COSC 2100 – Assignment 3

This assignment is an individual assignment. Submit your work via DC Connect by the due date provided.

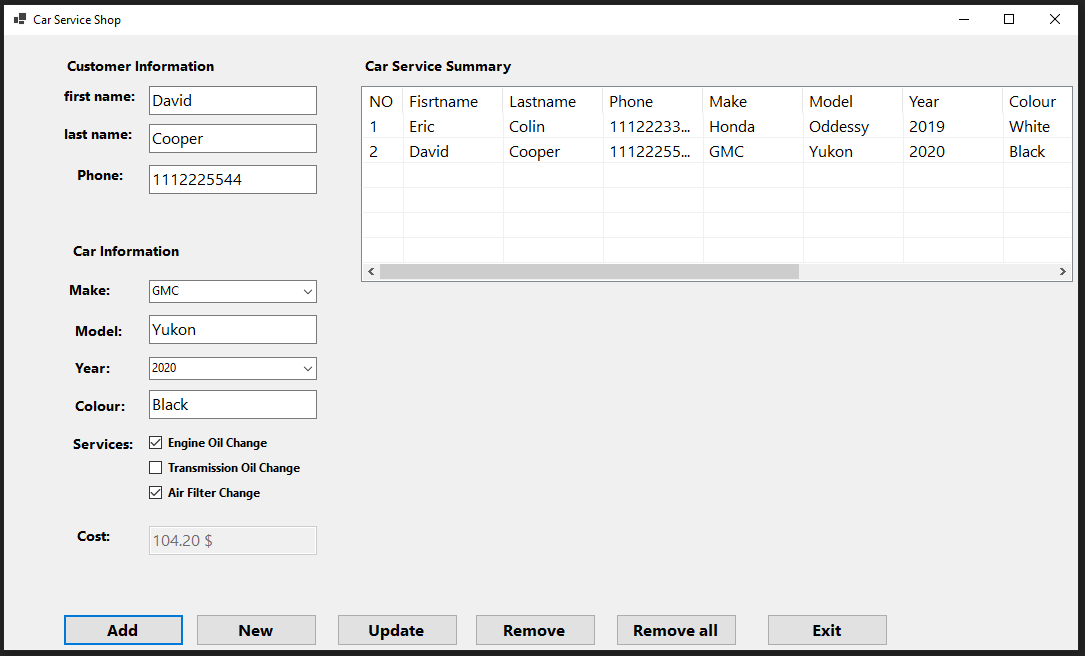
# Client Requirements

Create a **car service shop reservation** application for new clients using C#. The car service shop application will need to maintain the following information:

* Customer information:
  + first name
  + last name
  + phone number
* Car information:
  + make
  + model
  + year
  + colour
  + services
    - engine oil change
    - transmission oil change
    - air filter change
  + cost (automatically calculated based on the selected services)

The employee will be able to enter the customer’s first name, last name, and phone number in the text boxes. The employee will be able to select the Make from a combo box based on a list that will be provided. The Model of the car will be entered in a text box. The Year is the Year the vehicle was manufactured, and the employee can select the Year from a combo box from any year in the last 20 years. The colour of the car will be entered in a text box. The service cost will be calculated based on the selected three services (if engine oil change costs $60.00 plus tax, transmission oil change costs $120.00 plus tax, air filter change costs $40.50 plus tax), and the final cost will be printed into the cost text box. As usual for your applications, the default behavior for the Add and Update buttons will be to validate and accept the entered data. Any valid entries will be added to a list and displayed below. The functionality to reset the form to its default state and to exit the application will be available via buttons and access keys.

# Approximate Design



# Instructions from the UX (User Experience) Department

* The form should not allow minimizing or maximizing the screen.
* The form should have its AcceptButton (Add) and CancelButton (Exit) set appropriately.
* The form should open in the centre of the screen.
* The cost output text box should not be automatically sized.
* ComboBox(es) should have their dropdown style set to DropDownList.
* ComboBox(es) should have their Items collection populated with appropriate data for that list. You may instead choose to use a NumericUpDown for the Year
* The CheckBox’s default state should be unchecked.
* The ListView should have several columns based on the entered information, each uniquely named, with their Text property set appropriately.
* The ListView’s full row select should be set to True.
* The ListView’s view should be set to Details.
* Where possible, all interactive controls (including TextBoxes) should have access keys (hotkeys) set up.
* All controls should have their tab indexes set in a useful way.
* All controls that the user interacts with, including the cost output text box, should have a tooltip that explains each control’s purpose.

# General Operation

* An employee can exit by:
  + Clicking the Exit button with a mouse cursor
  + Using an access key to activate the Exit button.
  + Tabbing to the Exit button and tapping Enter on the keyboard.
* An employee can reset the application by:
  + Clicking the New button with a mouse cursor
  + Pressing the Esc key on the keyboard
  + Using an access key to activate the New button.
  + Tabbing to the New button and tapping Enter on the keyboard.
* The employee can:
  + Select a Make in a ComboBox representing the manufacturer.
  + Enter a Model in a textbox representing a car name.
  + Select a Year in a ComboBox or NumericUpDown representing when the car was built.
  + Enter colour in a textbox representing a car colour.
  + Select required services using 3 Checkboxes.
  + The cost will be calculated based on the checked (selected) services, and the resulting cost will be printed on the cost text box.
* When the Add button is used:
  + The employee can enter data by:
  + Clicking the Add button with a mouse cursor.
  + Pressing the Enter key on the keyboard.
  + Using an access key to activate the Enter button.
  + If the entered data is invalid, validation messages should be displayed in a Message Box or error tooltip.
  + If the entered data is valid, all input fields should be cleared, and the new car’s data should be added to the ListView control, along with any prior existing entries.
* When the New button is used:
  + All input controls will clear or be set to their default state.
  + The ListView control will NOT clear
  + The ListView control’s list index will be set to -1
* When the Update button is used:
  + The employee can select cars from the ListView control.
  + When a car is selected, the input controls will be populated with the selected data.
  + The employee can freely modify the vehicle in the input controls.
  + When the Update button is used, all input fields should be cleared, and the car’s newly modified data should be updated in the ListView control.
* When the Remove button is used:
  + The selected ListView car information will be removed from the reservation list.
  + All input controls will clear or be set to their default state.
* When the Remove all button is used:
  + All car’ information will be removed from the reservation list.
  + All input controls will clear or be set to their default state.
* When the Exit button is used:
  + Ends the application.
  + All input controls will clear or be set to their default state.

# Instructions from the Business Analyst/Tech Lead

* For input to be considered valid:
  + User first name, last name and Phone number must be entered; these fields cannot be blank or filled only with spaces.
  + A valid Make must be selected from a list (you may choose which Make values are in this list)
  + A Model must be entered; this field cannot be blank or filled only with spaces.
  + A valid Year must be selected.
  + A Colour must be entered; this field cannot be blank or filled only with spaces.
  + Services must be at least one selected.
  + A valid Cost must be a real number greater than or equal to 0 (see assumptions above)
* Validation should take place in a function that takes the input fields as parameters and returns True if all these fields are valid and False if any of these fields are invalid.
* For each field that is invalid when the function is called, an error message specific to that field should be written to the result label at the bottom of the form.
* There must be a function (method) to reset the form's input fields to their default state. This function will be triggered when either the Exit button is pressed or when a valid entry is submitted using the Add button.
* A class must be defined representing the car shop services; code reusability is something to consider.
* The class must be defined in a separate file (Service.cs or ServiceClass.cs).
* Class Properties: The following properties must exist with these exact names and casing:
  + Count (Integer): a read-only property referring to the total number of services objects.
  + IdentificationNumber (Integer): a read-only property with a unique identifier for entered cars.
  + firstName (String)
  + lastName (String)
  + phoneNumber (int)
  + Make (String)
  + Model (String)
  + Colour (String)
  + Year (Integer)
  + EngOilChange (Boolean)
  + TransOilChange (Boolean)
  + AirFilterChange (Boolean)
  + Price (Decimal)
* Constructors: There must be two constructors.
  + The Default constructor must:
    - increment the service number count.
    - set this service’s IdentificationNumber based on the updated service number count.
  + The Parametrized constructor must:
    - call the Default constructor.
    - include needed parameters representing the customer’s first name, last name and phone number, car’s Make, Model, Year, colour, three services and cost.
    - the private variables within the class will be set using the values passed into these parameters.
* Methods: There will only be one method for the Service class
  + GetServiceData() as String: a string representing the service object’s data.
* Ensure that your design incorporates suitable exception handling.

# Development Tips

With regards to the processing that happens when the Add button is activated, start by checking whether the service is a new entry (i.e., nothing is selected in the ListView) or an existing service from the list (i.e., something is selected in the ListView).

* If the service is a new entry:
  + Instantiate a new service object using the parametrized constructor for customer’s first name, last name and phone number, car’s Make, Model, Year, colour, services, and cost.
  + Add the new service object to a collection representing all entered services, with the list key equal to the service’s IdentificationNumber (NO)
* If the service is an existing entry:
  + Get the service object from the collection using the IdentificationNumber (NO) as a key.
  + Assign its customer’s first name, last name and phone number, car’s Make, Model, Year, colour, services, and cost using the properties of the service object.
* Regardless of whether it is new or not:
  + Clear the list view’s items.
  + Loop through the cars collection, adding each car to the list view’s items to repopulate it.

# Additional Considerations

* The program must be adequately documented:
  + methods, functions, and event handlers should all have block comments.
  + calculations, decisions, and iteration should be explained with brief comments.
  + there should be a header at the top of each human-generated code files, including your name, the last modified date, and a description.
* Adhere to an approved style guide and ensure your variable names, form controls, and other elements are properly cased and adequately descriptive.

# Assessment

Each student’s work will be submitted individually as a compressed folder containing the entire solution. The instructor will assess Your completed work using the COSC2100 Assignment 3 Rubric available on DC Connect. Detailed feedback will be provided. For the Specific Requirements portion of the rubric, the most important criteria are:

• A Windows Form created in .NET 6 or .NET 7

• Customers can be entered into the list

• Selecting a customer in the list populates the input fields

• A class (in a separate code file) has been implemented

• A GetServiceData () or equivalent function has been created as part of the class

• A minimum of one-third in the Functionality section of the rubric.