Take Home Assignment

Notes:

- Requirements may be incomplete or unclear. If so, note any assumptions made and continue.
- If any section of the assignments takes too long, move on the next section.

Background

An order book is the list of orders that a trading venue uses to record the interest of buyers and sellers in a particular financial instrument. An order will be one of either a buy or sell order and contain a price and a volume.

e.g.

Buy Orders			Sell Orders	
Volume	**Price**		**Price**	**Volume**
5	21		23	10
10	20		26	5
10	19			

Back End

1. Create a Spring Boot Maven Project.
2. Create a Rest Controller with the following methods
    a. Method: GET, Path /market/orderbook
        i. Description: returns the current open buy and sell orders
        ii. Input: none
        iii. Output an object containing array of buy orders and an array of sell orders
            1. An order contains two fields:
                a. price: a positive integer
                b. volume: a positive integer
            2. e.g.

Buy Orders			Sell Orders	
Volume	**Price**		**Price**	**Volume**
5	21		23	10
10	20		26	5
10	19			

```
{buyOrders: [{price: 21, volume:5}, {price: 20,
volume:10} , {price: 19, volume:10} ],
{sellOrders: [{price: 24, volume:10}, {price: 26,
volume:5} ]
```

    b. Method: GET, Path /market/placeOrder

i. Description: Place order validates the input, and adds the order to the appropriate side of the order book.
ii. Method: Post,
iii. Path /market/placeOrder
iv. Input:
- side: String either "Buy" or "Sell"
- price: a positive integer
- volume : a positive integer

v. Output:
- Return: 200 for success, 400 for invalid data.
- Order is added to internal list of buy or sell orders
- e.g. /market/placeOrder data:

```
{side: "Buy", price: 15, volume 10}
```

- Return 200:
- Order added to list of buy orders

Front End

1. Create an Angular Front End
2. Implement functionality to call the /market/orderbook when application starts.
3. Format the orderbook into a table showing the list of bids and list of offers. Style it to make look nice.
   a.

Buy Qty	Buy Price	Sell Price	Sell Qty
400	20.00	23.90	50
850	12.00	24.00	1000
25	10.00	24.45	1000
750	6.00	24.90	925
500	5.10	25.00	3950

4. Add a form to allow entry of an order. Form fields for an order are:
   a. Side: Buy or Sell
   b. Price: a positive integer
   c. Volume: a positive integer
5. Implement client side validation on the form
6. When the user clicks on a buy or sell order in the orderbook table, the form must populate with the values defaulted to the order clicked.
7. When the form is submitted,
   a. publish the order to /market/placeOrder
   b. if /market/placeOrder returns success then update the display of the orderbook by calling /market/orderbook

Back End - Part 2

- For /market/orderbook
  - Ensure that buy orders are sorted in descending orders by price
  - Ensure that sell orders are sorted in ascending orders by price.

Front End - Part 2

- Highlight orders in the orderbook table that match an order entered on the orderbook form.
  - i.e. If you are placing a buy order, your buy orders would be matched against the array of sell orders where your buy order price >= sell order price.
    - e.g
      - buy order price: 12, volume: 10

- Would match the sell order in red.

Buy Orders			Sell Orders	
Volume	Price		Price	Volume
5	20		**10**	**10**
5	15		15	10
10	10		20	10
5	5		25	10

Email a copy of your projects or make the code available through a version control system.