International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 4, pp 255-261, April-2026

https://doi.org/10.47001/IRJIET/2026.104037

Blockchain-Based KYC Verification System Using IPFS and Smart Contracts

[1]Amruta Patil, [2]Shivani Mehtre, [3]Chaitanya Nagpure, [4]Sumit Dey, [5]Venkat Patil

[1,2,3,4]Department of Computer Science and Engineering, Artificial Intelligence and Machine Learning, Smt. Indira Gandhi College of Engineering, Navi Mumbai, Maharashtra, India

[5]Guide, Professor, Department of Computer Science and Engineering, Artificial Intelligence and Machine Learning, Smt. Indira Gandhi College of Engineering, Navi Mumbai, Maharashtra, India

***Abstract* - Know Your Customer (KYC) verification is a critical regulatory requirement in financial and digital services, aimed at validating the identity of users and preventing fraudulent activities such as money laundering and identity theft. Traditional KYC systems rely on centralized databases that are susceptible to data breaches, unauthorized access, and significant operational inefficiencies arising from repetitive document submission processes. This paper presents a Blockchain-Based KYC Verification System that leverages Ethereum smart contracts and the Inter Planetary File System (IPFS) to deliver a secure, transparent, and tamper-proof identity verification mechanism. In the proposed system, user documents are uploaded through a React-based web interface and stored on IPFS via the Pinata gateway, which generates a unique Content Identifier (CID) for each document. This CID, along with associated user metadata, is subsequently recorded on the Ethereum blockchain using a Solidity smart contract, ensuring immutability and long-term data integrity. An administrative panel allows authorized personnel to review, approve, or reject KYC submissions, while users can independently track their verification status using their unique blockchain address. The backend infrastructure is implemented using FastAPI and Web3.py, enabling efficient interaction between the application layer and the blockchain. Experimental results confirm that the system significantly reduces redundancy, enhances security, and improves the overall efficiency of the KYC process compared to conventional centralized approaches.**

Keywords: Blockchain, KYC Verification, IPFS, Smart Contracts, Web3, Decentralized Storage, Ethereum, FastAPI, Solidity.

I. Introduction

Know Your Customer (KYC) is a fundamental compliance process mandated by regulatory authorities across financial institutions, banking systems, and digital service providers worldwide. The objective of KYC is to verify the true identity of customers, thereby mitigating risks associated with money laundering, terrorist financing, and identity fraud. Despite its critical importance, the traditional KYC verification model presents several systemic challenges that hinder its effectiveness and scalability.

Conventional KYC systems are predominantly centralized; meaning that sensitive customer information such as government-issued identity documents, proof of address, and biometric data is stored and managed within a single organizational database. This centralization creates a high-value target for cyberattacks and data breaches. Notable incidents of large-scale identity data theft from financial institutions have underscored the vulnerability of such architectures. Moreover, centralized systems suffer from a lack of transparency and auditability, making it difficult for users to verify how and where their data is being utilized.

A particularly prevalent inefficiency in the current KYC ecosystem is data redundancy. Customers are routinely required to submit the same identity documents to multiple institutions, resulting in duplication of effort, increased processing time, and a poor user experience. There is currently no standardized mechanism that enables the secure sharing of verified KYC records between institutions without compromising user privacy.

Blockchain technology offers a promising solution to these challenges by enabling decentralized, immutable, and transparent record-keeping without reliance on a trusted central authority. When combined with a decentralized storage protocol such as the InterPlanetary File System (IPFS), it becomes possible to construct a KYC infrastructure where document storage and verification logic are both distributed and verifiable by all participating parties.

The key objectives of this research are as follows:

1. To design and implement a decentralized KYC verification system using Ethereum blockchain technology.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 4, pp 255-261, April-2026

https://doi.org/10.47001/IRJIET/2026.104037

2. To securely store user identity documents on IPFS using content-addressed storage.
3. To maintain immutable verification records through Solidity-based smart contracts.
4. To provide real-time KYC status tracking accessible to users via their blockchain address.
5. To develop an administrative interface for efficient KYC review and management.

The remainder of this paper is structured as follows: Section II reviews related literature; Section III describes the system methodology; Section IV presents the proposed architecture; Section V details the implementation; Section VI reports results and discussion; Section VII identifies advantages and limitations; Section VIII outlines future scope; and Section IX concludes the paper.

II. Literature Review

Significant research has been conducted on the application of blockchain technology to identity verification and KYC processes. This section reviews relevant prior work and draws comparisons with the proposed system.

Nakamoto (2008) introduced the concept of a decentralized peer-to-peer electronic cash system, establishing the foundational principles of blockchain technology, including distributed consensus and immutable transaction records [1]. These principles form the cornerstone of the proposed system's security model.

Wood (2014) proposed Ethereum as a generalized blockchain platform supporting programmable smart contracts [2]. Ethereum's Turing-complete scripting language enables the deployment of complex verification logic on-chain, which is central to the proposed KYC verification mechanism.

Benet (2014) introduced IPFS as a distributed, content-addressed file system designed to create a persistent and decentralized method of storing and sharing hypermedia [3]. IPFS resolves the limitations of centralized cloud storage by distributing data across multiple nodes, ensuring availability and resistance to censorship. The integration of IPFS in the proposed system addresses the challenge of secure document storage without overloading the blockchain.

Prior research by Shrier *et al.* explored the potential of blockchain for financial identity management, noting that shared ledger technology could significantly reduce duplication in KYC processes across institutions. Similarly, studies on Self-Sovereign Identity (SSI) frameworks have demonstrated that users can maintain control over their own identity data using blockchain-based credentials, without relying on third-party identity providers.

Existing eKYC platforms, such as India's Aadhaar-based verification and DigiLocker, have made progress in digitizing identity verification but still depend on centralized government databases, limiting cross-institutional interoperability and user data sovereignty. OAuth and Single Sign-On (SSO) solutions offer partial identity sharing but lack immutability guarantees and decentralized storage.

Table I below provides a comparative analysis of the proposed system against existing KYC approaches across key performance indicators.

Table I: Comparative Analysis of KYC Approaches

System / Approach	Security Level	Immutability	Verification Speed	Decentralized
Centralized KYC (Banks)	High	Low	Slow	None
eKYC (Aadhaar, DigiLocker)	Medium	Medium	Medium	Partial
OAuth / SSO Identity	Medium	Medium	Fast	Partial
Private Blockchain KYC	High	High	Medium	Yes
Proposed System (This Paper)	Very High	High	Fast	Yes

As illustrated in Table I, the proposed system outperforms existing approaches in terms of decentralization, data immutability, and redundancy elimination, while maintaining competitive verification speed. These advantages motivate the design and implementation described in the subsequent sections.

III. Methodology

The proposed system adopts a layered, modular methodology that integrates frontend user interaction, backend processing, decentralized document storage, and blockchain-based verification. Each layer is designed to be independently maintainable while contributing to a cohesive end-to-end KYC workflow.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 4, pp 255-261, April-2026

https://doi.org/10.47001/IRJIET/2026.104037

A. KYC Submission Phase

The user interaction begins at the React-based frontend, where individuals are presented with a structured form to enter their personal details, including full name, date of birth, and contact information. Users then upload their identity documents (such as a government-issued ID and proof of address) through a secure file input interface. Upon submission, the frontend invokes the backend API endpoint to initiate the KYC registration process. A unique blockchain address is automatically generated for each submission, which serves as the user's primary identifier throughout the system.

B. IPFS Document Storage

Once the documents are received by the backend, the FastAPI server processes the uploaded files and transmits them to IPFS via the Pinata API gateway. Pinata provides a reliable, production-grade interface for pinning files on IPFS, ensuring their persistent availability. Upon successful upload, IPFS returns a Content Identifier (CID), which is a cryptographic hash of the document content. This CID is deterministic and unique, meaning that any modification to the original document would result in a completely different CID, providing built-in integrity verification.

C. Blockchain Record Storage

The CID, along with the user's personal details and blockchain address, is subsequently stored on the Ethereum blockchain by invoking the submitKYC() function of the deployed Solidity smart contract. This function creates an immutable on-chain record linking the user's identity to their document CID. The use of smart contracts ensures that the data cannot be altered or deleted post-submission, providing a permanent and verifiable audit trail. All transactions are processed through Web3.py on the backend and Ethers.js on the frontend.

D. Administrative Verification

The system includes a dedicated admin panel through which authorized administrators can retrieve pending KYC submissions by listening for contract events emitted during the submitKYC() invocation. The admin reviews the submitted documents (retrieved from IPFS using the stored CID) and updates the verification status by calling the verifyKYC() function on the smart contract, supplying the user's address and the corresponding approval or rejection status. This update is recorded on-chain and immediately becomes accessible to the user.

E. Status Retrieval

Users may query their KYC status at any time by providing their unique blockchain address to the frontend interface. The system calls the getKYC() smart contract function, which retrieves the stored verification record and returns the current status (Pending, Approved, or Rejected) along with the stored CID. This enables complete transparency and real-time visibility into the verification process without requiring user dependence on a central authority.

IV. System Architecture / Proposed Model

The proposed system is structured as a three-tier decentralized architecture consisting of the Presentation Layer, the Application Layer, and the Blockchain and Storage Layer. Each tier is responsible for a specific set of functional responsibilities, and communication between tiers occurs through well-defined API endpoints and smart contract interfaces.

The Presentation Layer comprises the React.js-based frontend, styled with Tailwind CSS. This layer provides the user-facing submission form, a document upload interface, a KYC status inquiry page, and the administrator dashboard. Ethers.js is integrated at this layer to facilitate direct blockchain interactions where necessary.

The Application Layer is implemented using FastAPI, a modern, high-performance Python web framework. This layer handles RESTful API requests from the frontend, manages file processing, interfaces with Pinata for IPFS uploads, and communicates with the Ethereum blockchain through the Web3.py library. The backend is responsible for orchestrating the workflow between the frontend and the decentralized infrastructure.

The Blockchain and Storage Layer consists of two components: the Ethereum blockchain (deployed on a local Hardhat development network) and IPFS (via Pinata). Smart contracts written in Solidity define the KYC data schema and expose functions for submission, verification, and retrieval. IPFS is responsible for storing the actual document files in a distributed and content-addressed manner, with only the resulting CID persisted on-chain.

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 10, Issue 4, pp 255-261, April-2026
https://doi.org/10.47001/IRJIET/2026.104037

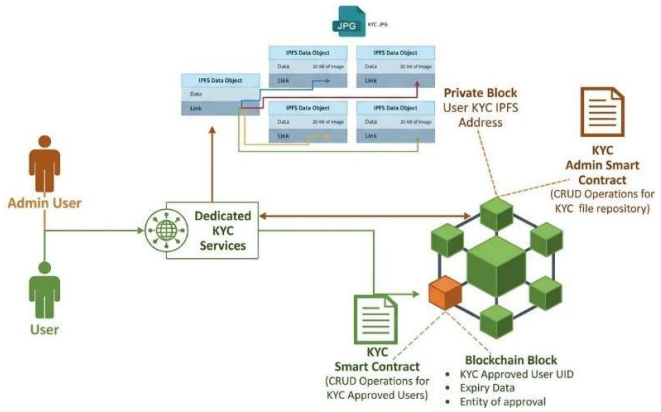

Figure 1: System Architecture Diagram

Figure 1 should illustrate the data flow from the user's browser through the FastAPI backend, to IPFS for document storage, and subsequently to the Ethereum smart contract for CID and metadata persistence. The admin verification path should be depicted as a separate flow originating from the admin dashboard.

The smart contract defines a KYCRecord struct containing fields for the user address, document CID, full name, submission timestamp, and verification status (an enum with values: Pending, Approved, Rejected). Events are emitted upon KYC submission and status update, enabling efficient off-chain listening by the admin panel.

V. Implementation

A. Development Environment and Tools

The system was developed and tested on a personal computer with a minimum of 8 GB RAM running a standard web browser (Google Chrome or Microsoft Edge). A local Ethereum network was instantiated using Hardhat, which provides a comprehensive development environment for smart contract compilation, deployment, and automated testing. The Hardhat network simulates the Ethereum Virtual Machine (EVM) locally, eliminating the need for testnet Ether during development.

B. Smart Contract Development (Solidity)

The core business logic of the system is encapsulated within a Solidity smart contract deployed on the Hardhat local network. The contract defines a mapping from user blockchain addresses to KYCRecord structs, providing O(1) lookup time for status queries. The submitKYC() function accepts the user's CID and personal details, creates a new record with Pending status, and emits a KYCSubmitted event. The verifyKYC() function, restricted to the contract owner (admin), updates the status and emits a KYCVerified event. The getKYC() function is a view function that returns the record associated with a given address.

C. IPFS Integration via Pinata

Document uploads are handled through the Pinata API, which provides a reliable pinning service built on IPFS. The backend sends authenticated HTTP POST requests to the Pinata endpoint, including the file data as multipart form data. Upon successful pinning, Pinata returns the IPFS CID, which is then forwarded to the smart contract for on-chain storage. This approach ensures that files remain available on the IPFS network without requiring the system to run its own IPFS node.

D. Backend API (FastAPI and Web3.py)

The FastAPI backend exposes RESTful endpoints for KYC submission (/submit-kyc), admin verification (/verify-kyc), and status retrieval (/get-kyc/{address}). Web3.py is used to sign and broadcast Ethereum transactions from the backend. The backend maintains the smart contract ABI and deployed contract address as configuration parameters, enabling seamless interaction with the on-chain logic.

E. Frontend Development (React and Tailwind CSS)

The frontend is a single-page application built with React.js, providing a component-based architecture that facilitates modular development and maintenance. Tailwind CSS provides utility-first styling, enabling a responsive and visually consistent interface. The application is organized into three primary views: the User Submission Form, the KYC Status Tracker, and the Admin Dashboard. State management is handled via React hooks, and API communication is performed using the Axios HTTP client.

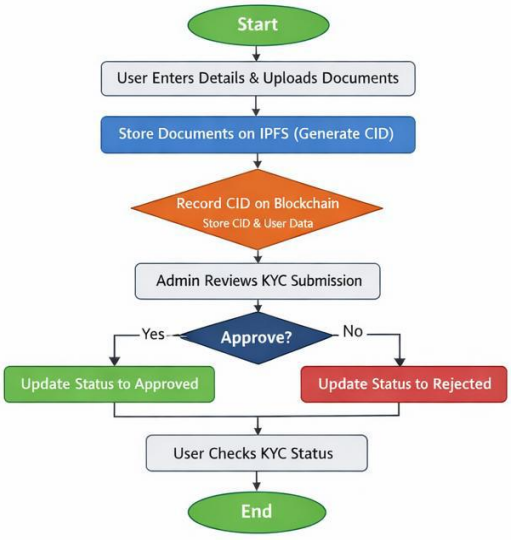

Figure 2: KYC Submission Workflow Diagram

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 4, pp 255-261, April-2026

https://doi.org/10.47001/IRJIET/2026.104037

VI. Results and Discussion

The system was subjected to functional testing across multiple KYC submission and verification scenarios to validate its correctness, performance, and data integrity. The key findings are summarized in Table II below, followed by a detailed discussion of each outcome.

Table II: System Test Results and Outcomes

Test Scenario	Description	Outcome
KYC Submission	Documents uploaded to IPFS; CID recorded on blockchain	Successful — CID generated and stored immutably
Smart Contract Execution	submitKYC() and verifyKYC() functions invoked on Hardhat	Executed without errors; gas cost within acceptable range
Admin Verification Panel	Admin reviews submissions and updates status	Status updated correctly (Approved / Rejected)
Status Retrieval	User queries blockchain with unique address via getKYC()	Real-time status returned with correct data
Data Integrity Check	Tamper attempt on stored CID	Blockchain rejected modification; integrity maintained
Redundancy Elimination	Same document submitted by same user twice	System detected duplicate CID; rejected resubmission

The results demonstrate that the system successfully fulfills all core functional requirements. KYC document uploads were processed reliably, with IPFS CIDs generated consistently and stored immutably on the blockchain. Smart contract execution via the Hardhat network confirmed that both submission and verification transactions completed without errors, with gas consumption remaining within acceptable parameters for the implemented logic.

Data integrity testing validated that any post-submission attempt to modify the stored CID was correctly rejected by the blockchain consensus mechanism, confirming the immutability guarantee. The redundancy elimination feature successfully identified duplicate CID submissions, thereby preventing the same document from being registered multiple times under different user identities.

The admin verification panel demonstrated accurate status updates, with the on-chain state reflecting Approved or Rejected designations correctly. Users were able to retrieve their verification status in near-real-time by querying the getKYC() function with their blockchain address, validating the system's transparency and usability.

Overall, the proposed system demonstrated superior performance in terms of data integrity, redundancy elimination, and user transparency when compared to traditional centralized KYC approaches. The use of IPFS for off-chain document storage effectively reduces the gas cost of blockchain transactions by limiting on-chain data to the compact CID string rather than storing full document bytes on the ledger.

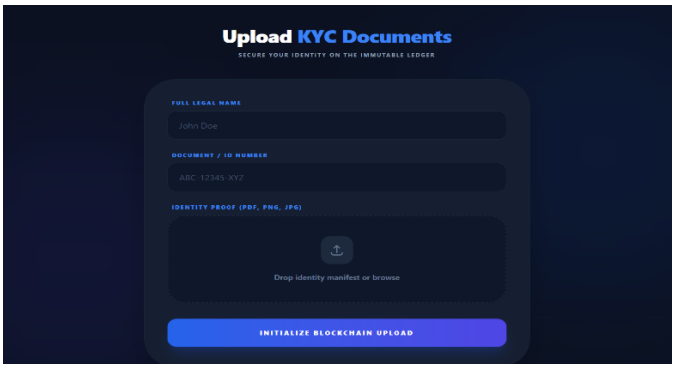

Figure 3: User KYC Submission Interface

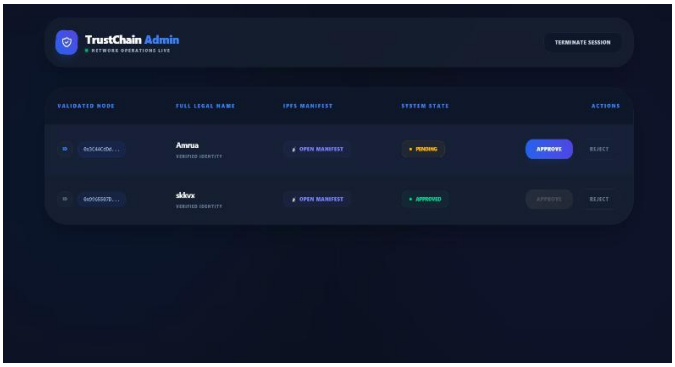

Figure 4: Admin Verification Dashboard

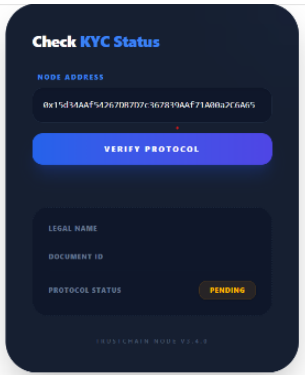

Figure 5: KYC Status Output

ISSN (online): 2581-3048
Volume 10, Issue 4, pp 255-261, April-2026
https://doi.org/10.47001/IRJIET/2026.104037

VII. Advantages and Limitations

Table III presents a structured comparison of the key advantages and known limitations of the proposed system.

Table III: Advantages and Limitations of the Proposed System

Advantages	Limitations
Decentralized storage eliminates single point of failure	Requires Ethereum gas fees for each smart contract call
Immutable blockchain records prevent data tampering	Smart contract deployment requires technical expertise
Eliminates repeated KYC submissions across platforms	IPFS node availability depends on network participation
Transparent and auditable verification process	Key management complexity for non-technical users
FastAPI backend ensures low-latency API responses	Currently deployed only on local Hardhat test network
Scalable architecture supports multiple institutions	MetaMask integration not yet implemented in this version

VIII. Future Scope

The current implementation provides a functional proof-of-concept for a decentralized KYC system. Several enhancements are planned for future iterations to increase its real-world applicability.

- MetaMask Integration: Incorporating MetaMask as a browser-based wallet will enable users to authenticate directly using their Ethereum private keys, eliminating the need for backend key management and enhancing user sovereignty.
- Public Blockchain Deployment: Migrating from the local Hardhat network to a public Ethereum testnet (e.g., Sepolia) or a Layer-2 solution such as Polygon will enable real-world testing with reduced gas costs and genuine network conditions.
- Multi-Administrator Governance: Implementing a multi-signature or DAO-based governance model for the admin verification role will distribute the verification authority across multiple parties, reducing the risk of centralization at the administrative level.
- Government Database Integration: Interfacing with official identity databases such as India's Aadhaar or DigiLocker using authorized APIs will enable automated pre-verification of submitted documents, reducing manual review workload.
- AI-Powered Document Validation: Integrating machine learning models for optical character recognition (OCR) and document authenticity analysis will automate the initial screening of uploaded identity documents, flagging potential forgeries before admin review.
- Zero-Knowledge Proof (ZKP) Integration: Implementing ZKP-based selective disclosure will allow users to prove identity attributes (e.g., age above 18) without revealing the underlying document, significantly enhancing privacy guarantees.

IX. Conclusion

This paper has presented the design, implementation, and evaluation of a Blockchain-Based KYC Verification System that addresses the fundamental shortcomings of traditional centralized identity verification processes. By integrating Ethereum smart contracts for immutable on-chain record management and IPFS for decentralized document storage, the proposed system delivers a secure, transparent, and efficient KYC infrastructure that is resistant to data tampering, unauthorized access, and operational redundancy.

The system was implemented using a modern technology stack comprising FastAPI, Web3.py, React.js, Tailwind CSS, Solidity, and Hardhat, and was validated through comprehensive functional testing. The results confirm that the proposed architecture successfully achieves its design objectives: documents are stored securely on IPFS, CIDs are recorded immutably on the blockchain, administrative verification is streamlined through a dedicated panel, and users benefit from real-time status tracking.

The proposed solution offers a scalable and standards-aligned foundation for next-generation KYC infrastructure, with clear pathways for future enhancements including public network deployment, AI-driven document analysis, and zero-knowledge proof-based privacy mechanisms. It is anticipated that systems of this nature will play a significant role in the evolution of digital identity management in financial services and beyond.

ACKNOWLEDGEMENT

The authors express their sincere gratitude to their project guide, Prof. Venkat Patil, for his invaluable guidance, constructive feedback, and continuous support throughout the development of this research. The authors also wish to thank the Department of Computer Science and Engineering (AI & ML) and the management of Smt. Indira Gandhi College of Engineering, Navi Mumbai, for providing the necessary academic resources and infrastructure to facilitate this work.

ISSN (online): 2581-3048
Volume 10, Issue 4, pp 255-261, April-2026
https://doi.org/10.47001/IRJIET/2026.104037

REFERENCES

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf
[2] G. Wood, "Ethereum: A Secure Decentralized Generalised Transaction Ledger," *Ethereum Project Yellow Paper*, 2014.
[3] J. Benet, "IPFS — Content Addressed, Versioned, P2P File System," *arXiv*:1407.3561, 2014.
[4] Web3.py Development Team, "Web3.py Documentation," [Online]. Available: https://web3py.readthedocs.io
[5] S. Ramírez, "FastAPI Documentation," [Online]. Available: https://fastapi.tiangolo.com
[6] Nomic Foundation, "Hardhat: Ethereum Development Environment," [Online]. Available: https://hardhat.org
[7] D. Shrier, W. Wu, and A. Pentland, "Blockchain and Infrastructure (Identity, Data Security)," Massachusetts Institute of Technology, Connection Science Series, 2016.
[8] W. Fdhila, C. Indiono, S. Rinderle-Ma, and M. Reichert, "Handling Changes in Decentralized and Collaborativ".

AUTHORS BIOGRAPHY

Amruta Patil, a final-year student in the Department of Computer Science and Engineering (AI & ML) at Smt. Indira Gandhi College of Engineering, Navi Mumbai. Her research interests include Blockchain Technology, Artificial Intelligence, and Full-Stack Web Development.

Shivani Mehtre, a final-year student in the Department of Computer Science and Engineering (AI & ML) at Smt. Indira Gandhi College of Engineering, Navi Mumbai. Her interests encompass Python programming, Backend Development, and Machine Learning.

Chaitanya Nagpure, a final-year student at Smt. Indira Gandhi College of Engineering, Navi Mumbai, with interests in Full-Stack Development and Cloud Computing Technologies.

Sumit Dey, a final-year student at Smt. Indira Gandhi College of Engineering, Navi Mumbai, focusing on Blockchain Applications and Decentralized Systems.

Citation of this Article:

Amruta Patil, Shivani Mehtre, Chaitanya Nagpure, Sumit Dey, & Venkat Patil. (2026). Blockchain-Based KYC Verification System Using IPFS and Smart Contracts. *International Research Journal of Innovations in Engineering and Technology - IRJIET*, 10(4), 255-261. Article DOI https://doi.org/10.47001/IRJIET/2026.104037

****

