TokkaLabs

Tokka Labs Engineering Challenge 1

Instructions

For this take-home challenge, you will be building a simple full stack system to obtain the transaction fee in USDT for all Uniswap WETH-USDC Transactions.

The purpose of this assignment is to assess your practical software engineering skills. In addition to functionality, we will also be evaluating the quality of your code and adherence to software engineering best practices (e.g. proper commenting).

We recommend that you complete the exercise by using Python and/or JavaScript/TypeScript as there are numerous readily available libraries and tutorials for cryptocurrencies trading in these languages. That said, you are free to use any language to complete the assignment.

We encourage you to use a modern framework to build the UI application, such as React, Angular, and Vue. However, if you think the UI is too easy to handle by vanilla JS, CSS and HTML, any solution is also welcomed.

You can expect to spend around 4 to 8 hours on this assignment, with additional time needed to work on the bonus items, but that is not required.

We take a serious view on cheating. You are free to consult and reference any material for the assignment, but please acknowledge them if you do. Do not attempt to pass off other people's work as your own.

Finally, only time and effort are required for this assignment. Besides having a working computer, you are not required to purchase anything else to complete it.

Background

- What is Uniswap? https://docs.uniswap.org/protocol/introduction
- What are Gas and Fees in Ethereum? https://ethereum.org/en/developers/docs/gas/

Requirement

- Use Git for version control, and the code should be accessible on GitHub.com or any other public Git services so that we can look at your commit logs and understand how your solution evolved. Submitting only a single commit will result in rejection. Please share the publicly accessible link with us. No other forms of submission will be entertained.
- Write a comprehensive test suite for your code. Submitting your solution without any tests will result in rejection.
- Dockerize your applications with docker-compose . If you are not using full-stack framework, which UI and Backend are separate apps, please run UI and Backend in 2 different containers.

- Include a clear README.md file to show how to build, test and run your code. If we cannot figure out how to run the code (e.g. instructions are too complicated), we will be forced to assume that it doesn't work. Mention any architectural principles and the reasons for you to choose them (if any). Also, include any other design considerations for the tools you've chosen (like DB, caching, message queue etc).
- Follow one of the well-known coding style guides for the programming language you are using, such as pep8 for Python or Google C++ Style guide for C++.

Problem Statements: Get transaction fee in USDT for all Uniswap WETH-USDC Transactions (Txns)

Backend

- The backend system will need to keep track of all the transactions involved in the Uniswap V3 USDC/ETH pool. A single transaction can contain lots of information, but what we are interested in is just the transaction fee in USDT at the time when this Txn was first confirmed on the blockchain.
- The backend system should support both real time data recording and historical batch data recording, which means after starting the system, it should try to continuously record live Txn data; In addition, it should also be able to process batch job requests to retrieve historical Txns for a given period of time. The system should provide endpoint/s to get the transaction fee for any given transaction hash provided. Please design a RESTful API and state clearly the interface specifications (consider following the Swagger UI standards).

UI

- The one-page UI application need to have a query form that allows user to search the transaction(s) by txId/hash and time range. Initially, the form is empty, and the system should query with no restrictions (except for pagination limit).
- The page needs to show the list of transaction (latest first) with pagination if the search result too big. The default pagination limit could be 50 transactions per page. Users should be able to change the page size and page index.
- The page needs to have a static component to help user summarise the data, such as:
  - total transaction fee in USDT ◦ total transaction fee in ETH
  - current ETH/USDT price

Hints

- You can view all the historical Txns (Transactions) for Uniswap WETH-USDC at https://etherscan.io/address/0x88e6a0c2ddd26feeb64f039a2c41296fcb3f5640#tokentxns.
- And to get the historical Txns programmatically, you can use api provided by Etherscan: https://docs.etherscan.io/api- endpoints/accounts#get-a-list-of-erc20-token-transfer-events-by-address.
- An example of the single Txn return from the Etherscan API looks like the following for Txn hash 0x8395927f2e5f97b2a31fd63063d12a51fa73438523305b5b30e7bec6afb26f48.

```
{
        "blockNumber": "14630148",
        "timeStamp": "1650569060",
        "hash": "0x8395927f2e5f97b2a31fd63063d12a51fa73438523305b5b30e7bec6afb26f48",
        "nonce": "1277",
        "blockHash": "0x462d25f39f29cb1db6bfed16e05c82db653e364a567f3674f78856228c472a06",
        "from": "0x68b3465833fb72a70ecdf485e0e4c7bd8665fc45",
        "contractAddress": "0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2", "to":
        "0x88e6a0c2ddd26feeb64f039a2c41296fcb3f5640",
            "value": "655381908401979660",
            "tokenName": "Wrapped Ether",
            "tokenSymbol": "WETH",
            "tokenDecimal": "18",
            "transactionIndex": "158",
            "gas": "188976",
            "gasPrice": "55506492524",
            "gasUsed": "133516",
            "cumulativeGasUsed": "12995231",
            "input": "deprecated", "confirmations":
            "1"
    }
```

To get live/historical price for ETH/USDT, you can use the orderbook/kline from the Binance SPOT api (https://binance- docs.github.io/apidocs/spot/en/#change-log).

Bonuses (optional)

- Availability, scalability, and reliability - Please take these into consideration when you design the system and explain how your system fulfills them.
- Additional features - Decode the actual Uniswap swap price executed for each Txn. As a single Txn may contain multiple swaps, you will only need to decode and save the executed price from the Uniswap event. Provide a RESTful API for it. Handy tools include (but are not limited to)
    - Infura
    - web3 packages (web.js, web3py etc)