Shopping App

A group of friends is managing their shopping list using a mobile application. They are able to manage the list of items and track what is needed or bought.

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

- Id - the internal item id. Integer value greater than zero.
- Name - A string of characters representing the item name.
- Quantity - Integer value greater than zero, representing the item quantity.
- Price - Integer value greater than zero, representing the item price.
- Status - A string of characters representing the status. Eg. “desired”, “needed”, “bought”, “canceled”, etc.

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

- Items Section (separate activity)
  a. (1p) Register a new item. Using **POST /item** call by specifying all the item details. Available online and offline.
  b. (2p) View all the items found in the system, in a list, using **GET /items** call. The list should display at least the id, name, quantity, and status. If offline, the app will display an offline message and a way to retry the connection and the call. Once the list is retrieved it should be available offline and online.
  c. (1p) Remove an item, by selecting the item from the list and using **DELETE /item** call with specifying the item id. Available online and offline.
- Shopping Section (separate activity) - Available online only.
  a. (1p) View all the items in the “desired” or “needed” status. Using the same **GET /items** call, the app will retrieve all the items and will present only the ones having the “desired” or “needed” status. The list will be ordered by price and quantity.
  b. (1p) Buy the selected item from the previous list. Using the **POST /buy** call by specifying the item id, the item status will be changed.
- Stats Section (separate activity) - Available online only.
  a. (1p) View the top 10 most “bought” items. Using the **GET /bought** call retrieve all the bought items. The app will present only the top 10 by quantity descending.

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

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

(0.5p) On all server/DB interactions, if an error message is generated, the app should display the error message using a toast or snack bar. On all interactions (server or DB calls), a log message should be recorded.