Backend Engineering Assignment

Goal

Build a read-only REST API for serving blockchain data. It will have one endpoint that returns the ERC20 token transfers for a given transaction hash.

- From a user perspective, the result is a simple api that serves block, transaction and token data in a quick, responsive manner.
- From a technical perspective, we're interested in how you structure your code, handle computational complexity, construct caching layers and that you build in a way that's easily extensible in the future. We're less interested in how well you know a given language than your overall experience building and structuring APIs.

Implementation Notes

- The Data Platform team uses Node.js (with Typescript) and Golang as our primarily development languages. We'd prefer the following to be implemented in either of those, but are open to implementations in Java, Python and C# as well. If you'd like to implement in a different language, but please ask recruiting first.
- Unit tests are not required for the assignment, but please leave notes about how you'd set up a test suite in the future.
- Much of this data is static, please keep that mind when deciding on any caching mechanisms.
- Please keep in mind that the APIs you're fetching data from all have rate limits. Your implementation does not need to factor in any of the limits, however, please be aware so you don't lock yourself out while testing locally.
    - Coingecko has a free tier with a 20-50/minute request rate limit. For all rpc APIs (ie api.avax.network/ext/bc/C/rpc), try to keep your request usage to under 30/minute.

ERC20 Token Transfer API

Create an API with an endpoint that returns the ERC20 token transfers for a given transaction hash. The endpoint that accepts a transaction hash, fetches/parses the corresponding eth logs and returns the erc20 transfers within that transaction.

In order to build this endpoint you will need to fetch data from the avalanche rpc url: https://api.avax.network/ext/bc/C/rpc. Documentation for how to call that url can be found here: https://ethereum.org/en/developers/docs/apis/json-rpc/ and a full postman collection for interacting with that url is here: https://docs.avax.network/apis/avalanchego/postman-avalanche-collection)

Unlike native blockchain transactions, you cannot directly request ERC20 transfers from a blockchain node. Instead you will need to fetch and parse the evm logs for a given transaction. Logs can be fetched via the rpc endpoint eth_getTransactionReceipt (not needed for this assignment, but you can read more about eth logs here).

Here is an example of an eth_getTransactionReceipt response with an erc20 transfer contained

Transaction Hash:
0x0459a3eacf4c0dad19e02316e5e80287a77096ac71dc7f6a9f6a6668ecdda7d2

Returns


```
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
    "blockHash": "0x2e4f69d1404bce022d39375f4394ea81010ecd00358d5c76a33bc5d5b73de4dd",
    "blockNumber": "0x14d2eef",
    "contractAddress": null,
    "cumulativeGasUsed": "0x22192",
    "effectiveGasPrice": "0x5d21dba00",
    "from": "0x187b2d576ba7ec2141c180a96edd0f202492f36b",
    "gasUsed": "0x8761",
    "logs": [
      {
        "address": "0x6e84a6216ea6dacc71ee8e6b0a5b7322eebc0fdd",
        "topics": [
          "0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef",
          "0x000000000000000000000000187b2d576ba7ec2141c180a96edd0f202492f36b",
          "0x00000000000000000000000057538553aab34f162b1bedd89914aa540a26073"
        ],
        "data": "0x00000000000000000000000000000000000000000000000081103cb9fb2200000",
        "blockNumber": "0x14d2eef",
        "transactionHash": "0x0459a3eacf4c0dad19e02316e5e80287a77096ac71dc7f6a9f6a6668ecdda7d2",
        "transactionIndex": "0x2",
        "blockHash": "0x2e4f69d1404bce022d39375f4394ea81010ecd00358d5c76a33bc5d5b73de4dd",
        "logIndex": "0x2",
        "removed": false
      }
    ],
    "logsBloom":
"0x000000000200200000000000000000000000100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000090000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000020000000200000000000000000000000000000000000000000000002000020000000000000000000000000000000000000008000000000000000000000000000000000000000000000000000000000000000000000000000000800000",
```

```
    "status": "0x1",
    "to": "0x6e84a6216ea6dacc71ee8e6b0a5b7322eebc0fdd",
    "transactionHash": "0x0459a3eacf4c0dad19e02316e5e80287a77096ac71dc7f6a9f6a6668ecdda7d2",
    "transactionIndex": "0x2",
    "type": "0x0"
  }
}
```


ERC20 Transfer logs always have the following hash as the first topic in an array and exactly 3 topics in total:

0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef

The second topic in the array is the address initiating the token transfer and the third topic is the address receiving the token transfer. The **data** field contains the value (ie Amount) of the token being transferred. The **address** field is the smart contract address of the erc20 token being transferred.

The above log parses out to the following ERC20 transfer

Value: 148800000000000000000
From: 0x187b2d576ba7ec2141c180a96edd0f202492f36b
To: 0x057538553aab34f162b1bedd89914aa540a26073
Contract Address: 0x6e84a6216eA6dACC71eE8E6b0a5B7322EEbC0fDd

You can see this transaction fully parsed out in the avalanche explorer here https://snowtrace.io/tx/0x0459a3eacf4c0dad19e02316e5e80287a77096ac71dc7f6a9f6a6668ecdda7d2#eventlog. You can also navigate through this explorer to find other transactions to test against

Summary:

Stretch Goal

Coingecko offers a free tier API for fetching pricing data: https://www.coingecko.com/en/api/documentation . Using the Simple Token Price endpoint (/simpe/token_price/{id}), add the current token price, when available, to the erc20 transfer responses.

For this example above, the simple token endpoint would take the following parameters:
id: avalanche
contract_address: 0x6e84a6216eA6dACC71eE8E6b0a5B7322EEbC0fDd
vs_currencies: usd

Turning in the assignment

When turning in the assignment please upload the entire codebase to a private github repository and provide read access to the github handles provided by recruiting.

We **DO NOT** accept the code ourselves for security reasons so as such this will be a hard requirement.

Thanks for taking the time to complete the homework and we look forward to your submission!