MS.NET

Mini Project

Car Information Management System

(CIMS)

Document Revision History

Date	Revision No.	Author	Summary of Changes

Table of Contents

INTRODUCTION

This document outlines a mini project for the .NET Line of Technology (LOT). The project is to develop Car Information Management System. This document contains the requirements, work flow of the system and gives guidelines on how to build the functionality gradually in each of the course modules of the .NET LOT.

SETUP CHECKLIST

Minimum System Requirements

- Intel Pentium 4 and above Windows 2007, 2008 and 2010
- Memory 4 GB
- Internet Explorer 8.0 or higher
- SQL Server 2012 client and access to SQL Server 2012 server
- Visual Studio 2017

INSTRUCTIONS

- The code modules in the mini project should follow all the coding standards.
- Create a directory by your name in drive **<drive>**. In this directory, create a subdirectory **MiniProject**. Store your Project here.
- You can refer to your course material.
- You may also look up the help provided in the MSDN
- Since this project work will span over couple of months, you will need to take care of maintaining the code

PROBLEM STATEMENT

OBJECTIVE

Development of Car Information Management System (CIMS)

Abstract of the project

Car Information Management System (CIMS) helps Customers view the complete information about a given company's car like Features, Engine & Performance, Safety, etc. There are two actors in this application.

(Note: Currently we have to create application for Administrators only)

1) Administrator: Can perform the following tasks:

- Add Car
- Update Car
- Delete Car
- Can Search for a car and view its details, on the basis of its Name.
- Get List of Cars and view the details like Features, Engine & Performance Safety on the basis of its Manufacturer and Type.

2) Customer: Can perform the following tasks:

- Can Search for a car and view its details, on the basis of its Name.
- Get List of Cars and view the details like Features, Engine & Performance Safety on the basis of its Manufacturer and Type.

- **Phase 1 :** The system will first develop using C# only – where car data will be store as a Collection Classes. For user interaction, system will use Console Application
- **Phase 2 :** Later on data will be store in MS SQL Server database; system will use ADO.NET or LINQ and Entity Framework for the same. User Interface will be designed using WPF
- **Phase 3** : CIMS will become web based application, following MVC design pattern. Here the application will be develop in ASP.NET MVC.

Macro level Operations/offerings:

1. Add Car Details (Manufacturer, Model, Engine, Mileage, BHP, Transmission, Seats, Boot Space, AirBagDetails, Features, Type, etc.).

   Type Options: Hatchback, Sedan or SUV.

2. Modify Car Details
3. Remove Car
4. Get List of Cars and view the details like Features, Engine & Performance, AirBagDetails on the basis of its Manufacturer and Type.
5. Search for a car and view its details, on the basis of its Name.

MODULE LIST and MODULE DETAILS

CREATE CAR

Following info need to capture

- Manufacturer Name
- Model
- Type (Hatchback, Sedan or SUV)
- Engine
- BHP
- Transmission (Manual/Automatic)
- Mileage
- Seats
- AirBagDetails
- BootSpace
- Other Features
- Ex. Showroom Price (Delhi)

SEARCH CAR

User should be able to search a Car by model.

MODIFY CAR

Search (By model) a Car and modify its details. System should show existing data/info of Car and support modify one, more or all info.

REMOVE CUSTOMER

Search (by Model) a Car and remove the Car. System should ask for confirmation and on confirmation the data will be removed.

CAR SUMMARY (VIEW)

System should show (display) Car list in a tabular format (one row for each Car, and columns for Car details). It is not required to show all the details of a Car in a table; only important info like – Manufacturer Name, Model, Type, Ex. Showroom Price should be displayed.

Constrains

- Proper validation is required
- System must show appropriate massage on all activity (whether activity is successful or failure)
- User must have proper menu to select the activity (create, modify, search, view, remove) that user want to perform.
- For “Car” Entity:
    - ManufacturerName : String
    - Model : String (Must be unique)
    - Type : String: “Hatchback”, “Sedan” or “SUV”

- Engine : String. Should be 4 characters long. With first and third character number, second should be a "." And last character should be "L"
- BHP: Should be a Number
- Transmission: Either Manual or Automatic
- Mileage: Should be a Number
- Seat: Should be a Number
- AirBagDetails : String
- BootSpace: Should be a Number
- Price: Should be a Number

- For "Manufacturer" Entity:
  - Id: Numeric
  - Name: String
  - ContactNo: String. Should be 10 numbers
  - RegisteredOffice: String.
- For "CarType" Entity:
  - Id: Numeric

- Type: String. (“Hatchback”, “Sedan” or “SUV”)

- For “CarTransmissionType” Entity:
  - Id: Numeric
  - Type: String. (“Manual” or “Automatic”)

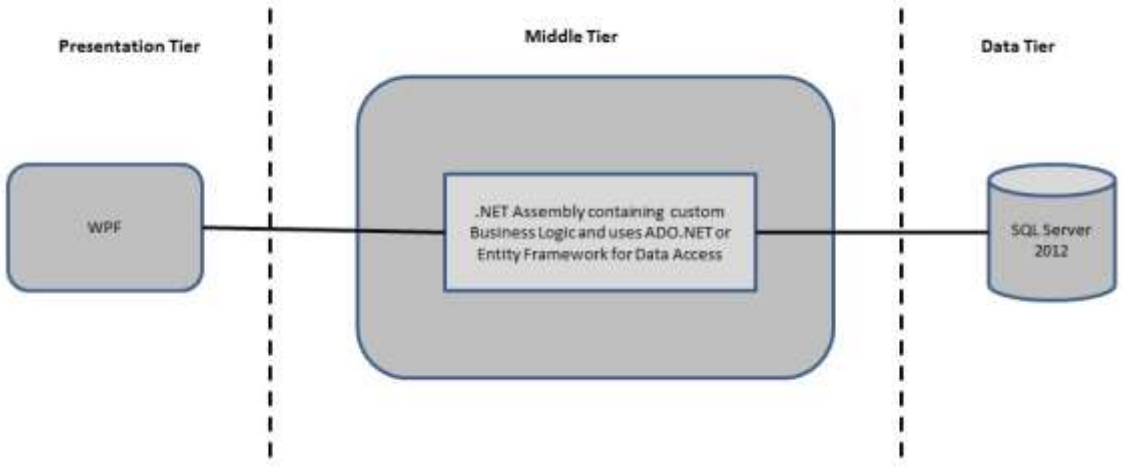
FUNCTIONAL COMPONENTS OF THE PROJECT

Application Architecture:

Distributed web applications traditionally to be designed and built across three logical tiers:

- Database Access Layer (DAL)
- Business Logic Layer (BLL)
- Presentation Layer

The DAL refers to the database itself, the stored procedures, and the component that provides an interface to the database. The BLL refers to the component that encapsulates all the business logic of the application. And, the Presentation layer refers to the web application pages.



Design guidelines

- All the exceptions/errors to be captured and user friendly message to be displayed on the CommonError page.
- Data access layer of 3-tier use Entity Framework data access using SQL stored procedures - All the database interaction would be performed using Data Access Component.

TECHNOLOGY USED:

- *Presentation Layer*
    1. *Console Application, WPF, ASP.NET MVC 5*
- *Business Layer*
    1. *Business Logic Components and Services :-*
        a. C# 5.0
- *Database Layer*
    1. *Databases:-*
        a. SQL Server 2012

IMPLEMENTATION

SUMMARY OF THE FUNCTIONALITY TO BE BUILT:

The participants need to develop the Customer Management System by building the functionality incrementally in each of the course modules of .NET LOT.

Sr. No	Course	Duration (in PDs)	Functionality to be built
1	MS SQL Server 2012	4	Creating relevant database tables and stored procedures
2	NET Framework 4.6 + C# 7.0 + Introduction to WPF	10	Developing Business components (C# classes)
3	ADO.NET with LINQ and Entity Framework	4	Creating data model and data context and using LINQ to entities
4	ASP.NET MVC 5	4	Incorporating advanced UI functionality with ASP.NET MVC 5
5	Mini Project Presentation	1	The Mini Project Presentation day

Note: Saturday half day will be devoted for Mini project

GUIDELINES ON THE FUNCTIONALITY TO BE BUILT:

The functionality and components to be built in each of the course modules of .NET LOT is as follows:

1. Course: SQL Server 2012

This section describes some of the basic steps involved in designing and creation of the database for the application.

Create Data Model - identify the different tables and fields that we will need, which would later be used for building the rest of the application.

Database Schema - Taking these objects, we can easily identify our main tables in the database.

a. Create the following database tables with following fields: [make your assumptions in case you require few more fields]

Table Name: CAR		
Field Name	**Constraint**	**Data Type**
Id	Primary Key (System Generated)	Int
Model	Unique	Varchar
ManufacturerId	Foreign Key (References Id column from Table - Manufacturer)	Varchar
TypeId	Foreign Key (References Id column from Table - CarType)	Int
Engine	Check	Varchar
BHP		Int
TransmissionId	Foreign Key (References Id Column from Table - CarTransmissionType)	Int

Mileage	Not Null	Int
Seat	Not Null	Int
AirBagDetails	Not Null	String
BootSpace	Not Null	Varchar
Price	Not Null	Decimal

Table Name: Manufacturer		
Field Name	**Constraint**	**Data Type**
Id	Primary Key (System Generated)	Int
Name	Unique	Varchar
ContactPerson	Unique	Varchar
RegisteredOffice	Not Null	Varchar

Table Name: CarType		
Field Name	**Constraint**	**Data Type**
Id	Primary Key (System Generated)	Int
Type	Unique	Varchar

Table Name: CarTransmissionType		
Field Name	**Constraint**	**Data Type**
Id	Primary Key (System Generated)	Int
Name	Unique	Varchar

