# Description

In this exercise, you are asked to build a simple stock trading service.

The service accepts buy and sell orders through a REST API with create, update, and delete endpoints. All orders are limit orders. See [definition](https://www.investopedia.com/terms/l/limitorder.asp)

An order has the following properties:

* id: String – *Auto generated* *ID of the order*
* user\_id: String – *ID of the user making the order*
* stock\_symbol: String – *Unique symbol of the stock*
* order\_type: String [one of buy | sell]
* units: Integer – *no of stock units to buy or sell*
* price: Double – *buy limit price or sell limit price.*
* order\_time: Timestamp – *time when the order was created or last updated.*

An order can only be updated or deleted if it’s not filled (partially or fully).

The service will use the Price-Time Priority (FIFO) matching algorithm to match buy and sell orders. See [explanation](https://jellepelgrims.com/posts/matching_engines#matching-algorithms)

When an order is filled (partially or fully), the service records the transaction(s) in the database.

A transaction has the following properties:

* id: String – *Auto generated* *ID of the transaction*
* stock\_symbol: String – *Symbol of the stock*
* buy\_order\_id: String – *ID of the buy order*
* sell\_order\_id: String – *ID of the sell order*
* units: Integer – *No of units in this transaction*
* price: Double – *Price at which the transaction was executed.*

You do not have to worry about the users’ account balance for this service. You can assume that all orders are executable.

# Task

* You must build the service using Java 8 or above. You can use any Java frameworks although Spring/Springboot is preferred.
* You can use any database (Postgres, Mongo, etc) for your backend. The properties of the Order and Transaction defined above are for your guidance only. You should design your database schema that is suitable for the service.
* You can use any reputable Java libraries.
* The solution should be uploaded to a github repo with clear instructions for compiling and running the service.
* Your solution should accompany reasonable unit tests.

**Note:**

* Multiple instances of the service will run in parallel (horizontal scaling) therefore your solution should handle any race conditions that may arise as a result.
* If you need any clarifications or have any questions, please feel free to reach out to us.