Blockchain-Based Zero Trust Network Access System

Akshay Agarwal[1], Kola Rahul[2], Samudrala Varshitha[3], Surapaneni Aishwarya[4], Ramesh Babu[5]
[1,2,3,4] Student, Department Of CSE-DS, MREC, Maissamaguda, Hyderabad -500100, India
[5]Assistant Professor, Department Of CSE-DS, MREC, Maissamaguda, Hyderabad -500100, India
doi.org/10.64643/IJIRTV12I11-198978-459

***Abstract*—The rapid expansion of cloud computing, remote access, and distributed digital systems has increased the attack surface of modern enterprise networks. Traditional perimeter- based security models assume that internal entities are trustworthy, resulting in vulnerabilities such as insider threats, credential misuse, lateral movement, and large-scale data breaches. Zero Trust Network Access (ZTNA) addresses these weaknesses by enforcing continuous verification based on identity, device status, and contextual attributes. However, most ZTNA deployments still depend on centralized authentication and policy servers, which create single points of failure and reduce overall reliability.**

To overcome these limitations, the proposed approach integrates decentralized blockchain mechanisms with Zero Trust principles to achieve secure and transparent access control. Smart contracts on the Ethereum blockchain handle authentication, authorization, and immutable activity logging, while AES encryption protects sensitive data stored off-chain. Every access request is verifiable and recorded on a distributed ledger, improving accountability and data integrity. Experimental evaluation indicates strengthened network resilience, removal of centralized trust dependencies, and improved protection against unauthorized access.

Keywords: Zero Trust Network Access, Blockchain Security, Smart Contracts, Access Control

I. INTRODUCTION

The widespread adoption of cloud services, mobile technologies, and remote working environments has fundamentally changed how organizational networks are accessed and managed. As a result, traditional security models that rely on a well-defined network perimeter are no longer effective. These models assume that users and devices within the network can be trusted, which exposes systems to risks such as insider misuse, stolen credentials, and privilege abuse. Once an attacker gains initial access, they can often move laterally across the network with minimal resistance.

To mitigate these risks, the Zero Trust Network Access (ZTNA) model enforces a security philosophy centered on continuous validation rather than implicit trust. Under this approach, every access request is assessed dynamically using factors such as user identity, device condition, and contextual information. Although ZTNA significantly strengthens access control, many real-world implementations depend on centralized authentication and policy enforcement mechanisms. This centralization can introduce operational weaknesses, including reduced availability, limited transparency, and increased susceptibility to targeted attacks.

Blockchain technology presents an effective solution to these challenges by enabling decentralized, tamper-resistant record keeping supported by cryptographic validation. When combined with ZTNA, blockchain allows access decisions and user activities to be securely recorded on a distributed ledger, reducing reliance on centralized authorities. The proposed system integrates blockchain-based smart contracts, encryption techniques, and role-based access controls to establish a secure, auditable, and trustworthy network access framework.

II. LITERATURE SURVEY

The transition from perimeter-based network security to dynamic access models has been a prominent research focus in recent years. The Zero Trust Architecture introduced by Rose et al., in NIST SP 800-207, established the guiding principles of continuous authentication and least-privilege access for all network interactions. Their work emphasized eliminating implicit trust assumptions; however, they also acknowledged that most practical deployments depend heavily on centralized authorization servers, which remain potential failure points. Zhang and Xue explored foundational blockchain security mechanisms, highlighting how decentralization and immutability can replace centralized verification and enable tamper-resistant logging in distributed environments.

Research has increasingly examined the intersection of blockchain and access control. Lin, Wang, and Sadek investigated smart-contract driven authorization systems for IoT, demonstrating that policy enforcement can be automated on-chain with improved auditability, though latency and transaction overhead remain technical challenges. Khan et al. reviewed blockchain-based identity management systems and suggested that integrating blockchain ledgers with Zero Trust—rather than fully replacing traditional identity providers—could improve transparency without disrupting enterprise workflows. Their study also identified privacy concerns when sensitive data is stored directly on the blockchain, recommending hybrid designs that retain encryption and off-chain storage.

Several studies have also focused on smart-contract models for access control. Jannes et al. proposed a decentralized and dynamic policy enforcement framework using smart contracts, achieving resistance to unauthorized changes in security rules. Their findings indicate that blockchain can serve as both a decision verification layer and an immutable audit trail, provided that role-based and attribute-based policies are carefully defined. Complementary work by Yuan and Zhou presented decentralized logging systems where all access operations are recordable on-chain, supporting transparent forensic analysis following security incidents.

From the existing literature, it is evident that Zero Trust provides a strong conceptual foundation for modern access systems, while blockchain contributes distributed trust, auditability, and resilience. However, current research either applies blockchain only for logging or adopts Zero Trust without removing central control dependencies. A fully integrated solution—where authentication, authorization, and activity records are managed through decentralized smart contracts—remains underexplored. This motivates the development of a blockchain-enabled Zero Trust Network Access framework that combines continuous verification, cryptographic encryption, and immutable smart contract enforcement to address single-point-of-failure risks and provide verifiable, tamper-proof access control.

III. PROPOSED METHODOLOGY

In this, we address the limitations of centralized network access models that rely on implicit trust once a user is authenticated. Traditional infrastructures often grant continuous access after login without further verification, exposing systems to unauthorized resource traversal, privilege escalation, data manipulation, and insider misuse. Since authentication records and activity logs are kept in a single database, any compromise or administrative tampering can alter identity information or erase evidence of malicious activity, making it difficult to ensure confidentiality and accountability in distributed or cloud-based environments.

To overcome these issues, we integrate Zero Trust Network Access with blockchain-based identity management, smart contract authorization, and AES encryption to eliminate implicit trust and secure data interactions. Zero Trust enforces continuous validation for every request, blockchain maintains immutable identity and activity records, smart contracts automate authorization decisions without manual intervention, and AES protects file contents even if exposed. The system operates across three stages: identity registration on the blockchain, Zero Trust authentication at each request, and smart contract enforcement of file-level access control, where uploaders assign Public or Private visibility. All permitted and denied actions are recorded on the blockchain to build a tamper-proof audit trail, preventing unauthorized access and ensuring end-to-end accountability.

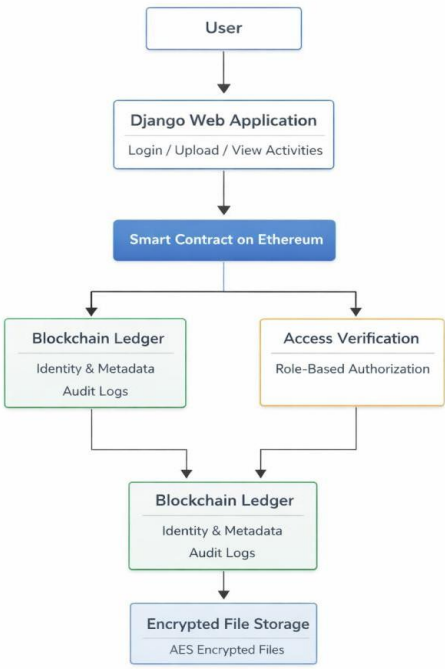

Figure 1. Methodology for Blockchain based ZTNA system

3.1 Blockchain-Based Identity Registration

The first stage establishes user identity and initializes verifiable authentication records on the blockchain. Users register through a Django-based interface by submitting only basic credentials, with no roles or privileges assigned at this stage to align with Zero Trust principles that avoid fixed trust states. Submitted data is processed through a Solidity smart contract, hashed, and stored immutably on the Ethereum blockchain, preventing alteration or deletion and eliminating the risks associated with centralized authentication databases.

During login, credentials are re-hashed and compared with the on-chain reference; successful matches grant access, while mismatches are denied and recorded on-chain for accountability. This cryptographic validation method prevents data tampering, credential spoofing, and administrator-based manipulation, making identity management trust-minimized and forming the foundation for continuous verification in later access stages.

Figure 2. Identity Registration

3.2 Zero Trust Authentication and Request Evaluation

After registration, all user interactions are subjected to Zero Trust authentication, ensuring that no session or user is ever trusted by default. Even after successful login, the system does not grant unrestricted access. Every request— whether upload, view, or download—is treated as an independent event and is validated against identity consistency and session legitimacy. If the request satisfies authentication integrity, it progresses to authorization. Otherwise, the request is rejected, and the failed attempt is written to the blockchain as an immutable log entry. This prevents administrators or malicious users from concealing unauthorized behaviour and establishes a verifiable audit trail. The system therefore enforces authentication at each step rather than granting continuous access after login, eliminating lateral trust movement and enforcing strict request-by-request evaluation.

3.3 Smart Contract Authorization and Role-Based File Access

The final stage manages authorization, encryption, and access enforcement. When a file is uploaded, the system encrypts it using AES before storage, ensuring no plaintext is ever saved. The encrypted file is stored off-chain on the server, while the blockchain records a metadata entry containing the file hash, uploader identity, timestamp, and visibility classification. This metadata is immutable and operates as the authoritative reference for ownership and access rules.

At upload, the user designates the file as Public or Private, which permanently determines its visibility. Public files are available to all authenticated users, whereas Private files are accessible only to the uploader. By assigning permissions to the file rather than the user, the system adopts a file-centric authorization model that aligns with Zero Trust by eliminating reliance on static user roles or assumed trust post- login.

During access requests, the smart contract validates the requester's identity hash against the stored metadata. Public files are released upon successful authentication, while Private files are authorized only if the requester matches the recorded owner. Unauthorized attempts are automatically denied and logged as blockchain transactions, producing an immutable record of rejected activity. All authorization decisions are enforced by the smart contract, preventing privilege escalation, backend manipulation, or administrative override.

This approach ensures that access control is deterministic, tamper-proof, and continuously validated, with every interaction audited on-chain to maintain accountability and enforce Zero Trust principles.

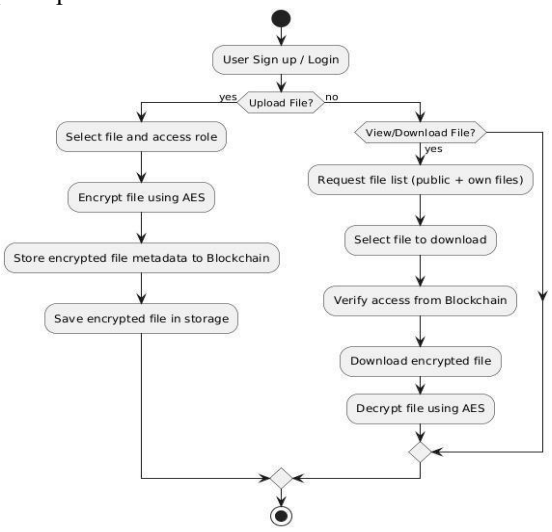

Figure 3. Smart Contract Authorization and Role-Based File Access Workflow

IV.PROPOSED METHODS

The proposed system introduces a set of integrated security methods designed to overcome the limitations of traditional perimeter-based and centralized authentication models. Each method contributes to eliminating implicit trust, preventing privilege persistence, and ensuring that access to resources is determined dynamically and verifiably. The approach combines decentralized identity management on the blockchain, continuous request evaluation based on Zero Trust principles, smart contract–driven authorization, and encrypted off-chain data handling. Together, these methods establish a cryptographically enforced environment in which authentication, authorization, confidentiality, and auditing operate independently of administrators or centralized servers. The subsections that follow outline each method in detail and clarify its role in the overall security model.

4.1 Decentralized Identity Verification Using Blockchain

This method replaces centralized user identity storage with a decentralized verification mechanism executed through a smart contract on the Ethereum blockchain. During registration, user credentials are processed into hashed representations and permanently recorded on-chain, preventing alteration, deletion, or administrative manipulation. The blockchain functions as an immutable identity reference layer, ensuring that authentication depends on cryptographic proof rather than server trust. When a user attempts to log in, the submitted credentials are hashed again and compared against the reference stored in the contract. A match confirms authenticity, while mismatches are rejected without granting access. Because identity validation occurs through on-chain comparison rather than database lookup, the system becomes resistant to credential forgery, SQL injection attacks, data tampering, and insider threats. This method establishes a trust-minimized identity foundation, where authentication integrity is guaranteed through blockchain consensus rather than institutional authority or fixed perimeter controls.

4.2 Zero Trust Request Validation for Every Interaction

Zero Trust Request Validation eliminates the assumption that authenticated users remain trustworthy throughout a session. Instead of granting broad or persistent permissions after logging in, every interaction—such as uploading, viewing, or downloading files—is treated as an untrusted request that must be independently verified. The system evaluates each action against the user's identity hash, session validity, and contextual integrity before allowing it to proceed to authorization. If verification fails at any stage, the request is rejected, and the attempt is immutably recorded on the blockchain for forensic accountability. This continuous verification model prevents privilege persistence, lateral movement, and unauthorized resource traversal, which are common attack vectors in traditional perimeter-based architectures. By enforcing authentication checkpoints at each resource interaction, the system restricts access strictly to intended operations, preventing misuse of valid credentials and reducing the attack surface available to compromised accounts.

4.3 Smart Contract-Driven Access Authorization

Access authorization is executed through a Solidity smart contract that replaces traditional administrator-controlled privilege mechanisms. Instead of relying on backend logic or manual approval, authorization decisions are triggered automatically when users attempt to interact with protected resources. Each file stored in the system is associated with metadata on the blockchain, including its hash value, uploader identity, timestamp, and visibility classification (Public or Private). When a user requests access, the smart contract retrieves the corresponding metadata and evaluates the request by comparing the requester's identity reference to the access rules defined at upload. Public files are accessible to any authenticated user, while Private files are restricted exclusively to the original uploader. Unauthorized attempts— such as users trying to access Private files owned by others— are immediately denied and written as transactions to the blockchain, preventing suppression or modification of the event. This method ensures that authorization is transparent, tamper-proof, and independent of server administrators, providing deterministic access control enforced by code rather than institutional trust or assumed privilege.

4.4 File-Centric Role Assignment Instead of User Roles

This method departs from conventional access control models where predefined roles are permanently assigned to users and used to determine

authorization scope. Instead, the system applies roles at the file level during the upload process, allowing the uploader to classify each file as either Public or Private. The classification determines visibility and accessibility independent of the user's identity or status, ensuring that privileges are tied to the sensitivity of the resource rather than to static trust assumptions. A Public classification allows any authenticated user to view and download the file, while a Private classification restricts access exclusively to the uploader, even if other users share the same platform or organizational context. This eliminates the risks of privilege escalation, unnecessary privilege inheritance, and role leakage across the system. Because access rules are recorded immutably on-chain as part of the file's metadata, neither administrators nor users can modify permissions after the fact, making access control deterministic and tamper-resistant. The result is a resource- centric authorization strategy that aligns with Zero Trust principles by removing reliance on predefined or persistent user roles and enforcing visibility rules directly from the protected object itself.

4.5 AES Encryption with Off-Chain File Storage

The system encrypts each uploaded file using the AES algorithm before storage, ensuring that only ciphertext is stored and that confidential information is never retained in plain text. The encrypted files are saved *off-chain* within the server environment, while the blockchain records only essential metadata such as file hash, uploader identity, timestamp, and visibility classification. Although not labeled as a hybrid model in the original documentation, this structure functions as one by combining cryptographic protection of data at the storage layer with blockchain-based metadata integrity. By separating encrypted content from on- chain metadata, the system reduces blockchain overhead, preserves performance, and ensures confidentiality even if the storage environment is compromised. This approach prevents direct data exposure on the blockchain while retaining verifiable proof of file ownership and access rules.

4.6 Immutable Blockchain-Based Audit Trails

This method leverages the immutable nature of blockchain transactions to create verifiable and tamper- resistant audit trails for all access events within the system. Whenever a user attempts to interact with a stored file— whether uploading, viewing, downloading, or attempting to access a restricted resource—the action triggers a transaction that is recorded on the blockchain through the smart contract. These records include essential information such as the identity reference of the requester, the type of action attempted, and the outcome of the access decision. Because the blockchain ledger cannot be modified or overwritten once a transaction is confirmed, this mechanism prevents administrators or malicious actors from altering system history, erasing traces of unauthorized behavior, or fabricating access events. Even failed or denied attempts remain visible as part of the record, enabling post-incident analysis and reliable forensic investigation.

By decentralizing audit storage and removing reliance on an internal logging system or centralized monitoring server, the method ensures that accountability is preserved independently of organizational control, aligning with Zero Trust principles and providing an objective security history that persists beyond infrastructure boundaries.

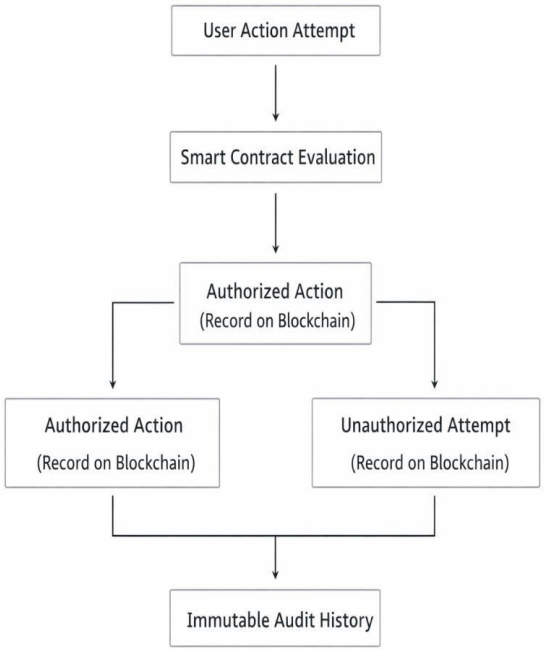

Figure 4. Audit Trail

V.RESULTS

The developed Blockchain-Based Zero Trust Network Access System was executed to validate the end-to-end workflow from user registration to restricted data access, as demonstrated through the application interface. When the project server was initiated and the application homepage was accessed

through the browser, users were able to create new accounts using the New User Sign Up module.

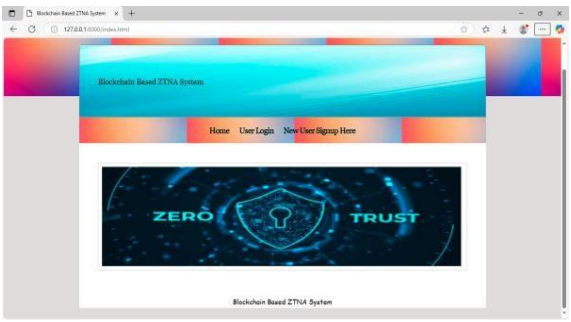

Figure 4. Audit Trail

The registration screen prompted users to enter basic identification details, which were then processed by the backend and converted into a hashed identity reference before being forwarded to the smart contract on the blockchain. Instead of storing the submitted information directly, the system performed hashing operations to ensure that no plaintext credentials were retained within the application or server database. This hashing mechanism served as a protective layer, preventing credential exposure even if the server environment were compromised. Once the hash was generated, the Django backend invoked the deployed Solidity smart contract and transmitted the hash for ledger entry, where it was permanently recorded as the user's identity reference. The blockchain transaction returned parameters such as the block confirmation, transaction hash

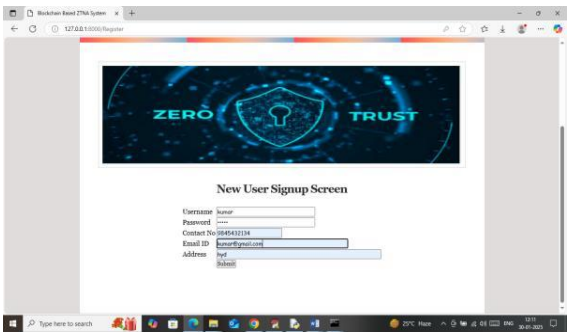

Figure 6. Sign Up Screen

On submission, the registration data was successfully recorded on the blockchain and the system returned transaction details such as block number, transaction hash, and timestamp, confirming that identity storage is handled through decentralized ledger entries rather than a central database. The registration results screen listed blockchain log output, proving that identity hashes had been immutably stored and verified on-chain.

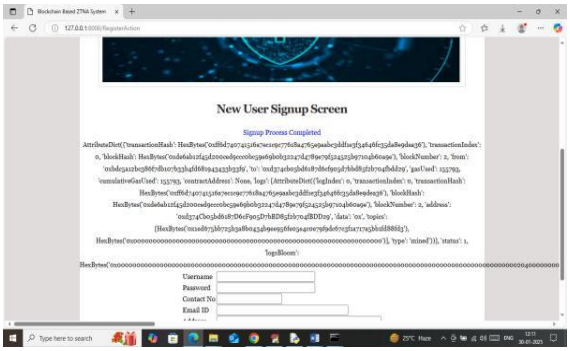

Figure 7. Hash code

After successful registration, the login module authenticated users by validating credentials against blockchain references. Valid inputs led to dashboard access, while incorrect or mismatched credentials were rejected.

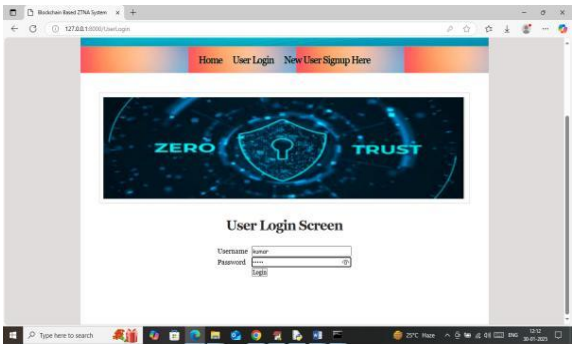

Figure 8. Login Screen

This demonstrated that authentication occurs strictly through hash matching and not through server-side credential comparison, eliminating database dependency for identity validation. Once authenticated, the home dashboard enabled users to upload files through the Save Data To Blockchain module.

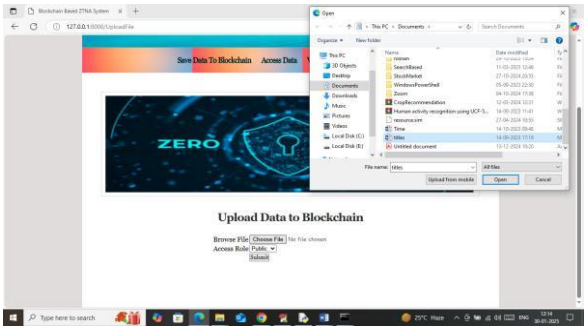

Figure 9. Uploading Data to Blockchain

On upload, the selected file was encrypted and the user specified an access type as Public or Private. After processing, the interface confirmed successful storage by displaying the generated file hash along

with blockchain transaction logs, including block numbers, transaction hashes, and timestamps.

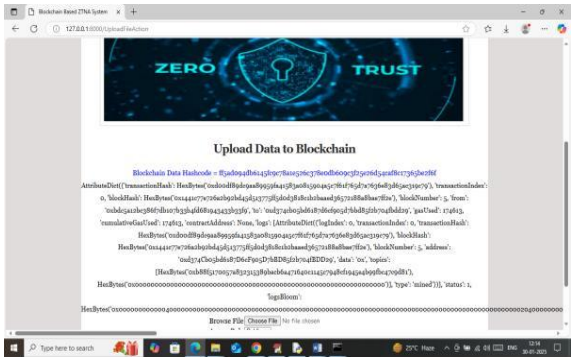

Figure 10. Hash Key Generation after Data Upload

This response verified that the upload operation resulted in two distinct storage outputs: the ciphertext version of the file being written to the server's local storage, and the corresponding metadata being recorded immutably on the blockchain. The metadata included the file hash, owner identity, access classification, and the encryption reference required for retrieval, while the blockchain transaction provided cryptographic proof that the entry could not be modified or removed.

To test authorization behavior, multiple users accessed the system to validate file visibility based on the assigned file category. When logged in as the file owner, the user interface displayed both public and private files along with download options for each, confirming that ownership grants full access rights regardless of the visibility classification. The owner could retrieve decrypted content through the download button, which triggered the AES decryption module only after the smart contract verified the requester's identity hash against the on-chain metadata. This demonstrated that even the uploader's access is not assumed by default but is granted only after successful validation of identity at the blockchain level.

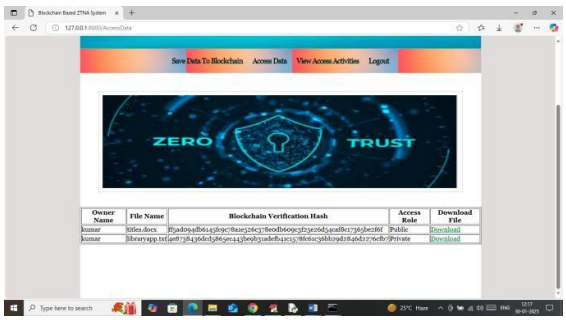

Figure 11. Owner Accessing Data

When logged in as a different user, private files belonging to others were visible but not downloadable, and the system restricted access while still allowing access to public files. The screen displayed Access Restricted for private file requests, confirming that smart contract rules blocked unauthorized downloads without administrative intervention. Attempts to access unauthorized files generated blockchain logs, proving that every request—whether permitted or denied—was recorded as a tamper-proof entry.

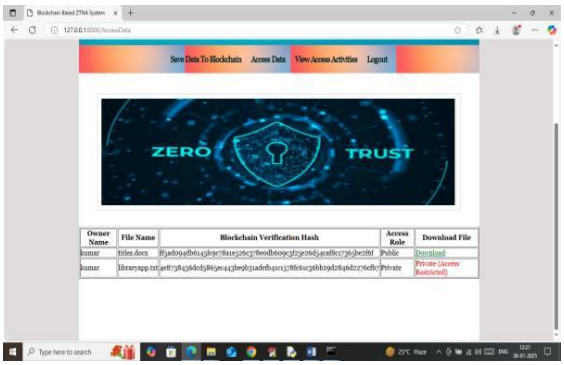

Figure 12. Another User Accessing Data

Additionally, the View Access Activities module listed all access events, including usernames, activity names, dates, and times, serving as immutable audit trails for monitoring and verification.

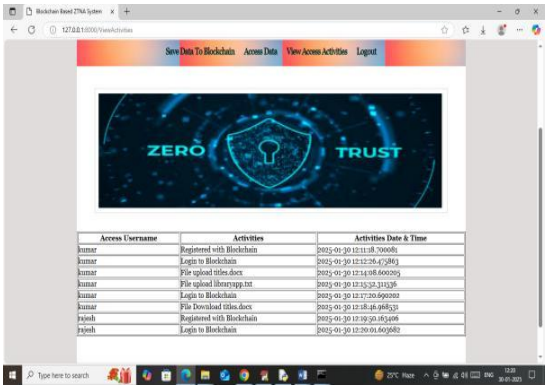

Figure 13. View Access Activities

Overall, the results obtained from the execution screens validate that the system performed as intended. Identity registration and login were independently verified through blockchain hashing, access evaluation occurred at every interaction without persistent trust, files were encrypted prior to storage, access permissions were enforced based on file- centric roles, and all operations generated permanent audit records. The successful execution of these features demonstrates that the integration of

Zero Trust principles with blockchain improves authentication, authorization, confidentiality, and auditability in comparison to traditional centralized architectures.

VI.CONCLUSION

The Blockchain-Based Zero Trust Network Access System was developed to address the limitations of centralized authentication models and implicit trust mechanisms in traditional security architectures. The integration of blockchain identity verification, request-by-request Zero Trust validation, smart contract–based authorization, and AES-encrypted off-chain storage enabled secure and verifiable data handling across the platform. Execution results confirmed that identity records were stored immutably on the blockchain, file access decisions were enforced without administrative intervention, and unauthorized interactions were logged as permanent blockchain entries. The system successfully restricted private file access to the original uploader while allowing controlled access to public files for authenticated users, demonstrating effective file- centric authorization. Overall, the project achieved its objective of creating a trust-minimized access control framework that enhances confidentiality, integrity, and accountability compared to conventional perimeter-based solutions.

REFERENCES

[1] Rose S., Borchert O., Mitchell S., Connelly S., Zero Trust Architecture, NIST SP 800-207, 2020.https://csrc.nist.gov/publications/detail/sp/800- 207/final

[2] Zhang R., Xue R., Liu L., "Security and Privacy on Blockchain," ACM Computing Surveys, 52(3), 2019. DOI: 10.1145/3316481

[3] Khan M.K., et al., "Blockchain-Based Identity Management Systems: A Review," 2020. https://cspecc.utsa.edu

[4] Jannes K., et al., "DEDACS: Decentralized and Dynamic Access Control for Smart Contracts," 2023. https://imec-publications.be

[5] Farukh K, Taha A, Zeshan I, Muneel A.,"IoT Devices User Authentication and Data Management in a Secure Validated Manner through the Blockchain System," 2022.https://www.researchgate.net/publication/3584 55085

[6] Ebtihal A, Suhair A, Asma C.,"Blockchain-Based Access Control for the Internet of Things: A Survey," 2021. https://www.researchgate.net

[7] Gartner, "Market Guide for Zero Trust Network Access (ZTNA)," Gartner Research, 2021.

[8] A. Omar, M. K. Khan, and A. H. Tareq, "Blockchain-Based Identity Management Systems: A Survey," IEEE Access, vol. 8, pp. 1–29, 2020.

[9] J. Donahue, M. Rose, and S. Nagaraj, "Zero Trust Architecture for Cloud Environments," IEEE International Conference on Cloud Engineering, 2021

[10] Y. Nakajima, T. Kitahara, and S. Yoshida, "Policy- Based Access Control for Smart Contracts," Computers & Security, vol. 118, pp. 102–118, 2022.

[11] K. Mahbub, et al., "Zero Trust Architecture: Systematic Literature Review," Journal of Network and Computer Applications, vol. 213, pp. 1–24, 2023.

[12] F. Alharbi, A. Alghamdi, and R. Alsuwaiyel, "A Survey on Blockchain-Based Access Control for IoT," Sensors, vol. 22, no. 18, pp. 1–35, 2022.