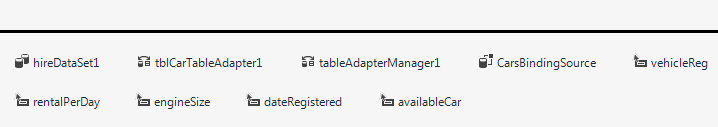
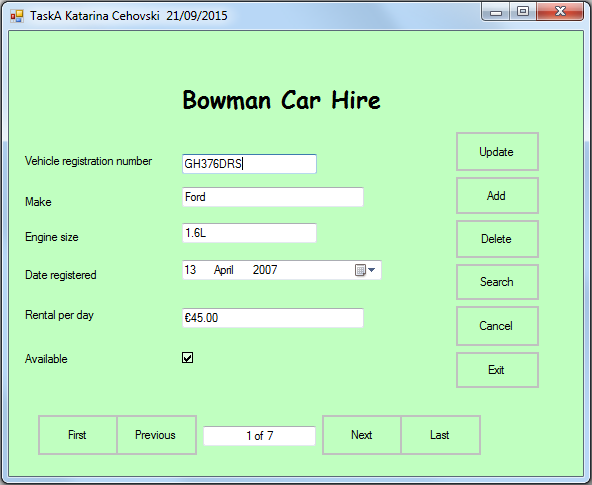
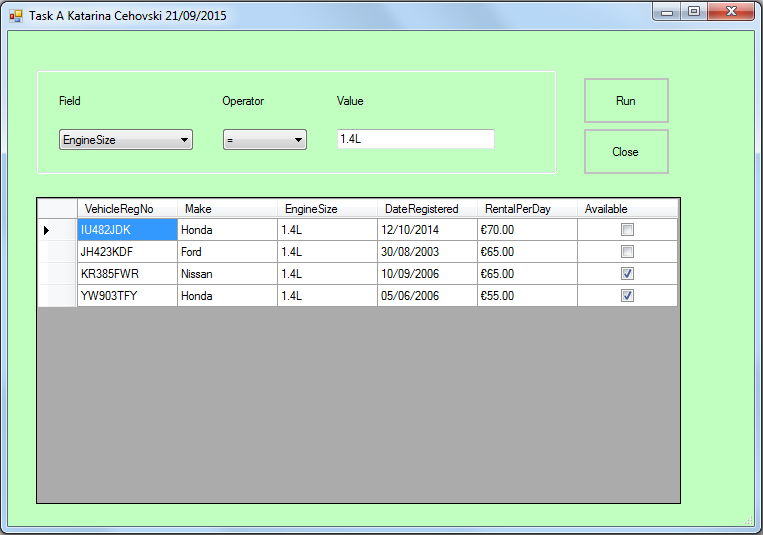
KATARINA CEHOVSKI  
ASSIGNMENT A TASK B

PART 1: TEST PLAN AND TEST LOG  
  
**Type of Test:**  
Validation of Update button.  
**Expected Result:**  
When you click the Update button should assign any changes of a record to the table.  
**Actual Result:**  
It passes the test.  
  
**Type of Test:**  
Validation of Add button.  
**Expected Result:**  
When you click the Add button should add a new row/record.   
**Actual Result:**  
It passes the test.  
  
**Type of Test:**Validation of Delete button.  
**Expected Result:**When you click the Delete button should delete the record.   
**Actual Result:**  
It passes the test.  
  
**Type of Test:**Validation of Cancel button.  
**Expected Result:**  
When you click the Cancel button should disregard any changes made to a record.  
**Actual Result:**  
It passes the test.  
  
**Type of Test:**  
Validation of Search button.  
**Expected Result:**   
When you click the Search button should open the form frmSearch.  
**Actual Result:**It passes the test.  
  
**Type of Test:**Validation of Run button.  
**Expected Result:**  
When you click the Run button should execute query that match criteria entered using combo boxes and the value in the data entry text box. All fields from table tblCar for all the records which match the criteria should be displayed in the data grid. The search should be run only if data exists in all three controls. A criteria string that is not matched by any record should return nothing.  
Error messages should show when:  
a) there is no match,  
b) customer didn’t enter value in text box,  
c) there is SQL Error  
**Actual Result:**It passes the test.  
  
  
PART 2:   
  
  
  
hireDataSet1   
Datasets are objects that contain data tables where you can temporarily store the data for use in your application.  
  
tblCarTableAdapter1  
TableAdapters provide communication between your application and a database.  
  
tableAdapterManager1  
The TableAdapterManager is a component that provides the functionality to save data in related data tables.   
  
CarsBindingSource  
It supports the binding of control elements in a form, and can be seen as a link between a *data source* and a control element.   
  
Purpose of the software is to allow clients to access an external database. Client can:  
- display individual records  
- add a new record  
- delete a record  
- edit a record  
- update a record  
- cancel amendments for a record  
- search records

PART 3:  
  




Code for frmCars  
  
using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Globalization;

namespace CarsDatabase

{

public partial class frmCars : Form

{

public frmCars()

{

InitializeComponent();

}

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

{

//When form frmCars is loaded the dataset is loaded automatically and the data for the first record is displayed in the controls.

tblCarTableAdapter1.Fill(hireDataSet1.tblCar);

updatePosition();

}

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

{

try

{

//Checking that text box txtVehicleReg is not empty.  
 //Updating all changes.

if (txtVehicleReg.Text != "")

{

this.Validate();

this.CarsBindingSource.EndEdit();

this.tblCarTableAdapter1.Update(this.hireDataSet1.tblCar);

MessageBox.Show("Update successful");

}

//If text box txtVehicleReg is empty this message appear on screen.

else

{

MessageBox.Show("Vehicle Registration Number can't be empty");

}

}

//If there is already row with the same value for column VehicleRegNo this message appear on screen.

//Column VehicleRegNo is a primary key and it must be unique.

catch (ConstraintException)

{

MessageBox.Show("That Vehicle Registration Number already exists");

}

}

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

{

//Populates all text boxes, check box and data time picker with next record and also updates text box txtOrder to current position.

CarsBindingSource.MoveNext();

updatePosition();

}

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

{

//Populates all text boxes, check box and data time picker with previous record and also updates text box txtOrder to current position.

CarsBindingSource.MovePrevious();

updatePosition();

}

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

{

//Populates all text boxes, check box and data time picker with first record and also updates text box txtOrder to current position.

CarsBindingSource.MoveFirst();

updatePosition();

}

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

{

//Populates all text boxes, check box and data time picker with last record and also updates text box txtOrder to current position.

CarsBindingSource.MoveLast();

updatePosition();

}

private void updatePosition()

{

//Populates text box txtOrder with current and total record number.

txtOrder.Text = CarsBindingSource.Position +1 + " of " + CarsBindingSource.Count;

}

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

{

//Closes the program.

this.Close();

}

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

{

//Opens a form frmSearch.

frmSearch searchForm = new frmSearch();

searchForm.ShowDialog();

}

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

{

try

{

//Adding a new row.

//Populating column VehicleRegNo because it can't be empty, and column Available with 0 (False) because it's by default set to True.

DataRow row = hireDataSet1.tblCar.NewRow();

row["VehicleRegNo"] = "Regis";

row["Available "] = 0;

hireDataSet1.tblCar.Rows.Add(row);

//Row is added to last place, so this is going to show that record.

CarsBindingSource.MoveLast();

updatePosition();

}

//If there is already row with the same value for column VehicleRegNo this message appear on screen.

//Column VehicleRegNo is a primary key and it must be unique.

catch (ConstraintException)

{

MessageBox.Show("Vehicle Registration Number 'Regis' already exist");

}

}

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

{

//Deletes current record and shows next record with updated position.

//If you are sure you want to delete that record then it's necessary to click Update button.

hireDataSet1.tblCar[CarsBindingSource.Position].Delete();

updatePosition();

}

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

{

//Cancels all changes.

hireDataSet1.RejectChanges();

CarsBindingSource.ResetBindings(false);

updatePosition();

}

}

}

Code for frmSearch

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace CarsDatabase

{

public partial class frmSearch : Form

{

public frmSearch()

{

InitializeComponent();

}

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

{

//When form frmSearch is loaded combo boxes cboField and cboOperator are going to be populated with data.

cboField.Items.Add("Make");

cboField.Items.Add("EngineSize");

cboField.Items.Add("RentalPerDay");

cboField.Items.Add("Available");

cboField.SelectedIndex = 0;

cboOperator.Items.Add("=");

cboOperator.Items.Add("<");

cboOperator.Items.Add(">");

cboOperator.Items.Add("<=");

cboOperator.Items.Add(">=");

cboOperator.SelectedIndex = 0;

}

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

{

//Closes the program.

this.Close();

}

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

{

//Checking that text box txtValue is not empty, making connection to database and running the query.

if (txtValue.Text != "")

{

try

{

string sql = String.Format("SELECT VehicleRegNo, Make, EngineSize, DateRegistered, '€' + CAST(RentalPerDay AS varchar) AS RentalPerDay, Available FROM tblCar WHERE {0} {1} @Third", cboField.SelectedItem, cboOperator.SelectedItem);

SqlConnection connection = new SqlConnection(@"Data Source=(LocalDB)\v11.0;AttachDbFilename=|DataDirectory|\Hire.mdf;Integrated Security=True");

connection.Open();

SqlCommand command = connection.CreateCommand();

command.CommandType = CommandType.Text;

command.Parameters.AddWithValue("@Third", txtValue.Text);

command.CommandText = sql;

command.ExecuteNonQuery();

//Creates new data table and populates DataGridView with those records.

DataTable table = new DataTable();

SqlDataAdapter dataAdapter = new SqlDataAdapter(command);

dataAdapter.Fill(table);

tblCarDataGridView.DataSource = table;

//If there is no match message will appear.

if (tblCarDataGridView.Rows.Count == 0)

{

MessageBox.Show("There is no match!");

}

}

//If there is a sql error it will show this message.

catch (SqlException)

{

MessageBox.Show("Error in your query!");

}

//If there is any other error it will show this message.

catch (Exception)

{

MessageBox.Show("Something is wrong, try again");

}

}

//If text box txtValue is empty this message will appear.

else

{

MessageBox.Show("You need to enter value text!");

}

}

}

}