3D Printing App

A small business is using a mobile application to manage the 3D printing models from its customers. The customers are able to record a model and view its status. The employees are able to process the models. The manager will be able to see usage reports.

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

- Id - the internal print id. Integer value greater than zero.
- Model - the model name. A string of characters.
- Status - printing status. A string of characters. Eg. “ordered”, “printing”, “done”, “failed”, “canceled”, etc.
- Client - the client id. An integer value representing the client id.
- Time - the time needed to print the model. An integer value representing the number of seconds that the printers need to print the model.
- Cost - the printing cost. An integer value representing the printing cost.

The application should provide at least the following features:

- Customer Section (separate activity)
  a. (1p) Record the current client id in application settings. Persisted to survive app restarts.
  b. (1p) Record a model. Using **POST /model** call by specifying all the model details and the id persisted in the previous step as the client id. Available online and offline.
  c. (2p) View all the models added in the system belonging to him. Using **GET /models** call, the client will retrieve all his models. If offline, the app will display an offline message and a way to retry the connection and the call. Once retrieved, the models should be available offline.
- Employee Section (separate activity)
  a. (1p) By specifying a model id, the employee will be able to convert the status to “done”, “failed” or “canceled”. Using **POST /process** by specifying the model id and the new status. Available online only.
- Manager Section (separate activity)
  a. (1p) View the top 5 most expensive models, in a list containing the model name and its cost. Using the **GET /all** call the manager will be able to retrieve all the models available in the system. The list should present the result in descending order by their cost value.
  b. (1p) View the top 10 easiest to print models, in a list containing the model name and its printing time. Using the same **GET /all** call. The list should present the result in ascending order by their printing time.

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