Fitness App

A mobile application designed to help users manage their workout routines and track their fitness progress. The app will maintain the following details on the server side:

- Id - the unique identifier for the workout. Integer value greater than zero.
- Name - The name of the workout. A string of characters.
- Type - The type of workout. A string of characters. Eg. "cardio", "strength training", "yoga", etc.
- Duration - The duration of the workout in minutes. An integer value.
- Calories - The number of calories burned during the workout. An integer value.
- Date - The date the workout was completed. A string in the format "YYYY-MM-DD".
- Notes - Additional notes or comments about the workout. A string of characters.

The application should provide the following features:

- Record Section (separate screen)

  a. **(2p)**(1p) View all the workouts in the system using **GET /all** calls. The user can retrieve all the workouts. If offline, the app will display an offline message and a way to retry the connection and the call. Once retrieved, even online, the workout details should always be available, no other server calls are needed.

  b. **(1p)**(0.5p) Record a new workout using **POST /workout** call by specifying all the workout details. Available online and offline.
- Manage Section (separate screen, available only online)

  a. **(1p)**(0.5p) View all the available workout types in the system. Using **GET /types** calls, the user will retrieve all the workout type names.

  b. **(1p)**(0.5p) View all the available workouts for the selected type in a list. Using **GET /workouts** calls, the user will retrieve all the available workouts of the selected type.

  c. **(1p)**(0.5p) Delete a workout, the user will be able to delete the selected workout. Using **DELETE /workout** call, by sending the workout id.
- Reports Section (separate screen, available only online)

  a. **(1p)**(0.5p) View the top 10 workouts in a list containing the workout name, type, duration, and date. Using the same **GET /all** call. The list should present the result in descending order by their date and duration value. Note that the list received from the server is not ordered.

  b. (1p) View the total number of calories burned for each workout type. Using the same **GET /all** call. The list should display the total number of calories burned for each workout type.

  c. (1p) View the workout history for a selected date range. Using the same **GET /all** call the user should be able to filter the values by specifying the start and end date. The list should present the workout name, type, duration, calories burned, and date. The list should be ordered by the workout date.
- Profile Section (separate screen)

  a. (0.5p) Record the user's personal details, like name, height, weight, and primary workout type. in the application settings. Persisted to survive app restarts.

  b. (0.5p) View the user's personal details on a separate screen/dialog.

  c. (0.5p) Edit the user's personal details and update the details in the application settings.
- Notifications (should work on all screens)

  (1p) On the server side, once a new workout is added to the system, the server will send a message using a WebSocket channel to all the connected clients/applications with the new workout object. Each application that is connected will display the received workout name, type, and duration values in human form using an in-app "notification" (like a snackbar or toast or a dialog on the screen).
- Developer Details

  (0.5p) On all server/DB 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 and log the initial error message. A log message should be recorded on all interactions (server or DB calls).