Smart Contract Challenge

A) Challenge

1) Setup a project and create a contract

Summary

ETHPool provides a service where people can deposit ETH and they will receive weekly rewards. Users must be able to take out their deposits along with their portion of rewards at any time. New rewards are deposited manually into the pool by the ETHPool team each week using a contract function.

Requirements

- Only the team can deposit rewards.
- Deposited rewards go to the pool of users, not to individual users.
- Users should be able to withdraw their deposits along with their share of rewards considering the time when they deposited.

Example:

> Let say we have user **A** and **B** and team **T**.
>
> **A** deposits 100, and **B** deposits 300 for a total of 400 in the pool. Now **A** has 25% of the pool and **B** has 75%. When **T** deposits 200 rewards, **A** should be able to withdraw 150 and **B** 450.
>
> What if the following happens? **A** deposits then **T** deposits then **B** deposits then **A** withdraws and finally **B** withdraws. **A** should get their deposit + all the rewards. **B** should only get their deposit because rewards were sent to the pool before they participated.

Goal

Design and code a contract for ETHPool, take all the assumptions you need to move forward.

You can use any development tools you prefer: Hardhat, Truffle, Brownie, Solidity, Vyper.

Useful resources:

- Solidity Docs: https://docs.soliditylang.org/en/v0.8.4
- Educational Resource: https://github.com/austintgriffith/scaffold-eth
- Project Starter: https://github.com/abarmat/solidity-starter

2) Write tests

Make sure that all your code is tested properly

3) Deploy your contract

Deploy the contract to any Ethereum testnet of your preference. Keep record of the deployed address.

Bonus:

- Verify the contract in Etherscan

4) Interact with the contract

Create a script (or a Hardhat task) to query the total amount of ETH held in the contract.

You can use any library you prefer: Ethers.js, Web3.js, Web3.py, eth-brownie