Recipe App

A group of users is sharing their favorite recipes using a mobile application. Each user 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.
- Description - the recipe description. A string of characters.
- Ingredients - the recipe ingredients. A string representing the ingredients needed for the recipe.
- Instructions - the recipe instructions. A string representing the steps to make the recipe.
- Category - the recipe category. A string of characters. Eg. “desserts”, “main dishes”, “vegan”, etc.
- Difficulty - the recipe difficulty. A string of characters. Eg. “easy”, “medium”, or “hard”.

The application should provide at least the following features:

- Main Section (separate activity)
    A. (1p) View the categories available in the system in a list. Using the **GET /categories** call, the user will retrieve the list of all recipe categories 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 category, the user will be able to get to the list of recipes that are having that category. To retrieve the list of recipes having the specified category the **GET /recipes** call can be used by specifying the category. 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 selecting a recipe from the list, and using the **DELETE /recipe** call, the user will be able to delete a recipe. Available online only.
- Difficulty Section (separate activity)
    A. (1p) The list of the top 10 easiest recipes sorted ascending by difficulty and category. The list will be retrieved using the **GET /easiest** call, in this list along with the name, ingredients, and instructions, the app will display the current difficulty and the category. Note that from the server you are retrieving all the recipes.
    B. (1p) Change the difficulty level of a recipe. From the above list, the user should be able to select a recipe and change its difficulty level using the **POST /difficulty** call by specifying the recipe id and the new difficulty level.
- (1p) On the server side once a new recipe is added to 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 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.