Settle.

STLHarvest Challenge

1) Setup a project and create a contract

Summary

STLHarvest offers a platform enabling users to deposit **STL tokens** to earn weekly incentives. Participants should have the flexibility to withdraw their initial investments and corresponding reward shares whenever they choose. Each week, the STLHarvest team manually adds new rewards to the pool through a specific contract function.

Requirements

- Only the STLHarvest team is authorized to add rewards to the pool.
- These rewards are allocated to the collective pool rather than to individual accounts.
- The only token allowed to be deposited in STLHarvest's pool is the STL token.
- Participants have the ability to withdraw both their initial deposits and their proportionate share of the rewards, which is calculated based on the timing of their deposits.

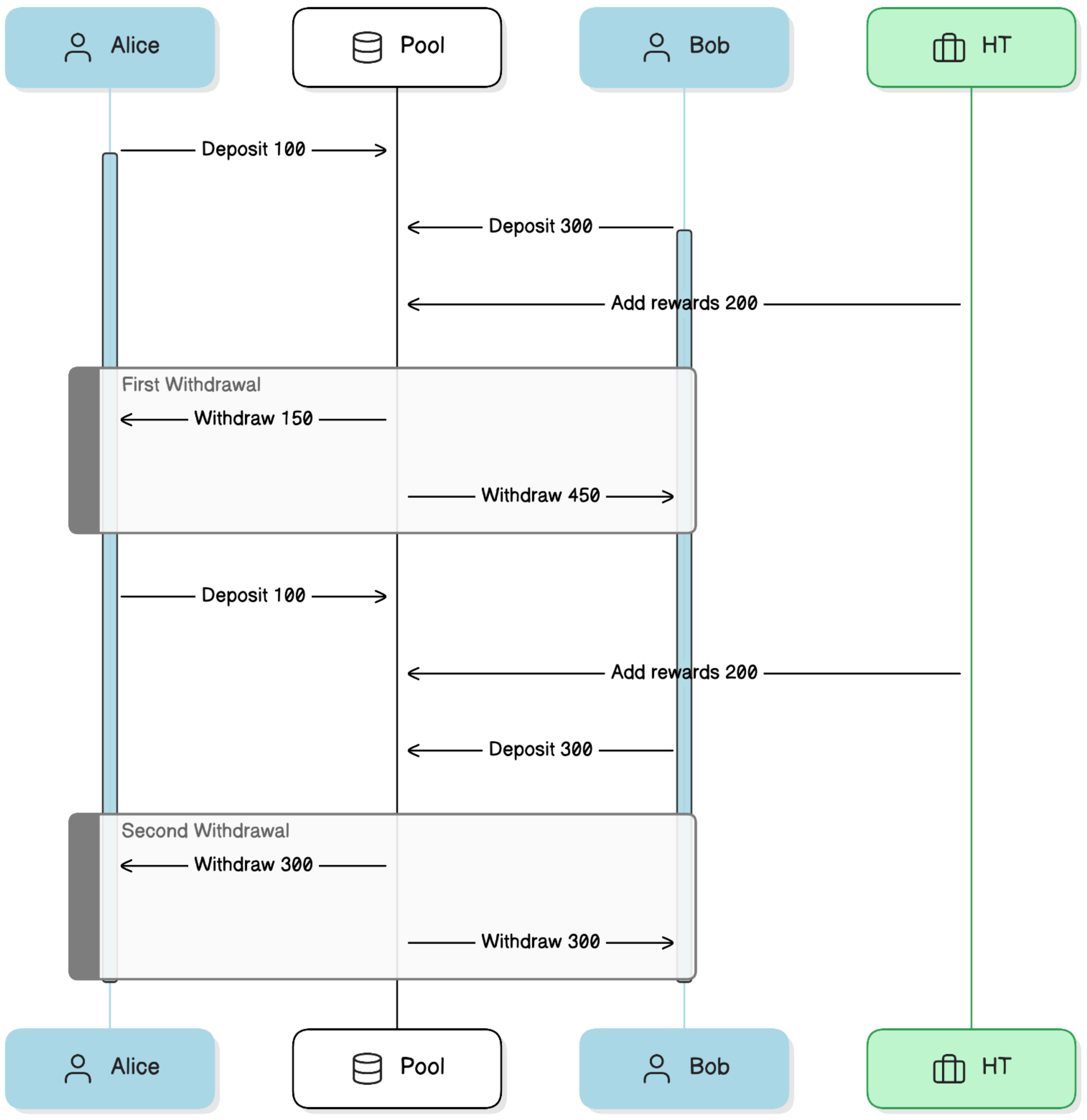
Example:

Imagine a scenario with users Alice and Bob, and the HarvestTeam (HT).

Alice contributes 100 and Bob adds 300, bringing the total in the pool to 400. This means Alice holds 25% of the pool, and Bob has 75%. When HT deposits 200 as rewards, Alice is eligible to withdraw 150, while Bob can take out 450.

In another sequence of events, suppose Alice makes a deposit first, then HT contributes rewards, followed by Bob's deposit. If Alice withdraws next and Bob withdraws afterward, Alice would receive her initial deposit plus all the rewards. Bob, however, would only retrieve his deposit, as he contributed after the rewards were distributed to the pool.

Pool and Reward Distribution Example Scenario



Goal

Design and develop a set of smart contracts for STLHarvest, incorporating any necessary assumptions to progress with the project.

2) Write tests

Ensure thorough testing of all your code to verify its functionality and reliability. The test cases **should** include, at minimum, all the scenarios proposed in this challenge.

3) Deploy your contracts in a public testnet

Deploy the contracts on an **Ethereum testnet of your choice** and maintain a record of the deployment address. Additionally, verify the contracts on Etherscan for transparency and accessibility.

4) Deploy your contracts in a local testnet

Deploy the contracts on a local instance of Ethereum using HardHat. The solution must include a Dockerfile to facilitate the deployment process.

5) Interact with the contract

Create a Hardhat task to query and display the total amount of STL tokens held in the contract.

Additional Requirements

1) **Development Environment**: Utilize Hardhat as the primary development framework for the project.
2) **Project structure**: Provide a clear project structure, separating each core module to provide readability to the code.
3) **Token Asset**: Instead of using Ethereum (ETH), create a specific token asset (STL) for this challenge. This asset will be used for deposits, reward distribution, and withdrawals.
4) **Docker Integration**: Provide a Dockerfile to facilitate the deployment process, ensuring that the project environment is easily reproducible.