[3 hours]

Problem statement

Objective: Solidity + web3 assessment.
Recommended tools: IDE, Solidity, Ethers/async-js, hardhat, EIP 2535 diamond pattern for upgrade

Part I

1. Create a contract **A** with a getter & a setter function.
2. Create a uint variable, which increases in value every time you call the setter function. The value increase is the same as the value provided in the setter function. While, when you call the getter function, it returns the value set in the uint variable.
3. Implement reentrancy guard & admin access for both functions.
4. Deploy to Binance smart chain testnet.
5. Deploy this contract through proxy upgradeability.

PART II

1. Create a contract B, that serves as an access registry. This contract serves as a ledger of the admin details(role, wallet address) of contract A.
2. Contract B must have addAdmin, removeAdmin, transferAdminRole, and renounceAdminRole.
3. Contract B also has its admin address. Let u2s call it superAdmin.
4. You can use OpenZeppelin's access.sol contract, and customize it to your needs.

Part III

1. Create a simple dapp that uses web3modal or wallet connect to connect user wallets.
2. Create an interface to enable the below interactions
    a. Button-getter function.
    b. Input field + button: Setter function.
    c. Input field + button: Contract upgradeability.

Part IV

1. Function call
    1. calling the getter function in contract A must return 0.
    2. call the setter function with an input of 10.
    3. Now call the getter function. It must return the value 10.
2. Fetch the admin address of contract A

3. Now, upgrade contract A so that it inherits contract B.
4. Change the admin address of contract A to some other address from the access registry.
5. Fetch the admin address of contract A. Should be different from the previous admin address.
6. Function call
    1. calling the getter function must return 10.
    2. call the setter function with an input of 81
7. Now, call the getter function, it must return 91.