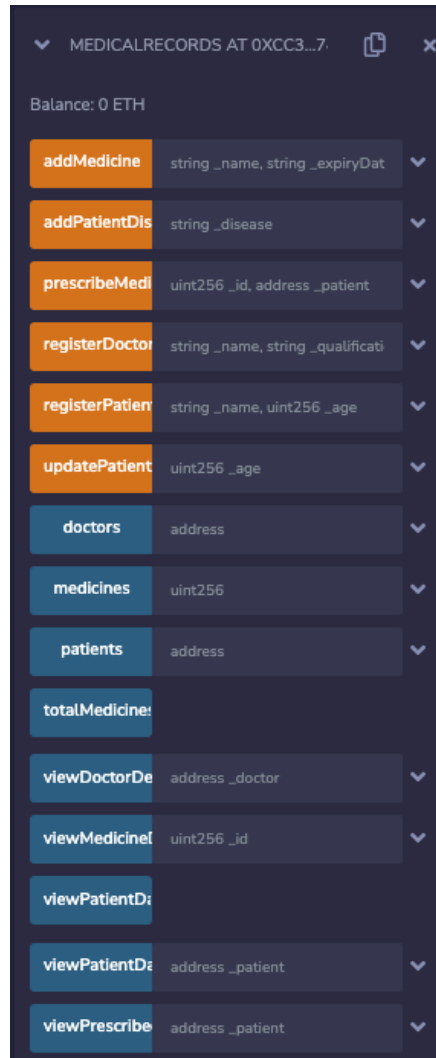
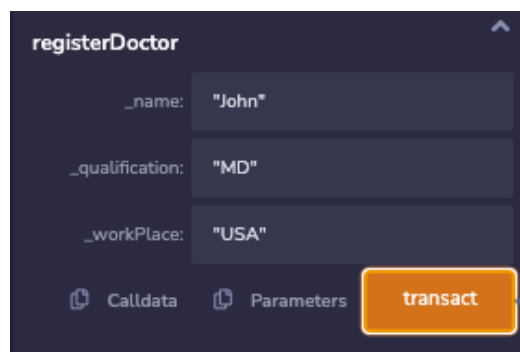


Screenshot

Functions Involved:



Register Doctor:



Register Patient:

registerPatient

_name:

Mark

_age:

34

Calldata

Parameters

transact

Patient Disease:

addPatientDisease

_disease:

Allergy

Calldata

Parameters

transact

Medicine:

addMedicine

_id:

1

_name:

Citrizine

_expiryDate:

June-2025

_dose:

1 Tablet Per Day

_price:

5

Calldata

Parameters

transact

Prescribe Medicine:

prescribeMedicine

_id:

1

_patient:

0x891fdcd69570b9c8ff7a38484

Calldata

Parameters

transact

Update Patient Details:

updatePatientDetails

_age 36

Calldata Parameters transact

Patient Details:

viewPatientData: viewPatientData - call

0: uint256: 99969593768672345668
1: string: Mark
2: uint256: 36
3: string[]: Allergy

Medicine Details:

viewMedicineDetails

_id: 1

Calldata Parameters call

0: string: Citrizine
1: string: June-2025
2: string: 1 Tablet Per Day
3: uint256: 5

Patient Data By Doctor:

viewPatientDataByDoctor

_patient: 0x891fdcd69570b9c8ff7a38484

Calldata Parameters call

CODE:

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 contract MedicalRecords {
5     struct Doctor {
6         string name;
7         string qualification;
8         string workplace;
9     }
10
11     struct Patient {
12         string name;
13         uint age;
14         string[] diseases;
15     }
16
17     struct Medicine {
18         uint id;
19         string name;
20         string expiryDate;
21         string dose;
22         uint price;
23     }
24
25     mapping(address => Doctor) public doctors;
26     mapping(address => Patient) public patients;
27     mapping(uint => Medicine) public medicines;
28     uint public totalMedicines;
29
30     constructor() {
31         @ 150000 gas 150000 gas
32         totalMedicines = 0;
33     }
34
35     function registerDoctor(string memory _name, string memory _qualification, string memory _workPlace) public {
36         doctors[msg.sender] = Doctor(_name, _qualification, _workPlace);
37     }
38
39     function registerPatient(string memory _name, uint _age) public {
40         patients[msg.sender] = Patient(_name, _age, new string[](0));
41     }
42
43     function addPatientDisease(string memory _disease) public {
44         patients[msg.sender].diseases.push(_disease);
45     }
46
47     function addMedicine(uint _id, string memory _name, string memory _expiryDate, string memory _dose, uint _price) public {
48         totalMedicines++;
49         medicines[totalMedicines] = Medicine(_id, _name, _expiryDate, _dose, _price);
50     }
51
52     function prescribeMedicine(uint _id, address _patient) public {
53         require(doctors[msg.sender].workPlace.length != 0, "Doctor not registered.");
54         require(patients[_patient].age > 0, "Patient not registered.");
55         patients[_patient].diseases.push(medicines[_id].name);
56     }
57
58     function updatePatientDetails(uint _age) public {
59         patients[msg.sender].age = _age;
60     }
61
62     function viewPatientData() public view returns (uint, string memory, uint, string[]) {
63         Patient storage patient = patients[msg.sender];
64         return (msg.sender.balance, patient.name, patient.age, patient.diseases);
65     }
66
67     function viewMedicineDetails(uint _id) public view returns (string memory, string memory, string memory, string memory, uint) {
68         uint id = _id;
69         Medicine storage medicine = medicines[id];
70         return (medicine.name, medicine.expiryDate, medicine.dose, medicine.price);
71     }
72
73     function viewPatientDataByDoctor(address _patient) public view returns (uint, string memory, uint, string[]) {
74         Patient storage patient = patients[_patient];
75         return (patient.balance, patient.name, patient.age, patient.diseases);
76     }
77
78     function viewPrescribedMedicine(address _patient) public view returns (uint[] memory) {
79         Patient storage patient = patients[_patient];
80         uint[] memory ids = new uint[](patient.diseases.length);
81         uint count = 0;
82         for (uint i = 0; i < patient.diseases.length; i++) {
83             if (keccak256(abi.encodePacked(medicines[i].name)) == keccak256(abi.encodePacked(patient.diseases[i]))) {
84                 ids[count] = i;
85                 count++;
86             }
87         }
88         return ids;
89     }
90
91     function viewDoctorDetails(address _doctor) public view returns (string memory, string memory, string memory) {
92         Doctor storage doctor = doctors[_doctor];
93         return (doctor.name, doctor.qualification, doctor.workPlace);
94     }
95 }
```

```
35 doctors[msg.sender] = Doctor(_name, _qualification, _workPlace);
36
37 function registerPatient(string memory _name, uint _age) public {
38     patients[msg.sender] = Patient(_name, _age, new string[](0));
39 }
40
41 function addPatientDisease(string memory _disease) public {
42     patients[msg.sender].diseases.push(_disease);
43 }
44
45 function addMedicine(uint _id, string memory _name, string memory _expiryDate, string memory _dose, uint _price) public {
46     totalMedicines++;
47     medicines[totalMedicines] = Medicine(_id, _name, _expiryDate, _dose, _price);
48 }
49
50 function prescribeMedicine(uint _id, address _patient) public {
51     require(doctors[msg.sender].workPlace.length != 0, "Doctor not registered.");
52     require(patients[_patient].age > 0, "Patient not registered.");
53     patients[_patient].diseases.push(medicines[_id].name);
54 }
55
56 function updatePatientDetails(uint _age) public {
57     patients[msg.sender].age = _age;
58 }
59
60 function viewPatientData() public view returns (uint, string memory, uint, string[]) {
61     Patient storage patient = patients[msg.sender];
62     return (msg.sender.balance, patient.name, patient.age, patient.diseases);
63 }
64
65 function viewMedicineDetails(uint _id) public view returns (string memory, string memory, string memory, string memory, uint) {
66     uint id = _id;
67     Medicine storage medicine = medicines[id];
68     return (medicine.name, medicine.expiryDate, medicine.dose, medicine.price);
69 }
```

```
62 Patient storage patient = patients[msg.sender];
63 return (msg.sender.balance, patient.name, patient.age, patient.diseases);
64 }
65
66 function viewMedicineDetails(uint _id) public view returns (string memory, string memory, string memory, string memory, uint) {
67     uint id = _id;
68     Medicine storage medicine = medicines[id];
69     return (medicine.name, medicine.expiryDate, medicine.dose, medicine.price);
70 }
71
72 function viewPatientDataByDoctor(address _patient) public view returns (uint, string memory, uint, string[]) {
73     Patient storage patient = patients[_patient];
74     return (patient.balance, patient.name, patient.age, patient.diseases);
75 }
76
77 function viewPrescribedMedicine(address _patient) public view returns (uint[] memory) {
78     Patient storage patient = patients[_patient];
79     uint[] memory ids = new uint[](patient.diseases.length);
80     uint count = 0;
81     for (uint i = 0; i < patient.diseases.length; i++) {
82         if (keccak256(abi.encodePacked(medicines[i].name)) == keccak256(abi.encodePacked(patient.diseases[i]))) {
83             ids[count] = i;
84             count++;
85         }
86     }
87     return ids;
88 }
89
90 function viewDoctorDetails(address _doctor) public view returns (string memory, string memory, string memory) {
91     Doctor storage doctor = doctors[_doctor];
92     return (doctor.name, doctor.qualification, doctor.workPlace);
93 }
94 }
```

Ganache Accounts

ACCOUNTS

BLOCKS

TRANSACTIONS

CONTRACTS

EVENTS

LOGS

SEARCH FOR BLOCK NUMBERS OR TX HASHES

MINERIALS
DECENTRALISED PATIENT RECORD STORAGE

SWITCH

MNEMONIC

snap miss wisdom sail fabric easily reject arrest dragon neutral fresh mail

HD PATH

m44'68'0"baccount_index

ADDRESS

0xCc21f5aE1257AE8524ff1aA7f8c18033eBA87Afa

BALANCE

99.97 ETH

TX COUNT

16

INDEX

0

ADDRESS

0x19d87EbE7A0729Ae14BbF1d08ae5aB99FE0ead0f

BALANCE

100.00 ETH

TX COUNT

0

INDEX

1

ADDRESS

0xbd23e11E4796604D0B5daAEd37A861EeCBeCaF89

BALANCE

100.00 ETH

TX COUNT

0

INDEX

2

ADDRESS

0x5Ffb04D248d6b2c671266c3750d9090751437165

BALANCE

100.00 ETH

TX COUNT

0

INDEX

3

ADDRESS

0xc59EdF872A0e30bA1Ee20a279A3F309bf56d055

BALANCE

100.00 ETH

TX COUNT

0

INDEX

4

ADDRESS

0x2Ef5162eb709ddC742dD42820a18E44794E175b2

BALANCE

100.00 ETH

TX COUNT

0

INDEX

5

ADDRESS

0x67E57a7C6354b4A67F82b57df09478060F16B190

BALANCE

100.00 ETH

TX COUNT

0

INDEX

6

ADDRESS

0x29Ec0d712ACe471fCB95256b263b9cF8081DFD2

BALANCE

100.00 ETH

TX COUNT

0

INDEX

7

ADDRESS

0x30c44aDA4Fd2549899E42Cc0677Fe4B45bd653DD

BALANCE

100.00 ETH

TX COUNT

0

INDEX

8

Blocks:

ACCOUNTS		BLOCKS	TRANSACTIONS	CONTRACTS	EVENTS	LOGS	SEARCH FOR BLOCK NUMBERS OR TX HASHES	
CURRENT BLOCK 16	GAS PRICE 2000000000	GAS LIMIT 6721975	BASEFEE MERGE	NETWORK ID 5777	RPC ENDPOINT HTTP://127.0.0.1:7545	MINING STATUS AUTOMINING	MINERIALS DECENTRALISED PATIENT RECORD STORAGE	
							SWITCH	+
BLOCK 16	MINED ON 2023-05-19 22:29:43					GAS USED 67621	1 TRANSACTION	
BLOCK 15	MINED ON 2023-05-19 22:29:32					GAS USED 20764	1 TRANSACTION	
BLOCK 14	MINED ON 2023-05-19 22:28:32					GAS USED 35128	1 TRANSACTION	
BLOCK 13	MINED ON 2023-05-19 22:24:35					GAS USED 160710	1 TRANSACTION	
BLOCK 12	MINED ON 2023-05-19 22:23:45					GAS USED 50521	1 TRANSACTION	
BLOCK 11	MINED ON 2023-05-19 22:23:28					GAS USED 67621	1 TRANSACTION	
BLOCK 10	MINED ON 2023-05-19 22:23:07					GAS USED 70393	1 TRANSACTION	
BLOCK 9	MINED ON 2023-05-19 22:22:30					GAS USED 93190	1 TRANSACTION	
BLOCK 8	MINED ON 2023-05-19 22:21:54					GAS USED 32893	1 TRANSACTION	
BLOCK 7	MINED ON 2023-05-19 22:20:29					GAS USED 2150973	1 TRANSACTION	
BLOCK 6	MINED ON 2023-05-19 22:18:36					GAS USED 67587	1 TRANSACTION	
BLOCK 5	MINED ON 2023-05-19 22:17:20					GAS USED 70393	1 TRANSACTION	

Transactions:

ACCOUNTS		BLOCKS	TRANSACTIONS	CONTRACTS	EVENTS	LOGS	SEARCH FOR BLOCK NUMBERS OR TX HASHES		
CURRENT BLOCK 16	GAS PRICE 2000000000	GAS LIMIT 6721975	BASEFEE MERGE	NETWORK ID 5777	RPC ENDPOINT HTTP://127.0.0.1:7545	MINING STATUS AUTOMINING	MINERIALS DECENTRALISED PATIENT RECORD STORAGE	<div>SWITCH</div>	<div>+</div>
FROM ADDRESS 0xCc21f5aE1257AE8524ff1aA7f8c18033eBA87Afa							TO CONTRACT ADDRESS 0xCc30015646FCb318703EA78ED052C2338ED0874c73	GAS USED 67587	VALUE 0
TX HASH 0xf01caa198890ea79168c063b67b65d6264a204536f6be25dd70a55dfbaaad10d							<div>CONTRACT CALL</div>		
FROM ADDRESS 0xCc21f5aE1257AE8524ff1aA7f8c18033eBA87Afa							TO CONTRACT ADDRESS 0xCc30015646FCb318703EA78ED052C2338ED0874c73	GAS USED 70393	VALUE 0
TX HASH 0x6b95081385da3c5765ba9eb999a9ab4c5ee22d50b76869dfe0ca82ed90d8265							<div>CONTRACT CALL</div>		
FROM ADDRESS 0xCc21f5aE1257AE8524ff1aA7f8c18033eBA87Afa							TO CONTRACT ADDRESS 0xCc30015646FCb318703EA78ED052C2338ED0874c73	GAS USED 93214	VALUE 0
TX HASH 0x6cb4efe06b06c12cb014e46c875a76b6c66ddf36f2ce1467f4f919320036ccaa							<div>CONTRACT CREATION</div>		
FROM ADDRESS 0xCc21f5aE1257AE8524ff1aA7f8c18033eBA87Afa							CREATED CONTRACT ADDRESS 0xCc30015646FCb318703EA78ED052C2338ED0874c73	GAS USED 2146845	VALUE 0
TX HASH 0x12c24d3a460a2ce0880b8e9bc19547a83ff46016cf6a24ac4f056442c3da4163							<div>CONTRACT CREATION</div>		
FROM ADDRESS 0xCc21f5aE1257AE8524ff1aA7f8c18033eBA87Afa							CREATED CONTRACT ADDRESS 0x3f2b320b31005100524E405322636567d3058311	GAS USED 2146845	VALUE 0
TX HASH 0x70ae56cfcd57eb6593632136c0195481e81a9c4f4a0c18d0fb0f7a39213b5ccf							<div>CONTRACT CREATION</div>		
FROM ADDRESS 0xCc21f5aE1257AE8524ff1aA7f8c18033eBA87Afa							CREATED CONTRACT ADDRESS 0x0f6370f90f20f5a093d0aaf11d61220a7e48ff7c	GAS USED 2146845	VALUE 0

Files Involved:



RPC Server:

