Development Abilities Test

Thank you for your interest in working at Wisiex!

We at Wisiex are looking for minds that are trouble solving and able to pass over challenges during development, writing concise, clean and readable code, with good practices, keeping in mind security and fast processing, while respecting deadlines.

For this test, you are going to develop an order matching system, similar to those that run in exchanges.

You are free to design the project folders structure, choose a framework and database schema at your will (as well as choosing whether to use migrations or a single SQL file).

The recommended stack is:

Front-end	**Back-end**
React (CRA or Parcel)	NodeJS: ES6 or Typescript (Babel)
Bootstrap or Bulma	MySQL or Postgre
Socket.io	Redis

Your application should consist in two pages: authentication and orders.

1. Authentication

Only the user name should be provided. Such a username is then registered (if not existing already). Each registered user starts with 100 BTC and 100000 USD. You should generate a JWT token for keeping the authentication session (we suggest using HTTP Bearer authentication).

After the login, it should redirect to your second page:

2. Orders

You should simulate an exchange website for the USD x BTC trading pair. This means that users can buy BTC using USD or sell BTC and get USD. Recommended layout:

Statistics	Buy	Sell	Bid
Global matches	My activity orders	My history	Ask

2.1 Statistics

This place should show the following data:

- Last price: The price of the last order match (i.e. US$ 10000)
- BTC volume: the sum of all BTC traded in order matches last 24 hours (i.e. 10 BTC)
- USD volume: the sum of all USD traded in order matches last 24 hours (i.e. US$ 100000)
- High: maximum USD to BTC price in the last 24 hours (i.e. US$ 13000)
- Low: minimum USD to BTC price in the last 24 hours (i.e. US$ 9000)
- User USD balance (i.e. US$ 100000)
- User BTC balance (i.e. BTC 100)

2.2 Global matches

This place should display a table showing the volumes and price of the latest matches, the first is the most recent.

Price	Volume
US$ 10205	BTC 0.004
US$ 10307	BTC 0.005
...	...

2.3 Buy and Sell

This place should contain forms for issuing new orders, as the following template.

Amount	[BTC_____]
Price	[USD_____]
Total	USD x (auto-calculated)
Submit	

2.4 My active orders

List the user's active orders and enable them to cancel.

Amount	Price	Type	
BTC 1	US$ 10000	Buy	X
BTC 1.05	US$ 10500	Sell	X
BTC 1.055	US$ 10550	Sell	X
...	...	...	...

2.5 My history

Shows the user's history of recent matches.

Price	Volume	Type
US$ 10205	BTC 0.004	Buy
US$ 10307	BTC 0.005	Sell
...	...	...

2.5 'Bid' and 'Ask'

This is the order book/offer book. It lists all active orders. It sums the volumes of all the orders at the same price.

If the user clicks a Bid, the "Sell" form should fill up with the row data. If the user clicks a Ask, the "Buy" form should fill up with the row data.

Price	Volume
US$ 10205	BTC 0.004
US$ 10307	BTC 0.005
...	...

Order matching

Your order matching algorithm should consider the following points:

- It should consider limit orders. This means that it should execute the order in the price that the user specifies or at a better condition. For example, if the user specifies

a Buy order at the price of US$ 10000 for 1 BTC, and an existing Sell order is selling 0.5 BTC for US$ 9000, then half of the order should be executed at US$ 9000.

- After no more orders are found in the book, the order should be saved in the order book. If, otherwise, the whole order could have been executed against existing orders in the book, then it should be marked as complete.

- For preventing double-executing and race conditions, your order matching system should run as a separate daemon, and only run one order at a time. You could use Redis or proper message brokers like RabbitMQ for creating a queue.

- You should consider a 0.5% ‘maker fee’ and 0.3% ‘taker fee’. A maker fee is deducted from the order that is pending on the order book and has been matched by a new order. A taker fee is deducted from the order that matches against another order in the order book at its creation time.

There is no deadline for completing this task, but the time taken is also being evaluated. The result should be in a private GitHub repo with a README file with instructions about how to install and add our CTO, **@jesobreira**, as reader.

Good luck!