Cars App

An big car company is using a mobile app to manage their fleet. The employees are able manage all their cars and create simple reports.

On the server side at least the following details are maintained:

- Id - the internal car id. Integer value greater than zero.
- Name - the car name. A string of characters representing the car name.
- Model - the car model. A string of characters representing the car model. Ie. Audi, Bmw, Skoda, Seat, Toyota.
- Status - the game status. Eg. “old”, “new”.
- Year - the year when the car was made. Eg. 1996.
- km - the number of kilometers that the car has. Eg. 10000, 1000.

The application should provide at least the following features:

- Employee Section (separate activity - available offline too)
    a. (1p) View all the available cars. Using **GET /cars** call, the client will receive the list of cars available in the system. If offline the app will display the local stored list of cars and a way to retry the connection. Once back online the local list will be replaced with the online result.
    b. (1p) Edit car details. The name, status and the year should be available to be updated using a **POST /modify** call, by using a valid car id. Available both online and offline. If offline the information will be saved locally and once connected the app will perform the server call.
    c. (1p) Add more miles. Using **POST /km** call with the car id the user will be able to specify the number of km that should be added. The server will respond with the car object with the total number of kilometers.
    d. (0.5p) Remove the local persisted information.
- Manager Section (separate activity - available only online)
    a. (1p) Using **POST /add** call by specifying a car name, model and a year a new car will be added.
    b. (1p) Remove a car. Using **DELETE /car** by specifying the car id.
    c. (1p) List car ascending by the total number of kilometers. Using the same **GET /cars** call (note that the server is maintaining the cars unsorted).

(1p) On the server side once a new car is added in the system, the server will send, using a websocket channel, a message, to all the connected clients/applications, with the new car object. Each application, that is connected, will add the new car in the local persisted list of cars.

(0.5p) On all server operations a progress indicator will be displayed.

(0.5p) On all server interactions, if an error message is received, the app should display the error message using a toast or snackbar.

(0.5p) On all interactions (server or db calls), a log message should be recorded.