pragma solidity >=0.4.21 <0.9.0;

contract supplyChain {

    uint32 public product\_id = 0; // Product ID

    uint32 public participant\_id = 0; // Participant ID

    uint32 public owner\_id = 0; // Ownership ID

    struct product {

        string modelNumber;

        string partNumber;

        string serialNumber;

        address productOwner;

        uint32 cost;

        uint32 mfgTimeStamp;

    }

    mapping(uint32 => product) public products;

    struct participant {

        string userName;

        string password;

        string participantType;

        address participantAddress;

    }

    mapping(uint32 => participant) public participants;

    struct ownership {

        uint32 productId;

        uint32 ownerId;

        uint32 trxTimeStamp;

        address productOwner;

    }

    mapping(uint32 => ownership) public ownerships; // ownerships by ownership ID (owner\_id)

    mapping(uint32 => uint32[]) public productTrack; // ownerships by Product ID (product\_id) / Movement

    //track for a product

    event TransferOwnership(uint32 productId);

    function addParticipant(string memory \_name, string memory \_pass, address \_pAdd, string memory \_pType) public returns (uint32){

        uint32 userId = participant\_id++;

        participants[userId].userName = \_name;

        participants[userId].password = \_pass;

        participants[userId].participantAddress = \_pAdd;

        participants[userId].participantType = \_pType;

        return userId;

    }

    function getParticipant(uint32 \_participant\_id) public view returns (string memory,address,string memory) {

        return (participants[\_participant\_id].userName,

        participants[\_participant\_id].participantAddress,

        participants[\_participant\_id].participantType);

    }

    function addProduct(uint32 \_ownerId,string memory \_modelNumber,string memory \_partNumber,string

    memory \_serialNumber,uint32 \_productCost) public returns (uint32) {

        if(keccak256(abi.encodePacked(participants[\_ownerId].participantType)) == keccak256("Manufacturer")) {

        uint32 productId = product\_id++;

        products[productId].modelNumber = \_modelNumber;

        products[productId].partNumber = \_partNumber;

        products[productId].serialNumber = \_serialNumber;

        products[productId].cost = \_productCost;

        products[productId].productOwner = participants[\_ownerId].participantAddress;

        products[productId].mfgTimeStamp = uint32(block.timestamp);

        return productId;

        }

        return 0;

    }

    modifier onlyOwner(uint32 \_productId) {

        require(msg.sender == products[\_productId].productOwner,"");

        \_;

    }

    function getProduct(uint32 \_productId) public view returns (string memory,string memory,string

    memory,uint32,address,uint32){

        return (products[\_productId].modelNumber,

        products[\_productId].partNumber,

        products[\_productId].serialNumber,

        products[\_productId].cost,

        products[\_productId].productOwner,

        products[\_productId].mfgTimeStamp);

    }

    function newOwner(uint32 \_user1Id,uint32 \_user2Id, uint32 \_prodId) onlyOwner(\_prodId) public returns (bool) {

        participant memory p1 = participants[\_user1Id];

        participant memory p2 = participants[\_user2Id];

        uint32 ownership\_id = owner\_id++;

        if(keccak256(abi.encodePacked(p1.participantType)) == keccak256("Manufacturer")&&

        keccak256(abi.encodePacked(p2.participantType))==keccak256("Supplier")){

            ownerships[ownership\_id].productId = \_prodId;

            ownerships[ownership\_id].productOwner = p2.participantAddress;

            ownerships[ownership\_id].ownerId = \_user2Id;

            ownerships[ownership\_id].trxTimeStamp = uint32(block.timestamp);

            products[\_prodId].productOwner = p2.participantAddress;

            productTrack[\_prodId].push(ownership\_id);

            emit TransferOwnership(\_prodId);

            return (true);

        }

        else if(keccak256(abi.encodePacked(p1.participantType)) == keccak256("Supplier") &&

        keccak256(abi.encodePacked(p2.participantType))==keccak256("Supplier")){

            ownerships[ownership\_id].productId = \_prodId;

            ownerships[ownership\_id].productOwner = p2.participantAddress;

            ownerships[ownership\_id].ownerId = \_user2Id;

            ownerships[ownership\_id].trxTimeStamp = uint32(block.timestamp);

            products[\_prodId].productOwner = p2.participantAddress;

            productTrack[\_prodId].push(ownership\_id);

            emit TransferOwnership(\_prodId);

            return (true);

        }

        else if(keccak256(abi.encodePacked(p1.participantType)) == keccak256("Supplier") &&

        keccak256(abi.encodePacked(p2.participantType))==keccak256("Consumer")){

            ownerships[ownership\_id].productId = \_prodId;

            ownerships[ownership\_id].productOwner = p2.participantAddress;

            ownerships[ownership\_id].ownerId = \_user2Id;

            ownerships[ownership\_id].trxTimeStamp = uint32(block.timestamp);

            products[\_prodId].productOwner = p2.participantAddress;

            productTrack[\_prodId].push(ownership\_id);

            emit TransferOwnership(\_prodId);

            return (true);

        }

        return (false);

    }

    function getProvenance(uint32 \_prodId) external view returns (uint32[] memory) {

        return productTrack[\_prodId];

    }

    function getOwnership(uint32 \_regId) public view returns (uint32,uint32,address,uint32) {

        ownership memory r = ownerships[\_regId];

        return (r.productId,r.ownerId,r.productOwner,r.trxTimeStamp);

    }

    function authenticateParticipant(uint32 \_uid,string memory \_uname,string memory \_pass,string memory

    \_utype) public view returns (bool){

        if(keccak256(abi.encodePacked(participants[\_uid].participantType)) ==

        keccak256(abi.encodePacked(\_utype))) {

            if(keccak256(abi.encodePacked(participants[\_uid].userName)) ==

            keccak256(abi.encodePacked(\_uname))) {

                if(keccak256(abi.encodePacked(participants[\_uid].password)) ==

                keccak256(abi.encodePacked(\_pass))) {

                return (true);

                }

            }

        }

        return (false);

    }

}