# Robots App

A group of students is sharing their robot characteristics using a mobile application. Each group is able to manage its own robots.

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

- Id - the internal robot id. Integer value greater than zero.
- Name - the robot name. A string of characters representing the robot name.
- Specs - the robot specifications. A string of characters.
- Height - the height of the robot. Integer value greater than zero.
- Type - the robot type. A string of characters. Eg. “lego”, “industrial”, “custom”, etc.
- Age - the robot age. An integer value.

The application should provide at least the following features:

- Main Section (separate activity)
    a. (1p) View the types available in the system in a list. Using **GET /types** call, the user will retrieve the list of all robot types found in the system. If offline, the app will display an offline message and a way to retry the connection and the call. Once retrieved it should be available offline.
    b. (2p) By selecting a type, the user will be able to get to the list of robots that are having the selected type. To retrieve the list of robots by type the **GET /robots** call can be used by specifying the type. Once retrieved the list should be available offline.
    c. (1p) Add a robot. Using **POST /robot** call by specifying all the robot details the user will be able to create a new robot. Available online only.
    d. (1p) Update the height of a robot. By specifying the robot id and the new height, using the **POST /height** call, the user will be able to update the robot height. Available online only.
- Age Section (separate activity)
    a. (1p) The list of the top 10 oldest robots sorted descending by age. The list will be retrieved using the **GET /old** call, in this list along with the name, specs, and type, the app will display the current age. Note that from the server you are retrieving all the robots.
    b. (1p) Update the age of a robot. From the above list, the user should be able to select a robot and update its age by using the **POST /age** call by specifying the robot id and the new age.

(1p) On the server-side, once a new robot is added in the system, the server will send, using a WebSocket channel, a message to all the connected clients/applications with the new robot object. Each application, that is connected, will display the received robot details, in human form (not JSON text) 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.