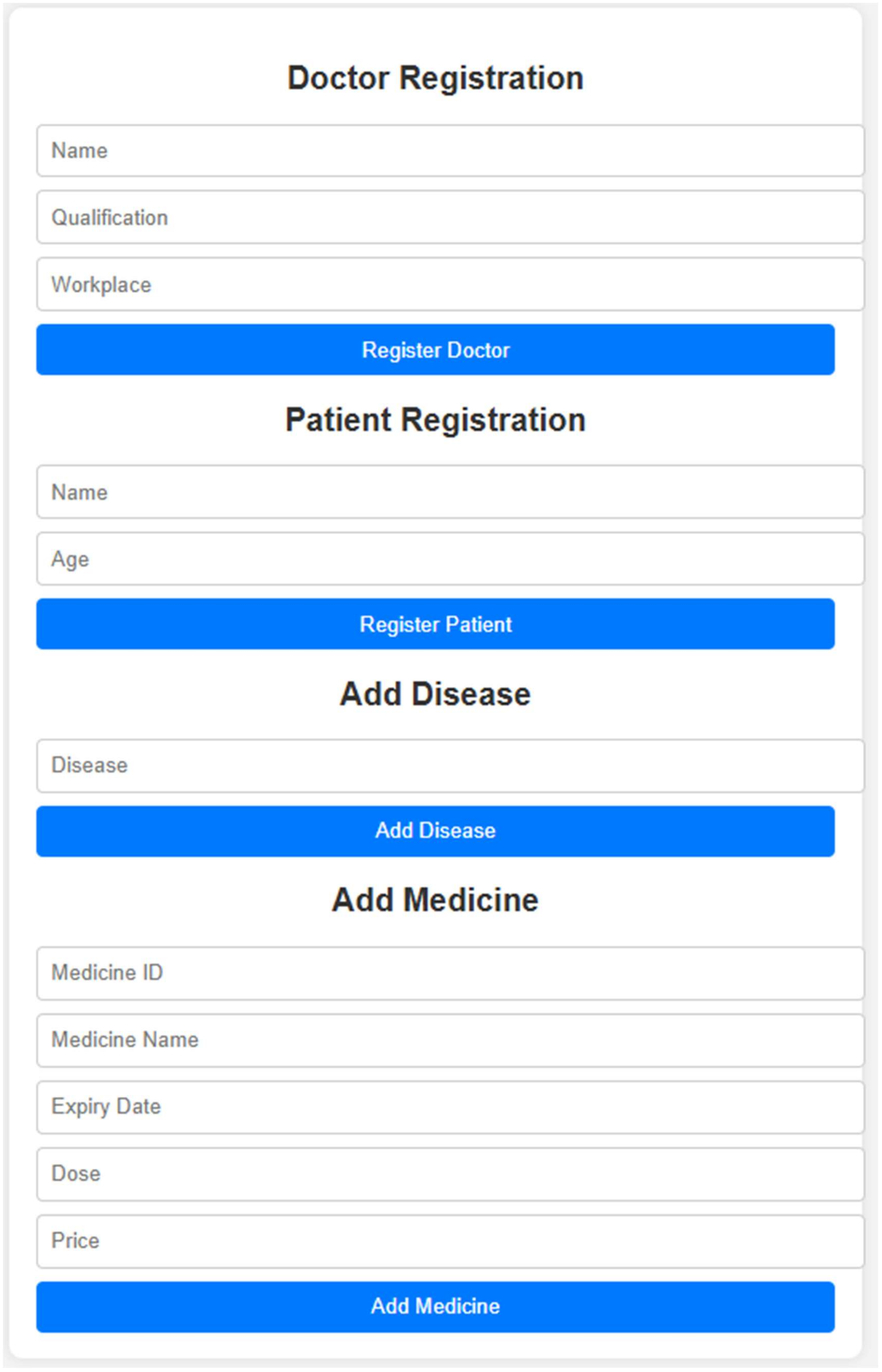
DecentralizedPatientRecord.sol

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract DecentralizedPatientRecords {
    struct Doctor {
        uint id;
        string name;
        string qualification;
        string workPlace;
        bool registered;
    }

    struct Patient {
        uint id;
        string name;
        uint age;
        string[] diseases;
        bool registered;
    }

    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;
    mapping(address => uint[]) public prescribedMedicines;

    uint public doctorCount;
    uint public patientCount;
    uint public medicineCount;

    modifier onlyRegisteredDoctor() {
        require(doctors[msg.sender].registered, "Only registered doctors can perform this action");
        _;
    }

    modifier onlyRegisteredPatient() {
        require(patients[msg.sender].registered, "Only registered patients can perform this action");
        _;
    }
    function registerDoctor(string memory _name, string memory _qualification, string
memory _workPlace) public {
        require(!doctors[msg.sender].registered, "Doctor already registered");
        doctorCount++;
        doctors[msg.sender] = Doctor(doctorCount, _name, _qualification, _workPlace,
true);
    }

    function registerPatient(string memory _name, uint _age) public {
        require(!patients[msg.sender].registered, "Patient already registered");
        patientCount++;
        patients[msg.sender] = Patient(patientCount, _name, _age, new string[](0),
true);
    }

    function addDisease(string memory _disease) public onlyRegisteredPatient {
        patients[msg.sender].diseases.push(_disease);
    }

    function addMedicine(uint _id, string memory _name, string memory _expiryDate,
string memory _dose, uint _price) public {
        require(medicines[_id].id == 0, "Medicine ID already exists");
        medicines[_id] = Medicine(_id, _name, _expiryDate, _dose, _price);
        medicineCount++;
    }

    function prescribeMedicine(uint _id, address _patient) public onlyRegisteredDoctor
{
        require(patients[_patient].registered, "Patient not registered");
        require(medicines[_id].id != 0, "Medicine does not exist");
        prescribedMedicines[_patient].push(_id);
    }

    function updateAge(uint _age) public onlyRegisteredPatient {
        patients[msg.sender].age = _age;
    }

    function viewPatientData(address _patient) public view onlyRegisteredDoctor returns
(uint, string memory, uint, string[] memory) {
        require(patients[_patient].registered, "Patient not registered");
        Patient storage patient = patients[_patient];
        return (patient.id, patient.name, patient.age, patient.diseases);
    }

    function viewMedicineDetails(uint _id) public view returns (string memory, string
memory, string memory, uint) {
        require(medicines[_id].id != 0, "Medicine not found");
        Medicine storage medicine = medicines[_id];
        return (medicine.name, medicine.expiryDate, medicine.dose, medicine.price);
    }

    function viewPrescribedMedicines(address _patient) public view onlyRegisteredDoctor
returns (uint[] memory) {
        require(patients[_patient].registered, "Patient not registered");
        return prescribedMedicines[_patient];
    }

    function viewDoctorDetails(address _doctor) public view returns (uint, string
memory, string memory, string memory) {
        require(doctors[_doctor].registered, "Doctor not registered");
        Doctor storage doctor = doctors[_doctor];
        return (doctor.id, doctor.name, doctor.qualification, doctor.workPlace);
    }

    function deletePatientRecord(address _patient) public onlyRegisteredDoctor {
        require(patients[_patient].registered, "Patient not registered");
        delete patients[_patient];
    }

    function deleteDoctorRecord(address _doctor) public onlyRegisteredDoctor {
        require(doctors[_doctor].registered, "Doctor not registered");
        delete doctors[_doctor];
    }
}
```

DecentralizedPatientRecord.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Decentralized Patient Records</title>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/web3/1.7.5/web3.min.js"></script>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
            background-color: #f4f4f4;
        }
        .container {
            max-width: 600px;
            margin: auto;
            background: white;
            padding: 20px;
            border-radius: 10px;
            box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
        }
        h2 {
            text-align: center;
            color: #333;
        }
        input, button {
            display: block;
            margin-top: 10px;
            width: 100%;
            padding: 10px;
            border-radius: 5px;
            border: 1px solid #ccc;
            font-size: 16px;
        }
        button {
            background-color: #007bff;
            color: white;
            border: none;
            cursor: pointer;
        }
        button:hover {
            background-color: #0056b3;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>Doctor Registration</h2>
        <input type="text" id="docName" placeholder="Name">
        <input type="text" id="docQual" placeholder="Qualification">
        <input type="text" id="docWork" placeholder="Workplace">
        <button onclick="registerDoctor()">Register Doctor</button>

        <h2>Patient Registration</h2>
        <input type="text" id="patName" placeholder="Name">
        <input type="number" id="patAge" placeholder="Age">
        <button onclick="registerPatient()">Register Patient</button>

        <h2>Add Disease</h2>
        <input type="text" id="disease" placeholder="Disease">
        <button onclick="addDisease()">Add Disease</button>

        <h2>Add Medicine</h2>
        <input type="number" id="medId" placeholder="Medicine ID">
        <input type="text" id="medName" placeholder="Medicine Name">
        <input type="text" id="medExpiry" placeholder="Expiry Date">
        <input type="text" id="medDose" placeholder="Dose">
        <input type="number" id="medPrice" placeholder="Price">
        <button onclick="addMedicine()">Add Medicine</button>
    </div>

    <script>
        let contract;
        const contractAddress = "0xfd758D0F473d5f916C619411b1BCd8397f3C79eD";
        const contractABI = [
    {
        "inputs": [
            {
                "internalType": "string",
                "name": "_disease",
                "type": "string"
            }
        ],
        "name": "addDisease",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "uint256",
                "name": "_id",
                "type": "uint256"
            },
            {
                "internalType": "string",
                "name": "_name",
                "type": "string"
            },
            {
                "internalType": "string",
                "name": "_expiryDate",
                "type": "string"
            },
            {
                "internalType": "string",
                "name": "_dose",
                "type": "string"
            },
            {
                "internalType": "uint256",
                "name": "_price",
                "type": "uint256"
            }
        ],
        "name": "addMedicine",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "address",
                "name": "_doctor",
                "type": "address"
            }
        ],
        "name": "deleteDoctorRecord",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "address",
                "name": "_patient",
                "type": "address"
            }
        ],
        "name": "deletePatientRecord",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "uint256",
                "name": "_id",
                "type": "uint256"
            },
            {
                "internalType": "address",
                "name": "_patient",
                "type": "address"
            }
        ],
        "name": "prescribeMedicine",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "string",
                "name": "_name",
                "type": "string"
            },
            {
                "internalType": "string",
                "name": "_qualification",
                "type": "string"
            },
            {
                "internalType": "string",
                "name": "_workPlace",
                "type": "string"
            }
        ],
        "name": "registerDoctor",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "string",
                "name": "_name",
                "type": "string"
            },
            {
                "internalType": "uint256",
                "name": "_age",
                "type": "uint256"
            }
        ],
        "name": "registerPatient",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "uint256",
                "name": "_age",
                "type": "uint256"
            }
        ],
        "name": "updateAge",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "doctorCount",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "address",
                "name": "",
                "type": "address"
            }
        ],
        "name": "doctors",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "id",
                "type": "uint256"
            },
            {
                "internalType": "string",
                "name": "name",
                "type": "string"
            },
            {
                "internalType": "string",
                "name": "qualification",
                "type": "string"
            },
            {
                "internalType": "string",
                "name": "workPlace",
                "type": "string"
            },
            {
                "internalType": "bool",
                "name": "registered",
                "type": "bool"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "medicineCount",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "name": "medicines",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "id",
                "type": "uint256"
            },
            {
                "internalType": "string",
                "name": "name",
                "type": "string"
            },
            {
                "internalType": "string",
                "name": "expiryDate",
                "type": "string"
            },
            {
                "internalType": "string",
                "name": "dose",
                "type": "string"
            },
            {
                "internalType": "uint256",
                "name": "price",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "patientCount",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "address",
                "name": "",
                "type": "address"
            }
        ],
        "name": "patients",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "id",
                "type": "uint256"
            },
            {
                "internalType": "string",
                "name": "name",
                "type": "string"
            },
            {
                "internalType": "uint256",
                "name": "age",
                "type": "uint256"
            },
            {
                "internalType": "bool",
                "name": "registered",
                "type": "bool"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "address",
                "name": "",
                "type": "address"
            },
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "name": "prescribedMedicines",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "address",
                "name": "_doctor",
                "type": "address"
            }
        ],
        "name": "viewDoctorDetails",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            },
            {
                "internalType": "string",
                "name": "",
                "type": "string"
            },
            {
                "internalType": "string",
                "name": "",
                "type": "string"
            },
            {
                "internalType": "string",
                "name": "",
                "type": "string"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "uint256",
                "name": "_id",
                "type": "uint256"
            }
        ],
        "name": "viewMedicineDetails",
        "outputs": [
            {
                "internalType": "string",
                "name": "",
                "type": "string"
            },
            {
                "internalType": "string",
                "name": "",
                "type": "string"
            },
            {
                "internalType": "string",
                "name": "",
                "type": "string"
            },
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "address",
                "name": "_patient",
                "type": "address"
            }
        ],
        "name": "viewPatientData",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            },
            {
                "internalType": "string",
                "name": "",
                "type": "string"
            },
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            },
            {
                "internalType": "string[]",
                "name": "",
                "type": "string[]"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "address",
                "name": "_patient",
                "type": "address"
            }
        ],
        "name": "viewPrescribedMedicines",
        "outputs": [
            {
                "internalType": "uint256[]",
                "name": "",
                "type": "uint256[]"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    }
];

        async function loadWeb3() {
            if (window.ethereum) {
                window.web3 = new Web3(window.ethereum);
                await window.ethereum.request({ method: "eth_requestAccounts" });
                const accounts = await web3.eth.getAccounts();
                contract = new web3.eth.Contract(contractABI, contractAddress, { from:
accounts[0] });
            } else {
                alert("Please install MetaMask!");
            }
        }

        async function registerDoctor() {
            const name = document.getElementById("docName").value;
            const qual = document.getElementById("docQual").value;
            const work = document.getElementById("docWork").value;
            await contract.methods.registerDoctor(name, qual, work).send();
            alert("Doctor Registered Successfully");
        }

        async function registerPatient() {
            const name = document.getElementById("patName").value;
            const age = document.getElementById("patAge").value;
            await contract.methods.registerPatient(name, age).send();
            alert("Patient Registered Successfully");
        }

        async function addDisease() {
            const disease = document.getElementById("disease").value;
            await contract.methods.addDisease(disease).send();
            alert("Disease Added Successfully");
        }

        async function addMedicine() {
            const id = document.getElementById("medId").value;
            const name = document.getElementById("medName").value;
            const expiry = document.getElementById("medExpiry").value;
            const dose = document.getElementById("medDose").value;
            const price = document.getElementById("medPrice").value;
            await contract.methods.addMedicine(id, name, expiry, dose, price).send();
            alert("Medicine Added Successfully");
        }

        window.onload = loadWeb3;
    </script>
</body>
</html>
```

Output:


Doctor Registration
Name
Qualification
Workplace
Register Doctor
Patient Registration
Name
Age
Register Patient
Add Disease
Disease
Add Disease
Add Medicine
Medicine ID
Medicine Name
Expiry Date
Dose
Price
Add Medicine