Car Rental Project

# Requirements

## Stories:

* As a Manager I want to Add, Modify and Delete Car Rental Branches
* As a Manager I want to define the types of cars that will be used for rental.
* As a Manager I want to manage (Add and Delete) cars in the inventory.
* As a Manager I want to establish where the cars are.
* As a Manager I’d like to get a report of the current and past Reservations.
* As a Manager I’d like to Mange (add, modify, delete) Employees.
* As an Employee of a Branch I’d like to receive cars that are returned.
* As a User I’d like to be able to sign up to become a Member.
* As a User I’d like to Sign In as a Member.
* As a User I’d like to make searches on cars.
* As a User I’d like to I’d like to get a quote for a specific car.
* As a Member I’d like to make a reservation.
* As a Member I’d like to get a report of the history of my reservations.

## Technical Requirements:

* The system should run on the web.
* The following technologies will be used: MVC, JavaScript, JQuery, HTML, Entity Framework.
* The system should try to follow the guidelines of Separation of Concern.
* The system should follow the SOLID principles.

## Classes:

* Manager
* Employee
* Member
* User
* Car Rental Branch
* CarBrand
* CarModel
* CarType
* Car
* Reservation
* Reservation Report

### ViewModel Classes

* EditCarViewModel

## Class Employee: Member

Properties: Shift, Salary

## Class Manager: Employee ?

## Class Member

Properties: SSN\*, UserName\*, HashedPassword\*, Name\*, LastName\*, Gender\*, Email\*, DOB, Address, Phone, Picture, FromDate, Discount

## Class Branch

Properties: BranchId, Name, Location, Coordinates?, Phone

## Class CarBrand

Properties: CarBrandId, BrandName

Class CarModel

Properties: CarModelId, CarBrandId, ModelName

## Class CarType

Properties: CarTypeID, Model, DailyPrice, DailyLatePrice, Gear

## Class Car:

Properties: CarId, CarType, Mileage, Picture, Available, Plates, Branch, Rented

## Class Reservation:

Properties: ReservationId, Branch, FromDate, ToDate, ActualReturnDate, User, Car, Status

## ViewModels:

By creating ViewModels, we simplified the access to lists of CarBrands, CarTypes and Branches, which we used as DropDowns in Edit and Create pages.

## ViewModelFactory:

We used this class to generate all the ViewModel classes.

## Class EditCarViewModel:

Properties: CarData, Branches, CarTypes

## Class EditCarTypeViewModel:

Properties: CarTypeData, CarModels

# Process

I defined the Models with the classes described above.

I used Entity Framework to generate the database.

I added some Data Validation Annotations.

I started a Migration and used (by calling Update-Database as I made changes to the DB).

I used the scaffolding feature of Visual Studio.

I used the MVVM pattern. Created a ViewModels folder where I created the non-mapped classes.

I modified the communication between Controller and Views. Instead of ViewBag, I passed on the ViewModel as a parameter.

I added Custom Validations for things such as testing the unicity of a CarBrand or a CarModel.

Todo:

+ Relational to compound key

* Membership
* Logo
* DatabaseInitializer: restructure it based on the new DB structure and Call it in “Main”
* Understand DB naming
* Make Cars the central point where cars are added and include there buttons to modify related tables: CarType, CarBrand, CarModel.
* Define if we have a startup DB
* WCF to define Cars
* Search for reservations
* List of all CarTypes of a specific CarModel, All CarModels of a specific CarBrand, All Cars from a specific CarBrand or CarBrand-CarModel.
* Publish to Azure
* Bind only the fields that need to be bound
* Currency Format when editing
* Eliminate the debug db.Database.Log = s => System.Diagnostics.Debug.WriteLine(s);