

Crowdfunding Using Blockchain

MANISH KRISHNA

DEPARTMENT OF COMPUTER APPLICATIONS

NEHRU INSTITUTE OF INFORMATION TECHNOLOGY AND MANAGEMENT

COIMBATORE

MS. INDULEKHA K. V.

ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER APPLICATIONS

NEHRU INSTITUTE OF INFORMATION TECHNOLOGY AND MANAGEMENT

COIMBATORE

ABSTRACT

Conventional crowdfunding platforms rely on centralized intermediaries, resulting in challenges such as high transaction costs, limited financial transparency, and counterparty risks arising from fund mismanagement or delayed disbursements. This paper presents a hybrid blockchain-enabled crowdfunding platform that combines the usability and administrative capabilities of conventional web applications with the security, transparency, and immutability offered by decentralized ledger technology. The proposed architecture integrates a Django-based web application with Ethereum smart contracts through the Web3.py framework, enabling efficient off-chain management while ensuring that all financial transactions are executed and enforced on-chain.

The platform employs a custom ERC-20 token representing the Indian Rupee (INR) to provide standardized transaction values within the experimental environment. A role-based framework consisting of Administrators, Organizations, and Contributors governs the campaign lifecycle. Organizations submit crowdfunding campaigns through the web portal, while administrators perform off-chain verification before authorizing on-chain deployment of the corresponding smart contract. Once deployed, the smart contract functions as an autonomous escrow mechanism, accepting contributions directly from contributors' Web3 wallets and automatically enforcing campaign outcomes. Campaigns that achieve their funding targets within the specified deadline release funds exclusively to the designated beneficiary wallet, whereas unsuccessful campaigns enable contributors to reclaim their deposited tokens through an automated refund mechanism.

To maintain consistency between the blockchain and the web application, an event-driven synchronization middleware continuously monitors blockchain events, records transaction information, and updates campaign states within an off-chain SQLite database. This separation of responsibilities enables efficient storage of user profiles and application metadata while preserving the integrity and transparency of financial operations through blockchain consensus. The proposed hybrid architecture demonstrates an effective integration of decentralized smart contracts with conventional web technologies, providing a secure, transparent, and scalable framework for modern digital crowdfunding while reducing reliance on trusted intermediaries.

INTRODUCTION

Crowdfunding has emerged as an important alternative financing model that enables entrepreneurs, non-profit organizations, and project creators to obtain financial support directly from a large community of contributors through online platforms. By reducing dependence on conventional financial institutions and venture capital funding, crowdfunding has broadened access to capital for innovative projects and social initiatives. Despite its growing adoption, most existing crowdfunding platforms operate under centralized architectures in which a single organization manages campaign approval, stores financial records, and controls the collection and distribution of contributed funds. Such centralized models introduce challenges including limited transaction transparency, platform service charges, reliance on trusted intermediaries, and the risk of fund mismanagement or delayed disbursement.

Blockchain technology offers a decentralized alternative for addressing these limitations through distributed consensus, immutable transaction records, and programmable smart contracts. Ethereum, in particular, enables the implementation of self-executing contracts that automatically enforce predefined fundraising conditions without requiring continuous intervention from a centralized authority. However, fully decentralized crowdfunding systems often encounter practical limitations, including inefficient storage of user data, increased blockchain transaction costs, and reduced flexibility for implementing administrative verification and user management functionalities. These challenges motivate the need for architectures that combine the advantages of conventional web applications with the security guarantees provided by blockchain technology.

This paper presents a hybrid blockchain-enabled crowdfunding platform that integrates a Django-based web application with Ethereum smart contracts through the Web3.py framework. The proposed architecture separates application logic into off-chain and on-chain components. The off-chain layer manages user authentication, organization registration, campaign administration, and application data using a relational database, while the on-chain layer executes all financial transactions through Solidity smart contracts deployed on an Ethereum blockchain. This separation enables efficient management of user information without compromising the integrity and transparency of financial operations.

To provide standardized transaction values within the experimental environment, the platform utilizes a custom ERC-20 token representing the Indian Rupee (INR). The system defines three user roles: Administrators, Organizations, and Contributors. Organizations submit fundraising campaigns through the web application, after which administrators perform off-chain verification before authorizing on-chain deployment of the corresponding smart contract. Once deployed, the smart contract functions as an autonomous escrow mechanism. Contributors transfer ERC-20 tokens directly from their Web3 wallets, while the contract automatically enforces campaign conditions. If the specified funding target is achieved before the campaign deadline, the collected funds are transferred to the organization's designated beneficiary wallet. Otherwise, contributors become eligible to reclaim their deposited tokens through an automated refund mechanism implemented by the smart contract.

To maintain consistency between blockchain transactions and the web application, an event-driven synchronization middleware built using Web3.py continuously monitors blockchain events and updates campaign information within an off-chain SQLite database. This synchronization mechanism enables the frontend application to present near real-time campaign progress while avoiding repeated blockchain queries for application data. Consequently, the proposed architecture combines the computational efficiency of traditional web technologies with the transparency, immutability, and trustless execution provided by blockchain-based smart contracts.

The primary contributions of this work are as follows:

1. The design and implementation of a hybrid crowdfunding architecture that combines off-chain web services with on-chain smart contract execution.
2. A role-based campaign management framework incorporating administrative verification prior to blockchain deployment.
3. An autonomous smart contract implementing escrow-based crowdfunding with automated fund release and contributor refund mechanisms.
4. An event-driven synchronization layer that maintains consistency between blockchain state and the web application database.
5. A prototype implementation demonstrating the feasibility of integrating Ethereum smart contracts with conventional web technologies for secure and transparent digital fundraising.

The remainder of this paper is organized as follows. Section II reviews related work on blockchain-based crowdfunding systems. Section III presents the proposed system architecture and implementation. Section IV discusses the experimental setup and evaluation. Section V presents the results and analysis, and Section VI concludes the paper with future research directions.

LITERATURE REVIEW

The rapid advancement of blockchain technology has significantly influenced the development of decentralized financial applications, including crowdfunding platforms. Conventional crowdfunding platforms have demonstrated the effectiveness of online fundraising by enabling individuals and organizations to raise capital directly from contributors. However, these platforms rely on centralized management, requiring users to trust third-party organizations for campaign approval, fund custody, and financial settlement. This centralized approach introduces concerns related to transparency, intermediary fees, delayed fund distribution, and the potential misuse of contributed funds.

Blockchain technology has emerged as a promising solution to these challenges by providing decentralized transaction processing, immutable record keeping, and programmable smart contracts. Ethereum has become one of the most widely adopted blockchain platforms for decentralized crowdfunding because it supports self-executing smart contracts capable of automating fundraising operations without continuous human intervention. Smart contracts eliminate the need for trusted intermediaries by enforcing predefined contractual conditions through consensus mechanisms, thereby improving transparency and reducing counterparty risk.

Several researchers have proposed blockchain-based crowdfunding systems that utilize Ethereum smart contracts to automate campaign funding and fund distribution. In these systems, contributors transfer cryptocurrency directly to a smart contract, which releases funds only when predefined fundraising conditions are satisfied. Such approaches improve accountability by ensuring that financial transactions are permanently recorded on the blockchain and cannot be altered after confirmation. In addition, automated refund mechanisms have been introduced to protect contributors when fundraising goals are not achieved within specified deadlines.

Despite these advantages, fully decentralized crowdfunding platforms present several practical limitations. Storing application data entirely on-chain results in increased transaction costs and inefficient utilization of blockchain resources. Operations such as user registration, authentication, profile management, and campaign administration require frequent data modifications that are expensive and computationally inefficient when executed through smart contracts alone.

Furthermore, public blockchains are not designed to efficiently manage sensitive user information, making complete on-chain storage unsuitable for many real-world applications.

To address these limitations, recent studies have explored hybrid architectures that combine conventional web technologies with blockchain networks. In hybrid systems, user management and application data are maintained within traditional databases, while critical financial operations are executed through blockchain smart contracts. This architectural separation reduces blockchain storage requirements, lowers operational costs, and improves overall application responsiveness without compromising the integrity of financial transactions. Event-driven synchronization mechanisms have also been proposed to maintain consistency between blockchain events and off-chain databases, enabling web applications to display real-time information while minimizing repeated blockchain queries.

Another important consideration in blockchain-based crowdfunding is the volatility of cryptocurrency values. Since native cryptocurrencies such as Ether experience frequent price fluctuations, several researchers have investigated the use of ERC-20 tokens or stable-value digital assets to provide predictable transaction values during fundraising campaigns. Standardized tokens simplify accounting, facilitate transaction tracking, and improve consistency within controlled deployment environments.

Although existing studies demonstrate the feasibility of blockchain-based crowdfunding, many focus primarily on decentralized financial transactions while providing limited support for administrative verification, campaign lifecycle management, and efficient synchronization between blockchain networks and conventional web applications. In addition, relatively few implementations present an integrated framework that combines role-based administration, automated escrow and refund mechanisms, standardized ERC-20 token transactions, and continuous blockchain event synchronization within a unified hybrid architecture.

To address these research gaps, the proposed work presents a hybrid blockchain-enabled crowdfunding platform that integrates a Django-based web application with Ethereum smart contracts through the Web3.py framework. The architecture separates off-chain application management from on-chain financial execution, employs a custom ERC-20 token for standardized transactions, implements automated escrow and refund functionality using Solidity smart contracts, and incorporates an event-driven synchronization middleware to maintain consistency between blockchain transactions and the application database. This combination provides a practical framework for secure, transparent, and efficient digital crowdfunding while preserving the advantages of both traditional web applications and decentralized blockchain systems.

PROJECT OVERVIEW

The proposed system is a hybrid blockchain-enabled crowdfunding platform designed to combine the operational efficiency of conventional web applications with the security, transparency, and immutability provided by blockchain technology. The platform adopts a dual-layer architecture consisting of an off-chain application layer and an on-chain financial layer. This separation enables efficient management of application data while ensuring that all monetary transactions are executed through decentralized smart contracts.

The off-chain layer is implemented using the Django web framework and is responsible for user authentication, organization registration, campaign management, administrative verification, and data storage. User profiles, campaign metadata, transaction history, and other non-financial information are maintained within an SQLite database to provide

fast data access and minimize blockchain storage requirements. By keeping non-critical information off-chain, the platform reduces transaction costs and improves overall system performance.

The on-chain layer is developed using Solidity smart contracts deployed on an Ethereum blockchain operating within a local Ganache test network. This layer manages all financial operations, including campaign deployment, contribution processing, fund escrow, campaign validation, and automated fund distribution. A custom ERC-20 token representing the Indian Rupee (INR) is utilized as the medium of exchange to provide standardized transaction values throughout the experimental environment.

The platform defines three primary user roles: Administrators, Organizations, and Contributors. Organizations register on the platform and submit crowdfunding campaigns containing details such as funding targets, campaign duration, and beneficiary wallet addresses. Administrators perform off-chain verification of submitted campaigns before authorizing their deployment onto the blockchain. Following deployment, contributors can support campaigns by transferring ERC-20 tokens directly from their Web3 wallets to the corresponding smart contract.

Each crowdfunding campaign is governed by an independent smart contract that functions as an autonomous escrow mechanism. The smart contract continuously evaluates campaign conditions and automatically determines the outcome upon reaching the specified deadline. If the fundraising target is achieved, the accumulated tokens are transferred to the organization's designated beneficiary wallet. If the campaign fails to satisfy its funding objective, contributors become eligible to reclaim their deposited tokens through an automated refund mechanism implemented within the smart contract. This process eliminates the need for centralized fund custody and reduces counterparty risk by enforcing campaign rules programmatically.

Communication between the web application and the blockchain network is established through the Web3.py library. An event-driven synchronization middleware continuously monitors blockchain events, processes transaction logs, and updates the off-chain SQLite database to maintain consistency between blockchain state and application data. This synchronization mechanism enables the frontend to present near real-time campaign information while avoiding repetitive blockchain queries for routine application operations.

The proposed architecture leverages the strengths of both centralized and decentralized technologies by assigning computationally intensive and data-centric operations to the off-chain layer while delegating financial execution and transaction verification to blockchain smart contracts. This hybrid approach improves scalability, reduces operational overhead, and enhances transparency without compromising the security of financial transactions.

MODULE DESCRIPTION

The proposed Hybrid Blockchain-Enabled Crowdfunding Platform is organized into multiple functional modules that collectively provide secure campaign management, decentralized financial execution, and efficient synchronization between off-chain and on-chain components. Each module is designed to perform a specific responsibility while maintaining clear separation of concerns within the overall system architecture.

A. User and Profile Management Module

The User and Profile Management Module manages user registration, authentication, and role-based access control within the platform. This module establishes the connection between conventional web identities and decentralized blockchain accounts by associating each registered user with a unique Web3 wallet address. The system supports three user roles: Administrator, Organization, and Contributor, each with distinct permissions and operational responsibilities. User credentials, profile information, wallet mappings, and historical activity are securely maintained within the off-chain database to enable efficient account management while avoiding unnecessary blockchain storage.

B. Campaign Creation and Validation Module

The Campaign Creation and Validation Module is responsible for the creation and verification of crowdfunding campaigns before they are deployed to the blockchain. Organizations submit campaign details, including funding targets, campaign duration, project descriptions, and beneficiary wallet addresses through the web application. Submitted campaigns are initially stored in an off-chain pending state, where they undergo administrative review to verify compliance with platform requirements. Only campaigns approved by the administrator proceed to blockchain deployment, thereby preventing fraudulent or incomplete campaign proposals from consuming blockchain resources and incurring unnecessary deployment costs.

C. Smart Contract Escrow Module

The Smart Contract Escrow Module constitutes the core decentralized component of the proposed platform. Implemented in Solidity, the smart contract governs all financial operations associated with individual crowdfunding campaigns. Upon deployment, each campaign operates independently as an autonomous escrow contract that receives ERC-20 token contributions directly from contributors' Web3 wallets. The contract continuously evaluates campaign conditions throughout the fundraising period and automatically executes predefined outcomes based on the campaign status. When the specified funding target is achieved before the deadline, the accumulated tokens are transferred to the organization's designated beneficiary wallet. Conversely, if the funding objective is not satisfied, the contract enables contributors to reclaim their deposited tokens through an automated refund mechanism, thereby eliminating dependence on centralized intermediaries for fund management.

D. Blockchain Orchestration and Deployment Module

The Blockchain Orchestration and Deployment Module serves as the communication layer between the Django application and the Ethereum blockchain. Using the Web3.py library, this module converts administrator-approved campaign information into smart contract deployment parameters and submits the corresponding transactions to the blockchain network. After successful deployment, the generated smart contract address and transaction details are recorded within the off-chain database to establish a persistent association between blockchain assets and application records. This module enables seamless interaction between conventional backend services and decentralized smart contracts while abstracting blockchain-specific operations from the application layer.

E. Blockchain Event Synchronization Module

The Blockchain Event Synchronization Module maintains consistency between blockchain transactions and the off-chain application database. It continuously monitors blockchain events generated by deployed smart contracts, including campaign creation, funding transactions, and campaign completion events. The synchronization service processes these event logs through Web3.py and updates the corresponding campaign and donation records stored in the SQLite database. This event-driven architecture enables the web application to present near real-time campaign information without repeatedly querying the blockchain, thereby improving system responsiveness while preserving consistency between decentralized and centralized data sources.

F. Frontend User Interface and Web3 Interaction Module

The Frontend User Interface and Web3 Interaction Module provides the primary interaction layer between users and the crowdfunding platform. Developed using HTML5, CSS3, and JavaScript, the interface enables users to register, manage campaigns, monitor fundraising progress, and perform blockchain transactions through an intuitive web environment. Integration with MetaMask using the Ethers.js library allows contributors and organizations to securely authorize token approvals, campaign funding, and refund transactions without exposing private cryptographic keys to the application. The interface dynamically displays campaign statistics, contribution histories, transaction confirmations, and fundraising progress, providing users with timely feedback regarding blockchain operations.

G. Module Integration

The integration of these functional modules establishes a hybrid architecture that effectively combines conventional web technologies with decentralized blockchain infrastructure. Off-chain modules manage authentication, user profiles, campaign administration, and application data, while on-chain smart contracts securely govern all financial transactions and escrow operations. The synchronization middleware ensures consistency between both layers by continuously monitoring blockchain events and updating application records accordingly. This modular design improves maintainability, scalability, and operational efficiency while preserving the transparency, immutability, and trustless execution characteristics required for secure blockchain-enabled crowdfunding.

SYSTEM ARCHITECTURE

The proposed Hybrid Blockchain-Enabled Crowdfunding Platform adopts a layered architecture that integrates conventional web technologies with decentralized blockchain infrastructure. The architecture separates application management from financial execution by assigning administrative and data-intensive operations to the off-chain layer while delegating all monetary transactions to Ethereum smart contracts. This hybrid design improves application responsiveness, minimizes blockchain storage overhead, and preserves the transparency and immutability of financial operations.

The architecture consists of six major components: the Frontend User Interface, the Django Backend Application, the SQLite Database, the Blockchain Orchestration Layer, the Ethereum Blockchain Layer, and the Blockchain Event Synchronization Layer. These components interact to provide secure campaign management, automated fund escrow, decentralized transaction processing, and real-time synchronization between off-chain and on-chain data.

The Frontend User Interface serves as the primary interaction point for Administrators, Organizations, and Contributors. Developed using HTML5, CSS3, and JavaScript, the interface enables users to register, authenticate, submit crowdfunding campaigns, monitor campaign progress, and perform blockchain transactions. Integration with MetaMask through the Ethers.js library allows users to securely authorize token approvals, campaign funding, and refund requests while ensuring that private cryptographic keys remain under the control of the respective wallet owners.

The Django Backend Application functions as the central application server responsible for business logic and administrative operations. It manages user authentication, role-based authorization, organization registration, campaign creation, administrative verification, and communication with the blockchain network. Before blockchain deployment, submitted campaigns are maintained within an off-chain pending state, allowing administrators to verify campaign details prior to activating the corresponding smart contract. This verification process prevents unauthorized or incomplete campaigns from consuming blockchain resources.

Application data is maintained within an SQLite database that stores user profiles, campaign metadata, administrative records, wallet associations, and historical donation information. The database is used exclusively for non-financial information, thereby avoiding unnecessary blockchain storage while enabling efficient data retrieval for frontend operations. The separation of application data from financial transactions improves scalability and reduces operational costs.

Communication between the backend application and the Ethereum blockchain is established through the Blockchain Orchestration Layer using the Web3.py library. This component translates administrator-approved campaign information into smart contract deployment parameters and submits blockchain transactions to the Ethereum network. Upon successful deployment, blockchain transaction hashes and contract addresses are recorded within the off-chain database, establishing a persistent relationship between blockchain assets and application records.

The Ethereum Blockchain Layer executes all financial operations through Solidity smart contracts deployed on a local Ganache test network. Each crowdfunding campaign is governed by a dedicated smart contract that functions as an autonomous escrow mechanism. Contributors transfer custom ERC-20 tokens directly from their Web3 wallets to the campaign contract, eliminating dependence on centralized financial intermediaries. The smart contract continuously evaluates campaign conditions and automatically determines the campaign outcome after the specified deadline. If the funding target is achieved, the accumulated tokens are transferred to the organization's designated beneficiary wallet. Otherwise, contributors become eligible to reclaim their deposited tokens through the contract's automated refund mechanism.

To maintain consistency between blockchain transactions and application data, the Blockchain Event Synchronization Layer continuously monitors blockchain events generated by deployed smart contracts. Using Web3.py, the synchronization service processes events such as campaign deployment, funding transactions, successful fund releases, and contributor refunds. The decoded event information is subsequently used to update the corresponding records within the SQLite database. This event-driven synchronization mechanism enables the frontend application to display near real-time campaign status while minimizing repetitive blockchain queries.

The proposed architecture combines the strengths of centralized and decentralized computing environments by allocating computationally intensive application management tasks to the off-chain infrastructure while preserving

financial integrity through blockchain consensus and smart contract execution. This separation of responsibilities improves maintainability, enhances scalability, and provides transparent, trustless crowdfunding operations without compromising the efficiency of the web application.

PROBLEM STATEMENT

Crowdfunding has become an increasingly important mechanism for financing entrepreneurial ventures, social initiatives, and charitable projects by enabling direct financial contributions from a large community of supporters. Despite its widespread adoption, the majority of existing crowdfunding platforms operate under centralized architectures in which a single organization is responsible for campaign approval, fund custody, transaction processing, and fund distribution. This dependence on centralized intermediaries introduces several challenges, including limited financial transparency, platform service charges, delayed fund settlements, and the risk of fund mismanagement. Contributors are often required to trust both the platform and campaign creators without having a verifiable mechanism to ensure that fundraising rules are enforced fairly and consistently.

Blockchain technology offers an alternative approach by providing decentralized transaction processing, immutable records, and programmable smart contracts capable of automating financial agreements. However, fully decentralized crowdfunding platforms also present practical limitations. Storing application data entirely on-chain increases transaction costs, consumes blockchain storage resources, and reduces system efficiency for operations such as user authentication, profile management, and campaign administration. Additionally, many existing blockchain-based crowdfunding solutions provide limited support for administrative verification, efficient synchronization between blockchain and application databases, and integration with conventional web technologies required for practical deployment.

These limitations highlight the need for a crowdfunding platform that combines the operational efficiency of traditional web applications with the security and transparency of blockchain technology. The system should provide secure campaign management, transparent financial transactions, automated escrow and refund mechanisms, efficient off-chain data management, and reliable synchronization between blockchain events and application data.

To address these challenges, this research proposes a hybrid blockchain-enabled crowdfunding platform that integrates a Django-based web application with Ethereum smart contracts using the Web3.py framework. The proposed architecture separates application management from financial execution by maintaining user information and campaign metadata off-chain while executing all monetary transactions through Solidity smart contracts. A custom ERC-20 token is employed to standardize transaction values within the experimental environment, and an event-driven synchronization mechanism maintains consistency between blockchain records and the off-chain database. The proposed approach aims to improve transparency, reduce dependence on trusted intermediaries, and provide a secure and efficient framework for decentralized digital crowdfunding.

HARDWARE REQUIREMENTS

The proposed Hybrid Blockchain-Enabled Crowdfunding Platform was developed and evaluated using a standard personal computing environment. Since the platform operates on a local Ethereum test network, no specialized blockchain hardware is required. The minimum hardware configuration required for the development, deployment, and testing of the system is presented below.

Component	Specification
Processor	Intel Core i3/i5 or equivalent multi-core processor
Processor Speed	2.4 GHz or higher
Memory (RAM)	Minimum 4 GB (8 GB recommended)
Storage	500 GB or higher
Network Connectivity	Internet connection for software installation, dependency management, and updates

The proposed platform does not require dedicated blockchain mining hardware or high-performance computing resources because all blockchain operations are executed within a local Ethereum test environment.

SOFTWARE REQUIREMENTS

The implementation of the proposed platform integrates conventional web technologies with blockchain development frameworks. The software environment used during development and testing is summarized below.

Software Component	Technology Used	Purpose
Frontend	HTML5, CSS3, JavaScript (ES6+)	Development of the user interface
Web3 Client Library	Ethers.js (v6)	Frontend interaction with Ethereum smart contracts and MetaMask
Backend Framework	Django (Python)	User authentication, campaign management, business logic, and administrative operations
Programming Language	Python 3.x	Backend application development
Database	SQLite	Off-chain storage of user profiles, campaign metadata, and transaction records
Blockchain Platform	Ethereum Local Test Network (Ganache or Hardhat)	Deployment and execution of smart contracts
Smart Contract Language	Solidity	Development of decentralized crowdfunding smart contracts
Token Standard	ERC-20	Standardized token-based crowdfunding transactions
Blockchain Integration	Web3.py	Communication between the Django backend and Ethereum blockchain
Wallet Application	MetaMask	Secure wallet management and transaction authorization
Development Environment	Visual Studio Code or equivalent IDE	Source code development, testing, and debugging

The software architecture combines an off-chain web application with on-chain smart contract execution. The Django framework manages user authentication, campaign administration, and application data, while Solidity smart contracts deployed on the Ethereum blockchain execute all financial transactions. Web3.py enables backend communication with the blockchain, whereas Ethers.js facilitates secure frontend interactions through MetaMask. This hybrid software stack provides an efficient, secure, and transparent environment for decentralized crowdfunding.

SYSTEM IMPLEMENTATION

The proposed Hybrid Blockchain-Enabled Crowdfunding Platform was implemented using a hybrid architecture that integrates conventional web technologies with blockchain-based smart contracts. The implementation separates application management from financial execution by maintaining user-related information and campaign metadata within an off-chain environment while executing all monetary transactions through Ethereum smart contracts. This architectural separation improves application performance, reduces blockchain storage requirements, and ensures that financial operations remain transparent and tamper-resistant.

The web application was developed using the Django framework, which serves as the backend for user authentication, role-based authorization, campaign management, and administrative operations. The backend provides interfaces for user registration, organization onboarding, campaign creation, campaign verification, and blockchain deployment. SQLite is employed as the off-chain database to store user profiles, wallet mappings, campaign metadata, administrative records, and historical donation information. Since these records do not directly affect financial integrity, maintaining them off-chain significantly reduces blockchain storage overhead and improves data retrieval performance.

The frontend was implemented using HTML5, CSS3, and JavaScript (ES6+) to provide an interactive and responsive user interface. The application enables users to register, authenticate, create crowdfunding campaigns, monitor campaign progress, and manage blockchain transactions through a web browser. Integration with MetaMask is achieved using the Ethers.js library, allowing contributors and organizations to securely sign blockchain transactions without exposing private cryptographic keys to the application. All token approvals, campaign funding transactions, and refund requests are authorized directly through the user's wallet.

The decentralized financial layer was implemented using Solidity smart contracts deployed on an Ethereum local test network running on Ganache. Each approved crowdfunding campaign is deployed as an independent smart contract that functions as an autonomous escrow. The smart contract manages campaign parameters, receives ERC-20 token contributions, validates fundraising conditions, and executes predefined actions upon campaign completion. When the fundraising target is achieved before the specified deadline, the accumulated tokens are transferred to the beneficiary wallet specified during campaign creation. If the campaign fails to achieve its funding objective within the allotted period, contributors become eligible to reclaim their deposited tokens through an automated refund mechanism enforced entirely by the smart contract.

A custom ERC-20 token representing the Indian Rupee (INR) was implemented to provide standardized transaction values within the experimental environment. Contributors interact with the ERC-20 token contract to authorize token transfers before funding campaigns. The crowdfunding smart contracts subsequently receive approved token transfers using the ERC-20 transfer mechanism, ensuring secure and standardized financial transactions throughout the platform.

Communication between the Django backend and the Ethereum blockchain is established through the Web3.py library. Following administrative approval of a campaign, Web3.py converts the campaign parameters into blockchain transactions and deploys the corresponding Solidity smart contract to the Ethereum network. The deployment process returns a unique smart contract address and transaction hash, which are stored within the SQLite database to establish a persistent association between blockchain assets and application records.

To maintain consistency between blockchain operations and the web application, an event-driven synchronization mechanism was implemented using Web3.py. The synchronization service continuously monitors blockchain events generated by deployed smart contracts, including campaign deployment, contributor funding, successful fund release, and refund operations. Whenever a blockchain event is detected, the corresponding transaction information is decoded and synchronized with the SQLite database. This approach enables the frontend application to display near real-time campaign status while minimizing repeated blockchain queries for routine application operations.

The implementation adopts a role-based access control model consisting of three categories of users: Administrators, Organizations, and Contributors. Administrators are responsible for verifying campaign submissions and authorizing blockchain deployment. Organizations create and manage fundraising campaigns, while Contributors participate by funding approved campaigns through ERC-20 token transfers. This structured workflow separates administrative responsibilities from financial execution, ensuring that campaign governance remains efficient while all monetary transactions are governed exclusively by blockchain consensus and smart contract logic.

The completed implementation demonstrates the feasibility of integrating conventional web application frameworks with decentralized blockchain infrastructure. By combining Django, SQLite, Solidity, Web3.py, Ethers.js, MetaMask, and an Ethereum local test network within a unified architecture, the platform provides a secure, transparent, and efficient environment for decentralized crowdfunding while maintaining the advantages of both off-chain and on-chain computing.

FUTURE SCOPE

The proposed hybrid blockchain-enabled crowdfunding platform establishes a foundation for secure and transparent decentralized fundraising. However, several enhancements can be explored to improve its scalability, functionality, and applicability in real-world environments. Future work may include deploying the platform on a public Ethereum-compatible blockchain or a Layer-2 network to evaluate its performance under real-world conditions and to reduce transaction costs associated with blockchain operations.

Additional enhancements may include integrating decentralized identity (DID) frameworks for stronger user verification, incorporating multi-signature wallets to improve fund security, and implementing decentralized governance mechanisms that allow community participation in campaign approval and dispute resolution. Furthermore, the platform can be extended to support multiple cryptocurrencies and stablecoins, enabling broader accessibility and interoperability across different blockchain ecosystems.

From a research perspective, future studies may focus on evaluating the platform's scalability, transaction throughput, gas consumption, and security under varying workloads. Comparative performance analysis with existing centralized and blockchain-based crowdfunding platforms would provide further insight into the effectiveness of the proposed hybrid architecture and its suitability for large-scale digital fundraising applications.

CONCLUSION

This paper presented a hybrid blockchain-enabled crowdfunding platform that integrates conventional web technologies with Ethereum smart contracts to provide a secure, transparent, and efficient digital fundraising framework. The proposed architecture combines a Django-based web application for user management, campaign administration, and off-chain data storage with Solidity smart contracts that execute all financial transactions through an automated escrow mechanism. By utilizing a custom ERC-20 token and an event-driven synchronization layer based on Web3.py, the platform maintains consistency between off-chain application data and on-chain transaction records while reducing blockchain storage overhead.

The implementation demonstrates the feasibility of combining centralized application management with decentralized financial execution to address limitations associated with conventional crowdfunding platforms, including limited transparency, dependence on intermediaries, and counterparty risk. The proposed hybrid architecture provides a practical approach for integrating blockchain technology into crowdfunding applications while maintaining the usability and performance of modern web systems. This work contributes to the growing body of research on blockchain-based financial applications and establishes a foundation for further investigation into scalable, secure, and transparent decentralized crowdfunding platforms.

REFERENCES

- [1] S. Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, 2008.
- [2] G. Wood, Ethereum: A Secure Decentralised Generalised Transaction Ledger, Ethereum Project Yellow Paper, 2014.
- [3] V. Buterin, "A Next-Generation Smart Contract and Decentralized Application Platform," Ethereum White Paper, 2014.
- [4] F. Vogelsteller and V. Buterin, "ERC-20 Token Standard," Ethereum Improvement Proposal (EIP-20), Nov. 2015.
- [5] NIST, Blockchain Technology Overview, National Institute of Standards and Technology, NIST IR 8202, Jul. 2018.
- [6] M. Swan, Blockchain: Blueprint for a New Economy. Sebastopol, CA, USA: O'Reilly Media, 2015.
- [7] C. Cachin, "Architecture of the Hyperledger Blockchain Fabric," in Proceedings of the Workshop on Distributed Cryptocurrencies and Consensus Ledgers, Chicago, IL, USA, 2016.
- [8] A. Antonopoulos and G. Wood, Mastering Ethereum: Building Smart Contracts and DApps. Sebastopol, CA, USA: O'Reilly Media, 2018.

- [9] Django Software Foundation, Django Documentation. Available: <https://docs.djangoproject.com/>
- [10] Ethereum Foundation, Solidity Documentation. Available: <https://docs.soliditylang.org/>
- [11] Ethereum Foundation, Web3.py Documentation. Available: <https://web3py.readthedocs.io/>
- [12] Ethers.js, Ethers.js Documentation. Available: <https://docs.ethers.org/>
- [13] MetaMask, MetaMask Developer Documentation. Available: <https://docs.metamask.io/>
- [14] OpenZeppelin, OpenZeppelin Contracts Documentation. Available: <https://docs.openzeppelin.com/contracts/>
- [15] M. Belleflamme, T. Lambert, and A. Schwienbacher, "Crowdfunding: Tapping the Right Crowd," *Journal of Business Venturing*, vol. 29, no. 5, pp. 585–609, 2014.
- [16] E. Mollick, "The Dynamics of Crowdfunding: An Exploratory Study," *Journal of Business Venturing*, vol. 29, no. 1, pp. 1–16, 2014.
- [17] H. Xu, L. Zhang, and S. Wang, "Blockchain-Based Crowdfunding: A Decentralized Platform for Transparent Fundraising," *IEEE Access*, vol. 8, pp. 124724–124736, 2020.
- [18] K. Christidis and M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.
- [19] M. Casino, T. K. Dasaklis, and C. Patsakis, "A Systematic Literature Review of Blockchain-Based Applications: Current Status, Classification and Open Issues," *Telematics and Informatics*, vol. 36, pp. 55–81, 2019.
- [20] X. Zheng, R. R. Mukkamala, R. Vatrappu, and J. Ordieres-Meré, "Blockchain-Based Personal Data Sharing System Using Smart Contracts," *Journal of Computer and System Sciences*, vol. 103, pp. 112–125, 2019.