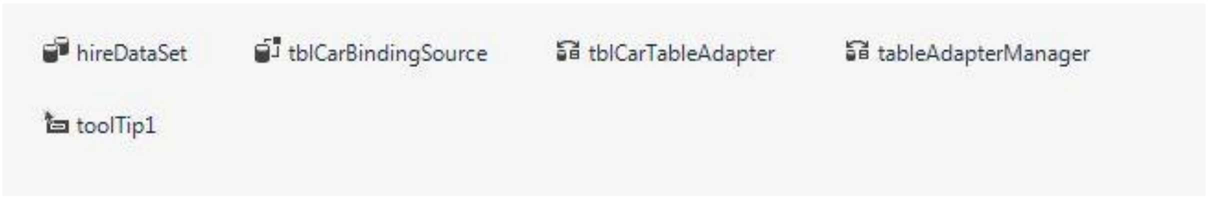
Technical Documentation – Joe Marsh

Purpose of the software:

The purpose of the application is to provide a GUI frontend for a database of cars for a car hire company which will allow them to add, remove, update and search the database and provide information on Vehicle registration, make, engine size, registration date, rental cost and availability.

Connection details:

frmCars database connection



hireDataSet

Datasets are objects that contain data tables where you can temporarily store the data for use in your application. If your application requires working with data, you can load the data into a dataset, which provides your application with a local in-memory cache of the data to work with. You can work with the data in a dataset even if your application becomes disconnected from the database. The dataset maintains information about changes to its data so updates can be tracked and sent back to the database when your application becomes reconnected.

tblCarBindingSourse

The BindingSource component serves many purposes. First, it simplifies binding controls on a form to data by providing currency management, change notification, and other services between Windows Forms controls and data sources.

BindingSource provides members for accessing the underlying data. The current item can be retrieved through the Current property, and the entire list can be retrieved through the List property. Editing operations are supported on the current item through Current and the RemoveCurrent, EndEdit, CancelEdit and Add and AddNew methods.

Data sources that are bound to a BindingSource component can also be navigated and managed with the BindingNavigator class, which provides a VCR-like user interface (UI) for navigating items within a list. E.g. tblCarBindingSource.MoveNext();

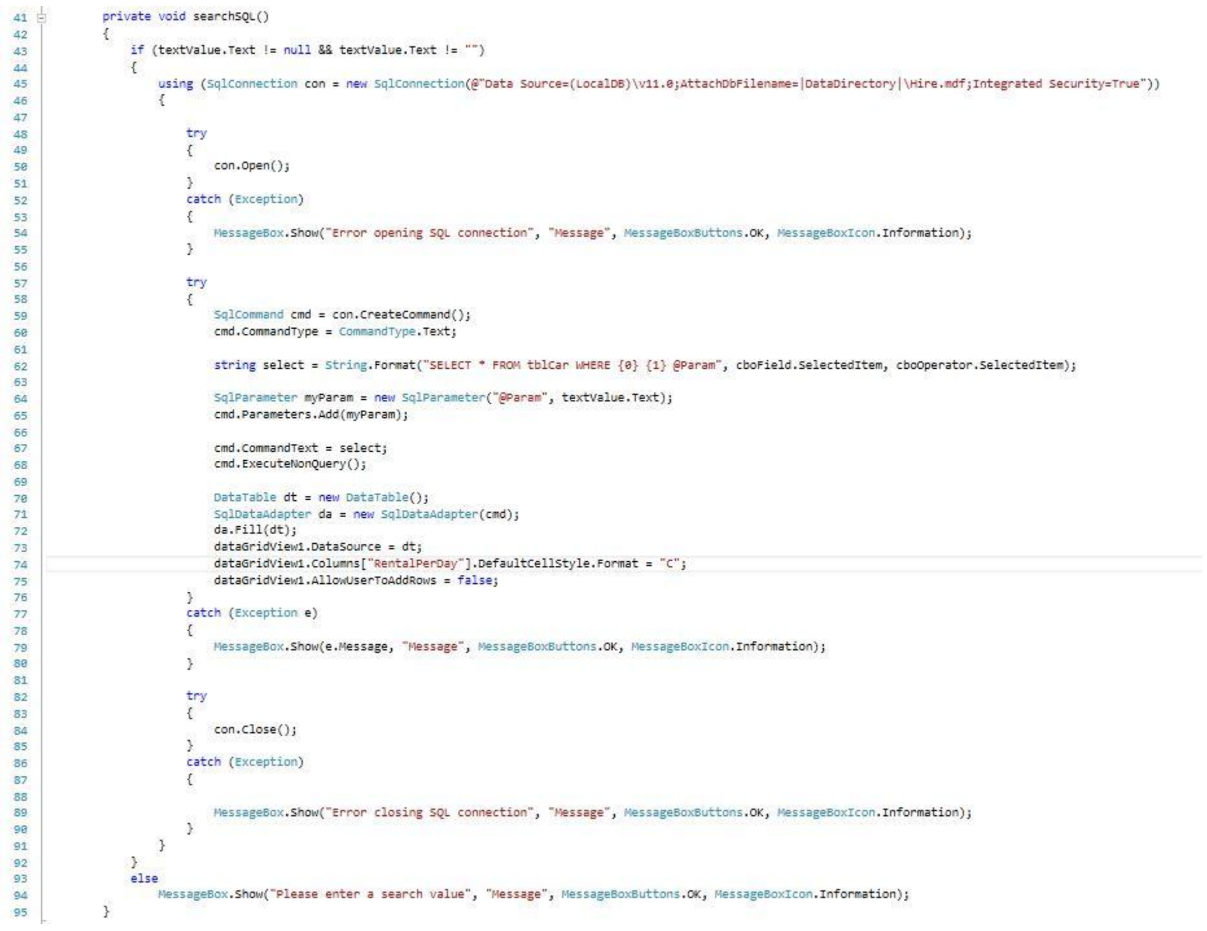
tblCarTableAdapter

TableAdapters provide communication between your application and a database. More specifically, a TableAdapter connects to a database, executes queries or stored procedures, and either returns a new data table populated with the returned data or fills an existing DataTable with the returned data. TableAdapters are also used to send updated data from your application back to the database.

tableAdapterManager

The TableAdapterManager is a component that provides the functionality to save data in related data tables. The TableAdapterManager uses the foreign-key relationships that relate data tables to determine the correct order to send the Inserts, Updates, and Deletes from a dataset to the database without violating the foreign-key constraints (referential integrity) in the database.

frmSearch database connection:



```
        private void searchSQL()
        {
            if (textValue.Text != null && textValue.Text != "")
            {
                using (SqlConnection con = new SqlConnection(@"Data Source=(LocalDB)\v11.0;AttachDbFilename=|DataDirectory|\Hire.mdf;Integrated Security=True"))
                {

                    try
                    {
                        con.Open();
                    }
                    catch (Exception)
                    {
                        MessageBox.Show("Error opening SQL connection", "Message", MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }

                    try
                    {
                        SqlCommand cmd = con.CreateCommand();
                        cmd.CommandType = CommandType.Text;

                        string select = String.Format("SELECT * FROM tblCar WHERE {0} {1} @Param", cboField.SelectedItem, cboOperator.SelectedItem);

                        SqlParameter myParam = new SqlParameter("@Param", textValue.Text);
                        cmd.Parameters.Add(myParam);

                        cmd.CommandText = select;
                        cmd.ExecuteNonQuery();

                        DataTable dt = new DataTable();
                        SqlDataAdapter da = new SqlDataAdapter(cmd);
                        da.Fill(dt);
                        dataGridView1.DataSource = dt;
                        dataGridView1.Columns["RentalPerDay"].DefaultCellStyle.Format = "C";
                        dataGridView1.AllowUserToAddRows = false;
                    }
                    catch (Exception e)
                    {
                        MessageBox.Show(e.Message, "Message", MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }

                    try
                    {
                        con.Close();
                    }
                    catch (Exception)
                    {

                        MessageBox.Show("Error closing SQL connection", "Message", MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                }
            }
            else
                MessageBox.Show("Please enter a search value", "Message", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
```

SqlConnection

A SqlConnection object represents a unique session to a SQL Server data source. With a client/server database system, it is equivalent to a network connection to the server. SqlConnection is used together with SqlDataAdapter and SqlCommand to increase performance when connecting to a Microsoft SQL Server database.

SqlCommand

Represents a Transact-SQL statement or stored procedure to execute against a SQL Server database.

SqlDataAdapter

The SqlDataAdapter, serves as a bridge between a DataSet and SQL Server for retrieving and saving data. The SqlDataAdapter provides this bridge by mapping Fill, which changes the data in the DataSet to match the data in the data source, and Update, which changes the data in the data source to match the data in the DataSet, using the appropriate Transact-SQL statements against the data source.

DataTable

Represents one table of in-memory data.

Cars Database Testing – Joe Marsh

Test 1: Add Button - frmCars

Expected Result:

Clicking on the Add button should add a new row/entry to the car dataset and display a new entry in the data fields to the left that the user can fill in with the car details they wish to include.

Actual Result:

Pass - The test performed as expected.

Test 2: Update Button - frmCars

Expected Result:

Any changes made to the car dataset should be saved to the database so that they persist when the program is closed/opened. A message box should appear to confirm that the data has been successfully updated.

Actual Result:

Pass - The test performed as expected.

Test 3: Delete Button - frmCars

Expected Result:

The currently selected record/row in the dataset should be removed and the next record displayed in the data fields. If it is the last row being removed then the previous record will be displayed instead.

Actual Result:

Pass - The test performed as expected.

Test 4: Cancel Button - frmCars

Expected Result:

Using the Cancel button should undo any changes made to the dataset that have not yet been committed to the database (Update button). For example any changed made with Add, Cancel or individual edits to the data set fields will be reverted as long as the changes have not yet been saved using the Update button.

Actual Result:

Pass - The test performed as expected.

Test 5: Search Button - frmCars

Expected Result:

Clicking the Search button should open the form frmSearch.

Actual Result:

Pass - The test performed as expected.

Test 6: Run Button - frmSearch

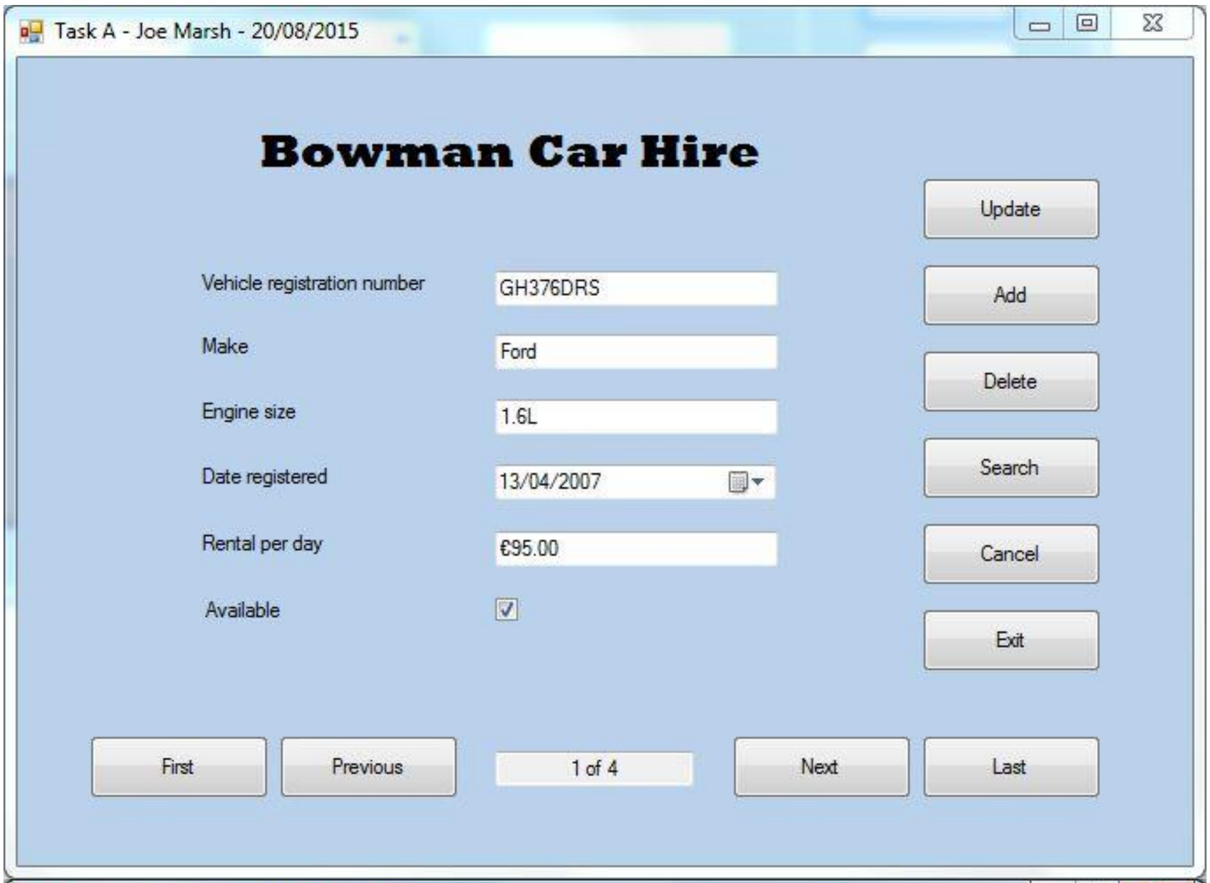
Expected Result:

Display a table of results based on search criteria selected by the user. For example selecting "Make" from the Field dropdown menu, "=" from the Operator dropdown menu and entering "Ford" in the Value text box should display a list of any Ford cards from the database.

Actual Result:

Pass - The test performed as expected.

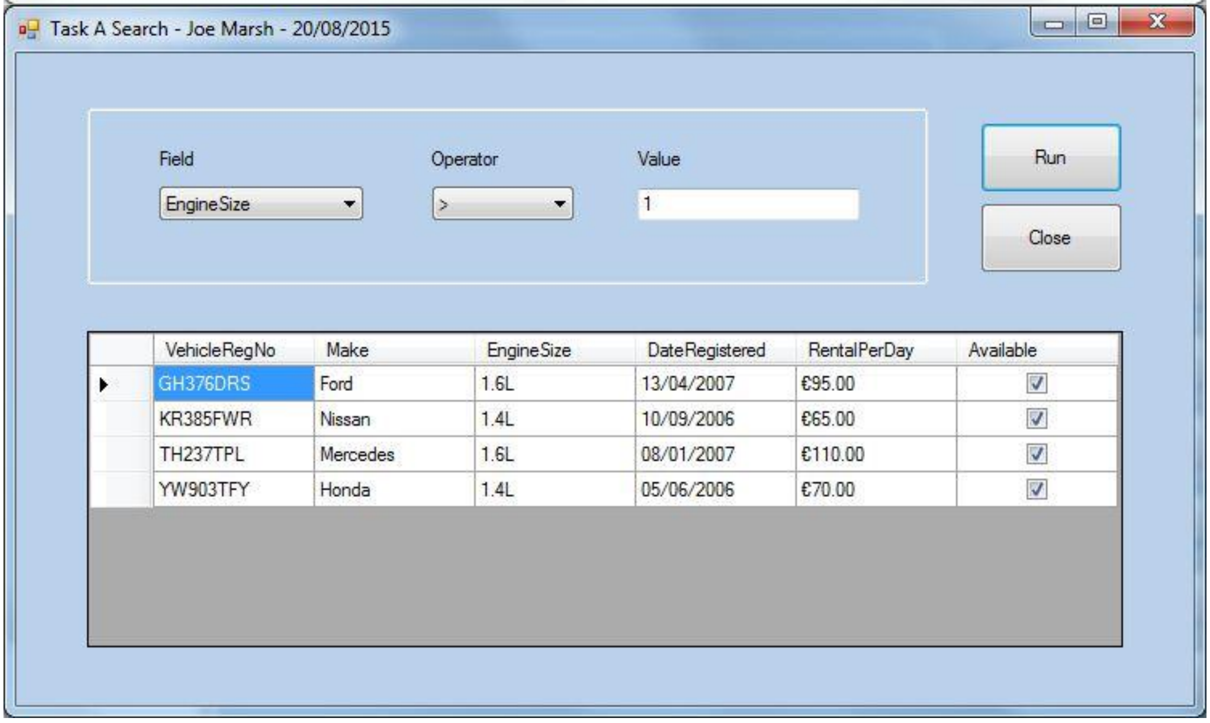
Screen Prints & Program Listing – Joe Marsh



Bowman Car Hire

Vehicle registration number	GH376DRS
Make	Ford
Engine size	1.6L
Date registered	13/04/2007
Rental per day	€95.00
Available	✓

1 of 4



Field	Operator	Value
EngineSize	>	1

VehicleRegNo	Make	EngineSize	DateRegistered	RentalPerDay	Available
GH376DRS	Ford	1.6L	13/04/2007	€95.00	✓
KR385FWR	Nissan	1.4L	10/09/2006	€65.00	✓
TH237TPL	Mercedes	1.6L	08/01/2007	€110.00	✓
YW903TFY	Honda	1.4L	05/06/2006	€70.00	✓

frmCars Code

```
public frmCars()
        {
            InitializeComponent();
        }

        private void tblCarBindingNavigatorSaveItem_Click(object sender, EventArgs e)
        {
            this.Validate();
            this.tblCarBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.hireDataSet);
        }

        private void tblCarBindingNavigatorSaveItem_Click_1(object sender, EventArgs
e)
        {
            this.Validate();
            this.tblCarBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.hireDataSet);
        }

        private void frmCars_Load(object sender, EventArgs e)
        {
            this.tblCarTableAdapter.Fill(this.hireDataSet.tblCar);
            position();
        }

        private void button9_Click(object sender, EventArgs e)
        {
            tblCarBindingSource.MoveNext();
            position();
        }

        private void button8_Click(object sender, EventArgs e)
        {
            tblCarBindingSource.MovePrevious();
            position();
        }

        private void button7_Click(object sender, EventArgs e)
        {
            tblCarBindingSource.MoveFirst();
            position();
        }

        private void button10_Click(object sender, EventArgs e)
        {
            tblCarBindingSource.MoveLast();
            position();
        }

        private void position()
        {
            textCounter.Text = ((tblCarBindingSource.Position + 1) + " of " +
tblCarBindingSource.Count);
        }

        private void btnUpdate_Click(object sender, EventArgs e)
        {
            try
            {
                if (textReg.Text != "")
                {
                    this.Validate();
                    //hireDataSet.AcceptChanges();
                    this.tblCarBindingSource.EndEdit();
                    this.tblCarTableAdapter.Update(this.hireDataSet.tblCar);
                    MessageBox.Show("Record Updated", "Message", MessageBoxButtons.OK, 
MessageBoxIcon.Information);
                }
                else
                {
                    MessageBox.Show("Registration field must contain a value", 
"Message", MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Some Error in Update: " + ex.Message, "Message", 
MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            try
            {
                this.Validate();
                this.tblCarBindingSource.EndEdit();

                this.hireDataSet.tblCar.AddtblCarRow("REG", null, null, DateTime.Now, 
0, false);

                this.tblCarBindingSource.MoveLast();
                position();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error Adding Row", "Message", MessageBoxButtons.OK, 
MessageBoxIcon.Information);
            }
        }

        private void btnDelete_Click(object sender, EventArgs e)
        {
            try
            {
                this.Validate();
                this.tblCarBindingSource.EndEdit();

                hireDataSet.tblCar.FindByVehicleRegNo(textReg.Text).Delete();

                position();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error Deleting Row", "Message", MessageBoxButtons.OK, 
MessageBoxIcon.Information);
            }
        }

        private void btnCancel_Click(object sender, EventArgs e)
        {
            try
            {
                hireDataSet.RejectChanges();
                tblCarBindingSource.ResetBindings(false);
                position();
            }
            catch (Exception x)
            {
                MessageBox.Show(x.Message, "Message", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            }
        }

        private void btnExit_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void btnSearch_Click(object sender, EventArgs e)
        {
            frmSearch frm = new frmSearch();
            frm.Show();
        }

    }
```

frmSearch Code

```
    public partial class frmSearch : Form
    {
        public frmSearch()
        {
            InitializeComponent();
        }

        private void frmSearch_Load(object sender, EventArgs e)
        {
            cboField.Items.Add("Make");
            cboField.Items.Add("EngineSize");
            cboField.Items.Add("RentalPerDay");
            cboField.Items.Add("Available");
            cboField.SelectedIndex = 0;
            cboField.DropDownStyle = ComboBoxStyle.DropDownList;

            cboOperator.Items.Add('=');
            cboOperator.Items.Add('<');
            cboOperator.Items.Add('>');
            cboOperator.Items.Add("<=");
            cboOperator.Items.Add(">=");
            cboOperator.SelectedIndex = 0;
            cboOperator.DropDownStyle = ComboBoxStyle.DropDownList;
        }

        private void searchSQL2()
        {
            SqlConnection con = new SqlConnection(@"Data 
Source=(LocalDB)\v11.0;AttachDbFilename=|DataDirectory|\Hire.mdf;Integrated 
Security=True");

            string select = String.Format("SELECT * FROM tblCar WHERE {0} {1} '{2}'", 
cboField.Text, cboOperator.Text, textValue.Text);

            SqlDataAdapter da = new SqlDataAdapter(select, con);

            DataTable dt = new DataTable();
            da.Fill(dt);
            dataGridView1.DataSource = dt;
        }

        private void searchSQL()
        {
            if (textValue.Text != null && textValue.Text != "")
            {
                using (SqlConnection con = new SqlConnection(@"Data 
Source=(LocalDB)\v11.0;AttachDbFilename=|DataDirectory|\Hire.mdf;Integrated 
Security=True"))
                {

                    try
                    {
                        con.Open();
                    }
                    catch (Exception)
                    {
                        MessageBox.Show("Error opening SQL connection", "Message", 
MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }

                    try
                    {
                        SqlCommand cmd = con.CreateCommand();
                        cmd.CommandType = CommandType.Text;


                        if (textValue.Text.ToUpper() == "YES")
                            textValue.Text = "1";
                        else if (textValue.Text.ToUpper() == "NO")
                            textValue.Text = "0";

                        string select = String.Format("SELECT * FROM tblCar WHERE {0} 
{1} @Param", cboField.SelectedItem, cboOperator.SelectedItem);

                        SqlParameter myParam = new SqlParameter("@Param", 
textValue.Text);
                        cmd.Parameters.Add(myParam);

                        cmd.CommandText = select;
                        cmd.ExecuteNonQuery();

                        DataTable dt = new DataTable();
                        SqlDataAdapter da = new SqlDataAdapter(cmd);
                        da.Fill(dt);
                        dataGridView1.DataSource = dt;
                        dataGridView1.Columns["RentalPerDay"].DefaultCellStyle.Format 
= "C";

                        dataGridView1.AllowUserToAddRows = false;
                    }
                    catch (Exception e)
                    {
                        MessageBox.Show(e.Message, "Message", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                    }

                    try
                    {
                        con.Close();
                    }
                    catch (Exception)
                    {

                        MessageBox.Show("Error closing SQL connection", "Message",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                }
            }
            else
                MessageBox.Show("Please enter a search value", "Message",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        }

        private void btnRun_Click(object sender, EventArgs e)
        {
            searchSQL();
        }

        private void btnClose_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
```