Question 1) We need you to develop the solid-state blockchain environment for NFC trading. Read about it and try to make one small module in solidity.

Answer )

// contracts/GameItems.sol

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.7;

import "@openzeppelin/contracts/token/ERC1155/ERC1155.sol";

contract NftTrader {

mapping(address => mapping(uint256 => Listing)) public listings;

mapping(address => uint256) public balances; // Added to track balances

struct Listing {

uint256 price;

address seller;

}

function addListing(uint256 price, address contractAddr, uint256 tokenId) public {

ERC1155 token = ERC1155(contractAddr);

require(token.balanceOf(msg.sender, tokenId) > 0, "caller must own given token");

require(token.isApprovedForAll(msg.sender, address(this)), "contract must be approved");

listings[contractAddr][tokenId] = Listing(price, msg.sender);

}

function purchase(address contractAddr, uint256 tokenId, uint256 amount) public payable {

Listing memory item = listings[contractAddr][tokenId];

require(msg.value >= item.price \* amount, "insufficient funds sent");

balances[item.seller] += msg.value;

ERC1155 token = ERC1155(contractAddr);

token.safeTransferFrom(item.seller, msg.sender, tokenId, amount, "");

}

function withdraw(uint256 amount, address payable destAddr) public {

require(amount <= balances[msg.sender], "insufficient funds");

destAddr.transfer(amount);

balances[msg.sender] -= amount;

}

}

I tried to make an improved version of this (using suggestions from Chatgpt :

// contracts/GameItems.sol

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.7;

import "@openzeppelin/contracts/token/ERC1155/ERC1155.sol";

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

import "@openzeppelin/contracts/security/ReentrancyGuard.sol";

contract NftTrader is Ownable, ReentrancyGuard {

mapping(address => mapping(uint256 => Listing)) public listings;

mapping(address => uint256) public balances; // Added to track balances

struct Listing {

uint256 price;

address seller;

}

constructor() Ownable(msg.sender) ReentrancyGuard() {

// The constructor of Ownable takes msg.sender as the initial owner

}

function addListing(uint256 price, address contractAddr, uint256 tokenId) public {

ERC1155 token = ERC1155(contractAddr);

require(token.balanceOf(msg.sender, tokenId) > 0, "caller must own given token");

require(token.isApprovedForAll(msg.sender, address(this)), "contract must be approved");

listings[contractAddr][tokenId] = Listing(price, msg.sender);

}

function purchase(address contractAddr, uint256 tokenId, uint256 amount) public payable nonReentrant {

Listing memory item = listings[contractAddr][tokenId];

require(msg.value >= item.price \* amount, "insufficient funds sent");

balances[item.seller] += msg.value;

ERC1155 token = ERC1155(contractAddr);

token.safeTransferFrom(item.seller, msg.sender, tokenId, amount, "");

}

function withdraw(uint256 amount, address payable destAddr) public nonReentrant {

require(amount <= balances[msg.sender], "insufficient funds");

balances[msg.sender] -= amount;

destAddr.transfer(amount);

}

}