Recipe App

A group of restaurants is sharing their favorite recipes using a mobile application. Each restaurant is able to manage their own recipes.

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

- Id - the internal recipe id. Integer value greater than zero.
- Name - the recipe name. A string of characters representing the recipe name.
- Details - the recipe details. A string of characters.
- Time - the estimated preparation time. Integer value greater than zero representing the number of estimated seconds.
- Type - the recipe type. A string of characters. Eg. “beginner”, “medium”, “advanced”.
- Rating - the recipe rating. An integer value.

The application should provide at least the following features:

- Main Section (separate activity)
  a. (1p) View the types available in the system in a list. Using **GET /types** call, the user will retrieve the list of all recipe types found in the system. 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.
  b. (2p) By selecting a type, the user will be able to get to the list of recipes that are having the selected type. To retrieve the list of recipes by type the **GET /recipes** call can be used by specifying the type. Once retrieved the list should be available offline.
  c. (1p) Add a recipe. Using **POST /recipe** call by specifying all the recipe details the user will be able to create a new recipe. Available online only.
  d. (1p) Delete a recipe. By specifying the recipe id, using the **DELETE /recipe** call, the user will be able to delete a recipe. Available online only.
- Rate Section (separate activity)
  a. (1p) The list of top 10 underrated recipes sorted ascending by rating. The list will be retrieved using the **GET /low** call, in this list along with the name, details, and type, the app will display the current rating. Note that from the server you are retrieving all the recipes.
  b. (1p) Rate a recipe. From the above list, the user should be able to select a recipe and increment its rating by one unit using the **POST /increment** call by specifying the recipe id.

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