Airline App

An big airline company is using a mobile app to manage their fleet. The employees are able manage all their airplanes and create simple reports.

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

- Id - the internal airplane id. Integer value greater than zero.
- Name - the airplane name. A string of characters representing the airplane name.
- Manufacturer - the airplane company name. A string of characters representing the airplane company name. Ie. Airbus, Antonov, Boeing, Bombardier, Embraer, Sukhoi, Tupolev, Saab.
- Status - the airplane status. Eg. “retired”, “in service”.
- Year - the year when the first flight was made. Integer value greater than zero. Eg. 1996.
- Miles - the number of miles that the airplane gained. Integer value greater than zero. Eg. 10000, 1000.

The application should provide at least the following features:

- Employee Section (separate activity - available offline too)
  a. (1p) View all the available airplanes. Using **GET /airplanes** call, the client will receive the list of airplanes available in the system. If offline the app will display the local stored list of airplanes and a way to retry the connection. Once back online the local list will be replaced with the online result.
  b. (1p) Edit airplane details. The name, status and the year should be available to be updated using a **POST /update** call, by using a valid airplane id. Available both online and offline. If offline the information will be saved locally and once connected the app will perform the server call.
  c. (1p) Add more miles. Using **POST /miles** call with the airplane id the user will be able to specify the number of miles that should be added. The server will respond with the airplane object with the total number of miles.
  d. (0.5p) Remove the local persisted information.
- Manager Section (separate activity - available only online)
  a. (1p) Using **POST /add** call by specifying a airplane name, manufacturer and a year a new airplane will be added.
  b. (1p) Remove a airplane. Using **DELETE /airplane** by specifying the airplane id.
  c. (1p) List airplanes descending by the total number of miles. Using the same **GET /airplanes** call (note that the server is maintaining the airplanes unsorted).

(1p) On the server side once a new airplane is added in the system, the server will send, using a websocket channel, a message, to all the connected clients/applications, with the new airplane object. Each application, that is connected, will add the new airplane in the local persisted list of airplanes.

(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.