* In the Task, I was asked to create a smart contract in which a person can deposit the ETH token and can create a NFT, and then later he can also sell his NFT back and get the token back.
* So there are two function in the smart contract one for minting and another for withdrawing the ETH token.

//SPDX-License-Identifier: MIT

pragma solidity ^0.8.4;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";

import "@openzeppelin/contracts/access/Ownable.sol";

import "@openzeppelin/contracts/utils/Counters.sol";

contract MNFT is ERC721, Ownable {

using Counters for Counters.Counter;

Counters.Counter private \_tokenIdCounter;

constructor() ERC721("mNFT", "MFT") {} //Constructor for ERC721

mapping(uint=>uint) public idToAmount

function \_baseURI() internal pure override returns (string memory) {

return "www.nft.com/tokens/";

}

function safeMint(address to) public payable returns(uint) {

require(msg.value> 0.01 ether);

uint256 tokenId = \_tokenIdCounter.current();

idToAmount[tokenId] = msg.value;

\_tokenIdCounter.increment();

\_safeMint(to, tokenId);

return tokenId;

}

function withdraw(uint tokenId) public payable{

require(ownerOf(tokenId) == msg.sender, "Sorry you are not the correct owner");

safeTransferFrom(msg.sender, owner(), tokenId);

payable(msg.sender).transfer(idToAmount[tokenId]);

idToAmount[tokenId] = 0;

}

}

/\*

In the code there is a idToAmount value which just simpy stores the amount contributed by the minter on each token id.

1. There is a baseURI function which is helping to generate the link for storing metadata of NFT.
2. when someone mints the NFT, he enters the address which will be the owner of the NFT has to deposit a minimum amount of 0.01 ether.So for each NFT a unique token id is generated.
3. And when the owner withdraws fund for his NFT, we firstly check if she is the true owner of the given tokenid and then we transfer the ownership of the NFT to address of the smart contract manager. And after that amount deposited by the minter is transferred to the owner and the idToAmount mapping is reset to 0.

Test cases

1) the amount of the person minting the smart contract will decrease by same amount he contruibuted to mint the NFT,

and the balance of smart contract will increase by the same amount.

2)After the minting the NFT, we can check the owner of the returned tokenid as same the person specified to be owner of the NFT.

3)we can check the ownership of NFT after withdrawing money and it will be address of the manager of the smart contract.

4) After withdrawing the amount on idToAmount mapping will reset to 0.

\*/

Link for the rinkeby test Network Contract: https://rinkeby.etherscan.io/address/0xc2bb8a570e4b048095c86261080d2adf0de06858