Homework 6 - Creating an ERC721

ERC721 Recap:

The ERC721 contract is a standard for non-fungible tokens. Non-fungible tokens allow us to tokenize unique assets - digital and physical.

In ERC721, tokens can be minted and burned, and each token should have a unique ID.

1. In Remix or Hardhat, create a new file called `VolcanoToken.sol`.
2. Import Open Zeppelin's ERC721 and Ownable contract.
   Have a look at the ERC721 standard and Open Zeppelin's implementation of this.
   https://eips.ethereum.org/EIPS/eip-721 (https://eips.ethereum.org/EIPS/eip-721)
   https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC721/ERC721.sol
   (https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC721/ERC721.sol)
3. Create a `contract` called `VolcanoToken` that inherits `ERC721` and `Ownable`.
4. After the contract is deployed, the tokens can be minted, burned and transferred to users. To identify the token we need a token ID. Declare a `uint256` for the token ID.
5. Make a struct for each token's metadata. The struct should have a timestamp, the token ID and a `string` for the tokenURI.

Token URI:

A token URI provides additional metadata about a token. Each URI should be unique to the token.
In many cases, IPFS is used to store this file.

6. Create a public record for token ownership with a `mapping` of the users `address` to an array of structs.

Internal ERC721 functions:

Similar to ERC20, there are internal ERC721 functions available to the contract it's defined in.

Useful methods are `_safeMint` and `_burn` ,

7. Make a function that mints a token to a user with the token ID. You can use the struct from part 5 to store the metadata for the token, for example
   `StructName memory newTokenData = StructName(...)`

   Use `block.timestamp` for timestamp and any `string` for the tokenURI.

   Store the struct to the user's record.

   Increment the token ID.
8. Make a function that burns tokens, taking the token ID.
9. Make an internal function that loops over the array of structs and removes the burned tokenID. Call the function inside your burn function.
10. Add a require statement to ensure that only the owner of the token can burn the token.
11. Have a look at the implementation of function tokenURI in the parent Open Zeppelin contract, make the necessary changes to your contract to return the correct tokenURI for a particular token ID.
12. We need to remove the token from the mapping. Make an function that deletes the token from the mapping. You can make this an internal function, which can then be called within the burn function.

Resources

https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC721/ERC721.sol
(https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC721/ERC721.sol)