Car Inventory Management App

In the automotive domain, an application is developed to assist dealerships in managing their car inventory. The app allows dealership staff to record and manage details of cars, enabling suppliers and customers to access and interact with pertinent information.

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

- Id: Integer value greater than zero.
- Name: A string representing the car model.
- Supplier: A string representing the car supplier's name.
- Details: A string containing car details (e.g., features, specifications).
- Status: A string representing the car status (e.g., "available," "sold," "pending").
- Quantity: An integer value representing the quantity of the car available.
- Type: A string representing the car type (e.g., "sedan," "SUV," "truck").

The application should provide the following features (available without restarting the app):

● Car Organizer Section (Separate Activity/Screen):

A. (1p) Add New Car: Using **POST /car** endpoint, the organizer can add a new car to the inventory, both online and offline.

B. (2p) View All Cars: Using **GET /cars** call, organizers can retrieve and display a list of all cars in the inventory. The list should include id, name, supplier, and type for each car. In offline mode, an offline message and a retry option should be provided. The data should persist on the device after retrieval, regardless of online, offline, or restart conditions. Upon successful retrieval, as the data is now available on the device, additional server calls are unnecessary.

C. (1p) View Car Details: By selecting a car from the list, the organizer can view all details. Using **GET /car** endpoint with the car id, the data should be retrieved from the server each time and made available on the device.

● Staff Section (Separate Activity/Screen) - Available Online Only:

A. (1p) View Cars Types: Using **GET /carstypes** call, staff can retrieve a list of cars with their status. The server will return all the cars in the system, the app should group them by type, and display for each type the total quantities.

B. (1p) Request Cars: Staff members can request additional cars by selecting a type from the previous list, using **PUT /requestcar** with the car type.

● Supplier Section (Separate Activity/Screen) - Available Online Only:

(1p) View Car Orders: Using **GET /carorders** call, suppliers can view a list of car orders placed by dealership organizers. Display a proper message if no orders are available.

● (1p) On the server side, once a new car is added to the system, the server will send, using a WebSocket channel, a message to all the connected clients/applications with the new object. Each application that is connected will display the received object fields, in a human form (not JSON text or toString) using an in-app “notification” (e.g., using a snack bar, toast, or an on-screen dialog).

(0.5p) Throughout all server or database operations, a progress indicator will be displayed.

(0.5p) On all server or DB interactions, if an error message is received, the app should display the error message using a toast or snackbar. A log message should be recorded on all interactions (server or DB calls).