INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD)
© 2026 IJNRD | Volume 11, Issue 7, July 2026 | ISSN: 2456-4184 | IJNRD.ORG

A Blockchain-Based Hybrid Framework for Transparent State Government Fund Allocation, Hierarchical Approval, and Closed-Loop Utilization Verification

[1] Vrushali R. Bhivase, [2] Dr.Jaydeep B.Patil

[1]MTech Student, [2]Professor,
[1] Computer Science and Engineering Department,
[1] DYP Agriculture and Technical University,
Talsande Kolhapur, Maharashtra, India

Abstract : Public Financial Management at the state level in India suffers from systemic corruption, fund diversion, and accountability failures, with the Comptroller and Auditor General's FY 2022–23 report flagging INR 3.17 lakh crore as idle, misallocated, or lapsed a crisis rooted not in resource scarcity, but in the absence of enforceable transparency. Existing blockchain-based governance solutions address only linear fund disbursement, failing to encode multi-tier approval hierarchies, verify actual fund utilization, or provide end-to-end public auditability. This paper proposes the State Government Fund Allocation and Tracking System (SGFATS), a decentralized application enforcing the complete fund lifecycle from legislative sanction through Finance scrutiny, Governor approval, contractor disbursement, and verified utilization closure via tamper-proof smart contracts deployed on a live four-node Hyperledger Besu network configured with IBFT 2.0 consensus. Unlike prior research relying on off-chain simulations, SGFATS permanently commits critical transactional records to an immutable ledger while storing evidentiary documents off-chain under AES-256 encryption, anchored via cryptographic hashing. The central contribution is an algorithmic fund utilization verification mechanism that prevents project closure until contractor-uploaded expense proofs reconcile exactly with the allocated budget, structurally deterring last-mile fund diversion through immutable, cryptographically verifiable audit trails. A formal threat model validates the architecture's cryptographic resilience, demonstrating successful mitigation against privilege escalation, off-chain data tampering, and man-in-the-middle interception through EVM-level RBAC and AES-256 pre-transit encryption. Ultimately, SGFATS shifts governance accountability from human trust to cryptographic certainty, providing a deployable blueprint for transparent, corruption-resistant public fund management.

IndexTerms **- State Government Fund Allocation and Tracking System (SGFATS), Public Financial Management (PMF), Hyperledger Besu, Smart Contract Escrow, Hybrid Blockchain Storage.**

I. INTRODUCTION

Public Financial Management (PFM) dictates the efficacy of state governance, welfare distribution, and infrastructure procurement. In India, despite the transition to digital governance, PFM remains plagued by systemic opacity, fund diversion, and bureaucratic delays. As Saxena and Kaur (2023) note, digitalization has improved operational velocity but remains constrained by isolated data silos and the absence of mathematically verifiable audit trails [21].

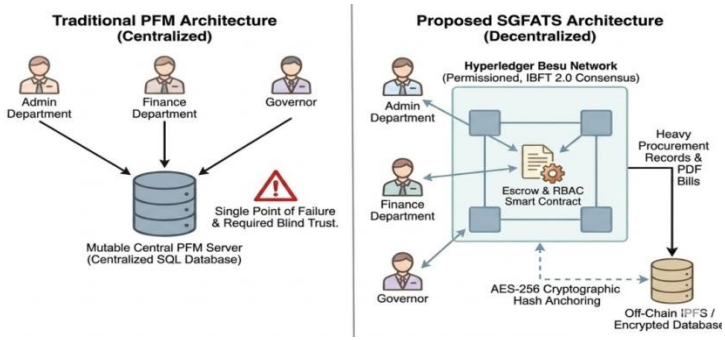

Fig.1 Architectural contrast:Traditional centralized PFM architectures (left) versus the proposed SGFATS decentralized escrow and hybrid storage model (right).

This inefficiency is severe: the Comptroller and Auditor General's (CAG) FY 2022–23 report flagged INR 3.17 lakh crore across government accounts as idle, misallocated, or lapsed [32]. This crisis stems not from resource scarcity, but from centralized, mutable database architectures that rely entirely on human trust rather than protocol-level enforcement. While Distributed Ledger Technology (DLT) can mitigate financial fraud [14, 28], traditional public blockchains fail to meet the privacy and scalability requirements of state operations [29]. Furthermore, existing solutions focus merely on linear fund disbursement, failing to natively replicate complex administrative hierarchies or verify actual infrastructural fund utilization.

To address these architectural gaps, this paper proposes the State Government Fund Allocation and Tracking System (SGFATS), as illustrated in Fig. 1. Engineered on a live, permissioned Hyperledger Besu network utilizing Istanbul Byzantine Fault Tolerance

(IBFT 2.0) consensus, SGFATS shifts administrative accountability from human oversight to cryptographic certainty. EVM-compatible smart contracts enforce a strict Role-Based Access Control (RBAC) hierarchy, ensuring public funds cannot be moved without cryptographically signed, multi-departmental consensus.

To circumvent the prohibitive computational costs of on-chain data storage [29], SGFATS employs a secure hybrid architecture. Transaction state changes are permanently recorded on the immutable ledger, while heavy evidentiary documents are stored off-chain under AES-256 encryption. These documents are anchored to the blockchain via cryptographic hashes, guaranteeing data integrity without inducing state bloat.

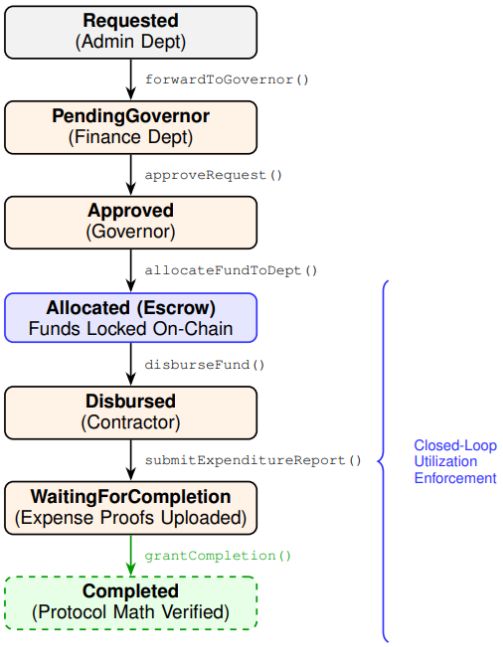

Fig. 2. The SGFATS Hierarchical State Machine. Funds remain cryptographically locked until the grantCompletion() protocol-level utilization verification is mathematically satisfied

The central contribution of this research is a novel, algorithmic fund utilization verification mechanism. The SGFATS hierarchical state machine, detailed in Fig. 2, operates as an immutable escrow vault that mathematically prevents the administrative closure of a project until the cumulative sum of contractor-uploaded, verified expense proofs reconciles exactly with the allocated budget. This protocol-level enforcement structurally deters last-mile fund diversion.

While early conceptual models of a State Government Fund Allocation and Tracking System (SGFATS) have been recently proposed [11], they lack the strict algorithmic enforcement and secure data architectures necessary for enterprise-level deployment. Building upon these foundational theories, the primary contributions of this paper are fivefold:

- The design and live-network deployment of a permissioned blockchain framework that cryptographically enforces the state government's multi-tier financial approval hierarchy through RBAC-enabled smart contracts on Hyperledger Besu with IBFT 2.0 consensus.
- The implementation of a mathematically enforced, closed-loop utilization audit that directly targets and eliminates end-point fund diversion.
- The integration of a gas-optimized hybrid storage model employing AES-256 encryption and SHA-256 cryptographic hashing to handle heavy procurement documentation securely.
- A formal threat model identifying four critical attack vectors privilege escalation, off-chain data tampering, fraudulent proof submission, and man-in-the-middle interception with corresponding cryptographic mitigation strategies validated against the deployed contract architecture.
- A comprehensive security analysis demonstrating the system's strong cryptographic resistance to unauthorized role escalation, arbitrary state manipulation, and post-submission evidentiary tampering.

1.1. Problem Statement

Effective Public Financial Management (PFM) is critical for state infrastructure and welfare development. However, current PFM architectures suffer from systemic opacity, multi-layered bureaucratic friction, and severe vulnerability to fund diversion. As evidenced by the CAG's FY 2022–23 report, INR 3.17 lakh crore across Indian government accounts was flagged as misallocated, idle, or lapsed [32]. This financial crisis is fundamentally an architectural failure. Current e-governance systems rely on centralized, mutable databases that depend entirely on the integrity of human administrators, leaving ledgers susceptible to ex-post facto alteration. Furthermore, while existing blockchain deployments address basic linear fund disbursement, they fail to natively encode complex multi-tier government hierarchies. Most critically, existing systems lack a closed-loop verification mechanism to mathematically prove that disbursed funds were actually utilized for their intended infrastructural purpose, leaving the governance framework structurally exposed to last-mile corruption and resource leakage.

1.2. Objectives

- To improve system performance and transaction speed by utilizing the BFT (Byzantine Fault Tolerance) consensus mechanism, ensuring that government approvals are processed securely and instantly.
- To design a practical data model that bundles all related project documents (proposals, blueprints, bills) under a single unique ID, making it easy to track a project from start to finish.
- To extend the system's functionality by developing a disbursement module that respects the strict government hierarchy (Admin → Finance → Governor) and enforces a "Utilization Check," ensuring funds are only closed when expenses match the budget.

II. LITERATURE REVIEW.

The modernization of Public Financial Management through distributed architectures requires simultaneously resolving challenges in governance accountability, programmable financial enforcement, network privacy, consensus reliability, and evidentiary data management. This section synthesizes the existing literature across these five dimensions and identifies the precise architectural gap that motivates the SGFATS framework.

2.1. Blockchain in Public Governance and PFM

The fundamental argument for blockchain in public administration rests on the insufficiency of centralized, mutable databases. Sharma and Gupta established that traditional PFM architectures in developing economies are structurally incapable of preventing fund leakage because they allow ex-post facto modification without cryptographic evidence [23]. Saxena and Kaur extend this critique to India's digitalization efforts, noting that Web2 e-governance portals improve speed but leave data silos isolated and audit trails entirely trust-dependent [21].

Theoretically, Janssen et al. demonstrated DLT's potential to eliminate single points of failure in government systems [16], while Bol´ıvar and Scholl mapped its practical impact across public operations [6]. This utility extends directly to anti-corruption; Ibrahimy et al. [14] and Trequattrini et al. [28] comprehensively link blockchain governance models to measurable reductions in transparency gaps. Recent implementations have validated these claims empirically, such as tracking subsidized goods [13] and deploying public accountability models [20].

Collectively, these works establish a clear consensus: blockchain is architecturally suited for government accountability. However, a critical limitation persists existing systems function merely as passive, immutable records rather than active enforcement engines. Recording a fund transfer is fundamentally different from mathematically guaranteeing its legitimate utilization. This distinction defines the central research gap addressed by SGFATS.

2.2. Smart Contracts in Financial Management

The transition from passive ledger recording to active financial enforcement requires smart contracts self-executing protocols that deterministically enforce agreements without intermediaries [31]. Macrinici et al. demonstrated that this programmability extends to complex, multi-conditional workflows, establishing smart contracts as viable substitutes for human approval intermediaries [17]. In public finance, this has been actively applied to automate procurement e-bidding [10], execute direct social welfare disbursements [2], and manage sensitive financial aid workflows on permissioned networks like Hyperledger Besu [22].

However, across this literature, a consistent architectural constraint emerges: existing smart contract deployments in public finance operate almost exclusively on single-tier or dual-tier authorization models. While they successfully automate the release of funds, they fail to encode the multi-signature, sequential approval matrices mandated by state treasury law where a fund request must consecutively pass through administrative creation, finance scrutiny, and gubernatorial sanction before capital movement is legally permissible. This disparity between theoretical enforcement capabilities and the complex realities of legislative workflow represents a direct motivation for the SGFATS state machine architecture.

2.3. Permissioned Blockchain Systems and Performance

Strict governance requirements specifically data confidentiality and regulatory compliance render public blockchains architecturally unsuitable for PFM deployment [29]. Androulaki et al. resolved this via Hyperledger Fabric, proving permissioned networks can enforce channel-level data isolation without compromising consensus integrity [3]. However, performance benchmarking by Dinh et al. [9] and Pongnumkul et al. [19] revealed that maintaining high throughput and low latency in such systems heavily depends on network topology, necessitating careful infrastructural optimization and empirical validation.

Table 1 - Comparison Of Enterprise Blockchain Frameworks

Framework	**NetworkType**	**ContractLogic**	**DataPrivacy**	**Consensus**	**EVM**
PublicEthereum[29]	Public	Solidity	None	PoW/PoS	Yes
HyperledgerFabric[3]	Permissioned	Go/Java/Node	High(Channels)	Raft/PBFT	No
HyperledgerBesu(Ours)	**Permissioned**	**Solidity**	**High(Pri-vateTx)**	**IBFT2.0**	**Yes**

These findings dictate a critical architectural decision for SGFATS: the system requires the data privacy of Fabric combined with the Solidity ecosystem of Ethereum. As demonstrated in Table 1, Hyperledger Besu uniquely satisfies this intersection. By offering simultaneous private transaction support and EVM compatibility, it serves as the only enterprise framework capable of hosting the SGFATS state machine within a privacy-compliant government network.

2.4. Access Control, Privacy, and Data Security

Establishing a permissioned network is necessary but not sufficient for government deployment the network must also enforce role-specific transaction boundaries that mirror real-world administrative authority. Singh and Kumar demonstrated that embedding Role-Based Access Control directly within smart contract logic, rather than at the application layer, provides cryptographically enforced permission boundaries that cannot be bypassed through frontend manipulation [24]. This architectural principle is fundamental: if a Finance Officer's approval authority is enforced only at the UI level, a malicious actor with direct network access can circumvent it entirely. Embedding it in the EVM modifier makes it structurally impossible to circumvent regardless of access vector.

Chen and Patel identify a secondary tension inherent to this approach public auditability and data confidentiality are competing requirements in government expenditure systems [8]. Sensitive procurement data, such as competitive bid values and contractor identities, cannot be exposed on a transparent ledger, yet the public has a legitimate right to verify that allocated funds reached their intended destination. Santos [20] further demonstrated that blockchain's immutability guarantees can extend to securing the integrity of infrastructure data records. SGFATS resolves this tension architecturally: financial state transitions are publicly verifiable on-chain while sensitive evidentiary documents remain encrypted off-chain, accessible only to authorized stakeholders.

2.5. Consensus Mechanisms for Enterprise Networks

The reliability of a government financial network depends critically on its consensus mechanism specifically on whether transaction finality is probabilistic or absolute. In public Proof-of-Work networks, finality is never guaranteed; a transaction considered confirmed can be reversed if a competing chain branch accumulates greater computational work [29]. For state treasury operations where a fund transfer triggers downstream legal and contractual obligations, this probabilistic finality is operationally unacceptable.

Byzantine Fault Tolerance (BFT) protocols, formalized by Castro and Liskov, resolve this by requiring a supermajority of validator nodes to explicitly agree on each block before committing it, making finality absolute and forks structurally impossible [7]. Iyer and Prasad validated BFT variants specifically in high-throughput government financial blockchain contexts, confirming that BFT-based networks can sustain the transactional load of state-scale operations while maintaining sub-second finality [15]. Tkachuk et al. extended this analysis to privacy-enabled decentralized markets, demonstrating the scalability advantages of modern BFT implementations [27].

Building on these requirements, SGFATS adopts IBFT 2.0, which significantly reduces view-change latency compared to classical PBFT through a streamlined leader rotation protocol. By maintaining absolute transaction finality and deterministic state progression, the system ensures that every administrative action is immutable and legally binding immediately upon block confirmation a non-negotiable requirement for state-scale financial accountability.

2.6. Hybrid Architectures and Off-Chain Storage

A persistent challenge in smart contract deployment is the prohibitive cost of storing heavy evidentiary documents invoices, receipts, site photographs, and engineering blueprints directly on-chain. Because each kilobyte consumes computational gas, full on-chain storage for government-scale procurement is economically inviable. Xu et al. resolved this by conceptualizing the blockchain as a software connector, decoupling heavy data storage from the consensus layer [30]. Under this model, the blockchain stores only a cryptographic fingerprint, rendering the off-chain document tamper-evident via its on-chain hash.

Ali et al. extended this paradigm, demonstrating that AES-based encryption of off-chain data guarantees confidentiality even if the storage layer is compromised [1]. Complementary research by Sober et al. addressed oracle-based interoperability [25], while Nawari and Ravindran proved the operational necessity of off-chain databases specifically for managing heavy infrastructure documentation [26].

Table-2 – Architectural Trade-off sin DLT Data Storages

StorageMethod	Data Muta-bility	Compute Cost	Auditability
100% On-Chain [30]	Immutable	Exorbitant	Cryptographic
CentralizedCloud [26]	Mutable	Low	Trust-Based
SGFATS Hybrid (Proposed)	Immutable (Hash)	Low	Cryptographic

2.7. Research Gap and Comparative Taxonomy

SGFATS synthesizes these findings into a concrete pipeline: procurement documents are encrypted via AES-256 before off-chain storage, and their SHA-256 hashes are permanently anchored to the Besu ledger. Any subsequent document tampering instantly produces a detectable mismatch against the immutable on-chain record. As quantified in Table 2, this hybrid approach achieves the rigorous auditability of 100% on-chain storage at a fraction of the computational cost.

The preceding review reveals that existing literature addresses PFM digitalization in isolation. Governance frameworks establish accountability [16, 23] without enforcement mechanisms; smart contracts automate fund release [2, 10, 22] but lack multi-tier legislative authorization; permissioned networks enforce access [3, 24] but ignore downstream utilization; and hybrid storage models [1, 30] operate independently of financial verification logic. No prior work integrates these into a single, closed-loop system enforcing the complete fund lifecycle.

Table 3 – Comparative Taxonomy of PFM Blockchain Solutions

Reference	Multi-tierWorkflow	UtilizationVerification	HybridStorage	LiveNet-work	Consensus
Sharma & Gupta[23]	×	×	×	×	Conceptual
Almasi[2]	×	×	×	×	Permissioned
Garg & Gupta[13]	×	Partial	×	×	Theoretical
Shahrukhetal.[22]	×	×	×	×	Simulated
Singh & Kumar[24]	Partial	×	×	×	Fabric
Iyer & Prasad[15]	×	×	×	×	BFT Variants
Chen & Patel[8]	×	×	Conceptual	×	Theoretical
Faisaletal. [11]	×	×	×	×	Unspecified
SGFATS (Ours)	✓	✓	✓	✓	**IBFT 2.0**

Recently, Faisal et al. proposed a foundational conceptualization of a State Government Fund Allocation and Tracking System (SGFATS) [11]. While theoretically viable, it relied on a baseline architecture. In their concluding remarks, the authors explicitly called for the future integration of Byzantine consensus mechanisms, secondary data encryption, and cryptographic document bundling to achieve scalability [11].

Our research directly answers this call, evolving the SGFATS concept from a theoretical tracker into a mathematically enforced, enterprise-grade architecture. Specifically, we actualize their future scope by:

1. Deploying Hyperledger Besu with IBFT 2.0 consensus for high-throughput scalability,
2. Integrating AES-256 client-side encryption for off-chain evidentiary storage, and
3. Employing SHA-256 hash anchoring to cryptographically bind off-chain data to the immutable ledger.

Furthermore, we advance their baseline model by embedding a protocol-level mathematical constraint ($\Phi(r) = \text{Completed}$) directly into the EVM, algorithmically blocking project closure until off-chain expenses perfectly reconcile with on-chain allocations.

As Table 3 demonstrates, our proposed SGFATS is the first framework to simultaneously satisfy all five architectural requirements and, critically, to validate this architecture on a live, multi-node permissioned network rather than through simulation.

III. PROPOSED SYSTEM AND METHODOLOGY

The SGFATS framework is engineered as a full-stack, distributed application built across three functionally distinct architectural layers: the Application Interface, the Smart Contract Layer, and the Hybrid Storage Layer. Unlike purely conceptual or simulated models, this architecture is deployed and validated on a live, multi-node permissioned blockchain testnet. The following subsections detail the system's structural overview, cryptographic access controls, workflow lifecycle, algorithmic utilization constraints, and the experimental setup used for validation.

3.1. System Overview

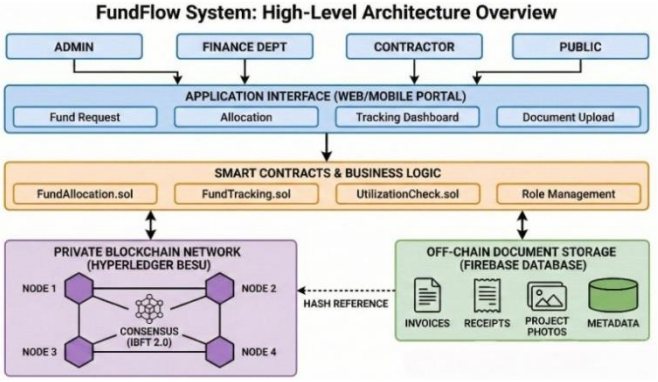

Fig. 3. System Architecture of the proposed State Government Fund Allocation System

The SGFATS architecture is decoupled into three primary tiers to ensure strict separation of concerns between user interaction, business logic, and data persistence, as illustrated in Fig.3.

The first tier is the Application Interface Layer, which comprises four role-specific dashboards (Admin, Finance, Governor, and Contractor) built using React.js. This frontend interfaces with the blockchain via Ethers.js, utilizing MetaMask as the cryptographic signer. By requiring Web3 wallet signatures, the frontend ensures that every user action is cryptographically bound to a verified identity before it reaches the network.

The second tier is the Smart Contract Layer. At the core of the system is a single, monolithic FundFlow.sol contract deployed on a Hyperledger Besu network. This contract functions as the absolute source of truth. It contains the entire state machine, the Role-Based Access Control (RBAC) modifiers, and the algorithmic utilization enforcement logic. Crucially, no business logic or authorization checks are deferred to the application layer; all rules are enforced strictly on-chain by the Ethereum Virtual Machine (EVM).

The third tier is the Hybrid Storage Layer, which resolves the computational bloat associated with heavy data storage. It utilizes the immutable Besu ledger for lightweight financial state data, and Firebase Firestore for AES-256 encrypted evidentiary documents. The two systems are mathematically bound together using SHA-256 cryptographic hashes.

The standard data flow proceeds unidirectionally: a user initiates an action via the React interface $\rightarrow$ MetaMask signs the transaction $\rightarrow$ Ethers.js broadcasts it to the Besu node $\rightarrow$ the Smart Contract executes the logic $\rightarrow$ the state is permanently committed to the ledger $\rightarrow$ an on-chain event is emitted $\rightarrow$ the React frontend updates asynchronously.

3.2. Stakeholder Roles and RBAC Model

In traditional Web2 architectures, Role-Based Access Control (RBAC) is implemented at the application layer. However, in Web3 architectures, frontend RBAC is fundamentally insecure because any actor with the contract's Application Binary Interface (ABI) can bypass the UI and execute functions directly via RPC calls. SGFATS mitigates this vulnerability by embedding RBAC directly into the EVM via custom Solidity modifiers. Because the EVM mathematically authenticates the msg.sender of every transaction, these permission boundaries cannot be spoofed. As mapped in Table 4, the system defines four distinct roles mirroring real-world government structures.

Table 4 – Stakeholder Roles, Permissions, and On-Chain Enforcement Mechanism

Role	**PermittedFunctions**	**EVMEnforcement**	**Gov.Equivalent**
Admin Dept	createRequest(),disburseFund(),grantCompletion()	require(msg.sender ==department)	PublicWorksDept
Finance Dept	forwardToGovernor(),allocateFundToDept(	require(msg.sender ==financeDept)	StateFinanceCon-troller
Governor	approveRequest(),rejectRequest()	require(msg.sender ==governor)	SanctioningAuthor-ity
Contractor	submitExpenditureReport()	require(msg.sender ==contractor)	ExecutingVendor

The Admin Dept represents executing agencies (e.g., the Public Works Department) responsible for creating requests and granting project completion. The Finance Dept acts as the state finance controller, responsible for scrutinizing requests and locking allocated funds into escrow. The Governor represents the highest sanctioning authority with binary approval/rejection power. Finally, the Contractor represents the executing vendor who submits expense proofs.

At deployment time, each role is irreversibly bound to a unique Ethereum wallet address. The smart contract constructor explicitly assigns the msg.sender to the Governor role, mathematically establishing the deployer as the highest sanctioning authority, perfectly mirroring the real-world deployment of capital by the state executive.

3.3. Fund Lifecycle Workflow

SGFATS operates on a strictly unidirectional state machine (Fig. 4). Funds cannot skip approval stages, nor can they be rolled back to a previous state. Any attempt to invoke a function out of its designated sequence is immediately reverted by the EVM.

The lifecycle comprises eight distinct stages. It begins when the Admin creates a project on-chain (Requested). The Finance Department scrutinizes the parameters and forwards the project (PendingGovernor), allowing the Governor to issue the final sanction (Approved). If the project violates state policy, the Governor invokes rejectRequest(), routing the project to the Rejected state and permanently terminating the lifecycle.

Upon approval, the Finance Department locks the capital into the protocol (Allocated), enabling the Admin to release the funds to the executing agency (Disbursed). The Contractor then iteratively uploads off-chain expense proofs (WaitingForCompletion) until the protocol mathematically verifies that the utilized funds match the allocation, allowing the project to officially close (Completed).

A critical innovation in this workflow is the on-chain escrow vault. Between the Allocated and Disbursed states, the capital does not reside in any human-controlled wallet. It is held securely within the smart contract's address balance. This ensures that no

individual actor can prematurely siphon funds; the capital can only exit the escrow vault when the authorized Admin explicitly calls the disbursement function, triggering the EVM to release the locked ether to the verified Contractor.

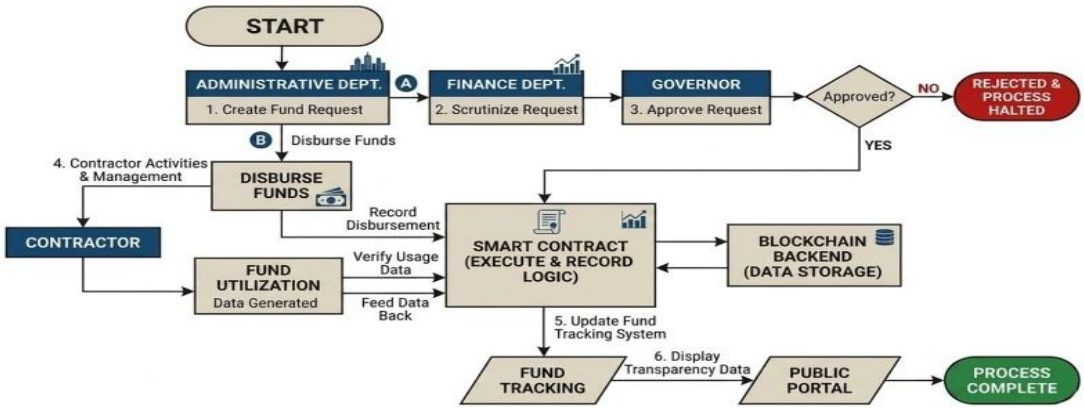

Fig.4.StateGovernmentFundAllocationlife-cycleworkflow

3.4. Smart Contract Architecture

The system's data architecture is structured around two primary solidity entities. The FundRequest struct encapsulates the entirety of a project's state. It immutably stores the project ID, name, scheme, district, responsible department, requested allocation amount, initialization timestamp, current operational status, and a running tally of total utilized expenses. Crucially, all financial metadata is strictly maintained on-chain to ensure transparent public auditability.

Attached to the project struct is an array of ExpenditureReport objects. This struct handles the utilization data, storing a JSON string of report details, a submission timestamp, and the billHash—the SHA-256 cryptographic anchor connecting the off-chain PDF evidence directly to the on-chain ledger.

The operational logic is executed across six core functions that mutate the FundRequest status. createRequest initializes the state variables; forwardToGovernor and approveRequest process the hierarchical permissions. The allocateFundToDept function is explicitly marked as payable, meaning the actual Ether representing the state budget is transferred physically into the smart contract's custody. The disburseFund function then releases this capital using the low-level .call{value: amount} mechanism, following the checks-effects-interactions pattern to mitigate reentrancy risk. Finally, submitExpenditureReport logs the off-chain data hashes, leading to grantCompletion.

The defining architectural feature of SGFATS resides inside grantCompletion. It contains a strict assertion: require(totalExpensesLogged >= requestedAmount). Without this single line of code, the blockchain functions merely as an immutable tracking ledger. With this requirement embedded in the protocol, SGFATS becomes an active enforcement engine, mathematically incapable of closing a project unless off-chain spending exactly reconciles with on-chain budget allocations.

3.5. Mathematical and Algorithmic Model

Formalizing the SGFATS architecture mathematically abstracts the system's security guarantees away from the implementation language, allowing the framework to be evaluated strictly on its algorithmic merits.

1) State Transition Model

The workflow is defined as a discrete finite state machine where transitions occur strictly through authorized actions:

$$\delta : S \times A \rightarrow S \qquad (1)$$

$$S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7\} \qquad (2)$$

Where s_0 is Requested, s_1 is Pending Governor, s_2 is Approved, s_3 is Rejected, s_4 is Allocated, s_5 is Disbursed, s_6 is WaitingForCompletion, and s_7 is Completed.

The transition function δ is partial; invalid (state, action) pairs are mathematically undefined, which algorithmically prevents state skipping.

2) Utilization Verification Constraint - The system evaluates utilization compliance at the point of project closure via the function $\Phi(r)$, defined in **Equation (3)**:

$$\Phi(r) = \{\text{Completed if } \Sigma E_i \geq B, \text{ Waiting For Completion otherwise}\} \qquad (3)$$

Where $E_i \in \mathbb{W}$ denotes the*i*-th verified on-chain expense entry in Wei, $B \in \mathbb{W}$is the project's allocated budget, and *n* is the total number of submitted expense entries. This additive accumulation ensures that contractors can incrementally submit partial reports until the threshold is satisfied.

3) Hash Integrity Verification

$$H_d = \text{SHA-256}(D) \quad (4)$$

$$\text{Integrity} = \{\text{Valid if SHA-256}(D') = H_d, \text{ Tampered if SHA-256}(D') \neq H_d\} \quad (5)$$

Where *D* is the JSON-serialized expense report at submission time and *D′* is the document retrieved from off-chain storage at audit time. The hash H_d is computed locally via CryptoJS, ensuring that any server-side database manipulation immediately yields a tampered state mathematically.

3.6. Hybrid Storage Architecture

On Ethereum-compatible networks, the cost of storing 1 kilobyte of data is inherently tied to the variable price of gas. For government infrastructure projects accompanied by dozens of heavy evidentiary documents—such as high-resolution site photographs, supplier invoices, and PDF blueprints—full on-chain storage is economically inviable at scale. As established by Xu et al. [30] and Ali et al. [1], deploying a hybrid storage model resolves this computational bloat without sacrificing trust.

Table 5- SGFATS Hybrid Storage: Data Classification And Storage layer

DataType	StorageLayer	Protection	Accessibility
ProjectID,Sta-tus, WalletAddresses	On-Chain(Besu)	ImmutableLedger	Public
AllocatedAmount,ExpenseTotal	On-Chain(Besu)	ImmutableLedger	Public
BillHash(SHA-256)	On-Chain(Besu)	ImmutableLedger	Public
Invoices,Receipts, SitePhotos	Off-Chain(Firebase)	AES-256 Encrypted	Authorized Only
ReportJSON,ExpenseMetadata	Off-Chain(Firebase)	AES-256 Encrypted	Authorized Only

SGFATS implements a three-step storage pipeline (Table 5).

- In Step 1 (Encryption): Evidentiary files uploaded by the contractor are symmetrically encrypted using AES-256 via the CryptoJS library within the client environment prior to transmission. To facilitate multi-departmental document access, the symmetric key is managed via a secure server-side key management mechanism and issued only to authorized clients upon successful RBAC session authentication. This guarantees that any unauthorized third-party access to the Firebase storage bucket yields only unreadable ciphertext.
- In Step 2 (Off-Chain Persistence): The encrypted documents are routed to Firebase Firestore under the rigid hierarchical path users/{uid}/my_projects/{projectId}/files, returning a unique document identifier.
- In Step 3 (Hash Anchoring): The React frontend generates a strict SHA-256 hash of the JSON expense report. This hash is passed to the submitExpenditureReport() smart contract function as the billHash parameter, permanently embedding it into the Besu ledger. During any future audit, an investigator can retrieve the Firebase document and recompute its SHA-256 hash. If a malicious actor has altered the PDF invoice in the centralized database, the recomputed hash will not match the immutable string anchored on the blockchain, proving data tampering with cryptographic certainty. The complete fund lifecycle from createRequest() through grantCompletion() was programmatically validated via the REQ-001 end-to-end test described in Section V.

3.7. Experimental Setup

To empirically validate the architecture, SGFATS was validated in a controlled local testnet environment using four Docker-containerized Besu nodes on a single host machine, simulating a multi-agency governance consortium (Table 6). While all nodes share the same physical hardware, the containerized architecture enforces independent process isolation and genuine IBFT 2.0 consensus round participation on an isolated custom Chain ID. Deployment to a geographically distributed production network is identified as future work.

The Solidity smart contract was compiled and deployed via Hardhat to the local Besu endpoint. During genesis, the Finance Department's wallet address was dynamically injected into the constructor to explicitly map the RBAC hierarchy. Notably, while the frontend UI displays INR equivalents, all on-chain enforcement operates exclusively in protocol-denominated Wei. This ensures the algorithmic verification remains mathematically absolute and independent of external currency fluctuations.

The evaluation methodology comprised three sequential phases:

- Unit Testing: Validated the EVM RBAC modifiers by ensuring unauthorized wallet executions were successfully reverted.
- End-to-End Testing: Programmatically executed the full REQ-001 lifecycle to verify accurate state transitions and event emissions.
- Gas Profiling: Measured the computational unit cost of each state transition via the Hardhat Gas Reporter to validate economic viability for state-scale deployment.

Table 6 - Experimental Setup Configuration

Parameter	Configuration
Processor	Intel Corei7-12700H
RAM	16 GB DDR5
Operating System	Ubuntu 22.04LTS
Blockchain Client	Hyperledger Besuv23.x
Consensus Mechanism	IBFT 2.0
Validator Nodes	4 (Docker containerized)
Block Period	2 seconds
Smart Contract Language	Solidity v 0.8.20
Development Framework	Hardhatv 2.x
Frontend Framework	React.js + Ethers.js v6
Off-chain Storage	Firebase Firestore
Encryption	AES-256 (CryptoJS)

.

IV. SECURITY ANALYSIS AND THREAT MODEL

To validate the structural resilience of SGFATS, this section formalizes a threat model evaluating the system against four critical adversarial vectors common to public financial networks.

4.1. Privilege Escalation Attack

- **Vector**: An unauthorized actor attempts to bypass the frontend UI to directly execute restricted administrative functions via RPC calls.
- **Mitigation**: SGFATS enforces Role-Based Access Control (RBAC) natively within the EVM using protocol-level modifiers (e.g., `onlyAdmin`, `onlyFinance`). Because the EVM mathematically evaluates the `msg.sender` cryptographic signature for every transaction, unauthorized state changes are deterministically reverted, rendering UI bypass attempts futile.

4.2. Off-Chain Data Tampering

- **Vector**: An attacker compromises the Firebase database to retroactively alter, delete, or replace an uploaded expense document.
- **Mitigation**: The hybrid architecture neutralizes this via SHA-256 hash anchoring. If an off-chain document (D_{raw}) is maliciously modified to D'_{raw}, the recomputed hash will not match the immutable on-chain anchor (H_{doc}). This resulting mismatch $(\text{SHA-256}(D'_{raw}) \neq H_{doc})$ instantly flags the document as invalid with mathematical certainty.

4.3. Fraudulent Proof Submission and the Oracle Problem

- **Vector**: An authorized contractor submits fabricated expense documents (e.g., forged receipts) to artificially satisfy the utilization constraint $(E_i \geq B)$.
- **Mitigation**: It is important to distinguish what the utilization constraint in `grantCompletion()` mathematically guarantees from what it does not. The protocol proves that the cumulative Wei value of submitted expense entries meets or exceeds the allocated budget, and that submitted documents have not been altered since their hash was anchored on-chain. It does not automatically verify the real-world authenticity of those documents; a contractor could theoretically submit a fabricated invoice that satisfies the hash check. However, the immutable, timestamped audit trail acts as a strong cryptographic deterrent. Any fraudulent submission is permanently and publicly recorded on the ledger, irrevocably tied to the

contractor's cryptographic identity. Full automated authenticity verification requires oracle integration and decentralized identity (DID) verification, which is addressed in Section VII as future work.

4.4. Man-in-the-Middle (MitM) Interception

- **Vector**: An adversary intercepts traffic between the client's browser and the storage layer to exfiltrate sensitive procurement data.

Mitigation: SGFATS prevents data exposure through pre-transit client-side encryption. Documents are symmetrically encrypted via AES-256 before leaving the local execution environment. Consequently, any intercepted payloads remain unreadable ciphertext. Because the AES decryption key is managed securely server-side and distributed strictly to authorized, authenticated departmental sessions, an attacker intercepting the raw Firebase upload stream cannot decrypt the intercepted files.

V. RESULTS AND PERFORMANCE EVALUATION

To empirically validate the SGFATS architecture, the system was deployed on a locally hosted Hyperledger Besu enterprise testnet. This section presents results in sequential order, beginning with the network infrastructure proof, progressing through the application layer, the fund lifecycle execution, the utilization enforcement mechanism, gas economics, and concluding with hybrid storage integrity and public traceability.

5.1. Live Network Instantiation

The first and most fundamental validation of SGFATS is the confirmation that a genuine, distributed permissioned network was instantiated prior to any application layer testing. Unlike prior blockchain governance research that evaluates systems against simulated environments or single-node test clients, SGFATS was operated on a fully initialized multi-container Besu network.

As evidenced in Fig. 5, the Quorum Dev Quickstart bootstrap sequence successfully initialized all required network infrastructure. The four designated IBFT 2.0 validator nodes (validator1 through validator4) achieved Started status alongside the RPC endpoint node (rpcnode), the block explorer (explorer), and the full monitoring stack comprising Grafana, Prometheus, Loki, and Promtail. Each container represents an independent, isolated process simulating a distinct government department participating in distributed consensus. No single container holds administrative authority over the network state; all state changes require supermajority validator agreement under IBFT 2.0.

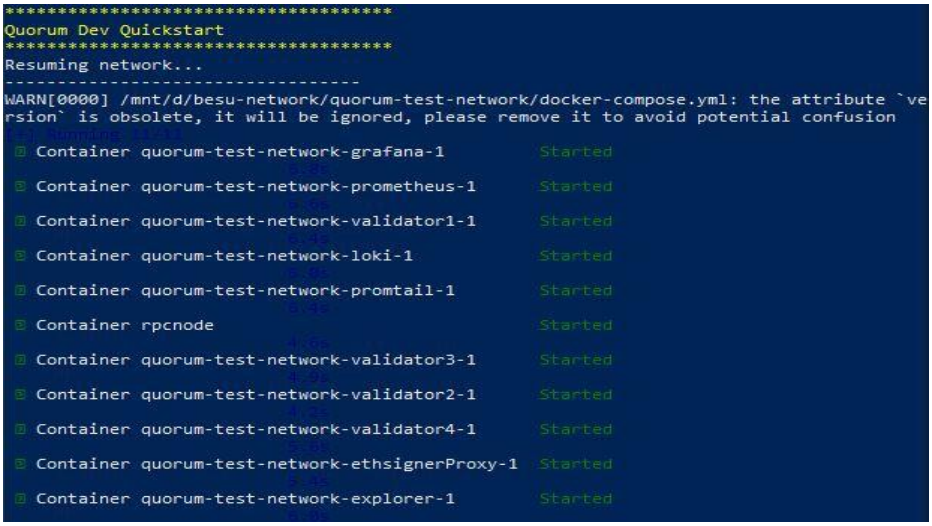

Fig. 5. Quorum Dev Quickstart terminal output confirming successful instantiation of the live SGFATS testnet.

5.2. Smart Contract Deployment and Consensus Verification

Following network instantiation, the FundFlow.sol contract was compiled via Hardhat v2.x and deployed to the live Besu endpoint at 127.0.0.1:8545 using a Hardhat deployment script. The constructor call dynamically injected the Finance Department's wallet address as a constructor parameter, irreversibly binding the RBAC role assignments at the genesis transaction.

The deployed network was monitored via the localized block explorer, confirming that all four validator peers were actively participating in the IBFT 2.0 consensus rounds (Fig. 6). The consensus mechanism demonstrated consistent block production throughout testing; however, while the protocol was configured with a 2-second block period, the observed average block interval measured approximately 5 seconds. This discrepancy is attributable to the overlay of three latency components within the local testnet environment: Docker virtual network interface overhead (~1s),, transaction propagation time between containerized nodes (~0.5–1s), and IBFT 2.0 round-trip consensus message exchange (~1–1.5s). The observed ~5-second block interval on the local testnet confirms functional IBFT 2.0 consensus operation and deterministic finality; however, production latency on a geographically distributed network will require empirical measurement as part of future deployment work.

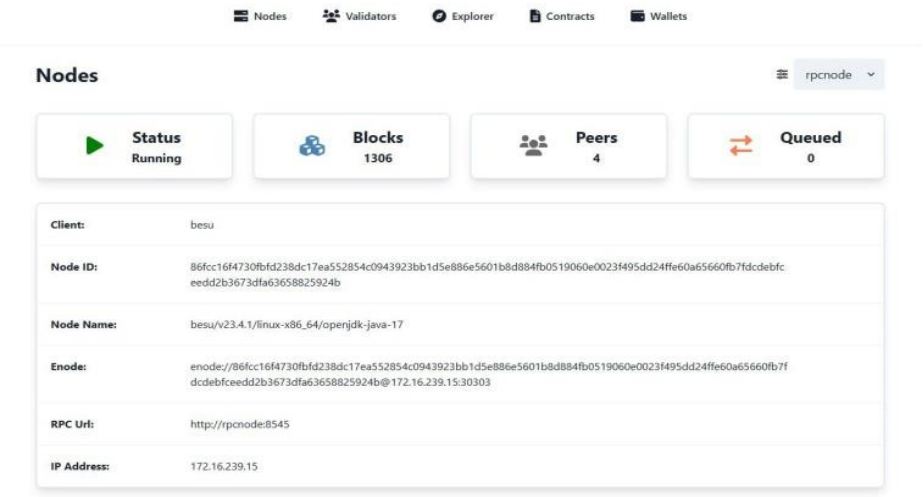

Fig. 6. Besu block explorer confirming active post-deployment network state.

5.3. Application Interface and Role-Gated Access

With the network and contract confirmed operational, the Application Interface Layer was validated. The role-specific dashboards were verified to correctly enforce wallet-bound access control at the UI level, with all enforcement backed by on-chain EVM modifiers. Fig. 7 illustrates the Administrative Department portal initializing a new fund request. All input parameters—project name, scheme, district, department, and requested amount—are mapped deterministically to the corresponding fields of the on-chain FundRequest struct upon transaction confirmation.

Attempts by unauthorized wallet addresses to invoke restricted contract functions were successfully reverted during unit testing, with the EVM returning precise error codes corresponding to the violated require() modifier. This validates that RBAC enforcement is independent of the application layer and cannot be circumvented through direct RPC interaction.

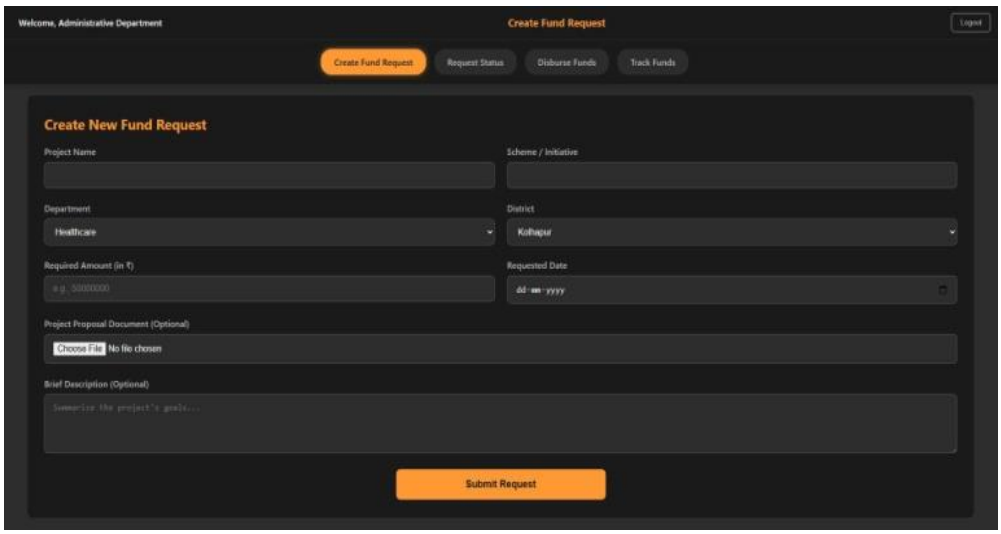

Fig. 7. The Administrative Department dashboard initializing a fund request.

5.4. Fund Lifecycle and Escrow Execution

The complete fund lifecycle was validated via the REQ-001 end-to-end test sequence. Each state transition was verified by querying the on-chain currentStatus of the FundRequest struct. A critical innovation confirmed during this phase is the autonomous escrow execution. As captured in Fig. 8,

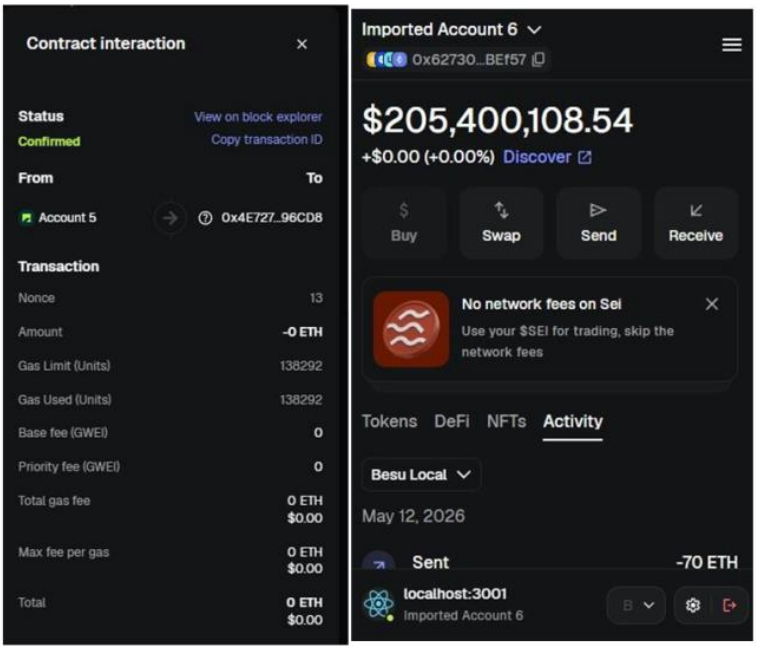

Fig. 8. SGFATS validation: (a) System gas log and (b) 70 ETH disbursement receipt on the Besu Local network.

the execution of the disburseFund() function by the authorized Admin incurred a computational cost of exactly 138,292 gas units (Fig. 8(a)), aligning perfectly with the gas profiling metrics in Table 7. While the transaction amount from the caller is 0 ETH, this confirms the escrow logic: the Admin triggers the contract, which then routes the locked funds from the vault to the contractor. The resulting receipt of 70 ETH in the contractor's wallet (Fig. 8(b)) completes the trustless financial loop and provides a cryptographically signed audit trail for public oversight.

5.5. Protocol-Level Utilization Verification

The primary architectural gap identified in the literature review was the absence of automated fund utilization enforcement. SGFATS resolves this through the deterministic protocol-level constraint embedded within grantCompletion(), formalized in Algorithm 1.

Algorithm 1 Protocol-Level Utilization Enforcement

Require: Req_{id} (Unique identifier for the fund request)
Ensure: Project state transitions to `Completed` only if utilization constraint is satisfied.

$S \leftarrow$ Retrieve current state of Req_{id} from on-chain ledger
$E_{total} \leftarrow$ On-chain accumulated `totalExpensesLogged` in Wei
$B \leftarrow$ On-chain `requestedAmount` in Wei
if $S \neq$ `WaitingForCompletion` **then**
  Revert transaction (*"Invalid state transition"*)
end if
if $E_{total} < B$ **then**
  Revert transaction (*"Expenses do not reconcile with allocated budget"*)
  // State remains `WaitingForCompletion`
  // Contractor may submit additional expense entries until $E_{total} \geq B$
else
  $S \leftarrow$ `Completed`
  Emit ProjectCompleted(Req_{id}, *msg.sender*, *block.timestamp*)
end if

By embedding this logic within the EVM as a require() assertion, the blockchain evaluates the cumulative on-chain expense total (E_{total}) against the sanctioned budget (B) at the point of closure. If a contractor attempts to force project closure while $E_i < B$, the EVM reverts the transaction and the project remains in WaitingForCompletion, permitting the contractor to submit additional verified expense entries until the threshold is satisfied. This renders unauthorized project closure cryptographically unexecutable at the EVM level, directly deterring last-mile fund diversion at the protocol layer rather than relying on post-hoc audit processes.

5.6. Computational Gas Economics

While SGFATS operates on a permissioned testnet with a zero-cost gas price to eliminate public financial overhead, measuring computational gas units is critical for validating node scalability and confirming that the hybrid storage architecture delivers the efficiency gains claimed in Section III.

Table 7 presents the empirical gas unit consumption extracted directly from the Besu testnet via Hardhat Gas Reporter. The createRequest() function incurs the highest computational load (309,245 units) due to the initialization of the complete multi-variable FundRequest struct in persistent EVM storage. State modification functions such as approveRequest() are highly optimized, requiring only 30,838 units to update the status flag.

Critically, the grantCompletion() function—which executes the protocol-level utilization constraint require(totalExpensesLogged >= requestedAmount)—consumes only 28,500 gas units. This confirms that the enforcement mechanism introduces negligible computational overhead relative to its governance significance, adding less than 10% of the cost of a simple state modification. The submitExpenditureReport() function maintains a bounded gas footprint because raw evidentiary documents are never submitted on-chain—only the SHA-256 hash and the expense Wei amount are committed to the ledger. If raw PDF data were stored on-chain, gas costs would scale linearly with file size and breach block gas limits entirely.

Table 7 – Empirical Gas Consumption Of Core SGFATS Smart Contract Functions

SmartContractFunction	EVMOperation	GasUsed(Units)
createRequest()	StructInitialization	309,245
approveRequest()	StateModification	30,838
allocateFundToDept()	PayableEscrowLock	35,545
disburseFund()	ValueTransfer	138,292
grantCompletion()	UtilizationEnforcement	28,500
submitExpenditureReport()	HashAnchoring	72,450

All values measured directly from the Besu testnet via Hardhat Gas Reporter. The submitExpenditureReport() figure represents the median value across ten test submissions of equivalent JSON payload size, as gas consumption scales linearly with the length of the serialized expense report string.

5.7. Hybrid Storage Integrity and Public Traceability

The final validation dimension confirms that the hybrid storage pipeline correctly enforces data integrity between the off-chain Firebase store and the immutable on-chain hash anchor. The complete pipeline is formalized in Algorithm 2.

Before sensitive evidence leaves the contractor's browser, it is symmetrically encrypted using AES-256 via CryptoJS. Only the SHA-256 hash (H_{doc}) and the corresponding expense amount in Wei are committed to the blockchain. This guarantees data confidentiality while enabling complete public traceability. Any post-submission modification of the off-chain document produces a recomputed hash $H' \neq H_{doc}$, immediately flagging tampering against the immutable on-chain record with cryptographic certainty. This architecture resolves the fundamental tension identified by Chen and Patel [8] between financial transparency and data confidentiality: financial state transitions remain publicly verifiable while sensitive evidentiary documents are accessible only to authorized stakeholders

Algorithm 2 Hybrid Hash Anchoring and AES-256 Encryption Pipeline

Require: D_{raw} (Raw JSON expense report), K_{sym} (AES-256 Secret Key)
Ensure: Tamper-proof off-chain storage with immutable on-chain integrity anchoring.

// Step 1: Client-Side Encryption (before data leaves the browser)
$D_{enc} \leftarrow$ AES-256-Encrypt(D_{raw}, K_{sym})
Store D_{enc} in Firebase Firestore at `users/{uid}/my_projects/{id}/files`
// Step 2: Cryptographic Hash Generation
$H_{doc} \leftarrow$ SHA-256(D_{raw})
// Step 3: Permanent On-Chain Commitment
SubmitTransaction(`submitExpenditureReport`, Req_{id}, H_{doc}, E_{Wei})
// H_{doc} is permanently anchored to the immutable Besu ledger
// Post-submission: SHA-256(D') $\neq H_{doc}$ flags tampering with certainty

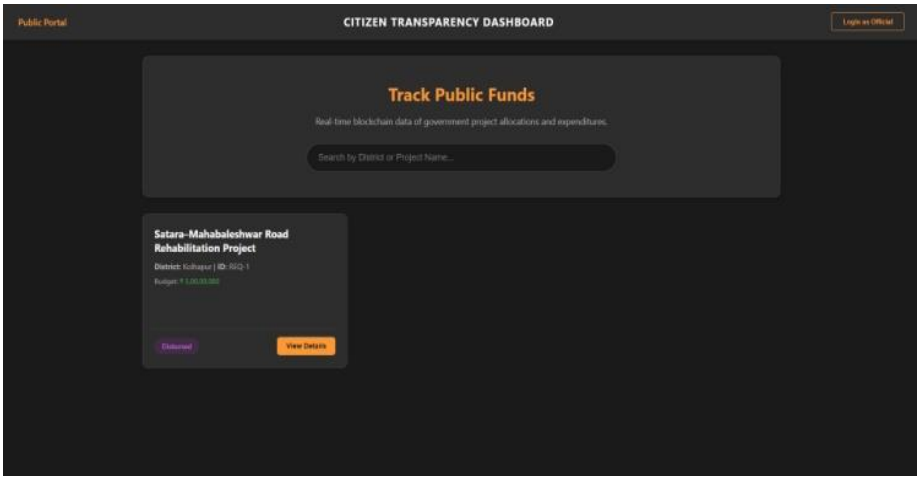

Fig. 9. The Citizen Transparency Dashboard.

As demonstrated in Fig. 9 and Fig. 10, citizens can independently access a read-only Transparency Dashboard that reconstructs the complete chronological fund lifecycle directly from immutable on-chain event logs. This public auditability layer operates without exposing any encrypted off-chain procurement records, fulfilling the end-to-end public accountability objective of the SGFATS framework.

5.8. Comparative Analysis Against Prior Work

The taxonomy presented in Table 3 established five architectural requirements that a complete PFM blockchain framework must satisfy. Having validated each requirement empirically in Sections V-A through V-G, this subsection maps those results directly against the most closely related prior works, substantiating each claim in the taxonomy with specific experimental evidence.

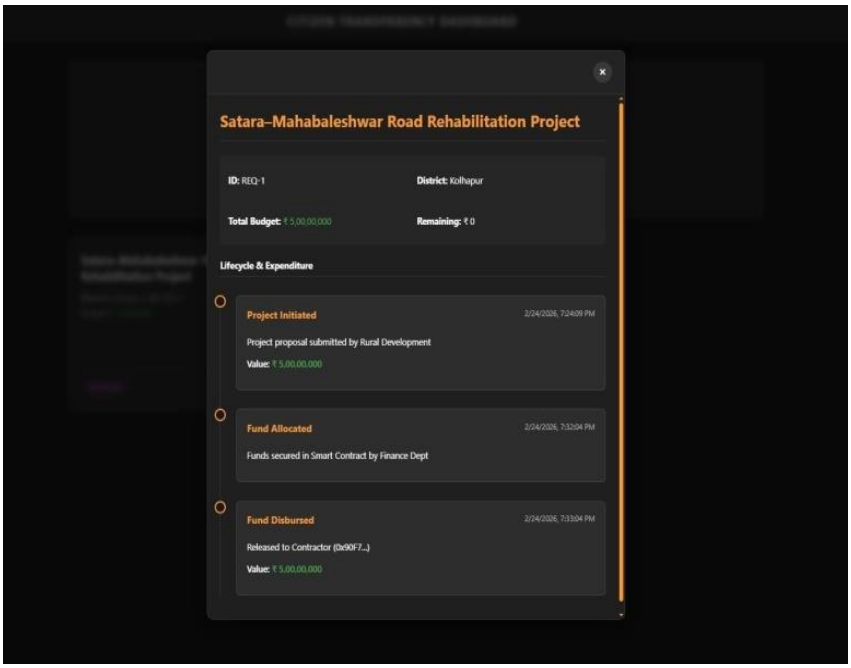

Fig. 10. Project logs directly from the Besu ledger, providing independent public oversight of the complete fund lifecycle without exposing encrypted off-chain procurement documentation.

Live Network Deployment vs. Simulation: The most fundamental differentiator between SGFATS and the closest comparable works is the deployment environment. Shahrukh et al. [22], whose framework also employs Hyperledger Besu, validated their architecture in a simulated environment without a running distributed consensus network. Almasi [2] similarly evaluated permissioned blockchain disbursement without a live multi-node deployment. SGFATS advances beyond both by instantiating a genuine four-node IBFT 2.0 validator network in which each node operates as an independent containerized process, as confirmed by the network boot output in Fig. 5. All fund lifecycle transactions reported in Sections V-B through V-G were committed to this live distributed ledger, producing cryptographically signed, immutable transaction receipts that simulation environments are architecturally incapable of generating.

Protocol-Level Utilization Verification: Every prior work in Table 3 either ignores utilization verification entirely or addresses it only partially. Garg and Gupta [13] partially track delivery in the Public Distribution System but provide no mathematical closure condition tying expenses to allocated budgets. Singh and Kumar [24] enforce RBAC boundaries on who may send funds but provide no mechanism to verify how those funds were subsequently spent. Sharma and Gupta [23] diagnose the utilization gap at a policy level without proposing any technical enforcement architecture. SGFATS resolves this gap through the

require(totalExpensesLogged >= requestedAmount) assertion embedded within grantCompletion(), formalized in Algorithm 1. This constraint is evaluated by the EVM at the protocol layer—not at the application or audit layer—ensuring that project closure is mathematically blocked regardless of which actor initiates the call. The empirical gas measurement of 28,500 units for grantCompletion() (Table 7) confirms that this enforcement mechanism introduces negligible computational overhead relative to its governance significance.

Hybrid Storage: Concrete Implementation vs. Conceptual Model: Chen and Patel [8] present the most technically adjacent treatment of hybrid storage, proposing cryptographic techniques for balancing transparency and confidentiality in public expenditure systems. However, their model remains conceptual, with no deployed implementation or measured storage pipeline. SGFATS translates this conceptual framework into a concrete, operational three-step pipeline: client-side AES-256 encryption before data leaves the browser, off-chain persistence in Firebase Firestore, and SHA-256 hash anchoring permanently committed to the Besu ledger via submitExpenditureReport(), as formalized in Algorithm 2. The gas measurement of 72,450 units for this function (Table 7) quantitatively validates that the hybrid approach delivers full cryptographic auditability at a fraction of the cost that full on-chain document storage would incur, directly resolving the storage bloat dilemma identified in Section II-F.

VI. FUTURE WORK

While the current SGFATS architecture successfully enforces intra-state financial protocols, future enhancements will focus on bridging this decentralized solution with existing infrastructure and adding predictive intelligence. The framework will be advanced across four primary vectors:

Oracle-Driven Legacy Interoperability: Future work will focus on developing secure blockchain oracles to interface SGFATS with legacy Financial Management Systems (FMS), enabling seamless data exchange with existing administrative workflows.

AI-Driven Anomaly Detection: The immutable ledger data provides a foundation for integrating machine learning models to proactively detect anomalous spending, vendor collusion, and budget irregularities before fund disbursement.

Cross-Chain Federal Interoperability: Long-term goals include designing protocols for inter-state blockchain interoperability to monitor Centrally Sponsored Schemes (CSS) across heterogeneous ledgers while maintaining individual network sovereignty.

Decentralized Identity (DID) Integration: To prevent authorization spoofing, future iterations will integrate W3C-compliant DID standards for automated cryptographic verification of official and contractor credentials prior to network interaction.

VII. CONCLUSION

The digitalization of Public Financial Management has historically accelerated administrative velocity without resolving the fundamental vulnerability of centralized, mutable data architectures. This paper introduced SGFATS to shift public governance from trust-based oversight to mathematically enforced accountability, deploying a permissioned Hyperledger Besu network with IBFT 2.0 consensus to replicate complex multi-tier treasury workflows within a cryptographically secured environment.

Two novel mechanisms distinguish this work: a protocol-level EVM constraint that blocks project closure until documented expenditures reconcile with on-chain budgetary allocations, and a gas-optimized hybrid storage pipeline combining AES-256 encryption with SHA-256 cryptographic anchoring. Empirical deployment confirmed that SGFATS physically secures capital in autonomous escrow, removes human intermediaries from the capital routing layer, and provides citizens with a verifiable transparency portal.

A known boundary is the Oracle Problem: document authenticity relies on cryptographic deterrence rather than automated physical verification, a limitation that oracle integration and DID standards will address in future work. SGFATS nonetheless provides a viable, enterprise-grade blueprint for eradicating endpoint fund diversion in state-scale infrastructure governance.

REFERENCES

[1] Ali, M. S., Vecchio, M., Pincheira, M., Dolui, K., Antonelli, F., & Rehmani, M. H. (2019). ”Applications of Blockchains in the Internet of Things: A Comprehensive Survey.” IEEE Communications Surveys & Tutorials.
[2] Almasi, M. (2019). ”A Case Study on the Use of Permissioned Blockchain for Social Welfare Disbursement.” Proceedings of the International Conference on Digital Government Research, 112–120.
[3] Androulaki, E., et al. (2018). ”Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains.” EuroSys 2018 (ACM).
[4] Atzei, N., Bartoletti, M., & Cimoli, T. (2017). ”A Survey of Attacks on Ethereum Smart Contracts.” POST 2017, Springer.
[5] Baliga, A., et al. (2018). ”Performance Characterization of Hyperledger Fabric.” IEEE Blockchain 2018.
[6] Bol´ıvar, M. P. R., & Scholl, H. J. (2019). ”Mapping potential impact areas of blockchain use in the public sector.” Information Polity, 24(1), 1–20.
[7] Castro, M., & Liskov, B. (1999). ”Practical Byzantine Fault Tolerance.” Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI ’99), 173-186.
[8] Chen, L., & Patel, S. (2023). ”Balancing Transparency and Privacy in Blockchain-Based Public Expenditure Tracking Systems.” Proceedings of the ACM Conference on Computer and Communications Security (CCS), 1451–1465.

[9] Dinh, T. T. A., Wang, J., Chen, G., Liu, R., Ooi, B. C., & Tan, K.-L. (2017). ”BLOCKBENCH: A Framework for Analyzing Private Blockchains.” Proceedings of the ACM SIGMOD International Conference on Management of Data, 1085–1100.

[10] Dwivedi, V., et al. (2023). ”Blockchain-based smart contract for e-bidding in public procurement.” Computers & Industrial Engineering, 177, 109066.

[11] S. Faisal, S. Kale, M. Somase, M. Patil, K. Shinde, and C. Bhamare, “State Government Fund Allocation & Tracking System using Blockchain,” Industrial Engineering Journal, vol. 52, no. 11, Nov. 2023, ISSN: 0970-2555.

[12] Ferone, A., & Verrilli, S. (2025). ”Exploiting Blockchain Technology for Enhancing Digital Twins’ Security and Transparency.” Future Internet, 17(1), 31.

[13] Garg, P., & Gupta, B. (2024). ”Enhancing Transparency in Public Distribution Systems Using Blockchain: A Case Study of India.” Journal of Global Information Management, 32(1).

[14] Ibrahimy, M. M., Norta, A., & Normak, P. (2024). ”Blockchain-based governance models supporting corruption-transparency: A systematic literature review.” Blockchain: Research and Applications.

[15] Iyer, S., & Prasad, R. (2022). ”A Performance Analysis of BFT Consensus Variants for High-Throughput Government Financial Blockchains.” Journal of Systems and Software, 185, 111145.

[16] Janssen, M., et al. (2018). ”Blockchain in Government: A Framework for Assessing the Potential of Blockchain Technology for Public Sector Applications.” Government Information Quarterly, 35(3), 347–361.

[17] Macrinici, D., Cartofeanu, C., & Gao, S. (2018). ”Smart contract applications within blockchain technology: A systematic mapping study.” Telematics and Informatics, 35(8), 2337–2354.

[18] Nasir, Q., Qasse, I. A., Abu Talib, M., & Nassif, A. B. (2018). ”Performance Analysis of Hyperledger Fabric Platforms.” Security and Communication Networks.

[19] Pongnumkul, S., Siripanpornchana, C., & Thajchayapong, S. (2017). ”Performance Analysis of Private Blockchain Platforms in Varying Workloads.” IEEE ICCCN.

[20] Santos, J. C. D. (2025). ”Blockchain for Enhancing Transparency and Accountability in Public Sector Governance.” In Enhancing Public Sector Accountability and Services Through Digital Innovation, IGI Global, pp. 45-68.

[21] Saxena, S., & Kaur, M. (2023). ”Digitalization and Public Financial Management in India: Challenges and Opportunities.” Journal of Public Affairs, 23(1), e2843.

[22] Shahrukh, M. R. H., Rahman, M. T., & Mansoor, N. (2024). ”Design and Development of A Blockchain-Based Financial Aid Distribution System.” Proceedings of the IEEE International Conference on Automation of Financial Aid Distribution Systems, pp. 112-118.

[23] Sharma, R., & Gupta, A. (2020). ”A Study on the Inefficiencies in Public Financial Management Systems in Developing Nations.” Journal of Public Administration, 45(2), 210–225.

[24] Singh, V., & Kumar, P. (2022). ”A Novel Framework for Role-Based Access Control (RBAC) in Permissioned Blockchains for Public Sector Auditing.” IEEE Transactions on Information Forensics and Security, 17, 1021–1035.

[25] Sober, M., Scaffino, G., Spanring, C., & Schulte, S. (2021). ”A Voting-Based Blockchain Interoperability Oracle.” IEEE ICBC 2021.

[26] Nawari, N. O., & Ravindran, S. (2019). ”Blockchain and Building Information Modeling (BIM): Review and Applications in Post-Disaster Recovery.” Buildings, 9(6), 149.

[27] Tkachuk, R.-V., Ilie, D., Robert, R., Kebande, V., & Tutschku, K. (2024). ”On the performance and scalability of consensus mechanisms in privacy-enabled decentralized renewable energy marketplace.” Annals of Telecommunications, 79(3), 271–288.

[28] Trequattrini, R., Palmaccio, M., Turco, M., & Manzari, A. (2024). ”The contribution of blockchain technologies to anti-corruption practices: A systematic literature review.” Business Strategy and the Environment, 33(1), 4–18.

[29] Replicating Master Nodes: Proof-of-Work vs. BFT Replication. iNetSec 2015, Springer LNCS.

[30] Xu, X., et al. (2019). ”The blockchain as a software connector.” IEEE WICSA.

[31] Zheng, Z., et al. (2020). ”An overview on smart contracts: Challenges, advances and platforms.” Future Generation Computer Systems, 105, 475–491.

[32] Comptroller and Auditor General of India. (2023). ”Report of the Comptroller and Auditor General of India for the year 2022–23.” Government of India Press, New Delhi.