Medical Document Management Blockchain System

By:

ADARSH MISHRA (2215002)
SARTHAK SRIVASTAVA (2215024)
MADHAV SINGH (2215029)

Under the guidance of

Dr. RANJAY HAZRA

In Partial Fulfillment of the Fourth year UG Project Work

Department of Electronics & Instrumentation Engineering
NIT Silchar, Assam

Table of Contents

1.Introduction

- **What is Blockchain?**
  - A decentralized, immutable digital ledger that records data securely and transparently.
  - Prevents tampering, duplication, or unauthorized changes in stored information.
  - Ensures trust and transparency without a central authority.

- **Our Project : Medical Document Management Blockchain System**
  - A system designed to securely manage and store medical documents using Blockchain and IPFS.
  - Provides confidentiality, integrity, and traceability of all medical records.
  - Ensures patient-controlled access and full transparency in data usage.
  - **Identity in Our System**
    - Admin Identity
    - Patient Identity
    - Doctor Identity

  Each identity is unique, verifiable, and securely stored on the blockchain.

- **Structure of a Medical Record Block**
  - Data: Encrypted medical record or metadata. Timestamp: Creation or modification time.
  - Nonce: Random value ensuring block
  - uniqueness. Hash: Cryptographic link to the previous block — ensures immutability.

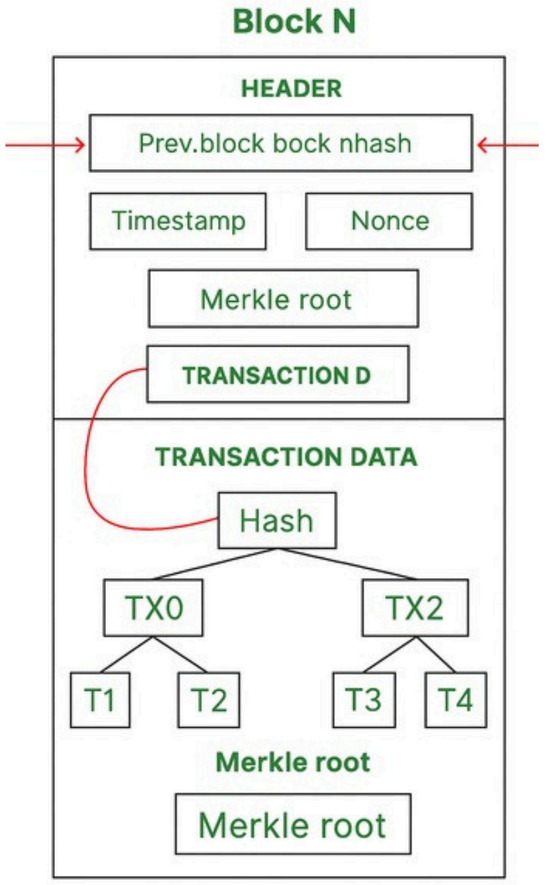


Fig 1: Merkel Tree.

2. Literature Survey

S.N	Details of the Paper	Findings	Limitations
1	A. Ekblaw, A. Azaria, J.D. Halamka, and A. Lippman, “MedRec: Medical Data Management on the Blockchain,” IEEE Open & Big Data Conference, 2016.	Introduced one of the first blockchain-based EHR systems; used Ethereum smart contracts for metadata sharing between providers and patients; improved transparency and data provenance.	No decentralized storage for documents; relies on provider nodes for validation; lacks multi-platform usability; does not implement patient-side access governance.
2	Q. Zhang, J. Walker, and B. White, “A Decentralized Identity and Storage System for Healthcare Data Using Blockchain,” IEEE Blockchain Conference, 2018.	Proposed decentralized identity management with blockchain-backed verification; uses distributed storage for secure patient data exchange; enhances interoperability.	Primarily focused on identity—not auditing, versioning, or full medical-record workflows; no multi-platform demonstration; limited clinical applicability.
3	M. Hyland et al., “Blockchain for Secure and Transparent Clinical Data Sharing,” IBM Journal of Research and Development, 2020.	Demonstrated permissioned blockchain for clinical data sharing; high trust, regulatory compliance, and enterprise-grade auditability; supports controlled access among institutions.	Works only on consortium (permissioned) networks; limited decentralization; not optimized for patient-governed access or public-chain transparency.

2. Literature Survey(Cont.)

S.N	Details of the Paper	Findings	Limitations
4	C.C. Agbo, Q.H. Mahmoud, and J.M. Eklund, “Blockchain Technology in Healthcare: A Systematic Review,” Healthcare, 2019.	Comprehensive survey showing blockchain improves trust, data provenance, and decentralized record management. Highlights benefits of transparency and patient empowerment.	No implementation model; does not provide architectural guidelines; limited analysis of deployment constraints such as scalability or storage.
5	A. Kakade et al., “Blockchain and IPFS-Based Framework for Medical Document Storage,” 2023.	Demonstrated blockchain + IPFS hybrid design; ensures immutability and decentralization; reduces blockchain storage burden through content addressing.	No audit trail; no multi-version record handling; lacks multi-platform user applications; limited real-world deployment structure.
6	P. Singh, S. Sagar, S. Singh, H.M. Alshahrani, M. Getahun, and O. Soufiene, “Blockchain-Enabled Verification of Medical Records Using Soul-Bound Tokens and Cloud Computing,” Scientific Reports, 2024.	Uses Ethereum-based soul-bound tokens (SBTs) to verify authenticity of medical documents; stores IPFS hashes and encrypted files; integrates Blockcert-style matching for anti-tampering assurances.	Heavy reliance on external encryption (AES/password-based); key loss risks permanent data inaccessibility; lacks decentralized storage recovery; no unified encryption key lifecycle management.

3. MOTIVATION AND OBJECTIVE

- **Motivation**
  - Medical records are often stored centrally, making them prone to data breaches, tampering, and unauthorized access. Patients have limited control over who accesses their health information.
  - Traditional systems lack transparency and traceability of medical data.
  - Blockchain and IPFS offer a way to securely store, share, and verify medical documents while ensuring patient privacy and trustless access control.

- **Objectives**
  - To develop a decentralized system for secure medical document storage using Ethereum and IPFS.
  - To enable patient-controlled access, ensuring only authorized users can view or modify records.
  - To maintain a transparent audit trail of all interactions for accountability.
  - To provide both web and desktop applications for accessibility and ease of use.
  - To demonstrate how blockchain technology enhances data integrity, confidentiality, and traceability in healthcare systems.

4. Methodology

This study builds upon the baseline framework proposed by Singh et al. (2024), which integrates Soul-Bound Tokens (SBTs), Ethereum smart contracts, and IPFS storage to verify medical records through decentralized identity and hash-based integrity checks. While the baseline model focuses primarily on conceptual verification architecture, our methodology extends it into a complete, deployable, and multi-platform system with end-to-end operational capabilities.

The methodology consists of four integrated layers:

- Blockchain Layer (Ethereum Smart Contracts)
- Decentralized Storage Layer (IPFS + Docker)
- Application Layer
- Access Control and Audit Layer

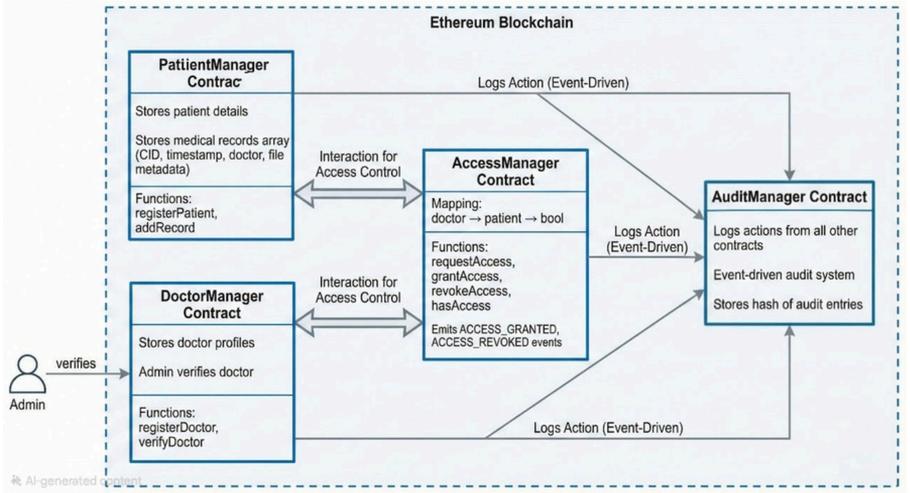


Fig 4.1: Smart Contract Arch

4. Methodology(Cont.)

- **Blockchain Layer (Ethereum Smart Contracts)**

  We designed and deployed smart contracts on an Ethereum-compatible network using Hardhat and Remix IDE.
  The contracts manage:
  - Document registration
  - Hash verification
  - Access permissions
  - User roles and identities

  Contract deployment includes automated build pipelines, ABI extraction, and address synchronization across both applications (Web and Desktop).

- **Decentralized Storage Layer (IPFS + Docker)**

  Following the baseline paper's approach of storing only document hashes on-chain, our system uses:
  - A dedicated IPFS Kubo node deployed via Docker
  - Persistent storage volumes
  - Customized CORS configuration for API access
  - Gateway-based document retrieval

  Each uploaded document is encrypted (off-chain), stored on IPFS, and referenced by a unique content identifier (CID).
  The CID and hash are then linked with the blockchain smart contract to ensure immutability.

4. Methodology(Cont.)

- **Application Layer**

  A responsive React-based interface provides:

  - MetaMask-based blockchain authentication
  - Role-based dashboards (patient, doctor, admin)
  - Secure document upload, download, and verification
  - Automatic IPFS communication
  - Real-time blockchain event listening

  This layer transforms the theoretical verification process into a practical, user-friendly workflow.

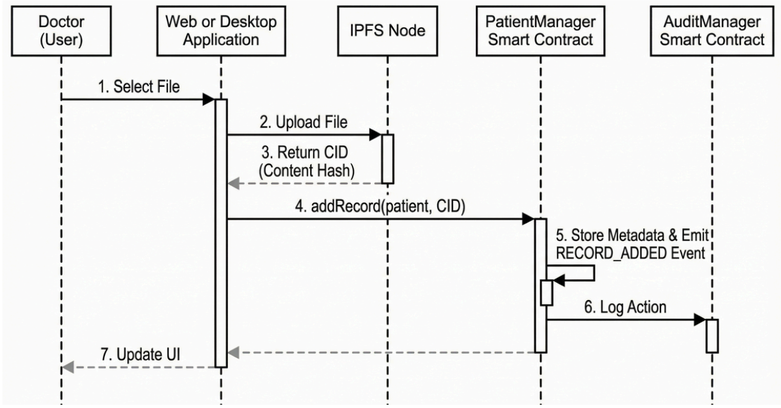


Fig 4.2: Seq Model

- **Access Control and Audit Layer**

  Extending the baseline paper, which mentions identity but does not implement practical access control, our system introduces:

  - Patient-controlled permission grants
  - Document-level access rights
  - Comprehensive activity logs
  - Verification event tracking
  - Transparent audit trails for accountability

  This layer enables real-world regulatory compliance and traceability.

5. Work Done

- Project Setup and Contract Development
  - Initialized Hardhat workspace
  - Installed dependencies and toolbox
  - Built Solidity smart contracts for document registration and verification
  - Deployed contracts to local blockchain and configured with applications
  - Extracted and integrated ABI files for both clients

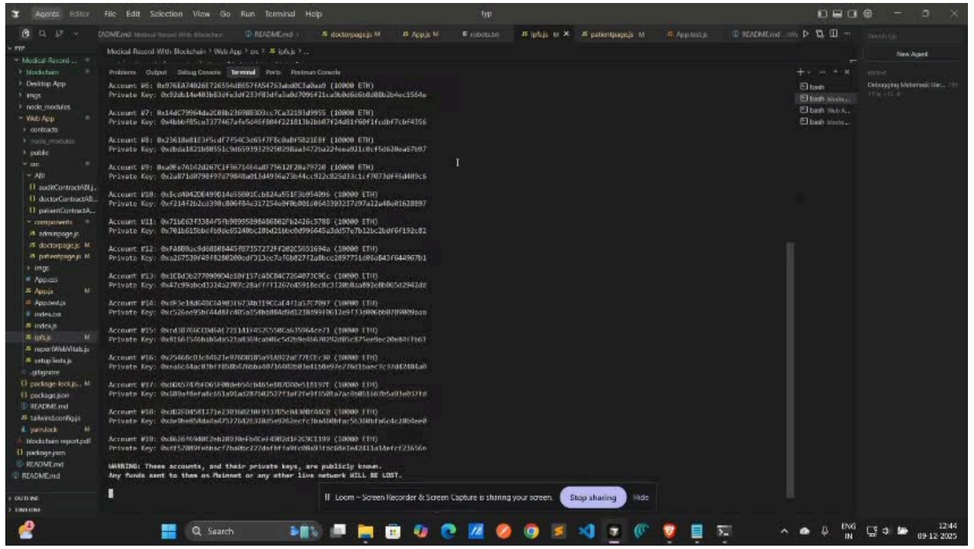


Fig 5.1: Demo video

- Smart Contract Deployment via Remix & Hardhat
  - Compiled and deployed via Remix for debugging
  - Automated deployment scripts using Hardhat for repeatability
  - Ensured correct network mapping for MetaMask and desktop client

5. Work Done(Cont.)

- IPFS Node Deployment
  - Pulled and executed Kubo IPFS Docker image
  - Configured persistent volumes
  - Set CORS headers for application usage
  - Implemented IPFS API connection in both apps
  - Tested upload/retrieval cycles and hash verification

- Web Application Development (React.js)
  - Designed patient–doctor–admin UI
  - Implemented MetaMask login
  - Added document upload + CID retrieval
  - Implemented smart-contract calls using ethers.js
  - Integrated verification workflow and notifications

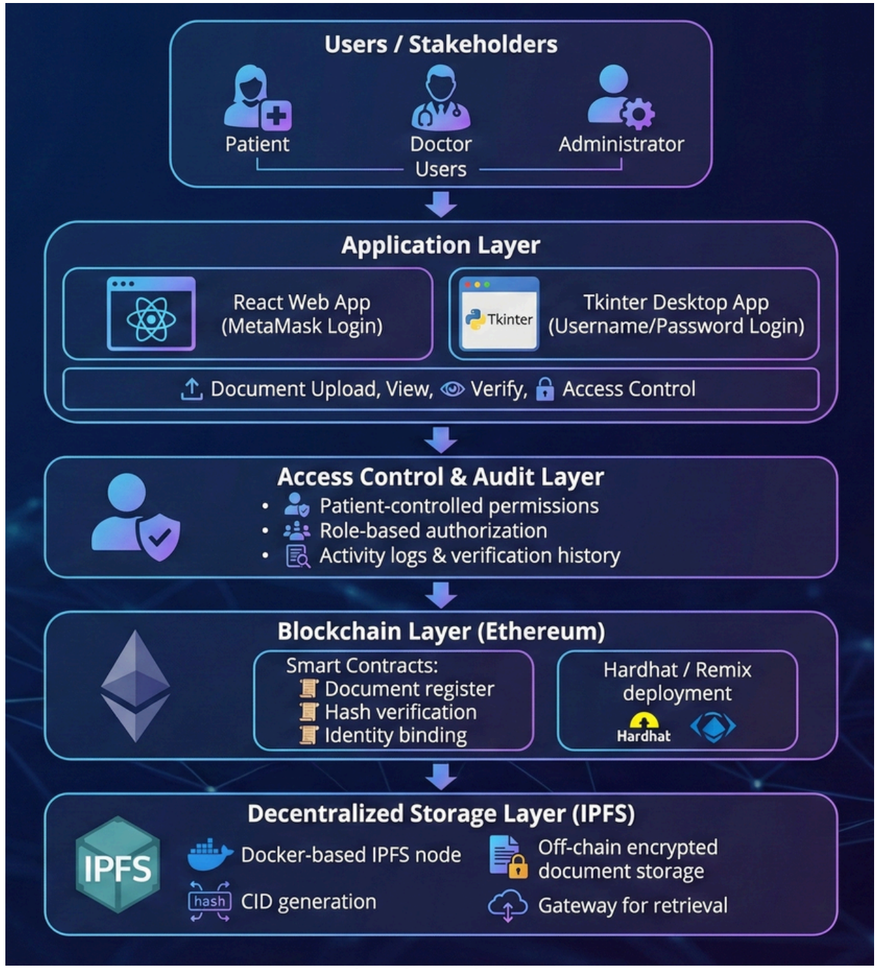


Fig 5.2: WorkFlow

5. Work Done(Cont.)

- System Design & Architecture

  We designed and implemented a Blockchain-Based Medical Document Management System using:

  - Ethereum Smart Contracts (Solidity 0.8.20)
  - IPFS (InterPlanetary File System) for off-chain storage
  - Hardhat for development and testing
  - React Web App + Python Desktop App
  - Three-Contract Architecture:
    - PatientManagement
    - DoctorManagement
    - HealthcareAudit

  Only the IPFS hash (CID) of medical documents is stored on-chain to reduce gas cost and improve scalability

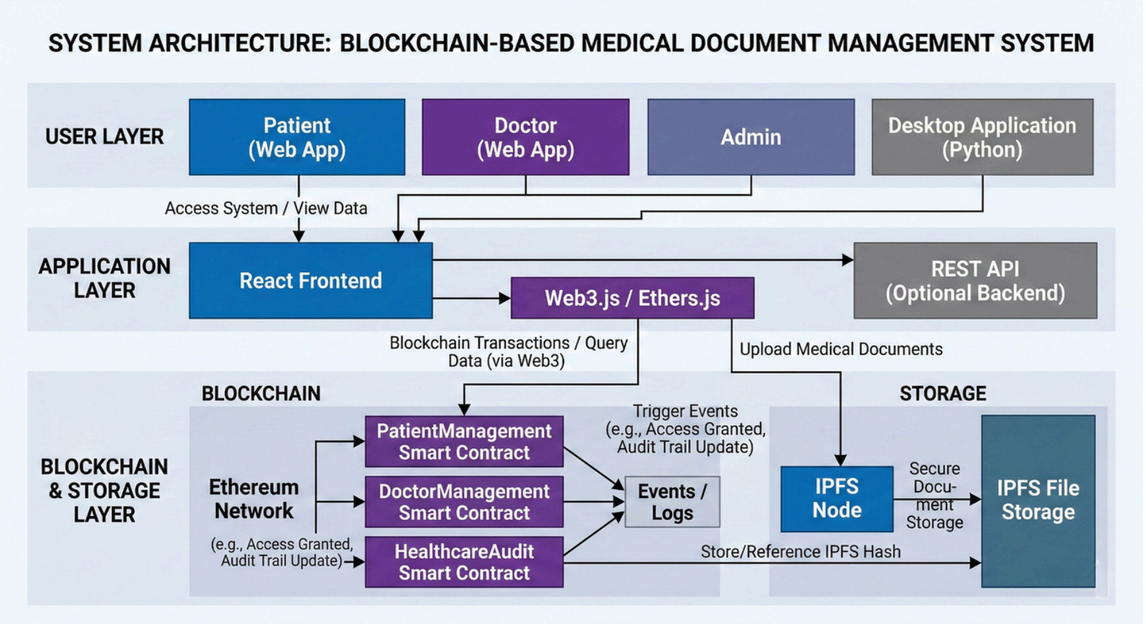


Fig 5.3: System Design

5. Work Done(Cont.)

- Smart Contract Development

  Implemented and deployed:

  ✅ PatientManagement
  - Register patients
  - Add medical records
  - Version control of records
  - Fetch active records

  ✅ DoctorManagement
  - Register doctors
  - Grant access to patient records
  - Revoke access (patient-only revocation logic)
  - Maintain authorized patient list

  ✅ HealthcareAudit
  - Immutable audit logging
  - Filter logs by actor or subject
  - Timestamp preservation

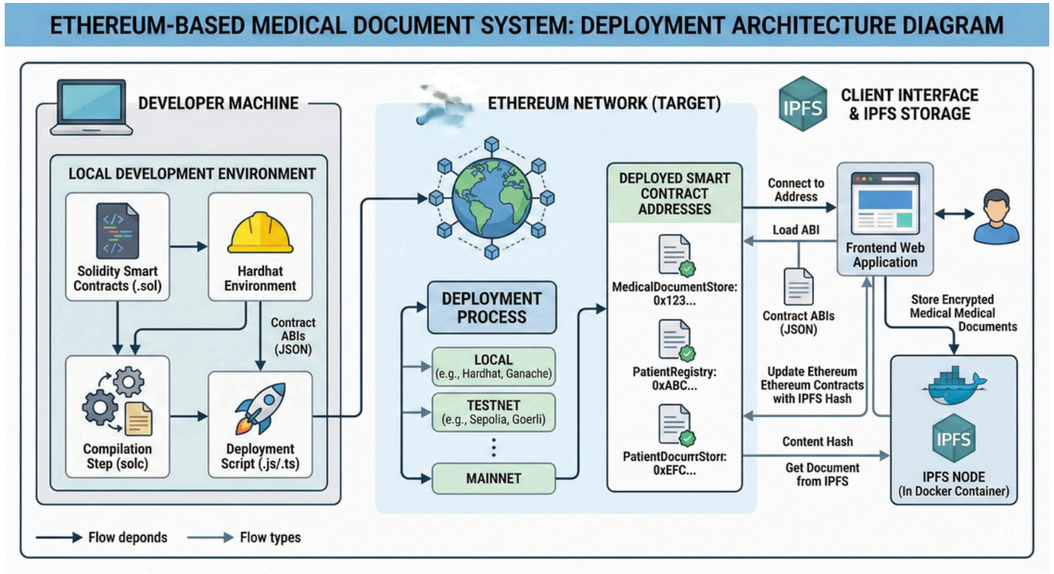


Fig 5.4: Smart Contract WorkFlow

5. Work Done(Cont.)

- Testing & Validation

We developed 92 automated test cases across:

- Integration tests
- Security tests
- Unit tests for each contract

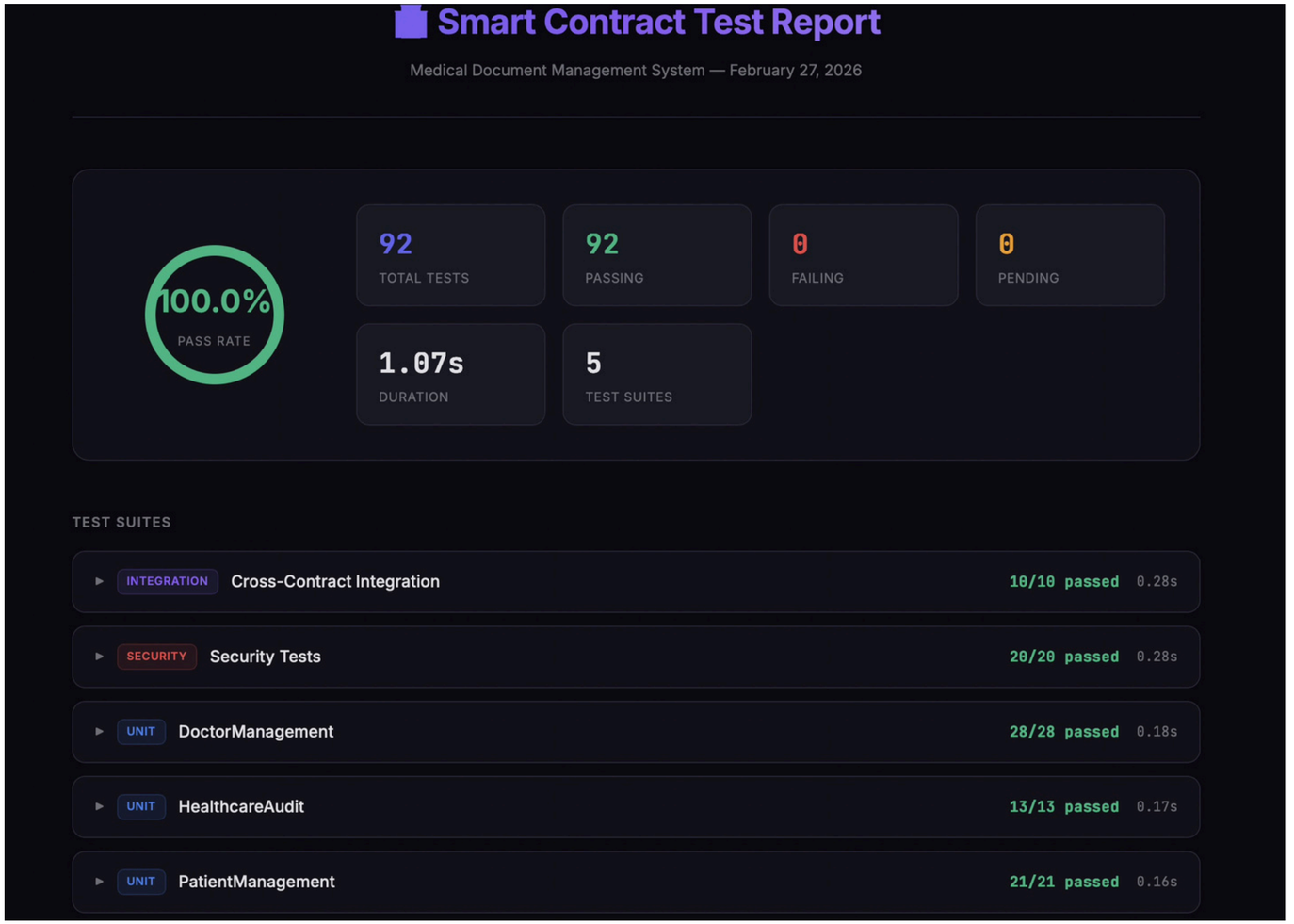


Fig 5.5: Smart Contract Test Report

5. Work Done(Cont.)

- Testing & Validation

📊 Test Report Summary:

- Total Tests: 92
- Passing: 92
- Failing: 0
- Pass Rate: 100%
- Execution Time: 1.07s
- Test Suites: 5

INTEGRATION Cross-Contract Integration — 10/10 passed 0.28s

#	TEST	STATUS	TIME
1	should complete the patient journey end-to-end	✅ Pass	35ms
2	should complete the doctor journey end-to-end	✅ Pass	10ms
3	should complete the grant-view-revoke flow	✅ Pass	23ms
4	should handle patient granting access to 3 doctors then revoking 1	✅ Pass	25ms
5	should correctly version documents: v1 → v2 → only v2 active	✅ Pass	24ms
6	should log full register → grant → revoke audit trail	✅ Pass	61ms
7	should allow querying audit trail by subject after mixed actions	✅ Pass	28ms
8	should complete register → grant → record → audit → revoke → records survive	✅ Pass	28ms
9	should handle revoking all doctors then re-granting one	✅ Pass	22ms
10	should handle complex selective revocations	✅ Pass	23ms

Fig 5.52: Cross Contract Test Report

SECURITY Security Tests — 20/20 passed 0.28s

#	TEST	STATUS	TIME
1	should prevent non-admin from registering a doctor	✅ Pass	26ms
2	should prevent non-admin from registering a patient	✅ Pass	4ms
3	should prevent doctor from registering other doctors	✅ Pass	6ms
4	should prevent patient from calling admin functions	✅ Pass	5ms
5	should prevent doctor from revoking their own patient's access	✅ Pass	3ms
6	should prevent stranger from revoking access	✅ Pass	3ms
7	should prevent admin from revoking on behalf of patient	✅ Pass	3ms
8	should handle empty string username and role	✅ Pass	10ms
9	should handle very long string input	✅ Pass	22ms
10	should handle special characters and unicode in strings	✅ Pass	9ms
11	should handle SQL injection attempts in strings (no-op on blockchain)	✅ Pass	6ms
12	should not allow modifying existing audit logs (append-only)	✅ Pass	19ms
13	should preserve audit log timestamps	✅ Pass	8ms
14	should preserve original record's IPFS hash after adding new records	✅ Pass	16ms
15	should resist reentrancy on revokePatientAccess	✅ Pass	21ms
16	should allow registering zero address as doctor (no guard in contract)	✅ Pass	9ms
17	should allow registering zero address as patient	✅ Pass	7ms
18	should allow same address to be registered as both doctor and patient	✅ Pass	17ms

Fig 5.51: Security Test Report

UNIT HealthcareAudit — 13/13 passed 0.17s

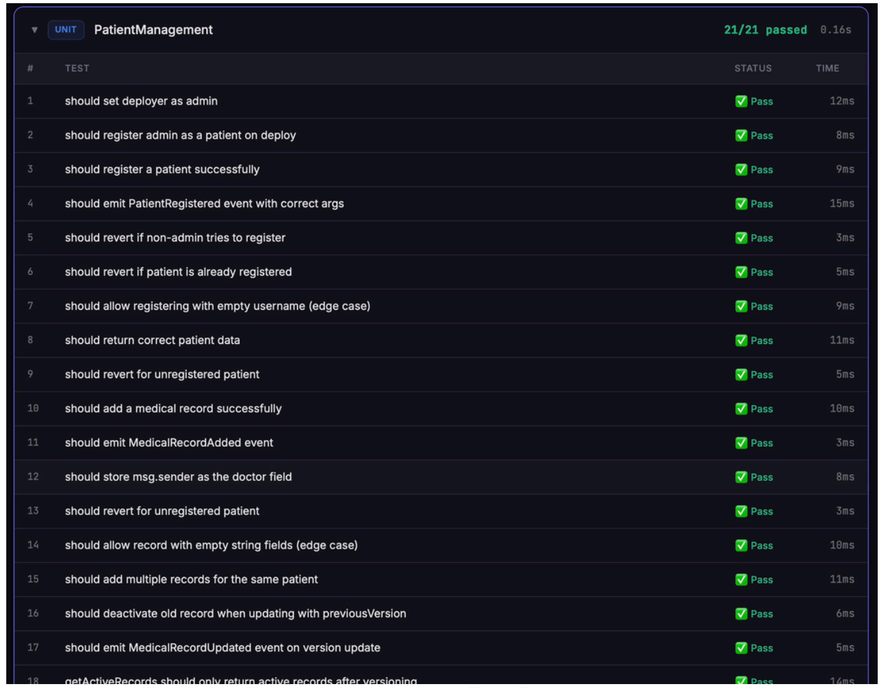
#	TEST	STATUS	TIME
1	should set deployer as admin	✅ Pass	5ms
2	should have correct constant action type strings	✅ Pass	13ms
3	should start with empty audit trail	✅ Pass	4ms
4	should add an audit log successfully	✅ Pass	29ms
5	should emit AuditLogAdded event with correct args	✅ Pass	3ms
6	should store block.timestamp in the record	✅ Pass	4ms
7	should allow anyone to add audit logs (no access restriction)	✅ Pass	8ms
8	should allow audit log with empty string fields (edge case)	✅ Pass	13ms
9	should return all logs in order	✅ Pass	28ms
10	should filter logs by actor correctly	✅ Pass	25ms
11	should return empty array for actor with no logs	✅ Pass	11ms
12	should filter logs by subject correctly	✅ Pass	24ms
13	should return empty array for subject with no logs	✅ Pass	6ms

Fig 5.53: HealthCare Test Report

5. Work Done(Cont.)

- Testing & Validation

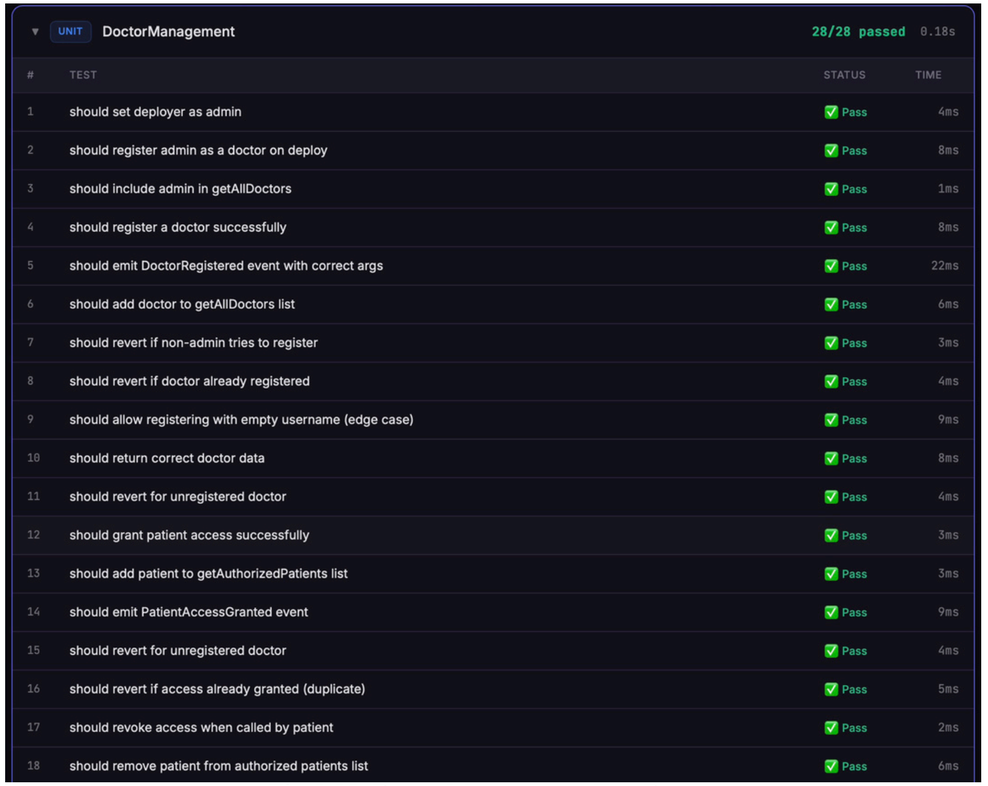
  All tests passed successfully test-report
  This validates:
  - Correct smart contract behavior
  - Security enforcement
  - Stable integration between contracts



UNIT PatientManagement 21/21 passed 0.16s

#	TEST	STATUS	TIME
1	should set deployer as admin	✅ Pass	12ms
2	should register admin as a patient on deploy	✅ Pass	8ms
3	should register a patient successfully	✅ Pass	9ms
4	should emit PatientRegistered event with correct args	✅ Pass	15ms
5	should revert if non-admin tries to register	✅ Pass	3ms
6	should revert if patient is already registered	✅ Pass	5ms
7	should allow registering with empty username (edge case)	✅ Pass	9ms
8	should return correct patient data	✅ Pass	11ms
9	should revert for unregistered patient	✅ Pass	5ms
10	should add a medical record successfully	✅ Pass	10ms
11	should emit MedicalRecordAdded event	✅ Pass	3ms
12	should store msg.sender as the doctor field	✅ Pass	8ms
13	should revert for unregistered patient	✅ Pass	3ms
14	should allow record with empty string fields (edge case)	✅ Pass	10ms
15	should add multiple records for the same patient	✅ Pass	11ms
16	should deactivate old record when updating with previousVersion	✅ Pass	6ms
17	should emit MedicalRecordUpdated event on version update	✅ Pass	5ms
18	getActiveRecords should only return active records after versioning	✅ Pass	14ms

Fig 5.54: PatientManagement Test Report



UNIT DoctorManagement 28/28 passed 0.18s

#	TEST	STATUS	TIME
1	should set deployer as admin	✅ Pass	4ms
2	should register admin as a doctor on deploy	✅ Pass	8ms
3	should include admin in getAllDoctors	✅ Pass	1ms
4	should register a doctor successfully	✅ Pass	8ms
5	should emit DoctorRegistered event with correct args	✅ Pass	22ms
6	should add doctor to getAllDoctors list	✅ Pass	6ms
7	should revert if non-admin tries to register	✅ Pass	3ms
8	should revert if doctor already registered	✅ Pass	4ms
9	should allow registering with empty username (edge case)	✅ Pass	9ms
10	should return correct doctor data	✅ Pass	8ms
11	should revert for unregistered doctor	✅ Pass	4ms
12	should grant patient access successfully	✅ Pass	3ms
13	should add patient to getAuthorizedPatients list	✅ Pass	3ms
14	should emit PatientAccessGranted event	✅ Pass	9ms
15	should revert for unregistered doctor	✅ Pass	4ms
16	should revert if access already granted (duplicate)	✅ Pass	5ms
17	should revoke access when called by patient	✅ Pass	2ms
18	should remove patient from authorized patients list	✅ Pass	6ms

Fig 5.55: DoctorManagement Test Report

5. Work Done(Cont.)

- Gas Cost Benchmarking

  We measured gas usage for each operation:

Operation	Avg Gas	Cost (@20 gwei)
addMedicalRecord	269,173	~$10.95
addAuditLog	195,488	~$7.95
registerDoctor	125,544	~$5.11
registerPatient	98,223	~$3.99
grantAccess	93,419	~$3.80
revokeAccess	34,960	~$1.42

Key observation:

- Gas cost remains constant even as number of documents increases

5. Work Done(Cont.)

- Security Implementation

  We enforced:
  - Role-based access control (Admin / Doctor / Patient)
  - Patient-only revocation logic
  - Prevention of duplicate registrations
  - Double-revoke protection
  - Event emission for audit tracking
  - Reentrancy resistance

- Gas Price Sensitivity Analysis

  We analyzed cost under different gas prices (5–200 gwei).
  Example:
  - addMedicalRecord at 5 gwei → ~$2.74
  - addMedicalRecord at 200 gwei → ~$109.47

  📌 Finding:
  Cost increases linearly with gas price (up to 40× difference).

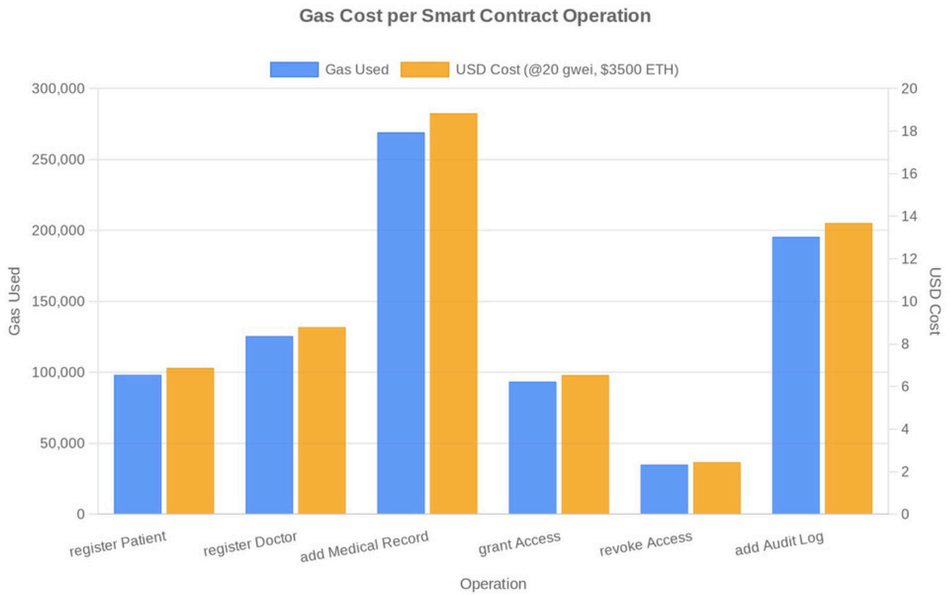


Fig 5.6 : Gas Cost per Smart Contract

5. Work Done(Cont.)

- IPFS Performance Testing

File Size	Upload	Retrieval	End-to-End
1KB	28 ms	3.5 ms	33 ms
100KB	23 ms	5.5 ms	29 ms
1MB	63 ms	34 ms	68 ms
5MB	118 ms	100 ms	124 ms

📌 Key Insight:
  - Blockchain cost is independent of file size.
  - Even 5MB files upload in under 125ms

- Scalability Testing

📌 Document Growth (10 → 1000 documents)
Gas increase: Only 0.038%
Conclusion:
Smart contract storage scales efficiently

5. Work Done(Cont.)

- Concurrent Users Testing

Users	Avg Response	Throughput
10	18 ms	452 tx/s
50	80 ms	628 tx/s
100	173 ms	559 tx/s

System handled concurrent requests efficiently.

- Comparative Cost Analysis

Compared with:

- Ethereum Mainnet
- Polygon L2
- Hyperledger Fabric

Key finding:

- Ethereum is expensive but offers strongest immutability.
- Polygon significantly reduces cost.
- Centralized is cheapest but lacks decentralization and trust guarantees

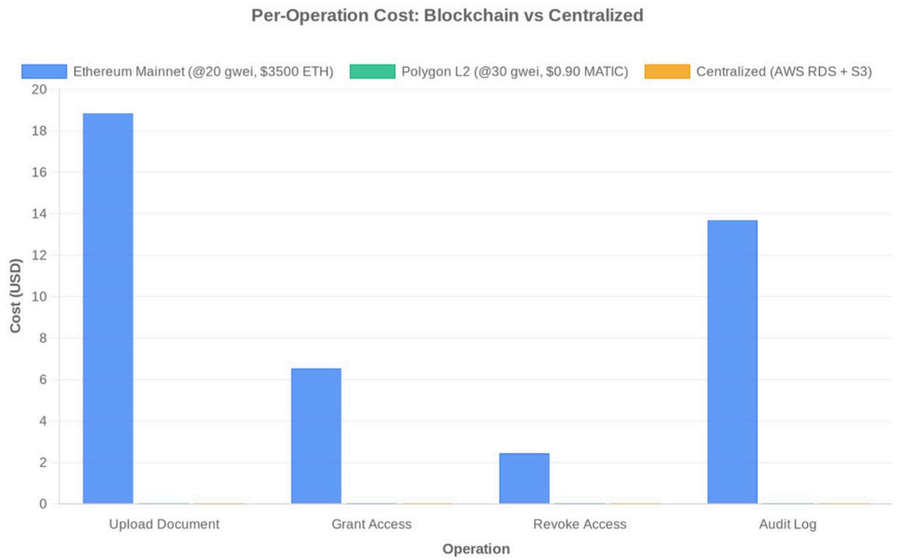


Fig 5.7: Per Operation Cost

5. Work Done(Cont.)

- Access Revocation Module Implementation

One of the most critical features implemented in this system is patient-controlled access revocation.

Objective

- Only the patient can revoke a doctor's access
- No admin, doctor, or third party can revoke on behalf of the patient
- The action is recorded immutably in the audit trail

Security Checks Implemented:

- Doctor must be registered
- Access must already exist
- msg.sender == patient (patient-only revocation)
- Prevent double revoke
- Emit PatientAccessRevoked event

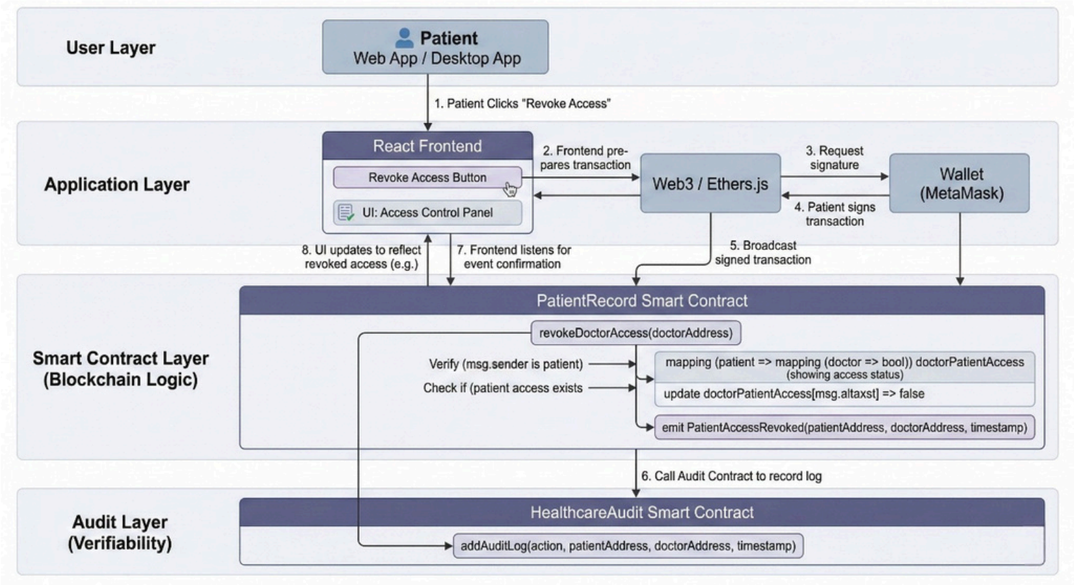


Fig 5.8: Revocation WorkFlow

5. Work Done(Cont.)

- Access Revocation Module Implementation (Cont.)

Revocation Flow

- Patient clicks Revoke Access
- Frontend connects to wallet
- Signed transaction sent to blockchain
- Smart contract:
  - Validates caller
  - Updates mapping
  - Removes patient from doctor's authorized list
  - Emits event
- Audit log entry created
- UI updates status

Security Guarantees Achieved

- Patient-centric data ownership
- No unauthorized revocation
- Immutable on-chain state update
- Audit trail logging
- Resistant to reentrancy
- Double-revoke protection

6. Results and Discussion

- **Functional Results**
  - Successfully implemented decentralized medical record system
  - Patients control access to their data
  - Doctors can access only authorized records
  - Immutable audit trail maintained
  - Versioning works correctly

- **Security Results**
  - 100% test pass rate
  - No unauthorized access allowed
  - Revocation works correctly
  - Audit logs cannot be modified

- **Performance Results**
  - IPFS provides sub-second storage performance
  - Gas cost remains predictable and constant
  - No degradation up to 1000 documents
  - Handles 100 concurrent users efficiently

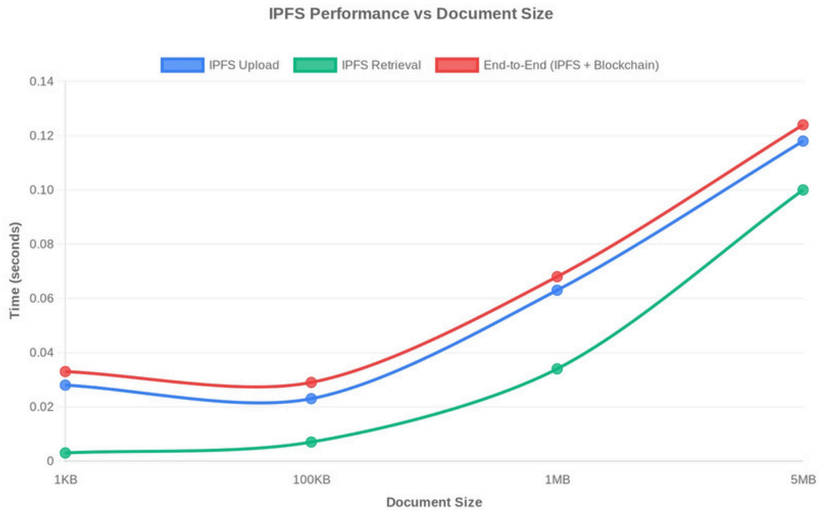


Fig 6.1: Ipfs Performance vs Document Size

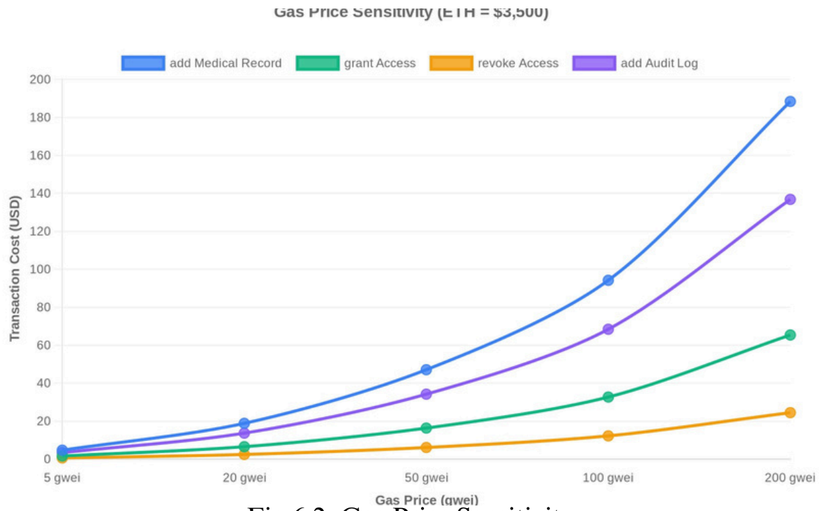


Fig 6.2: Gas Price Sensitivity

6. Results and Discussion

- **Economic Results**
  - Mainnet cost per medical upload ≈ $10–$11 (at 20 gwei)
  - Cost highly dependent on gas price
  - Layer 2 solutions recommended for production deployment

- **Final Outcome**

  The project successfully demonstrates that:
  - Blockchain can securely manage healthcare records
  - IPFS + on-chain hash is a scalable architecture
  - Smart contracts can enforce patient-centric access control
  - System is secure, tested, and performance-validated

  However:
  - Gas cost volatility remains a challenge
  - Layer 2 or hybrid architecture is ideal for real-world deployment

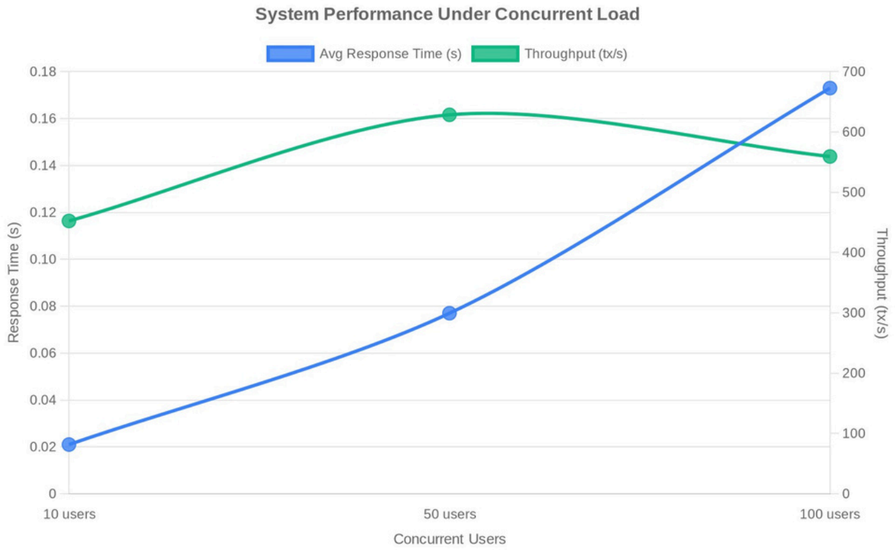


Fig 6.2: System Performance under Concurrent Load

7. CONCLUSION

- Designed and implemented a Blockchain-Based Medical Document Management System
- Developed three smart contracts: PatientManagement, DoctorManagement, HealthcareAudit
- Implemented patient-controlled access grant and revocation mechanism
- Integrated IPFS for off-chain medical document storage
- Stored only IPFS hash on-chain to reduce gas cost
- Achieved immutable audit logging for all critical actions
- Implemented medical record version control
- Completed 92 automated test cases with 100% pass rate test-report
- Verified predictable gas cost behavior through benchmarking gas-cost-analysis
- Confirmed scalability up to 1000 documents and 100 concurrent users
- Demonstrated secure, decentralized, patient-centric data ownership model

8. FUTURE WORK

- Deploy system on Polygon (Layer 2) to reduce transaction costs
- Compare Ethereum vs Polygon gas efficiency in real deployment
- Implement Zero-Knowledge Proofs for enhanced patient privacy
- Add encrypted IPFS storage with decentralized key management
- Integrate with hospital Electronic Health Record (EHR) systems
- Support multi-hospital and multi-admin architecture

9. Timeframe of the project:

S.N	Semester	Activities
1	8th Semester	Deploy smart contracts on Polygon (Layer 2)
2	8th Semester	Optimize gas cost and perform comparative analysis (Ethereum vs Polygon)

References

[1] C. C. Agbo, Q. H. Mahmoud, and J. M. Eklund, "Blockchain technology in healthcare: a systematic review," Healthcare, vol. 7, no. 2, p. 56, 2019. [Online]. Available: https://www.mdpi.com/2227-9032/7/2/56

[2] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "MedRec: Using blockchain for medical data access and permission management," in Proc. 2nd Int. Conf. Open Big Data (OBD), 2016, pp. 25–30. [Online]. Available: https://ieeexplore.ieee.org/document/7552024

[3] M. A. Cyran, "Blockchain as a foundation for sharing healthcare data," Blockchain Healthc. Today, 2018. [Online]. Available: https://www.researchgate.net/publication/323973053

[4] A. C. Ekblaw, "MedRec: blockchain for medical data access, permission management and trend analysis," Ph.D. dissertation, MIT, Cambridge, MA, USA, 2017. [Online]. Available: https://dspace.mit.edu/handle/1721.1/109658

[5] S. V. Kakade et al., "Blockchain-based medical record sharing in healthcare IoT: Building trust and transparency through secure provenance tracking," J. Electr. Syst., vol. 19, no. 3, 2023. [Online]. Available: https://journal.esrgroups.org/jes/article/view/650

[6] M. Mettler, "Blockchain technology in healthcare: The revolution starts here," in Proc. IEEE 18th Int. Conf. e-Health Netw., Appl. Serv. (Healthcom), 2016, pp. 1–3. [Online]. Available: https://ieeexplore.ieee.org/document/7749510

[7] P. Singh et al., "Blockchain-enabled verification of medical records using soulbound tokens and cloud computing," Sci. Rep., vol. 14, no. 1, p. 24830, 2024. [Online]. Available: https://www.nature.com/articles/s41598-024-52636-z

[8] P. S. Yoo, "Data communication networks—a comparative evaluation of the MIT and Harvard environments," Ph.D. dissertation, MIT, Cambridge, MA, USA, 1987. [Online]. Available: https://dspace.mit.edu/handle/1721.1/17913

[9] X. Yue, H. Wang, D. Jin, M. Li, and W. Jiang, "Healthcare data gateways: found healthcare intelligence on blockchain with novel privacy risk control," J. Med. Syst., vol. 40, no. 1 Available: https://pubmed.ncbi.nlm.nih.gov/27565509/