Journal of Information Technology, Software Engineering and Computer Science
Website : https://ejournal.techcart-press.com/index.php/itsecs
E-ISSN 2962-0635
P-ISSN 2962-0538
Volume 4, Number 3, July 2026
DOI : **https://doi.org/10.58602/itsecs.v4i3.379**
Page 74-84

Design and Implementation of a Blockchain-Based Digital Gold Trading System for Transaction History Recording

Gaizka Mendieta Bawias[1], Mans Lumiu Mananohas[2], Edwin Tenda[3]
[1,2,3]Information Systems, Sam Ratulangi University, Indonesia
[1]gaizkabawias106@student.unsrat.ac.id, [2]mansmananohas@unsrat.ac.id,
[3]tenda.edwin@unsrat.ac.id

Abstract: Digital-gold platforms require transaction histories that explain each change in rupiah and gold balances. However, when a centralized operator exclusively stores and presents these records, users have limited independently retrievable evidence when the application is unavailable or a completed transaction is disputed. This study designs and evaluates an approval-gated hybrid architecture that separates operational processing from blockchain-based evidence. MySQL manages mutable accounts, balances, prices, workflow states, and internal records, whereas Ethereum Sepolia stores supplementary metadata only for administrator-approved purchase and sale transactions. The prototype was developed using the Waterfall model, React, TypeScript, Node.js, Express.js, MySQL, MetaMask, and a Solidity 0.8.24 smart contract. Black-box evaluation covered 24 application scenarios and nine blockchain-integration scenarios. All 33 predefined scenarios passed (100%): pending and rejected requests created no blockchain record, while approved transactions generated hashes and block numbers consistent with the application ledger and retrievable through Sepolia Etherscan. The study's main contribution is a deliberately bounded hybrid architecture that improves external traceability without treating blockchain as the primary database or claiming full decentralization. The results demonstrate functional feasibility and cross-layer consistency under controlled testnet conditions. They do not establish production security, scalability, mainnet cost efficiency, resistance to adversarial attacks, or the existence and custody of physical gold.
Keywords: Blockchain; Digital Gold; Ethereum Sepolia; Smart Contract; Transaction History; Web-Based Information System.

1. INTRODUCTION

Gold remains a widely used investment asset and store of value because of its durability, scarcity, and broad economic recognition [1]. Digital services extend this role by allowing users to acquire, sell, and monitor fractional gold balances through web or mobile applications. In these services, the displayed gold balance is not an isolated value; it is the cumulative result of recorded purchases, sales, deposits, withdrawals, and price calculations. Consequently, the integrity and availability of transaction history directly affect a user's ability to understand balance changes [2].

The practical urgency of this issue has increased with the expansion of Indonesia's physical digital-gold market. The Commodity Futures Trading Regulatory Agency reported a transaction value of IDR 107.43 trillion and a volume of 55.58 million grams in 2025 [3]. In a digital-gold workflow, a request can be pending, rejected, or approved, but only an approved request should alter the customer's rupiah and gold balances. These states are

Gaizka Mendieta Bawias: *Corresponding Author

Journal of Information Technology, Software Engineering and Computer Science
Website : https://ejournal.techcart-press.com/index.php/itsecs
E-ISSN 2962-0635
P-ISSN 2962-0538
Volume 4, Number 3, July 2026
DOI : **https://doi.org/10.58602/itsecs.v4i3.379**
Page 74-84

ordinarily created, stored, and displayed within the same operator-controlled database. If a completed transaction is disputed or the platform is unavailable, the user cannot independently retrieve evidence that distinguishes the finalized balance-changing event from a request that never passed validation. The unresolved problem is therefore not the database's ability to process transactions, but the absence of an external reference bound to the final status of a high-value transaction.

Centralized databases remain appropriate for accounts, prices, balances, authorization, and low-latency queries, whereas a fully on-chain design would expose more data and introduce network latency and per-transaction cost. The proposed hybrid architecture assigns distinct responsibilities to the two layers. MySQL remains authoritative for mutable operational records and workflow states; only metadata from an administrator-approved purchase or sale is submitted to Ethereum Sepolia, after which its transaction hash and block number are retained with the local history. This allocation preserves efficient application processing while adding a network-anchored reference that can be retrieved independently of the platform [4], [5]. The blockchain record is therefore supplementary evidence, not a mechanism that corrects inaccurate input or guarantees the underlying asset.

Blockchain does not automatically solve every trust problem. Public records can disclose metadata, while data written before validation may still be inaccurate. Privacy-preserving audit-log research shows that immutability must be considered together with confidentiality and public verifiability [6]. A suitable design should therefore minimize the information placed on-chain, keep mutable operational data in an application database, and state precisely what the blockchain evidence proves.

Earlier studies address related but different design objectives. Regueiro et al. [5] implemented a general blockchain-based audit trail to strengthen the integrity and availability of event histories, without modelling digital-gold balances or approval states. Rizkyana [7] proposed a broader gold-based financial information system—covering savings, transfers, claims, appointments, authorization, and validation—on a private proof-of-stake network; it did not evaluate an approval-gated public-testnet evidence link for purchase and sale records. Damisa et al. [8] placed double-auction and settlement logic for energy trading in Ethereum smart contracts, whereas the present system deliberately retains pricing, balances, authorization, and state transitions in MySQL and places only evidence of approved trades on-chain. Consequently, the research gap concerns the design and functional evaluation of a hybrid digital-gold transaction-history architecture that balances operational control with externally retrievable evidence.

This study aims to design, implement, and test a blockchain-supported digital-gold trading prototype for transaction-history recording. Its main contribution is an approval-gated hybrid architecture that defines the responsibility and evidentiary scope of each data layer. First, a request cannot alter balances or generate a blockchain entry before administrative validation. Second, MySQL handles operational processing while Sepolia supplies a supplementary transaction hash and block reference for approved events. Third, explicit black-box scenarios evaluate both the application workflow and the cross-layer integration. The scope is deliberately limited to simulated balances and prices on a testnet; the system is not connected to physical gold, a bank account, a payment gateway, or an official gold-trading provider.

2. RESEARCH METHODOLOGY

2.1. Research Design and Data Collection

The study used applied research with a system-development orientation. The intended output was a working prototype that demonstrates how a blockchain record can

Gaizka Mendieta Bawias: *Corresponding Author

Journal of Information Technology, Software Engineering and Computer Science
Website : https://ejournal.techcart-press.com/index.php/itsecs
E-ISSN 2962-0635
P-ISSN 2962-0538
Volume 4, Number 3, July 2026
DOI : **https://doi.org/10.58602/itsecs.v4i3.379**
Page 74-84

supplement a conventional transaction database. Requirements were derived through a literature review and observation of common user-facing digital-gold workflows, including registration, authentication, price display, purchase, sale, balance display, and transaction history. The observation was not an audit of a named commercial platform; it was used only to identify a plausible sequence for the prototype.

2.2. Development Procedure

Development followed the Waterfall model because the prototype's actors, transaction states, and principal features were defined before implementation. The model organizes work into requirements, design, implementation, integration and testing, and operation and maintenance [9]. Table 1 maps these phases to the activities conducted in this study. Full operation and maintenance were outside the research scope because the system was not deployed in a financial institution.

Table 1. Waterfall phases and research outputs

Phase	Activities in this study	Output
Requirements	Identify actors, balance rules, transaction states, data, and technology constraints.	Functional and non-functional requirements
System design	Model the business process, classes, interaction sequence, database, interface, and blockchain layer.	BPMN, UML models, and architecture
Implementation	Build the web interface, API, MySQL schema, wallet connection, and Solidity contract.	Integrated application prototype
Integration and testing	Exercise application functions and verify approved records through Sepolia evidence.	Black-box and blockchain test results
Operation and maintenance	Limited to local/testnet operation; institutional deployment was not performed.	Research limitation and future-work requirements

2.3. Transaction Workflow and System Architecture

The prototype has two roles. End users register, authenticate, deposit simulated rupiah, submit buy or sell requests, withdraw simulated rupiah, review transaction status, and inspect their ledger. Administrators manage simulated buy and sell prices, inspect users and transactions, approve or reject requests, and inspect the complete ledger. Authentication and role checks restrict end users to their own account and transaction data.

For a purchase request, the backend calculates the gold quantity from the active buy price and checks the user's rupiah balance. For a sale request, it calculates the rupiah value from the active sell price and checks the gold balance. A sufficient balance creates a pending request but does not immediately change either balance. Rejection changes only the request status. Approval requires a MetaMask confirmation, after which the transaction proof is returned to the application; the backend then marks the transaction successful, updates the balances, and stores the operational history and internal ledger.

Gaizka Mendieta Bawias: *Corresponding Author

Journal of Information Technology, Software Engineering and Computer Science
Website : https://ejournal.techcart-press.com/index.php/itsecs
E-ISSN 2962-0635
P-ISSN 2962-0538
Volume 4, Number 3, July 2026
DOI : https://doi.org/10.58602/itsecs.v4i3.379
Page 74-84

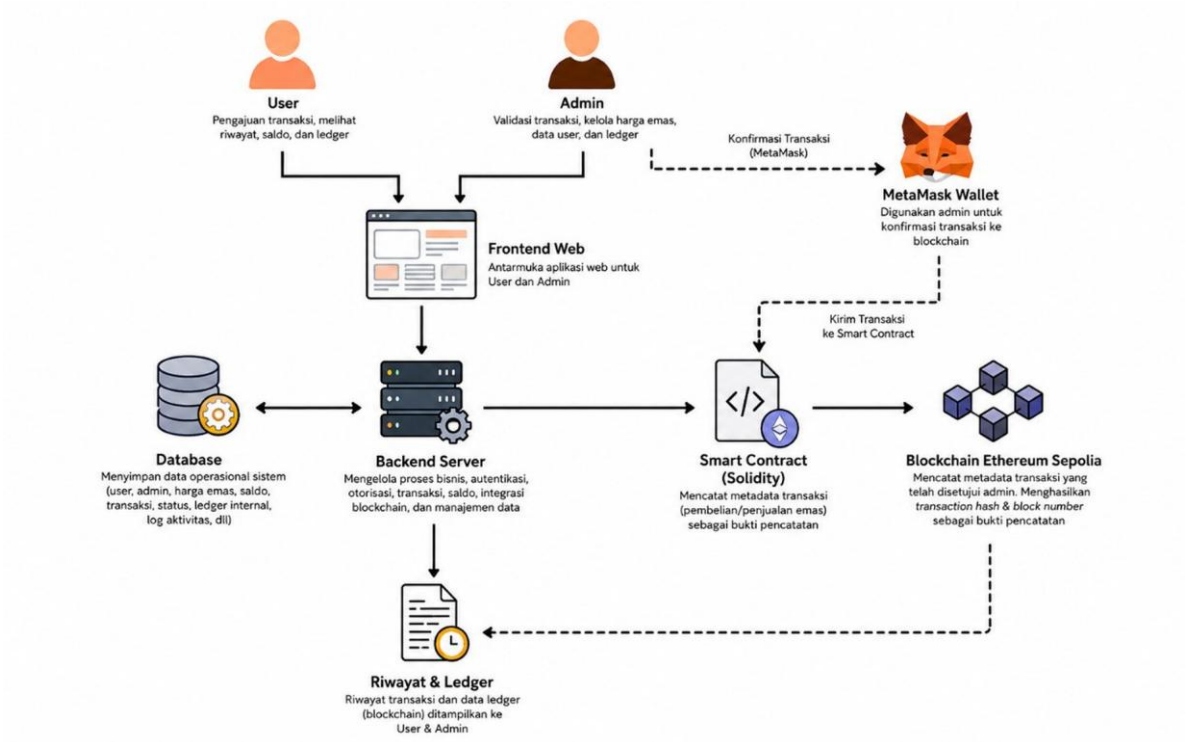

Figure 1. System architecture and separation of operational and blockchain records

Figure 1 shows the hybrid architecture. React, TypeScript, and Tailwind CSS implement the role-based client, while Node.js and Express.js implement the API and business rules. MySQL retains accounts, prices, balances, requests, statuses, and the internal ledger. MetaMask provides wallet access and transaction confirmation. The EmasChainLedger contract executes on Ethereum Sepolia, and the resulting proof is associated with the corresponding operational transaction. This separation avoids using a public blockchain as a general-purpose application database. The component technologies and responsibilities are summarized in Table 2.

Table 2. Implementation technologies and responsibilities

Component	Technology	Responsibility
Web client	React, TypeScript, Tailwind CSS	Role-based interface, forms, status, and ledger presentation
Backend	Node.js, Express.js	Authentication, authorization, pricing, balance rules, requests, and proof validation
Operational store	MySQL	Users, prices, balances, transaction states, and internal ledger
Wallet	MetaMask	Transaction authorization through user confirmation
Smart contract	Solidity 0.8.x	Hash-based transaction recording and duplicate prevention
Blockchain	Ethereum Sepolia	Testnet execution and generation of transaction receipts

Gaizka Mendieta Bawias: *Corresponding Author

Journal of Information Technology, Software Engineering and Computer Science
Website : https://ejournal.techcart-press.com/index.php/itsecs
E-ISSN 2962-0635
P-ISSN 2962-0538
Volume 4, Number 3, July 2026
DOI : **https://doi.org/10.58602/itsecs.v4i3.379**
Page 74-84

2.4. Smart Contract and Blockchain Record

Ethereum Sepolia was selected as a public test network because it supports smart-contract development without requiring mainnet assets [10]. The contract was implemented in Solidity 0.8.24 and deployed for experimental evaluation. Solidity provides the state, event, mapping, and hashing constructs used to record and validate transaction evidence [11].

The EmasChainLedger contract stores a sequential record identifier, a keccak256 hash of the approved transaction payload, the corresponding local transaction identifier, the transaction type, and a blockchain timestamp. A hash-indexed mapping rejects duplicate records, while a contract event indicates that a record has been accepted. Authentication credentials, balances, and other mutable account data remain in MySQL rather than being placed on-chain.

Through MetaMask, the client connects to the Ethereum provider, confirms the selected network, and submits an approved record. After network confirmation, the application retains the transaction hash and block number as references to the blockchain record. Sepolia Etherscan is used to verify transaction status and contract activity [12], [13]. Related DApp studies demonstrate the applicability of MetaMask-based integration and programmable blockchain rules in transaction systems [8], [14].

2.5. Testing Procedure

Black-box testing evaluated externally observable behavior without relying on source-code inspection [15]. Twenty-four application scenarios covered registration, authentication, dashboards, price management, user data, simulated deposits and withdrawals, valid and invalid purchase and sale requests, approval, rejection, history, ledger access, profile display, and logout. Nine blockchain scenarios covered wallet connection, contract invocation, wallet confirmation, approved purchase and sale recording, explorer verification, application-ledger consistency, rejection, and the pending state. A blockchain scenario passed only when its expected on-chain or no-on-chain outcome matched the business rule; a user-interface message alone was not treated as proof of blockchain success.

3. RESULTS AND DISCUSSION

3.1. Prototype Implementation

The implemented prototype supports the complete simulated workflow through separate end-user and administrator interfaces. End users can manage simulated rupiah balances, submit gold purchase or sale requests, and inspect the status of each request. Administrators can manage the active prices, review the submitted transaction data, and approve or reject a request. The implementation therefore preserves a visible distinction between submission, administrative decision, balance update, and blockchain recording.

The administrator dashboard in Figure 2 combines operational indicators with a queue of requests awaiting review. Each queue item presents the transaction type, local identifier, value, and gold quantity before the administrator chooses approval or rejection. Approval initiates the Sepolia confirmation stage, whereas rejection terminates the request without changing the user's balances. The interface consequently implements the approval-gated business rule specified during system design.

Gaizka Mendieta Bawias: *Corresponding Author

Journal of Information Technology, Software Engineering and Computer Science
Website : https://ejournal.techcart-press.com/index.php/itsecs
E-ISSN 2962-0635
P-ISSN 2962-0538
Volume 4, Number 3, July 2026
DOI : **https://doi.org/10.58602/itsecs.v4i3.379**
Page 74-84

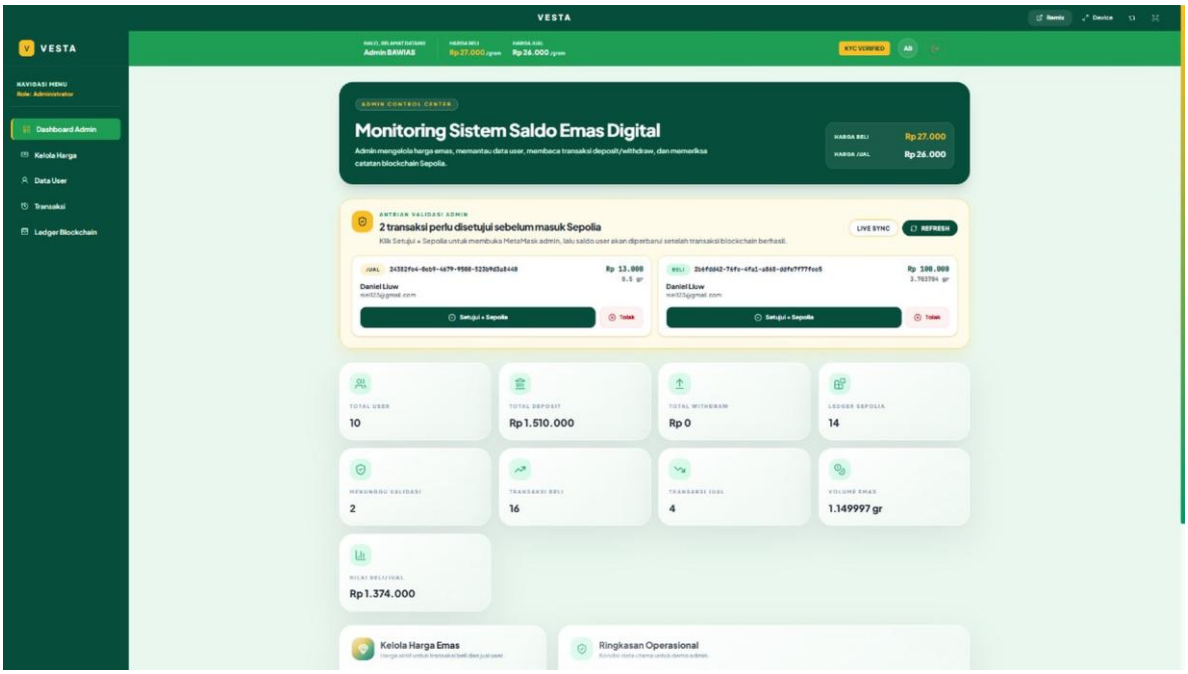

Figure 2. Administrator dashboard and transaction-approval queue

Figure 3 shows the end-user purchase interface. The system calculates the expected gold quantity from the entered rupiah value and the active purchase price, presents the calculation before submission, and labels the resulting request as pending administrative validation. This design allows the user to inspect the economic terms of the request while making clear that the displayed estimate is not yet a completed balance-changing transaction.

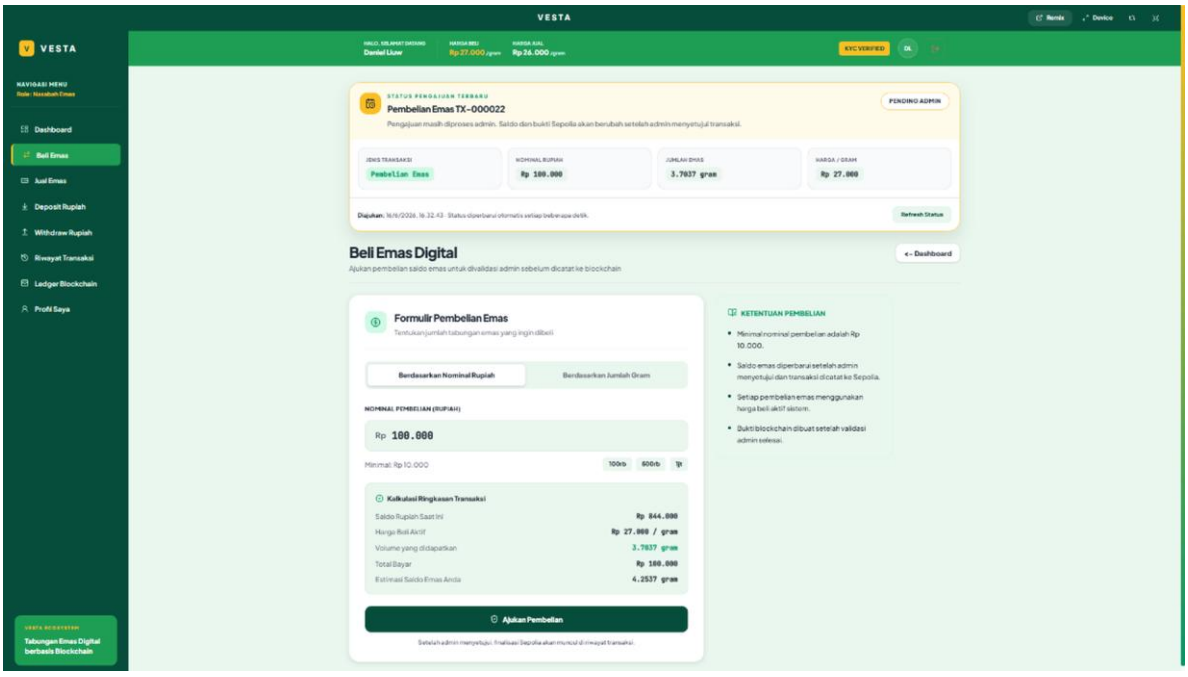

Figure 3. Digital-gold purchase request and pending-validation status

Gaizka Mendieta Bawias: *Corresponding Author

Journal of Information Technology, Software Engineering and Computer Science
Website : https://ejournal.techcart-press.com/index.php/itsecs
E-ISSN 2962-0635
P-ISSN 2962-0538
Volume 4, Number 3, July 2026
DOI : **https://doi.org/10.58602/itsecs.v4i3.379**
Page 74-84

3.2. Transaction-History and Blockchain-Ledger Presentation

A central implementation result is the separation between the operational transaction history and the blockchain-ledger view. The transaction-history page contains every submitted purchase and sale request, including pending, rejected, and approved records. The ledger page contains only transactions that passed administrative validation and completed the Sepolia submission. Figures 4 and 5 provide the corresponding implementation evidence from the original prototype.

As shown in Figure 4, each history row combines the economic data of a request with its workflow state. Pending records explicitly indicate that they have not entered the blockchain, rejected records indicate that no blockchain entry was created, and approved records display a shortened Sepolia transaction reference. This status design prevents the interface from presenting an unapproved request as an immutable transaction record.

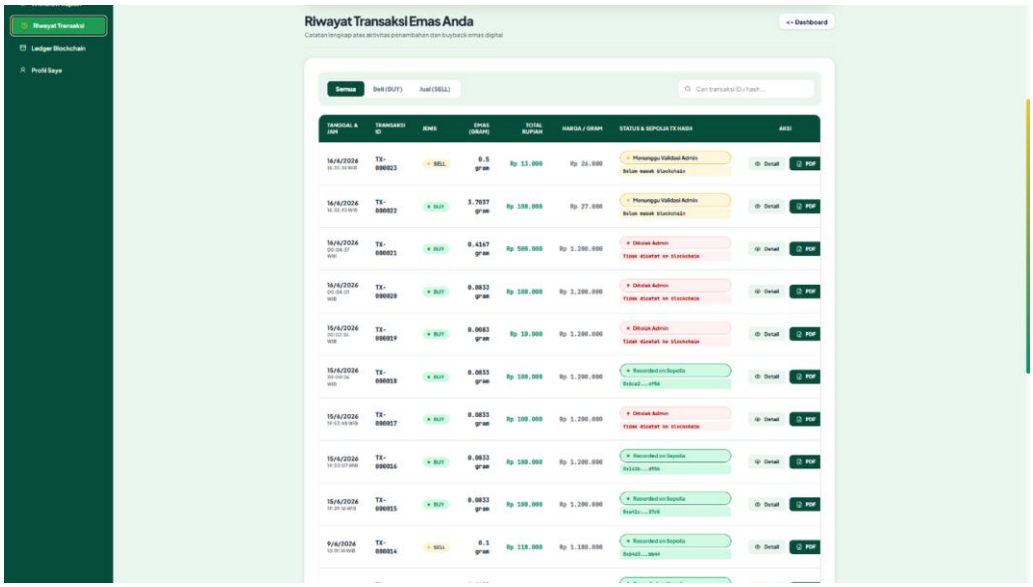

Figure 4. User transaction history with workflow and blockchain-recording status

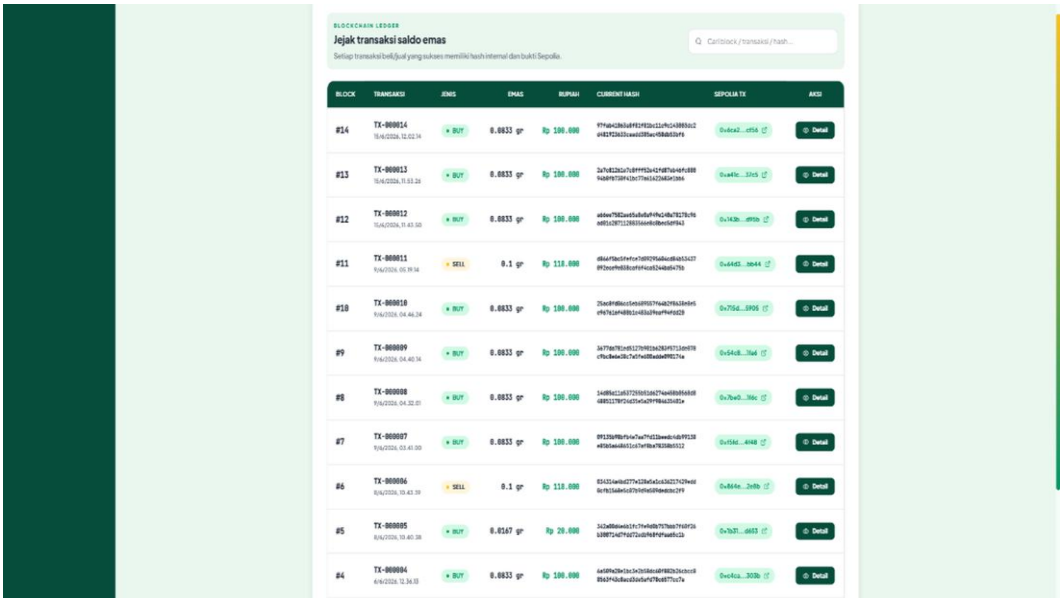

Figure 5. User-facing blockchain ledger for approved transactions

Gaizka Mendieta Bawias: *Corresponding Author

Journal of Information Technology, Software Engineering and Computer Science
Website : https://ejournal.techcart-press.com/index.php/itsecs
E-ISSN 2962-0635
P-ISSN 2962-0538
Volume 4, Number 3, July 2026
DOI : https://doi.org/10.58602/itsecs.v4i3.379
Page 74-84

Figure 5 presents the user-facing ledger after approval. The table links the local transaction identifier and transaction values to an application-level hash and a shortened Sepolia transaction reference. The page itself is a presentation layer rather than independent proof; verifiability depends on whether the associated transaction reference resolves to a successful Sepolia record. The smart contract was compiled and deployed to the testnet, and approved purchase and sale requests produced receipts containing transaction hashes and block numbers. Together, the history and ledger views demonstrate how the prototype connects an operational workflow to externally retrievable testnet evidence without treating the blockchain as the primary application database.

3.3. Functional and Blockchain Test Results

All 24 application scenarios produced the expected outputs. Authentication routed users and administrators to their respective dashboards; access was separated by role; insufficient rupiah or gold prevented a request; valid buy and sell inputs created pending requests; approval updated the correct balances; rejection left balances unchanged; and the history and ledger pages displayed the expected status and evidence. These outcomes confirm conformity with the specified prototype workflow, not general production reliability.

All nine blockchain scenarios also passed. The application established a MetaMask connection to Sepolia, contract interaction triggered a confirmation request, and approved purchase and sale records produced transaction hashes and block numbers. The hashes were locatable on Sepolia Etherscan with successful status, and the application ledger displayed evidence consistent with the blockchain transaction. Conversely, rejected and still-pending requests produced no blockchain transaction, as required.

Table 3. Summary of black-box test outcomes

Test group	Scenarios	Passed	Failed	Pass rate
Application functionality	24	24	0	100%
Blockchain integration	9	9	0	100%
Total	33	33	0	100%

Table 3 shows that 24 of 24 application scenarios (100%) and nine of nine blockchain-integration scenarios (100%) passed, producing an aggregate result of 33 of 33 scenarios (100%). Each percentage was calculated by dividing the number of passed scenarios by the number of executed scenarios and multiplying by 100. Accordingly, the 100% value denotes complete conformity with the predefined expected outputs in the controlled test environment; it is not a statistical estimate of security, availability, or production reliability.

Table 4. Blockchain-integration scenarios

No.	Scenario	Required evidence or behavior	Result
1	MetaMask connection	The application connects to Ethereum Sepolia.	Passed
2	Smart-contract invocation	Approval initiates a MetaMask confirmation request.	Passed
3	Blockchain confirmation	A successful receipt and transaction hash are returned.	Passed
4	Approved purchase	Purchase metadata produces a hash and block number.	Passed

Gaizka Mendieta Bawias: *Corresponding Author

Journal of Information Technology, Software Engineering and Computer Science
Website : https://ejournal.techcart-press.com/index.php/itsecs
E-ISSN 2962-0635
P-ISSN 2962-0538
Volume 4, Number 3, July 2026
DOI : https://doi.org/10.58602/itsecs.v4i3.379
Page 74-84

No.	Scenario	Required evidence or behavior	Result
5	Approved sale	Sale metadata produces a hash and block number.	Passed
6	Explorer verification	The transaction hash is found with successful status.	Passed
7	Ledger consistency	Application hash and block data match the Sepolia evidence.	Passed
8	Rejected request	No blockchain record is created and balances remain unchanged.	Passed
9	Pending request	No blockchain record exists before administrator approval.	Passed

Table 4 further shows that all nine blockchain-integration scenarios (9/9; 100%) met their specified behavior. The passed cases cover network connection, contract invocation and confirmation, approved purchase and sale recording, explorer retrieval, ledger consistency, and the absence of a blockchain entry for rejected or pending requests. This result confirms that the approval rule and Sepolia recording logic were integrated consistently during testing, but it does not evaluate adversarial attacks, concurrent submissions, wallet compromise, mainnet costs, or long-term network availability.

3.4. Discussion

The principal contribution demonstrated by these results is the allocation of responsibility between complementary data layers, rather than the use of blockchain in isolation. MySQL remains the authoritative operational store for identities, prices, balances, requests, and mutable state transitions, which require controlled updates and low-latency retrieval. Sepolia records only approved purchase and sale events and supplies a transaction hash and block number that can be inspected outside the application. In this configuration, blockchain functions as supplementary evidence: it extends the verification path without replacing the database or claiming complete decentralization.

This allocation entails a transparency–privacy trade-off. Publishing approved-event metadata improves transparency because a network record can be retrieved independently. However, public-chain event data are durable and broadly readable. The prototype therefore keeps credentials and balances off-chain, but its event payload remains more detailed than would be advisable in production. A deployed version should pseudonymize identifiers, minimize fields, or store a cryptographic commitment on-chain while retaining detailed records in access-controlled storage [6].

The approval gate entails a decentralization–governance trade-off. Administrative review prevents an unvalidated request from changing balances or generating a blockchain entry, but the administrator remains a centralized decision-maker and potential control point. Blockchain makes the approved event more traceable after submission; it does not verify whether the administrator's decision, the input quantity, the preceding database state, or the claimed physical gold is correct. Production use would therefore require segregation of duties, reconciliation with a regulated custodian, and an auditable authorization policy.

The architecture also trades cost and performance against independent evidence. Restricting blockchain writes to approved events reduces on-chain volume and avoids subjecting routine account and balance queries to block-confirmation latency. Nevertheless, each approved transaction still requires wallet authorization, network

Gaizka Mendieta Bawias: *Corresponding Author

Journal of Information Technology, Software Engineering and Computer Science
Website : https://ejournal.techcart-press.com/index.php/itsecs
E-ISSN 2962-0635
P-ISSN 2962-0538
Volume 4, Number 3, July 2026
DOI : **https://doi.org/10.58602/itsecs.v4i3.379**
Page 74-84

availability, block inclusion time, and gas. Because the study did not measure response time, throughput, confirmation latency, or gas consumption, it cannot determine whether the design is operationally or economically viable on Ethereum mainnet.

The 100% pass rate must consequently be interpreted as functional conformance within the executed scope. It means that all 33 predefined scenarios produced the expected observable outputs under controlled Sepolia test conditions. The tests did not cover unknown inputs beyond the selected cases, concurrent submissions, adversarial attacks, wallet compromise, long-duration operation, recovery, or production workloads. The result therefore supports prototype feasibility and internal workflow consistency, but it does not demonstrate production security, availability, scalability, or reliability.

4. CONCLUSION

This study produced a web-based prototype for buying and selling simulated digital-gold balances with blockchain-supported transaction-history recording. Its main contribution is an approval-gated hybrid architecture: MySQL remains authoritative for operational data and state changes, while Ethereum Sepolia provides supplementary, externally retrievable evidence only for approved purchase and sale events. The design thereby adds an independent verification path without treating blockchain as a replacement for the operational database.

Black-box evaluation showed that all 24 application scenarios (100%) and all nine blockchain-integration scenarios (100%) met their specified outcomes, yielding 33 of 33 passed scenarios (100%). Approved purchase and sale records generated transaction hashes and block numbers that could be inspected through Sepolia Etherscan, while pending and rejected requests produced no blockchain record. These findings demonstrate functional conformance and cross-layer consistency for the predefined test cases, not security or reliability under production conditions.

The conclusion is limited to prototype feasibility. The blockchain record proves inclusion of submitted metadata, not the existence or custody of physical gold, and the 33/33 result does not establish production security or scalability. Future work should integrate a regulated asset source, minimize or pseudonymize on-chain metadata, use an unambiguous versioned contract interface, test concurrent and adversarial conditions, measure gas and latency, and evaluate authorization and reconciliation controls in an operational environment.

5. REFERENCES

[1] K. Alimukhamedov, *Gold Investing Handbook for Asset Managers*. Washington, DC, USA: World Bank, 2024. Accessed: Aug. 10, 2026. [Online]. Available: https://hdl.handle.net/10986/41127

[2] I. A'yun, "Investasi emas digital di Indonesia; tinjauan sistemik maqashid syariah," *JIOSE: Journal of Indonesian Sharia Economics*, vol. 4, no. 2, pp. 203–220, Sep. 2025, doi: 10.35878/jiose.v4i2.1828.

[3] Badan Pengawas Perdagangan Berjangka Komoditi, "Buka perdagangan bursa berjangka komoditi tahun 2026, Wamendag Roro optimistis industri PBK kian bersinar," Kementerian Perdagangan Republik Indonesia, Jan. 2, 2026. Accessed: Aug. 10, 2026. [Online]. Available: https://bappebti.go.id/siaran_pers/detail/15905

[4] R. Soltani, M. Zaman, R. Joshi, and S. Sampalli, "Distributed ledger technologies and their applications: A review," *Applied Sciences*, vol. 12, no. 15, Art. no. 7898, 2022, doi: 10.3390/app12157898.

[5] C. Regueiro, I. Seco, I. Gutiérrez-Agüero, B. Urquizu, and J. Mansell, "A blockchain-based audit trail mechanism: Design and implementation," *Algorithms*, vol. 14, no. 12, Art. no. 341, 2021, doi: 10.3390/a14120341.

Gaizka Mendieta Bawias: *Corresponding Author

Journal of Information Technology, Software Engineering and Computer Science
Website : https://ejournal.techcart-press.com/index.php/itsecs
E-ISSN 2962-0635
P-ISSN 2962-0538
Volume 4, Number 3, July 2026
DOI : **https://doi.org/10.58602/itsecs.v4i3.379**
Page 74-84

[6] M. B. Thazhath, J. Michalak, and T. Hoang, "Harpocrates: Privacy-preserving and immutable audit log for sensitive data operations," in *Proc. 2022 IEEE 4th Int. Conf. Trust, Privacy and Security in Intelligent Systems, and Applications (TPS-ISA)*, Atlanta, GA, USA, 2022, pp. 229–238, doi: 10.1109/TPS-ISA56441.2022.00036.

[7] M. A. Rizkyana, "Rancangan sistem informasi finansial berbasis emas dengan penerapan blockchain," *Prosiding SISFOTEK*, vol. 4, no. 1, pp. 7–11, Aug. 2020. Accessed: Aug. 10, 2026. [Online]. Available: https://seminar.iaii.or.id/index.php/SISFOTEK/article/view/137

[8] U. Damisa, N. I. Nwulu, and P. Siano, "Towards blockchain-based energy trading: A smart contract implementation of energy double auction and spinning reserve trading," *Energies*, vol. 15, no. 11, Art. no. 4084, Jun. 2022, doi: 10.3390/en15114084.

[9] I. Sommerville, *Software Engineering*, 10th ed. Harlow, U.K.: Pearson Education, 2016.

[10] Ethereum Foundation, "Networks," *ethereum.org*. Accessed: Aug. 10, 2026. [Online]. Available: https://ethereum.org/developers/docs/networks/

[11] Solidity Team, "Solidity 0.8.24 documentation," *Solidity Programming Language*. Accessed: Aug. 10, 2026. [Online]. Available: https://docs.soliditylang.org/en/v0.8.24/

[12] Ethereum Foundation, "JSON-RPC API," *ethereum.org*. Accessed: Aug. 10, 2026. [Online]. Available: https://ethereum.org/developers/docs/apis/json-rpc/

[13] Etherscan, "Sepolia Etherscan," Accessed: Aug. 10, 2026. [Online]. Available: https://sepolia.etherscan.io/

[14] B. Kuglics, K. Egres, and M. Sipos, "Blockchain-based healthcare DApp with MetaMask integration," in *Proc. 2024 IEEE 22nd Jubilee Int. Symp. Intelligent Systems and Informatics (SISY)*, Pula, Croatia, 2024, pp. 455–462, doi: 10.1109/SISY62279.2024.10737534.

[15] M. Zen, Irwan, Hafni, and M. D. P. Ananda, "Implementasi dan pengujian menggunakan metode BlackBox Testing pada sistem informasi tracer study," *Bulletin of Computer Science Research*, vol. 4, no. 4, pp. 327–340, Jun. 2024, doi: 10.47065/bulletincsr.v4i4.359.

Gaizka Mendieta Bawias: *Corresponding Author