Exercise 12: Garage 2.0 - The Return

The Garage is coming to the web - in ASP.NET MVC! Since this is the first MVC-based version of the garage we will be focusing on functionality and the GUI.
This exercise description is written in priority order, the first parts are required for the application, while the later are more like bonus assignments.

Since this is a larger exercise it is a good idea to start out with some planning about what needs to be done and in what order to do it. How will you divide the tasks between you? When will you consider something as done in the group? How will you work with github?

Each morning you should have a **short** meeting of 10 min maximum. The purpose of this meeting will be to update the rest of the group on where you are at, what you have done since the last meeting and what you are planning to do during the day. As well as talking about anything that might be preventing you from proceeding.

Single model

In this version we will only be using a single model, just like in the previous MVC-exercise. So no model for vehicle type, only one model: *ParkedVehicle* with properties like: *Type, Registration number, Color, Brand, Model, Number of Wheels* and so on.

Basic functionality

- Vehicles should be able to Park and Check out.
- When a user parks they should be able to register all information about the vehicle.
- There should be an **overview-view** where all vehicles currently parked are displayed, but only with some basic data like: RegNr, Type, Color, Parking time*.
- There should be a **detail-view** where you can see all information about a specific vehicle.

Appearance

The application should have a uniform appearance, it should be clear and easy to navigate. When it comes to the general language of the application as well as the instructions it should be appropriate for its function, for example: we don't *create* a car -> we *park* a car, we also don't *delete* a car -> we *check it out* of the garage, and so on.

Input-control

The scaffolding gives us some input-control automatically (for instance by not allowing anything but numbers in a numbers field) , but the rest we will have to add manually.

*Parking time is a bonus task for now

Vehicle types

Vehicle types should not be manually inputted, but should instead make use of a dropdown list containing all the appropriate vehicle types. As long as you are using MVC 5.1 or newer this can be done easily with the help of an enum without needing any extra models: www.codeproject.com/Articles/776908/Dealing-with-Enum-in-MVC (In future projects we will solve this slightly differntly)

Timestamp

A timestamp for when the vehicle parked (to be saved in the database alongside the other vehicle information) should not be entered manually but should be added automatically when the user parks a vehicle. However, the scaffolding will add it as a text field - you will need to handle that.

Searching and sorting

As the number of parked vehicles increase so does the need for a search method. Implement a way to search for vehicle based on RegNr. If you have extra time add search options for the other fields. If after that you still have time left implement sorting buttons above each field that allows you to choose one column to sort descending.

Receipt

When you leave a garage it is customary to get a receipt that declares how long you were parked for. Implement a new view that shows the RegNr, Vehicle type check in/out times, total time parked and price automatically after a vehicle is checked out. The receipt should use a **ViewModel** as it's model. If you have time left over, make sure that the receipt is suitable to print.
You can decide the price for yourselfs.

Good luck!