Movie App

A production company is managing its movies using a mobile application. The directors, writers and the actors are able to manage and view the movies that they are involved in.

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

- Id - the internal movie id. An integer value greater than zero.
- Title - the movie title. A string of characters representing the movie title.
- Description - the movie description. A string of characters representing the movie plot.
- Genre - the movie genre. A string of characters representing the movie genre.
- Year - the release year. An integer value greater than zero.
- Rating - the rating received by the users. An integer value greater than zero.
- Length - the movie length in minutes. An integer value.

The application should provide at least the following features, in separate activities:

- Directors Section
  a. **(1p)** View the movies available in the system in a form of a list, ordered by title and year. Using **GET /movies** call, the user will retrieve the list of all the movies found in the system. Note that from the server the list is retrieved unsorted. 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. If the list is already available offline it should always be used, no new server calls of this type are needed anymore. The user should be able to trigger a manual refresh if needed.
  b. **(1p)**(0.5p) View all the details related to a selected movie. By selecting a movie from the previous list, the user will be able to see all the movie details in a separate screen. In order to get the movie details **GET /details** call should be used. Available only if online.
  c. **(1p)**(0.5p) In the details screen from above, using **POST /update** call by specifying the movie object with a valid movie id, the user will be able to update the movie details. Available online only. The operation should be reflected on the main list too.
  d. **(1p)**(0.5p) In the details screen from above, using **DELETE /delete** call by specifying the movie id, the user will be able to delete the selected movie. Available online only. The operation should be reflected on the main list too.
  e. (0.5p) While on the main list screen the user should have the option to add a new movie. Using **POST /add** call by specifying the movie object, the user will be able to add a new movie into the system. The operation should be available offline too. Once online the app should detect and resume the operation.
- Writers Section
  a. **(1p)**(0.5p) View all the movie genres. Using **GET /genres** call the user will be able to see the list of all the available genres found in the system. Available both online and offline. Once retrieved it should be used from the local storage.
  b. **(1p)**(0.5p) Once a specific genre is selected all the movies in that genre should be displayed in a list. The list should present the movie title and the year and should be sorted descending by year. In order to retrieve the list, the **GET /moviesByGenre** call should be used.

c. **(1p)**(0.5p) By selecting a movie the user should be able to update the movie description. Using **POST /updateDescription** call by specifying the movie id and the new description.

- Actors Section
  a. **(1p)** View all the movies in the current year. Using the same **GET /movies** call, the application should retrieve and display the filtered list by the year. The operation should be available only online. The list should display the movie title, year and current rating in descending order by rating.
  b. (0.5p) By selecting a movie the user should be able to increase the rating of the currently selected movie using a slider. The minimum value should be one and the maximum should be five. The operation should be available online only. The operation should use **POST /updateRating** call by specifying the movie id and the new rating.
  c. (0.5p) View all the movies in the current year ordered by length. Using the same **GET /movies** call, the application should retrieve and display the filtered list by the year. The operation should be available only online. The list should display the movie title, year and length in descending order by length.
  d. (0.5p) View the top ten movies by rating. Using the same **GET /movies** call, the application should retrieve and display the top ten movies by rating. The operation should be available only online. The list should display the movie title, year and rating in descending order by rating.

(1p) On the server side, once a new movie is added in the system, the server will send, using a web socket channel, a message to all the connected clients/applications with the new movie object. Each application, that is connected, will add the new movie in their main list, if visible, or add the details in the local database to be used later. A notification message should be displayed too.

(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. On all interactions (server or DB calls), a log message should be recorded.

NOTE: If your laboratory grade was above 4.5 you need to solve only the requirements that have the points in bold. If your laboratory mark is less then 4.5 than in order to compute the exam mark we are using the regular points (non-bold ones).