SCE Take Home Exercise - 1

Exercise:

The goal of this exercise is to design and implement a SP NFT (ERC-721) with different metadata revealing approaches. The solution should leverage Chainlink for random number generation and allow for two distinct revealing approaches, with the potential to support future approaches.

- You have 7 days to finish the take home exercise.
- You will be invited to a channel in Slack to talk to the team members.
- You are expected to discuss the design/code of the exercise during the interview process.

Requirements:

1. **Revealing approaches**: Implement two revealing approaches below, and the operator of SP NFT contract should be able to choose different revealing approaches before revealing started.

   1**.1 In-Collection Revealing**: The SP NFT and the revealed SP NFT reside in the same ERC-721 smart contract. The revealing approach switches the SP NFT's metadata to another set, effectively transforming it into the revealed NFT.

   1**.2 Separate Collection Revealing**: The SP NFT and the revealed SP NFT are stored in separate ERC-721 smart contracts. When revealing, the system burns the SP NFT, mints a new NFT in the revealed SP NFT smart contract, and transfers it to the end user.

2. **On-Chain Metadata**: All metadata are stored on-chain. The metadata should be returned and generated on the fly within the `tokenURI()` function. The metadata standard is here
3. **Chainlink Integration**: You should integrate Chainlink to generate random numbers. This involves setting up a Chainlink VRF Coordinator, a Key Hash, and a fee. Upon receiving the random number, the system should select the corresponding metadata.

4. **Purchase and Return**: Buy the SP NFT with Ether and return excessive fund if the final mint price is lower than the purchase price.
5. (bonus) **Staking and Claim**: Stake the revealed SP NFT to earn 5% APY in ERC20. Owner can claim the rewards at any time.

Deliverables:

- A private repo with all the code/doc, please invite `LeoHChen`, `Ramarti`, `kingster-will`, `leeren` as collaborators.
- A design document with detailed explanation for each contract about its role, how it integrates with the other contracts, and how it implements its intended functionality.
- A README file with setup and usage instructions.
- Deployed smart contract on a TestNet (Goerli)

Note:

- Remember to write well-documented code, including function/contract level comments.
- Try to optimize for gas where possible.
- Consider edge cases and add necessary checks in your contracts.
- Contracts should be developed using Solidity and you should use Foundry or Hardhat for testing.
- Consider contract security and avoid known vulnerabilities.

Hints:

Look into Chainlink's documentation for how to work with their VRF in your contracts. ERC721 standard and OpenZeppelin library will be beneficial for implementing the NFT related features.