Online Shop App

An online shop is managing client orders using a mobile application. The client is recording the order and it is placing it in the store. The store employees are able to fulfill or cancel the order. The manager will be able to see all the orders.

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

- Id - the internal order id. Integer value greater than zero.
- Details - the order details. A string of characters.
- Status - the order status. A string of characters. Eg. “pending”, “preparing”, “delivering”, “canceled”, etc.
- User - the user id. An integer value representing the user id.
- Age - the order age. An integer value representing the number of seconds that the order needs to be canceled or get to the delivering status.
- Type - the order type. A string of characters. Eg. “normal”, “priority”, etc.

The application should provide at least the following features:

- Client Section (separate activity)
    a. (1p) Record the current user id in application settings.
    b. (1p) Record an order. Using **POST /order** call by specifying all the order details using as the user the current user id. Available online and offline.
    c. (2p) View all the orders that were created by this user. Using **GET /orders** call, the client will retrieve all his orders by the specified user id. If offline, the app will display an offline message and a way to retry the connection and the call. Once retrieved it should be available offline.
- Store Section (separate activity)
    a. (1p) View all the orders that are having the status pending in the system in a list. Using **GET /pending** call, the employee will retrieve all the orders having this status. Available online only.
    b. (1p) By selecting an order from the list, the employee will be able to convert the status to “delivering”, “preparing” or “canceled”. Using **POST /status** by specifying the order id and the new status. Available online only.
- Manager Section (separate activity)
    a. (1p) View the top users, in a list containing the user id and his number of orders. Using the **GET /all** call the manager will be able to retrieve all the orders available in the system. The list should present the users in descending order by their number of orders.

(1p) On the server-side, once a new order is added in the system, the server will send, using a WebSocket channel, a message to all the connected clients/applications with the new order object. Each application, that is connected, will display the received order details, in human form (not JSON text or toString) using an in-app “notification” (like a snackbar or toast or a dialog on the 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 snackbar. On all interactions (server or DB calls), a log message should be recorded.