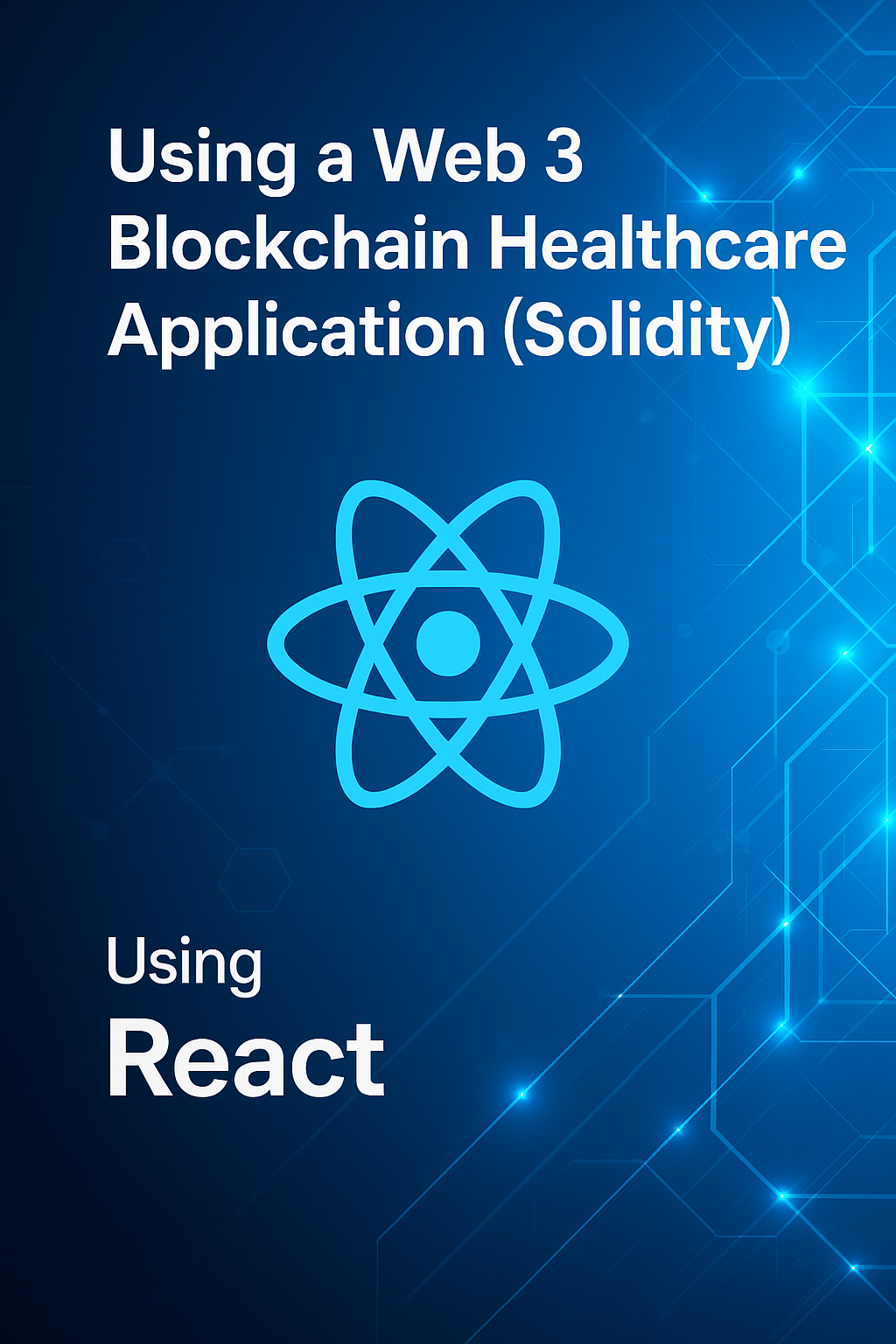
**Project Title**

**Using a Web 3 Blockchain Healthcare Application (Solidity)**

**Using React**



**Name - Sneha Gite**

**Class - Ty(01) IT**

**Abstract**

In the modern digital age, healthcare data has become one of the most critical assets, yet it remains highly vulnerable due to reliance on centralized systems that are prone to data breaches, manipulation, and loss of patient privacy. Current healthcare infrastructure often lacks transparency and the ability to offer patients full ownership and control over their personal medical records. This project aims to solve these problems by developing a secure, decentralized healthcare application using Web3 technology and Solidity smart contracts deployed on the Ethereum blockchain.

The proposed system enables patients and medical professionals to interact with a decentralized application (dApp) that securely manages healthcare reports. By leveraging blockchain's core features — decentralization, transparency, immutability, and cryptographic security — the system ensures that medical records cannot be tampered with or accessed without proper authorization. Smart contracts written in Solidity enforce rules regarding data access, modification, and sharing. These contracts record every interaction (such as uploads, permissions, and accesses) on the blockchain, providing a complete and verifiable audit trail.

The application also utilizes Web3 technologies to connect the front-end interface with blockchain-based smart contracts, allowing real-time and trustless interaction between users and the Ethereum network. Patients can upload reports, grant or revoke access to doctors, and view the history of data access securely. This gives users full ownership of their data and minimizes the risk of data leaks or unauthorized manipulation.

By combining Web3, blockchain, and smart contracts, this healthcare application proposes a modern, reliable, and user-centric solution to healthcare data management. It enhances trust, security, and accountability in the healthcare sector while empowering patients with greater control over their health records. This approach has the potential to significantly transform digital healthcare systems and set new standards for medical data protection and transparency.

**Introduction**

Blockchain technology has emerged as a revolutionary innovation, transforming the way digital information is stored, shared, and verified. At its core, blockchain is a **distributed ledger system** that maintains a **secure, transparent, and immutable record of transactions** across a decentralized network. This eliminates the need for central authorities and introduces trustless interaction among participants. One of the most impactful extensions of blockchain technology is the use of **smart contracts** — self-executing programs that automatically enforce predefined rules and agreements without the need for intermediaries.

Smart contracts, particularly those developed using **Solidity** on platforms like **Ethereum**, allow developers to create decentralized applications (dApps) that are secure, transparent, and tamper-proof. These applications can be used in various sectors, including finance, supply chain, identity verification, and most notably, **healthcare**.

In the healthcare industry, data security, privacy, and integrity are of paramount importance. However, traditional healthcare data systems are **centralized**, often siloed, and vulnerable to unauthorized access, data breaches, and manipulation. Patients typically have **limited control over their own medical records**, which are scattered across different hospitals and systems. This leads to issues in data sharing, verification, and long-term accessibility — especially in critical, time-sensitive medical situations.

To address these real-world challenges, this project presents a **Web3-based blockchain healthcare application** that utilizes **Solidity smart contracts** to securely manage and share medical records on the Ethereum blockchain. The proposed system allows patients to store their reports in a decentralized manner and control access to their data by granting permissions to healthcare professionals. Every access and update is recorded immutably, ensuring a transparent and auditable history of interactions.

This application is highly relevant to today’s healthcare landscape where **data breaches and loss of patient autonomy are frequent concerns**. By leveraging blockchain and Web3 technologies, the solution offers a new approach to **secure, private, and patient-centric healthcare management**. It not only improves data security and interoperability but also empowers individuals with **ownership of their health information**, paving the way for smarter, more ethical digital healthcare ecosystems.

**Problem Statement**

In today’s healthcare infrastructure, **data management systems are primarily centralized**, meaning that patient records are stored and controlled by hospitals, clinics, or third-party service providers. While these systems have helped digitize medical records, they come with a range of **critical limitations** that impact security, privacy, data ownership, and interoperability.

One of the major issues is the **lack of patient control** over personal health data. Medical records are often fragmented across various systems, making it difficult for patients to access or share their complete medical history with new healthcare providers. This fragmentation leads to **inefficiencies, repeated tests**, and **delays in diagnosis or treatment**. Additionally, patients have limited visibility into who has accessed or modified their data, reducing transparency and accountability.

Another significant problem is the **vulnerability to data breaches and unauthorized access**. Centralized servers are attractive targets for cyberattacks, and history has shown that even major healthcare institutions are not immune to large-scale data leaks. Once breached, sensitive medical records can be exposed, sold, or manipulated, leading to severe privacy concerns and loss of trust.

Furthermore, existing systems **lack interoperability** — the ability to seamlessly share and verify records between different healthcare providers and platforms. This limitation affects cross-institution collaboration and makes it challenging to maintain consistent and up-to-date patient histories, especially in emergencies.

Finally, current solutions often depend on **manual audits and administrative processes** to ensure compliance and traceability. These methods are time-consuming, error-prone, and susceptible to tampering, making it difficult to verify the integrity of medical records.

Given these challenges, there is a pressing need for a more **secure, transparent, and decentralized** approach to healthcare data management. The application of **blockchain technology and smart contracts** presents a promising solution by enabling patient-controlled, tamper-proof medical record systems with built-in auditability and access control — addressing the shortcomings of conventional methods

**Main Objectives**

1. **To develop a decentralized healthcare application using Web3 and Solidity smart contracts** for secure storage, access, and management of patient medical records on the Ethereum blockchain.
2. **To ensure patient data privacy, integrity, and ownership** by implementing a permission-based system that allows users to control who can access their health information.
3. **To enhance transparency and traceability** in healthcare data interactions through immutable logging of all actions (upload, access, update) on the blockchain ledger

**System Analysis**

**1. Integrated Development Environment (IDE)**

* **Remix IDE**: An online Solidity development environment used to write, compile, deploy, and test smart contracts.
* Offers built-in debugging tools, real-time deployment features, and easy integration with MetaMask and Ethereum test networks.

**2. Platform / Network**

* **Ethereum Blockchain** (Testnet: *Sepolia*): The application is deployed on the Ethereum test network to simulate real-world blockchain behavior without incurring actual gas fees.
* **Web3 Frontend**: Built using HTML, CSS, JavaScript, and **Ethers.js** for interaction with smart contracts through MetaMask.

**3. Smart Contract Features**

**Constructor**

* Initializes the smart contract upon deployment.
* Can be used to assign the contract owner or set default values (e.g., administrator address).

**State Variables**

* Used to store patient data, report hashes (e.g., IPFS links), and access permissions.
* Example:

address public owner;

mapping(address => string[]) public medicalReports;

mapping(address => mapping(address => bool)) public accessControl;

**Functions**

* **View Functions**: Used to read data without modifying the blockchain state. These do not consume gas.

function getReports(address user) public view returns (string[] memory);

**Non-payable Functions**: Handle actions like uploading reports or managing access permissions. These modify state and thus require gas.

function uploadReport(string memory reportHash) public;

function grantAccess(address doctor) public;

**Payable Functions** *(if any)*: Can be added for premium features like storage upgrades but are not the main focus of this basic healthcare application.

**Modifiers**

Used to restrict access or add reusable conditions.

**Access Control**

* Ensures only the right users (e.g., the patient or authorized doctor) can view or modify specific data.
* Implemented using **mappings** and **modifiers** to verify identity.

**4.Gas Optimization Techniques (If Any)**

* **Efficient storage usage**: Reports are stored as IPFS hashes instead of large text, reducing on-chain storage.
* **Use of memory instead of storage** in functions where persistent data isn't needed.
* **Avoiding redundant state updates**: Conditions are checked before writing to state variables to avoid unnecessary gas usage

**Frontend Details**

**1. index.html**

This is the main HTML file that structures the user interface.

* It includes:
  + A connection button to connect the user's Ethereum wallet (e.g., MetaMask).
  + Input fields and buttons to:
    - Add a healthcare record (record ID and data).
    - Retrieve a record by ID.
    - Display status messages and retrieved record data.
* Links to external libraries like Ethers.js via CDN.
* Includes a <script> tag pointing to Healthcare.js for blockchain interactions.

**2. styles.css**

This is the stylesheet responsible for the visual layout and design.

* Provides styling for:
  + Buttons
  + Inputs
  + Layout containers (e.g., centering content, adding padding/margins)
* Ensures a cleaner and more professional look for the app.

**3. Healthcare.js (assumed, referenced in index.html)**

Although this file isn't uploaded, based on the README and HTML, this JS file:

* Uses **Ethers.js** to connect to the Ethereum network.
* Contains the logic to:
  + Connect the user's wallet.
  + Interact with the smart contract methods like addRecord() and getRecord().
  + Handle ABI and contract address configuration.
  + Show results or errors to users.

**4. README.md**

Provides setup instructions:

markdown

CopyEdit

Before starting this app, run:

npm install

Update the smart contract address and abi in Healthcare.js file.

To run the project:

npm start

This confirms the project uses **React or a Node.js setup** with npm start, indicating that index.html is either served directly or wrapped in a framework.

**5. package.json and package-lock.json**

These manage dependencies. Notable packages might include:

* ethers (blockchain interaction)
* react-scripts or similar (if this is a React app)
* Other UI or tooling libraries

**Conclusion**

The Web3 Healthcare Application built using Solidity and deployed on the Ethereum blockchain successfully demonstrates how decentralized technology can be leveraged for secure, transparent, and tamper-proof management of health records. By using smart contracts, patients can safely store their medical information and selectively grant access to authorized healthcare providers. This eliminates the need for centralized databases, reduces the risk of data breaches, and ensures patient privacy and control. The project reflects how blockchain can solve real-life problems in healthcare data management through automation, trust, and traceability.

**Limitations**

While the application introduces an innovative solution, several limitations were observed:

1. **High Gas Costs**: Storing and interacting with data on Ethereum incurs gas fees, making frequent operations expensive.
2. **Limited File Support**: On-chain storage is not ideal for large files like medical images or PDFs.
3. **User Accessibility**: Users must understand how to use wallets like MetaMask and manage blockchain transactions, which may not be easy for non-technical users.
4. **Real-Time Updates**: Due to blockchain transaction confirmation times, updates are not instant and may delay user interactions.
5. **Compliance Issues**: Blockchain immutability makes it difficult to comply with data deletion requests under regulations like GDPR.

**Future Scope**

The project opens doors to several enhancements and broader real-world adoption:

1. **Off-Chain Storage (IPFS)**:
   * Integrate with decentralized file storage platforms like IPFS to store large medical records securely while keeping references on-chain.
2. **Layer 2 Integration**:
   * Use Layer 2 solutions like Polygon or Arbitrum to reduce gas fees and improve transaction speed.
3. **Role-Based Access and Consent Management**:
   * Implement finer-grained access control for different user roles such as patients, doctors, hospitals, and insurance providers.
4. **Mobile App Development**:
   * Create a mobile-friendly version of the application using React Native or Flutter for easy access on smartphones.
5. **AI for Predictive Healthcare**:
   * Use anonymized blockchain data to feed AI models for predicting health trends or providing medical insights.

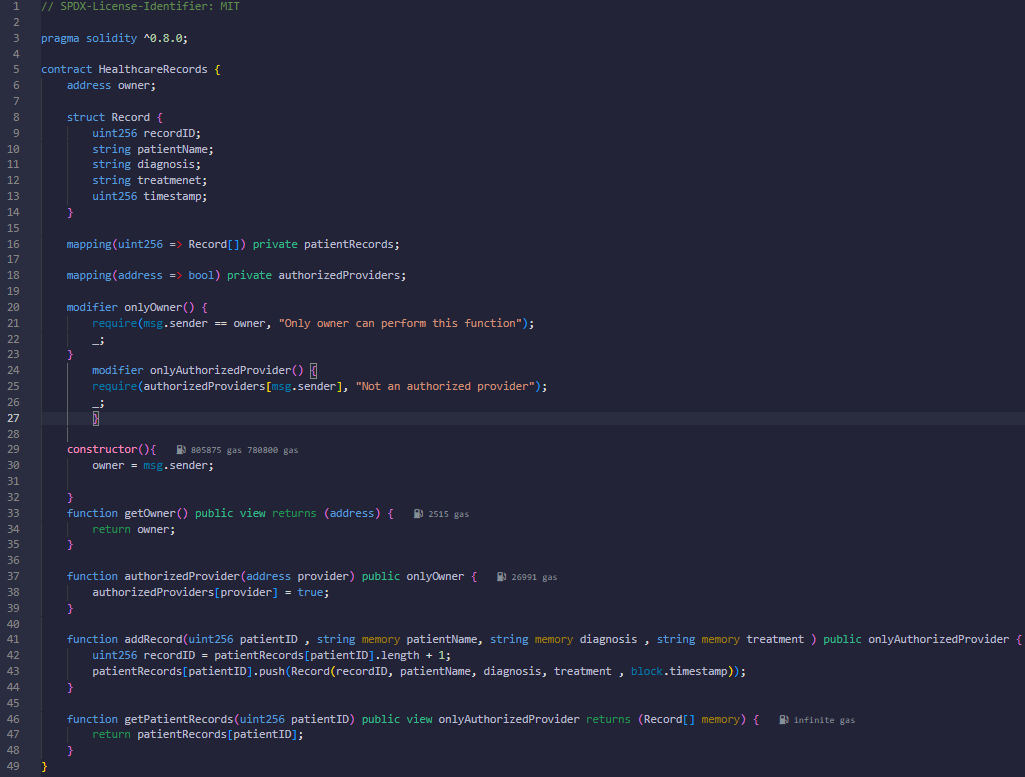
**Research Papers and Articles**

1. **Tapscott, D., & Tapscott, A.** (2016). *Blockchain Revolution: How the Technology Behind Bitcoin and Other Cryptocurrencies is Changing the World*. Penguin.
   * This book provides an in-depth overview of how blockchain is transforming various industries, including healthcare.
2. **Mettler, M.** (2016). *Blockchain Technology in Healthcare: The Revolution Starts Here*. *IEEE 18th International Conference on e-Health Networking, Applications and Services (HealthCom)*.
   * This paper discusses the potential applications of blockchain in healthcare, focusing on its impact on data security and patient privacy.
3. **Ethereum Smart Contract Development on Udemy**:

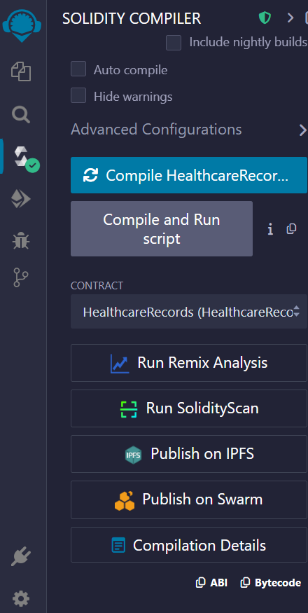
* *Ethereum and Solidity: The Complete Developer's Guide*: https://www.udemy.com/course/ethereum-and-solidity-the-complete-developers-guide/
* A beginner-to-advanced course on Ethereum and Solidity development, perfect for learning how to create decentralized applications and smart contracts.

**Appendix**

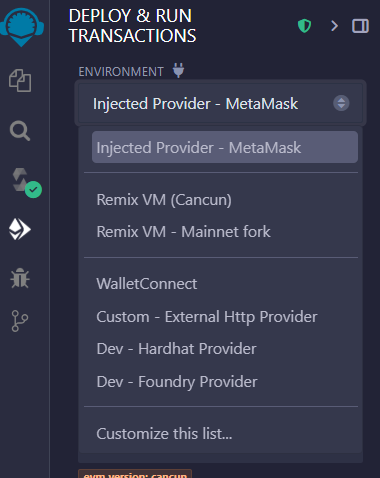
1. This Solidity smart contract, HealthcareRecords, securely manages patient medical records. The contract owner can authorize healthcare providers to add or view records. Each record contains patient details like name, diagnosis, treatment, and timestamp, stored using a mapping by patient ID. Access is controlled using modifiers to ensure only the owner or authorized providers can perform specific actions. This setup ensures secure and private handling of medical data on the blockchain**.**



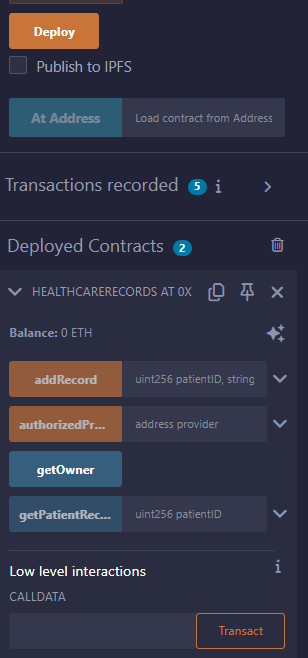
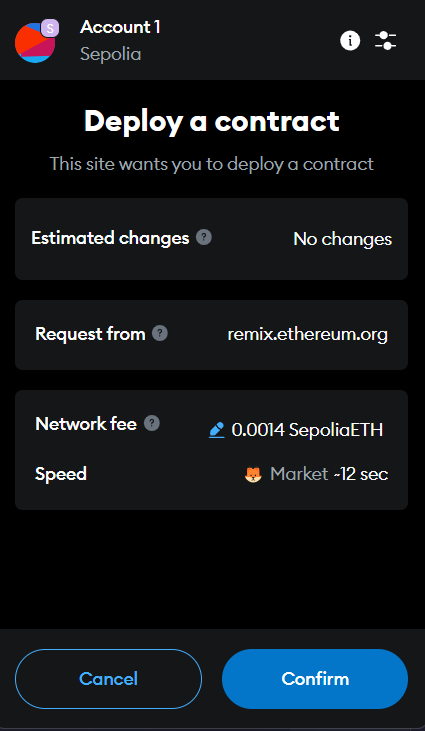
1. The Solidity Compiler tab in Remix IDE is used to compile smart contracts written in the Solidity language. It allows you to compile the selected contract, in this case, HealthcareRecords, by clicking the "Compile" button. You can also choose to auto compile or hide warnings using the checkboxes. Additional tools include running static analysis with Remix Analysis or performing a security audit using SolidityScan. There are options to publish the compiled contract to decentralized storage platforms like IPFS or Swarm. The "Compilation Details" section provides important technical data such as the contract’s ABI and bytecode, which are essential for deployment and interaction.



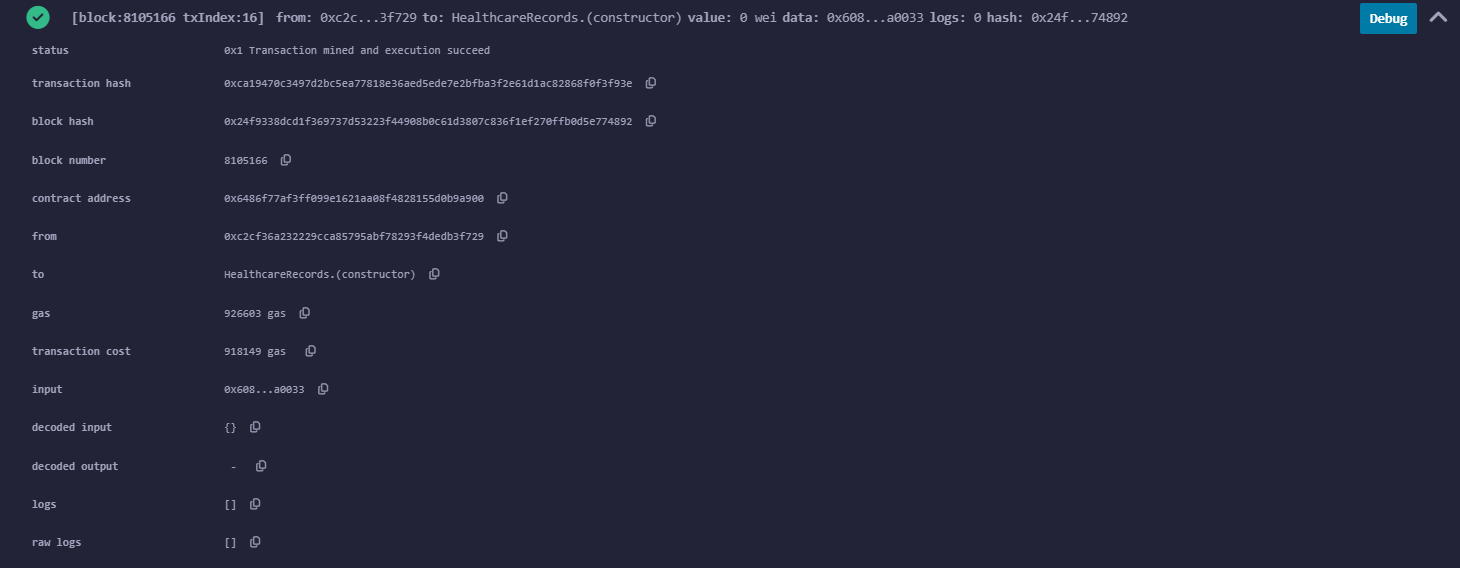
1. This image shows the **"Deploy & Run Transactions"** tab in Remix IDE, where you choose the environment to deploy and interact with your smart contract. The dropdown menu offers various options like **Injected Provider - MetaMask** for connecting to MetaMask wallets, **Remix VM** for local testing, and **WalletConnect** for mobile wallet connections. You can also use custom networks such as Hardhat, Foundry, or external HTTP providers. This setup lets you deploy your contract to different blockchain networks for testing or production use.



1. This image shows the **Deployed Contracts** section in Remix IDE after deploying the HealthcareRecords contract. It displays the contract's address and available functions like addRecord, authorizedProvider, getOwner, and getPatientRecords. These buttons let you interact with the smart contract directly from the interface by providing input values and executing functions. It also shows the contract's balance (0 ETH) and provides a section for low-level interactions using calldata.

1. This image shows the transaction details of successfully deploying the HealthcareRecords smart contract on the blockchain using Remix. It confirms the transaction was mined with status 0x1 (success), provides the **contract address**, **transaction hash**, **block number**, **gas used**, and **input data**. The contract was deployed from a specific wallet address and includes no logs, indicating a clean deployment without events.



1. This image shows interaction with the deployed HealthcareRecords smart contract in Remix IDE. Here’s a short explanation:

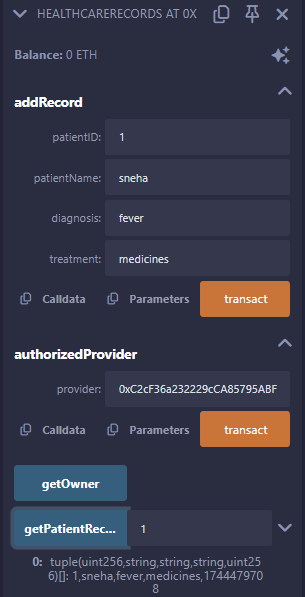
The **authorizedProvider** function is used to authorize an address (likely the user's own) to add medical records.

After authorization, the **addRecord** function is used to add a new patient record with ID 1, name Sneha, diagnosis fever, and treatment medicines.

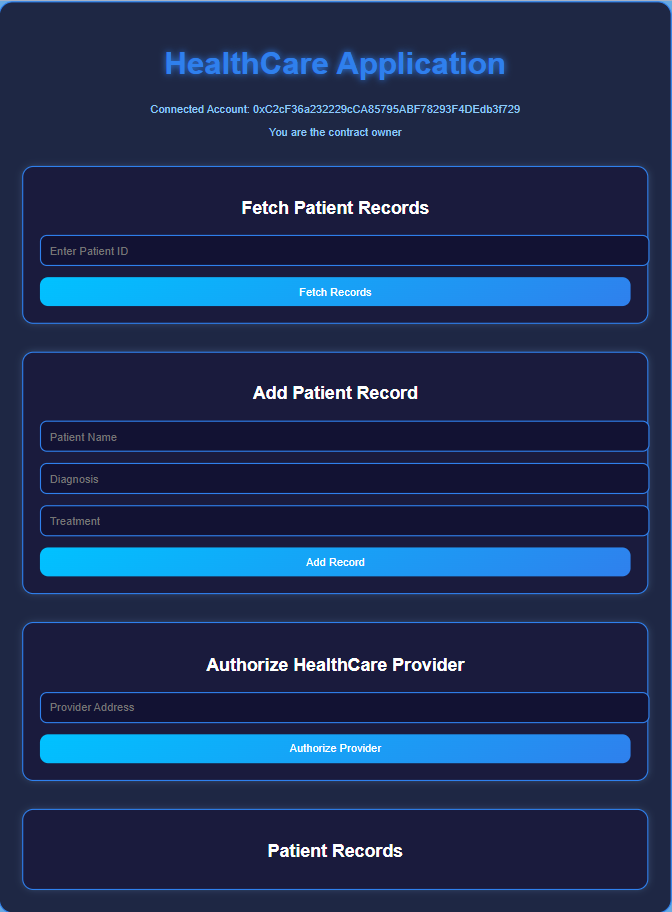
The **getOwner** button retrieves the contract owner's address.

The **getPatientRecords(1)** call fetches and displays the added record for patient ID 1, confirming successful storage with details and a timestamp.

This demonstrates secure and structured medical data entry on the blockchain.



1. This image shows the frontend interface of a **decentralized HealthCare Application** that interacts with a blockchain-based smart contract. The interface connects to MetaMask and displays the connected user's wallet address, confirming if they are the contract owner. It has several key features: users can fetch patient records by entering a patient ID, and authorized providers can add new patient records, including the patient's name, diagnosis, and treatment details. Additionally, the contract owner can authorize healthcare providers by entering their address. The application also has a section where patient records are displayed after being fetched. The interface is designed for ease of use, enabling smooth interaction with the smart contract for managing healthcare data on the blockchain.



1. **Fetch Patient Records Section:**

Allows the user to input a record ID.

On clicking "Fetch Records", it retrieves the corresponding patient data from the blockchain.

* **Add Patient Record Section:**

Contains input fields for:

Patient Name

Diagnosis

Treatment

On clicking "Add Record", the entered data is stored on the blockchain using a smart contract function.

* **Authorize HealthCare Provider Section:**

Allows the user to input an Ethereum wallet address.

On clicking "Authorize Provider", that address is granted permission to access or manage patient records.

* **Patient Records Display Section:**

Shows all stored patient records from the blockchain.

Each record includes:

Record ID

Patient Name

Diagnosis

Treatment

Timestamp (date and time the record was added)

* **Web3 Functionality:**

Likely built using JavaScript and Ethers.js or Web3.js to interact with a deployed smart contract.

Ensures data is securely stored and retrieved from a decentralized blockchain network.

* **UI Design:**

Uses a dark-themed layout with styled buttons and inputs for a clean and modern user experience.

Organized into clearly defined sections for each functionality.

