Market App

A market is providing services using a mobile app. The clients are able to view the available products and purchase one or more of them. The clerk is able to manage the stocks.

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

- Id - the internal product id. Integer value greater than zero.
- Name - the product name. A string of characters representing the product name.
- Description - a short product description. A string of characters.
- Quantity - the products available. An integer value greater or equal to zero.
- Price - the product price. Integer value greater than zero.
- Status - the product status. Eg. “new”, “sold”, ”popular”, “discounted”. A string type.

The application should provide at least the following features:

- Client Section (separate activity - available offline too)
  a. (1p) View the products in a list. Using **GET /products** call, the client will retrieve the list of product 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 product the name, quantity and the price are displayed.
  b. (1p) Buy a product. The client will buy a product, if available, using a **POST /buyProduct** call, by specifying the product id and the quantity. Available online only.
  c. (1p) Once the client purchased a product, all the product details are presented to the user.
  d. (1p) View the list of his purchased products. The list is persisted on the device, on the local storage, available offline.
- Clerk Section (separate activity - available only online)
  a. (1p) The list of products ascending by quantity. The list will be retrieved using the same **GET /all** call, in this list along with the name, quantity and price, the app will display the status also. Note that from the server you are retrieving an unsorted list.
  b. (1p) Add a product. Using a **POST /product** call, by sending the product object a new item will be added to the store list, on success the server will return the product object with the id field set.
  c. (1p) Delete a product. Using **DELETE /product** call, by sending a valid product id, the server will remove the product. On success 200 OK status will be returned.

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

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