Java Backend Developer Case (Brokage Firm Challenge)

You are expected to write backend api for a brokage firm so that their employees can send, list and delete stock orders for their customers. All orders have a status either PENDING, MATCHED or CANCELED.

Your backend service should have those endpoints:

- Create Order: Create a new order for a given customer, asset, side, size and price Side can be BUY or SELL. Customer is a unique id for a customer. Asset is the name of the stock customer wants to buy. Size represents how many shares customer wants to buy. Price represents how much customer wants to pay for per share. Orders should be created with PENDING status.
- List Orders: List orders for a given customer and date range. (you can add more filter if you want)
- Delete Order: Cancel a pending order. Other status orders cannot be deleted
- List Assets: List all assets for a given customer (you can add filters if you want)

Requirements:

- All endpoints should be authorized with an admin user and password
- All info should be stored in database as below:
  Asset: customerId, assetName, size, usableSize
  Order: customerId, assetName, orderSide, size, price, status, createDate

  Side: BUY, SELL
  Status: PENDING, MATCHED, CANCELED

  Don't open another table for TRY. TRY should be stored in asset table as it is an asset too.

  Note: Orders will be against TRY asset. That means you can buy and sell with only TRY.

- While creating new order customer TRY asset's usable size or usable size of asset that wanted to be sold should be checked to see if there is enough amount and then it should be updated accordingly.

- While cancelling and order, TRY asset' usable size or asset that wanted to be sold in order's usable size should be updated accordingly.
- You should write unit tests

Implementation:

We would like u to build a java application using Spring Boot Framework. You can use h2 db as database. Try to build and design your code as it will be deployed to prod (or at least test env ☺) Make sure to add information how to build and run the project. For business domain you can search on internet to learn more about how ordering in a stock market works.

Bonus 1: Add a customer table and authorize endpoints per customer(with a login endpoint). That means each customer can only access and manipulate their own data. Admin user can still manipulate all customer's data.

Bonus 2: Add an admin user endpoint that can match pending orders. (While matching, both TRY asset's and bought assets' size and usable size values should be updated accordingly) Orders will NOT be matched against each other. You just need to match orders and update assets accordingly.