DLSU Research Congress 2025
De La Salle University, Manila, Philippines
June 26 to 28, 2025

FuseVault: An Application Programming Interface (API) for Enhanced Blockchain Data Storage Using InterPlanetary File System (IPFS) and MongoDB

Jordyn William Anton Hay[1], Jeanine Marie Ignacio[1], John Kieffer Recato Dy[1], Timothy Joshua Tan[1], and Katrina Ysabel Solomon[1,*]

[1] *Advanced Research Institute for Informatics, Computing, and Networking, De La Salle University*
**Corresponding Author: katrina.solomon@dlsu.edu.ph*

Abstract: In response to the growing demand for scalable, secure, and decentralized solutions, this research develops an Application Programming Interface (API) that addresses blockchain's on-chain data storage limitations through a structured approach integrating the InterPlanetary File System (IPFS) and MongoDB. The API combines these technologies to create an efficient solution for handling large-scale data storage while ensuring data integrity and maintaining blockchain's decentralization benefits. It focuses on establishing efficient data operations and metadata management capabilities, providing an evaluation of the framework's performance compared to traditional approaches. Blockchain technology ensures data integrity by immutably storing IPFS content hashes on a public ledger, while MongoDB enables efficient metadata management with advanced querying and indexing capabilities. This integration offers a robust, scalable solution for managing large datasets, combining the benefits of IPFS for decentralized storage with MongoDB for flexible metadata management and blockchain for data integrity.

Key Words: blockchain technology; decentralized data storage; metadata management; IPFS; MongoDB

1. INTRODUCTION

1.1 Overview

In recent years, blockchain technology has revolutionized data management through its decentralized, immutable, and transparent approach. However, as organizations increasingly adopt blockchain for diverse applications beyond cryptocurrencies, the inherent limitations of on-chain data storage have become apparent.

Blockchain technology, originally introduced by Satoshi Nakamoto for Bitcoin, has emerged as a revolutionary tool across industries due to its decentralized, secure, and transparent nature. One of its most significant contributions is the ability to eliminate the need for trusted third parties while ensuring data integrity through mechanisms like proof-of-work (Zheng, Xie, Dai, Chen, & Wang, 2017). While blockchain's early applications focused on cryptocurrency, its potential stretches far beyond digital currencies, offering solutions in the supply chain, finance, and real estate sectors (Yli-Huumo, Ko, Choi, Park, & Smolander, 2016).

Blockchain technology can significantly enhance data management processes by providing tamper-proof records and ensuring data integrity. However, blockchain's limitations, especially its inability to efficiently store large volumes of data directly on-chain, present significant challenges for

modern applications that handle extensive datasets. This limitation restricts the deployment of large-scale systems needed for data-intensive applications, particularly those requiring substantial storage capacity and frequent data access (Reyna, Martín, Chen, Soler, & Díaz, 2018).

The InterPlanetary File System (IPFS) offers a decentralized approach to storing large files off-chain while retaining the integrity of data by linking it to the blockchain through cryptographic hashes. This combination allows systems to leverage blockchain's security and transparency without the burden of costly on-chain storage. IPFS ensures that large datasets are securely stored in a decentralized manner while only a small, immutable reference (hash) is stored on-chain, preserving the framework's integrity (Benet, 2014).

While IPFS solves the problem of storage, another challenge arises in efficiently managing metadata and facilitating dynamic querying for large-scale data operations. IPFS alone does not offer the flexibility required to handle frequent updates or advanced queries (Trautwein, et al., 2022). To address this, this research integrates MongoDB—a NoSQL database that allows for flexible, efficient metadata management. MongoDB complements IPFS by providing robust querying, indexing, and real-time update capabilities, enabling the framework to handle complex metadata retrieval and data management.

1.2 Current State of Technology

Blockchain technology faces significant storage constraints that impact its efficiency. The gas fees (GWEI) required for processing transactions scale with data size, making large-scale storage economically unsustainable (Buterin, 2014; Gervais, Karame, Capkun, & Capkun, 2014). Typically, blockchains only store essential data on-chain with size restricted by block limits and gas expenditure caps (Wood, 2014). For example, Ethereum's block gas limits constrain operations complexity, making it impractical to store large files directly on-chain. To address this, systems integrate solutions like IPFS to store data off-chain while maintaining only verification hashes on the blockchain (Benet, 2014).

Blockchain excels at immutability but struggles with large datasets. (Wüst & Gervais, 2018) note that blockchains aren't designed for extensive data storage due to cost, performance, and efficiency concerns. IPFS offers a solution through content-addressed decentralized file storage (Benet, 2014), generating unique cryptographic hashes that can be referenced on-chain. Projects like Filecoin (Protocol Labs, 2017) leverage this approach to offer decentralized storage while maintaining blockchain verification. Zyskind, Nathan, & Pentland (2015) demonstrated that storing file hashes on-chain ensures data integrity verification. Zheng, Li, Chen, & Dong (2018) discuss encryption for sensitive data before IPFS upload, with only encrypted file hashes stored on-chain. While IPFS excels at decentralized storage, it lacks advanced querying capabilities, which can be addressed by integrating databases like MongoDB (Bieri, 2021).

Doan, Psaras, Ott, & Bajpai (2022) examine how combining IPFS with traditional storage enhances performance, availability, and scalability. This hybrid approach uses IPFS for decentralized, immutable storage while traditional systems provide low-latency access for frequently queried data. A middleware layer determines optimal storage location based on access patterns, cost, and privacy considerations. Bieri (2021) demonstrates how systems often store metadata in databases like MongoDB while blockchain secures integrity through hash storage, enabling flexible collection management where metadata can be updated without altering underlying data.

2. METHODOLOGY

2.1 System Architecture Overview

FuseVault employs a hybrid storage approach that combines three distinct storage technologies:

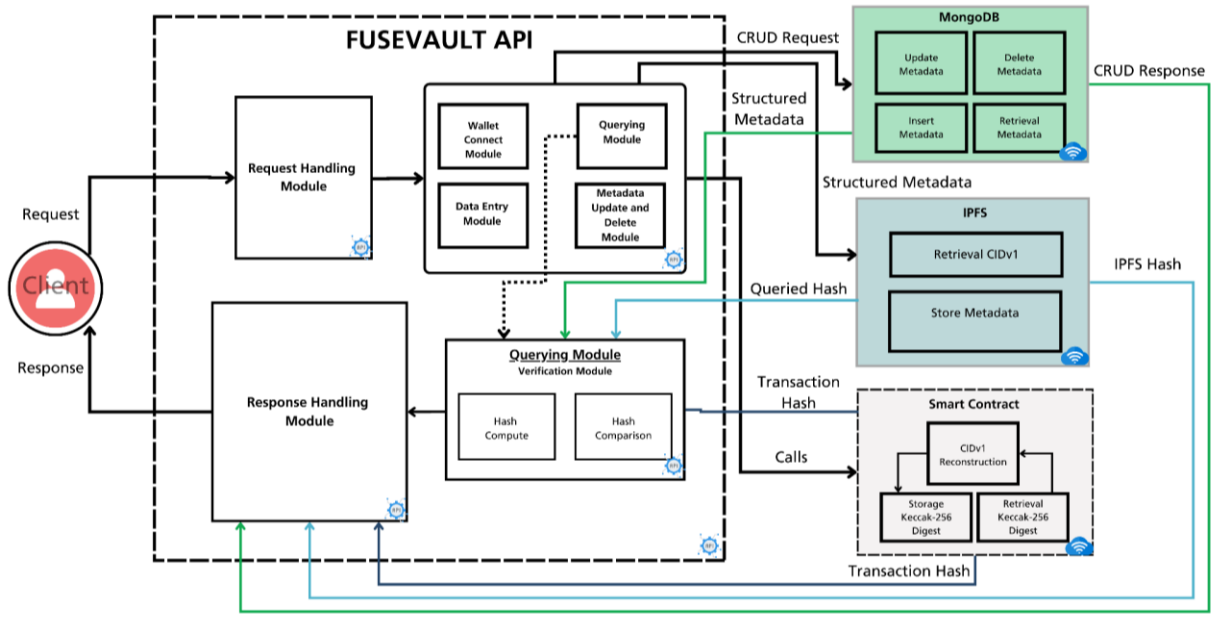

Fig. 1. FuseVault System Architecture

- **Blockchain Storage**: Using the Ethereum Sepolia Testnet with a custom smart contract (IPFSVersionRegistry) to store cryptographic hashes of critical metadata
- **Decentralized Storage**: Utilizing IPFS via Web3.Storage to store the actual metadata files
- **Database Storage**: Employing MongoDB to maintain relationships, enable efficient queries, and track version history

This hybrid approach allows FuseVault to balance security, decentralization, and efficiency. The backend is built with FastAPI, a modern Python web framework that enables high-performance API development.

2.2 The IPFSVersionRegistry Smart Contract

At the core of FuseVault's blockchain integration is the IPFSVersionRegistry smart contract. This contract has been optimized for gas efficiency and provides several key functions:

- Storing and retrieving cryptographic hashes of IPFS content identifiers (CIDs)
- Tracking version history of assets' critical metadata
- Managing asset ownership and transfers
- Providing role-based access control through admin and delegate systems
- Verifying content integrity through hash comparison

The contract uses a custom AssetIPFS data structure that efficiently packs data into storage slots, reducing gas costs for blockchain interactions. Each asset is identified by a unique asset ID and is linked to an owner's wallet address. The contract maintains versioning information that allows verification of the latest valid metadata for each asset.

The smart contract was deployed on the Ethereum Sepolia test network using Hardhat. This allows for testing the contract without having to spend real ETH on gas costs.

2.3 Data Flow Process

When a user uploads metadata to FuseVault, the following process occurs:

1. **Authentication**: The user authenticates with their wallet address, and the system verifies they have appropriate permissions.
2. **Metadata Processing**: The UploadHandler receives the metadata and determines whether this is a new asset or an update to an existing one.
3. **IPFS Storage**: Critical metadata is formatted and sent to the Web3.Storage microservice, which uploads it to IPFS and returns a unique CID.
4. **Blockchain Registration**: The CID's hash is stored on the blockchain using the IPFSVersionRegistry contract. Depending on who is performing the action, either the updateIPFS or updateIPFSFor function is called.
5. **Database Storage**: The complete metadata (both critical and non-critical), along with reference information like the IPFS CID and blockchain transaction hash, is stored in MongoDB.
6. **Transaction Recording**: The action is logged in the transaction history, maintaining a complete audit trail.

The system handles both new assets and updates to existing ones. For updates, FuseVault intelligently determines whether critical metadata has changed by comparing computed CIDs. If only non-critical metadata has changed, it skips the blockchain transaction, saving on gas costs while still maintaining data integrity.

2.4 Versioning System

FuseVault implements a comprehensive versioning system that tracks changes to assets over time:

- MongoDB Versioning: Every change to an asset creates a new version in MongoDB, with previous versions maintained for historical reference.
- Blockchain Versioning: The IPFSVersionRegistry contract tracks IPFS versions separately, incrementing only when critical metadata changes.
- Version Verification: When retrieving assets, FuseVault verifies that the version referenced in MongoDB matches the blockchain record, preventing rollback attacks.

This ensures that users can track changes to their assets while maintaining strong guarantees about the authenticity of the data.

2.5 Asset Transfers and Ownership

FuseVault was designed to support secure transfer of asset ownership between wallet addresses through a two-step process:

1. The current owner initiates a transfer by calling the `initiateTransfer` function, specifying the asset ID and the new owner's address.
2. The recipient must accept the transfer by calling the `acceptTransfer` function, confirming their willingness to take ownership.

This two-step process prevents unwanted or malicious transfers. During the transfer, the original asset is marked as deleted for the previous owner, and a copy is created under the new owner's address, maintaining the complete version history.

2.6 Role-Based Access Control

FuseVault implements a flexible permission system with three levels:

1. **Owners**: Users who own assets and have full control over them
2. **Admins**: Users with system-wide privileges who can perform actions on behalf of any user
3. **Delegates**: Users who are authorized by specific owners to act on their behalf

This structure allows organizations to implement workflows where certain team members can manage assets without transferring ownership. The blockchain contract enforces these permissions, ensuring that only authorized users can modify assets.

2.7 Soft Deletion and Recovery

Rather than permanently removing assets, FuseVault implements a soft deletion system:

- When an asset is deleted, it's marked as deleted in both MongoDB and on the blockchain.

- The asset remains accessible for historical purposes but is excluded from normal queries.
- Owners can "undelete" assets by creating a new version, which resets the deletion flag.

This approach prevents accidental data loss while maintaining a clean user experience.

2.8 Web3.Storage Integration

FuseVault uses a separate microservice written in JavaScript to manage interactions with Web3.Storage. Web3.Storage provides a developer-friendly API to manage IPFS files and ensure their persistence using Filecoin. A separate microservice was made because the Web3.Storage library does not support FastAPI. This microservice:

- Accepts file uploads from the main API
- Handles authentication with Web3.Storage
- Uploads content to IPFS and returns the resulting CIDs
- Provides methods to retrieve content from IPFS when needed

This separation of concerns allows the main API to focus on application logic while the microservice manages the complexities of IPFS interaction.

2.9 Technical Challenges and Solutions

Several technical challenges were addressed in FuseVault's implementation:

- **Gas Optimization**: The IPFSVersionRegistry contract uses efficient storage patterns and minimal on-chain data to reduce gas costs.
- **CID Verification**: FuseVault implements client-side CID computation to verify data integrity without requiring additional blockchain transactions.
- **Version Conflict Resolution**: The system includes logic to handle version conflicts in the case that the MongoDB database has been tampered.
- **Blockchain Transaction Management**: FuseVault handles blockchain transaction failures gracefully, with retry mechanisms and fallback options.

2.10 FuseVault as a Solution

FuseVault demonstrates how a hybrid storage approach can leverage the strengths of blockchain, IPFS, and traditional databases while minimizing their individual limitations. The IPFSVersionRegistry contract provides a secure, efficient foundation for tracking content integrity, while the surrounding system makes this technology accessible and practical for real-world applications.

By splitting data between on-chain references and off-chain storage, FuseVault achieves significant cost savings and storage flexibility compared to fully on-chain solutions while maintaining strong security guarantees through cryptographic verification. The comprehensive versioning system and ownership model make it suitable for managing valuable digital assets in a decentralized context.

The result is a storage solution that combines the best aspects of blockchain security, IPFS decentralization, and traditional database performance – providing a viable option for organizations that need secure, tamper-evident, flexible storage without the limitations of pure blockchain solutions.

3. RESULTS AND DISCUSSION

The evaluation of the FuseVault framework assessed its effectiveness in addressing blockchain storage limitations through IPFS and MongoDB integration. The researchers developed test scripts to measure performance across several key aspects: storage efficiency, query performance, data integrity verification, and concurrent operations. These tests provide quantitative insights into the framework's capabilities and trade-offs, particularly between data integrity guarantees and operational efficiency. The following sections present detailed metrics from the testing of the implemented framework.

3.1 Storage Performance Test

Table 1. Storage Performance Test for Ten (10) Assets

Metric	Result (seconds)
Total time	474.97
Average time per asset	36.54

These measurements seen in Table 1 encompass the entire pipeline from data preparation through blockchain confirmation. The reliance on Ethereum Sepolia Testnet and Web3 library introduces variability due to external network factors beyond experimental control.

Analysis of these results indicates that the multi-component storage process incurs significant time costs, predominantly from blockchain transaction confirmation times. The per-asset processing time of 36.54 seconds may present usability challenges for applications requiring near real-time data storage confirmation. This latency represents an inherent characteristic of decentralized storage approaches rather than an implementation inefficiency, as the consensus mechanisms in blockchain networks introduce necessary delays to maintain integrity guarantees.

3.2 Concurrency Handling Test

Table 2. Concurrency Test Results for 12 Requests

Metric	Result
Successful Requests	12/12 (100.00%)
Failed Requests	0/12 (0.00%)
Total Test Duration	382.22 seconds
Average Time Per Request	86.87 seconds
Max Concurrent Requests	3
Throughput Utilization	0.03 requests/second

As seen in Table 2, the throughput measurement of 0.03 requests per second represents a significant performance constraint for high-volume applications. This limitation stems directly from blockchain transaction finality requirements and is consistent with expected behavior for systems requiring on-chain confirmation. The perfect success rate (100%) demonstrates system stability under concurrent load, though the total operation time increases substantially compared to sequential processing. This suggests that while the system handles concurrency correctly, it does not achieve performance improvements through parallel execution due to the blockchain bottleneck.

3.3 Integrity Tests

Table 3. Integrity Test for Five (5) Tampering Instance Detection

Tampering Type	Result
`modify_critical`	Detected
`delete_critical_field`	Detected
`add_critical_field`	Detected
`modify_blockchain_ref`	Detected
`fake_deletion`	Detected

Table 4. Tampering Detection Performance

Metric	Result (seconds)
Average Detection Time	3.34
Shortest Detection Time	2.96
Longest Detection Time	3.64

Table 5. Recovery Performance for Five (5) Recovery Attempts

Metric	Result (seconds)
Average Recovery Time	8.67
Shortest Recovery Time	6.86
Longest Recovery Time	10.21

Analysis of these results shown in Tables 3-5 demonstrates the framework's effectiveness in identifying data tampering across diverse attack vectors. The relatively consistent detection times (standard deviation < 0.4 seconds) indicate a stable verification process regardless of tampering type. However, the recovery process requires significantly more time than detection alone, with an overhead of 159.70%. This increased latency during recovery operations represents the computational cost of retrieving authentic data from IPFS and reconstructing metadata integrity. These findings suggest that while automatic recovery is technically feasible, applications with strict response time requirements may need to implement additional measures to handle the recovery latency.

3.4 Query Performance Comparison

Table 6. Query Speed Test for Five (5) Queries

		MongoDB Queries Only	MongoDB + IPFS + Blockchain Queries
Time (seconds)	Average	0.07	3.86
	Minimum	0.06	3.15
	Maximum	0.08	5.40
	Median	0.07	3.56
	Standard Deviation	0.01	0.88

The direct MongoDB queries executed approximately 54.5 times faster than the fully verified queries. The ratio between MongoDB-only and fully verified query times varied across test cases (38.51% to 80.97%).

Analysis of these results highlights the fundamental performance trade-off inherent in the system architecture. The substantial difference in query performance (MongoDB being 54.5 times faster) quantifies the computational cost of cryptographic verification. This performance differential necessitates strategic API design decisions regarding when to employ full verification versus when to use direct database access. The high standard deviation in verified query times (0.88 seconds) compared to MongoDB-only queries (0.01 seconds) indicates that verification overhead is subject to greater variability, likely due to external factors such as blockchain and IPFS network conditions. These findings suggest that application-level caching strategies and selective verification approaches would be necessary for applications with strict performance requirements.

4. CONCLUSIONS

This research has presented FuseVault, a framework that addresses the limitations of on-chain data storage by integrating blockchain technology with IPFS and MongoDB. The preliminary findings demonstrate that this hybrid approach successfully balances the need for data integrity, decentralization, and efficient data management.

The implementation of the IPFSVersionRegistry smart contract on the Ethereum Sepolia Testnet provides a secure foundation for tracking content integrity through cryptographic hashes, while IPFS delivers decentralized storage capabilities, and MongoDB enables efficient metadata management and querying. This integration creates a robust solution that preserves the benefits of blockchain security while overcoming its inherent storage constraints.

Performance evaluation reveals the expected trade-offs inherent in this architecture. While blockchain transaction latency introduces significant processing time (averaging 36.5 seconds per asset), this represents a necessary compromise to maintain the integrity guarantees that blockchain technology provides. The substantial performance differential between verified and non-verified queries (with direct MongoDB queries executing approximately 54.5 times faster) quantifies this trade-off and highlights the importance of strategic API design decisions for different use cases.

The framework's integrity verification mechanisms have proven effective, demonstrating 100% success in detecting various types of data tampering across all test cases. This confirms that the hybrid storage model maintains the security benefits of blockchain technology while expanding practical storage capabilities. The consistency in detection times (standard deviation < 0.4 seconds) indicates a reliable verification process regardless of tampering method.

FuseVault successfully implements several key functionalities, including secure metadata storage, comprehensive versioning, integrity verification, and graceful recovery from tampering attempts. The role-based access control system and ownership transfer mechanisms designed at the smart contract level provide a foundation for secure, decentralized asset management, though implementation at the API level remains to be completed.

While the current implementation serves as a successful proof of concept, performance characteristics reveal areas for optimization, particularly in handling concurrent operations where throughput (0.03 requests per second) presents limitations for high-volume applications. These findings establish baseline metrics against which future optimizations can be measured.

As blockchain adoption continues to grow

across industries, solutions like FuseVault that address core technical limitations while maintaining security guarantees will play a crucial role in enabling practical, large-scale implementations. This research contributes to the understanding of hybrid storage approaches and provides a structured framework that organizations can adapt to their specific data management needs when implementing blockchain-based systems.

5. REFERENCES

Benet, J. (2014). IPFS - Content Addressed, Versioned, P2P File System. *arXiv preprint arXiv:1407.3561*.

Bieri, C. (2021). An overview into the InterPlanetary File System (IPFS): use cases, advantages, and drawbacks. *Communication Systems XIV 28*.

Buterin, V. (2014). Ethereum white paper: a next generation smart contract & decentralized application platform.

Doan, T. V., Psaras, Y., Ott, J., & Bajpai, V. (2022). Toward Decentralized Cloud Storage With IPFS: Opportunities, Challenges, and Future Considerations. *IEEE Internet Computing*, 7-15.

Gervais, A., Karame, G. O., Capkun, V., & Capkun, S. (2014). Is Bitcoin a Decentralized Currency? *IEEE Security & Privacy*, 54-60.

Protocol Labs. (2017, July 19). *Filecoin: A decentralized storage network.* Retrieved from Protocol Labs Research: https://research.protocol.ai/publications/filecoin-a-decentralized-storage-network/protocollabs2017a.pdf

Reyna, A., Martín, C., Chen, J., Soler, E., & Díaz, M. (2018). On blockchain and its integration with IoT. Challenges and opportunities. *Future Generation Computer Systems*, 173-190.

Trautwein, D., Raman, A., Tyson, G., Castro, I., Scott, W., Schubotz, M., . . . Psaras, Y. (2022). Design and evaluation of IPFS: a storage layer for the decentralized web. *Proceedings of the ACM SIGCOMM 2022 Conference* (pp. 739-752). Amsterdam: ACM.

Wood, G. (2014). Ethereum: A Secure Decentralised Generalised Transaction Ledger. *Ethereum Project Yellow Paper*.

Wüst, K., & Gervais, A. (2018). Do you Need a Blockchain? *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)* (pp. 45-54). Zug: IEEE.

Yli-Huumo, J., Ko, D., Choi, S., Park, S., & Smolander, K. (2016). Where Is Current Research on Blockchain Technology?—A Systematic Review. *PLoS ONE*.

Zheng, Q., Li, Y., Chen, P., & Dong, X. (2018). An Innovative IPFS-Based Storage Model for Blockchain. *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)* (pp. 704-708). Santiago: IEEE.

Zheng, Z., Xie, S., Dai, H., Chen, X., & Wang, H. (2017). An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. *2017 IEEE International Congress on Big Data (BigData Congress)* (pp. 557-564). Honolulu: IEEE.

Zyskind, G., Nathan, O., & Pentland, A. (2015). Decentralizing Privacy: Using Blockchain to Protect Personal Data. *2015 IEEE Security and Privacy Workshops* (pp. 180-184). San Jose: IEEE.