Market App

A market is managing its values using a mobile application. The traders and their customers are able to manage and view their securities.

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

- Id - the internal security id. An integer value greater than zero.
- Name - the security name. A string of characters representing the security name.
- Description - the security description. A string of characters.
- Type - the security type. A string of characters representing the type. Eg. sell, buy, etc.
- Date - the date. An integer value greater than zero.
- Status - order status. A string of characters. Eg. open, closed.
- Amount - the order amount. An integer value.

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

- Trader Section
  a. **(1p)** View the orders available in the system in a form of a list, ordered by name and status. Using **GET /orders** call, the user will retrieve the list of all the orders 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 the selected order. By selecting an order from the previous list, the user will be able to see all the details in a separate screen. In order to get the order details **GET /order** call should be used. Available only if online.
  c. **(1p)**(0.5p) In the details screen from above, using **POST /order** call by specifying the order object with a valid order id, the user will be able to update the order 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 /order** call by specifying the order id, the user will be able to delete the selected order. 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 order. Using **POST /create** call by specifying the order object, the user will be able to add a new order into the system. The operation should be available offline too. Once online the app should detect and resume the operation.
  f. (0.5p) While on the main list screen the user should have the option to update also the order status. Using **POST /updateStatus** call by specifying the order id and the new status, the user will be able to update only the order status. The operation should be available online only.
- Client Section
  a. **(0.5p)** Give the user the option to specify and store the security name. The name should be persisted in the local database and available for further operations.
  b. **(1p)**(0.5p) View all the open orders for the current saved name. Using **GET /bySecurity** call, the user will be able to see the list of all the available orders found

in the system for the specified security name. Available both online and offline. Once retrieved it should be used from the local storage.

c. **(1p)**(0.5p) Once an order is selected the user should be able to close the order. Using **POST /close** call by specifying the order id.

- Statistics Section
  a. **(0.5p)** Give the user the option to specify and store the security name. The name should be persisted in the local database and available for further operations.
  b. **(1p)**(0.5p) View all the open orders associated with the persisted name. Using the same **GET /orders** call, the application should retrieve and display the filtered list by the date. The operation should be available only online. The list should display the order description, amount and type in ascending order by date.
  c. (0.5p) View all the close orders ordered by amount. Using the same **GET /orders** call, the application should retrieve and display the filtered list by the date for the current persisted name. The operation should be available only online. The list should display the order description, date and amount in descending order by amount.
  d. (0.5p) View the top ten orders by amount. Using the same **GET /orders** call, the application should retrieve and display the top ten orders by amount. The operation should be available only online. The list should display the order description, date, type and amount in descending order by amount and date.

(1p) On the server side, once new orders are added in the system, the server will send, using a web socket channel, a message to all the connected clients/applications with the new order object. Each application, that is connected, will add the new order 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).