MediChain: Blockchain-Based Healthcare dApp

Shlok Nandanwar - D16ADBB (37)

Contents

1 Executive Summary

MediChain revolutionizes healthcare data privacy and insurance through a decentralized platform on Ethereum. Patients and insurers interact using smart contracts written in Solidity, a React.js frontend, decentralized storage (IPFS via Infura), and secure wallets (MetaMask). This architecture ensures permissioned control, transparency, and an immutable audit trail for all medical record and insurance interactions, directly addressing the risks from fragmented records, privacy breaches, and claim fraud.

2 Problem Statement

Patient data privacy is essential due to personally sensitive medical records, test results, insurance eligibility, and more. Traditional systems risk financial loss, identity theft, stigma, and discrimination through breaches or unauthorized sharing. Insurance access is further complicated by poor data and lack of patient authority.

3 Architecture and Components

3.1 System Diagram

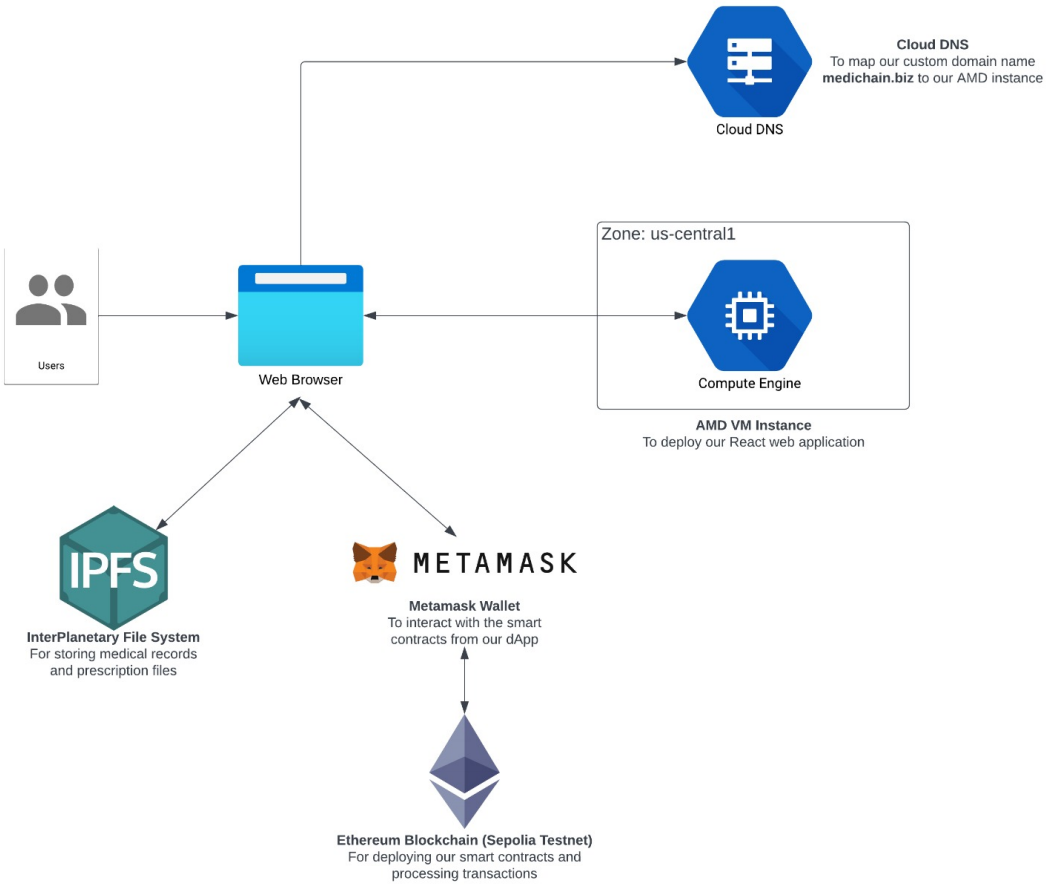

Figure 1: System Architecture Overview

3.2 Smart Contracts (Solidity)

- Implements roles: Patient, Doctor, InsuranceProvider, Admin.
- Functions for adding/updating records, policy creation and tracking, claims, permissions.
- Data stored: patient-to-IPFS hash mapping, policy structs, event logs for every state change.
- Access restrictions (modifiers): role-based access; e.g., only doctors add records, only patients file claims.
- Custom events used for external apps to listen for record changes, claims, policy creation.
- No mainnet deployment yet; contracts are unaudited.

3.3 Frontend (React)

- React.js web client handles registration/login, file upload, record viewing, policy management.
- Integrates directly with MetaMask for wallet selection and transaction signing.
- Asynchronous UI status (pending, confirmed, error) for on-chain interactions.
- Uses `ipfs-http-client` for IPFS file handling; references stored on-chain.
- Modular UI for different roles: dashboards for patients, doctors, insurance managers, admins.

3.4 Decentralized Storage (IPFS/Infura)

- Medical files are stored on IPFS; blockchain stores only resulting file hashes for verifiability.
- Infura acts as a gateway for reliable IPFS uploads/downloads.
- **Note:** Recent Infura API changes have caused integration bugs, requiring future updates.
- Data privacy: off-chain files must be encrypted if containing sensitive information before being uploaded.
- Pinning solutions may be required for long-term data persistence.

4 Key Features and Functionalities

4.1 User Roles

- Patients: Control record access, purchase policies, file/view claims.
- Doctors: Upload medical files, request access, update patient data.
- Insurance Providers: Create/manage insurance, verify and pay claims.
- Admins: Manage dispute resolution, consent logs, upgrade governance.

4.2 Data Management

- Every data event (add/update/share) emits a traceable on-chain log.
- Mappings tie Ethereum addresses to patient records and insurance policies for deterministic access.
- Solidity modifiers enforce role- and action-specific access.
- Patient consent tracked via smart contract events and mapping.

4.3 Insurance Policies and Claims

- Policy structs bind to patient wallet address, including coverage, premium, expiration, and state.
- Claims initiated on-chain by patient, reviewed (and paid) by insurer.
- Events for each stage of the claim allow real-time UI notifications.
- Planned upgrade: ERC-721 for insurance policies as unique tokens for direct trading/transfer.

4.4 Transactions

- Each sensitive action—record upload, policy update, claim submission—is a signed Ethereum transaction.
- Full transaction/audit trail provided by event logs retrievable via block explorers or UI.
- Gas costs present, but minimized with local testing.

5 Setup and Deployment Guide

5.1 Prerequisites

- Node.js and npm or yarn
- Truffle (contract compilation/migration)
- Ganache (local testnet)
- MetaMask extension or Brave browser
- Infura IPFS account/gateway

5.2 Project Installation

1. Clone the repository: `git clone <repo-url>`
2. Install core dependencies:

```
npm i -g truffle
npm run client:install
npm run truffle:install
```

3. Start Ganache: Launch Ganache, use provided seed/mnemonic for test accounts.
4. Infura Setup: Generate IPFS API key and key secret, set environment variables:

```
REACT_APP_INFURA_PROJECT_ID=...
REACT_APP_INFURA_API_KEY_SECRET=...
REACT_APP_INFURA_DEDICATED_GATEWAY=....
```

5. Deploy Contracts and Start App:

```
npm run truffle:migrate
npm run client:start
```

6. Configure MetaMask: Import Ganache accounts, connect to `http://localhost:8545/`.

7. Access dApp: Open `http://localhost:3000/` to interact with MediChain.

6 Implementation Screenshots

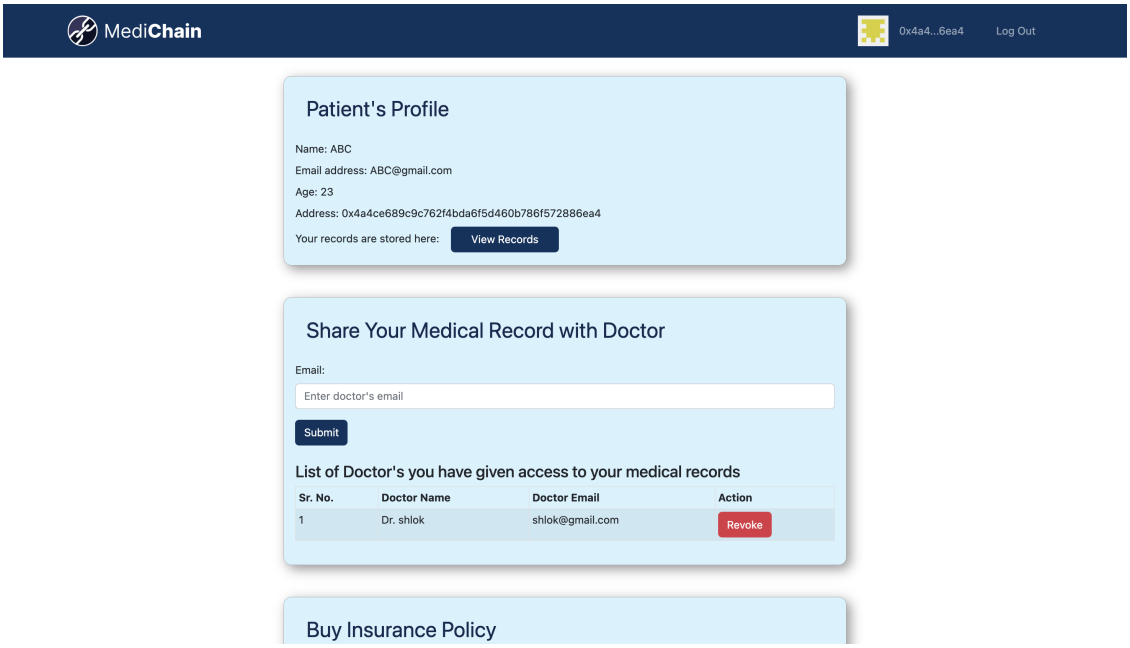

Figure 1: User registration and login interface

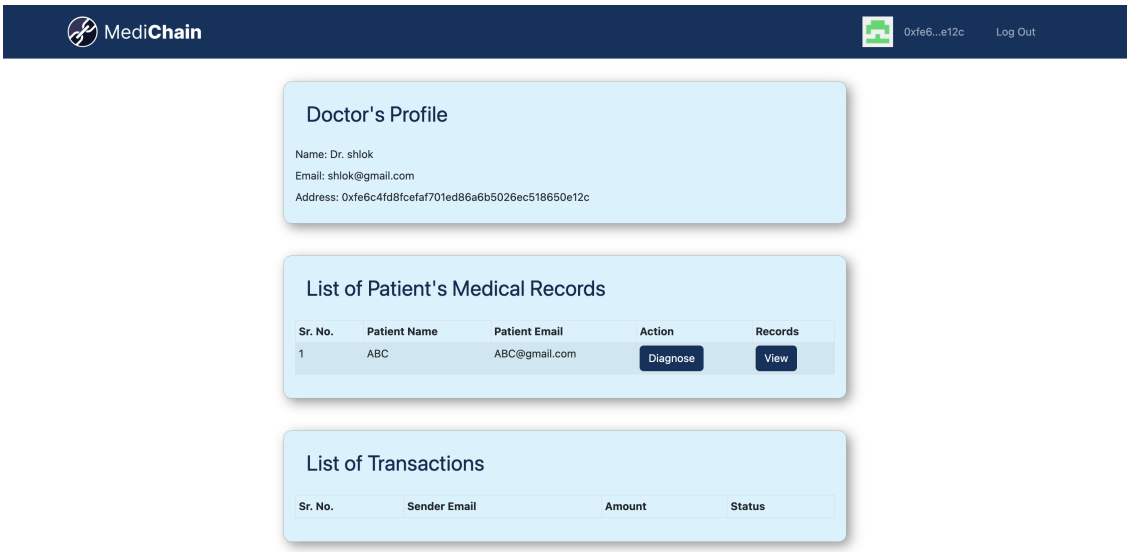

Figure 2: Doctor's dashboard; to view and diagnose patients

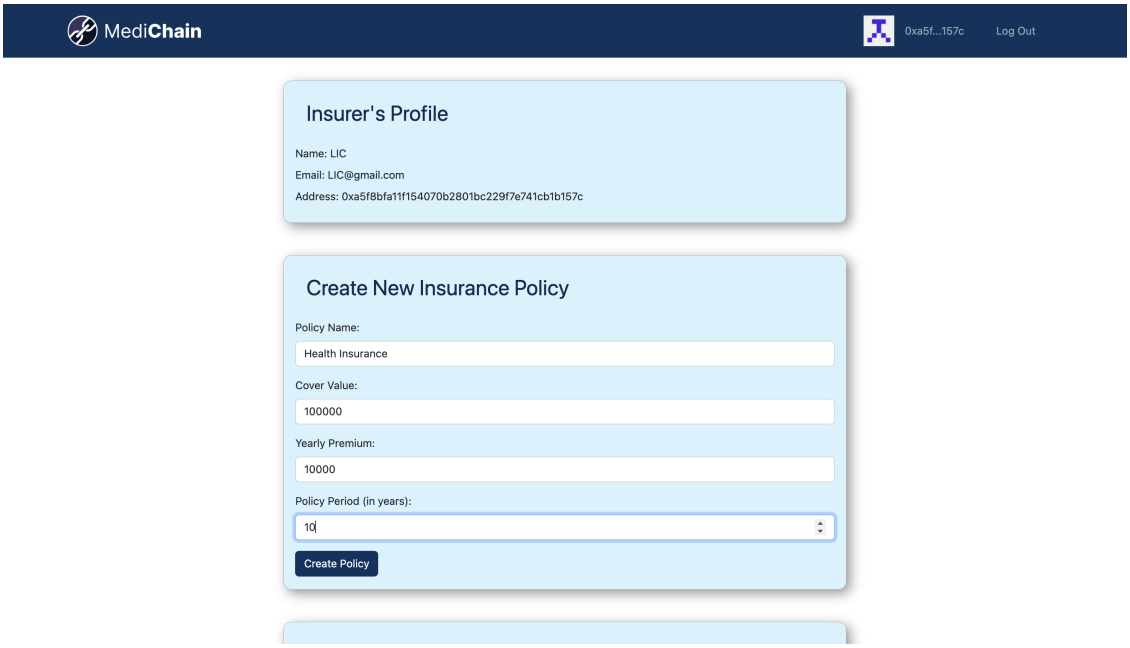

Figure 3: Insurance policy dashboard

7 Future Prospects

- Implement ERC-721 insurance policies for uniqueness/ownership.
- Allow patients to customize insurance plans based on preference, not insurer template.
- Integrate ML/AI models for premium prediction, fraud detection, and risk analytics.

- Develop advanced, privacy-preserving access controls and regulatory compliance modules.
- Pinning/IPFS reliability improvements for long-term clinical data storage.

8 Challenges and Limitations

- IPFS Integration: Bugs due to Infura API changes; requires updates and possible move to alternative pinning services.
- Smart Contract Audits: Contracts are unaudited, potentially vulnerable—do not use directly on mainnet.
- Off-chain Storage Security: Files must be encrypted before IPFS upload to protect privacy.
- Mainnet Risks: Gas cost volatility, transaction race conditions, risk of lost/failed operations.
- Compliance: Ongoing adaptation for HIPAA, GDPR, and other data protection standards.

9 Conclusion

MediChain demonstrates a forward-thinking solution for decentralized health data and insurance management, empowering user autonomy, secure medical recordkeeping, and transparent claims. The adoption of ERC-721 policies and ML-based analytics, combined with comprehensive security audits and reliable storage, are key next steps for full-scale deployment and trust in digital healthcare.