Parking App

A parking manager is using a mobile application to track all the lots. The users are able to reserve view and use the free spaces, the managers are able to manage and view statistics.

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

- Id - the internal space id. Integer value greater than zero.
- Number - A string of characters representing the space name, like “B101”.
- Address - A string of characters representing the space location.
- Status - A string of characters representing the status. Eg. “free”, “taken”.
- Count - An integer number representing the usage count.

The application should provide the following features (available without restarting the app):

- Manager Section (separate activity)
  a. (1p) Register a new space. Using **POST /space** call by specifying all the space details. Available online and offline.
  b. (2p) View all the spaces found in the system, in a list, using **GET /spaces** call. The list should display at least the id, number, address, and status. If offline, the app will display an offline message and a way to retry the connection and the call. Once the list is retrieved it should be available offline and online.
  c. (1p) Remove a space, by selecting the space, from the list, and using **DELETE /space** call with specifying the space id. Available online and offline.
- Users Section (separate activity) - Available online only.
  a. (1p) View all the free spaces. Using the **GET /free** call, the app will retrieve all the free spaces. The list will be ordered by number.
  b. (1p) Take the selected space from the previous list. Using the **POST /take** call by specifying the space id, the space status will be changed to “taken”.
- Stats Section (separate activity) - Available online only.
  a. (1p) View the top 15 most used spaces. Using the same **GET /spaces** call retrieve all the space. The app will present only the top 10 by count descending.

(1p) On the server-side, once a new space is added to the system, the server will send, using a WebSocket channel, a message to all the connected clients/applications with the new space object. Each application, that is connected, will display the received space details, in human form (not JSON text or toString) using an in-app “notification” (like a snack bar or toast or a dialog on the screen), regardless of the opened 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 snack bar. On all interactions (server or DB calls), a log message should be recorded.