Store App

A store is managing the supply chain using a mobile app. Each supplier is able to view the open requests in the system and fulfill some of them. The employees are able manage the requests.

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

- Id - the internal request id. Integer value greater than zero.
- Name - the request name. A string of characters representing the request name.
- Product - the product name. A string of characters.
- Quantity - the product quantity. A numeric value.
- Status - the status of the request. A string of characters. Eg. “open”, “closed”, “new”, etc.

The application should provide at least the following features:

- Supplier Section (separate activity)
  a. (1p) View the open requests in a list. Using **GET /requests** call, the supplier will retrieve the list of open request found in the system. If offline, the app will display an offline message and a way to retry the connection and the call. For each request the name, product and quantity are displayed.
  b. (1p) Fulfill a request. The supplier will be able to use a fulfill a request, using a **POST /fill** call, by specifying the request id. Available online only.
  c. (1p) See all his fulfilled requests, in a list. All request fields should be displayed. Available offline too.
- Employee Section (separate activity)
  a. (1p) The list of fulfilled request descending by quantity. The list will be retrieved using the **GET /closed** call, in this list along with the product name, the app will display the quantity too. Note that from the server you are retrieving an unsorted list.
  b. (1p) The list of top 10 unfilled (open) requests descending by quantity. The list will be retrieved using the **GET /big** call, in this list along with the request and product names, the app will display the status too. Note that from the server you are retrieving an unsorted list of all the request that have the quantity over 5.
  c. (1p) Add a new request. Using a **POST /request** call, by sending the request object a new request will be added in the system, on success the server will return the request object with the id field set.
  d. (1p) Delete a request. Using **DELETE /request** call, by sending a valid request id, the server will remove the request. On success 200 OK status will be returned.

(1p) On the server side once a new request is added in the system, the server will send, using a websocket channel, a message to all the connected clients/applications with the new request object. Each application, that is connected, will display the received request details using an in app “notification” (like snackbar or toast or a dialog or a message on the screen).

(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.