**Source Code of** **Decentralized Patient Record Storage**

**A Usecase of Application of Blockchain**

**In The Field Of Medical Science**

pragma solidity ^0.8.0;

contract PatientRecord {

    // Structure to represent a patient

    struct Patient {

        uint id;

        string name;

        uint age;

        string[] diseases;

    }

    // Structure to represent a doctor

    struct Doctor {

        uint id;

        string name;

        string qualification;

    }

    // Structure to represent a medicine

    struct Medicine {

        uint id;

        string name;

        string expiryDate;

        string dose;

        uint price;

    }

    mapping(address => Doctor) public doctors;

    mapping(address => Patient) public patients;

    mapping(uint => Medicine) public medicines;

// Events for logging activities

    event PatientAdded(address patientAddress, uint id, string

name,uint age);

    event DiseaseAdded(address patientAddress, string disease);

    event MedicineAdded(uint id, string name, uint price);

    event MedicinePrescribed(uint medicineId, address patientAddress);

    event AgeUpdated(address patientAddress, uint newAge);

    event DoctorRegistered(uint id, string name,string qualification);

// Function to register a new patient

    function registerPatient(string memory \_name, uint \_age) public {

        address patientAddress = msg.sender;

        patients[patientAddress] = Patient({

            id: block.timestamp,

            name: \_name,

            age: \_age,

            diseases: new string[](0)

        });

        emit PatientAdded(patientAddress, block.timestamp,

\_name,\_age);

    }

    // Function to register a new doctor

    function registerDoctor(string memory \_name,

string memory \_qualification) public {

        address doctorAddress = msg.sender;

        doctors[doctorAddress] = Doctor({

            id: block.timestamp,

            name: \_name,

            qualification: \_qualification

     });

     emit DoctorRegistered(block.timestamp, \_name, \_qualification);

    }

    // Function to add disease to a patient

    function addDisease(string memory \_disease) public {

        patients[msg.sender].diseases.push(\_disease);

        emit DiseaseAdded(msg.sender, \_disease);

    }

// Function to add medicine to the ledger

    function addMedicine(uint \_id, string memory \_name,

string memory \_expiryDate, string memory \_dose,

uint \_price) public {

        medicines[\_id] = Medicine({

            id: \_id,

            name: \_name,

            expiryDate: \_expiryDate,

            dose: \_dose,

            price: \_price

        });

     emit MedicineAdded(\_id, \_name, \_price);

    }

    // Function to prescribe medicine to a patient

    function prescribeMedicine(uint \_medicineId,

address \_patientAddress) public {

    require(medicines[\_medicineId].id != 0,

"Medicine does not exist");

    emit MedicinePrescribed(\_medicineId, \_patientAddress);

    }

    // Function to update patient's age

    function updateAge(uint \_newAge) public {

        patients[msg.sender].age = \_newAge;

        emit AgeUpdated(msg.sender, \_newAge);

    }

}