Task: WETH Vault dApp

Objective

Create a decentralized application (dApp) where users can deposit WETH (Wrapped Ether) and earn rewards based on their investment amounts. This application will involve both Solidity smart contract development and a React frontend to interact with the smart contract. The task is designed to be completed within 8-10 hours and will test both Solidity and React skills.

Requirements

1. Smart Contract:
    - Create a **Vault** smart contract.
    - The contract should allow users to:
        - Deposit WETH into the contract.
        - Withdraw their WETH along with the accrued rewards.
    - The contract should:
        - Initialize a predefined reward pool of WETH.
        - Set a reward percentage of 5% per annum, pro-rated based on deposit duration (e.g., 2.5% for 6 months).
        - Track each user's WETH deposits.
        - Track multiple deposits separately by duration.
        - Calculate rewards based on deposit amount and duration.
    - Implement functions for:
        - depositRewards(uint256 amount): Allows the contract owner to deposit WETH into the reward pool.
        - setRewardPercentage(uint256 percentage): Allows the contract owner to set the reward percentage.
        - deposit(uint256 amount): Users can deposit WETH.
        - withdraw(): Users can withdraw their WETH and rewards.
        - calculateReward(address user): Calculate the reward for a user based on their deposit and duration.
    - Include tests for smart contract to ensure functionality.
    - Deploy on a Testnet.

2. React Frontend:
    - Create a React application that interacts with the Vault smart contract.
    - The frontend should allow users to:
        - Connect their Ethereum wallet (e.g., using MetaMask).
        - Deposit WETH into the contract.
        - View their current deposit amount and accrued rewards.
        - Withdraw their WETH and rewards.

- Deploy frontend

3. Bonus Points:
    - Great UX - styling, input validation, error handling
    - Comprehensive tests for the smart contract
    - Use of oracles to provide the overall value of the portfolio in USD or another stable currency.

Deliverables

1. GitHub repo with source code and README
    - README should provide an overview of the project, setup instructions and any other relevant information
2. Deployment Details
    - Address of the deployed smart contract on a testnet
    - And the URL where the React application can be accessed

Evaluation Criteria

- Focus on quality over quantity. Try to finish as much as you can manage to do in the best manner possible. It’s totally fine if you couldn't finish the entire task.
    - Code quality and organization.
    - Functionality and correctness of the smart contract.
    - User experience and responsiveness of the React application.