnDel** with a selected customer and confirm delete | Selected customer should be deleted from the database and DataGridView should be updated | As expected |
|  | 5 | **btnDel** with a selected customer and cancel delete | No change should occur in the database or DataGridView | As expected |
|  | 6 | **btnDel** with no selected value | No action should occur as there is no value to delete | As expected |
| Add / Edit | 7 | **btnAddCancel** click | Returns back to the main form of the category chosen, in edit forms no changes will be processes if entered. | As expected |
|  | 8 | **btnAddAdd** click | Should add the content entered to the relevant database (details of input are outlined in input testing) and pop up message should show asking if another row of data would like to be started | As expected |
|  | 9 | **btnEditAdd** click | Should edit the content selected from the database, and re-entered to the relevant database (details of input are outlined in input testing) and pop up message should show asking if another row of data would like to be started | As expected |

## Processes / Outputs - Jordan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Area Tested | Test No. | Test value | Expected outcome | Actual outcome |
| Total Cost Calculation | 1 | Rental Duration: 2 days,  Daily Rate: £50 | Rental cost calculated as 2 x £50 = £100 | **As expected** |
| Car Mileage Update | 2 | Previous Mileage: 5000,  New Mileage: 6000 | Mileage updated to 6000 | **As expected** |
| Car Availability | 3 | Rental Start Date: 2023-04-01, Rental End Date: 2023-04-03 | Car availability status updated as unavailable for the specified duration | **As expected, the same car cannot be chosen on the same date, even by a different user** |
| Generate Unique Payment ID | 4 | **PaymentID = Max paymentID of PaymentType + 1** | Query the PaymentType table to retrieve the maximum payment ID, generate a new unique payment ID, and ensure it does not already exist in the PaymentType table | As expected |
| Get Number | 6 | **Gets max paymentID** | Fetch the next available Payment ID and display it in the appropriate control | As expected |
| Calculate Total Cost of Rentals | 7 | **PROCESS TESTING** | Calculate the total cost based on the selected number of days and the corresponding rental cost, and display the calculated total cost | As expected |

# Input Testing - David

**Add/Edit Supplier**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Area Tested | Test Number | Value/s Tested | Expected Outcome | Actual Outcome |
| Supplier Name | 1 | Abcd ef123 | System should accept value | As Expected |
|  | 2 | aaa | System should accept value | As Expected |
|  | 3 | aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa | System should accept value (50 characters exactly) | As Expected |
|  | 4 | aa | System should reject value and display “Supplier name must be 3 - 50 characters” | A screenshot of a computer  Description automatically generated with medium confidenceNot as Expected – Wrong error message displayed  Wrong ToString() method used  A screenshot of a computer  Description automatically generated  Changed the method name to validate()  Correct error message now showing for each field  A screenshot of a computer  Description automatically generated with medium confidence |
|  | 5 | aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa | System should reject value and display “Supplier name must be 3 - 50 characters” (51 characters) | As Expected |
|  | 6 | Leave blank | System should reject value and display “Supplier name must be 3 - 50 characters” | As Expected |
|  | 7 | 6 spaces | System should reject value and display “Supplier name must be 3 - 50 characters” | As Expected |
|  | 8 | !”£$%^%^ | System should reject value and display “Supplier name must be 3 - 50 characters” | As Expected |
| Street | 1 | Abcd ef123 | System should accept value | As Expected |
|  | 2 | aaaaa | System should accept value | As Expected |
|  | 3 | aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa | System should accept value (50 characters exactly) | As Expected |
|  | 4 | aaaa | System should reject value and display “Street must be 5 - 40 letters” | As Expected |
|  | 5 | aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa | System should reject value and display “Street must be 5 - 40 letters” (41 Characters) | As Expected |
|  | 6 | Leave blank | System should reject value and display “Street must be 5 - 40 letters” | As Expected |
|  | 7 | 6 spaces | System should reject value and display “Street must be 5 - 40 letters” | As Expected |
|  | 8 | !”£$%^%^ | System should reject value and display “Street must be 5 - 40 letters” | As Expected |
| Town | 1 | Abcd ef123 | System should accept value | As Expected |
|  | 2 | aa | System should accept value | As Expected |
|  | 3 | aaaaaaaaaaaaaaaaaaaa | System should accept value (20 characters exactly) | As Expected |
|  | 4 | a | System should reject value and display “Town must be 2 - 20 letters” | As Expected |
|  | 5 | aaaaaaaaaaaaaaaaaaaaa | System should reject value and display “Town must be 2 - 20 letters” (21 Characters) | As Expected |
|  | 6 | Leave blank | System should reject value and display “Town must be 2 - 20 letters” | As Expected |
|  | 7 | 6 spaces | System should reject value and display “Town must be 2 - 20 letters” | As Expected |
|  | 8 | !”£$%^%^ | System should reject value and display “Town must be 2 - 20 letters” | As Expected |
| County | 1 | Abcd ef123 | System should accept value | As Expected |
|  | 2 | aa | System should accept value | As Expected |
|  | 3 | aaaaaaaaaaaaaaaaaaaa | System should accept value (20 characters exactly) | As Expected |
|  | 4 | a | System should reject value and display “County must be 2 - 20 letters” | As Expected |
|  | 5 | aaaaaaaaaaaaaaaaaaaaa | System should reject value and display “County must be 2 - 20 letters” (21 Characters) | As Expected |
|  | 6 | Leave blank | System should reject value and display “County must be 2 - 20 letters” | As Expected |
|  | 7 | 6 spaces | System should reject value and display “County must be 2 - 20 letters” | As Expected |
|  | 8 | !”£$%^%^ | System should reject value and display “County must be 2 - 20 letters” | As Expected |
| Postcode | 1 | BT489PQ | System should accept value | As Expected |
|  | 2 | BT48 9PQ | System should accept value with space | As Expected |
|  | 3 | B T 4 8 9 P Q | System should reject value and display “Error: postcode must be 7-8 letters and alphanumeric only” | As Expected |
|  | 4 | BT489P | System should reject as too short | As Expected |
|  | 5 | 1111111 | System should reject as needs to be  [A-Z][A-Z][0-9][0-9] [0-9][A-Z][A-Z] | As Expected |
|  | 6 | AAAAAA | System should reject as needs to be  [A-Z][A-Z][0-9][0-9] [0-9][A-Z][A-Z] | As Expected |
|  | 7 | Leave blank | System should reject value and display “Error: postcode must be 7-8 letters and alphanumeric only” | As Expected |
|  | 8 | 6 spaces | System should reject value and display “Error: postcode must be 7-8 letters and alphanumeric only” | As Expected |
|  |  | !"£$%^& | System should reject value and display “Error: postcode must be 7-8 letters and alphanumeric only” | As Expected |
| Tel No | 1 | 12345678910 | System should accept value | As Expected |
|  | 2 | 123456789101234 | System should accept value | As Expected |
|  | 3 | 1234567891 (10 digits) | System should reject value and display “Telephone number must be 11 – 15 digits.”. | As Expected |
|  | 4 | 1234567891013245 (16 digits) | System should reject value and display “Telephone number must be 11 – 15 digits.”. | As Expected |
|  | 5 | AAAAAAAAAAA | System should reject value and display “Telephone number must be 11 – 15 digits.”. | As Expected |
|  | 6 | !”£$%^& | System should reject value and display “Telephone number must be 11 – 15 digits.”. | As Expected |
|  | 7 | Leave blank | System should reject value and display “Telephone number must be 11 – 15 digits.”. | As Expected |
|  | 8 | 6 spaces | System should reject value and display “Telephone number must be 11 – 15 digits.”. | As Expected |
| Email Address | 1 | John.doe@ter.com | System should accept value | As Expected |
|  | 2 | Johndoe@ter.com | System should accept value | As Expected |
|  | 3 | Abcd ef123 | System should reject value and display “Error: Email Address is invalid” | As Expected |
|  | 4 | 123145646 | System should reject value and display “Error: Email Address is invalid” | As Expected |
|  | 5 | Leave blank | System should reject value and display “Error: Email Address is invalid” | As Expected |
|  | 6 | 6 spaces | System should reject value and display “Error: Email Address is invalid” | As Expected |
|  | 7 | !”£$%^%^ | System should reject value and display “Error: Email Address is invalid” | As Expected |
|  | 8 | asdasdd@ | System should reject value and display “Error: Email Address is invalid” | As Expected |

**Navigation – David**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Area Tested | Test Number | Value/s Tested | Expected Outcome | Actual Outcome |
| Main Menu | 1 | Click Suppliers button | Sub menu opens with 2 buttons – Details and Add | As Expected |
|  | 2 | Click Suppliers button twice | Sub menu closes | As Expected |
|  | 3 | Click Details button in Supplier sub menu | Supplier Menu displays | As Expected |
|  | 4 | Click Add button in Supplier sub menu | Add Supplier form displays | As Expected |
|  | 5 | Click Stock order button | Sub menu opens with 2 buttons – Details and Add | As Expected |
|  | 6 | Click Stock order button twice | Sub menu closes | As Expected |
|  | 7 | Click Details button in Stock order sub menu | Order History menu appears | As Expected |
|  | 8 | Click Add button in Stock order sub menu | Order Stock menu appears | As Expected |
| Suppliers Details Menu | 1 | Click Supplier Details button | Data grid view loads with all supplier details | As Expected |
|  | 2 | Click Edit button | Edit form loads with selected Supplier’s details displayed | As Expected |
|  | 3 | Click Edit button | Edit form loads of highlighted supplier from Data grid view on Display Supplier details form | As Expected |
|  | 4 | Click Delete button | Delete form opens with list of Suppliers details to choose to delete | As Expected |
|  | 5 | Click Delete Yes button | Confirms deletion of supplier | As Expected |
|  | 6 | Click Delete No button | Cancels deletion of supplier | As Expected |
|  | 7 | Click Quit button | Exits Supplier form and returns to the Main Menu | As Expected |
| Add Supplier Menu | 1 | Click Add button once new supplier details have been entered | Supplier Added confirmation message displays | As Expected |
|  | 2 | Click OK on confirmation message after supplier added | Message Box displays “Do you want to add another supplier?” | As Expected |
|  | 3 | Click Yes on Message Box “Do you want to add another supplier?” | A blank Add Supplier form appears and the Supplier No increments by 1 | As Expected |
|  | 4 | Click No on Message Box “Do you want to add another supplier?” | User returned to the Main Menu | As Expected |
|  | 5 | Click Cancel button on the Add Supplier form | Message Box displays “Cancel addition of Supplier No: <num>?” | As expected |
|  | 6 | Click Yes on Message Box “Cancel addition of Supplier No: <num>?” | User returned to the Main Menu | As Expected |
|  | 7 | Click No on Message Box “Cancel addition of Supplier No: <num>?” | User remains on the Add Supplier form | As Expected |
| Order History Menu | 1 | Click Order History button | Data grid view loads with all Stock Orders logged | As Expected |
|  | 2 | Click Order Details button | Data grid view is hidden and the selected order details is displayed | As Expected |
|  | 3 | Click Close button | Order details are hidden and the data grid view is displayed | As Expected |
|  | 4 | Click Delete Order button | Message Box displays with “Are you sure you want to delete Order No: <num>?” | As Expected |
|  | 5 | Click Yes on Message Box “Are you sure you want to delete Order No: <num>?” | Order No and associated details are deleted from list | As Expected |
|  | 6 | Click No on Message Box “Are you sure you want to delete Order No: <num>?” | User is returned to Order History screen | As Expected |
|  | 7 | Click Order Delivered button on an order that is already delivered | Message Box displays “Order has already been delivered” | As Expected |
|  | 8 | Click Order Delivered button on an order that has not been delivered | Message Box displays “Confirm Order No: <num> has been delivered” | As Expected |
|  | 9 | Click Yes on Message Box “Confirm Order No: <num> has been delivered” | Message Box displays “Order Delivery Confirmed” and OrderDelivered column is true in Order History | As Expected |
|  | 10 | Click No on Message Box “Confirm Order No: <num> has been delivered” | User is returned to Order History screen | As Expected |
|  | 11 | Click Quit button | User is returned to Main Menu | As Expected |
| Order Stock Menu | 1 | Click Staff ID combo box | List of Staff is displayed | As Expected |
|  | 2 | Select staff name from list in Staff ID combo box | Staff name is displayed in Staff ID combo box and Supplier combo box becomes active | As Expected |
|  | 3 | Click Supplier combo box | List of suppliers is displayed | As Expected |
|  | 4 | Select supplier name from list in Supplier combo box | Supplier name is displayed in combo box and a list of cars from selected supplier is displayed in data grid view. Add to Order button becomes active | As Expected |
|  | 5 | Click Add to Order button | Selected car in data grid view is added to the order details list view, displaying the Car Reg, Make, Model and Cost with the Order Total updating at the bottom. | As Expected |
|  | 6 | Click Remove From Order | The selected car in the list view is removed from the order and the Order Total is adjusted accordingly | As Expected |
|  | 7 | Click Cancel Order button | Message Box displays “Do you want to cancel the order?” | As Expected |
|  | 8 | Click Yes on Message Box “Do you want to cancel the order?” | User is returned to the home screen | As Expected |
|  | 9 | Click No on Message Box “Do you want to cancel the order?” | User remains on the Order Stock form with their current selection | As Expected |
|  | 10 | Click Complete Order with no cars added to order | Message Box displays “Please add a car to the order”. | As Expected |
|  |  | Click Complete Order with one or more cars added to order | Message Box displays “Order No: <num> complete”. | As Expected |

## Processes/ Outputs – David

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Area Tested | Test Number | Value/s Tested | Expected Outcome | Actual Outcome |
| Order Total | 1 | Car at £15,000 added to order | Order Total should display £15,000 | As Expected |
|  | 2 | Car at £15,000 and  car at £32,000 added to order | Order Total should display £47,000 | As Expected |
|  | 3 | Car at £15,000 removed from order with car at £32,000 | Order Total should display £32,000 | As Expected |
| Complete Order | 1 | Order completed with 3 cars totalling £76,000 – | Details should appear in order history table | As Expected – See screenshots below    Order History Table |
| Order Delivered | 1 | Order Delivered button clicked on selected Order No | Cars in order updated on Car Details table | As Expected  Car Details table showing cars added |