Staff Software Engineer (Platform)

Objective:

Develop a system with TypeScript that will receive signed transfer objects from the user and relay them to a market maker for execution (the chain execution can be mocked). The system will only accept a transfer if the user has a high enough balance of this token.

```
// transfer object
{
  sender: Address,
    token: {
        address: Address,
        amount: Number,
    },
    targetChain: {
        chainId: Number,
        receiver: Address,
    },
    refund: {
      chainId: Number,
        tx: Tx,
        signedTx: String
    },
    signature: String,
}
```

The API service will sign the transfer object as well and approve with it the correctness of this user request after it validates the user balance (the user balances can be mocked static balances with a database). For the time of the execution, the user balances need to be locked such that no race conditions

appear in case the user sends multiple requests. The solver provides an interface that allows the API service to request the execution status. The API service will trigger the execution of the refund transaction after the solver has successfully executed the transfer object. The refund operation doesn't require an on-chain execution and can be mocked. But it should update the solver's balance in the database.

Requirements:

1. The signatures need to be verified (use ethers.js, web3.js or viem).
2. The solution must process requests concurrently using asynchronous best practices.
3. The token locks need to be kept in-memory as well as in a key-value store in case of a service failure.
4. All cases are verified with E2E tests.
5. The code follows the clean code and SOLID principles.

Assessment Criteria:

You will be assessed on your approach to solving the problem and the correct application of locks for each transaction. You will also be assessed on your understanding and usage of the TypeScript, algorithm choice, code structure, readability, and maintainability. We would like to be able to understand your progress through the task based on your commit history, so please avoid a monolithic commit. Please be prepared to answer questions on your choices in the interview. We will not assess tests in the submission, but you are welcome to use them to develop an effective solution.

Expectations:

We explicitly DO NOT want you to spend more than 4 hours on this task - we will discuss with you based on your commit history how you broke up your time on this task. It does not need to be contiguous but please note that we want to see what you chose to get done in the allotted time, not whether you finished. We have listed a lot of requirements to make sure that there is enough scope to challenge candidates. Please prioritize delivering your working solution in phases. If you cannot complete all the tasks within that time, then submit with

what you have completed, but please ensure that it is functional. Once you have finished the worktest, please email us your solution including a zip archive of your git repository for the project and instructions on how to execute it. We will review your submission and confirm next steps. This may involve inviting you to a 30 min call to discuss your solution, and your approach to solving it.