Smart Contract for //SPDX-License-Identifier:MIT

pragma solidity ^0.8.0;

// Importing OpenZeppelin contracts

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

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

contract Fishnet is ERC20, Ownable {

// Variables to store fishing vessel data

struct FishingVessel {

string name;

string safetyData;

string logBook;

}

// Variables to store warehouse data

struct Warehouse {

string Aaddress;

string temperature;

string dataManagement;

string coldStorage;

}

// Variables to store payment data

struct Payment {

uint256 amount;

bool received;

}

// Variables to store traceability data

struct Traceability {

string QRCode;

address[] truckCompanies;

address retailer;

address endUser;

}

// Mapping to store fishing vessel data by vessel ID

mapping(uint256 => FishingVessel) public fishingVessels;

// Mapping to store warehouse data by warehouse ID

mapping(uint256 => Warehouse) public warehouses;

// Mapping to store payment data by payment ID

mapping(uint256 => Payment) public payments;

// Mapping to store traceability data by traceability ID

mapping(uint256 => Traceability) public traceability;

// Event to log new fishing vessel data

event FishingVesselAdded(uint256 indexed id, string name);

// Event to log new warehouse data

event WarehouseAdded(uint256 indexed id, string Aaddress);

// Event to log new payment data

event PaymentAdded(uint256 indexed id, uint256 amount);

// Event to log new traceability data

event TraceabilityAdded(uint256 indexed id, string QRCode);

// Constructor function to initialize Fishnet token

constructor() ERC20("Fishnet", "FISH") {}

// Function to add fishing vessel data

function addFishingVessel(uint256 id, string memory name, string memory safetyData, string memory logBook) public onlyOwner {

fishingVessels[id] = FishingVessel(name, safetyData, logBook);

emit FishingVesselAdded(id, name);

}

// Function to add warehouse data

function addWarehouse(uint256 id, string memory Aaddress, string memory temperature, string memory dataManagement, string memory coldStorage) public onlyOwner {

warehouses[id] = Warehouse(Aaddress, temperature, dataManagement, coldStorage);

emit WarehouseAdded(id, Aaddress);

}

// Function to add payment data

function addPayment(uint256 id, uint256 amount) public onlyOwner {

payments[id] = Payment(amount, false);

emit PaymentAdded(id, amount);

}

// Function to mark payment as received

function receivePayment(uint256 id) public onlyOwner {

require(payments[id].amount > 0, "Payment does not exist");

require(!payments[id].received, "Payment has already been received");

payments[id].received = true;

}

// Function to add traceability data

function addTraceability(uint256 id, string memory QRCode, address[] memory truckCompanies, address retailer, address endUser) public onlyOwner {

traceability[id] = Traceability(QRCode, truckCompanies, retailer, endUser);

emit TraceabilityAdded(id, QRCode);

}

}