Expenses App

A group of students is managing their dorm expenses using a mobile app. Everybody is able to add requests, items needed for the dorm, fulfill requests or view reports.

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

- Id - the internal request identification. Integer value greater than zero.
- Name - the request name. A string of characters.
- Status - request current status. A string of characters. Eg. “open”, “filled”, “canceled”, “postponed”, etc.
- Student - the student name that initiated the request. A string of characters.
- eCost - the estimated request cost. An integer value representing the estimated cost.
- Cost - the real request cost. An integer value representing the real cost.

The application should provide at least the following features:

- My Section (separate activity)
  a. (1p) Record the student name in application settings. Persisted to survive app restarts.
  b. (1p) Record a request. Using **POST /request** call by specifying all the request details and the student name persisted in the previous step. Available online and offline.
  c. (2p) View all the requests added in the system belonging to him. Using **GET /my** call, the student will retrieve all his requests. If offline, the app will display an offline message and a way to retry the connection and the call. Once retrieved, the request should be available offline.
- All Section (separate activity)
  a. (1p) View all the open requests added to the system in a list. Using **GET /open** call, the student will retrieve all requests. The list should contain the name and the eCost. Available only online.
  b. (1p) Fulfill a request, the student will be able to convert the status to “filled”, “postponed” or “canceled”. Using **POST /change** call, by specifying the id and the new status. Available online only. When changing the status to ‘filled’, the cost field should be specified too.
- Report Section (separate activity)
  a. (1p) View all the filled request descending by cost, in a list containing the request name, eCost, and cost. Using the same **GET /filled** call. The list should present the result in descending order by their cost value. Note that the list received from the server is not ordered.

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