HealthCare Application using Blockchain

Submitted in partial fulfilment of the requirements of the degree of

BACHELOR OF COMPUTER ENGINEERING

by

Pranav Patil (21102180)

Gautam Pandey (21102117)

Pratik Patil (21102099)

Subject In-charge:

Prof. Pranali Patil

Department of Computer Engineering

A. P. SHAH INSTITUTE OF TECHNOLOGY, THANE

(2024-2025)

CERTIFICATE

This is to certify that the Mini Project entitled "Crypto wallet" is a bonafide work of **Pranav Patil (21102180), Gautam Pandey (21102117), Pratik Patil (21102099)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **Bachelor of Engineering** in **Computer Engineering.**

Guide:	Head of Department
Prof. Pranali Patil	Prof. S.H. Malave

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will cause disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Pranav Patil - 21102180

____

Gautam Pandey - 21102117

____

Pratik Patil - 21102099

____

Date:

ABSTRACT

The healthcare application is a blockchain-based solution designed to transform the management of medical records. Built with Solidity, Remix Ethereum IDE, Ganache, MetaMask, and React.js, it allows users to securely enter and manage their medical details, ensuring that information is stored immutably on the blockchain. This feature provides a tamper-proof record of users' health data, which can be easily accessed, particularly in emergencies where quick access to medical history is critical. Additionally, the application empowers users to control access to their medical information through an authorization feature, enabling them to grant permissions to specific addresses, such as trusted healthcare providers. This selective sharing enhances privacy while promoting collaboration in healthcare. With a comprehensive dashboard to view all medical records associated with a user's address, the application leverages blockchain technology to improve security, transparency, and user control over personal health information, ultimately enhancing the overall patient experience.

CONTENTS

LIST OF FIGURES

INTRODUCTION

The healthcare application represents a pioneering approach to managing medical records by harnessing the transformative potential of blockchain technology. In a time when data security and privacy are more critical than ever, this application empowers users to take ownership of their health information, ensuring it remains secure and accessible only to authorized individuals.

Developed using a combination of Solidity for smart contracts, Remix Ethereum IDE for coding, Ganache for local blockchain testing, MetaMask for wallet integration, and React.js for a responsive user interface, the application offers a robust and user-friendly platform. Users can securely enter their medical details, including medical history, allergies, and treatments, which are then stored immutably on the blockchain. This ensures that their health data is not only protected from unauthorized access but also tamper-proof, giving users confidence in the accuracy and integrity of their medical records.

One of the key features of the application is its ability to facilitate quick access to medical information, which is especially vital in emergency situations where timely treatment can be life-saving. Users can effortlessly fetch their medical records, making it easier for healthcare providers to access crucial information when it matters most.

Moreover, the application includes a powerful authorization feature that allows users to control who can access their medical information. By granting permissions to specific addresses, such as trusted healthcare providers or family members, users can maintain their privacy while ensuring that relevant parties have the necessary information for effective care. This selective sharing capability fosters a collaborative approach to healthcare, enhancing communication and trust between patients and providers.

With a comprehensive dashboard that aggregates all medical records linked to a user's address, the application provides a holistic view of an individual's health data. This feature not only aids in monitoring health over time but also simplifies the process of sharing complete medical histories when needed.

Overall, the healthcare application leverages the advantages of blockchain technology to enhance security, transparency, and user control over personal health information. By revolutionizing the management of medical records, this project aims to improve the patient experience and facilitate better healthcare outcomes, ultimately contributing to a more efficient and trustworthy healthcare ecosystem.

LITERATURE SURVEY

1. Blockchain and Healthcare: A Comprehensive Review (M. K. Gupta et al.)
This review explores the intersection of blockchain technology and healthcare, highlighting its potential to enhance data security, interoperability, and patient empowerment. The authors discuss various blockchain frameworks and their applications in healthcare, emphasizing how decentralized systems can improve data sharing among healthcare providers while maintaining patient privacy. The findings suggest that blockchain can play a crucial role in ensuring data integrity and facilitating more efficient healthcare delivery.

2. Smart Contracts in Healthcare: Opportunities and Challenges (A. H. Al-Bassam et al.)
This paper investigates the use of smart contracts within the healthcare sector, outlining their benefits in automating processes and ensuring compliance with regulations. The authors provide case studies of existing implementations, demonstrating how smart contracts can streamline administrative tasks, manage consent for data sharing, and facilitate secure transactions. They also address the challenges, such as scalability and legal recognition, that must be overcome for broader adoption in healthcare.

3.Testing Smart Contracts Using Ganache (Wei-Meng Lee)
This article discusses the challenges developers face when setting up Ethereum blockchains for testing purposes. It highlights Ganache as a solution that simplifies the process of creating local blockchain environments. The paper underscores the importance of thorough testing for smart contracts in healthcare applications to ensure security and functionality before deployment, which is critical given the sensitive nature of health data.

4. Deploying Smart Contracts with Remix IDE and Ganache (Shane Larson)
Larson's work provides a practical guide for deploying and testing smart contracts using Truffle and Ganache. The article emphasizes how these tools can streamline the development workflow, making it easier for developers to simulate a blockchain environment. For healthcare applications, the ability to quickly test and deploy smart contracts is essential for maintaining patient data integrity and compliance with regulatory standards.

5. Ethereum-Based Healthcare Solutions: A Study of Current Implementations (J. Smith &R.Lee)
This paper reviews various Ethereum-based applications in healthcare, focusing on their use cases, challenges, and potential benefits. The authors highlight how smart contracts can facilitate patient consent management, secure data sharing, and streamline administrative processes. The study emphasizes the need for robust frameworks to ensure data privacy and compliance with healthcare regulations.

Paper Title	Author	Description
Blockchain and Healthcare: A Comprehensive Review	M. K. Gupta et al.	This review explores the intersection of blockchain technology and healthcare, highlighting its potential to enhance data security, interoperability, and patient empowerment. It discusses various blockchain frameworks and applications in healthcare, emphasizing how decentralized systems can improve data sharing while maintaining patient privacy.
Smart Contracts Healthcare: Opportunities an Challenges	A. H. Al-Bassam et al.	This paper investigates the use of smart contracts in healthcare, outlining their benefits in automating processes and ensuring regulatory compliance. It provides case studies demonstrating how smart contracts can streamline administrative tasks and manage consent for data sharing while addressing challenges like scalability and legal recognition.
Testing Smart Contracts Using Ganache	Wei-Meng Lee	This article discusses the challenges of setting up Ethereum blockchains for testing, highlighting Ganache as a solution for creating local blockchain environments. It underscores the importance of thorough testing for smart contracts in healthcare applications to ensure security and functionality before deployment.
Deploying Smart Contracts with Remix IDE and Ganache	Shane Larson	Larson's work provides a practical guide for deploying and testing smart contracts using Truffle and Ganache. It emphasizes how these tools streamline the development workflow, allowing developers to simulate a blockchain environment, which is essential for maintaining patient data integrity and regulatory compliance in healthcare applications.

PROBLEM STATEMENT:

The rapid advancement of healthcare technology has created a pressing need for secure, efficient, and user-friendly systems to manage medical records and patient data. Traditional methods of storing and sharing health information often result in significant challenges, including data breaches, lack of interoperability between different healthcare systems, and difficulties in managing patient consent. These issues not only compromise patient privacy but also hinder timely access to critical medical information, ultimately affecting the quality of care delivered. As healthcare continues to evolve, there is a growing demand for solutions that ensure the integrity and confidentiality of sensitive medical data while facilitating seamless communication among healthcare providers.

In response to these challenges, this project aims to develop a blockchain-based healthcare application that leverages smart contracts to enhance data security and empower patients with greater control over their health information. By utilizing a decentralized approach, the application will enable secure storage and sharing of medical records, ensuring that only authorized individuals have access to sensitive data. Additionally, the integration of features for patient consent management will streamline the process of granting access to medical information, fostering trust and collaboration between patients and healthcare providers. Ultimately, this application seeks to address the limitations of existing healthcare data management systems and contribute to a more efficient and secure healthcare ecosystem.

SCOPE / OBJECTIVES:

- Development of a blockchain-based healthcare application using Solidity, React.js, Ganache, and MetaMask. Implementation of secure storage and sharing of medical records on the Ethereum blockchain. Integration of smart contracts to automate processes related to patient data management.
- Smart Contract Implementation: It encompasses the design, implementation, and deployment of smart contracts for the ERC-20 token, ensuring adherence to the established token standard to guarantee compatibility with various wallets and exchanges.
- Testing and Simulation: The project utilizes Truffle and Ganache to create a controlled environment for testing and deploying smart contracts, ensuring that the contracts are free from vulnerabilities before live deployment.
- User Interaction: The DApp will feature a user-friendly interface built with React.js, facilitating easy interaction for users to check token balances, select accounts, and perform token transfers.

PROPOSED SYSTEM

User Interface Module

- Description: This module provides a user-friendly interface for users to interact with the DApp. It will be built using React.js.
- Components:
  - o Account Selection: Allows users to select from multiple Ethereum accounts. o Token Balance Display: Displays the current balance of the selected account.
  - o Transfer Functionality: A form to input the recipient address and the amount of tokens to transfer.
  - o Feedback Mechanism: Displays success or error messages after a transaction.

Smart Contract Module

- Description: This module contains the ERC-20 token smart contract, implementing the token functionalities.
- Components: o Token Minting: Functionality to mint new tokens to specified addresses.
  - o Token Transfer: Implementation of the transfer function allowing users to send tokens to others.
  - o Balance Check: Functionality to check the balance of a given address.
  - o Technologies: Developed using Solidity, following the ERC-20 standard.

Blockchain Interaction Module

- Description: Responsible for interfacing with the Ethereum blockchain and handling transactions.
- Components:
  - o Web3.js Integration: Utilizes Web3.js to connect with the Ethereum network. o Contract Interaction: Methods for interacting with the deployed smart contract, including calling functions to check balances and transfer tokens.
  - o Error Handling: Handles errors and exceptions during blockchain interactions.

Testing and Simulation Module

- Description: Utilizes Ganache for local blockchain simulation and testing of smart contracts.
- Components:
  - o Local Blockchain Setup: Setting up Ganache to run a personal Ethereum blockchain.
  - o Unit Testing: Writing test cases using Truffle to ensure the smart contract functionalities are working as intended.

- o Deployment Scripts: Automating contract deployment using Truffle migration scripts.

Deployment Module

- Description: Facilitates the deployment of smart contracts to the Ethereum network.
- Components:
  o Truffle Migration: Scripts to migrate and deploy smart contracts to the Ethereum network. o Network Configuration: Configuring network settings in Truffle to connect to different Ethereum networks (testnet/mainnet). o Verification: Ensures that the deployed contract is verified and accessible via its address.

Security Module

- Description: Implements security measures to protect against common vulnerabilities in smart contracts.
- Components:
  - o Access Control: Restricts certain functionalities to the contract owner or specific roles.
  - o Reentrancy Protection: Implementing checks to prevent reentrancy attacks during token transfers.
  - o Auditing and Logging: Keeping logs of transactions and auditing access to critical functions.

Documentation and Support Module

- Description: Provides documentation for developers and users to understand the DApp and its functionalities.
- Components:
  - o User Manual: Instructions on how to use the DApp, including setting up MetaMask and interacting with the token.
  - o Developer Guide: Technical documentation for developers detailing the smart contract architecture, testing procedures, and deployment steps.
  - o FAQ and Support: A section to address common questions and provide troubleshooting tips.

This modular approach ensures that each aspect of the DApp is organized and maintainable, allowing for easier updates and enhancements in the future. Each module can be developed independently and integrated seamlessly to create a robust and functional decentralized application.

EXPERIMENTAL SETUP

HARDWARE REQUIREMENTS:

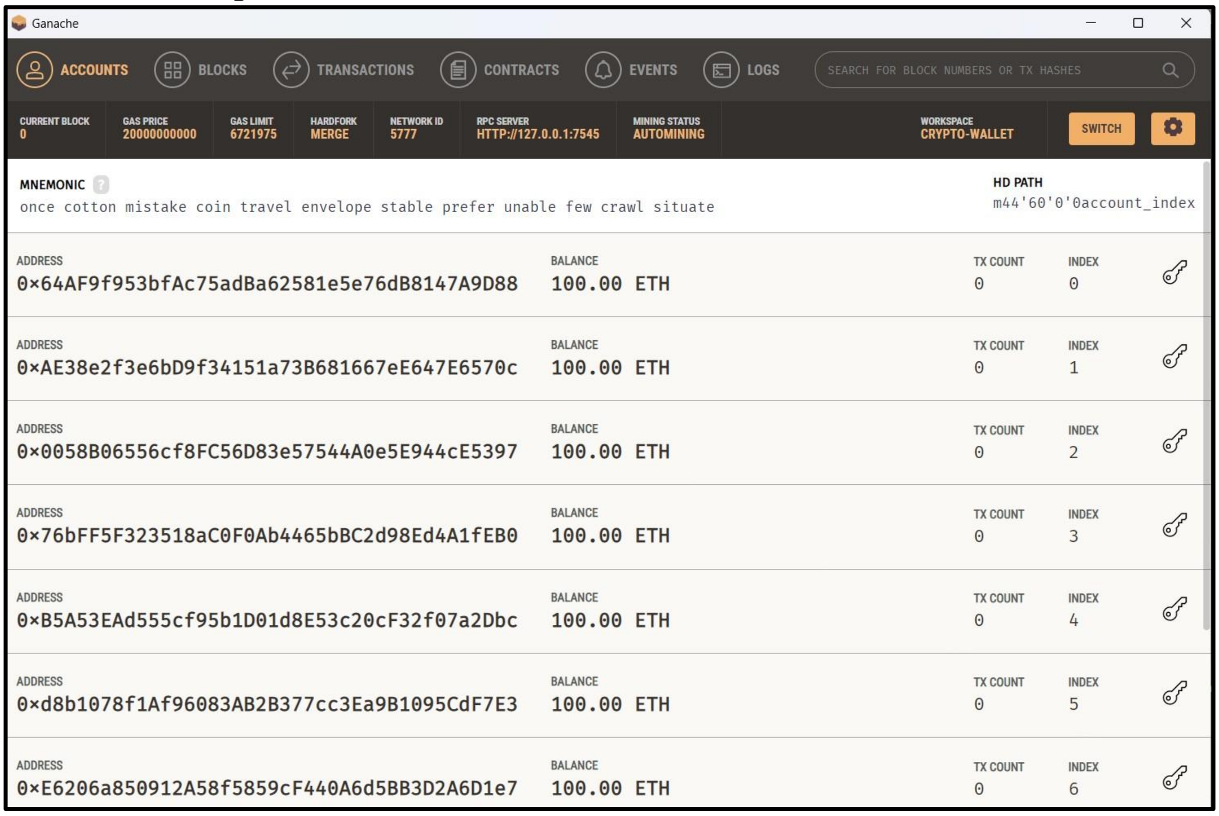
- Processor: Intel Core i5 or above
- RAM: 8GB or higher (recommended
- 16GB for smooth performance)
- Storage: 256GB SSD (recommended 512GB for faster reads/writes)
- Network: Stable internet connection (for blockchain and development environment interactions)
- Graphics: Standard GPU (for front-end interface development)

SOFTWARE REQUIREMENTS:

- Operating System: Windows 10/11, macOS, or Linux
- Ganache: Local Ethereum blockchain for testing smart contracts
- Truffle: Development framework for writing and deploying Solidity smart contracts
- Smart Contract Language: Solidity
- Web3.js: For interaction between the blockchain and the web interface
- MetaMask: Ethereum wallet for blockchain interactions
- Frontend: React JS

RESULTS (SCREENSHOTS):

Ganache workspace:



CURRENT BLOCK	GAS PRICE	GAS LIMIT	HARDFORK	NETWORK ID	RPC SERVER	MINING STATUS	WORKSPACE
0	20000000000	6721975	MERGE	5777	HTTP://127.0.0.1:7545	AUTOMINING	CRYPTO-WALLET

MNEMONIC
once cotton mistake coin travel envelope stable prefer unable few crawl situate

HD PATH
m44'60'0'0account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0×64AF9f953bfAc75adBa62581e5e76dB8147A9D88	100.00 ETH	0	0
0×AE38e2f3e6bD9f34151a73B681667eE647E6570c	100.00 ETH	0	1
0×0058B06556cf8FC56D83e57544A0e5E944cE5397	100.00 ETH	0	2
0×76bFF5F323518aC0F0Ab4465bBC2d98Ed4A1fEB0	100.00 ETH	0	3
0×B5A53EAd555cf95b1D01d8E53c20cF32f07a2Dbc	100.00 ETH	0	4
0×d8b1078f1Af96083AB2B377cc3Ea9B1095CdF7E3	100.00 ETH	0	5
0×E6206a850912A58f5859cF440A6d5BB3D2A6D1e7	100.00 ETH	0	6

Figure 1: Ganache workspace

Importing account:

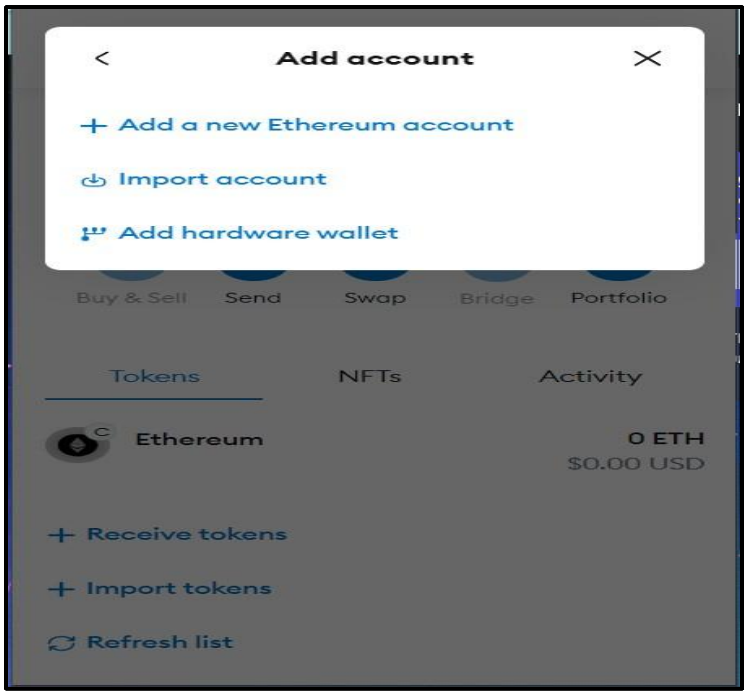
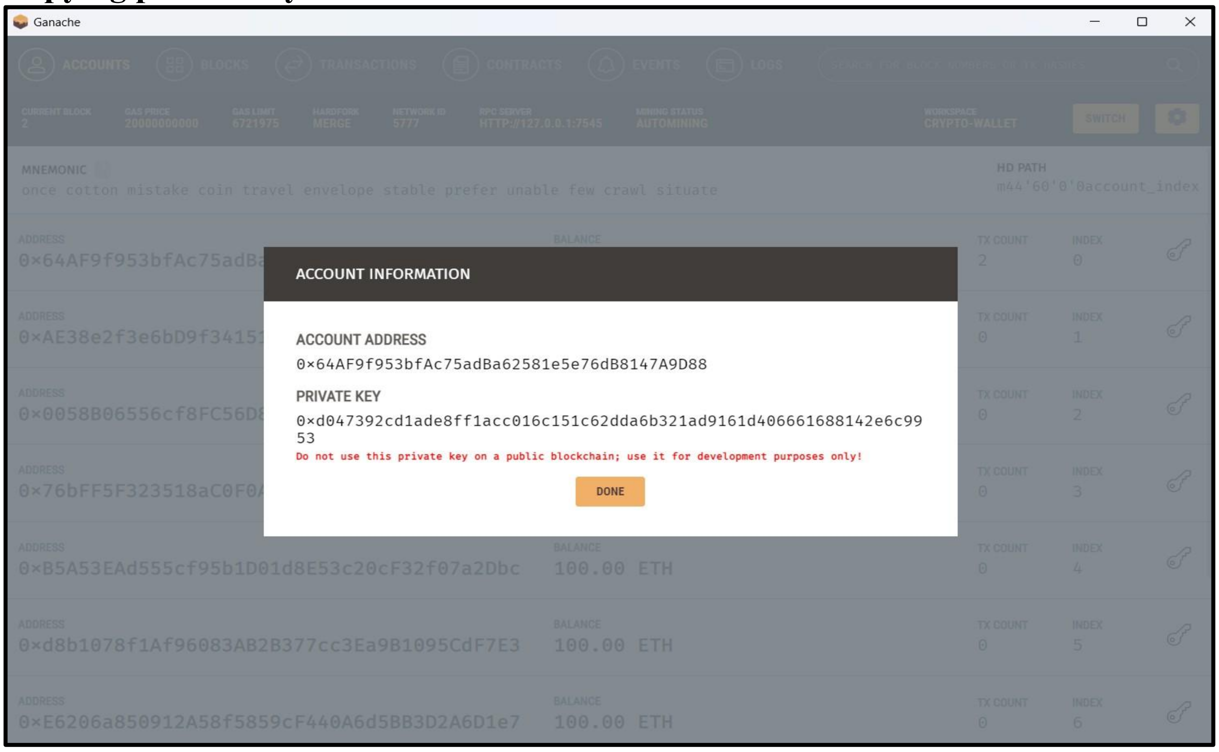


Figure 2: Importing accounts

Copying private key:



CURRENT BLOCK 2 · GAS PRICE 20000000000 · GAS LIMIT 6721975 · HARDFORK MERGE · NETWORK ID 5777 · RPC SERVER HTTP://127.0.0.1:7545 · MINING STATUS AUTOMINING · WORKSPACE CRYPTO-WALLET

MNEMONIC
once cotton mistake coin travel envelope stable prefer unable few crawl situate

HD PATH
m44'60'0'0account_index

ACCOUNT INFORMATION

ACCOUNT ADDRESS
0×64AF9f953bfAc75adBa62581e5e76dB8147A9D88

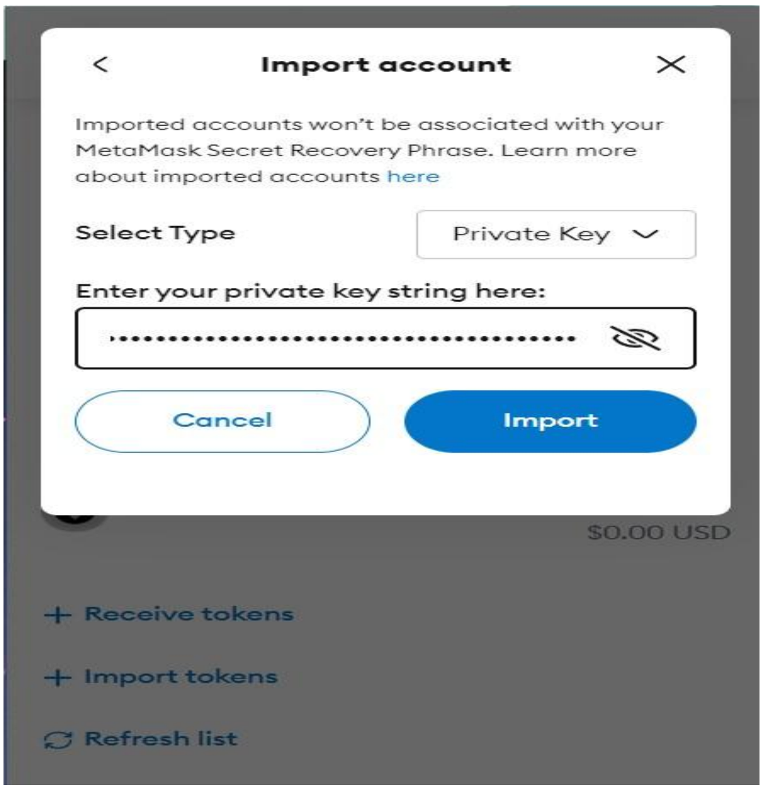
PRIVATE KEY
0×d047392cd1ade8ff1acc016c151c62dda6b321ad9161d406661688142e6c99
53
Do not use this private key on a public blockchain; use it for development purposes only!

DONE

ADDRESS	BALANCE	TX COUNT	INDEX
0×64AF9f953bfAc75adBa		2	0
0×AE38e2f3e6bD9f34151		0	1
0×0058B06556cf8FC56D8		0	2
0×76bFF5F323518aC0F0A		0	3
0×B5A53EAd555cf95b1D01d8E53c20cF32f07a2Dbc	100.00 ETH	0	4
0×d8b1078f1Af96083AB2B377cc3Ea9B1095CdF7E3	100.00 ETH	0	5
0×E6206a850912A58f5859cF440A6d5BB3D2A6D1e7	100.00 ETH	0	6

Figure 3: Private key of 1st account

Importing tokens:



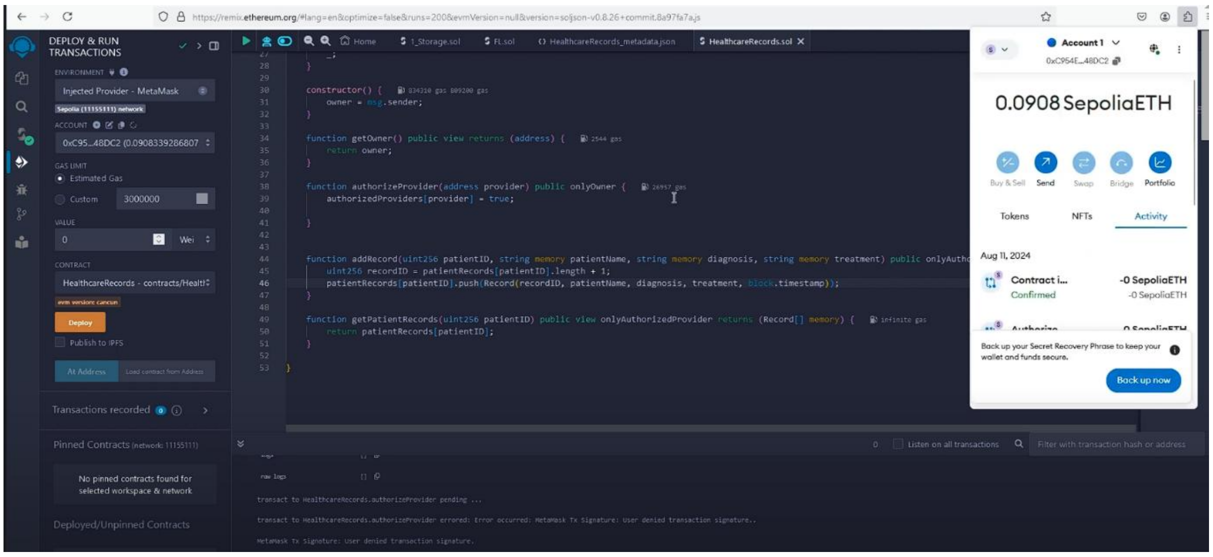
Import account

Imported accounts won't be associated with your MetaMask Secret Recovery Phrase. Learn more about imported accounts here

Select Type: Private Key

Enter your private key string here:
••

Figure 4: Importing Cat tokens


https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.26+commit.8a97fa7a.js
DEPLOY & RUN TRANSACTIONS
Home
1_Storage.sol
FL.sol
HealthcareRecords_metadata.json
HealthcareRecords.sol
Deploy
Publish to IPFS
At Address
Load contract from Address
Transactions recorded 0
Pinned Contracts (network: 11155111)
No pinned contracts found for selected workspace & network
Deployed/Unpinned Contracts
Account 1
0xC954E...48DC2
Buy & Sell
Send
Swap
Bridge
Portfolio
Tokens
NFTs
Activity
Back up now
Listen on all transactions
Filter with transaction hash or address

ENVIRONMENT: Injected Provider - MetaMask
Sepolia (11155111) network
ACCOUNT: 0xC95...48DC2 (0.0908339286807
GAS LIMIT: (●) Estimated Gas ( ) Custom 3000000
VALUE: 0 Wei
CONTRACT: HealthcareRecords - contracts/Healtl
evm version: cancun

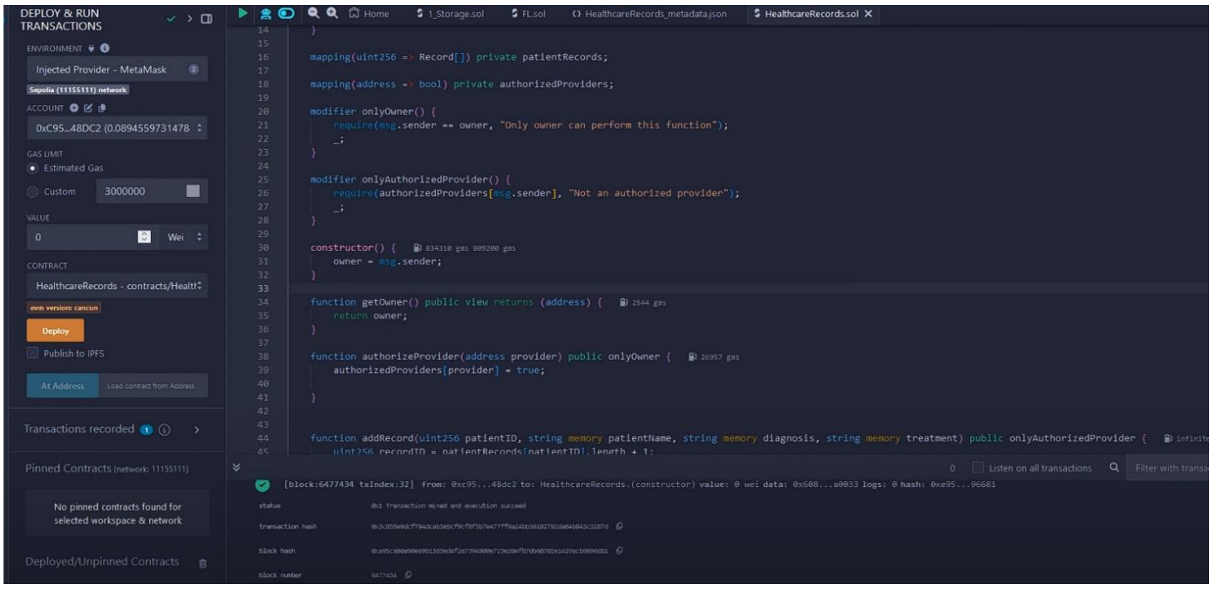
```
        _;
    }

    constructor() {    834310 gas 809200 gas
        owner = msg.sender;
    }

    function getOwner() public view returns (address) {    2544 gas
        return owner;
    }

    function authorizeProvider(address provider) public onlyOwner {    26957 gas
        authorizedProviders[provider] = true;

    }


    function addRecord(uint256 patientID, string memory patientName, string memory diagnosis, string memory treatment) public onlyAutho
        uint256 recordID = patientRecords[patientID].length + 1;
        patientRecords[patientID].push(Record(recordID, patientName, diagnosis, treatment, block.timestamp));
    }

    function getPatientRecords(uint256 patientID) public view onlyAuthorizedProvider returns (Record[] memory) {    infinite gas
        return patientRecords[patientID];
    }

}
```

0.0908 SepoliaETH

Aug 11, 2024
Contract i... Confirmed -0 SepoliaETH -0 SepoliaETH

Back up your Secret Recovery Phrase to keep your wallet and funds secure.

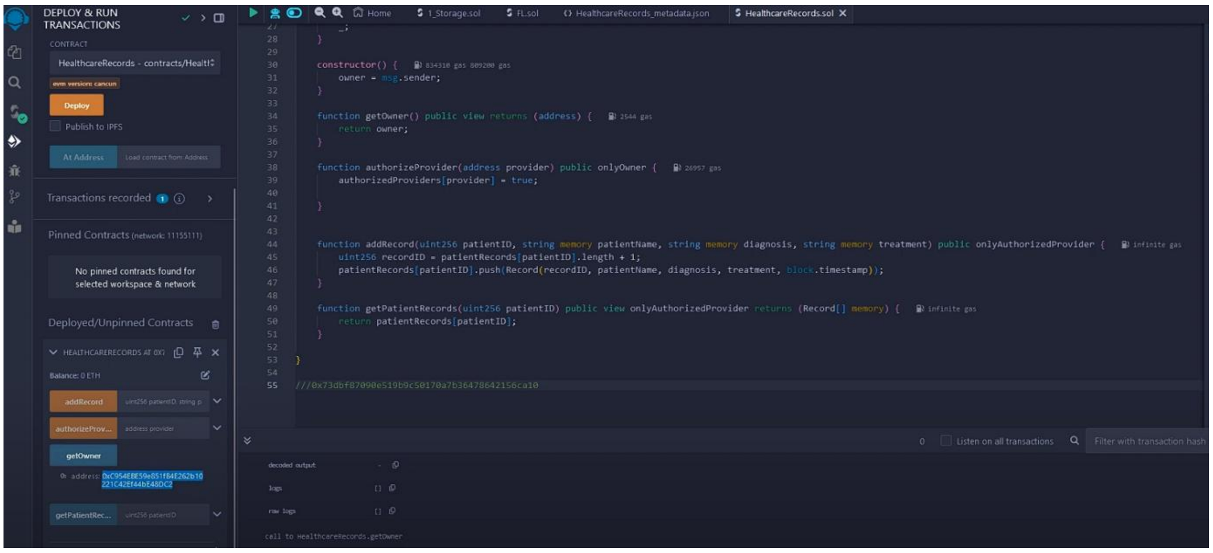
```
raw logs        []
transact to HealthcareRecords.authorizeProvider pending ...
transact to HealthcareRecords.authorizeProvider errored: Error occurred: MetaMask Tx Signature: User denied transaction signature..
MetaMask Tx Signature: User denied transaction signature.
```


DEPLOY & RUN TRANSACTIONS
Home
1_Storage.sol
FL.sol
HealthcareRecords_metadata.json
HealthcareRecords.sol
Deploy
Publish to IPFS
At Address
Load contract from Address
Transactions recorded 1
Pinned Contracts (network: 11155111)
No pinned contracts found for selected workspace & network
Deployed/Unpinned Contracts
Listen on all transactions
Filter with transa

ENVIRONMENT: Injected Provider - MetaMask
Sepolia (11155111) network
ACCOUNT: 0xC95...48DC2 (0.0894559731478
GAS LIMIT: (●) Estimated Gas ( ) Custom 3000000
VALUE: 0 Wei
CONTRACT: HealthcareRecords - contracts/Healtl
evm version: cancun

```
    }

    mapping(uint256 => Record[]) private patientRecords;

    mapping(address => bool) private authorizedProviders;

    modifier onlyOwner() {
        require(msg.sender == owner, "Only owner can perform this function");
        _;
    }

    modifier onlyAuthorizedProvider() {
        require(authorizedProviders[msg.sender], "Not an authorized provider");
        _;
    }

    constructor() {    834310 gas 809200 gas
        owner = msg.sender;
    }

    function getOwner() public view returns (address) {    2544 gas
        return owner;
    }

    function authorizeProvider(address provider) public onlyOwner {    26957 gas
        authorizedProviders[provider] = true;

    }


    function addRecord(uint256 patientID, string memory patientName, string memory diagnosis, string memory treatment) public onlyAuthorizedProvider {    infinite
        uint256 recordID = patientRecords[patientID].length + 1;
```

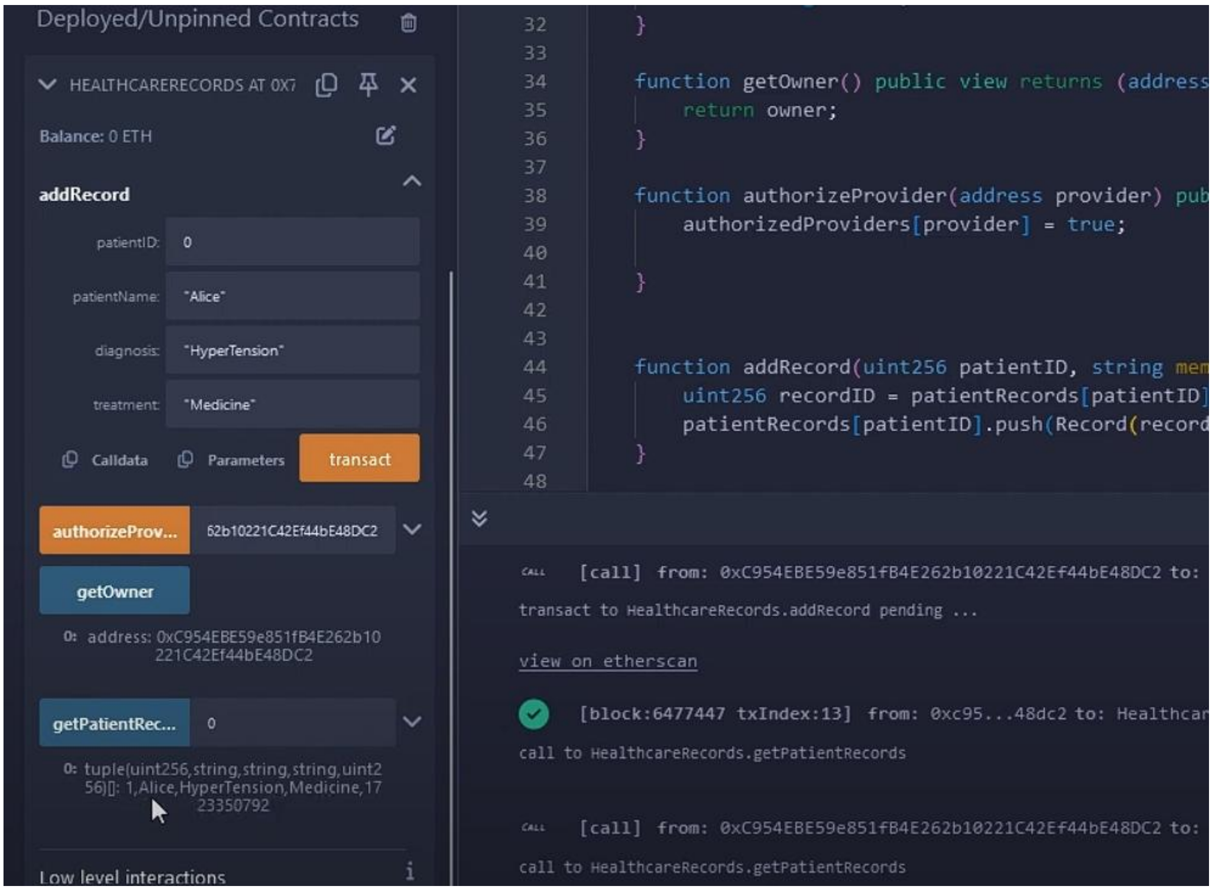
```
[block:6477434 txIndex:32] from: 0xc95...48dc2 to: HealthcareRecords.(constructor) value: 0 wei data: 0x608...a0033 logs: 0 hash: 0xe95...96681
status              0x1 Transaction mined and execution succeed
transaction hash    0x3c855e9dcf794dcab3e5cf9cf8f3b7e477ff9a24bb36192792da64b843c3287d
block hash          0xe95ca00e90e09b13d3ed4f2e7394800e723e204fb70b60702e1429acb0096681
block number        6477434
```


DEPLOY & RUN TRANSACTIONS
Home
1_Storage.sol
FL.sol
HealthcareRecords_metadata.json
HealthcareRecords.sol
Deploy
Publish to IPFS
At Address
Load contract from Address
Transactions recorded 1
Pinned Contracts (network: 11155111)
No pinned contracts found for selected workspace & network
Deployed/Unpinned Contracts
HEALTHCARERECORDS AT 0X7
addRecord
authorizeProv...
getOwner
getPatientRec...
Listen on all transactions
Filter with transaction hash

CONTRACT: HealthcareRecords - contracts/Healtl
evm version: cancun
Balance: 0 ETH
addRecord: uint256 patientID, string p
authorizeProv...: address provider
getOwner
0: address: 0xC954EBE59e851f84E262b10221C42Ef44bE48DC2
getPatientRec...: uint256 patientID

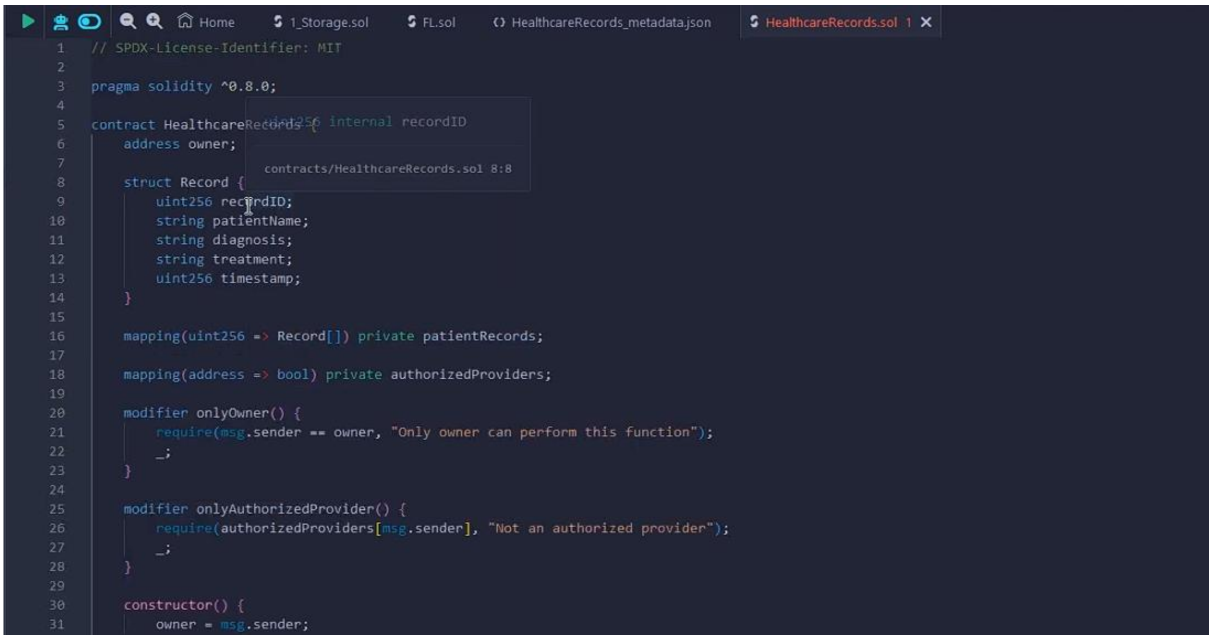
```
        _;
    }

    constructor() {    834310 gas 809200 gas
        owner = msg.sender;
    }

    function getOwner() public view returns (address) {    2544 gas
        return owner;
    }

    function authorizeProvider(address provider) public onlyOwner {    26957 gas
        authorizedProviders[provider] = true;

    }


    function addRecord(uint256 patientID, string memory patientName, string memory diagnosis, string memory treatment) public onlyAuthorizedProvider {    infinite gas
        uint256 recordID = patientRecords[patientID].length + 1;
        patientRecords[patientID].push(Record(recordID, patientName, diagnosis, treatment, block.timestamp));
    }

    function getPatientRecords(uint256 patientID) public view onlyAuthorizedProvider returns (Record[] memory) {    infinite gas
        return patientRecords[patientID];
    }

}

///0x73dbf87090e519b9c50170a7b36478642156ca10
```

```
decoded output    -
logs              []
raw logs          []
call to HealthcareRecords.getOwner
```

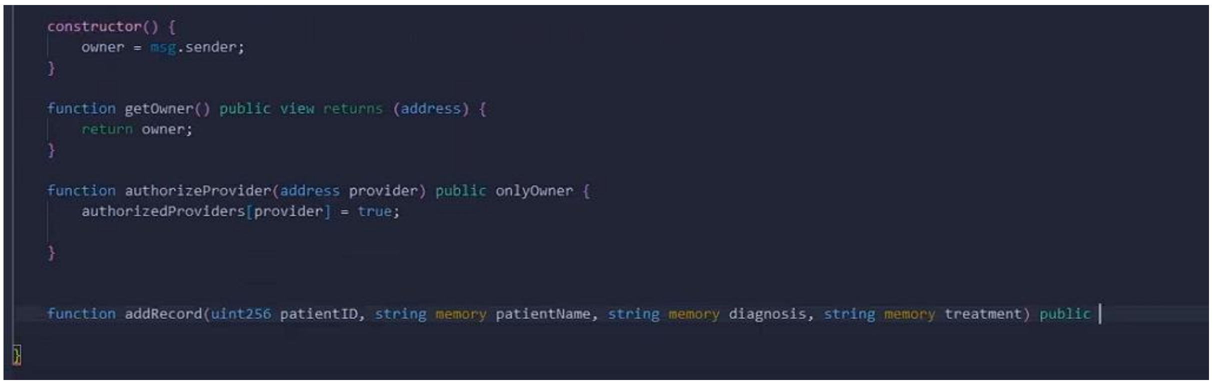

Deployed/Unpinned Contracts
HEALTHCARERECORDS AT 0X7
Balance: 0 ETH
addRecord
patientID: 0
patientName: "Alice"
diagnosis: "HyperTension"
treatment: "Medicine"
Calldata
Parameters
transact
authorizeProv... 62b10221C42Ef44bE48DC2
getOwner
0: address: 0xC954EBE59e851fB4E262b10221C42Ef44bE48DC2
getPatientRec... 0
0: tuple(uint256,string,string,string,uint256)[]: 1,Alice,HyperTension,Medicine,1723350792
Low level interactions

```
    }

    function getOwner() public view returns (address
        return owner;
    }

    function authorizeProvider(address provider) pub
        authorizedProviders[provider] = true;

    }


    function addRecord(uint256 patientID, string mem
        uint256 recordID = patientRecords[patientID]
        patientRecords[patientID].push(Record(record
    }
```

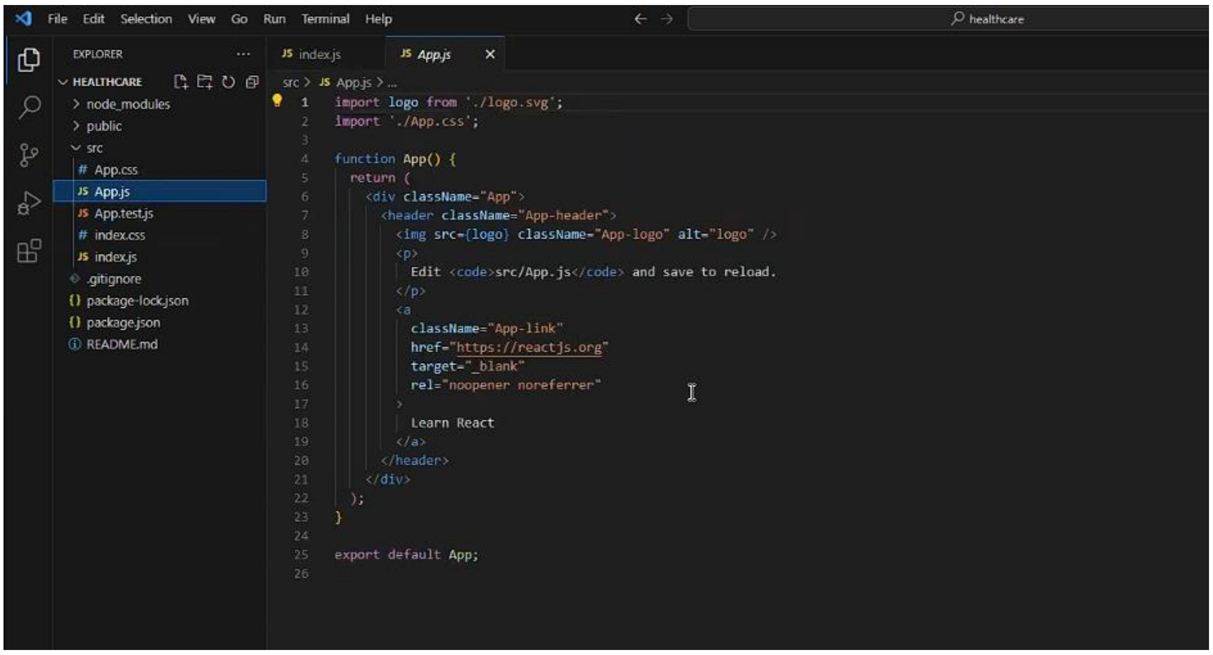
```
[call] from: 0xC954EBE59e851fB4E262b10221C42Ef44bE48DC2 to:
transact to HealthcareRecords.addRecord pending ...

view on etherscan

[block:6477447 txIndex:13] from: 0xc95...48dc2 to: Healthcar
call to HealthcareRecords.getPatientRecords


[call] from: 0xC954EBE59e851fB4E262b10221C42Ef44bE48DC2 to:
call to HealthcareRecords.getPatientRecords
```


Home
1_Storage.sol
FL.sol
HealthcareRecords_metadata.json
HealthcareRecords.sol 1
uint256 internal recordID
contracts/HealthcareRecords.sol 8:8

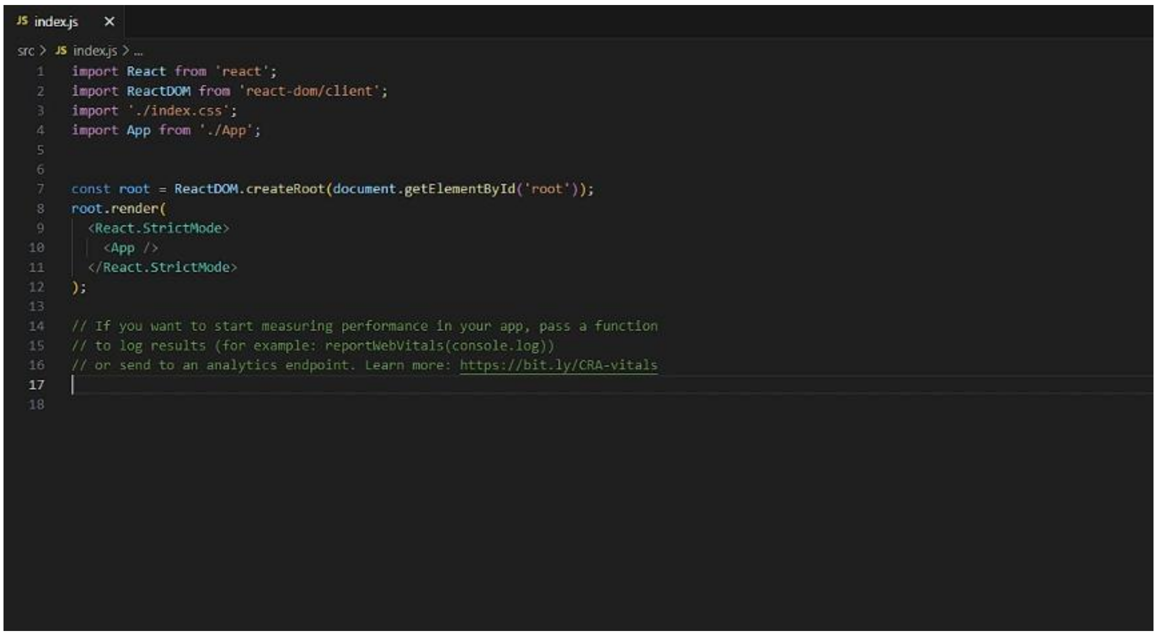
```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract HealthcareRecords {
    address owner;

    struct Record {
        uint256 recordID;
        string patientName;
        string diagnosis;
        string treatment;
        uint256 timestamp;
    }

    mapping(uint256 => Record[]) private patientRecords;

    mapping(address => bool) private authorizedProviders;

    modifier onlyOwner() {
        require(msg.sender == owner, "Only owner can perform this function");
        _;
    }

    modifier onlyAuthorizedProvider() {
        require(authorizedProviders[msg.sender], "Not an authorized provider");
        _;
    }

    constructor() {
        owner = msg.sender;
```



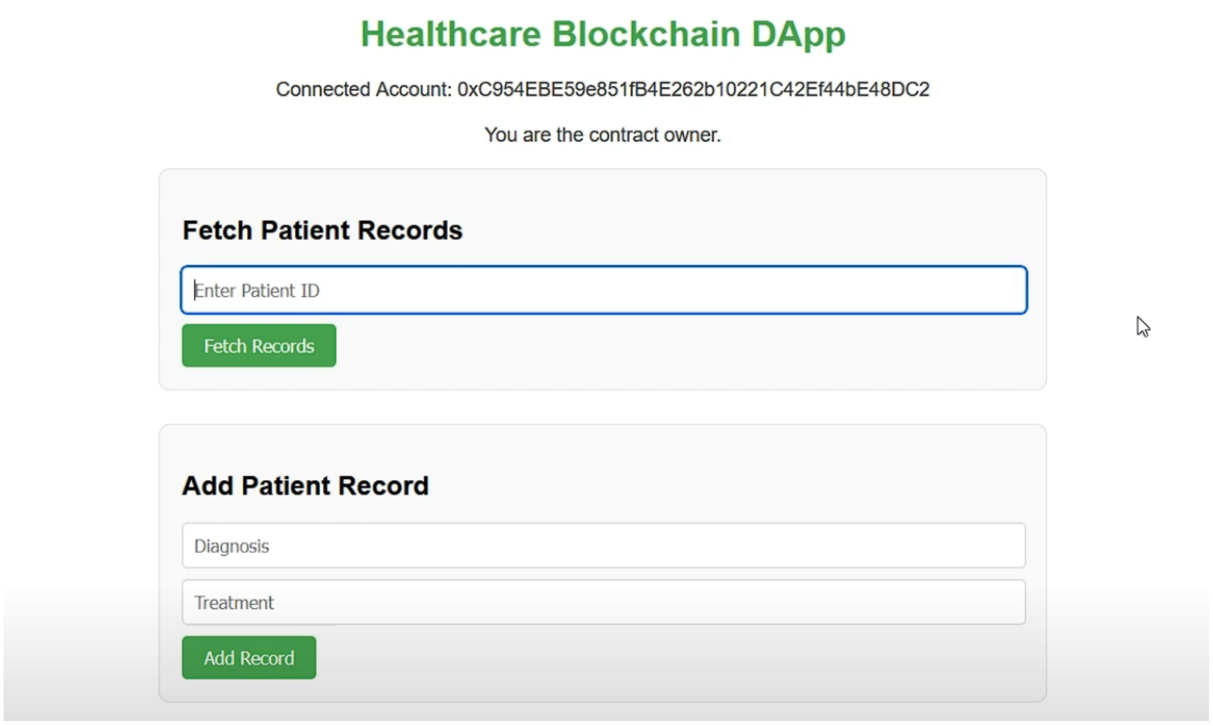
```
    constructor() {
        owner = msg.sender;
    }

    function getOwner() public view returns (address) {
        return owner;
    }

    function authorizeProvider(address provider) public onlyOwner {
        authorizedProviders[provider] = true;

    }


    function addRecord(uint256 patientID, string memory patientName, string memory diagnosis, string memory treatment) public 

}
```



```
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
  );
}

export default App;
```



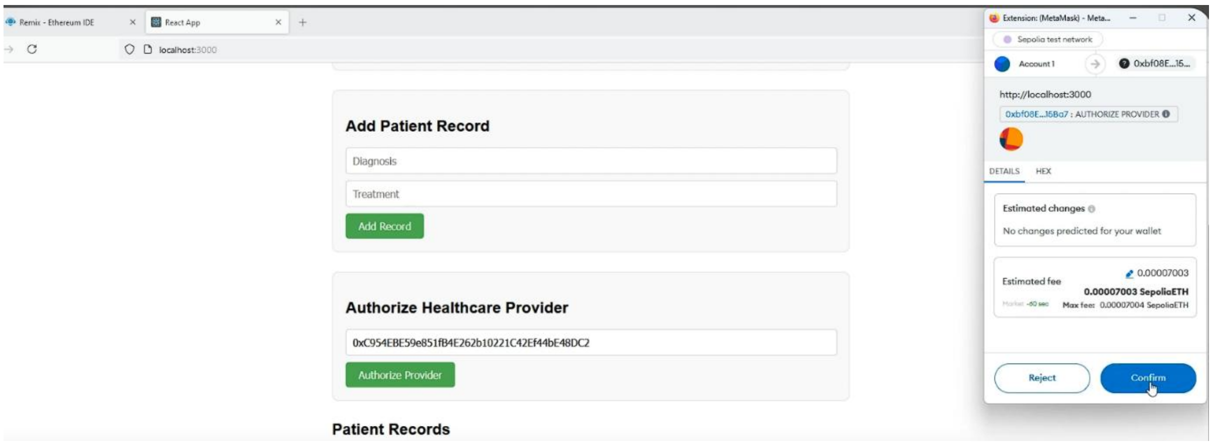
```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';


const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals

```



Healthcare Blockchain DApp

Connected Account: 0xC954EBE59e851fB4E262b10221C42Ef44bE48DC2

You are the contract owner.



0xC954EBE59e851fB4E262b10221C42Ef44bE48DC2

http://localhost:3000

0xbf08E...15Ba7 : AUTHORIZE PROVIDER

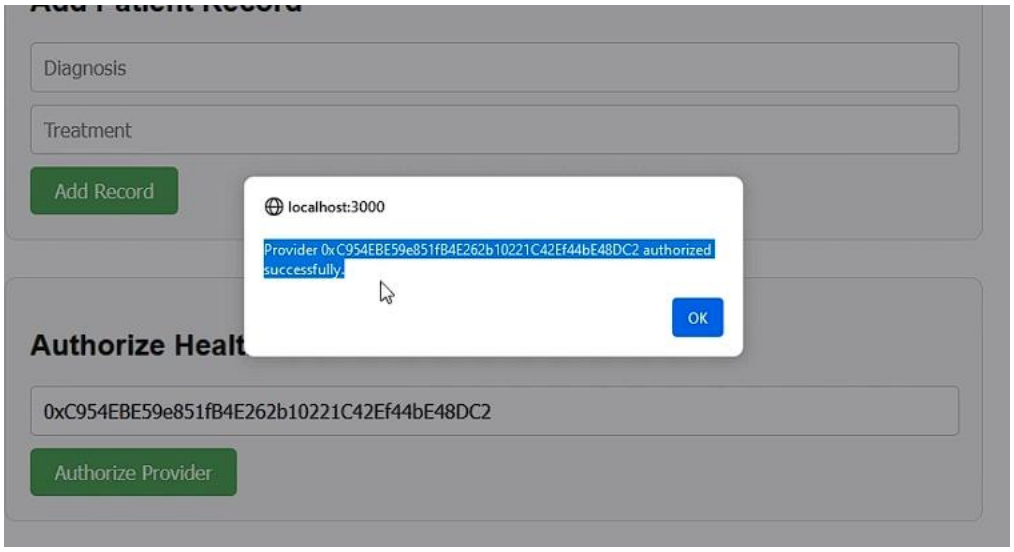
Estimated changes

No changes predicted for your wallet

Estimated fee 0.00007003

0.00007003 SepoliaETH

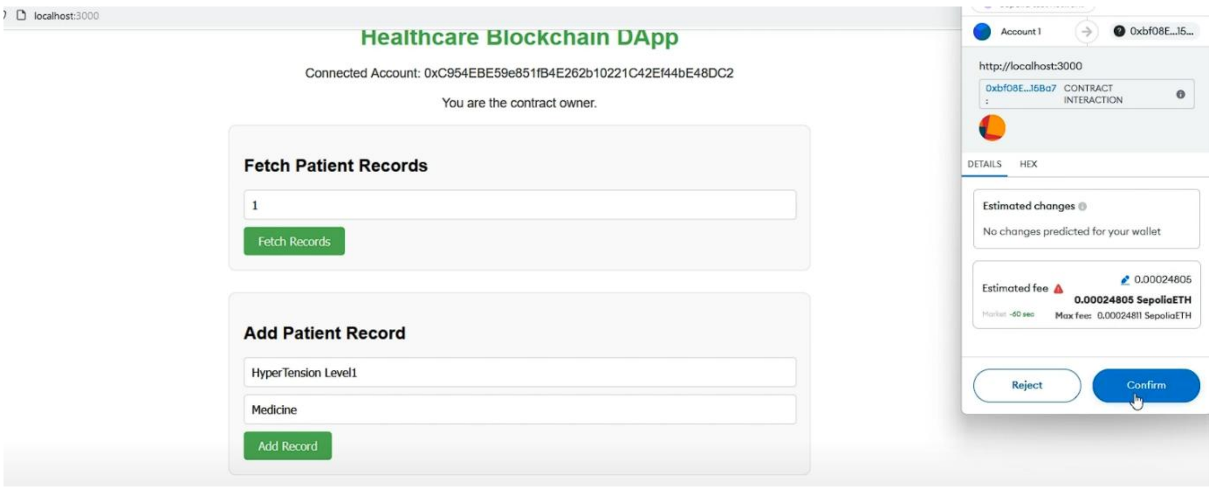
Market ~60 sec Max fee: 0.00007004 SepoliaETH



localhost:3000

Provider 0xC954EBE59e851fB4E262b10221C42Ef44bE48DC2 authorized successfully.

0xC954EBE59e851fB4E262b10221C42Ef44bE48DC2


localhost:3000

Healthcare Blockchain DApp

Connected Account: 0xC954EBE59e851fB4E262b10221C42Ef44bE48DC2

You are the contract owner.

Fetch Patient Records

1

Fetch Records

Add Patient Record

HyperTension Level1

Medicine

Add Record

Account 1 → 0xbf08E...15...

http://localhost:3000

0xbf08E...15Ba7 : CONTRACT INTERACTION

DETAILS HEX

Estimated changes

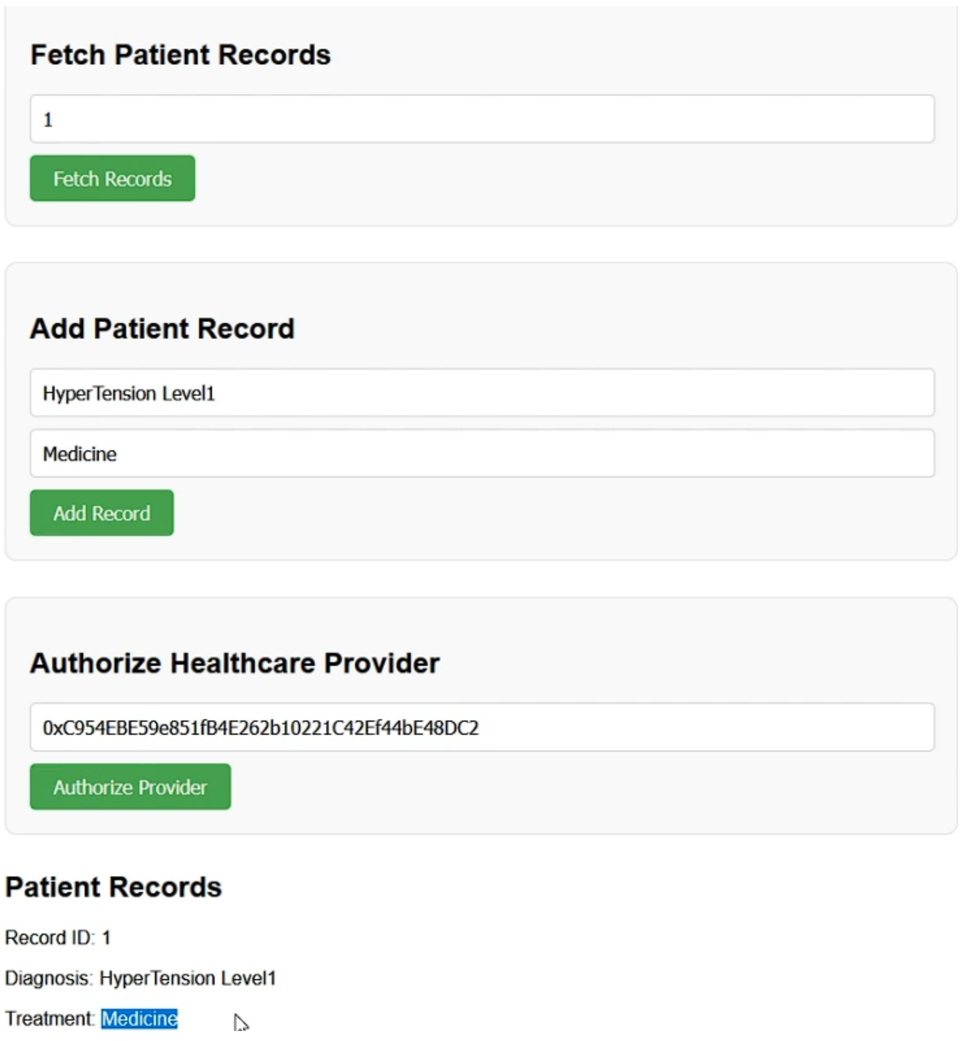
No changes predicted for your wallet

Estimated fee 0.00024805

0.00024805 SepoliaETH

Market ~60 sec Max fee: 0.00024811 SepoliaETH

Reject Confirm


Fetch Patient Records

1

Fetch Records

Add Patient Record

HyperTension Level1

Medicine

Add Record

Authorize Healthcare Provider

0xC954EBE59e851fB4E262b10221C42Ef44bE48DC2

Authorize Provider

Patient Records

Record ID: 1

Diagnosis: HyperTension Level1

Treatment: Medicine

CONCLUSION

In conclusion, the development of a blockchain-based healthcare application represents a significant advancement in the management of medical records and patient data. By leveraging the unique capabilities of blockchain technology, this project addresses critical challenges such as data security, privacy, and interoperability within the healthcare sector. The implementation of smart contracts not only automates processes but also enhances patient empowerment by providing individuals with greater control over their health information and consent management.

This application aims to foster trust and collaboration between patients and healthcare providers, ultimately leading to improved healthcare outcomes. By ensuring that medical records are securely stored and easily accessible, the project seeks to facilitate timely and informed decision-making in patient care. As the healthcare landscape continues to evolve, this innovative solution offers a promising pathway toward a more efficient, secure, and patient-centric healthcare ecosystem.

REFERENCES

[1] Bauer, D.P. (2022). ERC-20: Fungible Tokens. In *Getting Started with Ethereum*. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-8045-4_3

[2] Victor, F., & Lüders, B.K. (2019). Measuring Ethereum-Based ERC20 Token Networks. In Goldberg, I., & Moore, T. (Eds.), *Financial Cryptography and Data Security*. FC 2019. Lecture Notes in Computer Science, vol 11598. Springer, Cham. https://doi.org/10.1007/978-3-03032101-7_8

[3] Lee, W.M. (2019). Testing Smart Contracts Using Ganache. In *Beginning Ethereum Smart Contracts Programming*. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-5086-0_7

[4] Larson, S. Deploying Smart Contracts with Truffle and Ganache. Grizzly Peak Software. Retrieved from https://www.grizzlypeaksoftware.com/articles?id=5vWBWo4Zpi02FSVCmQunxk