HND Data Project

Units:

213 Data Analysis and Design

236 Object Oriented Programming

Car Rental Application

Paul McKillop

The attached resource files are extracts from the working demonstration application.

The support videos provide a timeline for the creation of the artefacts for the application. Some processes depend on others, so the application creation sequence should be followed. Some classes will be created progressively to gradually add functionality.

Broadly, the application creation sequence should be:

- Set up the SQL Server Database
- Import the 'core data' to the database
- Set up the Users text file for validation
- Create the application folder structure
    - Data
    - Models
    - Utility
- Create the Models classes
    - User
    - Person
    - ClientType
    - Manufacturer
    - Model
    - Client
    - Vehicle
    - VehicleCommercial
    - VehicleDomestic
    - ClientBySurname
    - ClientAddress
    - Staff
- Create the AppUtility class to connect to SQL Server
- Set up the Utility class Authenticate to validate users
- Create the Data classes
    - ImportData
    - ManufacturerDB
    - ModelDB
    - VehicleDB
    - ClientTypeDB
    - PersonDB
- Create the Pages using XAML Layouts
    - Login
    - Dashboard
    - Vehicle
    - VehicleCreate
    - Client
    - ClientCreate
    - ClientUpdate
    - ClientDelete
    - ClientAddress
- Implement the code behind the pages

```
//-- ***************************************
//--    Project:         HND Data Project
//--    Module:          ClientTypeDB
//--    Author:          Paul McKillop
//--    Date created:    19 January 2019
//-- ***************************************

using RentalWPF.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using Dapper;
using System.Windows;

namespace RentalWPF.Data
{
    public static class ClientTypeDB
    {
        //-- Method to get all ClientTypes
        public static List<ClientType> GetAllClientTypes(string myDb)
        {
            //-- Set up list to return
            var clientTypes = new List<ClientType>();
            //-- Connection to Db
            var conn = AppUtility.DbConnection(myDb);

            try
            {
                if(conn.State == System.Data.ConnectionState.Closed)
                {
                    conn.Open();
                }
                //-- Use Dapper to execute the stored procedure
                clientTypes = conn.Query<ClientType>("spClientType_GetAll",
                commandType: System.Data.CommandType.StoredProcedure).ToList();
            }
            catch (Exception ex)
            {

                MessageBox.Show(ex.Message);
            }
            finally
            {
                conn.Close();
            }

            return clientTypes;
        }

    }
}
```

```
//-- ***************************************
//--    Project:         HND Data Project
//--    Module:          ImportData
//--    Author:          Paul McKillop
//--    Date created:    19 January 2019
//-- ***************************************

using System.Data;
using System.IO;
using System.Text.RegularExpressions;

namespace RentalWPF.Data
{
    public class ImportData
    {
        /// <summary>
        /// Library function to import
        /// Text into a DataTable
        /// </summary>
        /// <param name="strFilePath"></param>
        /// <returns></returns>
        public static DataTable GetTextFileData(string strFilePath)
        {
            StreamReader sr = new StreamReader(strFilePath);
            // Read the first line only for column headers
            // and use these to create the DataTable columns
            string[] headers = sr.ReadLine().Split(',');
            DataTable dt = new DataTable();
            foreach (string header in headers)
            {
                dt.Columns.Add(header);
            }
            // Read the reamining data into the DataTable
            // to the EndOfStream
            while (!sr.EndOfStream)
            {
                // Regex with escape caharacters
                string[] rows = Regex.Split(sr.ReadLine(),
                ",(?=(?:[^\"]*\"[^\"]*\")*[^\"]*$)");
                DataRow dr = dt.NewRow();
                for (int i = 0; i < headers.Length; i++)
                {
                    dr[i] = rows[i];
                }
                dt.Rows.Add(dr);
            }

            // return the DataTable from the method
            return dt;
        }
    }
}
```

```
//-- ***************************************
//--    Project:         HND Data Project
//--    Module:          ManufacturerDB
//--    Author:          Paul McKillop
//--    Date created:    19 January 2019
//-- ***************************************


using System;
using System.Collections.Generic;
using System.Linq;
using RentalWPF.Models;
using Dapper;
using System.Windows;

namespace RentalWPF.Data
{
    public static class ManufacturerDB
    {
        //-- Method to get all manufacturers
        public static List<Manufacturer> ManufacturersGetAll(string myDb)

        {
            //-- Connection to the database on the server
            var conn = AppUtility.DbConnection(myDb);

            //-- holding list for returned data
            var manufacturers = new List<Manufacturer>();

            try
            {
                //-- open the connection to the database
                if (conn.State == System.Data.ConnectionState.Closed)
                {
                    conn.Open();
                }

                //-- Get Dapper to get the data
                manufacturers = conn.Query<Manufacturer>("spManufacturer_GetAll",
                commandType: System.Data.CommandType.StoredProcedure).ToList();

            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
            finally
            {
                conn.Close();
            }

            //-- fulfil the mthod with the return of the list
            return manufacturers;
        }
    }
}
```

```
//-- ***************************************
//--     Project:         HND Data Project
//--     Module:          ModelDB
//--     Author:          Paul McKillop
//--     Date created:    19 January 2019
//-- ***************************************


using System;
using System.Collections.Generic;
using System.Linq;
using RentalWPF.Models;
using Dapper;
using System.Windows;



namespace RentalWPF.Data
{
    public static class ModelDB
    {

        //-- Method to get all models
        public static List<Model> ModelsGetAll(string myDb)
        {
            //-- Connection to the database on the server
            var conn = AppUtility.DbConnection(myDb);

            //-- holding list for returned data
            var models = new List<Model>();




            return models;
        }


        //-- Methid to get restricted list by manufacturer
        public static List<Model> ModelsGetByManufacturerID(string myDb, int manuId)
        {
            //-- Connection to the database on the server
            var conn = AppUtility.DbConnection(myDb);

            //-- holding list for returned data
            var models = new List<Model>();

            try
            {
                //-- open the connection to the database
                if (conn.State == System.Data.ConnectionState.Closed)
                {
                    conn.Open();
                }

                DynamicParameters param = new DynamicParameters();
                param.Add("@ManufacturerID", manuId);

                models = conn.Query<Model>("spModel_GetByManufacturerID", param,
                commandType: System.Data.CommandType.StoredProcedure).ToList();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
            finally
            {
                conn.Close();
            }
```

```
                return models;
            }
        }
    }
```

```
//-- ***************************************
//--     Project:         HND Data Project
//--     Module:          PersonDB
//--     Author:          Paul McKillop
//--     Date created:    19 January 2019
//-- ***************************************


using System;
using System.Collections.Generic;
using System.Linq;
using RentalWPF.Models;
using Dapper;
using System.Windows;
using System.Data.SqlClient;

namespace RentalWPF.Data
{

    /// <summary>
    /// This class must perform FULL CRUD on Person objects
    /// </summary>
    public static class PersonDB
    {
        //-- Create
        public static int ClientInsert(string myDb, Client myClient)
        {
            //-- DB Connection
            var conn = AppUtility.DbConnection(myDb);

            try
            {
                //-- open the connection to the database
                if (conn.State == System.Data.ConnectionState.Closed)
                {
                    conn.Open();
                }

                //-- Set up the parameters
                DynamicParameters param = new DynamicParameters();
                param.Add("@ClientType_id", myClient.ClientTypeID);
                param.Add("@ClientForename", myClient.Forename);
                param.Add("@ClientSurname", myClient.Surname);
                param.Add("@AccountNumber", myClient.AccountNumber);
                param.Add("MobilePhone", myClient.MobilePhone);

                //-- Execute the stored procedure to store the vehicle
                conn.Execute("spClient_Insert", param, commandType: System.Data.
                CommandType.StoredProcedure);
            }
            catch (Exception ex)
            {

                MessageBox.Show(ex.Message);
            }
            finally
            {
                conn.Close();
            }

            return 1;
        }

        //-- Read all
        public static List<Client> ClientsGetAll(string myDb)
        {
            var clients = new List<Client>();

            var conn = AppUtility.DbConnection(myDb);

            try
```

```
        {
            //-- open the connection to the database
            if (conn.State == System.Data.ConnectionState.Closed)
            {
                conn.Open();
            }

            clients = conn.Query<Client>("spClient_GetAll", commandType: System.
            Data.CommandType.StoredProcedure).ToList();
        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
        }
        finally
        {
            conn.Close();
        }

        return clients;
    }

    //-- Retrieve limited fields from Clients
    public static List<ClientBySurname> GetClientsBySurname(string myDb)
    {
        var clientsBySurname = new List<ClientBySurname>();

        var conn = AppUtility.DbConnection(myDb);

        try
        {
            //-- open the connection to the database
            if (conn.State == System.Data.ConnectionState.Closed)
            {
                conn.Open();
            }

            //-- Get the list by Dapper
            clientsBySurname = conn.Query<ClientBySurname>(
            "spClient_NamesBySurname", commandType: System.Data.CommandType.
            StoredProcedure).ToList();
        }
        catch (Exception ex )
        {
            MessageBox.Show(ex.Message);
        }
        finally
        {
            conn.Close();
        }

        return clientsBySurname;

    }

    //-- Read by ID
    public static Client ClientGetByID(string myDb, int clientId)
    {
        var client = new Client();

        var conn = AppUtility.DbConnection(myDb);

        try
        {
            //-- open the connection to the database
            if (conn.State == System.Data.ConnectionState.Closed)
            {
                conn.Open();
            }
```

```
            //-- Get the Client
            DynamicParameters param = new DynamicParameters();
            param.Add("@ClientID", clientId);

            client = conn.Query<Client>("spClient_GetByID", param, commandType:
            System.Data.CommandType.StoredProcedure) as Client;

            MessageBox.Show($"Forename: { client.Forename }");
            MessageBox.Show($"Surname: { client.Surname }");
            MessageBox.Show($"ClientTypeID: { client.ClientTypeID }");
            MessageBox.Show($"AccountNumber: { client.AccountNumber }");
            MessageBox.Show($"MobilePhone: { client.MobilePhone }");
        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
        }
        finally
        {
            conn.Close();
        }

        return client;
    }

    //-- Update
    public static int ClientUpdate(string myDb, Client myClient)
    {
        var client = new Client();

        var conn = AppUtility.DbConnection(myDb);

        try
        {
            //-- open the connection to the database
            if (conn.State == System.Data.ConnectionState.Closed)
            {
                conn.Open();
            }

            DynamicParameters param = new DynamicParameters();
            param.Add("@ClientID",myClient.Id);
            param.Add("@ClientType_id", myClient.ClientTypeID);
            param.Add("@ClientForename",myClient.Forename);
            param.Add("@ClientSurname", myClient.Surname);
            param.Add("@AccountNumber",myClient.AccountNumber);
            param.Add("@MobilePhone", myClient.MobilePhone);

            conn.Execute("spClient_Update", param, commandType: System.Data.
            CommandType.StoredProcedure);

            MessageBox.Show($"{ myClient.Forename } { myClient.Surname } updated"
            );

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
        finally
        {
            conn.Close();
        }

        return 1;
    }

    //-- Delete
    public static int ClientDelete(string myDb, int myClientId)
    {
```

```
        var conn = AppUtility.DbConnection(myDb);

        try
        {
            //-- open the connection to the database
            if (conn.State == System.Data.ConnectionState.Closed)
            {
                conn.Open();
            }

            DynamicParameters param = new DynamicParameters();
            param.Add("@ClientID", myClientId);

            conn.Execute("spClient_Delete", param, commandType: System.Data.
            CommandType.StoredProcedure);
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
        finally
        {
            conn.Close();
        }

        return 1;
    }


    //-- Demo of data retrieval using traditional SQl
    //-- rather than Dapper
    public static Client GetClientByIDSQL(string myDb,int clientID)
    {
        var myClient = new Client();

        var conn = AppUtility.DbConnection(myDb);

        var selectStatement = "SELECT * FROM Client WHERE id = @clientId";

        var selectCommand = new SqlCommand(selectStatement, conn);

        selectCommand.Parameters.AddWithValue("@clientId", clientID);


        try
        {
            //-- open the connection to the database
            if (conn.State == System.Data.ConnectionState.Closed)
            {
                conn.Open();
            }

            var reader = selectCommand.ExecuteReader();

            if (!reader.Read())
            {
                throw new InvalidOperationException("No record found");
            }

            myClient.Id = Convert.ToInt32(reader["id"]);
            myClient.ClientTypeID = Convert.ToInt32(reader["ClientType_id"]);
            myClient.Forename = reader["ClientForename"].ToString();
            myClient.Surname = reader["ClientSurname"].ToString();
            myClient.AccountNumber = reader["AccountNumber"].ToString();
            myClient.MobilePhone = reader["MobilePhone"].ToString();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
        finally
```

```
            {
                conn.Close();
            }

            return myClient;
        }


        //-- Method to enter a new Client Address
        public static int EnterClientAddress(string myDb, ClientAddress myAddress)
        {
            var conn = AppUtility.DbConnection(myDb);

            try
            {
                //-- open the connection to the database
                if (conn.State == System.Data.ConnectionState.Closed)
                {
                    conn.Open();
                }

                DynamicParameters param = new DynamicParameters();
                param.Add("@Client_Id", myAddress.Client_Id);
                param.Add("@Address1",myAddress.Address1);
                param.Add("@Address2", myAddress.Address2);
                param.Add("@Town", myAddress.Town);
                param.Add("@County", myAddress.County);
                param.Add("@Postcode", myAddress.Postcode);
                param.Add("@Landline", myAddress.Landline);

                conn.Execute("spClient_InsertAddress", param, commandType: System.
                Data.CommandType.StoredProcedure);
            }
            catch (Exception ex)
            {

                MessageBox.Show(ex.Message);
            }
            finally
            {
                conn.Close();
            }

            return 1;
        }
    }
}
```

```
//-- ***************************************
//--     Project:         HND Data Project
//--     Module:          VehicleDB
//--     Author:          Paul McKillop
//--     Date created:    19 January 2019
//-- ***************************************


using System;
using System.Collections.Generic;
using RentalWPF.Models;
using Dapper;
using System.Windows;

namespace RentalWPF.Data
{
    public static class VehicleDB
    {
        //-- Methid to retrieve all vehicles from the database
        public static List<Vehicle> VehiclesGetAll()
        {
            var vehicles = new List<Vehicle>();

            return vehicles;
        }

        //-- Method to add a new domestic vehicle to the database
        public static int VehicleInsertDomestic(VehicleDomestic myVehicleDomestic)
        {
            var conn = AppUtility.DbConnection("RentalAzure");

            try
            {
                //-- open the connection to the database
                if (conn.State == System.Data.ConnectionState.Closed)
                {
                    conn.Open();
                }

                //-- Set up the parameters
                DynamicParameters param = new DynamicParameters();
                param.Add("@VehicleType_id",myVehicleDomestic.VehicleType_Id);
                param.Add("@Model_id",myVehicleDomestic.Model_Id);
                param.Add("@Registration", myVehicleDomestic.Registration);
                param.Add("@DateOnFleet",myVehicleDomestic.DateOnFleet);
                param.Add("@Gearbox",myVehicleDomestic.Gearbox);
                param.Add("@MaxLadenWeight", 0);

                //-- Execute the stored procedure to store the vehicle
                conn.Execute("spVehicle_Insert", param, commandType: System.Data.
                CommandType.StoredProcedure);


            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
            finally
            {
                conn.Close();
            }



            return 1;
        }

        //-- Method to add a new commercial vehicle to the database
        public static int VehicleInsertCommercial(VehicleCommercial
        myVehicleCommercial)
```

```
        {

            var conn = AppUtility.DbConnection("RentalAzure");

            try
            {
                //-- open the connection to the database
                if (conn.State == System.Data.ConnectionState.Closed)
                {
                    conn.Open();
                }

                //-- Set up the parameters
                DynamicParameters param = new DynamicParameters();
                param.Add("@VehicleType_id", myVehicleCommercial.VehicleType_Id);
                param.Add("@Model_id", myVehicleCommercial.Model_Id);
                param.Add("@Registration", myVehicleCommercial.Registration);
                param.Add("@DateOnFleet", myVehicleCommercial.DateOnFleet);
                param.Add("@Gearbox", "NA");
                param.Add("@MaxLadenWeight", myVehicleCommercial.MaxLadenWeight);

                //-- Execute the stored procedure to store the vehicle
                conn.Execute("spVehicle_Insert", param, commandType: System.Data.
                CommandType.StoredProcedure);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
            finally
            {
                conn.Close();
            }
            return 1;
        }
    }
}
```

```
//****************************
//-- Author:         Paul McKillop
//-- Date Created:   19 January 2019
//-- Description:    CLASS Client INHERITS Person
//-- Purpose:        Manage data for Client objects
//****************************


using System.Text;

namespace RentalWPF.Models
{
    public class Client : Person
    {
        public int ClientTypeID { get; set; }
        public string AccountNumber { get; set; }
        public string MobilePhone { get; set; }

        //-- Property to concatenate names
        public string ClientFullNameBySurname
        {
            get
            {
                return Surname + ", " + Forename;
            }
        }


        //-- Polymorphic method on parent class method through override
        //-- Call to method will use most derived function call
        public override string GetDetails()
        {
            StringBuilder sb = new StringBuilder();
            sb.Append("Account Number: ").Append("\t");
            sb.Append(AccountNumber).AppendLine();
            sb.Append("Mobile number: ").Append("\t");
            sb.Append(MobilePhone);

            return base.GetDetails() + "\n" + sb.ToString();
        }
    }


}
```

```
//-- ***************************************
//--    Project:         HND Data Project
//--    Module:          ClientAddress
//--    Author:          Paul McKillop
//--    Date created:    19 January 2019
//-- ***************************************


namespace RentalWPF.Models
{
    public class ClientAddress
    {
        public int Id { get; set; }
        public int Client_Id { get; set; }
        public string Address1 { get; set; }
        public string Address2 { get; set; }
        public string Town { get; set; }
        public string County { get; set; }
        public string Postcode { get; set; }
        public string Landline { get; set; }
    }
}
```

```
//-- ***************************************
//--    Project:         HND Data Project
//--    Module:          ClientBySurname
//--    Author:          Paul McKillop
//--    Date created:    19 January 2019
//-- ***************************************


namespace RentalWPF.Models
{
    public class ClientBySurname
    {
        public int Id { get; set; }
        public string NameBySurname { get; set; }
    }
}
```

```
//-- ***************************************
//--    Project:         HND Data Project
//--    Module:          ClientType
//--    Author:          Paul McKillop
//--    Date created:    19 January 2019
//-- ***************************************


namespace RentalWPF.Models
{
    public class ClientType
    {
        public int Id { get; set; }
        public string ClientTypeName { get; set; }
    }
}
```

```
//*****************************
//-- Author:         19 January 2019
//-- Description:    CLASS Manufacturer
//-- Purpose:        Manage data for Manufacturer objects
//*****************************


namespace RentalWPF.Models
{
    public class Manufacturer
    {
        public int Id { get; set; }
        public string ManufacturerName { get; set; }
    }
}
```

```
//****************************
//-- Author:         Paul McKillop
//-- Date Created:   19 January 2019
//-- Description:    CLASS Model
//-- Purpose:        Manage data for Model (as in models of cars) objects
//****************************


namespace RentalWPF.Models
{
    public class Model
    {
        public int Id { get; set; }
        public string ModelName { get; set; }
    }
}
```

```
//*****************************
//-- Author:         Paul McKillop
//-- Date Created:   23 December 2018
//-- Description:    CLASS Person
//-- Purpose:        Manage data for Person objects
//*****************************

namespace RentalWPF.Models
{
    public class Person
    {
        //-- Data members
        public int Id { get; set; }
        public string Forename { get; set; }
        public string Surname { get; set; }

        //-- Property to concatenate names
        public string FullNameBySurname
        {
            get
            {
                return $"{ Surname } , { Forename }";
            }
        }

        //-- Method to return name by surname
        public virtual string GetDetails()
        {
            return "Name: " + "\t" + "\t" + FullNameBySurname ;
        }
    }
}
```

```
//****************************
//-- Author:         Paul McKillop
//-- Date Created:   23 December 2018
//-- Description:    CLASS Staff INHERITS Person
//-- Purpose:        Manage data for Staff objects
//****************************

namespace RentalWPF.Models
{
    public class Staff : Person
    {
        public string Role { get; set; }
        public string ExtensionNumber { get; set; }
        public string Office { get; set; }
    }
}
```

```
//-- ***************************************
//--    Project:         HND Data Project
//--    Module:          User
//--    Author:          Paul McKillop
//--    Date created:    23 December 2018
//-- ***************************************


namespace RentalWPF.Models
{
    public class User
    {
        public string Username { get; set; }
        public string Password { get; set; }
    }
}
```

```
//****************************
//-- Author:         Paul McKillop
//-- Date Created:   23 December 2018
//-- Description:    CLASS Vehicle
//-- Purpose:        Manage data for Vehicle objects
//****************************


using System;

namespace RentalWPF.Models
{
    public class Vehicle
    {
        public int Id { get; set; }
        public int VehicleType_Id { get; set;  }
        public int Model_Id { get; set; }
        public string Registration { get; set; }
        public DateTime DateOnFleet { get; set; }
    }
}
```

```
//****************************
//-- Author:         Paul McKillop
//-- Date Created:   23 December 2018
//-- Description:    CLASS VehicleDomestic INHERITS Vehicle
//-- Purpose:        Manage data for Person objects
//****************************


namespace RentalWPF.Models
{
    public class VehicleCommercial : Vehicle
    {
        public int MaxLadenWeight { get; set; }
    }
}
```

```
//****************************
//-- Author:         Paul McKillop
//-- Date Created:   04 January 2019
//-- Description:    CLASS VehicleDomestic INHERITS Vehicle
//-- Purpose:        Manage data for VehicleDomestic objects
//****************************


namespace RentalWPF.Models
{
    public class VehicleDomestic : Vehicle
    {
        public string Gearbox { get; set; }
    }
}
```

```
<NavigationWindow x:Class="RentalWPF.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:RentalWPF"
        mc:Ignorable="d"
        Title="Rental" Height="800" Width="450" Source="PageLogin.xaml"
        WindowStartupLocation="CenterScreen">
</NavigationWindow>
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace RentalWPF
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : NavigationWindow
    {
        public MainWindow()
        {
            InitializeComponent();
        }
    }
}
```

```
<Page x:Class="RentalWPF.PageClient"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:RentalWPF"
      mc:Ignorable="d"
      d:DesignHeight="800" d:DesignWidth="450"
      Title="Client">

    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition />
            <ColumnDefinition />
        </Grid.ColumnDefinitions>

        <Grid.RowDefinitions>
            <RowDefinition Height="*"/>
            <RowDefinition Height="3*"/>
            <RowDefinition Height="3*"/>
        </Grid.RowDefinitions>

        <StackPanel
            HorizontalAlignment="Center"
            VerticalAlignment="Center"
            Grid.ColumnSpan="2">
            <Button
                x:Name="DashboardButton"
                Click="DashboardButton_Click"
                Margin="10"
                Padding="10"
                Width="200">
                Dashboard
            </Button>
            <TextBlock
                FontSize="36"
                FontWeight="Bold"
                Foreground="Blue">
                Select a Client Action
            </TextBlock>

        </StackPanel>


        <Button
            x:Name="UpdateClientButton"
            Click="UpdateClientButton_Click"
            Grid.Column="1"
            Grid.Row="1"
            Style="{StaticResource DashboardButtonStyle}"
            Margin="20 40"
            Background="AliceBlue">
            Update
        </Button>

        <Button
            x:Name="CreateClientButton"
            Click="CreateClientButton_Click"
            Grid.Column="0"
            Grid.Row="1"
            Style="{StaticResource DashboardButtonStyle}"
            Margin="20 40"
            Background="Bisque">
            Create
        </Button>

        <Button
            x:Name="DeleteClientButton"
            Click="DeleteClientButton_Click"
            Grid.Column="0"
            Grid.Row="2"
```

```
                Style="{StaticResource DashboardButtonStyle}"
                Margin="20 40"
                Background="Pink">
                Delete
            </Button>

            <Button
                x:Name="AddClientAddressButton"
                Click="AddClientAddressButton_Click"
                Grid.Column="1"
                Grid.Row="2"
                Style="{StaticResource DashboardButtonStyle}"
                Margin="20 40"
                Background="Beige">
                Add Address
            </Button>

    </Grid>
</Page>
```

```
//-- ***************************************
//--     Project:         HND Data Project
//--     Module:          PageClient
//--     Author:          Paul McKillop
//--     Date created:    04 January 2019
//-- ***************************************


using System.Windows;
using System.Windows.Controls;

namespace RentalWPF
{
    /// <summary>
    /// Interaction logic for PageClient.xaml
    /// </summary>
    public partial class PageClient : Page
    {
        public PageClient()
        {
            InitializeComponent();
        }



        //-- Command button
        private void CreateClientButton_Click(object sender, RoutedEventArgs e)
        {
            var myPage = new PageClientCreate();
            this.NavigationService.Navigate(myPage);
        }

        //-- Command button
        private void UpdateClientButton_Click(object sender, RoutedEventArgs e)
        {
            var myPage = new PageClientUpdate();
            this.NavigationService.Navigate(myPage);
        }

        //-- Command button
        private void DeleteClientButton_Click(object sender, RoutedEventArgs e)
        {
            var myPage = new PageClientDelete();
            this.NavigationService.Navigate(myPage);
        }

        //-- Command button
        private void ShowAllClientsButton_Click(object sender, RoutedEventArgs e)
        {

        }
        //-- Command button
        private void AddClientAddressButton_Click(object sender, RoutedEventArgs e)
        {
            var myPage = new PageClientAddress();
            this.NavigationService.Navigate(myPage);
        }
        //-- Command button
        private void DashboardButton_Click(object sender, RoutedEventArgs e)
        {
            var dashboard = new PageDashboard();
            this.NavigationService.Navigate(dashboard);
        }
    }
}
```

```
<Page x:Class="RentalWPF.PageClientAddress"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:RentalWPF"
      mc:Ignorable="d"
      d:DesignHeight="800" d:DesignWidth="450"
      Title="Client Address">

    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
        </Grid.RowDefinitions>
        <!-- Title -->
        <StackPanel
            HorizontalAlignment="Stretch"
            Margin="5">
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="2*"/>
                    <ColumnDefinition Width="3*"/>
                </Grid.ColumnDefinitions>
                <Button
                    x:Name="ClientHomeButton"
                    Click="ClientHomeButton_Click"
                    Padding="2"
                    Foreground="White"
                    Background="Blue"
                    FontWeight="SemiBold" Grid.ColumnSpan="2" Margin="0,0,280,0">
                    Client Home
                </Button>
                <TextBlock
                    Grid.Column="1"
                    HorizontalAlignment="Right"
                    VerticalAlignment="Center"
                    Style="{StaticResource ControlLabelStyle}"
                    FontSize="18">
            Enter client address
                </TextBlock>
            </Grid>
        </StackPanel>

        <StackPanel
            Grid.Row="1">
            <TextBlock
                Style="{StaticResource ControlLabelStyle}">
                Select a client
            </TextBlock>
            <ComboBox
                x:Name="SelectClientCombo"
                Loaded="SelectClientCombo_Loaded"
                SelectionChanged="SelectClientCombo_SelectionChanged"
                DisplayMemberPath="NameBySurname"
                SelectedValuePath="Id"
                SelectedValue="{Binding Id}"
                Style="{StaticResource DataControlStyle}" />
        </StackPanel>

        <StackPanel
            Grid.Row="2">
            <TextBlock
                Style="{StaticResource ControlLabelStyle}">
```

```
            Address Line 1
        </TextBlock>
        <TextBox
            x:Name="AddressLine1TextBox"
            Style="{StaticResource DataControlStyle}">

        </TextBox>
    </StackPanel>

    <StackPanel
        Grid.Row="3">
        <TextBlock
            Style="{StaticResource ControlLabelStyle}">
            Address Line 2
        </TextBlock>
        <TextBox
            x:Name="AddressLine2TextBox"
            Style="{StaticResource DataControlStyle}">

        </TextBox>
    </StackPanel>

    <StackPanel
        Grid.Row="4">
        <TextBlock
            Style="{StaticResource ControlLabelStyle}">
            Town
        </TextBlock>
        <TextBox
            x:Name="TownTextBox"
            Style="{StaticResource DataControlStyle}">

        </TextBox>
    </StackPanel>

    <StackPanel
        Grid.Row="5">
        <TextBlock
            Style="{StaticResource ControlLabelStyle}">
            County
        </TextBlock>
        <TextBox
            x:Name="CountyTextBox"
            Style="{StaticResource DataControlStyle}">

        </TextBox>
    </StackPanel>

    <StackPanel
        Grid.Row="6">
        <TextBlock
            Style="{StaticResource ControlLabelStyle}">
            Postcode
        </TextBlock>
        <TextBox
            x:Name="PostcodeTextBox"
            Style="{StaticResource DataControlStyle}">

        </TextBox>
    </StackPanel>

    <StackPanel
        Grid.Row="7">
        <TextBlock
            Style="{StaticResource ControlLabelStyle}">
            Landline
        </TextBlock>
        <TextBox
            x:Name="LandlineTextBox"
            Style="{StaticResource DataControlStyle}">
```

```
                </TextBox>
            </StackPanel>

            <Grid
                Grid.Row="8"
                VerticalAlignment="Center">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition />
                    <ColumnDefinition />
                </Grid.ColumnDefinitions>

                <Button
                    x:Name="ClearFormButton"
                    Click="ClearFormButton_Click"
                    Padding="10"
                    Margin="20 10 30 10"
                    FontWeight="Bold"
                    Foreground="Blue">
                    Clear form
                </Button>
                <Button
                    x:Name="EnterAddressButton"
                    Click="EnterAddressButton_Click"
                    Grid.Column="1"
                    Padding="10"
                    Margin="30 10 20 10"
                    FontWeight="Bold"
                    Foreground="Blue">
                    Add address
                </Button>
            </Grid>
        </Grid>
    </Page>
```

```
//-- ***************************************
//--    Project:         HND Data Project
//--    Module:          PageClientAddress
//--    Author:          Paul McKillop
//--    Date created:    04 January 2019
//-- ***************************************



using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using RentalWPF.Models;
using RentalWPF.Data;

namespace RentalWPF
{
    /// <summary>
    /// Interaction logic for PageClientAddress.xaml
    /// </summary>
    public partial class PageClientAddress : Page
    {
        //-- Module wide variables
        private List<ClientBySurname> clientsBySurname = new List<ClientBySurname>();
        private int currentSelectedClientId = 0;

        public PageClientAddress()
        {
            InitializeComponent();
            clientsBySurname = PersonDB.GetClientsBySurname("RentalAzure");
        }

        private void EnterAddressButton_Click(object sender, RoutedEventArgs e)
        {
            ClientAddress address = HarvestAddress();
            int result = PersonDB.EnterClientAddress("RentalAzure", address);

            MessageBox.Show("Address entered to database");


            GoToClientHome();
        }

        private void ClearFormButton_Click(object sender, RoutedEventArgs e)
        {
            ClearForm();
        }

        private void SelectClientCombo_Loaded(object sender, RoutedEventArgs e)
        {
            var combo = sender as ComboBox;
            combo.ItemsSource = clientsBySurname;
        }

        private void SelectClientCombo_SelectionChanged(object sender,
        SelectionChangedEventArgs e)
        {
            var combo = sender as ComboBox;

            //-- Update the current selected Id
            bool parseOk = Int32.TryParse(SelectClientCombo.SelectedValue.ToString(),
             out currentSelectedClientId);
        }

        private void ClearForm()
        {
            SelectClientCombo.SelectedValue = null;
            AddressLine1TextBox.Text = "";
            AddressLine2TextBox.Text = "";
            TownTextBox.Text = "";
```

```
            CountyTextBox.Text = "";
            PostcodeTextBox.Text = "";
            LandlineTextBox.Text = "";
        }

        //-- Harvest form data
        private ClientAddress HarvestAddress()
        {
            ClientAddress address = new ClientAddress();

            address.Client_Id = currentSelectedClientId;

            if(AddressLine1TextBox.Text == "")
            {
                address.Address1 = "Not known";
            }
            else
            {
                address.Address1 = AddressLine1TextBox.Text.Trim();
            }

            if (AddressLine2TextBox.Text == "")
            {
                address.Address2 = "";
            }
            else
            {
                address.Address2 = AddressLine2TextBox.Text.Trim();
            }

            if(TownTextBox.Text == "")
            {
                address.Town = "";
            }
            else
            {
                address.Town = TownTextBox.Text.Trim();
            }

            if(CountyTextBox.Text == "")
            {
                address.County = "";
            }
            else
            {
                address.County = CountyTextBox.Text.Trim();
            }

            if(PostcodeTextBox.Text == "")
            {
                address.Postcode = "";
            }
            else
            {
                address.Postcode = PostcodeTextBox.Text.Trim();
            }

            if(LandlineTextBox.Text == "")
            {
                address.Landline = "Not known";
            }
            else
            {
                address.Landline = LandlineTextBox.Text.Trim();
            }

            return address;
        }

        private void ClientHomeButton_Click(object sender, RoutedEventArgs e)
        {
```

```
                GoToClientHome();
            }

            private void GoToClientHome()
            {
                var clientHome = new PageClient();
                this.NavigationService.Navigate(clientHome);
            }
        }
    }
```

```
<Page x:Class="RentalWPF.PageClientCreate"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:RentalWPF"
      mc:Ignorable="d"
      d:DesignHeight="800" d:DesignWidth="450"
      Title="Create Client">

    <Grid
        Margin="5   ">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
        </Grid.RowDefinitions>

        <!-- Title -->
        <StackPanel
            HorizontalAlignment="Stretch"
            Margin="5">
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="2*"/>
                    <ColumnDefinition Width="3*"/>
                </Grid.ColumnDefinitions>
                <Button
                    x:Name="ClientHomeButton"
                    Click="ClientHomeButton_Click"
                    Padding="2"
                    Foreground="White"
                    Background="Blue"
                    FontWeight="SemiBold" Grid.ColumnSpan="2" Margin="0,0,280,0">
                    Client Home
                </Button>
                <TextBlock
                    Grid.Column="1"
                    HorizontalAlignment="Right"
                    VerticalAlignment="Center"
                    Style="{StaticResource ControlLabelStyle}"
                    FontSize="18">
            Create a client
                </TextBlock>
            </Grid>
        </StackPanel>

        <!-- ClientType combo -->
        <StackPanel
            VerticalAlignment="Center"
            Grid.Row="1">
            <TextBlock
                Style="{StaticResource ControlLabelStyle}">
                Client type
            </TextBlock>
            <ComboBox
                x:Name="ClientTypeCombo"
                Loaded="ClientTypeCombo_Loaded"
                SelectionChanged="ClientTypeCombo_SelectionChanged"
                DisplayMemberPath="ClientTypeName"
                SelectedValuePath="Id"
                SelectedValue="{Binding Id}"
                Style="{StaticResource DataControlStyle}">
            </ComboBox>
        </StackPanel>

        <!-- Forename -->
```

```
        <StackPanel
            VerticalAlignment="Center"
            Grid.Row="2">
            <TextBlock
                Style="{StaticResource ControlLabelStyle}">
                Forename
            </TextBlock>
            <TextBox
                x:Name="ForenameTextBox"
                Style="{StaticResource DataControlStyle}">

            </TextBox>
        </StackPanel>

        <!-- Surname -->
        <StackPanel
            VerticalAlignment="Center"
            Grid.Row="3">
            <TextBlock
                Style="{StaticResource ControlLabelStyle}">
                Surname
            </TextBlock>
            <TextBox
                x:Name="SurnameTextBox"
                Style="{StaticResource DataControlStyle}">

            </TextBox>
        </StackPanel>

        <!-- Account number-->
        <StackPanel
            VerticalAlignment="Center"
            Grid.Row="4">
            <TextBlock
                Style="{StaticResource ControlLabelStyle}">
                Account number
            </TextBlock>
            <TextBox
                x:Name="AccountNumberTextBox"
                Style="{StaticResource DataControlStyle}">

            </TextBox>
        </StackPanel>

        <!-- Mobile number -->
        <StackPanel
            VerticalAlignment="Center"
            Grid.Row="5">
            <TextBlock
                Style="{StaticResource ControlLabelStyle}">
                Mobile number
            </TextBlock>
            <TextBox
                x:Name="MobilePhoneTextBox"
                Style="{StaticResource DataControlStyle}">

            </TextBox>
        </StackPanel>

        <Grid
            Grid.Row="6"
            VerticalAlignment="Center">
            <Grid.ColumnDefinitions>
                <ColumnDefinition />
                <ColumnDefinition />
            </Grid.ColumnDefinitions>

            <Button
                x:Name="ClearDataButton"
                Click="ClearDataButton_Click"
                Padding="10"
```

```
                    Margin="20 10 30 10"
                    FontWeight="Bold"
                    Foreground="Blue">
                    Clear form
                </Button>
                <Button
                    x:Name="InsertClientButton"
                    Click="InsertClientButton_Click"
                    Grid.Column="1"
                    Padding="10"
                    Margin="30 10 20 10"
                    FontWeight="Bold"
                    Foreground="Blue">
                    Add client
                </Button>
            </Grid>

    </Grid>
</Page>
```

```
//-- ***************************************
//--    Project:        HND Data Project
//--    Module:         PageClientCreate
//--    Author:         Paul McKillop
//--    Date created:   06 January 2019
//-- ***************************************



using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using RentalWPF.Models;
using RentalWPF.Data;

namespace RentalWPF
{
    /// <summary>
    /// Interaction logic for PageClientCreate.xaml
    /// </summary>
    public partial class PageClientCreate : Page
    {

        private List<ClientType> clientTypes = new List<ClientType>();
        private int currentSelectedClientType = 0;

        public PageClientCreate()
        {
            InitializeComponent();
            //-- Load the ClientTypes
            clientTypes = ClientTypeDB.GetAllClientTypes("RentalAzure");
        }

        #region ClientTypes combo
        //-- Client Types load
        private void ClientTypeCombo_Loaded(object sender, RoutedEventArgs e)
        {
            var combo = sender as ComboBox;
            combo.ItemsSource = clientTypes;
            //-- set the current Id with combo selection
            //-- bool parseOk =
            Int32.TryParse(ClientTypeCombo.SelectedValue.ToString(), out
            currentSelectedClientType);
        }
        //-- Client Type changed
        private void ClientTypeCombo_SelectionChanged(object sender,
        SelectionChangedEventArgs e)
        {
            //-- set the current Id with combo selection
            bool parseOk = Int32.TryParse(ClientTypeCombo.SelectedValue.ToString(),
            out currentSelectedClientType);
        }
        #endregion

        private void InsertClientButton_Click(object sender, RoutedEventArgs e)
        {
            var client = HarvestClientData();

            if (client!= null)
            {
                int insertSuccess = PersonDB.ClientInsert("RentalAzure", client);
                //MessageBox.Show("Inserted " + client.Surname + ", " +
                client.Forename + " into Database");
                //-- This style is more processor efficient in string construction
                with variables
                if (insertSuccess == 1)
                {
                    MessageBox.Show($"Inserted {client.Surname}, {client.Forename}
                    into the database");
                }
```

```
            else
            {
                MessageBox.Show("Insert process not completed");
            }
        }

        GoToClientHome();
    }


    //-- Get all client data and ensure all fields filled
    private Client HarvestClientData()
    {
        //-- check the combo has been used
        if(currentSelectedClientType == 0)
        {
            MessageBox.Show("You must select a Client Type");
            return null;
        }

        var myClient = new Client
        {
            ClientTypeID = currentSelectedClientType,

        };


        //-- validate forename
        if (ForenameTextBox.Text == "")
        {
            MessageBox.Show("You must enter a Forename");

        }
        else
        {
            myClient.Forename = ForenameTextBox.Text.Trim();
        }

        //-- validate surname
        if (SurnameTextBox.Text == "")
        {
            MessageBox.Show("You must enter a Surname");
        }
        else
        {
            myClient.Surname = SurnameTextBox.Text.Trim();
        }

        //-- validate Account Number
        if(AccountNumberTextBox.Text == "")
        {
            MessageBox.Show("You must enter an Account Number");
        }
        else
        {
            myClient.AccountNumber = AccountNumberTextBox.Text.Trim();
        }

        //-- Mobile Phone
        if(MobilePhoneTextBox.Text == "")
        {
            myClient.MobilePhone = "Not known";
        }
        else
        {
            myClient.MobilePhone = MobilePhoneTextBox.Text.Trim();
        }

        return myClient;
    }
```

```
            //-- Utility method to clear the form
            private void ClearDataButton_Click(object sender, RoutedEventArgs e)
            {
                ClearForm();
            }

            private void ClearForm()
            {
                ClientTypeCombo.SelectedValue = null;
                ForenameTextBox.Text = "";
                SurnameTextBox.Text = "";
                AccountNumberTextBox.Text = "";
                MobilePhoneTextBox.Text = "";
            }

            private void ClientHomeButton_Click(object sender, RoutedEventArgs e)
            {
                GoToClientHome();
            }

            private void GoToClientHome()
            {
                var clientHome = new PageClient();
                this.NavigationService.Navigate(clientHome);
            }
        }
    }
```

```
<Page x:Class="RentalWPF.PageClientDelete"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:RentalWPF"
      mc:Ignorable="d"
      d:DesignHeight="800" d:DesignWidth="450"
      Title="PageClientDelete">

    <Grid
        Margin="10">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="3*"/>
            <RowDefinition Height="3*"/>
        </Grid.RowDefinitions>

        <!-- Title -->
        <StackPanel
            HorizontalAlignment="Stretch"
            Margin="5">
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="2*"/>
                    <ColumnDefinition Width="3*"/>
                </Grid.ColumnDefinitions>
                <Button
                    x:Name="ClientHomeButton"
                    Click="ClientHomeButton_Click"
                    Padding="2"
                    Foreground="White"
                    Background="Blue"
                    FontWeight="SemiBold" Grid.ColumnSpan="2" Margin="0,0,280,0">
                    Client Home
                </Button>
                <TextBlock
                    Grid.Column="1"
                    HorizontalAlignment="Right"
                    VerticalAlignment="Center"
                    Style="{StaticResource ControlLabelStyle}"
                    FontSize="18">
            Delete a client
                </TextBlock>
            </Grid>
        </StackPanel>

        <StackPanel
            Grid.Row="1"
            VerticalAlignment="Center"
            Margin="0 30 0 0">
            <TextBlock
                Style="{StaticResource ControlLabelStyle}">
                Select a client
            </TextBlock>

            <ComboBox
                x:Name="SelectClientComboBox"
               Loaded="SelectClientCombo_Loaded"
                SelectionChanged="SelectClientCombo_SelectionChanged"
                DisplayMemberPath="NameBySurname"
                SelectedValuePath="Id"
                SelectedValue="{Binding Id}"
                Style="{StaticResource DataControlStyle}">

            </ComboBox>

        </StackPanel>

        <StackPanel
            Grid.Row="2"
```

```
                VerticalAlignment="Center">
                <Button
                    x:Name="DeleteClientButton"
                    Click="DeleteClientButton_Click"
                    Padding="30"
                    Margin="40 10 40 80"
                    Background="Red"
                    Foreground="White"
                    FontSize="24"
                    FontWeight="Bold"
                    BorderBrush="DarkRed"
                    BorderThickness="5">
                    Delete
                </Button>
            </StackPanel>
        </Grid>
    </Page>
```

```
//-- ***************************************
//--    Project:        HND Data Project
//--    Module:         PageClientDelete
//--    Author:         18 December 2018
//-- ***************************************



using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using RentalWPF.Models;
using RentalWPF.Data;

namespace RentalWPF
{
    /// <summary>
    /// Interaction logic for PageClientDelete.xaml
    /// </summary>
    public partial class PageClientDelete : Page
    {
        //-- Module wide variables
        private List<ClientBySurname> clientsBySurname = new List<ClientBySurname>();
        private Client client = new Client();
        private int currentSelectedClientId = 0;

        public PageClientDelete()
        {
            InitializeComponent();
            //-- Get the clients by short data
            clientsBySurname = PersonDB.GetClientsBySurname("RentalAzure");
        }

        //-- populate the combo on form load
        private void SelectClientCombo_Loaded(object sender, RoutedEventArgs e)
        {
            var combo = sender as ComboBox;
            //-- Set the Combo data
            combo.ItemsSource = clientsBySurname;
        }

        //-- Update data based on client selected
        private void SelectClientCombo_SelectionChanged(object sender,
        SelectionChangedEventArgs e)
        {
            var combo = sender as ComboBox;

            //-- Update the current selected Id
            bool parseOk = Int32.TryParse(SelectClientComboBox.SelectedValue.ToString
            (), out currentSelectedClientId);
            //-- Populate the Client object
            client = PersonDB.GetClientByIDSQL("RentalAzure", currentSelectedClientId
            );
        }

        //-- Operation to delete a client
        private void DeleteClientButton_Click(object sender, RoutedEventArgs e)
        {
            if(currentSelectedClientId == 0)
            {
                MessageBox.Show("You must select a client");
                return;
            }


            string selectedClientData = client.GetDetails();
            //-- Check and confirm the delete operation
            var confirmDelete = MessageBox.Show(selectedClientData, "Are you sure?",
            MessageBoxButton.YesNo).ToString();
            //-- operation based on choice
```

```
                if (confirmDelete == "Yes")
                {
                    //-- Delete the client
                    int result = PersonDB.ClientDelete("RentalAzure",
                    currentSelectedClientId);

                    MessageBox.Show($"{ client.Forename } { client.Surname } deleted
                    from the database");
                }
                else
                {
                    //-- Tell cancelled
                    MessageBox.Show("Delete operation cancelled");
                }

                //-- Navigate to dashboard
                GoToClientHome();
            }

            private void ClientHomeButton_Click(object sender, RoutedEventArgs e)
            {
                GoToClientHome();
            }

            private void GoToClientHome()
            {
                var clientHome = new PageClient();
                this.NavigationService.Navigate(clientHome);
            }
        }
    }
```

```
<Page x:Class="RentalWPF.PageClientUpdate"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:RentalWPF"
      mc:Ignorable="d"
      d:DesignHeight="800" d:DesignWidth="450"
      Title="PageClientUpdate">

    <Grid
        Margin="5   ">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
        </Grid.RowDefinitions>

        <!-- Title -->
        <StackPanel
            HorizontalAlignment="Stretch"
            Margin="5">
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="2*"/>
                    <ColumnDefinition Width="3*"/>
                </Grid.ColumnDefinitions>
                <Button
                    x:Name="ClientHomeButton"
                    Click="ClientHomeButton_Click"
                    Padding="2"
                    Foreground="White"
                    Background="Blue"
                    FontWeight="SemiBold" Grid.ColumnSpan="2" Margin="0,0,280,0">
                    Client Home
                </Button>
                <TextBlock
                    Grid.Column="1"
                    HorizontalAlignment="Right"
                    VerticalAlignment="Center"
                    Style="{StaticResource ControlLabelStyle}"
                    FontSize="18">
            Edit client details
                </TextBlock>
            </Grid>
        </StackPanel>

        <!-- ClientType combo -->
        <StackPanel
            VerticalAlignment="Center"
            Grid.Row="1">
            <TextBlock
                Style="{StaticResource ControlLabelStyle}">
                Select the client to edit
            </TextBlock>
            <ComboBox
                x:Name="SelectClientCombo"
                Loaded="SelectClientCombo_Loaded"
                SelectionChanged="SelectClientCombo_SelectionChanged"
                DisplayMemberPath="NameBySurname"
                SelectedValuePath="Id"
                SelectedValue="{Binding Id}"
                Style="{StaticResource DataControlStyle}">

            </ComboBox>
        </StackPanel>
```

```
<!-- ClientType combo -->
<StackPanel
    VerticalAlignment="Center"
    Grid.Row="2">
    <TextBlock
        Style="{StaticResource ControlLabelStyle}">
        Client type
    </TextBlock>
    <ComboBox
        x:Name="ClientTypeCombo"
        Loaded="ClientTypeCombo_Loaded"
        SelectionChanged="ClientTypeCombo_SelectionChanged"
        DisplayMemberPath="ClientTypeName"
        SelectedValuePath="Id"
        SelectedValue="{Binding Id}"
        Style="{StaticResource DataControlStyle}">

    </ComboBox>
</StackPanel>

<!-- Forename -->
<StackPanel
    VerticalAlignment="Center"
    Grid.Row="3">
    <TextBlock
        Style="{StaticResource ControlLabelStyle}">
        Forename
    </TextBlock>
    <TextBox
        x:Name="ForenameTextBox"
        Style="{StaticResource DataControlStyle}">

    </TextBox>
</StackPanel>

<!-- Surname -->
<StackPanel
    VerticalAlignment="Center"
    Grid.Row="4">
    <TextBlock
        Style="{StaticResource ControlLabelStyle}">
        Surname
    </TextBlock>
    <TextBox
        x:Name="SurnameTextBox"
        Style="{StaticResource DataControlStyle}">

    </TextBox>
</StackPanel>

<!-- Account number-->
<StackPanel
    VerticalAlignment="Center"
    Grid.Row="5">
    <TextBlock
        Style="{StaticResource ControlLabelStyle}">
        Account number
    </TextBlock>
    <TextBox
        x:Name="AccountNumberTextBox"
        Style="{StaticResource DataControlStyle}">

    </TextBox>
</StackPanel>

<!-- Mobile number -->
<StackPanel
    VerticalAlignment="Center"
    Grid.Row="6">
    <TextBlock
```

```
                    Style="{StaticResource ControlLabelStyle}">
                    Mobile number
                </TextBlock>
                <TextBox
                    x:Name="MobilePhoneTextBox"
                    Style="{StaticResource DataControlStyle}">

                </TextBox>
            </StackPanel>

            <Grid
                Grid.Row="7"
                VerticalAlignment="Center">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition />
                    <ColumnDefinition />
                </Grid.ColumnDefinitions>

                <Button
                    x:Name="ClearFormButton"
                    Click="ClearFormButton_Click"
                    Padding="10"
                    Margin="20 10 30 10"
                    FontWeight="Bold"
                    Foreground="Blue">
                    Clear form
                </Button>
                <Button
                    x:Name="UpdateClientButton"
                    Click="UpdateClientButton_Click"
                    Grid.Column="1"
                    Padding="10"
                    Margin="30 10 20 10"
                    FontWeight="Bold"
                    Foreground="Blue">
                    Update client
                </Button>
            </Grid>

        </Grid>
    </Page>
```

```
//-- ***************************************
//--    Project:        HND Data Project
//--    Module:         PageClientUpdate
//--    Author:         Paul McKillop
//--    Date created:   14 December 2018
//-- ***************************************



using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using RentalWPF.Models;
using RentalWPF.Data;

namespace RentalWPF
{
    /// <summary>
    /// Interaction logic for PageClientUpdate.xaml
    /// </summary>
    public partial class PageClientUpdate : Page
    {
        //-- Module wide variables
        private List<ClientBySurname> clientsBySurname = new List<ClientBySurname>();
        private List<Client> clients = new List<Client>();
        private Client client = new Client();
        private List<ClientType> clientTypes = new List<ClientType>();
        private ClientType clientType = new ClientType();
        private int currentSelectedClientId = 0;
        private int currentSelectedClientType = 0;

        public PageClientUpdate()
        {
            InitializeComponent();
            clientsBySurname = PersonDB.GetClientsBySurname("RentalAzure");
            clientTypes = ClientTypeDB.GetAllClientTypes("RentalAzure");
        }




        private void SelectClientCombo_Loaded(object sender, RoutedEventArgs e)
        {
            var combo = sender as ComboBox;
            combo.ItemsSource = clientsBySurname;
        }

        private void SelectClientCombo_SelectionChanged(object sender,
        SelectionChangedEventArgs e)
        {
            var combo = sender as ComboBox;

            //-- Update the current selected Id
            bool parseOk = Int32.TryParse(SelectClientCombo.SelectedValue.ToString(),
             out currentSelectedClientId);


            Client myClient = new Client();

            //-- Now get the client with that Id
            myClient = PersonDB.GetClientByIDSQL("RentalAzure",
            currentSelectedClientId);

            //-- Populate client type
            ClientTypeCombo.SelectedIndex = myClient.ClientTypeID;

            //-- Fill in the text values
            ForenameTextBox.Text = myClient.Forename;
            SurnameTextBox.Text = myClient.Surname;
```

```
        AccountNumberTextBox.Text = myClient.AccountNumber;
        MobilePhoneTextBox.Text = myClient.MobilePhone;
    }

    private void ClientTypeCombo_Loaded(object sender, RoutedEventArgs e)
    {
        var combo = sender as ComboBox;
        combo.ItemsSource = clientTypes;
        combo.SelectedIndex = 0;
    }

    private void ClientTypeCombo_SelectionChanged(object sender,
    SelectionChangedEventArgs e)
    {
        bool parseOk = Int32.TryParse(ClientTypeCombo.SelectedValue.ToString(),
        out currentSelectedClientType);
    }

    private void UpdateClientButton_Click(object sender, RoutedEventArgs e)
    {
        var client = HarvestClientData();

        int result = PersonDB.ClientUpdate("RentalAzure", client);

        MessageBox.Show("Updated");

        //-- Navigate to dashboard
        GoToClientHome();

    }


    //-- Get all client data and ensure all fields filled
    private Client HarvestClientData()
    {
        var myClient = new Client();

        myClient.Id = currentSelectedClientId;
        myClient.ClientTypeID = currentSelectedClientType;
        myClient.Forename = ForenameTextBox.Text.Trim();
        myClient.Surname = SurnameTextBox.Text.Trim();
        myClient.AccountNumber = AccountNumberTextBox.Text.Trim();
        myClient.MobilePhone = MobilePhoneTextBox.Text.Trim();

        return myClient;
    }

    //-- Clear
    private void ClearFormButton_Click(object sender, RoutedEventArgs e)
    {
        ClearForm();
    }


    //-- Clear the form's data controls
    private void ClearForm()
    {
        SelectClientCombo.SelectedValue = null;
        ClientTypeCombo.SelectedValue = null;
        ForenameTextBox.Text = "";
        SurnameTextBox.Text = "";
        AccountNumberTextBox.Text = "";
        MobilePhoneTextBox.Text = "";
    }

    private void ClientHomeButton_Click(object sender, RoutedEventArgs e)
    {
        GoToClientHome();
    }

    private void GoToClientHome()
```

```
            {
                var clientHome = new PageClient();
                this.NavigationService.Navigate(clientHome);
            }
        }
    }
```

```
<Page x:Class="RentalWPF.PageDashboard"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:RentalWPF"
      mc:Ignorable="d"
      d:DesignHeight="800" d:DesignWidth="450"
      Title="Dashboard"
      ShowsNavigationUI="False">

    <Grid Margin="10">
        <Grid.ColumnDefinitions>
            <ColumnDefinition />
            <ColumnDefinition />
        </Grid.ColumnDefinitions>

        <Grid.RowDefinitions>
            <RowDefinition Height="*"/>
            <RowDefinition Height="2*"/>
            <RowDefinition Height="2*"/>
            <RowDefinition Height="2*"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <StackPanel
            HorizontalAlignment="Center"
            VerticalAlignment="Center"
            Grid.ColumnSpan="2">
            <TextBlock
                FontSize="36"
                FontWeight="Bold"
                Foreground="Blue">
                Select an option
            </TextBlock>
        </StackPanel>

        <Button
            x:Name="PageClientButton"
            Click="PageClientButton_Click"
            Grid.Row="1"
            Margin="20"
            Background="AliceBlue"
            BorderBrush="Blue"
            BorderThickness="2"
            FontSize="16"
            FontWeight="Bold">
            Clients
        </Button>

        <Button
            x:Name="PageVehicleButton"
            Click="PageVehicleButton_Click"
            Grid.Column="1"
            Grid.Row="1"
            Style="{StaticResource DashboardButtonStyle}"
            Background="Bisque">
            Vehicles
        </Button>

        <Button
            x:Name="UserManageButton"
            Grid.Column="0"
            Grid.Row="2"
            Style="{StaticResource DashboardButtonStyle}"
            Background="Beige">
            Users
        </Button>

        <Button
            x:Name="DashboardHelpButton"
```

```
                Click="DashboardHelpButton_Click"
                Grid.Column="1"
                Grid.Row="2"
                Margin="20"
                BorderBrush="#FF2802FC">
                <StackPanel
                    VerticalAlignment="Center"
                    HorizontalAlignment="Center"
                    Background="#FF2802FC">
                    <Image Source="bighelp.png" Stretch="UniformToFill" />
                </StackPanel>

            </Button>

            <Button
                x:Name="FutureFeature1Button"
                Grid.Column="0"
                Grid.Row="3"
                Style="{StaticResource DashboardButtonStyle}"
                Background="Yellow">
                Future Feature 1
            </Button>

            <Button
                x:Name="FutureFeature2Button"
                Grid.Column="1"
                Grid.Row="3"
                Style="{StaticResource DashboardButtonStyle}"
                Background="Aqua">
                Future Feature 1
            </Button>
            <StackPanel
                Grid.Row="4"
                HorizontalAlignment="Center"
                Margin="20 5"
                Grid.ColumnSpan="2">
                <Button
                    x:Name="QuitButton"
                    Click="QuitButton_Click"
                    Width="150"
                    Height="30">
                    Quit the application
                </Button>
            </StackPanel>

        </Grid>
    </Page>
```

```
//-- ***************************************
//--    Project:         HND Data Project
//--    Module:          PageDashboard
//--    Author:          Paul McKillop
//--    Date created:    14 December 2018
//-- ***************************************


using System.Windows;
using System.Windows.Controls;

namespace RentalWPF
{
    /// <summary>
    /// Interaction logic for PageDashboard.xaml
    /// </summary>
    public partial class PageDashboard : Page
    {
        public PageDashboard()
        {
            InitializeComponent();
        }

        //-- Navigate to PageClient
        private void PageClientButton_Click(object sender, RoutedEventArgs e)
        {
            var myPage = new PageClient();
            this.NavigationService.Navigate(myPage);
        }

        //-- Navigate to PageVehicle
        private void PageVehicleButton_Click(object sender, RoutedEventArgs e)
        {
            var myPage = new PageVehicle();
            this.NavigationService.Navigate(myPage);
        }

        //-- Help button feature
        private void DashboardHelpButton_Click(object sender, RoutedEventArgs e)
        {
            var myPage = new PageHelp();
            this.NavigationService.Navigate(myPage);
        }

        //-- Quit the application completely
        private void QuitButton_Click(object sender, RoutedEventArgs e)
        {
            System.Windows.Application.Current.Shutdown();
        }
    }
}
```

```
<Page x:Class="RentalWPF.PageHelp"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:RentalWPF"
      mc:Ignorable="d"
      d:DesignHeight="900" d:DesignWidth="610"
      Title="PageHelp">

    <Grid>

        <Grid.RowDefinitions>
            <RowDefinition Height="auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!-- Title -->
        <StackPanel
            HorizontalAlignment="Stretch"
            Margin="5">
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="2*"/>
                    <ColumnDefinition Width="3*"/>
                </Grid.ColumnDefinitions>
                <Button
                    x:Name="DashboardHomeButton"
                    Click="DashboardHomeButton_Click"
                    Padding="2"
                    Foreground="White"
                    Background="Blue"
                    FontWeight="SemiBold" Grid.ColumnSpan="2" Margin="0,0,280,0">
                    Dashboard
                </Button>
                <TextBlock
                    Grid.Column="1"
                    HorizontalAlignment="Right"
                    VerticalAlignment="Center"
                    Style="{StaticResource ControlLabelStyle}"
                    FontSize="18">
            Help page title
                </TextBlock>
            </Grid>
        </StackPanel>
        <WebBrowser
            x:Name="HelpBrowser"
            Grid.Row="1">

        </WebBrowser>
    </Grid>
</Page>
```

```
//-- ***************************************
//--    Project:         HND Data Project
//--    Module:          PageHelp
//--    Author:          Paul McKillop
//--    Date created:    14 December 2018
//-- ***************************************


using System.Windows;
using System.Windows.Controls;

namespace RentalWPF
{
    /// <summary>
    /// Interaction logic for PageHelp.xaml
    /// </summary>
    public partial class PageHelp : Page
    {
        public PageHelp()
        {
            InitializeComponent();
            this.HelpBrowser.Navigate("http://bbc.co.uk");
        }

        private void DashboardHomeButton_Click(object sender, RoutedEventArgs e)
        {
            var dashboard = new PageDashboard();
            this.NavigationService.Navigate(dashboard);
        }


    }
}
```

```
<Page x:Class="RentalWPF.PageLogin"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:RentalWPF"
      mc:Ignorable="d"
      d:DesignHeight="800" d:DesignWidth="450"
      Title="Login"
      ShowsNavigationUI="False">

    <Grid>
        <StackPanel>
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition />
                </Grid.ColumnDefinitions>

                <Grid.RowDefinitions>
                    <RowDefinition Height="*" />
                    <RowDefinition Height="*" />
                    <RowDefinition Height="*" />
                    <RowDefinition Height="3*" />
                </Grid.RowDefinitions>
            </Grid>

            <TextBlock
                Grid.Row="0"
                HorizontalAlignment="Center"
                FontSize="24"
                Margin="10 30"
                FontWeight="Bold"
                Foreground="Blue">
                Enter your credentials
            </TextBlock>

            <StackPanel Grid.Row="1">
                <TextBlock
                    Margin="10,100,10,10"
                    FontSize="16" FontWeight="Bold">
                    Enter your username
                </TextBlock>

                <TextBox
                    x:Name="UsernameTextBox"
                    FontSize="14"
                    Margin="10"
                    Padding="5,10">
                </TextBox>
            </StackPanel>

            <StackPanel
                Grid.Row="2">
                <TextBlock
                    Margin="10,30,10,10"
                    FontSize="16" FontWeight="Bold">
                    Enter your password
                </TextBlock>
                <TextBox
                    x:Name="PasswordTextBox"
                    FontSize="14"
                    Margin="10"
                    Padding="5 10">
                </TextBox>
            </StackPanel>

            <StackPanel Orientation="Horizontal"
                Grid.Row="3"
                HorizontalAlignment="Center"
                Margin="10 40">
```

```
                    <Grid
                        VerticalAlignment="Center"
                        HorizontalAlignment="Center">
                        <Grid.ColumnDefinitions>
                            <ColumnDefinition Width="*"/>
                            <ColumnDefinition Width="*"/>
                        </Grid.ColumnDefinitions>
                        <Grid.RowDefinitions>
                            <RowDefinition />
                            <RowDefinition />
                        </Grid.RowDefinitions>

                        <!-- This is the button control set for the form data-->
                        <Button
                        x:Name="ClearButton"
                        Click="ClearButton_Click"
                        Grid.Column="0"
                        HorizontalAlignment="Center"
                        Margin="10,10,50,10"
                        FontSize="24"
                            Padding="5" FontWeight="Bold" Width="150">
                            Clear
                        </Button>

                        <Button
                        x:Name="LoginButton"
                        Click="LoginButton_Click"
                        Grid.Column="1"
                        HorizontalAlignment="Center"
                        Margin="50,10,10,10"
                        FontSize="24"
                        Padding="5"
                        FontWeight="Bold"
                        Width="150"
                        >
                            Login
                        </Button>

                        <StackPanel
                            Grid.Row="1"
                            Grid.ColumnSpan="2">

                            <Button
                                x:Name="HelpButton"
                                Click="HelpButton_Click"
                                HorizontalAlignment="Stretch"
                                Margin="30 30">
                                Help
                            </Button>
                        </StackPanel>
                    </Grid>
                </StackPanel>
            </StackPanel>

        </Grid>
    </Page>
```

```
//-- ***************************************
//--    Project:         HND Data Project
//--    Module:          PageLogin
//--    Author:          Paul McKillop
//--    Date created:    14 December 2018
//-- ***************************************


using System.Windows;
using System.Windows.Controls;
using RentalWPF.Models;
using RentalWPF.Utility;

namespace RentalWPF
{
    /// <summary>
    /// Interaction logic for PageLogin.xaml
    /// </summary>
    public partial class PageLogin : Page
    {
        public PageLogin()
        {
            InitializeComponent();
            //-- ****************************
            //-- REMOVE BEFORE RELEASE
            FillTempCredentials();
            //--****************************
        }

        //-- Method to clear text box controls
        private void ClearButton_Click(object sender, RoutedEventArgs e)
        {
            this.UsernameTextBox.Text = "";
            this.PasswordTextBox.Text = "";
        }


        //-- Launch the help page
        private void HelpButton_Click(object sender, RoutedEventArgs e)
        {
            //-- navigate to a pdf help file
            var pageHelp = new PageHelp();
            this.NavigationService.Navigate(pageHelp);
        }

        //-- REMOVE BEFORE RELEASE
        private void FillTempCredentials()
        {
            this.UsernameTextBox.Text = "Paul";
            this.PasswordTextBox.Text = "tt5487%";
        }

        //-- validate the user and open PageDashboard
        private void LoginButton_Click(object sender, RoutedEventArgs e)
        {
            //-- Button to authenticate the user
            //-- Object for user data
            User myUser = new User();

            //-- Check entry in text box
            if (this.UsernameTextBox.Text != "")
            {
                myUser.Username = this.UsernameTextBox.Text.Trim();
            }
            else
            {
                MessageBox.Show("You must enter a USERNAME");
                return;
            }

            //-- Check entry in text box
```

```
                if (this.PasswordTextBox.Text != "")
                {
                    myUser.Password = this.PasswordTextBox.Text;
                }
                else
                {
                    MessageBox.Show("You must enter a PASSWORD");
                    return;
                }

                //-- Data is available to authenticate
                if (Authenticate.AuthenticateUser(myUser))
                {
                    var pageDashboard = new PageDashboard();
                    this.NavigationService.Navigate(pageDashboard);
                }
                else
                {
                    MessageBox.Show("The credentials are incorrect");
                    return;
                }
            }

        }
    }
```

```
<Page x:Class="RentalWPF.PageVehicle"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:RentalWPF"
      mc:Ignorable="d"
      d:DesignHeight="800" d:DesignWidth="450"
      Title="PageVehicle">

    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition />
            <ColumnDefinition />
        </Grid.ColumnDefinitions>

        <Grid.RowDefinitions>
            <RowDefinition Height="*"/>
            <RowDefinition Height="3*"/>
            <RowDefinition Height="3*"/>
        </Grid.RowDefinitions>

        <StackPanel
            HorizontalAlignment="Center"
            VerticalAlignment="Center"
            Grid.ColumnSpan="2">
            <Button
                x:Name="DashboardButton"
                Click="DashboardButton_Click"
                Margin="10"
                Padding="10"
                Width="200">
                Dashboard
            </Button>
            <TextBlock
                FontSize="36"
                FontWeight="Bold"
                Foreground="Blue">
                Select a Vehicle Action
            </TextBlock>
        </StackPanel>

        <Button
            x:Name="UpdateVehicleButton"
            Grid.Column="1"
            Grid.Row="1"
            Style="{StaticResource DashboardButtonStyle}"
            Margin="20 40"
            Background="AliceBlue">
            Update
        </Button>

        <Button
            x:Name="CreateVehicleButton"
            Click="CreateVehicleButton_Click"
            Grid.Column="0"
            Grid.Row="1"
            Style="{StaticResource DashboardButtonStyle}"
            Margin="20 40"
            Background="Bisque">
            Create
        </Button>

        <Button
            x:Name="DeleteVehicleButton"
            Click="DeleteVehicleButton_Click"
            Grid.Column="0"
            Grid.Row="2"
            Style="{StaticResource DashboardButtonStyle}"
            Margin="20 40"
            Background="Pink">
```

```
            Delete
        </Button>

        <Button
            x:Name="ShowAllVehiclesButton"
            Grid.Column="1"
            Grid.Row="2"
            Style="{StaticResource DashboardButtonStyle}"
            Margin="20 40"
            Background="Beige">
            Show All
        </Button>

    </Grid>
</Page>
```

```
//-- ***************************************
//--    Project:        HND Data Project
//--    Module:         PageVehicle
//--    Author:         Pa19 January 2019
//-- ***************************************


using System.Windows;
using System.Windows.Controls;

namespace RentalWPF
{
    /// <summary>
    /// Interaction logic for PageVehicle.xaml
    /// </summary>
    public partial class PageVehicle : Page
    {
        public PageVehicle()
        {
            InitializeComponent();
        }

        //-- Navigate to Create Vehicle
        private void CreateVehicleButton_Click(object sender, RoutedEventArgs e)
        {
            var myPage = new PageVehicleCreate();
            this.NavigationService.Navigate(myPage);
        }

        //-- FUTURE DEV - Delete Vehicle
        private void DeleteVehicleButton_Click(object sender, RoutedEventArgs e)
        {

        }

        private void DashboardButton_Click(object sender, RoutedEventArgs e)
        {
            var dashboard = new PageDashboard();
            this.NavigationService.Navigate(dashboard);
        }
    }
}
```

```
<Page x:Class="RentalWPF.PageVehicleCreate"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:RentalWPF"
      mc:Ignorable="d"
      d:DesignHeight="800" d:DesignWidth="450"
      Title="PageVehicleCreate"
      Loaded="Page_Loaded">

    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition />
        </Grid.ColumnDefinitions>

        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
        </Grid.RowDefinitions>

        <StackPanel
            HorizontalAlignment="Stretch"
            Margin="5">
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="2*"/>
                    <ColumnDefinition Width="3*"/>
                </Grid.ColumnDefinitions>
                <Button
                    x:Name="VehicleHomeButton"
                    Click="VehicleHomeButton_Click"
                    Padding="2"
                    Foreground="White"
                    Background="Blue"
                    FontWeight="SemiBold" Grid.ColumnSpan="2" Margin="0,0,280,0">
                    Vehicle Home
                </Button>
                <TextBlock
                    Grid.Column="1"
                    HorizontalAlignment="Right"
                    VerticalAlignment="Center"
                    Style="{StaticResource ControlLabelStyle}"
                    FontSize="18">
            Enter a new vehicle
                </TextBlock>
            </Grid>
        </StackPanel>

        <!-- Manufacturer Combo -->
        <StackPanel
            Grid.Row="1"
            VerticalAlignment="Center">
            <TextBlock
                Style="{StaticResource ControlLabelStyle}">
                Manufacturer
            </TextBlock>
            <ComboBox
                x:Name="ManufacturersCombo"
                Loaded="ManufacturersCombo_Loaded"
                SelectionChanged="ManufacturersCombo_SelectionChanged"
                ItemsSource="{Binding Path=manufacturers}"
                DisplayMemberPath="ManufacturerName"
                SelectedValuePath="Id"
                SelectedValue="{Binding Id}"
```

```
            Margin="5"
            FontSize="16"
            Padding="5">
        </ComboBox>
    </StackPanel>

    <!-- Models Combo -->
    <StackPanel
        Grid.Row="2"
        VerticalAlignment="Center">
        <TextBlock
            Style="{StaticResource ControlLabelStyle}">
            Model
        </TextBlock>
        <ComboBox
            x:Name="ModelsCombo"
            SelectionChanged="ModelsCombo_SelectionChanged"
            ItemsSource="{Binding Path=models}"
            DisplayMemberPath="ModelName"
            SelectedValuePath="Id"
            SelectedValue="{Binding Id}"
            Margin="5"
            FontSize="16"
            Padding="5">
        </ComboBox>
    </StackPanel>

    <!-- Type of vehicle Combo -->
    <StackPanel
        Grid.Row="3"
        VerticalAlignment="Center">
        <TextBlock
            Style="{StaticResource ControlLabelStyle}">
            Type of vehicle
        </TextBlock>
        <ComboBox
            x:Name="VehicleTypesCombo"
            Loaded="VehicleTypesCombo_Loaded"
            SelectionChanged="VehicleTypesCombo_SelectionChanged"
            Margin="5"
            FontSize="16"
            Padding="5">
        </ComboBox>
    </StackPanel>

    <StackPanel
        Grid.Row="4"
        VerticalAlignment="Center"
        HorizontalAlignment="Center">
        <TextBlock
            Style="{StaticResource ControlLabelStyle}"
            HorizontalAlignment="Center">
            Registration Number
        </TextBlock>
        <TextBox
            x:Name="RegistrationTextBox"
            Margin="5"
            FontSize="16"
            Padding="5"
            TextAlignment="Center"
            Width="250"
            VerticalAlignment="Center">
        </TextBox>
    </StackPanel>

    <StackPanel
        Grid.Row="5"
        VerticalAlignment="Center">
        <TextBlock
            Style="{StaticResource ControlLabelStyle}">
            Date on Fleet
```

```
        </TextBlock>
        <DatePicker
            x:Name="FleetDatePicker"
            SelectedDateChanged="FleetDatePicker_SelectedDateChanged"
            SelectedDateFormat="Short"
            Padding="5"
            FontSize="16"
            Margin="5"
            VerticalAlignment="Center">
        </DatePicker>
    </StackPanel>

    <Grid
        Grid.Row="6">
        <Grid.ColumnDefinitions>
            <ColumnDefinition />
            <ColumnDefinition />
        </Grid.ColumnDefinitions>

        <!--`Gearbox selection-->
        <StackPanel
        VerticalAlignment="Center"
        x:Name="GearboxPanel">
            <TextBlock
            Style="{StaticResource ControlLabelStyle}">
            Gearbox
            </TextBlock>
            <ComboBox
            x:Name="GearboxesCombo"
            Loaded="GearboxesCombo_Loaded"
            SelectionChanged="GearboxesCombo_SelectionChanged"
            Margin="5 10 20 5"
            FontSize="16"
            Padding="5">
            </ComboBox>
        </StackPanel>


        <StackPanel
        Grid.Column="1"
        VerticalAlignment="Center"
        x:Name="LadenWeightPanel">
            <TextBlock
            Style="{StaticResource ControlLabelStyle}">
            Maximum Laden Weight
            </TextBlock>
            <TextBox
            x:Name="LadenWeightTextBox"
                InputScope="Number"
            Margin="20 10 10 5"
            FontSize="16"
            Padding="5"
            TextAlignment="Center">

            </TextBox>
        </StackPanel>
    </Grid>

    <!-- Command buttons -->
    <StackPanel
        Orientation="Horizontal"
        HorizontalAlignment="Center"
        Grid.Row="8">
        <Grid
            HorizontalAlignment="Center">
            <Grid.ColumnDefinitions>
                <ColumnDefinition />
                <ColumnDefinition />
            </Grid.ColumnDefinitions>

            <Button
```

```
                        x:Name="ClearFormButton"
                        Click="ClearFormButton_Click"
                        Margin="10 10 20 10"
                        HorizontalAlignment="Stretch"
                        FontSize="16"
                        FontWeight="Bold"
                        Width=" 150">
                        Clear
                    </Button>

                    <Button
                        x:Name="AddVehicleButton"
                        Click="AddVehicleButton_Click"
                        Margin="20 10 10 10"
                        Grid.Column="1"
                        HorizontalAlignment="Stretch"
                        FontSize="16"
                        FontWeight="Bold"
                        Width="150">
                        Add
                    </Button>
                </Grid>
            </StackPanel>

        </Grid>
    </Page>
```

```
//-- ***************************************
//--    Project:        HND Data Project
//--    Module:         PageVehicleCreate
//--    Author:         Paul McKillop
//--    Date created:   14 December 2018
//-- ***************************************

using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using RentalWPF.Models;
using RentalWPF.Data;


namespace RentalWPF
{
    /// <summary>
    /// Interaction logic for PageVehicleCreate.xaml
    /// </summary>
    public partial class PageVehicleCreate : Page
    {
        //-- Module wide variables to track current values
        #region Module data variables
        private List<Manufacturer> manufacturers = new List<Manufacturer>();
        private List<Model> models = new List<Model>();
        private int currentManufacturerId = 0;
        private int currentModelId = 0;
        private string currentVehicleType = string.Empty;
        private int currentVehicleTypeID = 1;
        private DateTime selectedDate = DateTime.Today;
        private string formattedDate = string.Empty;
        private int maxLadenWeight = 0;
        private string currentSelectedGearbox = string.Empty;
        #endregion

        //-- Initialise the page
        public PageVehicleCreate()
        {
            InitializeComponent();
            manufacturers = ManufacturerDB.ManufacturersGetAll("RentalAzure");
        }

        //-- on Page load
        private void Page_Loaded(object sender, RoutedEventArgs e)
        {
            //-- Page level events

        }

        #region Manufacturers and Models drill-down
        //-- Get values to the manufacturers combo
        private void ManufacturersCombo_Loaded(object sender, RoutedEventArgs e)
        {
            var combo = sender as ComboBox;
            combo.ItemsSource = manufacturers;
            combo.SelectedItem = 0;
        }

        //-- get current ID for Manufacturer and load Models combo based on choice
        private void ManufacturersCombo_SelectionChanged(object sender,
        SelectionChangedEventArgs e)
        {
            //-- Getting the index from the Combo
            bool parseOk = Int32.TryParse(ManufacturersCombo.SelectedValue.ToString
            (), out currentManufacturerId);
            //-- use index selected to filter models by manufacturer
            //-- populate the ModelsCombo
            models = ModelDB.ModelsGetByManufacturerID("RentalAzure",
            currentManufacturerId);
            this.ModelsCombo.ItemsSource = models;
```

```
        this.ModelsCombo.SelectedIndex = 0;
        bool parseModel = Int32.TryParse(ModelsCombo.SelectedValue.ToString(),
        out currentModelId);
    }

    //-- Harvest current Model_Id
    private void ModelsCombo_SelectionChanged(object sender,
    SelectionChangedEventArgs e)
    {
        bool parseOk = Int32.TryParse(ModelsCombo.SelectedValue.ToString(), out
        currentModelId);
        //-- DEBUG CHECK: MessageBox.Show(currentModelId.ToString());
    }

    #endregion

    #region Vehicle Type management

    //-- Set up values for vehicle types
    private List<string> VehicleTypes()
    {
        List<string> types = new List<string>
        {
            "Commercial",
            "Domestic"
        };
        return types;
    }

    //-- Load the Vehicle Types
    private void VehicleTypesCombo_Loaded(object sender, RoutedEventArgs e)
    {
        var combo = sender as ComboBox;
        combo.ItemsSource = VehicleTypes();
        combo.SelectedIndex = 0;
    }

    //-- Record type of vehicle chosen
    private void VehicleTypesCombo_SelectionChanged(object sender,
    SelectionChangedEventArgs e)
    {
        var selectedComboItem = sender as ComboBox;
        currentVehicleType = selectedComboItem.SelectedItem as string;
        if (currentVehicleType == "Commercial")
        {
            //-- hide Gearboxes and show Laden Weigh
            GearboxPanel.Visibility = Visibility.Collapsed;
            LadenWeightPanel.Visibility = Visibility.Visible;
            currentVehicleTypeID = 1;
        }
        else
        {
            //-- Hide LadenWeight and show Gearboxes
            GearboxPanel.Visibility = Visibility.Visible;
            LadenWeightPanel.Visibility = Visibility.Collapsed;
            currentVehicleTypeID = 2;
        }
    }
    #endregion


    #region Gearbox type management
    //-- Set up values for Gearbox types
    private List<string> Gearboxes()
    {
        List<string> gearboxes = new List<string>
        {
            "Manual",
            "Automatic"
        };
        return gearboxes;
```

```
        }

        //-- Load the geabox types
        private void GearboxesCombo_Loaded(object sender, RoutedEventArgs e)
        {
            var combo = sender as ComboBox;
            combo.ItemsSource = Gearboxes();
            combo.SelectedIndex = 0;
        }

        //-- set current gearbox type
        private void GearboxesCombo_SelectionChanged(object sender,
        SelectionChangedEventArgs e)
        {
            var selectedComboItem = sender as ComboBox;
            currentSelectedGearbox = selectedComboItem.SelectedItem as string;
        }

        #endregion

        //-- Get the selected date
        private void FleetDatePicker_SelectedDateChanged(object sender,
        SelectionChangedEventArgs e)
        {
            selectedDate = FleetDatePicker.DisplayDate;
            //-- work with string values of the selected date
            formattedDate = FleetDatePicker.DisplayDate.ToShortDateString();
            // -- DEBUG MessageBox.Show(formattedDate);
        }

        #region Vehicle data harvesting
        //-- get commercial data
        private VehicleCommercial HarvestCommercial()
        {
            bool parseOk = Int32.TryParse(LadenWeightTextBox.Text, out maxLadenWeight
            );

            var commercial = new VehicleCommercial
            {
                VehicleType_Id = currentVehicleTypeID,
                Model_Id = currentModelId,
                Registration = RegistrationTextBox.Text.Trim(),
                DateOnFleet = FleetDatePicker.SelectedDate.Value.Date,
                MaxLadenWeight = maxLadenWeight
            };

            return commercial;
        }

        //-- Get domestic vehicle data
        private VehicleDomestic HarvestDomestic()
        {
            var domestic = new VehicleDomestic
            {
                VehicleType_Id = currentVehicleTypeID,
                Model_Id = currentModelId,
                Registration = RegistrationTextBox.Text.Trim(),
                DateOnFleet = FleetDatePicker.SelectedDate.Value.Date,
                Gearbox = currentSelectedGearbox
            };

            return domestic;
        }
        #endregion

        //-- add the current vehicle data to the database.
        private void AddVehicleButton_Click(object sender, RoutedEventArgs e)
        {
            if (currentVehicleTypeID == 1)
            {
                var vehicleCommercial = HarvestCommercial();
```

```
                    VehicleDB.VehicleInsertCommercial(vehicleCommercial);
                    MessageBox.Show("Commercial Vehicle " + vehicleCommercial.
                    Registration + " inserted into database");
                }
                else
                {
                    var vehicleDomestic = HarvestDomestic();
                    VehicleDB.VehicleInsertDomestic(vehicleDomestic);
                    MessageBox.Show("Domestic Vehicle " + vehicleDomestic.Registration +
                    " inserted into database");
                }

                GoToVehicleHome();
            }


            //-- Clear form data
            private void ClearFormButton_Click(object sender, RoutedEventArgs e)
            {
                ClearForm();
            }

            private void ClearForm()
            {
                LadenWeightTextBox.Text = "";
                GearboxesCombo.SelectedValue = null;
                FleetDatePicker.Text = "";
                RegistrationTextBox.Text = "";
                VehicleTypesCombo.SelectedValue = null;
                ModelsCombo.SelectedValue = null;
                ManufacturersCombo.SelectedValue = null;


            }

            private void VehicleHomeButton_Click(object sender, RoutedEventArgs e)
            {
                GoToVehicleHome();
            }

            private void GoToVehicleHome()
            {
                var vehicleHome = new PageVehicle();
                this.NavigationService.Navigate(vehicleHome);
            }
        }
    }
```

```
//-- ***************************************
//--    Project:        HND Data Project
//--    Module:         Authenticate
//--    Author:         Paul McKillop
//--    Date created:   19 January 2019
//-- ***************************************



using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using RentalWPF.Models;
using RentalWPF.Data;

namespace RentalWPF.Utility
{
    public class Authenticate
    {
        /// <summary>
        /// Authentication method
        /// </summary>
        /// <param name="myUser"></param>
        /// <returns></returns>
        public static bool AuthenticateUser(User myUser)
        {
            bool validated = false;

            //-- define path of user database
            //string driveLetter = ConfigurationManager.AppSettings("driveLetter");
            //-- Home string userDatabase = @"H:\users.txt";
            string userDatabase = @"I:\users.txt";
            //string userDatabase = @"E:\users.txt";
            //string userDatabase = @"F:\users.txt";
            //-- Define DataTable
            DataTable userData = new DataTable();

            //-- Create data table
            userData = ImportData.GetTextFileData(userDatabase);

            //-- validate credentials
            //-- Loop through the DataTable
            foreach (DataRow row in userData.Rows)
            {
                var currentUser = new User
                {
                    Username = row.Field<string>(0),
                    Password = row.Field<string>(1)
                };

                if (currentUser.Username == myUser.Username)
                {
                    if (currentUser.Password == myUser.Password)
                    {
                        validated = true;
                        break;
                    }
                }

            }

            return validated;

        }
    }
}
```

```
CREATE TABLE Manufacturer (
  id INTEGER  NOT NULL   IDENTITY ,
  ManufacturerName VARCHAR(45)  NOT NULL    ,
PRIMARY KEY(id));
GO



CREATE TABLE VehicleType (
  id INTEGER  NOT NULL   IDENTITY ,
  VehicleTypeName VARCHAR(45)      ,
PRIMARY KEY(id));
GO



CREATE TABLE ClientType (
  id INTEGER  NOT NULL   IDENTITY ,
  ClientTypeName VARCHAR(255)  NOT NULL    ,
PRIMARY KEY(id));
GO



CREATE TABLE Tariff (
  id INTEGER  NOT NULL   IDENTITY ,
  TariffName VARCHAR(20)  NOT NULL  ,
  TariffRate FLOAT  NOT NULL    ,
PRIMARY KEY(id));
GO



CREATE TABLE StaffType (
  id INTEGER  NOT NULL   IDENTITY ,
  StaffTypeName VARCHAR(20)  NOT NULL    ,
PRIMARY KEY(id));
GO



CREATE TABLE Staff (
  id INTEGER  NOT NULL   IDENTITY ,
  StaffType_id INTEGER  NOT NULL  ,
  StaffRef VARCHAR(20)    ,
  StaffForename VARCHAR(45)  NOT NULL  ,
  StaffSurname VARCHAR(255)  NOT NULL  ,
  Extension INTEGER      ,
PRIMARY KEY(id)  ,
  FOREIGN KEY(StaffType_id)
    REFERENCES StaffType(id));
GO


CREATE INDEX Staff_FKIndex1 ON Staff (StaffType_id);
GO


CREATE INDEX IFK_StaffRole ON Staff (StaffType_id);
GO


CREATE TABLE Client (
  id INTEGER  NOT NULL   IDENTITY ,
  ClientType_id INTEGER  NOT NULL  ,
  ClientForename VARCHAR(45)  NOT NULL  ,
```

```
  ClientSurname VARCHAR(255)  NOT NULL  ,
  AccountNumber VARCHAR(20)  NOT NULL  ,
  MobilePhone VARCHAR(20)      ,
PRIMARY KEY(id)  ,
  FOREIGN KEY(ClientType_id)
    REFERENCES ClientType(id));
GO


CREATE INDEX Client_FKIndex1 ON Client (ClientType_id);
GO


CREATE INDEX IFK_ClientStatus ON Client (ClientType_id);
GO


CREATE TABLE ClientAddress (
  id INTEGER  NOT NULL   IDENTITY ,
  Client_id INTEGER  NOT NULL  ,
  Address1 VARCHAR(255)    ,
  Address2 VARCHAR(255)    ,
  Town VARCHAR(255)    ,
  County VARCHAR(255)    ,
  Postcode VARCHAR(10)  ,
  Landline VARCHAR(20)      ,
PRIMARY KEY(id)  ,
  FOREIGN KEY(Client_id)
    REFERENCES Client(id));
GO


CREATE INDEX ClientAddress_FKIndex1 ON ClientAddress (Client_id);
GO


CREATE INDEX IFK_ClientLocation ON ClientAddress (Client_id);
GO


CREATE TABLE Model (
  id INTEGER  NOT NULL   IDENTITY ,
  Manufacturer_id INTEGER  NOT NULL  ,
  ModelName VARCHAR(255)  NOT NULL    ,
PRIMARY KEY(id)  ,
  FOREIGN KEY(Manufacturer_id)
    REFERENCES Manufacturer(id));
GO


CREATE INDEX Model_FKIndex1 ON Model (Manufacturer_id);
GO


CREATE INDEX IFK_Makes ON Model (Manufacturer_id);
GO


CREATE TABLE Vehicle (
  id INTEGER  NOT NULL   IDENTITY ,
  VehicleType_id INTEGER  NOT NULL  ,
  Model_id INTEGER  NOT NULL  ,
  Registration VARCHAR(20)  NOT NULL  ,
  DateOnFleet DATE    ,
  Gearbox VARCHAR(20)    ,
  MaxLadenWeight VARCHAR(20)      ,
PRIMARY KEY(id)    ,
  FOREIGN KEY(Model_id)
    REFERENCES Model(id),
  FOREIGN KEY(VehicleType_id)
    REFERENCES VehicleType(id));
```

```
GO


CREATE INDEX Vehicle_FKIndex1 ON Vehicle (Model_id);
GO
CREATE INDEX Vehicle_FKIndex2 ON Vehicle (VehicleType_id);
GO


CREATE INDEX IFK_ModelVehicle ON Vehicle (Model_id);
GO
CREATE INDEX IFK_TypeOfVehicle ON Vehicle (VehicleType_id);
GO


CREATE TABLE Rental (
  id INTEGER  NOT NULL   IDENTITY ,
  Tariff_id INTEGER  NOT NULL  ,
  Client_id INTEGER  NOT NULL  ,
  Staff_id INTEGER  NOT NULL  ,
  Vehicle_id INTEGER  NOT NULL  ,
  RentalDate DATE  NOT NULL  ,
  PeriodDays INTEGER  NOT NULL    ,
PRIMARY KEY(id)        ,
  FOREIGN KEY(Vehicle_id)
    REFERENCES Vehicle(id),
  FOREIGN KEY(Staff_id)
    REFERENCES Staff(id),
  FOREIGN KEY(Client_id)
    REFERENCES Client(id),
  FOREIGN KEY(Tariff_id)
    REFERENCES Tariff(id));
GO


CREATE INDEX Rental_FKIndex1 ON Rental (Vehicle_id);
GO
CREATE INDEX Rental_FKIndex2 ON Rental (Staff_id);
GO
CREATE INDEX Rental_FKIndex3 ON Rental (Client_id);
GO
CREATE INDEX Rental_FKIndex4 ON Rental (Tariff_id);
GO


CREATE INDEX IFK_VehicleRental ON Rental (Vehicle_id);
GO
CREATE INDEX IFK_StaffRental ON Rental (Staff_id);
GO
CREATE INDEX IFK_ClientRental ON Rental (Client_id);
GO
CREATE INDEX IFK_RentalRate ON Rental (Tariff_id);
GO
```

```
CREATE TABLE ClientType (
  id INTEGER  NOT NULL   IDENTITY ,
  ClientTypeName VARCHAR(45)  NOT NULL    ,
PRIMARY KEY(id));
GO



CREATE TABLE Manufacturer (
  id INTEGER  NOT NULL   IDENTITY ,
  ManufacturerName VARCHAR(45)  NOT NULL    ,
PRIMARY KEY(id));
GO



CREATE TABLE Model (
  id INTEGER  NOT NULL   IDENTITY ,
  Manufacturer_id INTEGER  NOT NULL  ,
  ModelName VARCHAR(255)  NOT NULL    ,
PRIMARY KEY(id)  ,
  FOREIGN KEY(Manufacturer_id)
    REFERENCES Manufacturer(id));
GO


CREATE INDEX Model_FKIndex1 ON Model (Manufacturer_id);
GO


CREATE INDEX IFK_ManufacturerModel ON Model (Manufacturer_id);
GO


CREATE TABLE Vehicle (
  id INTEGER  NOT NULL   IDENTITY ,
  Model_id INTEGER  NOT NULL  ,
  Registration VARCHAR(20)  NOT NULL  ,
  DateOnFleet DATE  NOT NULL  ,
  Gearbox VARCHAR(20)    ,
  MaxLadenWeight INTEGER      ,
PRIMARY KEY(id)  ,
  FOREIGN KEY(Model_id)
    REFERENCES Model(id));
GO


CREATE INDEX Vehicle_FKIndex1 ON Vehicle (Model_id);
GO


CREATE INDEX IFK_ModelVehicle ON Vehicle (Model_id);
GO


CREATE TABLE Client (
  id INTEGER  NOT NULL   IDENTITY ,
  ClientType_id INTEGER  NOT NULL  ,
  ClientForename VARCHAR(255)  NOT NULL  ,
  ClientSurname VARCHAR(255)  NOT NULL  ,
  AccountNumber VARCHAR(20)  NOT NULL  ,
  MobilePhone VARCHAR(20)      ,
PRIMARY KEY(id)  ,
  FOREIGN KEY(ClientType_id)
    REFERENCES ClientType(id));
GO


CREATE INDEX Client_FKIndex1 ON Client (ClientType_id);
```

```
GO


CREATE INDEX IFK_ClientTypeClient ON Client (ClientType_id);
GO


CREATE TABLE ClientAddress (
  id INTEGER  NOT NULL   IDENTITY ,
  Client_id INTEGER  NOT NULL  ,
  AddressLine1 VARCHAR(255)  NOT NULL  ,
  AddressLine2 VARCHAR(255)    ,
  Town VARCHAR(255)  NOT NULL  ,
  County VARCHAR(255)    ,
  Postcode VARCHAR(20)  NOT NULL  ,
  Landline VARCHAR(20)      ,
PRIMARY KEY(id)  ,
  FOREIGN KEY(Client_id)
    REFERENCES Client(id));
GO


CREATE INDEX ClientAddress_FKIndex1 ON ClientAddress (Client_id);
GO


CREATE INDEX IFK_ClientClientAddress ON ClientAddress (Client_id);
GO
```

```
--BULK IMPORT
BULK
INSERT Manufacturer
FROM 'C:\Rental\Manufacturer.txt'
WITH
(
FIELDTERMINATOR = ',',
ROWTERMINATOR = '\n'
)
GO


--Check
SELECT * FROM Manufacturer
GO
```

```
-- ************************************
-- Author:      Paul McKillop
-- Create Date: 10 November 2018
-- Description: Get all ClientType records
-- ************************************
Use [Rental]

GO

CREATE PROC spClientType_GetAll

AS

BEGIN
    SELECT id,
    ClientTypeName

    FROM ClientType
END
```

```sql
-- ************************************
-- Author:      Paul McKillop
-- Create Date: 19 January 2019
-- Description: Update a client record by the ID of the Client
-- ************************************
Use [Rental]

GO

CREATE PROC spClient_Delete
    @ClientID int

AS
BEGIN
    DELETE
    FROM Client
    WHERE id = @ClientID
END
```

```
Use [Rental]

GO

CREATE PROC spClient_GetAll

AS

BEGIN
    SELECT
        ClientType_id,
        ClientForename,
        ClientSurname,
        AccountNumber,
        MobilePhone

        FROM

        Client
END
```

```
-- ************************************
-- Author:      Paul McKillop
-- Create Date: 19 January 2019
-- Description: Get Clients by ClientType
-- ************************************
Use [Rental]

GO

CREATE PROC spClient_GetByClientTypeID
    @ClientTypeID int
AS

BEGIN
    SELECT
        id,
        ClientType_id,
        ClientForename,
        ClientSurname,
        AccountNumber,
        MobilePhone

        FROM

        Client

        WHERE
        ClientType_id = @ClientTypeID
END
```

```
-- ************************************
-- Author:      Paul McKillop
-- Create Date: 07 December 2018
-- Description: Get Client by their ID
-- ************************************
Use [Rental]

GO

CREATE PROC spClient_GetByID
    @ClientID int
AS

BEGIN
    SELECT
        id,
        ClientType_id,
        ClientForename,
        ClientSurname,
        AccountNumber,
        MobilePhone

        FROM

        Client

        WHERE
        id = @ClientID
END
```

```
-- ************************************
-- Author:      Paul McKillop
-- Create Date: 08 January 2019
-- ************************************
Use [Rental]

GO

CREATE PROCEDURE spClient_Insert
(
    @ClientID int OUTPUT,
    @ClientType_id int,
    @ClientForename varchar(255),
    @ClientSurname varchar(255),
    @AccountNumber varchar(20),
    @MobilePhone varchar(20)
)

AS
BEGIN
INSERT INTO Client (
    ClientType_id,
    ClientForename,
    ClientSurname,
    AccountNumber,
    MobilePhone)

    VALUES (
    @ClientType_id,
    @ClientForename,
    @ClientSurname,
    @AccountNumber,
    @MobilePhone
    )

    SET @ClientID = SCOPE_IDENTITY()
END
```

```
-- ************************************
-- Author:      Paul McKillop
-- Create Date: 03 December 2018
-- Description: Update a client record by the ID of the Client
-- ************************************
Use [Rental]

GO

CREATE PROCEDURE spClient_Update
(
    @CustomerID
    @ClientType_id int,
    @ClientForename varchar(255),
    @ClientSurname varchar(255),
    @AccountNumber varchar(20),
    @MobilePhone varchar(20)
)

AS
BEGIN
UPDATE Client (

SET
    ClientType_id = @ClientType_id,
    ClientForename = @ClientForename,
    ClientSurname = @ClientSurname,
    AccountNumber = @AccountNumber,
    MobilePhone = @MobilePhone

WHERE id = CustomerID

END
```

```
-- ************************************
-- Author:      Paul McKillop
-- Create Date: 06 December 2018
-- Description: Return all records of Manufacturer, all fields
-- ************************************

Use [Rental]

GO

CREATE PROC spManufacturer_GetAll

AS

BEGIN

    SET NOCOUNT ON;

    SELECT id,
    ManufacturerName

    FROM Manufacturer;
END

GO
```

```
-- ************************************
-- Author:      Paul McKillop
-- Create Date: 03 December 2018
-- Description: Return all records of Model for a particular Manufacturer,
-- all fields returned
-- ************************************

Use [Rental]

GO

CREATE PROC spModel_GetByManufacturerID
    @ManufacturerID int

AS

BEGIN
    SELECT id,
    ModelName

    FROM Model

    WHERE (Manufacturer_id = @ManufacturerID)
END
```

```
-- ************************************
-- Author:      Paul McKillop
-- Create Date: 03 January 2019
-- Description: Get all VehicleType records
-- ************************************
Use [Rental]

GO

CREATE PROC spVehicleType_GetAll

AS

BEGIN
    SELECT id,
    VehicleTypeName

    FROM dbo.VehicleType
```

```
-- ************************************
-- Author:      Paul McKillop
-- Create Date: 03 January 2019
-- Description: Get all Vehicle records
-- ************************************

USE [Rental]

GO
CREATE PROC dbo.spVehicle_GetAll

AS

BEGIN
    SELECT id, Registration, DateOnFleet, Gearbox, MaxLadenWeight
    FROM dbo.Vehicle
END

GO
```

```
-- ************************************
-- Author:      Paul McKillop
-- Create Date: 03 January 2019
-- Description: Insert a new vehicle record
-- ************************************

USE [Rental]

GO

CREATE PROCEDURE spVehicle_Insert

(
    @VehicleID int OUTPUT,
    @Model_id int,
    @Registration varchar(20),
    @DateOnFleet date,
    @Gearbox varchar(20),
    @MaxLadenWeight int
)

AS

BEGIN
    INSERT INTO Vehicle (
        Model_id,
        Registration,
        DateOnFleet,
        Gearbox,
        MaxLadenWeight
    )

    VALUES (
    @Model_id,
    @Registration,
    @DateOnFleet,
    @Gearbox,
    @MaxLadenWeight
    )

    SET @VehicleID = SCOPE_IDENTITY()

END


EXEC dbo.spVehicle_Insert 0,3,'HW67 AAA','2018-03-03','NA',1800
```

```
1,Private on account
2,Private Pay Advance
3,Commercial on Account
4,Commercial Pay Advance
```

```
1,Ford
2,General Motors
3,Toyota
```

```
1,1,Mondeo
2,1,Fiesta
3,1,Transit
4,2,GM Saloon
5,3,Hi-Lux
6,3,Avensis
```

```
1,1,SMIJ01,John,Smith,2200
2,2,BROM01,Mary,Brown,2201
2,1,VANM01,Mariam,Vance,2202
4,2,REYP01,Reynolds,Philip,2203
```

```
1,Administrator
2,Booking clerk
```

```
1,Tariff A,25
2,Tariff B,30
3,Tariff C,35
4,Tariff D,40
5,Tariff E,45
6,Tariff F,50
7,Tariff G,55
```

```
1,Commercial
2,Car
```

PageClientUpdate.pdf

PageClientUpdate.xaml.pdf

PageDashboard.pdf

PageDashboard.xaml.pdf

PageHelp.pdf

PageHelp.xaml.pdf

PageLogin.pdf

PageLogin.xaml.pdf

PageVehicle.pdf

PageVehicle.xaml.pdf

PageVehicleCreate.pdf

PageVehicleCreate.xaml.pdf

Authenticate.pdf

DBScript.pdf

RentalCreateScript.pdf

RentalImportScript.pdf

spClientType_GetAll.pdf

spClient_Delete.pdf

spClient_GetAll.pdf

spClient_GetByClientTypeID.pdf

spClient_GetByID.pdf

spClient_Insert.pdf

spClient_Update.pdf

spManufacturer_GetAll.pdf

spModel_GetByManufacturerID.pdf

spVehicleType_GetAll.pdf

spVehicle_GetAll.pdf

spVehicle_Insert.pdf

ClientType.pdf

00 Title Page.pdf

ClientTypeDB.pdf

ImportData.pdf

ManufacturerDB.pdf

ModelDB.pdf

PersonDB.pdf

VehicleDB.pdf

Client.pdf

ClientAddress.pdf

ClientBySurname.pdf

ClientType.pdf

Manufacturer.pdf

Model.pdf

Person.pdf

Staff.pdf

User.pdf

Vehicle.pdf

VehicleCommercial.pdf

VehicleDomestic.pdf

MainWindow.pdf

MainWindow.xaml.pdf

PageClient.pdf

PageClient.xaml.pdf

PageClientAddress.pdf

PageClientAddress.xaml.pdf

PageClientCreate.pdf

PageClientCreate.xaml.pdf

PageClientDelete.pdf

PageClientDelete.xaml.pdf

Manufacturer.pdf

Model.pdf

Staff.pdf

StaffType.pdf

Tariff.pdf

VehicleType.pdf