Games App

A group of friends is managing their games using a mobile app. Everybody is able to add a game, borrow some or view reports.

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

- Id - the internal game identifier. Integer value greater than zero.
- Name - the game name. A string of characters.
- Status - game current status. A string of characters. Eg. “available”, “missing”, “canceled”, “borrowed”, etc.
- User - who borrowed the game. A string of characters. If empty it means that the game is not borrowed.
- Size - An integer value representing the size in KB.
- PopularityScore - the number of times the games was borrowed. An integer value.

The application should provide at least the following features:

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

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