Car App

A car dealer shop is providing services using a mobile app. The clients are able to view the available cars, purchase one or more cars or return a car back to the shop.

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.
- Quantity - the number of cars of this type that are available. An integer value greater than zero.
- Type - the car type. Eg. “tesla”, “mercedes”, “bmw”, “toyota”. A string type.
- Status - the car status. Eg. “available”, “sold”, ”used”. A string type.

The application should provide at least the following features:

- Client Section (separate activity - available offline too)
  a. (1p) View 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 an offline message and a way to retry the connection and the call. For each car the name, quantity and the type are displayed.
  b. (0.5p) Buy a car. The client will buy a car, if available, using a POST /buyCar call, by specifying the car id and the quantity. Available online only.
  c. (1p) Once the client purchased a car, the list of his cars will be displayed. The list is persisted on the device, on the local storage, available offline too. The client can return a car, from his list, by doing a POST /returnCar call using the car id, operation available only when online. Only in the first 30 days after the purchase.
  d. (0.5p) View the list of his purchased cars.
  e. (0.5p) Remove all the cars from the local persisted list.
- Employee Section (separate activity - available only online)
  a. (1p) The list of all the available cars. The list will be retrieved using the GET /all call, in this list along with the name, quantity and type, the app will display the status also.
  b. (1p) Add a car. Using a POST /addCar call, by sending the car object a new car will be added to the store list, on success the server will return the car object with the id field set.
  c. (1p) Delete a car. Using DELETE /removeCar call, by sending a valid car id, the server will remove the car. On success 200 OK status will be returned.

(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 product in the list of available 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.