Restaurant App

A restaurant is managing client orders using a mobile application. The waiter is recording the order from the client and is placing it in the kitchen. When the order is prepared, the cook will notify the waiter. The client will be able to see the status of an order in all its phases.

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

- Id - the internal order id. Integer value greater than zero.
- Table - the table name. A string of characters representing the table name.
  Details - the order details. A string of characters.
- Status - the order status. A string of characters. Eg. “recorded”, “preparing”, “ready”, “canceled”, etc.
- Time - the order preparation time. An integer value representing the number of seconds needed to prepare the order.
- Type - the order type. A string of characters. Eg. “normal”, “delivery”, etc.

The application should provide at least the following features:

- Waiter Section (separate activity)
  a. (1p) Record an order. Using **POST /order** call by specifying all the order details the waiter will capture the client request. Available online and offline.
  b. (2p) View all the orders that are having the status ready in the system in a list. Using **GET /orders** call, the waiter will retrieve all the orders that are in the ready status. 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.
  c. (1p) By selecting an order from the list, the waiter will be able to view all the order details. To retrieve the order details the **GET /order** call can be used by specifying the order id. Available online only.
- Kitchen Section (separate activity)
  a. (1p) View all the orders that are having the status recorded in the system in a list. Using **GET /recorded** call, the cook will retrieve all the orders having this status. Available online only.
  b. (1p) By selecting an order from the list, the cook will be able to convert the status to “ready”, “preparing” or “canceled”. Using **POST /status** by specifying the order id and the new status. Available online only.
- Client Section (separate activity)
  a. (1p) View the order details for the specified table. Using **GET /my** call, by specifying the table name the client will be able to view all the details associated with his last order. If the order is missing the application will display a proper message.

(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.