Books App

A group of students is managing their books using a mobile app. Everybody is able to add books, borrow them or view reports.

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

- Id - the internal book identifier. Integer value greater than zero.
- Title - the book title. A string of characters.
- Status - book current status. A string of characters. Eg. “available”, “missing”, “canceled”, “borrowed”, etc.
- Student - the student name that borrowed the book. A string of characters. If empty it means that the book is not borrowed.
- Pages - An integer value representing the total number of pages.
- UsedCount - the number of times the book was borrowed. An integer value.

The application should provide at least the following features:

- Owner Section (separate activity)
  a. (1p) Record the student name in application settings. Persisted to survive app restarts.
  b. (1p) Record a book. Using **POST /book** call by specifying all the book details and the student name persisted in the previous step. Available online and offline.
  c. (2p) View all the borrowed books by the current student. Using **GET /books** call, the student will retrieve all his books. If offline, the app will display an offline message and a way to retry the connection and the call. Once retrieved, the books should be available offline.
- Borrow Section (separate activity)
  a. (1p) View all the available books in the system in a list. Using **GET /available** call, the student will retrieve all the available books. The list should contain the book title, pages, and the usedCount. Available only online.
  b. (1p) Borrow a book, the student will be able to borrow the selected book. Using **POST /borrow** call, by specifying the book id and the current student name. Available online only.
- Report Section (separate activity)
  a. (1p) View the top 10 books, in a list containing the book title, and usedCount. Using the **GET /all** call. The list should present the result in descending order by their usedCount value. Note that the list received from the server is not ordered.

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