Position: Blockchain Developer ***DEADLINE: 96 Hrs***

QUALIFICATION TASK: Blockchain Developer Task

Task 1: Lock and Release Contract on a Blockchain Bridge

Overview:

Develop a blockchain bridge featuring a lock and release mechanism to facilitate the transfer of assets between an EVM-compatible chain and a non-EVM chain. The bridge should support locking any ERC20 or native token on the EVM side and releasing a corresponding token on the non-EVM side, functioning as a basic centralized bridge.

Objectives:

EVM Side (Lock Contract):

- Implement a smart contract on an EVM-compatible chain (e.g., Ethereum, BSC) that can lock ERC20 tokens or native ETH.
- Ensure the contract emits events upon locking tokens, capturing details such as the amount, the sender's address, and the target chain/address.

Non-EVM Side (Release Contract):

- Implement a corresponding smart contract on a non-EVM chain (e.g., Solana, Tezos) that releases a pre-defined or equivalent token once it receives confirmation of the lock event from the EVM side.
- This contract should verify the authenticity of lock events, possibly through oracles or a centralized verification system.

Centralized Bridge Logic:

- Develop a backend service or use smart contract logic to monitor lock events on the EVM chain and trigger the release process on the non-EVM chain.
- Ensure secure and verifiable communication between chains.

Requirements:

- Support for ERC20 and native token locks.
- Event emission for tracking and verification purposes.
- Security measures against common vulnerabilities (e.g., reentrancy, overflow/underflow).
- A mechanism for refund or reversal in case of an error or failed cross-chain transfer.
- Documentation explaining the setup, deployment, and interaction with contracts.

Points to Consider:

- Gas optimization techniques.
- Security practices to prevent unauthorized access or manipulation.
- Handling of decimal differences between tokens across chains.
- Error handling and user feedback mechanisms.

Task 2: Proxy Contract for Load Balancing and Function Delegation

Overview:

Design and implement a proxy smart contract that functions as a load balancer for other contracts. It should act as the entry point for all requests, delegating them to specific implementation contracts based on the function requirement.

Objectives:

Proxy Contract Design:

- Create a proxy contract that maintains a registry of implementation addresses for different functionalities (e.g., token transfers, staking, voting).
- Implement a fallback function that takes a function ID as input and delegates the call to the corresponding contract address.

Function Delegation:

- Ensure the proxy contract can forward requests to the appropriate implementation contract along with any necessary data.
- Support updates to implementation addresses without disrupting the proxy contract or requiring redeployment.

Security and Efficiency:
Incorporate security measures to prevent unauthorized updates to the contract registry.
Optimize for gas efficiency and minimize execution costs for delegated functions.

Requirements:

- Dynamic registry of function IDs and corresponding contract addresses.
- Delegation mechanism that supports all types of function calls and data passing.
- Mechanisms for updating, adding, and removing addresses from the registry.

- Documentation detailing the architecture, setup, and interaction with the proxy contract.

Points to Consider:

- Upgradeability and maintenance of the contract registry.
- Security practices, including role-based access control for registry updates.
- Impact on transaction costs and efficiency when using the proxy for delegation.
- Testing strategies to ensure correct function delegation and execution.

Submission Guidelines:

- Choose and complete at least one of the tasks mentioned above within 96 hours.
- Document your development process, including design decisions, challenges faced, and how they were addressed.
- Deploy your contracts to a testnet and provide a detailed guide for interacting with them.
- Create a public GitHub repository with the title/description as Wasserstoff: Task 1/Task 2 (2024 Blockchain Interviews) to host your project code and documentation.
- Ensure your code is well-commented, and include unit tests to demonstrate the functionality and security of your contracts.
- This project aims to assess your skills in smart contract development, understanding of blockchain architectures, and ability to implement secure, efficient, and scalable blockchain solutions.