Files Catalog App

A user is managing his files using a mobile app. He is able to register a new file, query files or view reports about them.

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

- Id - the internal file identifier. Integer value greater than zero.
- Name - the file name. A string of characters.
- Status - file current status. A string of characters. Eg. “shared”, “open”, “draft”, “secret”, etc.
- Size - An integer value representing the size in KB.
- Location - A short description of the file path/location. A string of characters.
- Usage - the number of times the file was searched. An integer value.

The application should provide at least the following features:

- Record Section (separate activity)
  a. (1p) Record a file. Using **POST /file** call by specifying all the file details. Available online and offline.
  b. (2p) View all the files in the system. Using **GET /all** calls, the user will retrieve all the files. If offline, the app will display an offline message and a way to retry the connection and the call. Once retrieved, the file details should always be available, no other server calls are needed.
- Manage Section (separate activity)
  a. (1p) View all the available file locations in the system. Using **GET /locations** calls, the user will retrieve all the available locations. Available only online.
  b. (1p) View all the available files in the selected location in a list. Using **GET /files** calls, the user will retrieve all the available files in the selected location. Available only online.
  c. (1p) Delete a file, the user will be able to delete a selected file. Using **DELETE /file** call, by sending the file id. Available online only.
  d. (1p) View the top 10 files, in a list containing the file name, status, usage, and location. Using the same **GET /all** call. The list should present the result in descending order by their usage value. Note that the list received from the server is not ordered.

(1p) On the server-side, once a new file is added in the system, the server will send, using a WebSocket channel, a message to all the connected clients/applications with the new file object. Each application, that is connected, will display the received file name, size, and location values, 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.