Game Store App

A game shop is providing services using a mobile app. The clients are able to view the available games, purchase one or more games or rent a game for a limited time.

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

- Id - the internal game id. Integer value greater than zero.
- Name - the game name. A string of characters representing the game name.
- Quantity - the number of games of this type that are available. An integer value greater than zero.
- Type - the game type. Eg. “action”, “adventure”, “board”.
- Status - the game status. Eg. “available”, “sold”, ”rent”.

The application should provide at least the following features:

- Client Section (separate activity - available offline too)
  a. (1p) View the available games. Using GET /games call, the client will receive the list of games available in the system. If offline the app will display an offline message and a way to retry the connection and the call. For each game the name, quantity and the type are displayed.
  b. (0.5p) Buy a game. The client will buy a game, if available, using a POST /buyGame call, by specifying the game id and the quantity. Available online only.
  c. (1p) Once the client purchased a game, the list of his games will be displayed. The list is persisted on the device, on the local storage, available offline too. The client can return a game, from his list, by doing a POST /returnGame call using the game id.
  d. (1p) Rent a game for 30 days. Using POST /rentGame call using the gameId. Once rented, the app will display the list of his rented games. This will be available offline too.
- Employee Section (separate activity - available only online)
  a. (1p) The list of all the available game. The list will be retrieved using the GET /all call, in this list along with the name, quantity and type, the app will display the status also.
  b. (0.5p) Add a game. Using a POST /addGame call, by sending the game object a new game will be added to the store list, on success the server will return the game object with the id field set.
  c. (0.5p) Delete a game. Using DELETE /removeGame call, by sending a valid game id, the server will remove the game. On success 200 OK status will be returned.
  d. (1p) Update the game details. Using POST /updateGame call, by sending a valid game object, the server will update the game represented by the specified id.

(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 add the new game in the list of available games.

(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.

(0.5p) On all interactions (server or db calls), a log message should be recorded.